



Table 1. Installation and Configuration Guides for SaaS

Guide Links	Summary
VEN Installation Guide	Explains the two methods for installation: using the PCE web console to pair VENs with your hosts or to download the VEN software from the Illumio Support portal and install the software by using the VEN command line interface.
Illumio LW-VEN Installation Guide	Describes how to install and use the Legacy Windows VEN (LW-VEN) with the Illumio Core PCE to enforce security policies.
Illumio Core for Kubernetes and Open-Shift Guide	Provides information about how to use Illumio Core with containerized applications running in clusters orchestrated by Kubernetes, and/or by other similar operators like OpenShift.
NEN Installation and Usage Guide	Describes how to install the Illumio Network Enforcement Node (NEN), to configure Server Load Balancers (SLBs) and switches to work with it, and to use the NEN to secure workloads attached to those network devices.
Flowlink Configuration and Usage Guide	Describes how to install, configure, and use Flowlink, an Illumio standalone application that can be used with the Illumio Core to collect network flow data from different types of network sources, such as switches, routers, F5 load balancers, cloud monitoring tools, and syslog exporters.

Table of Contents

VEN Installation and Upgrade	8
Overview of VEN Installation	8
About This Installation Guide	8
Ways to Install the VEN	9
After VEN Installation	17
Prepare for VEN Installation	20
Details about VEN installation	20
Workflows for VEN Installation	22
Prerequisites for VEN Installation	23
VEN Support for Red Hat 5	28
VEN Proxy Support	32
Configure a VEN-specific Windows Proxy	37
Set Up PCE for VEN Installation	41
VEN Library Setup in the PCE	41
Set up Kerberos Authentication on PCE	51
Prepare Scripts	52
VEN Installation & Upgrade Using VEN Library	56
Pairing Profiles and Scripts	56
VEN Installation Using VEN Library in PCE	64
VEN Installation Troubleshooting	75
VEN Upgrade Using VEN Library in PCE	76
VEN Installation & Upgrade with VEN CTL	82
Windows: Install and Upgrade with CLI and VEN CTL	82
Linux: Install and Upgrade with CLI and VEN CTL	86
AIX: Install and Upgrade with CLI and VEN CTL	98
Solaris: Install and Upgrade with CLI and VEN CTL	104
Deploy an Illumio Endpoint VEN as a private app using In-tune	115
Reference	117
VEN Activate Command Reference	117
VEN Compatibility Check	123
Pairing Script and Package Installation (Linux & Windows)	126
VEN Compatibility with Workloads Hosting Containers	129
Illumio Legacy Windows VEN	134
About the Illumio Legacy Windows VEN	134
LW-VEN Requirements and Limitations	134
Set-up Sequence	135
Requirements	135
Limitations and Caveats	136
Install and Configure the Illumio LW-VEN Service	138
STEP 1: (Recommended) Back Up the Existing Firewall Configuration	139
STEP 2: Create or Find a Pairing Profile with the Appropriate Settings	139

STEP 3: Obtain or Generate a Pairing Key in the PCE Web Console	141
STEP 4: Install, configure, and pair the Illumio Legacy Windows VEN Service on a legacy Windows server	143
STEP 5: Enable Flow Reporting	145
STEP 6: Create Security Policy	147
Manage and Troubleshoot the Illumio LW-VEN	147
About Paring and Activation	147
View the Illumio rules applied to the native firewall	148
Tamper detection	148
Support report	149
Commands	150
Troubleshooting	151
Kubernetes and Openshift	153
Overview of Containers	153
Before You Begin	153
Recommended Skills	153
Introduction to Illumio Segmentation for Containers	153
Concepts	156
Architecture	159
Configure Labels for Namespaces, Pods, and Services	166
Use Container Workload Profiles	166
Container Workload Profile Restriction	175
Using Annotations	176
Deployment with Helm Chart (Kubernetes 3.0.0 and Later)	187
Helm Chart Deployment Overview	188
Host and Cluster Requirements	189
Prepare Your Environment	190
Create a Container Cluster in the PCE	201
Create a Pairing Profile for Your Cluster Nodes	203
Map Kubernetes Node or Workload Labels to Illumio Labels ..	204
Deploy with Helm Chart	207
Re-Label Your Cluster Nodes	223
Generating YAML Manifests for Manual Deployment	224
Deployment for C-VEN Versions 21.5.15 or Earlier	226
Host and Cluster Requirements	226
Prepare Your Environment	228
Create a Container Cluster in the PCE	240
Deploy C-VEs in Your Cluster	242
Re-Label Your Cluster Nodes	249
Configure Security Policies for Containerized Environments	250
IP and FQDN Lists	250
Rules for Kubernetes or OpenShift Clusters	253
Rules and Traffic Considerations with CLAS	257
Rules for Containerized Applications	260
Rules for Persistent Storage	270

Local Policy Convergence Controller	272
Firewall Coexistence on Pods	278
Upgrade and Uninstallation	280
Migrate from Previous C-VEN Versions (21.5.15 or Earlier)	281
Upgrade and Uninstall Helm Chart Deployments	285
Upgrade and Uninstall Non-Helm Chart Deployments	287
Upgrade to CLAS Architecture	290
Reference: General	294
Troubleshooting	295
Troubleshooting CLAS Mode Architecture	305
Known Limitations	308
Kubelink Monitoring and Troubleshooting	310
Aggregating Logs from Kubelink and C-VEN Pods	320
Reference: OpenShift Deployment	324
Prepare OpenShift	324
Deploy Kubelink	327
Implement Kubelink with a Private PKI	333
Install and Pair VENs for Containers	340
Manage OpenShift Namespaces	341
NEN Installation and Usage Guide	349
Introducing the Illumio Network Enforcement Node	349
Overview of the NEN	349
What's New in NEN 2.7.x Releases	351
What's New in NEN 2.6.x Releases	352
NEN Installation and Configuration	358
About NEN Installation and Architecture	358
Install and Activate the NEN	366
Install a New Standalone NEN	370
Upgrade Standalone NEN 2.1.0 to Standalone NEN 2.3.x or Later	374
Configure Non-default Ports and Protocols for NEN Flow Col- lection	379
Generate NEN Reports	382
NEN Integration with Load Balancers	383
Load Balancers and Virtual Servers for the NEN	383
Write SLB Policy	393
NEN Integration with Switches	399
Overview of Switch Integration	399
Supported switches and configurations	403
Configure Switches for the NEN	406
NEN Switch Configuration Using REST API	409
Apply Policy for Switches	416
Enhance network security for Top Of Rack routers using Illu- mio NEN 2.7.0 and Cisco IOS XR	422
Integrate the NEN with the NVIDIA BlueField®-3 DPU featur- ing OVS	423

Integrate the Illumio PCE with IBM i running Precisely Assure Security	426
Flowlink 2.0.0 Configuration and Usage	429
Overview	429
How Flowlink Works	429
Supported Flow Record Formats	430
What's New in Flowlink 2.0.0	431
Scale and Limitations	431
Flowlink Requirements	432
Prerequisites	432
CPU, Memory, and Storage Requirements	432
Configure Flowlink	434
STEP 1: Install the Flowlink RPM	434
STEP 2: Create a Service Account API Key	435
STEP 3: Configure HTTP/HTTPS Proxy (if needed)	436
STEP 4: Configure a Flowlink YAML File	437
STEP 5: Run Flowlink	441
Ingested Flow Types	443
Ingested Flow Examples	448
FIPS Compliance for Flowlink	450
Flowlink Usage	451
Collect Flow Records from F5	451
Troubleshooting	460
Illumio Core PCE CLI Tool Guide 1.4.3	463
Overview of the CLI Tool	463
Before You Begin Using the CLI Tool	463
CLI Tool Versioning	463
CLI Tool and PCE Resource Management	464
The ilo Command	465
HTTP Response Codes and Error Messages	466
Environment Variables	466
Installation and Authentication	468
Prerequisite Checklist	468
CLI Tool Installation Prerequisites	468
Install, Upgrade, and Uninstall the CLI Tool	470
Authenticate with the PCE	472
CLI Tool Commands for Resources	475
View Workload Rules	475
View Report of Workload Services or Processes	476
Use the list Option for Resources	477
List Draft or Active Version of Rulesets	483
Support for Proxy	484
Import and Export Security Policy	487
Upload Vulnerability Data	490
CLI Tool Tutorials	506
How to Import Traffic Flow Summaries	506

How to Create Kerberos-Authenticated Workloads	507
How to Work with Large Datasets	508
How to Upload Vulnerability Data	510
Illumio Core PCE CLI Tool Guide 1.4.2	512
Overview of the CLI Tool	512
About This Guide	512
CLI Tool and PCE Resource Management	514
The ilo Command	515
HTTP Response Codes and Error Messages	516
Environment Variables	516
Installation and Authentication	518
Prerequisite Checklist	518
Installation Prerequisites	518
Install, Upgrade, and Uninstall the CLI Tool	520
Authenticate with the PCE	522
CLI Tool Commands for Resources	525
View Workload Rules	525
View Report of Workload Services or Processes	527
Use the list Option for Resources	527
List Draft or Active Version of Rulesets	533
Import and Export Security Policy	534
Upload Vulnerability Data	537
CLI Tool Tutorials	553
How to Import Traffic Flow Summaries	553
How to Create Kerberos-Authenticated Workloads	554
How to Work with Large Datasets	555
How to Upload Vulnerability Data	557
Error Messages	559
Containers Error Messages	559
Containers	559
NEN, VEN Error Messages	559
NEN	560
VEN	561
PCE Error Messages	561
PCE Error Messages and Recommended Actions	562
Legal Notice	564

VEN Installation and Upgrade

This guide explains the two methods for installation:

1. Use the PCE web console to pair VENs with your hosts or
2. Download the VEN software from the Illumio Support portal and install the software by using the VEN command line interface.

Here are the overview topics.

- [Ways to install the VEN \[9\]](#)
- [VEN Installation Details \[20\]](#)
- [Prerequisites for VEN Installation \[23\]](#)
- [PCE for VEN Installation \[41\]](#)
- [VEN Installation and Upgrade \[56\]](#)
- [VEN Reference \[117\]](#)

Overview of VEN Installation

This section provides an overview on how to install the VEN on your hosts. It explains the two ways to install the VEN. When you've chosen an installation method, you can skip to the content for that method.

About This Installation Guide

Before installing VENs on the hosts in your environment, ensure that you meet the necessary technical background.

How to Use This Guide

This guide explains how to deploy the Virtual Enforcement Node (VEN) on your distributed, on-premises systems.

The guide provides the details to complete the following tasks:

- An explanation of the VEN installation methods: using the VEN Library in the PCE and using the VEN Control Interface (CTL)
- How to install VENs using the VEN Library
- How to install VENs using packaging technology and the workload operating systems' native command line interface
- How to uninstall, upgrade, activate, and deactivate VENs by using the VEN CTL

- How to set up the PCE to install VENs by using the PCE web console and the VEN Library

Before Reading This Guide

Illumio recommends that you be familiar with the following topics before you follow the procedures in this guide:

- Your organization's security goals
- The Illumio Segmentation for Data Centers platform
- General computer system administration of Linux and Windows operating systems, including startup/shutdown, and common processes or services
- Linux/UNIX shell (bash) and Windows command line
- TCP/IP networks, including protocols and well-known ports

Notational Conventions in This Guide

- Newly introduced terminology is italicized. Example: *activation code* (also known as pairing key)
- Command-line examples are monospace. Example: `illumio-ven-ctl --activate`
- Arguments on command lines are monospace italics. Example: `illumio-ven-ctl --activate activation_code`
- In some examples, the output might be shown across several lines but is actually on one single line.
- Command input or output lines not essential to an example are sometimes omitted, as indicated by three periods in a row. Example:

```
...  
some command or command output  
...
```

Ways to Install the VEN

In general, there are two main ways to install VENs. The methods have much in common and achieve the same goal: VEN installation and upgrade.

- Using the VEN Library integrated into the PCE. See [VEN Installation & Upgrade Using VEN Library \[56\]](#).
- Manual VEN installation on individual workloads and endpoints using your own software deployment tools. See [VEN Installation & Upgrade with VEN CTL \[82\]](#).

About VEN Installation Using the VEN Library

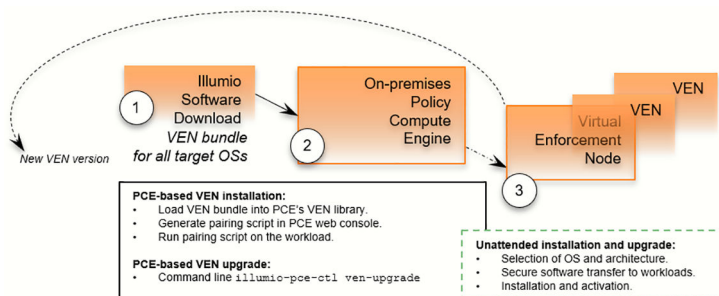


NOTE

For Server VENs, the VEN Library installation and upgrade feature in the PCE is available for the RPM, Debian, and Windows distributions of the VEN software. Other operating systems are not supported for Server VENs.

For Endpoint VENs, the VEN Library installation and upgrade feature in the PCE is available for Windows and macOS versions of the VEN software. Other operating systems are not supported for Endpoint VENs.

Using the VEN Library in the PCE to install the VEN is a more automated approach than installing the VEN CTL but it gives you less control over optional aspects of VEN installation and upgrade.



The VEN Library method of installation utilizes a *VEN software bundle*. The bundle is a collection of a particular VEN software version for all supported workload and endpoint operating systems.

- In the PCE, you load a VEN software bundle into the *VEN library*. The VEN library is a collection of all VEN software versions you have loaded.
- For VEN installation:
 - In the PCE web console, you set a default VEN version.
 - In the PCE web console, you generate a pairing script to install and activate the VEN on target workloads and endpoints.
 - You copy the pairing script to the target workload or endpoint and run it.
 - The pairing script:

- Determines the OS and CPU architecture of the target workload or endpoint.
- Securely transfers the VEN software to the target workloads or endpoints.
- Installs the VEN software.
- Pairs the VEN with its PCE.
- For VEN upgrade, use the VEN Library in the PCE to upgrade all or some of your workloads or endpoints.
- Some features are not available with the VEN Library method, such as Kerberos-based authentication and custom settings with environment variables.



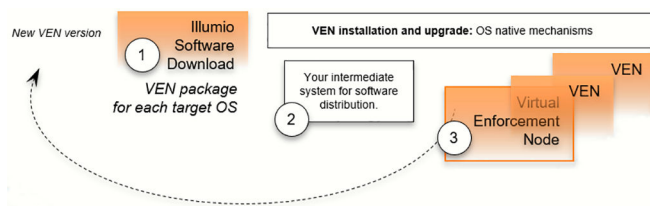
NOTE

Setting up the VEN Library in the PCE is required only for Illumio On-premises customers. If you are an Illumio Cloud customer, Illumio Operations performs this task for you.

About VEN Installation Using the VEN CTL

For installation procedures, see [VEN Installation & Upgrade with VEN CTL \[82\]](#).

This method gives you greater control over optional aspects of VEN installation, pairing, and upgrade.



The VEN installation method using the CTL starts with downloading a *VEN package*. A VEN package includes the VEN software for a single supported workload or endpoint OS and CPU architecture. Installation and upgrade rely on package managers, which are standard native OS tools.

For VEN installation with this method:

- Determine the OS and CPU architecture of the target workloads or endpoints.
- Download the appropriate VEN packages.

For example, installing a Server VEN on CentOS 8 x86-64 requires you to download the VEN package `illumio-ven-XXX.c8.x86_64.rpm`.



NOTE

You are responsible for securely transferring the VEN software to the target workload or endpoint with your own software deployment mechanisms.

- Optionally, set the following environment variables or command-line options:
 - Custom installation directories
 - Custom user and group names
 - Kerberos-based authentication for VEN-to-PCE communications
- Run the native OS installation mechanism.
For example: `rpm -ihv illumio-ven*.rpm`
- Pair the VEN with its PCE.
 - You can pair the VEN during installation or after installation using the VEN CTL activate command (`illumio-ven-ctl activate <options>`)
 - You can use a “prepare script” to install the VEN software on machine images and activate it at the next boot.

If you installed the VEN with the VEN CTL and packaging CLI and customized installation options (such as, a custom installation directory or alternate VEN user), you cannot later upgrade the VEN by using the VEN Library in the PCE. You must upgrade the VEN using the workload or endpoint’s OS package upgrade process.



TIP

If you try to upgrade a VEN using the VEN Library in the PCE but nothing happens, verify whether the VEN was installed by using the VEN CTL.

When to Use Which Method

You can use both methods at different stages of your VEN installation.

Installation Method	Use Cases
VEN Library in the PCE	- To demonstrate the ease of VEN installation and assess installing Linux VENs using the VEN Library - To evaluate and certify new versions of the VEN
VEN CTL	To obtain more control over VEN installation and upgrade with a proprietary software distribution method

VEN-to-PCE Authentication

Illumio Segmentation for Data Centers provides the following mechanisms for authentication between the VEN and the PCE:

- VEN pairing with the PCE
- Kerberos authentication with the PCE

While you can use one or both mechanisms across your organization, note that they are mutually exclusive for the same workload.



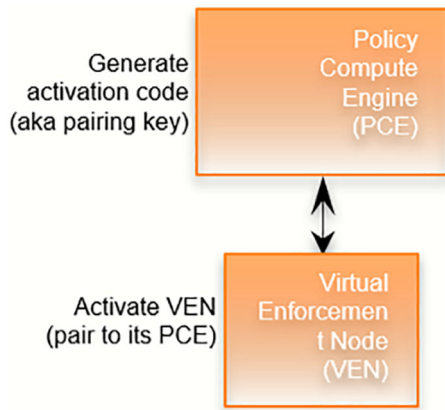
IMPORTANT

This guide assumes that you already have a functional Kerberos service with which to authenticate.

VEN Authentication by Pairing with PCE

This is the default mechanism. When you install a VEN on a workload, the VEN is activated with an activation code generated by the PCE. The activation code is an identifier passed to the VEN software at activation.

After the VEN is activated, it communicates with the PCE over a secure connection. This process of activating a VEN is referred to as pairing it with the PCE. The term *activation* also applies when installing the VEN package directly on a workload by using the VEN CTL.



About the VEN Activation Code

The activation code is an identifier passed to the VEN software at activation. It is obtained from the pairing key. An activation code can be created for one-time use for a single workload or multiple uses for many workloads.

You can get an activation code in the following ways:

- In the PCE web console, create a Pairing Profile. In the profile, you can specify one-time use or unlimited use for the activation code.
- With the REST API. For information, see "Create a Pairing Key" in the REST API Developer Guide.

Activation Details

An activation code is used only after initially installing the VEN. During activation, the PCE generates an agent token. The VEN stores the agent token in a local file on the workload. The PCE stores the hash of the agent token. The VEN uses the agent token to uniquely authenticate itself to the PCE. From that point forward, only the agent token is used in VEN-to-PCE communication.

The VEN communicates with the PCE using HTTPS over Transport Layer Security (TLS) for REST calls and TCP over TLS for the events channel. Additionally, a clone token is generated. If an agent token is mistakenly or maliciously reused on another workload, the clone token is used to detect the condition and disambiguate the hosts. The clone token is periodically rotated. The agent token is never rotated.

**NOTE****Automatic Cloned VEN Remediation**

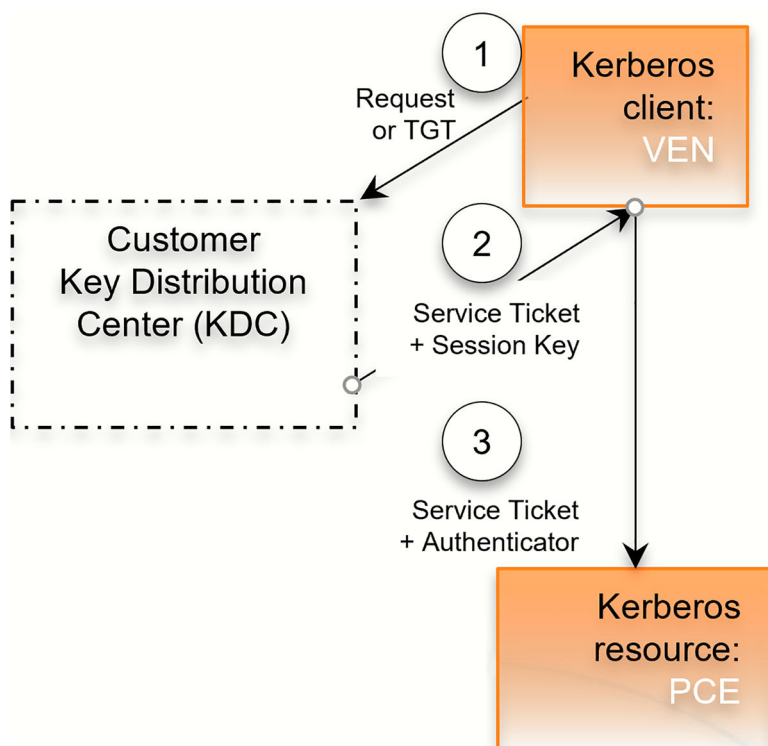
For on-prem domain joined Windows workloads, cloned VENs support automatic clone remediation by detecting changes to the workload's domain Security identifier (SID). After the VEN reports such changes to the PCE, the PCE tells the clone to re-activate itself, after which the cloned VEN is remediated and becomes a distinct agent from the original VEN.

VEN Authentication via Kerberos

You can configure the PCE and VEN to rely on authentication by a pre-configured Kerberos-based system such as Microsoft Active Directory.

**NOTE**

Kerberos-based authentication is supported when you install the VEN by using the VEN CTL. It is not supported when you use the VEN Library in the PCE to install the VEN.



The Key Distribution Center (KDC) is your pre-configured Kerberos service; the VEN is a Kerberos client; and the PCE is a Kerberos resource.

1. The VEN requests a session key or passes its ticket granting ticket (TGT).
2. The KDC returns a service ticket and session key.
3. The VEN passes the authenticated service ticket to the Kerberos-protected PCE.

For information about setting up Kerberos for VEN authentication with the PCE, see [Set up Kerberos Authentication on PCE](#)

For information about pairing workloads via Kerberos for each operating system, see the "Kerberos for Windows VEN-to-PCE Authentication" and "Kerberos for Linux VEN-to-PCE Authentication" topics.

Additionally, you can use the Illumio Segmentation for Data Centers REST API to set up VEN authentication with the PCE via Kerberos. See the "Workload Operations" and "Bulk Traffic Loader" topics in the REST API Developer Guide for information.

VEN-unactivated Golden Masters

When you create machine images for faster deployment of the VEN, consider preparing them to pair the VEN with the PCE the first time the workload is booted. See the [Prepare Golden Master for Workload Installation](#) topic for information.

Upgrading from pre-20.2 to Later Versions

When upgrading from a PCE version pre-20.2 to a later version, stopped VENs that have sent a *goodbye* message to the PCE will have their status value set to *stopped*. During the upgrade, this procedure can cause a burst of events to be emitted to the PCE event stream.

Reduced Banners During VEN Upgrades

The user interface experience on VENs has been enhanced. You will no longer see multiple banners during VEN upgrades. In lieu of an additional banner being displayed, a tally will indicate current upgrade status, show what process is suspended, or display what process is experiencing issues during the upgrade.

MSI to EXE Package Format

Starting with the 21.2.1 Illumio Core release, the Windows VEN installer switched from using the MSI package format to using the EXE package format.

Customers using the PCE-based VEN deployment must take an extra step for the transition. Specifically, Illumio Core customers running older, MSI-based Windows VENs must upgrade to 19.3.6+H1-VEN or 21.2.0+H2-VEN before upgrading their VENs to 21.2.1 or a later version. This release contains the necessary VEN changes to handle the transition in the VEN packaging from MSI to EXE.

After VEN Installation

This section describes some of the basic things you can see immediately after installing the VEN.



WARNING

Any adverse effects of using security scanners or other mechanisms intended to probe or exercise various parts of the VEN or its environment cannot be anticipated by Illumio and may interfere with VEN operation. Therefore, it may be necessary to allow-list some or all of the VEN software components. Consult your security solution vendor for details.

VEN Installation Directories

VEN installation comprises two main parts: the Program Directory and the Program Data Directory. The default location of these directories depends upon the operating system of the workload on which the VEN is installed.

Operating System	Default Program Directory	Default Program Data Directory
Linux, Solaris, AIX	/opt/illumio_ven	/opt/illumio_ven_data
Windows	C:\Program Files\Illumio	C:\ProgramData\Illumio

For details about using non-default installation directories, see the [RPM Only](#) section in *VEN Install and Upgrade > VEN Installation & Upgrade with VEN CTL > Linux: Install and Upgrade with CLI and VEN CTL*.

Program Directory

**CAUTION**

The Program Directory hierarchy contains executable and configuration files that are **not** expected to change except during install/upgrade/uninstall operations. The following table describes some of the more important parts of the Program Directory hierarchy for Linux workloads. This information is incomplete and subject to change without notice. Using any APIs that are not explicitly documented here as Public is unsupported. Precise details will vary by operating system but are generally similar to Linux.

Table 2. Program Directory details

Location relative to Program Directory	API Status	Permissions / Ownership	Contents at Installation
admin	Private	drwxr-x--- root ilo-ven	Helper scripts for VEN administration
bin	Private	drwxr-x--- root ilo-ven	VEN executables and scripts
etc	Private	drwxr-x--- root ilo-ven	Defaults and templates for VEN configuration
illumio-ven-ctl	Public	-rwxr-xr-x root ilo-ven	Illumio VEN control interface script
lib	Private	drwxr-x--- root ilo-ven	VEN libraries and embedded software
runtime_env.yml	Private	rw-r---- root ilo-ven	VEN install-time configuration file
system	Private	drwxr-x--- root ilo-ven	Defaults and templates for OS configuration

Program Data Directory

**CAUTION**

The Program Data Directory hierarchy contains configuration files that **are** expected to change. The following table describes some of the more important parts of the Program Directory hierarchy for Linux workloads. This information is incomplete and subject to change without notice. Using any APIs that are not explicitly documented here as being Public is prohibited and unsupported. Precise details will vary by operating system, but are generally similar to Linux.

Table 3. Program Data Directory details

Location relative to Program Data Directory	API Status	Permissions / Ownership	Contents at Installation
dumps	Private	drwxr-x--- root ilo-ven	Storage for VEN crash dump files
etc	Private	drwxr-x--- root ilo-ven	VEN run-time configuration files
log	Private	drwxr-x--- root ilo-ven	VEN logs and data files
reports	Private	drwxr-x--- ilo-ven ilo-ven	Storage for VEN support reports
tmp	Private	drwxr-x--- ilo-ven ilo-ven	Temporary storage for VEN data files

VEN Runtime User and Group

By default, VEN installation creates a runtime user and group named **ilo-ven** to run non-privileged parts of the VEN software. For security, the **ilo-ven** user is configured without a login shell or home directory.

**CAUTION**

For better security, do not give the **ilo-ven** user a login shell or home directory.

VEN Control Interface and Other Commands

The Illumio VEN control interface script `illumio-ven-ctl` is a command-line tool for performing key tasks for administering the VEN, such as starting and stopping processes, setting run-time configuration, and checking the VEN status.



IMPORTANT

- You must run the Illumio VEN control interface script as root. Do not attempt to use any other commands to administer the VEN.
- Use only the VEN Control Interface script to administer the VEN. Don't attempt to use any other mechanisms such as `initd`, `systemd`, **Solaris SMF**, or **Windows SCM** as these are intended only for automated use by the operating system.

```
sudo /opt/illumio_ven/illumio-ven-ctl ...
```

For details about using the VEN control interface script, see [Linux VEN Activation after Installation](#).

Prepare for VEN Installation

This section provides information that you need to know before installing the VEN software. For a smooth and successful installation of the VEN in your environment, meet the prerequisites outlined in this section.

Details about VEN installation

This topic describes how VEN installation works. Before installing VENs on workloads, review this section.

VEN Pairing and Activation

The term **pairing** applies when using the VEN Library through the PCE web console to install VENs on workloads. The term **activation** applies when installing the VEN package directly on a workload.

A workload pairs or activates with the PCE to become part of the distributed security system using one of these methods:

- By sending a pairing key or activation code
- With a Kerberos service principal name (SPN)
- With a PKI certificate

A pairing key or activation code is used only after initially installing the VEN. During pairing or activation, an agent token is generated and stored in a local file on the workload. The hash of the token is stored on the PCE. The VEN uses the agent token to uniquely authenticate itself to PCE. Only the agent token is used in VEN-to-PCE communication from that point on.

The VEN communicates with the PCE using HTTPS over Transport Layer Security (TLS). Additionally, a clone token is generated. When an agent token is mistakenly or maliciously reused on another workload, the clone token is used to detect the condition and disambiguate the hosts. The clone token is periodically rotated. The agent token is never rotated.



NOTE

Automatic Cloned VEN Remediation

For on-prem domain joined Windows workloads, cloned VENs support automatic clone remediation by detecting changes to the workload's domain Security identifier (SID). After the VEN reports such changes to the PCE, the PCE tells the clone to re-activate itself, after which the cloned VEN is remediated and becomes a distinct agent from the original VEN.

Linux Packages and Kernel Modules

Some packages, such as SecureConnect StrongSwan for enforcing IPsec, are included as part of the VEN package. Other packages are installed on the host itself if they are not already present.

If the following packages are not installed on the workload via RPM dependencies the VEN installation downloads and installs them.

- **curl**: Used for HTTPS client functionality

- **uuid-runtime**: Used for generating UUIDs
- **ipset**: Used for ipset functionality
- **libnfnetlink0**: Used for communicating with the Net Filter module
- **libcap2**: Used for selectively enabling/disabling capabilities
- **libgmp10**: Used for multi-precision arithmetic
- **bind-utils**: Used for DNS client functionality
- **iptables** and **iptables-ipv6**: Used for iptables functionality
- **apt-transport-https** (for apt-based OS): Used for HTTPS transport for apt

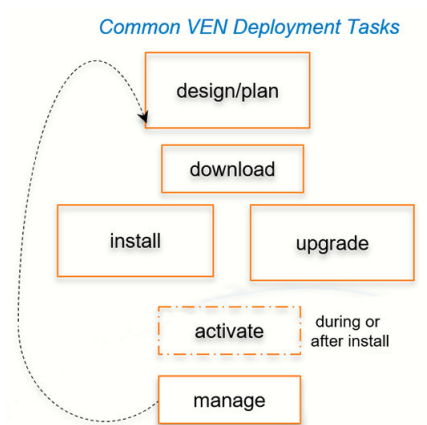


NOTE

If the ipset kernel module is not installed, the VEN downloads and installs it.

Workflows for VEN Installation

The following diagram illustrates the workflow for performing common VEN installation and upgrade tasks.



VEN Installation Planning Checklist

This checklist summarizes VEN planning considerations and requirements detailed in this guide. It compares the requirements and considerations of both deployment methods.

Tasks	VEN Li- brary	VEN CTL
1. Select VEN installation method.	✓	✓
2. Select VEN-to-PCE authentication mechanism: Activation Code or Kerberos.		✓
3. Select whether to activate the VEN to the PCE during or after VEN installation.		✓
4. Select whether to use a single-use or unlimited-use pairing key		✓
5. Review VEN-to-PCE communication requirements.	✓	✓
6. Review VEN workload disk sizing requirements for hosts.	✓	✓
7. Review OS and package dependencies.		✓
8. Determine VEN software package for workload CPU architecture.	✓ (Automatic)	✓
9. Remove parameters <code>ven_repo_url</code> and <code>ven_repo_ips</code> from PCE <code>runtime_env.yml</code> .	✓	
10. (Linux only) Configure mount <code>tracefs</code> or <code>debugfs</code> .	✓	✓
11. Download the VEN software: • VEN Library: All VEN versions in a VEN software bundle • Installation using the CLI and VEN CTL: All VEN versions in a package of VEN software for specific OS versions and CPU architectures.	✓	✓
12. (Optional) Verify signature of downloaded packages against Illumio's public key.		✓
13. Generate VEN pairing profiles and pairing key.	✓	✓
14. Securely copy VEN pairing script to the workload.	✓	
15. (Optional) Prepare VEN-unactivated golden master machine Images.		✓

Prerequisites for VEN Installation

Before installing VENs on the workloads and endpoints in your environment, you must understand and meet the following prerequisites.

PATH Environment Variable for `illumio-ven-ctl`

For more information about using the VEN CTL, see the "illumio-ven-ctl General Syntax" topic in the *VEN Administration Guide*.

For easier invocation of `illumio-ven-ctl` and other control scripts, set your `PATH` environment variable to the directories where they are located:

- Linux: default location is `/opt/illumio_ven`
- Windows: default location is `C:\Program Files\Illumio`

VEN OSs and Package Dependencies

- The VEN software package formally declares dependencies on certain other required software packages. Some of these other packages must be installed before the VEN is installed, while others are installed concurrently during VEN installation.
- The VEN software also makes use of certain other software packages without formally declaring dependencies on them. This allows the same VEN software to be installed on various workloads regardless of which other packages may also be installed. The VEN automatically detects and configures the other software as needed, just-in-time, and reports an error if it encounters any problems with the other software.

For the complete list of package dependencies by operating system, see the [VEN OS Support and Package Dependencies](#) page on the Illumio Support portal.

Minimum key length with RHEL8+ cryptographic policy set to FUTURE

If the cryptographic policy is set to FUTURE on RHEL8, RHEL9, and related operating systems, the RSA key length must be 3072 bits or greater. For more information, see [this Knowledge Base article](#).

VEN-to-PCE Communication

Illumio Segmentation for Data Centers uses Transport Layer Security (TLS) version 1.2 by default for VEN-to-PCE communications.

- The PCE default minimum version is TLS 1.2.
- For VEN versions 18.1 and later, all VENs use TLS 1.2.

For more information about the TLS requirements for VEN-to-PCE communication, see [TLS Versions for Communication](#).

Before installing a VEN, the workload or endpoint must meet the following requirements for VEN-to-PCE communication:

- The workload or endpoint can validate its certificate's chain of trust back to the root Certificate Authority (CA) of the server certificate on the PCE.
- The VEN can reach the PCE on the ports configured for the PCE in the PCE Runtime Environment File `runtime_env.yml`. Contact your Illumio Support representative for more information.
- To prevent time drift between the PCE and VENs, Network Time Protocol (NTP) must be installed and working on the PCE and the VENs.

Workload Disk Size Requirements

Illumio recommends that you reserve **10GB** of disk space on workloads for the VEN. The amount of disk space ultimately used will vary depending on the size and complexity of your microsegmentation environment.

Application logs are rotated from primary to backup when their size reaches 15 MB. Application log files are preserved at reboot, because application logs are stored in files on a workload.

IP Address Support

In Illumio Segmentation for Data Centers 20.2.0 and later releases, the VEN supports both IPv4 and Ipv6 address versions and the IP address version appears correctly in the PCE; for example, in the Workload section of the VEN summary page in the PCE web console.

You can configure how the PCE treats IPv6 traffic from workloads. For more information, see [Manage Security Settings](#).

Obtain the VEN Packages

PCE-based VEN software bundle

If you are an Illumio On-premises customer (you are running the PCE in your corporate data center), download the VEN packages to your PCE by running `illumio-pce-ctl` from your PCE. For more information, see [VEN Library Setup in the PCE \[41\]](#).



NOTE

Illumio Cloud customers don't have shell access to the PCE; therefore, the Illumio Operations team downloads and sets up the PCE-based VEN software bundle for customers. They download all necessary VEN packages for customers.

CLI-based VEN software packages

All VEN software is available for download from the Illumio Support portal. A VEN package is downloadable from the Illumio Support portal for each version of the VEN. Illumio provides the package as a tar file that contains a version of the VEN for all supported operating systems.

To download the VEN package:

1. Go to the Illumio Support site (login required).
2. Select **Software** > **Download** under the VEN section > VEN version.
3. In the VEN Packages row of the VEN table, click the filename for the VEN **tar** file.
4. Download the file to a convenient location.

VEN Package CPU Architecture

For VEN installation using the VEN CTL, after you have downloaded and unpacked the software, determine which VEN is appropriate for your operating system and hardware architecture.

See the [Supported Operating Systems for Illumio VEN](#) table - CPU Architecture Identifier in the Filename column on the Illumio Support portal.

(Optional) Verify Package Signature

For additional security, verify the identity of the downloaded VEN packages against the Illumio public key.



NOTE

- You can verify the signature of the VEN RPM packages for CentOS, Red Hat Enterprise Linux (RHEL), Ubuntu, and SUSE Linux Enterprise Server.
- Signature verification is not supported for AIX, Debian, Solaris, and Windows VEN packages.

The Illumio public key is available from the [Download VEN](#) page of the Illumio Support portal (login required).

For information about using a public key to verify package signatures, see [Checking a Package's Signature](#) on the Red Hat Customer Portal.

Firewall Tampering Protection on Linux

To enable faster host firewall tampering protection (within approximately three seconds) for Linux firewalls, make sure that:

- `tracefs` is mounted (newer Linux distributions)
- `debugfs` is mounted (older Linux distributions that include `tracefs` in `debugfs`)

For information, see "VEN Firewall Tampering Detection" in the VEN Installation and Upgrade Guide.



NOTE

Faster host firewall tampering protection is enabled for Windows automatically.

VEN Compatibility Check

In addition to meeting the requirements in this topic and being aware of the limitations for installing VENs on workloads and endpoints, you can use the VEN Compatibility Check feature to verify the functionality of the VEN. The compatibility information for the VEN is available only while the VEN is in Idle mode.

For information about this feature, see [VEN Compatibility Check](#).

SecureConnect Setup on Workloads

For information about SecureConnect requirements for VENs, see [SecureConnect](#).

MSI to EXE Package Format

Starting with the 21.2.1 Illumio Core release, the Windows VEN installer switched from using the MSI package format to using the EXE package format. Customers using the PCE-based VEN deployment must take an extra step for the transition. Specifically, Illumio Core customers running older, MSI-based Windows VENs must upgrade to 19.3.6+H1-VEN or 21.2.0+H2-VEN before up-

grading their VENs to 21.2.1 or a later version. This release contains the necessary VEN changes to handle the transition in the VEN packaging from MSI to EXE.

VEN Support for Red Hat 5

This section describes Illumio VEN support for the Red Hat 5 operating system.

For more about OS support for the VEN, see [VEN OS Support and Dependencies](#) in the Illumio Support portal.

History of Illumio Red Hat 5 Support

- Illumio's initial support for Red Hat Enterprise Linux (RHEL) 5.5 or greater began in VEN release **18.2.x**.
- Illumio discontinued support for RHEL 5.x in VEN **18.3.x** and later releases.
- Illumio re-introduced support for RHEL 5.x in VEN release **23.2.10** and later releases.
- Beginning in VEN release 24.2.20:
 - Illumio supports Oracle Linux 5 UEK
 - VEN activation fails if the ipset userspace binary is not installed on RHEL 5 workloads.

Requirements

- VEN support for RHEL 5 requires VEN release 23.2.10 or later.
- If no version number is listed for an OS package dependency, use the version included with the OS distribution.
- If you are using a public Certificate Authority to sign your OnPrem PCE, or if you've deployed a SaaS PCE, then to install the VEN on RHEL 5.x, make sure that the latest Certificate Authorities are installed with your operating system.
- Download the VEN software package (**illumio-ven-pkgs**) from the Illumio Support portal and use the VEN CTL to install and activate the VEN on the hosts in your environment.

Limitations for VEN Support on RHEL 5.x

Installing the VEN on RHEL 5 workloads presents the following limitations:

- **Must download VEN software package (RPM) and install and activate using the VEN CTL**

For RHEL 5 VENs, only the VEN CTL installation and activation methods are supported (see [Linux: Install and Upgrade with CLI and VEN CTL](#)). Download the VEN software package (**illumio-ven-pkgs**) from the Illumio Support portal. VEN installation using the VEN Library on the PCE is not supported.

- **On VEN release 24.2.20, VEN activation fails if ipset userspace binary isn't installed on the RHEL 5 workload**

The ipset userspace binary is not installed by default on RHEL 5 workloads. If the binary is missing while attempting to activate a 24.2.20 VEN on a RHEL 5 VEN workload, activation fails and the following Runtime Environment error occurs. To remedy, you must install the provided ipset userspace binary on the workload and then try again to activate the VEN:

```
Cannot find ipset command. Please install /opt/illumio_ven/etc/extras/pkgs/ipset-<version>.x86_64.rpm
```

On VEN releases from 23.2.10 to 24.2.10, VEN activation doesn't fail if the ipset userspace binary isn't detected on the workload but a failure may occur later when the VEN tries to apply policy.

- **Incompatible ipset modules may have been loaded on the workload**

If one or more incompatible ipset modules is loaded on the RHEL 5 workload, the VEN enters an error state and the following error message appears in the **platform.log**. To remedy, you must remove the incompatible ipset module from the workload.

```
ERROR:: ilo_ipsets load returned non-zero. /opt/illumio_ven/bin/ilo_ipsets load -w workload/ .... /new 2>&1
insmod: error inserting '/opt/illumio_ven/etc/extras/modules/el5uek/5.11/ipset/ip_set_iphash.ko': -1 Unknown symbol in
module Cannot load ip_set_iphash module. Cannot load
ip_set_iphash module. Please unload incompatible modules
ip_set.
```

- **Missing ipset modules can cause policy application failure**

On all VEN releases that support RHEL 5 (23.2.10 and later), applying policy fails if compatible ipset modules are missing. Beginning with VEN release 24.2.20, the VEN tries to resolve the failure if it occurs. VEN release 24.2.20 also provides the correct ipset modules for Oracle Linux 5 UEK.

- **VEN tampering may occur if the iptables service loads rules in iptables at boot**

Illumio recommends that you disable the iptables service before you install the VEN on RHEL workloads. This is necessary because the iptables service executes a script that loads rules in **/etc/sysconfig/iptables** at boot, which is undesirable for two reasons: (1) non-VEN rules may conflict with rules the VEN applies to the firewall, and (2) the VEN may regard these non-VEN rules as firewall tampering.

- **The VEN cannot enforce FQDN policy (DNS-based rules).**

- **The VEN doesn't support IPv6 or IPv6 ipsets**
- **The VEN doesn't support byte counting.**
- **AdminConnect (also known as Machine Auth) is not supported.**

ULOG for Traffic Flows

VENs on RHEL 5 workloads use ULOG to log traffic flows.

Illumio Changes to iptables for VENs on RHEL 5.x Workloads



NOTE

These changes to iptables affecting Red Hat 5 also affect other Linux distributions using iptables, including Red Hat 6 and 7, Ubuntu, Debian, and SUSE.

In its support for RHEL 5, Illumio has changed the type of set the VEN uses for ipset.

- To run the VEN on RHEL6/7, Ubuntu, Debian, SUSE versions, the VEN supports using only **hash:net** for ipset, used to store a single IP address or the CIDR.
- To run the VEN on RHEL 5.x, for every ipset, you can select from two ipset types: **iphash**, used to store a single IP address, or **nethash**, used to store a CIDR.

To summarize:

Changes that Illumio made to support VENs on RHEL 5.x include:

- The VEN uses **iphash** and **nethash** set types. Both have a limit of 65536 elements.
- **iphash** can store only single IP addresses; **nethash** can store only CIDRs.

Changes that Illumio made to support VENs on RHEL 6/7, Ubuntu, Debian, SUSE and later versions include:

- The VEN release 23.2.10 and later uses **hash:ip** to store single IP addresses and **hash:net** to store CIDRs.

Example to Illustrate the iptables Changes in Linux

The following example compares how Illumio generated iptables before and after VEN release 23.2.10 in RHEL 7.

RHEL 7 ipset prior to the RHEL 5 change ¹	RHEL 7 ipset after the RHEL 5 change ²	RHEL 5 ipset beginning in VEN release 23.2.10 ³
create ILON-F535B4CAC2EB950D hash:net	create ILON-F535B4CAC2EB95000 hash:ip	-N ILON-F535B4CAC2EB95000 iphash
family inet maxelem 6 hash-size 6	family inet maxelem 3 hash-size 3	
add ILON-F535B4CAC2EB950D 10.10.10.10/32	add ILON-F535B4CAC2EB95000 10.10.10.10	-A ILON-F535B4CAC2EB95000 10.10.10.10
add ILON-F535B4CAC2EB950D 20.20.20.20/32	add ILON-F535B4CAC2EB95000 20.20.20.20	-A ILON-F535B4CAC2EB95000 20.20.20.20
add ILON-F535B4CAC2EB950D 30.30.30.30/32	add ILON-F535B4CAC2EB95000 30.30.30.30	-A ILON-F535B4CAC2EB95000 30.30.30.30
add ILON-F535B4CAC2EB950D 40.40.40.40/24	create ILON-F535B4CAC2EB95001 hash:net family inet maxelem 3 hashsize 3	-N ILON-F535B4CAC2EB95001 ne-thash
add ILON-F535B4CAC2EB950D 50.15.50.50/28	add ILON-F535B4CAC2EB95001 40.40.40.40/24	-A ILON-F535B4CAC2EB95001 40.40.40.40/24
add ILON-F535B4CAC2EB950D 60.60.60.60/31	add ILON-F535B4CAC2EB95001 50.50.50.50/28	-A ILON-F535B4CAC2EB95001 50.50.50.50/28
	add ILON-F535B4CAC2EB95001 60.60.60.60/31	-A ILON-F535B4CAC2EB95001 60.60.60.60/31

- ¹ Prior to VEN release 23.2.10, this is how Illumio generated RHEL 7 for ipset.
- ² Beginning in VEN release 23.2.10, there is no need to change how you define IP Lists in the PCE; the VEN handles everything for you.
- ³ Beginning with VEN release 23.2.10, this is how Illumio generates RHEL 7 for ipset.

VEN Proxy Support

This section describes how to enable proxy support for the VEN on all supported operating systems: Windows, Linux, AIX, and Solaris.



CAUTION

Enforce an allow rule for proxy connectivity

If your environment includes a proxy server, make sure your Illumio policy includes an allow rule for the proxy's **IP:port** before applying a new policy in Selective or Full Enforcement mode. Otherwise, if the VEN discovers that no allow rule is in place allowing the proxy connection, it reports a policy sync error and tries continually to sync policy. In that circumstance, the VEN and the PCE will not be able to communicate.



NOTE

Proxy support setup for the VEN is different between Unix-based versus Windows operating systems due to platform differences. The VENs for Unix-based operating systems do not require system wide proxy setting. For Unix-based VENs, each application obtains the proxy settings from the user, for example, `curl --proxy myproxy:80`. On Windows, the operating system provides proxy settings; for example, the Chrome browser uses the same proxy setting as Microsoft Edge. See the topics below for the details of setting up VEN proxy support by platform.

VEN Connections via Windows Proxy Servers

For Windows workloads only, Illumio Segmentation for Data Centers supports a VEN-to-PCE connection through proxy servers.

- The default proxy configuration on the OS is used and proxy configuration might not be required or available on the VEN.
- Only non-authenticated proxy is supported, which might require that you add an exception for the PCE address.
- Only HTTP proxy is supported. The VEN will detect the proxy automatically and configuration or mode change will not be required.

Configuration for a Windows Proxy Server

- If the network environment supports WPAD protocol, the VEN will automatically use WPAD to discovery proxies and no special configuration is required.
- If proxy configuration is done via a PAC file, you will have to import Internet Explorer's (IE) proxy setting with the PAC file URL to the LocalSystem user (S-1-5-18). The VEN only supports **http://** PAC file URL. It does not support **file://** URLs.
- If proxies are statically configured, you can configure using one of the following two methods:
 - Using **netsh winhttp set proxy** command. This method takes precedence.
For **netsh winhttp** usage, refer to [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc731131\(v=ws.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc731131(v=ws.10)).
 - Importing IE setting with static proxies setting to the LocalSystem user.
For importing IE settings for the VEN, refer to <https://serverfault.com/questions/34940/how-do-i-configure-proxy-settings-for-local-system>.



NOTE

Both IE-based proxy setting and **netsh winhttp** can be pushed to the endpoints (PCs) using Group Policy.

For information about the proxy string format to use for Windows proxy servers, see also [WINHTTP_PROXY_INFO \(winhttp.h\) - Win32 apps](#) in the Microsoft documentation for information.

VEN Connections via Unix-based Proxy Servers

Release 21.1.0 and later releases extend VEN proxy support from Windows to Linux, AIX, and Solaris systems.

In comparison with Windows, the following limitation affects this feature for Unix-based proxy servers. This release doesn't support the Web Proxy Auto Discovery (WPAD) protocol or proxy discovery via the Proxy Auto Discovery (PAC) file for Unix-based proxy servers. This limitation occurs because VENs use LibCurl as the HTTP transport library, but LibCurl does not provide JavaScript execution capability needed to run proxy scripts. For a workaround, see [Proxies - Everything curl](#).

Configuration for Unix-based Proxy Servers

To set up your environment for a Unix-based proxy server, perform the following steps:

1. Set the proxy string during activation using the `--proxy-server` option. For example, use `illumio-ven-ctl activate --management-server example.com:8443 --activation-code <code> --proxy-server 172.24.88.114:3128` as shown:

```
root@qual-solaris11-L:/opt/illumio# /opt/illumio_ven/illumio-ven-ctl activate --management-server example.com:8443 --activation-code <code> --proxy-server 172.24.88.114:3128
Checking Runtime Environment.....
Activating Illumio
-----
Storing Activation Configuration .....
Starting Illumio Processes..... Pairing
Status
-----
Pairing Configuration exists .....SUCCESS
VEN Manager Daemon running .....SUCCESS
Master Configuration retrieval ....SUCCESS
VEN Configuration retrieval .....SUCCESS
VEN has been SUCCESSFULLY paired with Illumio
root@qual-solaris11-L:/opt/illumio# /opt/illumio_ven/illumio-ven-ctl show-proxy
proxy_server: 172.24.88.114:3128
```

2. Set or modify the proxy string using `illumio-ven-ctl set-proxy proxy-string` and clear the proxy setting using `illumio-ven-ctl re-set-proxy` as shown:


```
[root@ven-rhel illumio_ven]# ./illumio-ven-ctl set-proxy
http://proxy.example.com:3128
```

Updating proxy to http://proxy.example.com:3128. VEN restart needed.

```
[root@ven-rhel illumio_ven]# ./illumio-ven-ctl restart
Shutting down illumio-control:
- venAgentMonitor Stopping venAgentMonitor:
[ OK ]
```

<snip>

Starting illumio-control:

```
- Environment Setting up Illumio VEN Environment:
[ OK ]
```

<snip>

```
[root@ven-rhel illumio_ven]# ./illumio-ven-ctl show-proxy
proxy_server: http://proxy.example.com:3128
```

```
[root@ven-rhel illumio_ven]# ./illumio-ven-ctl reset-proxy
Resetting proxy. VEN restart needed.
```

```
[root@ven-rhel illumio_ven]# ./illumio-ven-ctl restart
Shutting down illumio-control:
```

```
- venAgentMonitor Stopping venAgentMonitor:
[ OK ]
```

<snip>

Starting illumio-control:

```
- Environment Setting up Illumio VEN Environment:
[ OK ]
```

<snip>

```
[root@ven-rhel illumio_ven]# ./illumio-ven-ctl show-proxy
No proxy is set
```

3. Restart the VEN after the proxy is set, modified, or cleared, except when the proxy is enabled using **--proxy-server** during activation. Query your current proxy setting using the **illumio-ven-ctl show-proxy** command.
4. Use the proxy string format: [**<scheme>://"**]**<server>**[**":"<port>**]
In the string format, **[]** indicates optional values in the command and **<>** indicates required values in the command; therefore, specifying either **--proxy-server 172.24.88.114:3128** or **http://172.24.88.114:3128** are both valid.

**NOTE**

When specified, only the “http” scheme is supported. Schemes such as “https” or any other schemes are not supported. For example, `http://myproxy:8080` or `http://10.0.0.2:80`.

For Linux RPM (or AIX **installp**) installation, you can set the proxy string by setting and exporting the proxy string from the **VEN_PROXY_SERVER** shell variable before invoking the RPM (or **installp**) command.

For Solaris **pkgadd**, you can set the proxy string by setting the **VEN_PROXY_SERVER** variable to an answer file (typically created using the **pkgask** command).

Linux Pairing Script Activation for Proxy Servers

Typically, VENS are paired with the PCE directly. However, if a workload is behind a Web Proxy, you must follow these steps to enable your Linux/Unix VEN to successfully pair to your PCE:

1. From the PCE web console menu, go to **Servers & Endpoints > Pairing Profile**.
2. Copy the pairing line from the Linux/Unix OS Pairing Script window.
3. Paste this pairing line into a text file so that you can edit it.
4. Edit the pairing line to make the following two changes (displayed in **bold**):
 - a. Add **-x <proxy-string>** to the curl command to indicate the proxy string.
 - b. Add **--proxy-server <proxy-string>** to the switch to pass the proxy string to the pairing script.

```
rm -fr /opt/illumio_ven_data/tmp && umask 026 &&
mkdir -p /opt/illumio_ven_data/tmp && curl -x <proxy-string> --tlsv1 "https://example.com:8443/api/v18/software/ven/image?pair_script=pair.sh&profile_id=1" -o /opt/illumio_ven_data/tmp/pair.sh && chmod +x /opt/illumio_ven_data/tmp/pair.sh && /opt/illumio_ven_data/tmp/pair.sh --management-server <server fqdn> --proxy-server <proxy-string>
```
5. Paste the revised script into the Linux/Unix terminal and press **Enter**.

The workload starts the pairing process. As the pairing script runs, you will see success messages appear. Wait until you see the message “Workload has

been SUCCESSFULLY paired with Illumio,” which means your VEN (behind a proxy server) and the PCE are paired.

Configure a VEN-specific Windows Proxy

Beginning in VEN release 22.5, you must explicitly configure the Windows proxy for the VEN. In previous VEN releases, you didn’t need to configure a proxy on Windows operating systems; instead, the VEN discovered proxy configurations automatically using the WPAD protocol or the Internet Explorer browser PAC file.



CAUTION

Enforce an allow rule for proxy connectivity

If your environment includes a proxy server, make sure your Illumio policy includes an allow rule for the proxy’s **IP:port** before applying a new policy in Selective or Full Enforcement mode. Otherwise, if the VEN discovers that no allow rule is in place allowing the proxy connection, it reports a policy sync error and tries continually to sync policy. In that circumstance, the VEN and the PCE will not be able to communicate.



IMPORTANT

This topic applies to VENs deployed on servers or virtual machines and Endpoint VENs (VENs deployed on endpoints, such as Windows laptops.)

Ways to Configure a VEN-specific Windows Proxy

There are two ways to configure a VEN-specific Windows Proxy:

- [Configure Windows Proxy by Editing the Pairing Script \[38\]](#)
- [Configure Windows Proxy using CTL Commands \[38\]](#)

Configure Windows Proxy by Editing the Pairing Script



NOTE

For details about generating a pairing key and pairing script, see "Pairing Profiles and Scripts" in the section *VEN Installation & Upgrade Using VEN Library* in this guide.

You can configure a VEN-specific Windows proxy by adding the **-proxy-server** parameter to the pairing script.

In the following example, the **proxy-server** parameter is placed at the end of the script after the **-activation-code** parameter:

```
illumio-ven-ctl.exe show-proxy PowerShell -Command "& {Set-ExecutionPolicy -Scope process remotesigned -Force; Start-Sleep -s 3; Set-Variable -Name ErrorActionPreference -Value SilentlyContinue; [System.Net.ServicePointManager]::SecurityProtocol=[Enum]::ToObject([System.Net.SecurityProtocolType], 3072); Set-Variable -Name ErrorActionPreference -Value Continue; (New-Object System.Net.WebClient).DownloadFile('https://example-server.io:443/api/v26/software/ven/image?pair_script=pair.ps1&profile_id=60', (echo $env:windir\temp\pair.ps1)); & $env:windir\temp\pair.ps1 -management-server example-server:443.io -activation-code <code> -proxy-server <proxy_server:port>;}"
```

Configure Windows Proxy using CTL Commands

Use the following CTL commands to explicitly configure a Windows proxy.

Installation:

```
<VEN Installation Directory>\illumio-ven-22.2.32-xxxx-preview.win.x64.exe /install
VEN_PROXY_SERVER=<proxy_server:port>
```

Activation:

```
<VEN Installation Directory>\illumio-ven-ctl.exe activate
-management-server <pce_server:port> -activation-code <code>
-proxy-server <proxy_server:port>
```

Restart:

You must restart the VEN after setting (or changing) its proxy configuration.

```
<VEN Installation Directory>\illumio-ven-ctl.exe restart
```

Manage the proxy configuration:

The VEN CTL supports using the **set-proxy**, **reset-proxy**, and **show-proxy** commands to configure a proxy on Windows. Use of these commands takes precedence over **netsh** and discovery using the Internet Explorer PAC file.

For more information about how the Windows VEN supports a proxy server, see "VEN Proxy Support" in the section *Prepare for VEN Installation* in this guide.

The **set-proxy** command sets the proxy server for the VEN to use.

```
<VEN Installation Directory>\illumio-ven-ctl.exe set-proxy
<proxy_server:port>
```

The **show-proxy** command shows the current proxy configuration.

```
<VEN Installation Directory>\illumio-ven-ctl.exe show-proxy
```

The **reset-proxy** command removes the current proxy configuration.

```
<VEN Installation Directory>\illumio-ven-ctl.exe reset-proxy
```

VEN-Specific Proxy Configuration

In the 23.4-VEN release, you can choose to explicitly configure the Windows proxy for the VEN.

**CAUTION**

Enforce an allow rule for proxy connectivity.

If your environment includes a proxy server, make sure that your Illumio policy includes an allow rule for the proxy's **IP:Port** before applying a new policy in Selective or Full Enforcement mode. Otherwise, if the VEN discovers that no allow rule is in place to allow the proxy connection, it reports a policy sync error and tries continually to sync policy. In that circumstance, the VEN and the PCE will not be able to communicate.

About the VEN-Specific Proxy

The VEN CTL (including pairing script) supports the set-proxy, reset-proxy, and show-proxy commands to configure a proxy on Windows.

When configured with these commands, the setting takes precedence over netsh and discovery using the Internet Explorer PAC file, as shown here:

Direct > VEN specific proxy (NEW) > WinHttp Proxy > IE setting for localSystem account

For more information about how the Windows VEN supports a proxy server, see [VEN Proxy Support](#).

Explicitly Configure a Windows Proxy

Use the following commands to explicitly configure a Windows proxy:

Installation:

```
<VEN Installation Directory>\illumio-ven-22.2.32-xxxx-  
preview.win.x64.exe/  
install  
VEN_PROXY_SERVER=<proxy_server:port>
```

Activation:

```
<VEN Installation Directory>\illumio-ven-ctl.ps1 activate -  
management-server <pce_server:port> - activation-code <code>  
-proxy-server <proxy_server:port>
```

Restart:

You must restart the VEN after setting or changing its proxy configuration:

```
<VEN Installation Directory>\illumio-ven-ctl.ps1 restart
```

Proxy Configuration Management:

The set-proxy command sets the proxy server for the VEN to use.

```
<VEN Installation Directory>\illumio-ven-ctl.exe set-proxy  
<proxy_server:port>
```

The show-proxy command shows the current proxy configuration:

```
<VEN Installation Directory>\illumio-ven-ctl.exe show-proxy
```

Set Up PCE for VEN Installation

Before you log into the PCE web console to install or upgrade the VENs in your environment, first make sure to complete the PCE setup tasks described in this section.

VEN Library Setup in the PCE

You can use your PCE cluster as a centralized mechanism for distributing, installing, and upgrading VENs in your organization.

About the VEN Library in the PCE

You can use the PCE web console to install and upgrade VENs in your organization in the following scenarios:

- To install or upgrade VENs on servers running supported versions of RPM, Debian, and Windows operating systems. Other operating systems are not supported for Server VENs.
- To install or upgrade VENs on endpoints running supported versions of Windows and macOS operating systems. Other operating systems are not supported for Endpoint VENs
- The PCE and VEN versions are 18.2 and later.

VEN installation from the PCE does not affect any processes you might already have for installing or upgrading VENs directly on workloads, such as installation or activation/pairing with `illumio-ven-ctl`. Those processes can continue until and after you decide to use the PCE to install and upgrade VENs.

This topic primarily describes how to use the PCE web console to install and upgrade VENs. However, you can also use the Illumio Segmentation for Data Centers REST API to upgrade (but not install) VENs. See the *REST API Developer Guide* for information.

VEN Library

Previously, VENs could be deployed from an external VEN repository (VEN repo) or by manually installing the VEN packages directly on your workloads and endpoints.

From the 18.2.0 release onwards, the PCE can act as a repository for distributing, installing and upgrading the VEN software. The PCE can host multiple VEN versions, allowing you to evaluate and certify new versions of the VEN while continuing to deploy older versions in production.

PCE-based installation and upgrade of VENs replaces the use of the external VEN repo, which is no longer supported for VEN version 18.2.0 or higher. A migration path is available for Illumio Secure Cloud customers and on-premises customers with VEN repos upgrading VENs to 18.2.0.

Using the VEN Library to install and upgrade VENs on your workloads and endpoints has the following benefits:

- The VEN software bundle loaded on a PCE is replicated to all PCE core nodes.
- You can view VEN versions from the VEN Library page in the PCE web console.
- You can download software on workstations.
- Multiple versions of VEN software can exist on the PCE.
- You can specify an initial VEN version in pairing profiles.
- You can specify a default VEN version when the PCE has multiple VEN versions uploaded.
- You can add and remove VEN versions from the PCE.
- You can specify server and endpoint VEN bundles separately.
- You can use the PCE to upgrade all VENs or selected VENs in your organization.

After setting up the VEN software bundle using the PCE control interface `illumio-pce-ctl`, the VEN Library page is available in **Support > VEN Library**. From this page, you can download individual VEN packages and view the dependencies and supported OS versions.

**NOTE**

You must set an initial VEN version when there is no system default version or an external repository has not been configured. If your PCE has existing pairing profiles created without versions, pairing will fail when use those un-versioned profiles.

Migration to PCE-Based VEN Library

Migration from the central VEN repo or an on-premises VEN repo to the VEN Library should be thoroughly planned and timed to not impact your current operations. Contact Illumio Customer Support for assistance.

PCE Runtime Parameters for PCE-based Installation

After you have migrated from any external VEN repo you might have, remove the following parameters from the PCE `runtime_env.yml` file:

- `ven_repo_url`
- `ven_repo_ips`

These parameters are not needed for the PCE-based installation of the VEN. They are deprecated and should no longer be used.

Workflow for VEN Library Setup

You do not have to make any configuration changes or other settings to enable the VEN Library on the PCE.

Loading the VEN bundle into the PCE VEN Library allows you to use the PCE web console or Illumio Segmentation for Data Centers REST API to install and upgrade VENs in your organization.

To set up the VEN Library, perform the following high-level tasks:

1. Upload the VEN upgrade compatibility matrix to the PCE. See [Upload VEN Upgrade Compatibility Matrix \[45\]](#).

**NOTE**

The compatibility matrix must be uploaded to the PCE before you upload any VEN software bundles or you will get an error.

2. Download the version of the VEN software bundle from the Illumio Support site.
 - a. On the Illumio Support site (login required), go to **Software > Download > VEN - Download**.
 - b. In the Download VEN page, select the radio button for the VEN version you want to set up. From the table, click the filename link for the “VEN Bundle for PCE-based deployment.”

**TIP**

Illumio recommends that you verify the checksum of the VEN software bundle after downloading it.

The VEN software for PCE-based deployment is a zipped tarball (tar file) of a version of VEN software for all supported workload platforms. This tarball is known as a *VEN software bundle*. The tar file downloads to your local drive.

3. Repeat step 1 for all the VEN versions you want to distribute to your workloads.

Additionally, when Illumio releases new versions of the VEN software, plan on repeating these steps when you are ready to deploy that VEN version.

4. Copy or move the VEN software bundle tar file to a convenient directory on your PCE core node or to any system that your PCE can reach with HTTP, SFTP, or SCP.

You do not need to unpack the VEN software bundle tar file.

5. Load the VEN software bundle into one of the PCE core node's VEN Library. From this node, the VEN software bundle is automatically copied to the other nodes.

See [Upload VEN Software Bundle into PCE VEN Library \[46\]](#) for information.

6. Install or upgrade VENs:
 - a. To install the VEN software on workloads, with the PCE web console, generate a pairing script. See [Pairing Profiles and Scripts \[56\]](#) for information.
 - b. To upgrade all VEN workloads or selective workloads, use the PCE web console. See [VEN Upgrade Using the PCE Web Console \[76\]](#) for information.

Upload the VEN Upgrade Compatibility Matrix



NOTE

The compatibility matrix must be uploaded to the PCE before you upload any VEN software bundles or you will get an error.

Alternatively, you can run the compatibility matrix upload command in one line with the command to install a VEN software bundle; for example:

```
sudo -u ilo-pce illumio-pce-ctl ven-software-install  
bundle_path --compatibility-matrix matrix_file_path
```

As part of setting up the VEN Library in the PCE, you must upload the VEN upgrade compatibility matrix to the PCE. The compatibility matrix contains information about valid VEN upgrade paths and VEN to PCE version compatibility. To use the PCE web console and the Illumio Segmentation for Data Centers REST API, you must upload this matrix for VEN upgrades to be successful.

In Supercluster, VENs are managed from the PCE they are paired to. You must upload VEN bundles and the compatibility matrix to each PCE.

The compatibility is a zipped tarball (tar file). You do not need to unpack the tar file to install it. The tarball contains a set of JSON files specifying the rules for upgrading VENs in your organization.

You can also view these VEN upgrade rules on the Illumio Support site (log in required). Go to **Software > Upgrade > VEN - Upgrade**. In the Upgrade VEN page, select your current VEN version and the version you want to upgrade to. Click **Find My Upgrade Path**.

**IMPORTANT**

Until you upload this file, you can only install VENs on workloads when the VEN version is the same as the version of the PCE managing those VENs. Attempting to upgrade a VEN version returns the message: “No valid upgrade paths were found for this release.”

To install the compatibility matrix:

**TIP**

Illumio recommends that you verify the checksum of the compatibility matrix file after downloading it.

1. Download the VEN upgrade compatibility matrix tar file from the Illumio Support site (log in required).
 - a. On the Support page, click **SOFTWARE** in the upper right of the page and then select **DOWNLOAD**. (Alternatively, you can click the Software tile on the body of the page.)
 - b. On the Software page, click **Compatibility Matrix**, and then in the table that appears click the link for the compatibility tar file. The file downloads to your local drive.
2. Copy or move the tar file to a convenient directory on your PCE core node.
3. To upload the file to the PCE, run this command on the PCE:

```
sudo -u ilo-pce illumio-pce-ctl compatibility-matrix-install
matrix_file_path
```

Upload the VEN Software Bundle into the PCE

**NOTE**

Before you upload a VEN software bundle into the PCE, you must first have uploaded the VEN upgrade compatibility matrix. See [Upload the VEN Upgrade Compatibility Matrix \[45\]](#) for information.

Loading the VEN software bundle consists of running **illumio-pce-ctl** on the PCE command line to load the VEN software bundle into the PCE's VEN Library. The VEN Library is then replicated to the other PCE core nodes.

Loading the VEN software bundle into the PCE's VEN Library is what configures the PCE as the VEN installation and upgrade method.

In Supercluster, VENs are managed from the PCE they are paired to. You must upload VEN bundles and the compatibility matrix to each PCE.

You can only upload VEN software bundles into a PCE if the VEN is compatible with that PCE. For example, you cannot upload VEN version 21.5.0 software bundles into a PCE version 21.2.0.

To load a VEN software bundle:

1. Copy the downloaded VEN software bundles to a convenient location on your PCE core node or to any system that the PCE can access via HTTP, SFTP, or SCP.
2. To load the VEN software bundle, run the following command on the core node's command line.

```
sudo -u ilo-pce illumio-pce-ctl ven-software-install
bundle_path
```

For example:

```
sudo -u ilo-pce illumio-pce-ctl
ven-software-install protocolAndFqdnOfVenBundleHost/
nameOfVenSoftwareBundleFile.tar.bz2
```

Where:

- **bundle_path** is any of the following locations of the VEN software bundle tar file:
 - The absolute or relative path to the directory on the PCE
 - The HTTP URL to the host and file
 - The SFTP URL to the host and file
 - The SCP URL to the host
- The filename of the VEN software bundle tar file uses the following format:

```
illumio-ven-repo-someVersionStamp.tar.bz2
```

Where **someVersionStamp** is the version and build number of the Illumio Segmentation for Data Centers release.

Example

The following example assumes you have copied the VEN software bundle into `/var/tmp` on your PCE:

```
sudo -u ilo-pce illumio-pce-ctl ven-software-install /var/tmp/illumio-ven-repo-someVersionStamp.tar.bz2
Reading /opt/pce_config/etc/runtime_env.yml.
```

```
Validating VEN release tarball file contents:
```

```
Valid.
```

```
Deploying VEN release tarball to 'PCE's IP address' .
```

```
Committing tarball manifest information to database.
```

```
Are you sure you want to continue? [yes/no]: yes
```

```
Release version_of_bundle Successful.
```

HTTP and SCP Examples

These examples show HTTP and SCP URLs on the `illumio-pce-ctl ven-software-install` command:

- HTTP:

```
sudo -u ilo-pce illumio-pce-ctl ven-software-install http://myVENrephost.BigCo.com/myRepoDir/pcerepo/illumio-ven-repo-someVersionStamp.tar.bz2
```

- SCP:

```
sudo -u ilo-pce illumio-pce-ctl ven-software-install scp://albert.einstein@myhost.BigCo.com:illumio-ven-repo-someVersionStamp.tar.bz2
```

Set the Default VEN Version in the Library

You can set a default version of the VEN software for all workloads or for selected pairing profiles. You can use both methods simultaneously. For example:

- Set a default VEN version for all workloads when you are ready to roll out that specific version.
- Create a separate pairing profile with a specific VEN version for test, evaluation, and certification before general rollout.

Set the Default VEN Version for All Workloads

To define the default VEN version for all workloads, run this command on the PCE:

```
sudo -u ilo-pce illumio-pce-ctl ven-software-release-set-  
default release
```

Where:

release is a release identifier like 19.3.0-6623. The PCE uses the default release to determine what release of the VEN to install when you pair a VEN with a workload. You can override the default release for specific pairing profiles. To obtain release IDs, run the `sudo -u ilo-pce illumio-pce-ctl ven-software-releases-list` command.

Set Default VEN Version for Specific Pairing Profile

You can selectively set a VEN version for specific pairing profiles. The profiles that have a defined VEN version create pairing profiles that install that specific VEN version on the workload. Other pairing profiles that have no VEN version set are unaffected.

To set a pairing profile's VEN version, see [Pairing Profiles and Scripts \[56\]](#).

For information about pairing scripts, see [Pairing Profiles and Scripts \[56\]](#).

Remove a Release from the VEN Library

To remove a VEN version from the VEN Library on the PCE, run this command on the PCE:

```
sudo -u ilo-pce illumio-pce-ctl ven-software-release-delete  
release
```

Where:

release is a release identifier like 19.3.0-6623. To obtain release IDs, run the `sudo -u ilo-pce illumio-pce-ctl ven-software-releases-list` command.

**IMPORTANT**

To remove a VEN version from the PCE database, the PCE cannot be using that VEN version in pairing profiles and it cannot be set as the default VEN version for pairing with workloads. When your orgs no longer use that VEN version, the **ven-software-release-delete** command will remove the VEN software bundle from the PCE file system.

View the VEN Library in the PCE

The VEN loading process with `sudo -u ilo-pce illumio-pce-ctl ven-software-install` prints its success or failure when it completes. You can also verify the successful loading in the following ways:

- In the PCE web console, look at the VEN Library. Navigate to **Support > VEN Library** to see that the bundle has been loaded.
- On the PCE command line, run the following command:

```
sudo -u ilo-pce illumio-pce-ctl ven-software-releases-list
```

PCE Maintenance for VEN Library

These are some points to consider about backing up and modifying your PCE cluster for the PCE-based deployment model.

About PCE Backups

Be sure that your backup included the PCE's VEN library and is not earlier than when you loaded the VEN software bundles into the PCE's VEN Library. If you restore from an earlier backup, you need to either reload the VEN library or redeploy from an existing core node.

About Complete PCE failure

In case of a catastrophic failure of the PCE cluster, after rebuilding or reinstalling the cluster, reload the VEN software bundles into a PCE core node's VEN library.

VEN-related Maintenance Commands on PCE

The `illumio-pce-ctl` control script has options for VEN maintenance, such as add new VEN software bundle, remove VEN version, and delete VEN version. See the `illumio-pce-ctl --help` details.

Some of the options for distributing VENs from the PCE show `org-id`, `org-list`, and other organization-related arguments. None of the organization-related options or arguments are needed for distributing VENs from your on-premises PCE and do not need to be specified.

Set up Kerberos Authentication on PCE

You can configure the PCE and VEN to rely on authentication by a pre-configured Kerberos-based system, such as Microsoft Active Directory.

About Enabling Kerberos Authentication

1. Enable Kerberos on the PCE.
2. Configure Kerberos-based authentication of the VEN at installation. Illumio Segmentation for Data Centers supports Kerberos authentication for Linux, Windows, Solaris, and AIX VENs.

For information, see the "Kerberos for Linux VEN-to-PCE Authentication" and "Kerberos for Windows VEN-to-PCE Authentication" topics.

Requirement for Kerberos Authentication

For all VENs to be paired via Kerberos, be sure to add policy rules allowing access to the required Kerberos servers.

Obtain an activation code for the VEN. When installing the VEN by using the VEN CTL, you can use the activation code either during installation or after installation. For information about activation codes for the VEN, see the "About the VEN Activation Code" section.

About Kerberos Authentication on the PCE

To use Kerberos authentication to pair a workload, you must enable Kerberos authentication on the PCE. Kerberos authentication requires configuring the following parameters in your PCE's `runtime_env.yml` file:

PCE Runtime Environment File Parameter	Description
<code>kerberos_device_auth_service_name:</code> <code>kerberos_device_auth_keytab_file</code>	Kerberos authentication for VENs on devices. These parameters enable Kerberos authentication for the VENs and other devices and provide a Kerberos service name and keytab file. These parameters are only used when the PCE node's role is set to agent_service. The kerberos_device_auth_service_name must contain the complete Service Principal Name (SPN); for example, servicename/fqdn@realm.
<code>kerberos_user_auth_service_name:</code> <code>kerberos_user_auth_keytab_file</code>	Kerberos user authentication for the Login Service. These parameters enable Kerberos authentication for the Login Service and provide a Kerberos service name and keytab. - The kerberos_user_auth_service_name must contain the complete Service Principal Name (SPN); for example, servicename/fqdn@REALM. Kerberos requires that the REALM be in all capital letters. - The kerberos_user_auth_service_name is the path to the PCE's Kerberos keytab file. Any key included in keytab can be used for authentication

Prepare Scripts

The **prepare** script is used for creating golden images to activate the VEN the first time the image is booted.

Prepare Golden Image for Workload Installation

Many organizations use “golden images” for faster deployment. When using a golden image to install a VEN, you have two options for pairing with the PCE:

- **Option 1:** [\[53\]](#) Use a modified version of the Illumio Core pairing script called **prepare** to ensure these golden images have the VEN pre-installed.
- **Option 2:** [\[53\]](#) Use the **illumio-ven-ctl** control script.

**IMPORTANT**

- **Avoid activating the VEN prematurely:** If you will enable your images with the prepare script, make sure to do so as the last step in your process for building the image. The prepare script takes effect at the next system boot, which means the VEN might be activated prematurely on the image itself. If you have other software to install on the image and the image requires reboot, the VEN is activated at once, which is not desirable.

Additional information (log in required):

- [Citrix VEN VDI Image Preparation Script](#)
- [VMWare Horizon VDI Instant Clone VEN Deployment](#)
- **Specify the correct activation code type:** There are two types of activation codes in the pairing profile available in the PCE web console: **one-time use** and **unlimited use**. Be sure to specify the correct type for your needs. For more information, see [Configure Pairing Key Usage and Lifespan](#).

Option 1: Prepare the workload using the Pairing Profile/Pairing script

This option relies on a modified version of the Illumio Core pairing script called **prepare** to ensure these golden images have the VEN pre-installed.

1. In the PCE web console, create a pairing profile or select an existing pairing profile. For information, see [Pairing Profiles and Scripts \[56\]](#).
2. Copy the pairing script.
3. In the copy of the script, change all occurrences of **pair** to **prepare**.
4. Execute the modified script on the image.

The **prepare** script installs the VEN on the image. When the prepare script finishes, the VEN is stopped. The script configures the VEN to start the next time the workload is booted.

Option 2: Prepare the workload with `illumio-ven-ctl`

Use `illumio-ven-ctl` to place the image in **prepare** mode.

From a command line, execute `illumio-ven-ctl`, making sure to include the **prepare** argument and the **management-server** and **activation-code** information shown in the following examples.

Windows: use single dashes (-)

```
<VEN Installation Directory>\illumio-ven-ctl prepare
-management-server <pce_fqdn:port> -activation-code
<activation_key>
```

Linux: use double dashes (--)

```
<VEN Installation Directory>/illumio-ven-ctl prepare
--management-server <pce_fqdn:port> --activation-code
<activation_key>
```

Auto Scaling Linux Workloads

To enable auto scaling for Linux workloads, perform the following steps:

1. Select an existing VM instance that you want to create a new instance for.
2. Inside the PCE web console, create a pairing profile (or select an existing pairing profile).
3. Copy the Linux pairing script and paste it into a text editor.

```
rm -fr /opt/illumio_ven_data/tmp && umask 026 && mkdir
-p /opt/illumio_ven_data/tmp && curl --tlsv1 "https://
pce.example.com:8443/api/v18/software/ven/image?
pair_script=pair.sh&profile_id=1" -o /opt/
illumio_ven_data/tmp/pair.sh && chmod +x /opt/
illumio_ven_data/tmp/pair.sh && /opt/illumio_ven_data/tmp/
pair.sh --management-server pce.example.com:8443 --
activation-code
11a12969c511197eb7ae1e175b9b49382fe1bc011b2a2228c8a184cc6c9f7
5663325146e5d5ac7c5d
```

4. Find all occurrences in the script where `pair.sh` is used and replace them with `prepare.sh` so that the script looks like this:

```
rm -fr /opt/illumio_ven_data/tmp && umask 026 && mkdir
-p /opt/illumio_ven_data/tmp && curl --tlsv1 "https://
pce.example.com:8443/api/v18/software/ven/image?
pair_script=prepare.sh&profile_id=1" -o /opt/
illumio_ven_data/tmp/prepare.sh && chmod +x /opt/
illumio_ven_data/tmp/prepare.sh && /opt/illumio_ven_data/tmp/
prepare.sh --management-server pce.example.com:8443 --
activation-code
11a12969c511197eb7ae1e175b9b49382fe1bc011b2a2228c8a184cc6c9f7
5663325146e5d5ac7c5d
```

The **prepare.sh** script installs the VEN on the new workload and configures it so the VEN starts running as soon as the new workload is instantiated.

5. Run the modified script on the Linux instance.
6. Configure your auto scaling policy to use an image that contains the **prepare** script.

The **prepare.sh** script installs the VEN on the new workload and configures it so the VEN starts running as soon as the new workload is instantiated.

Auto Scaling for Windows Workloads

To enable auto scaling for Windows workloads, perform the following steps:

1. Select an existing VM instance that you want to create a new instance for.
2. In the PCE web console, create a pairing profile (or use an existing pairing profile).
3. Copy the Windows pairing script and paste it into a text editor.

```
PowerShell -Command "& {Set-ExecutionPolicy -Scope process
remotesigned -Force; Start-Sleep -s 3; Set-Variable -Name
ErrorActionPreference -Value SilentlyContinue;
[System.Net.ServicePointManager]::SecurityProtocol=[Enum]::To
Object([System.Net.SecurityProtocolType], 3072); Set-
Variable -Name ErrorActionPreference -Value Continue; (New-
Object System.Net.WebClient).DownloadFile('https://
pce.example.com:8443/api/v18/software/ven/image?
pair_script=pair.ps1&profile_id=1', (echo
$env:windir\temp\pair.ps1)); & $env:windir\temp\pair.ps1
-management-server pce.example.com:8443 -activation-code
11a12969c511197eb7ae1e175b9b49382fe1bc011b2a2228c8a184cc6c9f7
5663325146e5d5ac7c5d;}"
```

4. Find all occurrences in the script where **pair.ps1** is used and replace them with **prepare.ps1**. so that the script looks like this:

```
PowerShell -Command "& {Set-ExecutionPolicy -Scope process
remotesigned -Force; Start-Sleep -s 3; Set-Variable -Name
ErrorActionPreference -Value SilentlyContinue;
[System.Net.ServicePointManager]::SecurityProtocol=[Enum]::To
Object([System.Net.SecurityProtocolType], 3072); Set-
Variable -Name ErrorActionPreference -Value Continue; (New-
Object System.Net.WebClient).DownloadFile('https://
pce.example.com:8443/api/v18/software/ven/image?
pair_script=prepare.ps1&profile_id=1', (echo
$env:windir\temp\prepare.ps1)); &
$env:windir\temp\prepare.ps1 -management-server
pce.example.com:8443 -activation-code
11a12969c511197eb7ae1e175b9b49382fe1bc011b2a2228c8a184cc6c9f7
5663325146e5d5ac7c5d;}"
```

5. Run the modified script on the Windows instance.
6. Configure your auto scaling policy to use the prepared image.

The `prepare.ps1` script installs the VEN and configures it such that the VEN starts running as soon as the new workload is instantiated.

VEN Installation & Upgrade Using VEN Library

The following topics describe how to install and upgrade the VEN by using the VEN Library in the PCE.



NOTE

Before you perform the tasks described in this section, the PCE must be set up with the VEN Library. For information, see [VEN Library Setup in the PCE \[41\]](#).

Pairing Profiles and Scripts

Pairing profiles allow you to specify the properties you want workloads to have when you pair them with the PCE. This includes properties such as specific labels, enforcement state, visibility mode, command line overrides, uses per key, maximum key age, and more.

The pairing profile includes the characteristics to create a pairing script to run on workloads. The pairing script installs the VEN software, activates it, and

gets the workloads ready to accept security policy from the PCE. “Pairing” is also known as “installation and activation.” The pairing script contains a unique pairing key (activation code) that identifies the VEN securely so it can authenticate with the PCE. You can configure the pairing key to be used one time or multiple times, and you can configure its time and use limit.

Workflow for Using Pairing Profiles

Creating and using pairing profiles follows this general workflow:

1. Create a pairing profile.
2. Generate a pairing script.
3. Copy the script to the workload and run it.

The following conditions apply when installing VENs by using pairing profiles:

- An activation code/pairing key is required. In the PCE web console, you can specify either a single, one-time activation code or an unlimited, multi-use activation code.
- The pairing script is not absolutely required. It is an alternative to installing VEN software installation and activation with the VEN CTL (`illumio-ven-ctl`).

Which VEN Version is Installed

When installing and activating a VEN using a pairing script generated by a pairing profile:

- If you set a specific VEN version in the PCE web console for the pairing profile used to generate the pairing script, that specific VEN version is installed on the workload.
- Otherwise, if you have set a default VEN version for all workloads, that VEN version is installed on the workload. The PCE web console shows the current default VEN version in the “Initial VEN Version” dropdown. If no default VEN version is specified, the PCE uses the latest VEN version available in the VEN library.

The Default Pairing Profile

Item	Description
Name	Default
Labels	Role=<Blank>
	Application=<Blank>
	Environment=Production
	Location=<Blank>
Workload State	Visibility Only
Uses per Key	Unlimited
Maximum Key Age	Unlimited
Command Line Overrides	Unlocked (CLI can override anything)

Last Pairing Key Generated and Last VEN Paired

The following information appears on the pairing profile details page:

- The last time a pairing key was generated using this pairing profile.
- The last time a VEN was paired using this pairing profile.

Filter the Pairing Profiles List

You can filter the pairing profiles list using the properties filter at the top of the list. You can filter the list by entering a label type to show only those pairing profiles that use the selected labels. You can further filter the list by selecting specific properties of the pairing profiles. For example, you can filter the list by a pairing profile's name.

Click **Export** and select JSON or CSV format to generate the pairing profiles report. Once generated, you can either click the download icon next to Export to download the generated report or select **Export > View All Exports** to view the report details.

Configure a Pairing Profile

You can configure a pairing profile to set the initial workload policy state at the time of pairing. For example, you might want to pair workloads in

the Visibility state so you can view network traffic to build policies before enforcing them.

On the other hand, if you are configuring an auto-scale policy and want to pair workloads automatically based on application demands, you can choose to have workloads paired in Full enforcement state.

1. Go to **Servers & Endpoints > Pairing Profiles**.
2. Click **Add**.
3. Enter a name and (optionally) a description for the pairing profile.
4. Configure the following options:
Enforcement Mode for Policy

You can choose one of the enforcement modes for workloads when you pair them:

- **Idle:** A state in which the VEN does not take control of the workload's iptables (Linux) or WFP (Windows), but uses workload network analysis to provide the PCE relevant details about the workload, such as the workload's IP address, operating system, and traffic flows. This snapshot is taken every four hours.



NOTE

SecureConnect is not supported on workloads in the Idle policy state. If you activate SecureConnect for a rule that applies to workloads that are in both Idle and non-Idle policy states, it could impact the traffic between these workloads.

- **Visibility:** In the Visibility Only state, the VEN inspects all open ports on a workload and reports the flow of traffic between it and other workloads to the PCE. In this state, the PCE displays the flow of traffic to and from the workload, providing insight into the datacenter and the applications running in it. No traffic is blocked in this state. This state is useful when firewall policies are not yet known. This state can be used for discovering the application traffic flows in the organization and then generating a security policy that governs required communication.
- **Selective:** Segmentation rules are enforced only for selected inbound services when a workload is within the scope of a Selective Enforcement Rule.
- **Full:** Segmentation Rules are enforced for all inbound and outbound services. Traffic that is not allowed by a Segmentation Rule is blocked.

For information about how these enforcement modes impact workload security policy, see the "Enforcement States" and "Enforcement States for Rules" topics in the *Security Policy Guide*.

You can choose one of three modes for the traffic visibility for workloads:

- **Off (no detail):** The VEN does not collect any details about traffic connections. This option provides no Illumination detail and utilizes the least amount of resources from workloads. This state is useful when you are satisfied with the rules that have been created and do not need additional overhead from observing workload communication.
- **Blocked:** The VEN only collects the blocked connection details (source IP, destination IP, protocol and source port and destination port), including all packets that were dropped. This option provides less Illumination detail but also demands fewer system resources from a workload than high detail.
- **Blocked + Allowed:** The VEN collects connection details (source IP, destination IP, protocol and source port and destination port). This applies to both allowed and blocked connections. This option provides rich Illumination detail but requires some system resources from a workload.

Enforcement Node Type

Each Pairing Profile includes information about the Enforcement Node. This section controls the type of device you can run the pairing script on, namely servers versus endpoints.

Or, you can choose not to set the type in the Pairing Profile. With this option, the VEN controls the Enforcement Node type used at activation. The node type value is found in the request from the VEN to the PCE. This node type is available as a legacy option for pairing VENs.

Label Assignment

You can specify in the pairing profile which labels you want to assign to workloads when they are paired. Labels group workloads into logical categories for use in rulesets.

The PCE provides four types of labels:

- **Role:** The role or function of a workload. In a simple two-tier application consisting of a web server and a database server, there are two roles: Web and Database.
- **Application:** The type of application the workload is supporting (for example, HRM, SAP, Finance, Storefront).
- **Environment:** The stage in the development of the application (for example, production, QA, development, staging).
- **Location:** The physical, geographic location of the workload (for example, Germany, US, Europe, Asia).

For information on creating labels, see Labels and Label Groups in the *Security Policy Guide*.

Configure Pairing Key Usage and Lifespan

You can control the usage and lifespan of the pairing key in the pairing profile.

- **Uses Per Key:** Choose if you want the key generated from this pairing profile to be used an unlimited number of times or only once.
- **Key lifespan:** Specify how long you want the pairing key to be valid, either unlimited (forever) or for a specified time frame.

You can choose from these options to define how you want the pairing profile to be used:

- **Unlimited:** This option provides a pairing script that can be used to pair as many workloads in the organization as you want. Each user in an organization is given the pairing script from this profile regardless of the workload, the application the workload is a part of, the location of the workload (data center or country), or the environment (development, testing, QA, production). Unlimited use pairing profiles can present a security risk because they never change; however, if a pairing script is stolen, a workload could be paired into your environment by an untrusted user.



IMPORTANT

Illumio recommends against configuring unlimited usage of pairing profiles from a security perspective. Instead, determine the appropriate lifespan for the pairing profile to minimize any security risk.

- **Custom Time Range:** If you do not want an unlimited use pairing profile, you can specify that the pairing profile can only be used to pair a workload one time, after which the pairing key cannot be used to pair more workloads.

Key Usage & Lifespan Requirements

The certification Key Usage requirements have changed to **CERT_DIGITAL_SIGNATURE_KEY_USAGE**, **CERT_KEY_ENCIPHERMENT_KEY_USAGE**, and **CERT_DATA_ENCIPHERMENT_KEY_USAGE**, so that the endpoint certificates can be set in the x.509 Windows environment.

Choose Command Line Overrides

For each of these Workload states, you can choose to either allow or block modifications to these settings when the pairing script is executed from the command line:

Command Line Overrides

- **Locked:** The policy state of the workloads being paired cannot be changed when the pairing script is run.
- **Unlocked:** The policy state of the workloads being paired can be changed when the pairing script is run.

Label Assignment

- **Lock Label assignment:** This option prevents a user running the pairing script from assigning labels to workloads during pairing except for what is configured with the pairing profile.
- **Allow custom Labels:** This option permits the user running the pairing script to assign labels to the workloads during pairing using this pairing profile. Selecting this option selects all the Label checkboxes. You can deselect any before saving.

5. Click **Save**.

The PCE saves the pairing profile and the page refreshes with the values you specified.

Working with a Pairing Profile

Open a pairing profile to display options to perform the following tasks.

Start/Stop Pairing

To enable or disable the pairing profile, click the pairing profile. The pairing profile details page opens and you can click **Stop Pairing** or **Start Pairing**.

Generate Key

Click **Generate Key** at the top of the page to create a unique pairing key that can be used with the pairing script. The key will not be accessible once you close the Pairing Profile details panel.

Every key that is generated under a pairing profile inherits the properties set in the pairing profile. The script can be used to pair Workloads, according to the parameters and time limits set in the pairing profile.

Delete a Pairing Profile

If you want to completely disable the pairing keys generated with a pairing profile, delete the pairing profile.

1. From the PCE web console menu, choose **Servers & Endpoints > Pairing Profiles**.
2. Select the checkbox of the pairing profiles you want to delete.
3. Click **Remove**.

All pairing keys that were associated with this pairing profile are no longer valid for pairing workloads.

The same process applies if you are instantiating new VMs in vSphere or Microsoft Azure. You can use the modified Illumio PCE pairing script for preparing your new VMs for auto scaling.

Pairing Script

When you choose to install VENs on workloads by using the VEN Library, you create the pairing profile in the PCE web console and run the pairing script on workloads. (The CLI VEN installation method bypasses the pairing script altogether. In that method, you use the native tools of the operating system to install the package, followed by the VEN CLI to activate the VEN. See [VEN Installation & Upgrade with VEN CTL \[82\]](#).)

Add Options to the Pairing Script

You can add additional pairing options to the pairing profile, such as assign labels to the workload, set the workload policy state (Command Line Overrides), and set logging levels for VEN traffic.

For the complete list of options to use with the pairing script, see [VEN Activate Command Reference \[117\]](#).

Example: Linux Pairing Script for VEN Library Installation



WARNING

On Ubuntu, elevate to a root shell before running the script (use "sudo su") to ensure that environment variables and permissions are applied correctly.

For example, if you want to add an Environment label to the workload, such as **--env Production**, include the option at the end of the pairing script as shown below.

```
rm -fr /opt/illumio_ven_data/tmp && umask 026 && mkdir -p /opt/illumio_ven_data/tmp && curl "https://example.com:8443/api/v18/software/ven/image?pair_script=pair.sh&profile_id=<pairing_profile_id>" -o /opt/illumio_ven_data/tmp/pair.sh && chmod +x /opt/illumio_ven_data/tmp/pair.sh && /opt/illumio_ven_data/tmp/pair.sh --management-server example.com:8443 --activation-code <code> --env Production
```

Example: Windows VEN Installation without the VEN Library

```
Set-ExecutionPolicy -Scope process remotesigned -Force; Start-Sleep -s 3;
(New-Object System.Net.WebClient).DownloadFile("https://repo.illum.io/Z3JldGVsbHVuZl0aGF0Y2hlcjg1dGgK/pair.ps1",
"$pwd\Pair.ps1"); .\Pair.ps1 -repo-host repo.illum.io -repo-dir Z3JldGVsbHVuZl0aGF0Y2hlcjg1dGgK/
-repo-https-port 443 -management-server pce.example.com:8443
-activation-code <code> -env Production;
Set-ExecutionPolicy -Scope process undefined -Force;
```

Running RHEL 5 Pairing Script with cURL

When using a cURL command to run a pairing script for a RHEL 5 VEN, first perform the following additional configuration.

Downgrade the Transport Layer Security (TLS) version that the PCE uses for VEN-to-PCE communications. In RHEL 5, the normal default minimum version, TLS 1.2, can not be used. Set the default minimum version to TLS 1.0 by setting the parameter `min_tls_version` to `tls1_0` in the PCE configuration file `runtime_env.yml`. For details, see [TLS Versions for Communications](#).

Update the CA certificate file on the RHEL 5 machine. Download the latest `cacert.pem` and append it to the `ca-bundle.crt` file. Use the following steps:

1. Download the latest `cacert.pem` to your RHEL 5 machine from <https://curl.se/docs/caextract.html>. If the download fails because the certificate is expired, perform the download again on a different machine with a valid certificate and then copy the certificate to the initial machine.
2. Append the certificate using the following command:

```
sudo cat /tmp/cacert.pem >> /etc/pki/tls/certs/ca-bundle.c
```
3. After changing the TLS version and the CA certificate, pair the VEN using a cURL command as described earlier in this section.

VEN Installation Using VEN Library in PCE

Installing VENs by using the VEN Library in the PCE is only available for Windows and Linux hosts. You install VENs on AIX and Solaris hosts by downloading the VEN packages for those platforms and using the VEN CTL.

About VEN Installation, Pairing, and Upgrade

These are some general considerations for installing and upgrading VENs by using the VEN Library in the PCE web console.

- The target VENs can be in any state for installation or upgrade.
- Environment variables supported by the VEN CTL are not supported with when using the VEN Library to install VENs.
- Exact time to install or upgrade a VEN depends on many factors, including the speed of the workload hardware, the speed of its network connections, and its performance load.
- Before installation or upgrade, ensure that all the workloads on which you want to install or upgrade the VEN are online and reachable from the PCE. If they are not reachable when the installation or upgrade is running, they will be skipped.

About Installing VENs by Using the VEN Library

Installing the VEN by using VEN Library in the PCE web console is a two-step process. For each workload, perform the following high-level steps:

1. In the PCE web console, generate a pairing profile. Generating a pairing profile generates a pairing script.
2. Copy that pairing script to the workload and run it.

About Pairing Workloads

Pairing is the process of installing a VEN on a workload.

When you pair a workload, you run a script that installs the VEN on the workload. The VEN then reports detailed workload information to the PCE, such as all services running on the workload, all of its open ports, details about the operating system, workload location, and more.

When you configure and then provision rules, the PCE calculates and configures policy for each paired workload.

When you pair workloads, you can choose to place those workloads in one of these policy states:

To start the Workload pairing process, you need to:

- Create a pairing profile or use the default pairing profile. See [Pairing Profiles and Scripts \[56\]](#).

- Pair workloads: Linux or Windows

Enforcement Mode for Policy

You can choose one of the enforcement modes for workloads when you pair them:

- **Idle:** A state in which the VEN does not take control of the workload's iptables (Linux) or WFP (Windows), but uses workload network analysis to provide the PCE relevant details about the workload, such as the workload's IP address, operating system, and traffic flows. This snapshot is taken every four hours.



NOTE

SecureConnect is not supported on workloads in the Idle policy state. If you activate SecureConnect for a rule that applies to workloads that are in both Idle and non-Idle policy states, it could impact the traffic between these workloads.

- **Visibility:** In the Visibility Only state, the VEN inspects all open ports on a workload and reports the flow of traffic between it and other workloads to the PCE. In this state, the PCE displays the flow of traffic to and from the workload, providing insight into the datacenter and the applications running in it. No traffic is blocked in this state. This state is useful when firewall policies are not yet known. This state can be used for discovering the application traffic flows in the organization and then generating a security policy that governs required communication.
- **Selective:** Segmentation rules are enforced only for selected inbound services when a workload is within the scope of a Selective Enforcement Rule.
- **Full:** Segmentation Rules are enforced for all inbound and outbound services. Traffic that is not allowed by a Segmentation Rule is blocked.

For information about how these enforcement modes impact workload security policy, see the "Enforcement States" and "Enforcement States for Rules" topics in the *Security Policy Guide*.

You can choose one of three modes for the traffic visibility for workloads:

- **Off (no detail):** The VEN does not collect any details about traffic connections. This option provides no Illumination detail and utilizes the least amount of resources from workloads. This state is useful when you are satisfied with the rules that have been created and do not need additional overhead from observing workload communication.

- **Blocked:** The VEN only collects the blocked connection details (source IP, destination IP, protocol and source port and destination port), including all packets that were dropped. This option provides less Illumination detail but also demands fewer system resources from a workload than high detail.
- **Blocked + Allowed:** The VEN collects connection details (source IP, destination IP, protocol and source port and destination port). This applies to both allowed and blocked connections. This option provides rich Illumination detail but requires some system resources from a workload.

VEN Package Format Changes (Windows only)

In Illumio Segmentation for Data Centers 21.2.1, the Windows VEN installer switched from MSI to EXE package format. Customers using the PCE-based VEN deployment must take an extra step for the transition. Specifically, Illumio Segmentation for Data Centers customers running older MSI-based Windows VENs must upgrade to 19.3.6+H1-VEN or 21.2.0+H2-VEN before upgrading to their VENs to 21.2.1 or a later version. Older VEN versions are not EXE package aware and cannot upgrade themselves without manual intervention on the CLI.

For the detailed steps required for this transition, see [New Windows VEN Installer Starting with 21.2.1](#), Knowledge Base Article 3561 (login required).

Pair a Windows Workload

Pairing a workload requires running the pairing script on it to install the VEN.

When you first log into the PCE web console, a default pairing profile containing a pairing script is provided so you can begin pairing workloads. You also have the option to create a new pairing profile if you want to configure your own workload pairing settings.



NOTE

When the Illumio VEN is installed on a Windows workload and paired with the PCE in the Full enforcement policy state, all Internet Group Management Protocol (IGMP) traffic will be blocked unless you add a rule to allow it. Windows servers typically use IGMP for things like Windows Internet Name Service (WINS), Windows Deployment Services (WDS), IGMP Router/Proxy mode, and Network Load Balancing (NLB) in multicast mode.

**WARNING**

Your pairing script to install a Windows VEN on a workload cannot contain colons in the values for command options. Including a colon in a option value causes VEN activation to fail. For example, including the following values in the **-role** option, causes VEN activation to fail:

```
-role "R: UNKNOWN" -app "A:UNKNOWN" -env "E: UNKNOWN"
```

Activation fails because Windows uses the colon as a special character and cannot interpret the value even when you include quotation marks around the value.

**NOTE**

You must be logged in as an Administrator user on the Windows workload to run the Illumio pairing script.

To pair a workload on Windows:

1. From the PCE web console menu, go to **Servers & Endpoints > Workloads**.
2. Click **Add > Pair Workload with Pairing Profile**.
3. From the drop-down list, select a pairing profile. The list contains the default pairing profile and the pairing profiles you've added.
To create a new pairing profile, see [Pairing Profiles and Scripts \[56\]](#).
4. From the Windows OS Pairing Script field, copy the Windows pairing script.
5. On the Windows workload you want to pair, open the Windows PowerShell as an Administrator user.
6. Paste the pairing script you copied into the PowerShell command prompt.
When the script finishes running, the following output appears:

```
PS C:\Program Files> PowerShell -Command "& {Set-
ExecutionPolicy -Scope process remotesigned -Force; Start-
Sleep -s 3; Set-Variable -Name ErrorActionPreference -Value
SilentlyContinue;
[System.Net.ServicePointManager]::SecurityProtocol=[Enum]::To
Object([System.Net.SecurityProtocolType], 3072); Set-
Variable -Name ErrorActionPreference -Value Continue; (New-
Object System.Net.WebClient).DownloadFile('https://
pce.example.com:8443/api/v18/software/ven/image?
pair_script=pair.ps1&profile_id=1',
'..\Pair.ps1'); .\Pair.ps1 -management-server
example.com:8443 -activation-code <code>;}"
```

Installing Illumio

```
-----
Setting up Illumio Repository .....
Retrieving Illumio Package .....
Installing Illumio Package .....
Validating Package Installation .....
Pairing with Illumio .....
      Pairing Status
      -----
Illumio Package installation .....SUCCESS
Pairing Configuration exists .....SUCCESS
VEN Manager Service running .....SUCCESS
Master Configuration retrieval ....SUCCESS
VEN Configuration retrieval .....SUCCESS
VEN has been SUCCESSFULLY paired with Illumio
```

```
PS C:\Program Files> cd .\Illumio\
PS C:\Program Files\Illumio> .\illumio-ven-ctl.exe status
Service venAgentMgrSvc:                Running
Service venPlatformHandlerSvc:         Running
Service venVtapServerSvc:              Running
Service venAgentMonitorSvc:            Running
Service venAgentMgrSvc:                Enabled
Service venPlatformHandlerSvc:         Enabled
Service venVtapServerSvc:              Enabled
Service venAgentMonitorSvc:            Enabled
Agent State: illuminated
```

7. To view the workload after it has finished pairing, choose **Servers & Endpoints > Workloads** from the PCE web console menu.

Unpair a Windows Workload

Unpairing is the process of uninstalling the VEN from a workload so that it no longer reports any information to the PCE and can no longer receive any

policy information. After uninstalling the VEN, the PCE will no longer maintain control over the workload.



NOTE

After you remove a workload from the PCE using the PCE web console or REST API (but not by using the VEN CTL), it remains in policy computation and can continue to appear (for example, in auto complete fields or API responses) until the VEN confirms that it has been uninstalled or a one-hour delay has passed.

Windows Unpair Options

Carefully consider the security state you want to return the Windows workload to after the VEN is uninstalled:

- **Remove Illumio policy:** (Recommended) Remove Illumio Windows Filtering Platform (WFP) filters and activate the Windows firewall.
- **Open all ports:** Uninstalls the VEN and leaves all ports on the workload open to traffic.
- **Close all ports except remote management:** Temporarily allow only RDP/3389 and WinRM/5985, 5986 until the system is rebooted.



NOTE

When you unpair a workload that uses a Windows GPO policy, the GPO policy overrides local WFP rules.

To unpair a Windows workload:

1. From the PCE web console menu, go to **Servers & Endpoints > Workloads**.
2. Click the VENs tab and then select the Windows workload(s) you want to unpair. You can select as many workloads as you want to unpair.
3. Click **Unpair**.
4. Select a final firewall status:
 - Remove Illumio Policy - DEFAULT.

- Open all ports
- Close all ports except remote management

5. Click **Unpair.**

Remove VEN Using Windows Control Panel

You can also use the Windows Control Panel Programs and Features utility to remove the VEN. When you use this utility to remove the VEN, the Windows workload is returned to the “Recommended” state.

Pair a Linux Workload

Pairing a workload requires running the pairing script on it to install the VEN.

When you first log into the PCE web console, a default pairing profile and corresponding pairing script is provided. You can use the default pairing profile to assess installing Linux VENs using the VEN Library.

Or, you can create a new pairing profile to configure your own workload pairing settings. Ultimately, you should create your own Linux pairing profile to designate your own workload pairing settings.

Before you begin, open an SSH connection to the workload you want to pair.



NOTE

You must be logged in as a user with root permissions to run the Illumio pairing script.

To pair a workload on Linux:

1. From the PCE web console menu, go to **Servers & Endpoints > Workloads**.
2. Click **Add > Pair Workload with Pairing Profile**.
3. From the drop-down list, select a pairing profile. The list contains the default pairing profile and the pairing profiles you've added.
To create a new pairing profile, see [Pairing Profiles and Scripts \[56\]](#).
4. From the Linux/Unix OS Pairing Script field, copy the Linux pairing script.
5. In a shell window on the workload you want to pair, paste the script you copied from the pairing profile.

When the script finishes running, the following output appears:

```
[root@ven-rhel tmp]# rm -fr /opt/illumio_ven_data/tmp &&
umask 026 && mkdir
-p /opt/illumio_ven_data/tmp && curl --tlsv1
"https://pce.example.com:8443/api/v18/software/ven/image?
pair_script=pair.sh&profile_id=1"
-o /opt/illumio_ven_data/tmp/pair.sh && chmod +x
/opt/illumio_ven_data/tmp/pair.sh &&
/opt/illumio_ven_data/tmp/pair.sh --management-server
pce.example.com:8443 --activation-code <code>
```

```
% Total % Received % Xferd Average Speed Time Time
Time Current Dload Upload Total Spent Left Speed
100 40891 100 40891 0 0 93526 0
--:--:-- --:--:-- --:--:-- 93572
```

Installing Illumio

```
-----
Retrieving Illumio Packages [x86_64][CentOS][7.4] .....
Validating sha256 .....
Installing Illumio Packages .....
EXPECTED_VERSION: <ven_version>
INSTALLED_VERSION: <ven_version>
Starting Illumio processes .....
Starting illumio-control:
- Environment Setting up Illumio VEN Environment:
[ OK ]
- venAgentMgr Starting venAgentMgr:
[ OK ]
- IPsec Starting IPsec: feature not enabled
[ OK ]
- venPlatformHandler Starting venPlatformHandler:
[ OK ]
- venVtapServer Starting venVtapServer:
[ OK ]
- venAgentMonitor Starting venAgentMonitor:
[ OK ]
Pairing with Illumio .....
```

Pairing Status

```
-----
Pairing Configuration exists .....SUCCESS
VEN Manager Daemon running .....SUCCESS
Master Configuration retrieval ....SUCCESS
VEN Configuration retrieval .....SUCCESS
VEN has been SUCCESSFULLY paired with Illumio
```

```
[root@ven-rhel tmp]#
```

6. To view the workload after it has finished pairing, go to **Servers & Endpoints > Workloads**.

Unpair a Linux Workload

Unpairing is the process of uninstalling the VEN from a workload so that it no longer communicates with the PCE. Once unpaired, the PCE no longer controls the workload.



NOTE

After you remove a workload from the PCE using the PCE web console or REST API (but not by using the VEN CTL), it remains in policy computation and can continue to appear (for example, in auto complete fields or API responses) until the VEN confirms that it has been uninstalled or a one-hour delay has passed.

Linux Unpair Options

Carefully consider the security state you want to return the Linux workload to after the VEN is uninstalled.

- **Remove Illumio policy:** (Recommended) Revert firewall rules to the state previous to pairing.
- **Open all ports:** Uninstalls the VEN and leaves all ports on the workload open to traffic.
- **Close all ports except remote management:** Temporarily allow only SSH/22 traffic until system is rebooted.



NOTE

If the Workload you are unpairing is offline during the unpairing process, the Workload may still appear in the Workloads list in the PCE web console, even though the Workload has been unpaired. The unpaired Workload will be removed within 30-35 minutes.

To unpair a Linux workload:

1. From the PCE web console menu, go to **Servers & Endpoints > Workloads**.
2. Click the VENs tab and then select the Linux workload(s) you want to unpair. You can select as many workloads as you want to unpair.
3. Click **Unpair**.
4. Select final firewall status:
 - Remove Illumio Policy - DEFAULT.
 - Open all ports
 - Close all ports except remote management
5. Click **Unpair**.

Ignored Interfaces

You can now set interfaces from “managed” to “ignored” in the PCE web console. Use this option when you want the workload to ignore visibility and enforcement specific interfaces; for example, on the interconnected interfaces of database clusters, such as Oracle RAC. You can set one or more interfaces to “ignored” during pairing. Using this setting causes the first downloaded firewall to ignore those interfaces. An ignored interface is not be included in the policy configuration and traffic will flow uninterrupted through it without any change in latency. You can see which interfaces are marked as “ignored” on the Workload details page.



IMPORTANT

Illumio recommends that you designate all private (non-routable) interfaces as ignored interfaces.

To set an interface to Ignored:

1. From the PCE web console menu, go to **Servers & Endpoints > Workloads**.
2. Click the VENs tab and then click the link of the workload that has the interface you want to ignore.
3. Click **Edit**.
4. In the Network Interfaces section, go to the **PCE Action** drop-down menu and change the interface to **Ignored**.
5. Click **Save**.

**NOTE**

After you set an interface to Ignored, it will not be included in policy configuration provided by the PCE and traffic will continue to flow uninterrupted through that interface.

VEN Installation Troubleshooting

This topic provides information about troubleshooting VEN installation issues you might encounter when using the PCE web console. For general troubleshooting, see "VEN Troubleshooting" in VEN Administration Guide.

Troubleshoot Pairing Errors

If you execute the pairing script and it fails, the system displays an error message indicating failure. One such error may be:

```
Unable to retrieve Illumio VEN configuration from management
server
illumio.example.com:8443 in a timely fashion.
```

If the script fails to install the VEN, the following output displays:

Pairing Status

```
-----
Secureware Package installation .....SUCCESS
Pairing Configuration exists .....SUCCESS
VEN Manager Daemon running .....SUCCESS
Master Configuration retrieval .....SUCCESS
VEN Configuration retrieval .....FAILED
2014-01-23T18:43:50Z AgentManager 13088
Verify activation code and retry pairing
Workload has FAILED pairing with Illumio
```

Possible Causes of Failure

Failures may be caused by a problem with the VEN communicating with PCE, or by exceeding the hard license limit, or by an attempt to install an Endpoint VEN or Server VEN on an operating system that doesn't support that VEN type.

- The hard license limit has been exceeded.

- An invalid pairing profile is being used.
- The wrong pairing script was copied and run on the workload.
- In the pairing script section, the pairing profile contains invalid pairing keys.
- The pairing profile was disabled.
- The use limit for the pairing key has been exceeded.
- The pairing key has expired.
- Wrong Enforcement Node Type setting (**Servers & Endpoints > Pairing Profiles**):
 - The pairing profile is for a server node (not an endpoint node) but the profile includes the parameter `-endpoint true`.
 - The pairing profile used to activate a macOS VEN is set to Server VEN.
 - The pairing profile used to activate a Linux VEN is set to Endpoint VEN.

Remediate

- Check the pairing profile in the PCE web console to try to verify the cause of the failure.
- Check the PCE Logs and VEN Logs.
- The VEN Library is available in the PCE web console. You can download VEN software bundle and also view the dependencies and supported OS versions.

VEN Upgrade Using VEN Library in PCE

You can use your PCE cluster as a centralized mechanism for upgrading VENs in your organization.

From the 20.2.0 release on, you can upgrade one or more VENs by using the PCE web console. In the PCE UI available since release 23.5, go to **Servers & Endpoints > Workloads > VENs**, select a VEN, and then click **Upgrade**.

You can also use the Illumio Segmentation for Data Centers REST API to upgrade (but not install) VENs. See the *REST API Developer Guide* for information.



NOTE

Before you use this feature, you must set up the VEN Library in the PCE. See [VEN Library Setup in the PCE \[41\]](#) for information.

If you are an Illumio Cloud customer, Illumio Operations set up the VEN Library in the PCE.



IMPORTANT

For Illumio Segmentation for Data Centers Cloud customers, the VEN Library only provides the option to upgrade to VENs that Illumio designated as a Long Term Support (LTS) release. See [Versions and Compatibility](#) in the Illumio Support Portal (login required) for information.

About VEN Upgrade

You can upgrade all VENs, upgrade a selected subset of VENs, or upgrade all VENs that match a set of filters. After you confirm an upgrade from the PCE web console, the VEN will download the new VEN image from the PCE and upgrade itself. The upgrade on the workload host only takes on average a few minutes.

If the VEN does not successfully upgrade within a certain amount of time (this timeout is configurable), the upgrade will time out and the PCE will put the VEN in a warning state. However, in most cases, the upgrade will complete within this window. To clear this warning, just start another upgrade on the VEN.

You can upgrade up to 25,000 VENs in a single PCE region. Selecting this large a number of VENs in one upgrade will result in some CPU and memory spikes and increased network bandwidth, because the VENs will be communicating with the PCE to request new firewalls and downloading new software versions.

The VEN upgrade feature includes upgrade validation. The PCE will validate that the VEN upgrade path is allowed and that the version of the VEN you are upgrading to is compatible with the version of the PCE. If you attempt

to upgrade VENs to a version incompatible with the PCE or Illumio does not support that upgrade path, the PCE web console provides feedback on which VENs can be upgraded.

VEN Package Format Changes

Starting with the 21.2.1 release, the Windows VEN installer switched from MSI to EXE package format. This package format change primarily affects Illumio Core On-Premises customers running older MSI-based Windows VENs. For information about using the VEN Library in the PCE to install Windows VENs on workloads, see [Pair a Windows Workload](#).

Prerequisites and Limitations for VEN Upgrade

Prerequisites

- The PCE must be a version 20.2.0 or later.
- The VENs selected for upgrade must be version 18.2.0 or later.
- The VEN Library must be set up in the PCE. See [Upload VEN Software Bundle into PCE VEN Library \[46\]](#) for information.
- The VEN upgrade compatibility matrix must be installed on the PCE. See [Upload VEN Upgrade Compatibility Matrix \[45\]](#) for information.

Limitations

- You must update the VENs in your environment using a supported upgrade path.
- You cannot upgrade a VEN to a later version than the PCE's current version.
- If a VEN has been installed with custom RPM installation options, you cannot use the VEN upgrade feature to upgrade it.

For example, you cannot upgrade VENs installed with a custom `--prefix` option because options like that aren't persisted when a VEN is upgraded from the PCE, and the VEN will be installed in the default directory. This outcome might cause data loss or operational issues with the VEN, depending on your environment.

- You cannot use the feature to downgrade a VEN to an earlier version.
- Using the PCE CLI to upgrade VENs is no longer supported in Illumio Segmentation for Data Centers 21.2.0 and later releases.
- If you installed the VEN with the VEN CTL and packaging CLI and customized installation options (such as, a custom installation directory or alternate VEN user), you cannot later upgrade the VEN by using the VEN Library in the PCE. You must upgrade the VEN using the workload's OS package upgrade process.

- If you are upgrading a Windows VEN while it's in Full Enforcement and the Illumio ilowfp driver provided in the upgrade differs from the driver that's already installed, the driver will be unloaded and reloaded, which will block all new connections during the upgrade (existing connections are not affected). **To avoid network disruption during production hours, Illumio recommends that you upgrade VENs only during a maintenance window.**

VEN Package Format Changes (Windows only)

Starting with the Illumio Segmentation for Data Centers 21.2.1 release, the Windows VEN installer switched from MSI to EXE package format. This package format change affects Illumio Core On-Premises customers running older MSI-based Windows VENs.

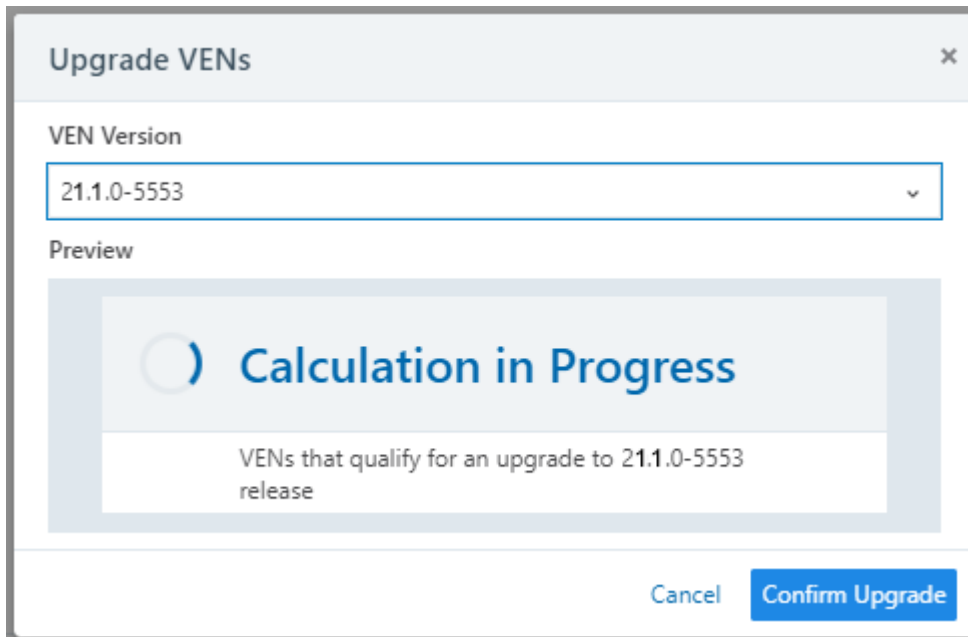
Customers using the VEN Library in the PCE to install or upgrade VENs must take an extra step for the transition. Specifically, Illumio Segmentation for Data Centers customers running older MSI-based Windows VENs must upgrade to Illumio Segmentation for Data Centers 19.3.6+H1-VEN or 21.2.0+H2-VEN before upgrading to their VENs to 21.2.1 or a later version. The 21.2.0+H2-VEN release contained the necessary VEN changes to handle the transition in the VEN packaging from MSI to EXE.

For the detailed steps required for this transition, see [New Windows VEN Installer Starting with 21.2.1](#), Knowledge Base Article 3561 (login required).

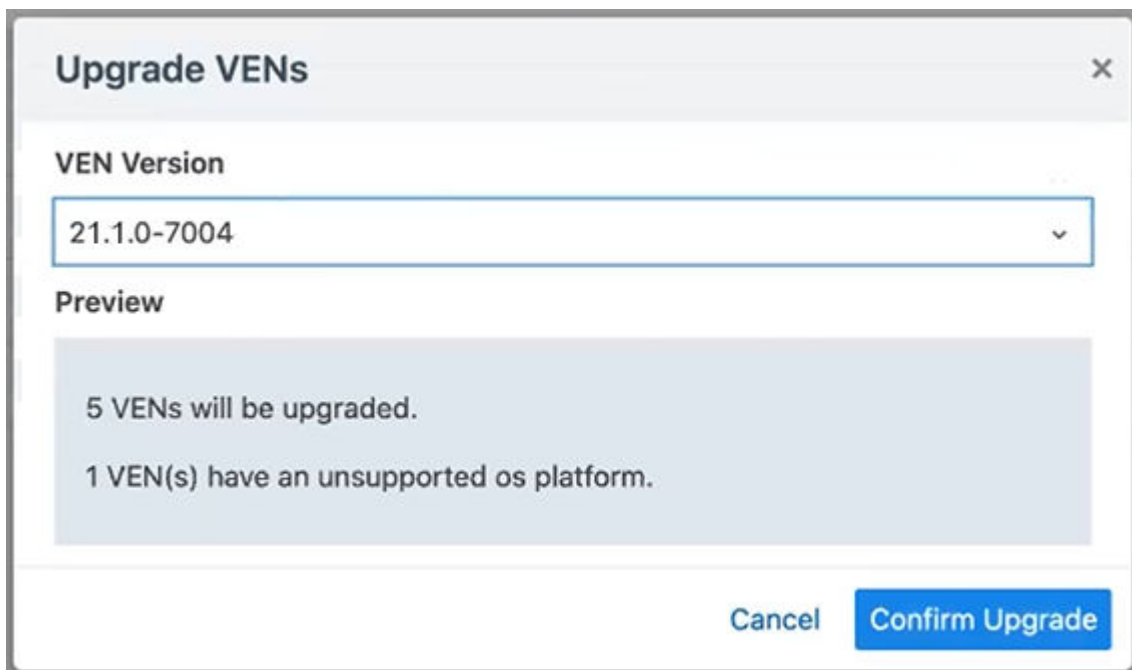
Upgrade All VENs to the Current Version

When you select "Upgrade All," the PCE upgrades all your deployed VENs and not just the VENs that appear in that page of the list (for example, you have 10 pages in the list, the VENs in all 10 pages are upgraded.)

1. Go to **Servers & Endpoints > Workloads**
2. Click the **VENs** tab.
3. From the **Upgrade** drop-down menu, choose **Upgrade All**.
The Upgrade VENs dialog box appears. By default, the most current VEN version in the VEN Library is selected.
4. (Optional) To upgrade to a version other than the current version, select the VEN version from the drop-down menu.
The PCE calculates the scope of the upgrade.



When the calculation is complete, the dialog box refreshes and informs you which VENs in your environment can be upgraded to the selected version. In this example, one VEN has an unsupported OS platform because it's a VEN running on Solaris, which cannot be upgraded using this feature in this release.



5. (Optional) Specify how much time the VENs have to successfully upgrade by entering a value and units of time in **VEN Upgrade Expiration**. The timeout must be between 15 minutes and 180 days. For server VENs, the recommended upgrade timeout is 1 day. For endpoint VENs, the recommended timeout is 7 days. After the expiration time passes, the PCE will no longer instruct the VEN to upgrade, and the VEN will be in a warning state.

6. Click **Confirm Upgrade**.

Upgrade Selective VENs

When upgrading, you can use all the filters available in the VEN list to manage the upgrade. For example, you could filter VENs by the location label to only upgrade VENs in a specific datacenter or you can filter the VENs in your environment by their operating systems and upgrade just those VENs.

You can filter and upgrade VENs based on specific conditions. For example, you can locate the VENs that weren't upgraded in your first attempt to upgrade all VENs to the default version; for example, some VENs might have been offline when you ran your first upgrade and the upgrade timed out for those VENs but succeeded on all the others in your environment.

You can also filter and upgrade VENs based on their Health status in the PCE.

VENs can have one of three Health statuses that you can filter for:

- **Healthy:** The VEN has no Health conditions.
- **Warning:** The VEN has one or more Warning conditions.
- **Error:** The VEN has one or more Error conditions or has both Error and Warning conditions.

☐	Status	Health	↕ Name	Version
☐	✓ Active	✓	client1	20.2.0-7004
☐	✓ Active	✓	client2	20.2.0-7004
☐	✓ Active	✗	client3	19.3.3-6329
☐	● Stopped	✗	client4	20.2.0-7004
☐	✓ Active	⚠	server2	19.3.3-6329
☐	✓ Active	✓	server3	20.2.0-7004

View VEN Upgrade Events

Upgrading VENs using the PCE web console or the REST API generate events that you can view in the Events page.

For example, go to **Troubleshooting** > **Events** and look for events indicating when upgrades succeeded and when they failed. If you use the **Upgrade All** option to upgrade large numbers of VENs in your environment, the PCE aggregates the event for `agent.upgrade_requested`; however, the PCE still generates a separate event for each successful upgrade.

VEN Installation & Upgrade with VEN CTL

The following topics describe how to use packages and the VEN CTL to install the VEN on hosts in your environment. To perform the tasks in this section, you must log into the Illumio Support portal to download the VEN software to your local environment.

Windows: Install and Upgrade with CLI and VEN CTL

This section discusses installing and upgrading the VEN for Windows by using packaging technology commands and the VEN CTL.

With the Windows VEN MSI, you have the option of activating (pairing) the VEN either during installation or after installation.

Windows VEN Installation Directories

By default, the Windows VEN is installed in the following directories:

- Installation: `C:\Program Files\Illumio`
- Data: `C:\ProgramData\Illumio`

VEN Package Format Changes

Starting with the Illumio Segmentation for Data Centers 21.2.1 release, the Windows VEN installer switched from MSI to EXE package format. This package format change primarily affects Illumio Core On-Premises customers running older MSI-based Windows VENs.

For information about using the VEN Library in the PCE to install Windows VENs on workloads, see [Pair a Windows Workload \[67\]](#).

Install the Windows VEN Using EXE Package

Starting with the version 21.2.1, the Windows VEN installer format changed from an MSI package to an EXE bundle. The installation file is now executable

and `msiexec.exe` is no longer used to install the Windows VEN in Illumio Segmentation for Data Centers 21.2.1 and later releases.

Command Line Interface

The Windows VEN installer supports following command line options:

- `/install`
- `/uninstall`
- `/quiet`
Disables the interactive installer so that you don't respond to installation prompts.
- `/passive`
Still displays a minimal user interface but does not provide installation prompts.
- `/norestart`
Suppresses any attempts at restart.
- `/log`
Logs installation information to a specific file.

The following installation command lines show how to install the VEN EXE bundle and activate the VEN after installation. See [Windows VEN Activation After Installation \[83\]](#).

Quiet VEN Installation Example

```
illumio-ven-24.2.20-2091.win.x64.exe /install /quiet /  
norestart /log VEN_install.log
```

Quiet VEN Installation with Custom Directories Example

```
illumio-ven-24.2.20-2091.win.x64.exe /install /quiet /  
norestart /log VEN_install.log INSTALLFOLDER="c:\illumio\ven"  
DATAFOLDER="c:\illumio\ven_data"
```

Interactive VEN Installation Example

```
illumio-ven-24.2.20-2091.win.x64.exe /install /norestart /log  
VEN_install.log
```

Windows VEN Activation After Installation

To activate the Windows VEN after installation, run the following command:

```
\illumio-ven-ctl.exe activate -activation-code
<activation_code> -management-server <pce_fqdn:pce_portnumber>
<activation_options>
```

Windows VEN Activation Options

You have several activation options you can set while pairing. You can set the workload policy state and apply labels at the time of activation.

This example shows how to activate a Windows workload with the following options:

- Set the VEN policy state to illuminated with no traffic logging: `-log_traffic false`
- Set the role as Web service: `-role Web`
- Set the application to HRM: `-app HRM`
- Set the environment to development: `-env Dev`
- Set the location of the VEN to New York City: `-loc NYC`

Kerberos for Windows VEN-to-PCE Authentication

To enable Kerberos authentication at installation, set the command-line variable `KERBEROS_PCE_SPN` on the installation program. Use the following value for this variable:

```
illumio-device-auth/<fqdn_of_your_pce>
```

Where:

- The literal `illumio-device-auth/` is required.
- `fqdn_of_your_pce` is the fully qualified domain name (FQDN) of your PCE.

Example:

```
illumio-ven-<ven_version>.<os_platform>.exe /install
KERBEROS_PCE_SPN=illumio-device-auth/pce.example.com
```

Activation with Kerberos

On the `illumio-ven-ctl --activate` or in the pairing script, do *not* use any option that sets a label. That is, do not use the `--env`, `--loc`, `--role`, or `--app` options. Labels should be set in the PCE web console. See the *Security Policy Guide* for information.

After installation with the command-line variable, when you activate the VEN, a message similar to the following is displayed:

```
# illumio-ven-ctl activate

    Activating Illumio
...
Enabling Kerberos Authentication .....
...
```

Windows VEN Upgrade



IMPORTANT

Illumio strongly recommends that you upgrade VENs only during maintenance windows.



NOTE

If the VEN was activated prior to the upgrade, it does not need to be activated again after the upgrade completes.

To upgrade the VEN, run this command:

```
illumio-ven-<ven_version>.<os_platform>.exe /install
```

Windows VEN Uninstallation Using CLI

To uninstall the Windows VEN by using the VEN CTL, see "Deactivate and Unpair VENs" in the VEN Administration Guide.

Offline VEN During Unpairing

If the workload you are unpairing is offline, the workload might still appear in the workloads list in the PCE web console, even though the workload has been unpaired. The unpaired workload is removed from the PCE web console within 30-35 minutes.

Alternative: Remove Windows VEN Using Control Panel

You can also use the Windows Control Panel Programs and Features utility to remove the VEN. When you remove the Windows VEN with the Windows Control Panel, the VEN unpairs the workload with the **Unpair and remove Illumio policy** option. This method removes any current Illumio policy and activates the Windows firewall.

Linux: Install and Upgrade with CLI and VEN CTL

This section discusses installing and upgrading the VEN for Linux by using packaging technology commands and the VEN CTL.

- Installing the VEN on Linux relies on native package management commands:
 - For Debian and Ubuntu (referred to as “Debian”): **dpkg**
 - For Red Hat and CentOS (referred to as “Red Hat”): **rpm**
- Root access on the workload is required for installation of the Linux VEN.
- Some of the optional installation features in the RPM are not available with the Debian package. These cases are marked in section titles below with “RPM only.”

About iptables Versions for Red Hat and CentOS

Red Hat Version 6 VENs

If the iptables version already on the workload is older than iptables version 1.4.7-16 or ipsets is older than version 6.11-4, the VEN installation process installs more recent versions of iptables and ipset, including libmnl. These unmodified distribution files (RPMs) are packaged with the VEN itself and installed in `/opt/illumio_ven/etc/extras`.

About Red Hat 8 Support and nftables

In 18.2.x, the VEN supported iptables. In 19.3.0 and later releases, the VEN supports nftables, which is the default host-based firewall used by Red Hat 8. Supporting nftables, which is used by Red Hat 8, simplifies rule writing and allows developers to write fewer rules and do so much more efficiently. The new support for nftables does not change VEN functionality or the VEN feature set because the underlying net filter capabilities are the same. Support for nftables provides the following usability enhancements for the VEN:

- **Simpler syntax:** Uses a simpler syntax, which is very similar to TCP dump.

- **Combined rules:** Rather than write 2 rules for every enforcement point, you can combine them into a single rule; such as combining multiple ports into a single rule, and IPv4 and IPv6 into a single rule.
- **Multiple actions:** A single rule can have multiple actions, such as LOG and DROP.
- **Built-in tracing:** Includes built-in support for named sets. To use lists or sets with `iptables`, you need to install `ipset`. `nftables` has integrated set support and can be used more naturally within the configuration.

Notes:

- Native support for `nftables` in Red Hat 8 does not change the VEN installation or upgrade process; if you've written installation scripts, they don't require updates.
- `nftables` does not impact Firewall Coexistence and it is still supported.
- Using the `nfttrace` tool on Illumio created tables is not supported because it requires specific filtering rules.
- `nftables` support does not impact the PCE; viewing a workload that is running Red Hat 8 in the PCE web console does not change. You can view all the workload details. Creating policy for workloads running Red Hat 8 does not change.



IMPORTANT

In 19.3.0 and later releases, the VEN continues to support earlier versions of Red Hat with no changes. For the complete list of Red Hat versions supported by the VEN by release, see [OS Support and Package Dependencies](#) on the Illumio Support portal.

Linux Default Installation Directories

The Linux VEN is installed into two directories by default:

- `/opt/illumio_ven`
- `/opt/illumio_ven_data`

Directory Ownership Pre- and Post-activation

- All directories are created with mode 0750.
- Post-activation user/group `ilo-ven:ilo-ven` allows processes running as that user to write to the VEN installation directory and VEN data directory.

- At installation, you can set various environment variable to override default settings. See the "Linux Installation with Environment Variables" topic.

VEN Pack- age For- mat	Path	Default Pre-Activa- tion Owner	Default Post-Activa- tion Owner
RPM	/opt/illumio_ven	root:ilo-ven	root:ilo-ven
	/opt/illumio_ven_data	ilo-ven:ilo-ven	ilo-ven:ilo-ven
DPKG	/opt/illumio_ven	root:ilo-ven	root:ilo-ven
	/opt/illumio_ven_data	root:ilo-ven	ilo-ven:ilo-ven

Dependency Check for Certificates

If your PCE-to-VEN SSL certificate was signed by a private CA and the signing CA's credentials have already been added to the workload's trusted certificate store, the ca-certificates package is not needed. To install the VEN without the dependency check, follow these examples:

- Red Hat: `rpm -vh -nodeps illumio-ven-<ven_version>.<os_platform>.rpm`
- Debian: `dpkg --ignore-depends=illumio-ven-<ven_version>.<os_platform>.deb`

Linux Installation with Environment Variables

The following table lists VEN environment variables that you can set for the package installation on Linux.



NOTE

Before installation, set any of the following environment variables.

Variable	Description
VEN_ACTIVATION_CODE	The activation code; see Example: Linux Environment Variables [92] and About the Command Options .
VEN_DATA_DIR	Directory where the <code>illumio_ven_data</code> directory is created.
	Installation options: - Initial installation: During your initial installation of the VEN from either a Debian or RPM package, you can set this variable to install the VEN data directory to either the default location or a different location. - Upgrade: This differs depending on whether you're upgrading from a Debian or an RPM package: RPM package: When upgrading the VEN from an RPM package, you can set this variable to install the VEN data directory to either the default directory or to a different directory such as <code>/var/opt/illumio_ven_data</code>. NOTE: If you specify a directory that is different from the previous installation directory, all data is moved to that new directory. Debian package: Always install Debian upgrade to the initial data directory: When upgrading the VEN from a Debian package, make sure to set this variable to install the VEN to the same data directory to which it was initially installed. Specifying a different data directory for Debian package upgrade (that is, attempting to move the data once already installed) is not supported. Don't use PCE for upgrade if non-default data directory was used: If the Debian package was initially installed in a non-default data directory, don't use the PCE to upgrade it because PCE installation always installs to the default data directory, which is not supported in this case (see above bullet).
VEN_DISABLE_MONITOR_RESTART	Disable the Linux VEN monitoring cronjob (which differs from the VEN Agent Monitor process). See Disable Agent Monitor cronjob [93] .
VEN_INSTALL_ACTION	Activate or prepare the VEN during installation. Valid values: activate: Used to automatically activate the VEN during installation instead of requiring <code>illumio-ven-ctl activate</code> to be invoked as a subsequent step. To use this value, make sure the <code>VEN_MANAGEMENT_SERVER</code> and <code>VEN_ACTIVATION_CODE</code> variables are also set. Requires an activation code on the <code>illumio-ven-ctl</code> control script or set within the <code>VEN_ACTIVATION_CODE</code> environment variable. prepare: This is necessary when preparing the VEN as part of a golden disk image or virtual machine template as it ensures that the VEN activates automatically during first use but not before. For more information, see Prepare Golden Master for Workload Installation.
VEN_KERBEROS_WORKLOAD_SPN	See Kerberos for Linux VEN-to-PCE Authentication [90] .
VEN_KERBEROS_MANAGER_SPN	See Kerberos for Linux VEN-to-PCE Authentication [90] .

Variable	Description
AGEMENT_SERV- ER_SPN	
VEN_KER- BEROS_LI- BRARY_PATH	See Kerberos for Linux VEN-to-PCE Authentication [90] .
VEN_MANAGE- MENT_SERVER	The FQDN of the PCE server and its port. For example, <code>pce.example.com:8443</code> .
VEN_NON- PRIV_UID	If <code>VEN_NONPRIV_USER</code> is not set, create the <code>ilo-ven</code> user with the specified UID.
VEN_NON- PRIV_GID	If <code>VEN_NONPRIV_USER</code> is not set, create the <code>ilo-ven</code> group with the specified GID.
VEN_NON- PRIV_USER	The name of an existing user account for use by non-privileged VEN functions such as logging. The group account is determined automatically by the specified user's primary group. The group name is the specified user's primary existing group name. If you don't want the VEN to use a non-privileged account, specify "root" as the existing user account. • If <code>VEN_NONPRIV_USER</code> is set, any values for <code>VEN_NONPRIV_UID</code> and <code>VEN_NONPRIV_GID</code> are ignored. • Conversely, if <code>VEN_NONPRIV_USER</code> is not set, any values for <code>VEN_NONPRIV_UID</code> and <code>VEN_NONPRIV_GID</code> take effect.
ILLUMIO_RUN- TIME_ENV	The absolute path to a non-default <code>runtime_env.yml</code> file. If you use this environment variable, make sure to configure it with the same value whenever the VEN is executed after installation. This environment variable is unique because it's relevant during and after installation.

Kerberos for Linux VEN-to-PCE Authentication

The `illumio-ven-ctl` command does not have any options for Kerberos, but when you activate the VEN with `illumio-ven-ctl`, at installation it honors the Kerberos values that have been set in environment variables.

Before installing the Linux VEN, set the following environment variables.

Environment variable	Value	Notes
VEN_KERBEROS_WORKLOAD_SPN	(Optional) See notes. The default host principal set in the Kerberos keytab file. The SPN of the server for renewing Ticket Granting Tickets (TGT) for Linux workloads. Format: <pre>host/fqdn_of_ven@REALM</pre> Where: • The literal host/ is required. • fqdn_of_ven is the FQDN of the workload where the VEN is installed. • @REALM is optional. If not specified, the default realm is used.	A workload might have more than one host principal in its keytab file, one of them the principal needed for PCE authentication. In this case VEN_KERBEROS_WORKLOAD_SPN must be set so that the VEN software knows which principal to use to acquire a TGT. The VEN relies on the default Kerberos keytab file, typically /etc/krb5.keytab. Therefore, the host SPN for PCE authentication must be added to the default keytab file. Before deploying Kerberos authentication, you can use kinit to verify that a TGT for the workload's SPN can be acquired: <pre>kinit -k</pre> If the command is successful, use klist to verify the TGT has been acquired for the correct host SPN. If the default SPN is not what you want for PCE authentication, use the following command to verify that you can reach the desired SPN: <pre>kinit -k host/fqdn_of_ven@REALM</pre> If the command is successful, set VEN_KERBEROS_WORKLOAD_SPN to host/fqdn_of_ven@REALM.
VEN_KERBEROS_MANAGEMENT_SERVER_SPN	SPN for the PCE Example: illumio-device-auth/pce.example.com	GSSAPI Authentication
VEN_KERBEROS_LIBRARY_PATH	Absolute path to libgssapi_krb5.so Example: /usr/lib	The exact path can vary by type of Linux OS. If libgssapi_krb5.so does not exist on your system, create a symlink of the same name to point to the libgssapi_krb5.so.n file, where n is the number on the actual installed

Environment variable	Value	Notes
		shared object library on your workload, like <code>libgssapi_krb5.so.2</code>

Activation with Kerberos

On the `illumio-ven-ctl --activate` or in the pairing script, do *not* use any option that sets a label. That is, do not use the `--env`, `--loc`, `--role`, or `--app` options. Labels should be set in the PCE web console. See the *Security Policy Guide* for information.

After installation with the command-line variable, when you activate the VEN, a message similar to the following is displayed:

```
# illumio-ven-ctl activate

    Activating Illumio
...
Enabling Kerberos Authentication .....
...
```

Example: Linux Environment Variables

To activate the VEN during installation, set the following environment variables before running the installation command.

- `VEN_MANAGEMENT_SERVER`
- `VEN_ACTIVATION_CODE`
- `VEN_INSTALL_ACTION`

For example, to activate a VEN during installation of a VEN package:

```
# VEN_MANAGEMENT_SERVER=pce.example.com:8443
VEN_INSTALL_ACTION=activate
VEN_ACTIVATION_CODE=<activation_code> rpm -ivh illumio-ven-
<ven_version>.<os_platform>.rpm
```

Or

```
# VEN_MANAGEMENT_SERVER=pce.example.com:8443
VEN_INSTALL_ACTION=activate
VEN_ACTIVATION_CODE=<activation_code> dpkg -i illumio-ven-
<ven_version>.<os_platform>.deb
```

Change Default Name of User at Installation

The default username for the VEN installation is **ilo-ven**. With the package installation, you can specify an environment variable to set a different, existing username to override this default. The group name is the specified user's primary group and does not need to be specified.

```
# VEN_NONPRIV_USER=desired_existing_username rpm -ivh illumio-ven-<ven_version>.<os_platform>.rpm
```

Or

```
# VEN_NONPRIV_USER=desired_existing_username dpkg -i illumio-ven-<ven_version>.<os_platform>.deb
```

Disable Agent Monitor cronjob

You can disable the agent monitor cronjob before or after VEN installation. When failing to hook into existing **init** systems like **systemd**, **upstart**, or **SysV**, the Linux VEN installation creates a cronjob to check the VEN agent monitor process and restart it if necessary. This cronjob runs every 10 minutes.

Some organizations prefer to rely on their own VEN agent monitoring processes. The Illumio-supplied VEN-checking cronjob might create logs whose size you consider excessive or whose frequency is not right for your needs.

To disable the Linux VEN monitoring cronjob before installation:

Set the following environment variable:

```
export VEN_DISABLE_MONITOR_RESTART=true
```

Any value other than **true** does not have any effect.

To modify or disable the Linux VEN monitoring cronjob after installation:

You have several options:

- Edit your crontab to decrease the cronjob's frequency.
- In your crontab, completely comment out the VEN agent monitoring cronjob.

To substitute your own VEN agent monitor checking process, consider the following points:

- Rely on your own organization's standard mechanisms for monitoring processes.
- Make sure your monitoring restarts the VEN if necessary.
 - Do not restart only the VEN agent monitoring process. Restart the entire VEN:

```
# illumio-ven-ctl restart
```

- Be sure that your monitoring process has sufficient permissions to restart the VEN.

Linux VEN Activation After Installation

To activate the VEN after installation, use the `illumio-ven-ctl` control script with the `activate` argument to activate the workload and pair the VEN with the PCE.

At a minimum, to activate the VEN using the VEN control script, you need the hostname or IP address of the PCE, an activation code (called a pairing key in the PCE web console) generated from a pairing profile, and any other required options, such as the workload policy state, label assignment, and workload name. For example, the following command shows how to activate the VEN that places the workload into the Illumination policy state (`--mode`).

```
# /opt/illumio_ven/illumio-ven-ctl activate --management-server pce.example.com:8443  
--activation-code <activation_code>
```

Upgrade Linux VEN Using CLI



IMPORTANT

Illumio strongly recommends that you upgrade VENs only during maintenance windows.



NOTE

If the VEN was activated prior to the upgrade, it does not need to be activated again after the upgrade completes.

Custom Username, Installation Directory, VEN Data Directory

If you installed the VEN with your own username, for upgrade you need to specify that same username with the `VEN_NONPRIV_USER` environment variable.

If you previously installed the VEN to non-default installation (RPM only) and data directories with environment variables, specify the same values before upgrade.

See [Linux Installation with Environment Variables \[88\]](#).

RPM Upgrade

```
# rpm -Uvh illumio-ven-<ven_version>.<os_platform>.rpm
```



IMPORTANT

If you don't specify the `VEN_DATA_DIR` environment variable and the `--prefix` option during the initial RPM installation, the operation creates the `illumio_ven` and `illumio_ven_data` directories in the default `/opt` directory.

When upgrading the VEN from an RPM package, you can set the `VEN_DATA_DIR` environment variable and `--prefix` option to install the VEN to either the same directory used for the initial installation or a different directory (such as `/var/opt/illumio_ven_data`). **NOTE:** If you specify a different directory during an RPM upgrade, all data is moved to that directory.

This example also includes a custom username that was used during VEN installation. It shows how to: (1) independently relocate both the VEN installation directory as well as the VEN data directory for RPM packages, and (2) specify a non-default user account for non-privileged functions.

For example, if you originally installed the VEN with this type of command:

```
# VEN_NONPRIV_USER=ven_software_acct VEN_DATA_DIR=/custom/data/
path/one rpm -ivh <orig-illumio-ven-pkg>.rpm --prefix/custom/
install/path/one
```

... you can upgrade the VEN by issuing the following command:

```
# VEN_NONPRIV_USER=ven_software_acct VEN_DATA_DIR=/custom/data/
path/two rpm -Uvh <orig-illumio-ven-pkg>.rpm --prefix /custom/
install/path/two
```

... where the VEN installation directory (specified by the `rpm --prefix` option) and the VEN data directory (specified by the `VEN_DATA_DIR` environment variable) are different settings that can be relocated independently for RPM packages. If the `--prefix` specified for an upgrade doesn't match the existing VEN installation directory, the VEN software is moved to the new `--prefix` as part of the upgrade. Similarly, if the `VEN_DATA_DIR` specified for an upgrade doesn't match the existing VEN data directory, the VEN data is moved to the new `VEN_DATA_DIR` as part of the upgrade.

Note that you can't change the `VEN_NONPRIV_USER` after installation. If you specified a non-default `ven_software_acct` for installation, you must use the same account for upgrade.

Debian Upgrade

```
# dpkg -i illumio-ven-<ven_version>.<os_platform>.deb
```



IMPORTANT

If you don't specify the `VEN_DATA_DIR` environment variable during the initial VEN installation, the `illumio_ven_data` directory is created in the default `/opt` directory.

However, if you specify a **non-default** directory for the initial Debian package installation and later upgrade the VEN without specifying that same directory, the VEN will not upgrade to that directory. Therefore, if you specified a non-default data directory during the initial installation, you need to continue to specify the same non-default data directory during upgrades. Unlike RPM, moving data to a different directory when upgrading a VEN from a Debian package is not supported.

This example also includes a custom username that was used during VEN installation.



IMPORTANT

Take careful note of the following:

- Debian package **installation directory**: You can't change the VEN installation directory to a non-default location because **dpkg** does not implement a **--prefix** option like RPM.
- Debian package **data directory**: You can change the VEN data directory to a non-default location but be aware that if you do, you can't change it again; you must always specify the same non-default data directory for all upgrades going forward.
- Debian **VEN_NONPRIV_USER**: You can't change the **VEN_NONPRIV_USER** after installation. If you use a non-default user account for installation, you must always use the same user account for upgrades going forward.

For example, if you originally installed the VEN with this type of command:

```
# VEN_NONPRIV_USER=ven_install_username VEN_DATA_DIR=/opt/  
my_data_dir dpkg -i <orig-illumio-ven-pkg>.deb
```

... you must continue to specify the same **VEN_DATA_DIR** for all subsequent upgrades:

```
# VEN_NONPRIV_USER=ven_software_acct VEN_DATA_DIR=/custom/data/  
path/one dpkg -i <orig-illumio-ven-pkg>.deb
```

Uninstall Linux VEN Using CLI

Unpair a Linux VEN before uninstalling it. See "Deactivate and Unpair VENs" in the VEN Administration Guide.

SUSE Linux: If a SUSE workload is unpaired in the Full enforcement policy state, the uninstallation might not complete when the workload does not have rules that allow it to connect to SUSE repositories. To avoid this issue, change the policy state to Visibility before unpairing the VEN. For more information see "Workload Policy States" in the VEN Administration Guide.

Uninstall the VEN

Security Implications: Production applications on this workload could break because after uninstalling the VEN this workload will no longer allow any connections to it other than SSH on port 22.

To uninstall the VEN, see "Deactivate and Unpair VENs" in the VEN Administration Guide.

AIX: Install and Upgrade with CLI and VEN CTL

The following topic describes how to install and upgrade the AIX VEN by using packaging technology commands and the VEN CTL.

Limitations and Considerations

General

- AIX 5.3 is not supported.
See [VEN OS Support and Package Dependencies](#) for the list of supported operating systems for AIX VENs.
- AIX native IPsec is not supported while the VEN is installed.
- The AIX VEN does not support SecureConnect and SecureConnect Gateway.
- The following directories must be present on the AIX host or the AIX VEN installation will fail. These directories are commonly present on AIX hosts.
 - `/var/lib`
 - `/var/log`
- By default, the AIX VEN is installed in the following directories:
 - `/opt/illumio_ven`
 - `/opt/illumio_ven_data`

Installing the AIX VEN in a custom directory is not supported. Do not change the default installation directory for the AIX VEN or the AIX VEN installation will fail.

Configuration Options for CA Bundle and CA DIR



NOTE

The options `trusted_ca_bundle` and `trusted_ca_dir` in `runtime_env.yml` are no longer used.

Core 22.2 introduced new options for configuring the CA bundle or CA directory it uses to verify the PCE TLS certificate. You can specify the new options in the `/etc/default/illumio-agent` file. The options are:

- **TRUSTED_CA_BUNDLE** can point to a specific certificate bundle. For example:

```
TRUSTED_CA_BUNDLE=/etc/ssl/certs/ca-bundle.crt
```

- **TRUSTED_CA_DIR** can point to a directory containing certificates. For example:

```
TRUSTED_CA_DIR=/etc/ssl/certs
```

IPFilter

- Illumio provides a custom IPFilter package for managing the packet filtering rules. Before you install the AIX VEN, install the Illumio-provided IPFilter package.



CAUTION

You must use the Illumio customized IPFilter package with the AIX VEN. Do not use IBM's IPFilter package or the AIX VEN will not function correctly.

- Avoid any changes to packet filtering with **genfilt**, **mkfilt** and other such network tools. Do not perform any such operation while VEN software is installed.
- The AIX system firewall's state table limit is 65,536 entries. When that limit is reached, IPFilter drops packets. If you anticipate a high number of network connections, configure higher limits in the IPFilter state table. See "Tuning the IPFilter State Table (AIX/Solaris)" in the VEN Administration Guide.

Change Default Username Before Installation

Before installing the VEN on AIX, you can set an environment variable to change the username that owns the non-privileged portions of the installed software. The privileged portions of the installed software are always owned by root, and the software can only be run as root.

Environment Variable	Description
VEN_NONPRIV_USER	Existing username to override the default username ilo-ven . The group name of the specified user is the primary existing group name of the specified user.

Boot Scripts Installed at VEN Installation

As part of installation, the VEN creates RC scripts (“run commands”) in `/etc/rc3.d` to start the VEN at boot.

Illumio Support for IPFilter

IBM has discontinued support and development of IPFilter and has put IPFilter on GitHub as an open source project. Consequently, Illumio provides its own version of IPFilter for the Illumio AIX VEN version 17.1.2 and later.



NOTE

Illumio supports *only* its provided version of IPFilter. We do not support installing the AIX VEN with the OEM version of IPFilter. Before installing the AIX VEN, you must install the Illumio-provided IPFilter package.

Illumio supports its version of IPFilter in the following ways:

- The Illumio IPFilter package will not be made public. Permissive licensing of IPFilter does not require that modifications of open source software be made public.
- Illumio can provide IPFilter source code patches for bug-fixes and improvements on request to your Illumio representative.

Download AIX VEN Tar File and IPFilter Package

Download the VEN Packages tar file from the Illumio Support site. The tar file contains the AIX VEN in Backup File Format (BFF) format.

Additionally, you must download the Illumio-provided IPFilter package from the Illumio Support site. The VEN package does *not* contain the required Illumio-provided IPFilter package.

To download the AIX VEN files:

1. Go to the Illumio Support site (login required).
2. Select **Software** > **Download** under the VEN section > the VEN version. The Download VEN page appears. The page contains two tables: “VEN” and “Other”

3. In the VEN Packages row of the VEN table, click the filename for the VEN tar file.
4. In the Other table, click the AIX IPFilter filename (**ipfl.5.3.0.5002.bff**) to download the Illumio-supported IPFilter 5.3.0.5002 package.

Upgrade to Illumio IPFilter

This procedure describes how to perform either of these tasks:

- Upgrade from the IBM IPFilter package to the current Illumio IPFilter 5.3.0.5002 package
- Upgrade from the previous version of the Illumio IPFilter 5.3.0.5001 package to the current 5.3.0.5002 package

The steps in this procedure apply to both of these IPFilter upgrades except for step #4, which applies only when upgrading from IBM IPFilter to the Illumio IPFilter 5.3.0.5002 package.

To upgrade to Illumio IPFilter:

1. Download the Illumio-supplied IPFilter package. See [Download AIX VEN Tar File and IPFilter Package \[100\]](#).
2. Stop the VEN if it's running:

```
illumio-ven-ctl stop
```

3. Stop the IBM ipf kernel extension using the following command:

```
/lib/methods/cfg_ipf -u
```



NOTE

In some cases, there may be multiple instances of ipf. Confirm there are no running instances by running the above **stop** command again until it returns **no such device**.

If the command fails with the error **Device Busy**, before continuing these steps, reboot the system.

4. **[For Upgrades from IBM IPFilter Only]** If IBM iFIX or ipfl is installed on the host, uninstall them. (In an earlier release, Illumio had recommended installation of some iFIXes.)

**NOTE**

Depending on your installed AIX version, you might have installed iFIX version IV89793s5a or IV89793s3a. Remove the version corresponding to the version already installed on your AIX server. Neither version is needed and must be removed with the appropriate **emgr** command. The following command uninstalls only version IV89793s5a.

```
emgr -r -L IV89793s5a.161102.epkg.Z
```

5. Change directory to where you downloaded the AIX VEN and the IPFilter package.
6. Upgrade the version of IPFilter with the Illumio custom IPFilter:

```
inutoc . && installp -acYd . ipfl
```

7. Proceed to installing or upgrading the AIX VEN.

Install the AIX VEN

1. Download theN package from the Illumio Support site. See [Download AIX VEN Tar File and IPFilter Package \[100\]](#).
2. Log in to the AIX host and become superuser.
3. If necessary, upgrade IPFilter on the AIX host to Illumio's custom IPFilter. See [Upgrade IBM IPFilter to Illumio IPFilter \[101\]](#).

**IMPORTANT**

You must upgrade to Illumio's custom IPFilter before installing the AIX VEN.

4. Copy your trusted root CA certificate in the following directory with a filename **ca-bundle.crt**. This path must be exactly as shown.

```
/var/ssl/certs/ca-bundle.crt
```

5. Make **ca-bundle.crt** world-readable.

```
# chmod 644 /var/ssl/certs/ca-bundle.crt
```

6. Install the VEN package on the AIX host by entering the following commands, where **path_to_bff_file** is the directory where you copied the AIX VEN BFF file.

```
# inutoc <path_to_bff_file>
# installp -acXgd path_to_bff_file illumio-ven
```

AIX VEN installation is complete. The next step is [Activate AIX VEN After Installation \[103\]](#).

Optional: If you anticipate a high number of network connections, you can configure higher limits in the IPFilter state table. See "Tuning the IPFilter State Table (AIX/Solaris)" in the VEN Administration Guide.

Activate AIX VEN After Installation



IMPORTANT

If you're using the GRE and IPIP protocols, before activating the VEN on AIX, edit the file in the `/etc/protocols` directory to support the GRE and IPIP protocols. If the GRE and IPIP protocol lines are commented out, un-comment them.

After installing the VEN package on the AIX host, activate the VEN. Use the Illumio VEN control script (`illumio-ven-ctl`) with the **activate** option to activate the workload and pair the AIX VEN with the PCE.

At a minimum, to activate the AIX VEN using the VEN control script, you need the hostname or IP address of the PCE, an activation code (called a pairing key in the PCE web console) generated from a pairing profile, and any other available options, such as the workload policy state, label assignment, workload name, and more.

For information about obtaining an activation code from the PCE web console, see "Pairing Profiles" in the Security Policy Guide.

```
# /opt/illumio_ven/illumio-ven-ctl activate --management-server <pce_fqdn:port> --activation-code <code>
```

See the following example command:

```
# /opt/illumio_ven/illumio-ven-ctl activate --management-server pce.example.com:8443 --activation-code <code>
```

Upgrade the AIX VEN



IMPORTANT

Illumio strongly recommends that you upgrade VENs only during maintenance windows.



NOTE

If the VEN was activated prior to the upgrade, it does not need to be activated again after the upgrade completes.

For the supported upgrade paths for the AIX VEN, see [Upgrade VEN](#) on the Illumio Support portal (login required).

1. Download the new version of the VEN package from the Illumio Support site. See [AIX Tar File and IPFilter Package \[100\]](#).
2. If necessary, upgrade the Illumio-supported IPFilter package to version 5.3.0.5002.

If you are upgrading the AIX VEN from an earlier release, such as 17.1.x, you might be running the Illumio-supported AIX IPFilter package version 5.3.0.5000. See [Upgrade to Illumio IPFilter \[101\]](#).

3. Stop the VEN if it's running:

```
illumio-ven-ctl stop
```

4. Upgrade the VEN package on the AIX host by entering the following commands, where **path_to_bff_file** is the directory where you copied the new version of the AIX VEN BFF file.

```
# inutoc <path_to_bff_file>
# installp -acXgd path_to_bff_file illumio-ven
```

Solaris: Install and Upgrade with CLI and VEN CTL

The following topic describes how to install the Solaris VEN by using packaging technology commands and the VEN CTL.

The VEN for Solaris supports two different Solaris machine architectures: SPARC and x86_64. The installation and upgrade steps for both machine architectures are identical but each architecture uses its own VEN package file.

Limitations and Requirements

General

- In Illumio Segmentation for Data Centers 19.3.1 and later releases, the Solaris VEN supports Solaris zones.

The Solaris operating system (OS) is architected such that all Solaris zones share the same underlying kernel networking stack of the global zone. Due to OS limitations, the VEN cannot reliably manage non-global zones that are configured with **ip-type: shared** and only non-global zones with **ip-type: exclusive** are fully supported. If you are using the VEN on workloads with a Solaris non-global zone and require additional assistance, contact your Illumio Customer representative.

- By default, the Solaris VEN is installed in the following directories:

- `/opt/illumio_ven`
- `/opt/illumio_ven_data`

Installing the Solaris VEN in a custom directory is not supported. Do not change the default installation directory for the Solaris VEN or the Solaris VEN installation will fail.

- Installing or activating the Solaris VEN on a workload running an LDAP client can take longer than on other workloads without an LDAP client.
- The Solaris VEN requires the bash shell and the Solaris XCU4 utilities (POSIX-compliant tools) be installed on the Solaris host. Verify that both are installed on the host. The XCU4 utilities are installed using the Solaris SUNWxcu4 package, typically in the `/usr/xpg4/bin/` directory. See the Oracle Solaris documentation for information about installing the XCU4 utilities.

IP Filter (Solaris version 11.3 and earlier)

- Avoid making any changes to packet filtering with Packet Filter. Do not use Packet Filter while VEN software is installed.
- The Solaris system's firewall state table limit is 65,536 entries. When that limit is reached, IP Filter drops packets. If you anticipate a high number of network connections, configure higher limits in the IP Filter state table. See "Tuning the IP Filter State Table for Solaris and AIX workloads" in VEN Administration Guide.



IMPORTANT

In Solaris 11.4, Packet Filter replaces IP Filter. When installing the VEN on Solaris 11.4, Illumio only supports Packet Filter. IP Filter is not supported in branded zones starting with Solaris 11.4.

Change Default Username

You can set an environment variable to change the username that owns the non-privileged portions of the installed software. The privileged portions of the installed software are always owned by root, and the software can only be run as root.

Environment Variable	Description
VEN_NONPRIV_USER	Existing username to override the default username <code>ilo-ven</code> . The group name of the specified user is the primary existing group name of the specified user.

You can reset this environment variable in your customized Solaris Response file or at a prompt during interactive installation.

About Solaris 11.4 Support

Prior to 11.4, Solaris used IP Filter as the firewall. In Solaris 11.4, Packet Filter is the only supported firewall.

The following details apply to Solaris 11.4 support by the VEN:

- Support for Solaris 11.4 does not change the VEN installation or upgrade process on Solaris workloads; if you've written installation scripts, they don't require updates. Package installation remains the same for 11.4 as for earlier supported versions of Solaris.
- Packet Filter support does not impact the PCE; viewing a workload that is running Solaris 11.4 in the PCE web console does not change. You can view all the workload details. Creating policy for workloads running Solaris 11.4 does not change.
- Packet Filter does not support customizable table sizes. However, state tables in Solaris 11.4 use a 1 million state table size.



IMPORTANT

For the complete list of all Solaris versions supported by the VEN in this release, see [OS Support and Package Dependencies](#) on the Illumio Support portal.

About the Solaris Response and Admin Files

In addition to the Solaris VEN, the VEN package includes two files to help with VEN installation on Solaris hosts: the Solaris Administration and Response files. For more information about these files, see [Avoiding User Interaction When Adding Packages \(pkgadd\)](#) in the *Oracle Solaris Administration Guide*.

Solaris Administration File

The Solaris Administration contains information about how the VEN installation or upgrade should proceed on the Solaris host. To perform a non-interactive VEN installation or upgrade (the VEN installation script will *not* prompt for settings when it runs), you must customize the Administration file.

In addition to settings, the file contains commented-out instructions for changing the settings.



CAUTION

If you choose to provide custom values in the Administration file, you must delete these commented-out lines or the VEN installation or upgrade will fail. Commented-out lines in a Solaris Administration file are not supported with the Solaris **pkgadd** command.

```
# This file is used in case of upgradation
# instance=ask allows multiple instance of the same software
# to be installed
# and hence the UPDATE flag is passed to us in procedural
# scripts of IPS.
mail=
instance=ask
partial=ask
runlevel=ask
# Require that our dependencies are met when installing.
idepend=quit
# However, if someone tries to uninstall us but another
# package depends on us,
# we should just warn them & ask if they want to proceed
# anyway.
rdepend=ask
space=ask
setuid=ask
conflict=ask
action=nocheck
networktimeout=60
networkretries=3
authentication=quit
keystore=/var/sadm/security
proxy=
basedir=default
```

Solaris Response File

The VEN package includes a template for the Solaris Response file. The template contains the environment variables that you can set when installing and upgrading the Solaris VEN.

In addition to the available environment variables, the template contains commented-out instructions for providing custom values for variables.



CAUTION

If you choose to provide custom values in the Response file, you must delete these commented-out lines or the VEN installation or upgrade will fail. Commented-out lines in a Solaris Response file are not supported with the Solaris **pkgadd** command.

```

#Parameter : VEN_NONPRIV_USER
#Type : String
#Description : VEN non-privileged user. If unspecified
(VEN_NONPRIV_USER=""), then the default account "ilo-ven" is
used. If that account does not exist on the system, it is
created automatically. If specified (VEN_NONPRIV_USER="foo"),
the provided account is used. If that account does not exist
on the system, then the installer fails. All non-root-owned
files that the VEN creates are owned by that user and that
user's primary group. For further information about this
feature, refer to the Illumio VEN deployment documentation.
VEN_NONPRIV_USER=""
#Parameter : VEN_PKI_CLIENT_CERT
#Type : String
#Description : PKI (public key infrastructure) authentication
certificate. Use with VEN_PKI_CLIENT_KEY. When specified,
these fields are appended to runtime_env.yml. Then, they
may be used to activate the VEN. I.e., ``$ /opt/illumio_ven/
illumio-ven-ctl activate`` uses these fields to authenticate
the VEN with the PCE.
VEN_PKI_CLIENT_CERT=""
VEN_PKI_CLIENT_KEY=""
VEN_KERBEROS_MANAGEMENT_SERVER_SPN=""
VEN_KERBEROS_LIBRARY_PATH=""
VEN_ACTIVATION_CODE=""
VEN_MANAGEMENT_SERVER=""
VEN_INSTALL_ACTION=""
#Parameter : VEN_NO_SUSPEND
#Type : Number
#Description : Custom setting to disable suspend. 1 - disable,
0 - default
VEN_NO_SUSPEND=0

```

If you leave the Response file as is, the VEN installation script uses the default values for these environment variables by displaying them at the prompts during an interactive installation or silently during installation (because you're using the Administration file).

To Use Customized Response and Administration Files

To customize the Response and Administration file for your Solaris VEN installation or upgrade, perform these steps:

1. Extract the Response and Administration files from the VEN package. See [Installation Preparation \[110\]](#) for information.
2. Copy and rename the files from the following directories for your machine architecture:

```
sudo bash
# cp illumio-ven/root/opt/illumio_ven/etc/templates/admin
/tmp/admin.custom
# cp illumio-ven/root/opt/illumio_ven/etc/templates/response
/tmp/response.custom
```

This command example suggests copying the files to the `/tmp` directory; however, you can copy the files to any directory.

3. Edit the files to set your own values. See [Solaris Response File \[108\]](#) and [Solaris Administration File \[107\]](#) for the requirements when customizing these files.

Installation Preparation

1. Download the VEN package from the Illumio Support portal. See "Obtain the VEN Packages" for information.

The Solaris VEN software downloaded from the Illumio Support portal is provided as a compressed tar archive file that contains one file for each of the supported Solaris machine architectures: SPARC and x86_64:

- `illumio-ven-<ven_version>.sol5.sparc.pkg`
- `illumio-ven-<ven_version>.sol5.i386.pkg`

2. Extract the Solaris VEN software:

```
# gunzip illumio-ven-<ven_version>.<architecture>.pkg.tgz
# tar -xvf illumio-ven-<ven_version>.<architecture>.pkg.tar
```

3. Install your trusted root CA certificate in the following directory with this exact specified filename:

```
/etc/certs/ca-certificates.crt
```

Ways to Install the Solaris VEN

You can install the Solaris VEN by specifying the Solaris `pkgadd` command and running the interactive VEN installation script; referred to as a "basic" installation in this topic.

Alternatively, you can use a Solaris Administration file or Solaris Response file (or both) to perform a non-interactive installation or set custom installation values (or both); referred to as an "advanced" installation in this topic.

Behavior During Each Type of Installation

SOLARIS FILES	BEHAVIOR
None	The VEN installation script performs a basic installation wherein you are prompted to set installation values or accept the default values.
Administration file only	The VEN installation script runs without prompting you for installation values (non-interactive) and uses only the default installation values; you cannot specify custom values.
Response file only	The VEN installation script launches an interactive installation; however, the prompts contain your custom values or the default value if not set. Press Enter to accept.
Both Administration and Response files	The VEN installation script runs without prompting you for installation values (non-interactive) and uses your custom values or the default value if not set. This method is the most automated of all the ways to install the Solaris VEN.

Basic Installation

1. Complete the tasks to prepare for Solaris VEN installation. See [Installation Preparation \[110\]](#) for information.
2. To install the Solaris VEN, enter the following command:

```
# pkgadd -d . illumio-ven-<ven_version>.<architecture>.pkg
```



IMPORTANT

When installing the Solaris VEN, enter the correct package for the Solaris machine architecture (SPARC or x86_64) you want to install:

- `illumio-ven-<ven_version>.sol5.sparc.pkg`
- `illumio-ven-<ven_version>.sol5.i386.pkg`

The interactive VEN installation scripts starts.

3. Provide custom VEN installation and configuration values at the prompts or accept the defaults.

Solaris VEN installation is complete. The next step is to activate the Solaris VEN.

Advanced Installation

1. Complete the tasks to prepare for Solaris VEN installation. See [Installation Preparation \[110\]](#) for information.

2. Prepare the Solaris Administration and Response files for use in the installation. See [To Use Customized Response and Administration Files \[109\]](#) for information.
3. Enter the following command to perform a customized, non-interactive installation:

```
# pkgadd -d . -a pkgadd -a /tmp/admin.custom -r /tmp/  
response.custom illumio-ven-<ven_version>.<architecture>.pkg
```

Where the paths to the customized Administration and the Response files are the same ones you created when you extracted and copied them locally or to a network share. See [To Use Customized Response and Administration Files \[109\]](#) for information.



IMPORTANT

When installing the Solaris VEN, enter the correct package for the Solaris machine architecture (SPARC or x86_64) you want to install:

- `illumio-ven-<ven_version>.sol5.sparc.pkg`
- `illumio-ven-<ven_version>.sol5.i386.pkg`

Solaris VEN installation is complete. The next step is to activate the Solaris VEN.

Quick Reference for VEN Installation on a Solaris Server

This quick installation is provided in Illumio Segmentation for Data Centers Knowledge Base article at [VEN Install on a Solaris Server](#).

Activate a Solaris VEN After Installation

After installing the VEN package on the Solaris host, activate the VEN with the Illumio VEN CTL (`illumio-ven-ctl`). The `--activate` option activates the workload and pairs the Solaris VEN with the PCE.



TIP

You can activate the Solaris VEN by using the VEN CTL or by specifying the values in the appropriate environment variables in the Solaris Response file. See [Solaris Response File \[108\]](#) for information.

**NOTE**

Activating the Solaris VEN on a workload that is running an LDAP client can take longer than on workloads not using LDAP.

At a minimum, to activate the Solaris VEN using the VEN CTL, you need the hostname or IP address of the PCE, an activation code (called a pairing key in the PCE web console) generated from a pairing profile, and any other available options, such as the workload policy state, label assignment, workload name, and more.

The following example shows how to activate the VEN and set its policy state to Illuminated:

```
# /opt/illumio_ven/illumio-ven-ctl activate --activation-code  
<code> --management-server <fqdn:port>
```

Upgrade the Solaris VEN

**IMPORTANT**

Illumio strongly recommends that you upgrade VENs only during maintenance windows.

**NOTE**

If the VEN was activated prior to the upgrade, it does not need to be activated again after the upgrade completes.

Illumio supports both Solaris machine architectures: SPARC and x86_64. The upgrade steps for both machine architectures are identical but each architecture uses its own VEN package file.

For the supported upgrade paths for the Solaris VEN, see [Upgrade VEN](#) on the Illumio Support portal (login required).

Requirement: To upgrade the Solaris VEN, you must perform the upgrade by using the Solaris Administration file. Using the Response file with the upgrade is optional.

1. Download the new version of the VEN package from the Illumio Support site. See "Obtain the VEN Packages" for information.
2. Extract the Solaris VEN software. See [Installation Preparation \[110\]](#) for information.
3. Prepare the Solaris Administration file for the upgrade. See [To Use Customized Response and Administration Files \[109\]](#) for information.

At a minimum, you must set the following values in the Administration file for the upgrade:

```
mail=
instance=overwrite
conflict=nocheck
action=nocheck
```

4. Stop the VEN if it's running:

```
illumio-ven-ctl stop
```

5. Enter the following command to perform a non-interactive installation:

```
# pkgadd -d . -a /tmp/admin.custom illumio-ven-
<ven_version>.<architecture>.pkg
```

Where the path to the customized Administration file is the same one you created when you extracted and copied it locally or to a network share.



NOTE

If you also need to customize settings for the upgrade, use a customized Response file for the upgrade and include the **-r** argument in the upgrade command; for example: **-r /tmp/response.custom**. See [Solaris Response File \[108\]](#) for information.

Uninstall the Solaris VEN



IMPORTANT

Before you uninstall the VEN software from a Solaris workload, unpair the VEN running on the workload from the PCE. See "Deactivate and Unpair VENs" in VEN Administration Guide.

1. Enter the following commands to uninstall the VEN:

```
sudo bash
# cd /tmp
# pkgrm illumio-ven
```

2. The following command output and prompts appear in the command window. Respond to the required prompts to uninstall the VEN:

```
The following package is currently installed:
  illumio-ven  illumio-ven
                (i386) <ven_version>.sol5.i386

Do you want to remove this package? [y,n,?,q] y

## Removing installed package instance <illumio-ven>

This package contains scripts which will be executed with
super-user
permission during the process of removing this package.

Do you want to continue with the removal of this package
[y,n,?,q] y
## Verifying package <illumio-ven> dependencies in global
zone
## Processing package information.
## Executing preremove script.
VEN_DATA : /opt/illumio_ven_data
Stopping venAgentMonitor: ...done
Stopping venAgentMgr:      ...done
Stopping venVtapServer:    ...done
Stopping venPlatformHandler: ...done
## Removing pathnames in class <none>
.
.
.
## Executing postremove script.
## Updating system information.

Removal of <illumio-ven> was successful.
```

Deploy an Illumio Endpoint VEN as a private app using Intune

This document describes how to use Microsoft Intune to easily deploy the Illumio Endpoint VEN (Virtual Enforcement Node) to remote Intune-managed

Windows devices at scale. The process creates a Win32 app package that becomes a Company Portal app. The portal serves as a private app repository for your organization.

Installing Endpoint VENs on your Intune-managed Windows devices allows you to visualize your traffic and secure your corporate assets through policy-driven network segmentation.

STEP 1: Download resources and create folders

1. Download Illumio Windows VEN release 24.2.x or later from the [Illumio Support Portal](#). (For example, `illumio-ven-24.2.10-1053.win.x64.exe`.)
2. Download the **Microsoft Win32 Content Prep Tool** from GitHub. This tool is required to convert the installer into a format suitable for Intune deployment.
3. Create three folders on your local machine:
 - **Source Folder:** Containing the Windows VEN.
 - **Input Folder:** For installation files.
 - **Output Folder:** Where you'll save the `.intunewin` file.

STEP 2: Create the .intunewin file

1. Place the Windows VEN (.exe) into the **Input** folder.
2. From a command line, navigate to the directory where you downloaded the Win32 Content Prep Tool.
3. Run the Prep tool by issuing the following command:

```
.\IntuneWinAppUtil.exe -c <path-to-input-folder> -s <Windows-  
VEN>.exe -o <path-to-output-folder>
```

STEP 3: Configure the application in Intune

1. Sign in to Microsoft Endpoint Manager.
2. Go to the Microsoft Intune Admin Center.
3. Navigate to **Apps > Windows > Add > Windows app (Win32)**.
4. Upload the `.intunewin` file you created in [STEP 2 \[116\]](#).
5. Enter application information:
 - **Name:** Illumio Windows VEN
 - **Description:** Virtual Enforcement Node from Illumio
 - **Publisher:** Illumio
6. Set the Installation Command:

```
<Windows-VEN>.exe /install /quiet /norestart
/log C:\Windows\temp\IllumioEndpointInstall.log
MANAGEMENT_SERVER=<PCE_URL> ACTIVATION_CODE=<ACTIVATION_CODE>
```

7. Set the Uninstallation Command:

```
"C:\Program Files\Illumio\illumio-ven-ctl.exe" unpair saved
```

8. Configure Detection Rules:
 - **Rule Type:** File
 - **Path:** %ProgramFiles%\Illumio\illumio-ven-ctl.exe
 - **File or Folder Exists:** Yes
9. Set Requirements (e.g., Windows 10 and later).

STEP 4: Assign the application to Windows devices

1. In the Assignments section, select the groups of devices for VEN deployment.
2. Set installation intent (required for automatic installation).

Reference

This section contains useful reference information for installing and upgrading VENs on workloads in your organization.

VEN Activate Command Reference

The following topic describes the commands for activating the VENs either during or after installation, and the ways that you can configure the VEN during activation.

About the Command Options

If you are activating with a PCE that has a pairing profile configured to block changes to policy state (the `illumio-ven-ctl` option `--mode`) or label assignment (the `illumio-ven-ctl` options `--env`, `--loc`, `--role`, `--app`), you must not use these options on these blocked configurations or the activation will fail.

**WARNING**

When you use the VEN CTL or a pairing script to install a Windows VEN on a workload, you cannot include colons in the values for the options. Including a colon in a command value causes VEN activation to fail. For example, including the following values in the **-role** option, causes VEN activation to fail:

```
-role "R: UNKNOWN" -app "A:UNKNOWN" -env "E: UNKNOWN"
```

Activation fails because Windows uses the colon as a special character and cannot interpret the value even when you include quotation marks around the value.

Description of the activate Command Options

The options and arguments are the same for Windows and Unix (Linux, Solaris, and Solaris), except the options with two dashes on Unix should be replaced with a single dash on Windows (for example, **--loc** on Linux should be replaced with **-loc** on Windows).

**NOTE**

The following options are optional unless noted in the description.

Option	Arguments	Description
activation-file	<activation_file>	REQUIRED: --activation-file is used to specify activation-code and management-server from a file rather than on the command line. The option activation-code is used for sensitive and confidential information. If you enter it on a command line, a non-root user on the machine could see it by running "ps -ef" or similar commands. For greater security, put the activation-code in the activation-file and specify the file path to the activate command.
activation-code -a	<activation_code>	REQUIRED: Inputs the activation code of the VEN into the pairing script. This code is auto-generated by the pairing profile. Activation code: one-time use or unlimited use In the PCE web console, you can specify that an activation code is for one-time use or for unlimited uses. Be sure you have generated the correct type for your needs. Do not use a single one-time use activation code for more than one workload. Example: --activation-code 1234567890abcdef
management-server -m	<PCE_FQDN:port> <IPAddress:port>	REQUIRED: Sets the domain name or IP address and port of the host where the VEN can retrieve master configuration information. Example: --management-server mypce.example.com:8443
name -n	<server_friendly_name>	Sets a friendly name that will be used for this workload when it appears in the PCE web console. Example: --name "Web Server 1"
env	<environment_label>	Assigns an Environment label for this workload. Example: --env Production
loc	<location_label>	Assigns a Location label for this workload. Example: --loc "US"
role	<role_label>	Assigns a Role label for this workload.

Option	Arguments	Description
app	<application_label>	Example: <code>--role "Dev Group"</code> Assigns an Application label for this workload. Example: <code>--app "Web Service"</code>
proxy_server	<proxy-string>	[Linux, Solaris, AIX only] You only need to specify this option when configuring a proxy server for a Linux, Solaris, or AIX workload. Windows automatically detects a proxy server; therefore, you do not need to specify this option when installing a VEN on a Windows workload. For information about configuring a proxy server, see VEN Proxy Support [32].
log-traffic	true false	Enables or disables traffic logging. If not specified, logging is set to true by default. Default: true Interacts with the visibility-level option. See Allowable Combinations of log-traffic and visibility-level [121].
mode	illuminated enforced idle	Sets the policy state for the workload. For an explanation of the various states, see "Workload Policy States" in the VEN Administration Guide.
enforcement_mode	full visibility_only selective idle	Default: visibility_only Enables the new selective mode for the VEN.
visibility-level	flow_summary flow_drops flow_off	Default: flow_summary Defines the extent of the data the VEN collects and reports to the PCE from a workload in the Full enforcement or Visibility policy states, so you can control resource demands on workloads. The higher levels of detail are useful for visualizing traffic flows in greater detail in the Illumination map inside the PCE web console. Interacts with the <code>--log-traffic</code> option. See Allowable Combinations of log-traffic and visibility-level [121].

visibility-level Arguments

Argument	Value in Policy States	Notes
<code>flow_summary</code>	Included in all policy states	Default. The VEN collects traffic connection details for both <i>allowed</i> and <i>blocked</i> connections: source and destination IP address and port and protocol. This argument creates traffic links in the Illumination map and is typically used initially after installing the VEN to determine the full scope of potential policy impact on the workload.
<code>flow_drops</code>	Valid only in full policy state	The VEN collects connection details only for <i>blocked</i> traffic: source and destination IP address and port and protocol. This argument produces less detail for Illumination but demands fewer workload system resources than <code>flow_summary</code>.
<code>flow_off</code>	Valid in all policy states	The VEN does not collect any details about traffic connections. This option produces no details for the Illumination map but requires the fewest number of workload resources. Useful when you are satisfied with policy rules and do not need additional detail.

Allowable Combinations of log-traffic and visibility-level

The following rules apply to using the `log-traffic` and `visibility-level` options together with the `activate` command:

- The `visibility-level` argument takes precedence over the `log-traffic` argument.
- `visibility-level flow_off` and `--log-traffic true` is an invalid combination.
- `visibility-level flow_drops` is invalid in Illuminated policy state.

VEN Modes in Illumio Segmentation for Data Centers 20.2.0 and later

In Illumio Segmentation for Data Centers 20.2.0, Illumio introduced a new feature called Selective Enforcement. For an explanation of how this feature changed policy functionality in the release, see "Selective Enforcement" in *What's New in This Release, 20.2.0*.

**NOTE**

This change to the VEN modes affects the VEN in Illumio Segmentation for Data Centers 20.2.0 and later releases.

In particular, the feature changed the workload policy states. To further understand how the policy state changed between releases, see the "Workload Policy State" and "Workload Enforcement States" topics in the Security Policy Guide.

The changes to policy states in 20.2.0 impacted the VEN **mode** option that you specify with the **activate** command in the following ways.

**CAUTION**

Do not use both the **mode** and **enforcement_mode** options together on the command line because you could specify contradictory options. Specify one or the other. To specify the new Selective Enforcement option, enter the **enforcement_mode selective** option and argument.

- Adds a new option for the **activate** command: **enforcement full|visibility_only|selective|idle**
- Retains the **mode** option from the previous release for backward compatibility
- The arguments for the **mode** option map to those for the **enforcement_mode** option in this way:
 - **illuminated** maps to **visibility_only**
 - **enforced** maps to **full**
 - **idle** is the same in both options
 - **enforcement_mode** option adds the **selective** argument

Use the **selective** argument to set the VEN mode as Selective Enforcement state. For information about using Selective Enforcement in policy, see "Enforcement Modes for Rules" in the Security Policy Guide.
- The **visibility-level** option and argument are unchanged in Illumio Segmentation for Data Centers 20.2.0 and later releases; however, the Selective Enforcement feature separated the workload policy visibility state from the policy enforcement state; in particular, the PCE web console has

separate drop-down menus (**Enforcement** and **Visibility**) in the **Workloads** page for these two options.

The arguments for the **visibility-level** option map to the new visibility states in 20.2.0 and later releases in this way:

- **flow_off** maps to Off
- **flow_drops** maps to Blocked
- **flow_summary** maps to Blocked + Allowed

VEN Compatibility Check

This topic explains how to use the VEN Compatibility Check feature after installing VENs on workloads.

About Compatibility Checks

When you pair a 19.3.x VEN or later release in the Idle state or change the VEN state to Idle, the VEN performs several compatibility checks and sends the results to the PCE. This process occurs every 24 hours and checks whether the preexisting workload state will have issues when the VEN is moved out of the Idle state.

After reviewing the results of the VEN Compatibility Check, you can determine if the VEN is ready to be moved out of the Idle state or whether you need to resolve any detected issues, such as backing up any system firewall rules.



NOTE

The VEN Compatibility Check is performed per-workload and is available only for VENs in the Idle state, not Visibility, Selective, or Full states. If a workload reverts from any of these states to the Idle policy state, the VEN Compatibility Check is performed.

All detected issues are categorized as:

- **Red:** Major incompatibility detected
- **Yellow:** A potential incompatibility detected
- **Green:** No major incompatibilities detected

The Compatibility Check results are displayed in the PCE web console. To view the results:

1. Go to Workloads and click the name of the workload whose Compatibility Report you want to see.



NOTE

The workload must be in the Idle state for the Compatibility Report tab to appear (**Edit** > **Enforcement** > **Idle**, and then click **Save**.)

2. Click the **Compatibility Report** tab.

If no incompatibilities are detected on the VEN, the page displays "No Data to Display."

After viewing the results, you can export the report as a text file by clicking **Export**.

Beginning in 22.3.0-PCE, the Compatibility Report displays VEN packages that are required but are either missing or there's a problem with their installation. This information helps you troubleshoot a failed installation.

The compatibility checks vary by the workload's operating system.

Linux Operating Systems

Incompatibility Type	Reason for incompatibility with Illumio Segmentation for Data Centers	Results
IPv4 forwarding enabled	At least 1 iptables forwarding rule is detected in the forwarding chain. VEN removes existing iptables rules in the non-Idle policy state.	Yellow
iptables rule count	At least 1 iptables filter rule is detected. VEN removes existing iptables rules in the non-Idle policy state.	Yellow
IPv6 global scope enabled	IPv6 is enabled for the workload.	Yellow
ip6tables rule count	At least 1 iptables filter rule is detected. VEN removes existing ip6tables rules in the Visibility policy state	Yellow
IPsec service enabled	UDP port 500/4500 is in use by other services. Do not enable SecureConnect for the workload.	Red
Routing table conflict	The StrongSwan routing table setting conflicts with existing networking routing tables. Do not enable SecureConnect for the workload.	Red

Windows Workloads

Incompatibility Type	Reason for incompatibility with Illumio Segmentation for Data Centers	Results
IPv6 enabled	IPv6 is enabled for the workload.	Yellow
Virtual loopback interfaces	Virtual loopback interface is detected. Untested and unsupported configuration.	Yellow
Firewall GPO	Windows firewall Group Policy Object (GPO) is detected. For more information, see KB Article #3545 Firewall GPO Warning Under Compatibility Report (login required).	Yellow
IPsec service enabled	IKEEXT service is disabled. Do not enable SecureConnect for the workload.	Yellow

AIX and Solaris Workloads

Incompatibility Type	Reason for incompatibility with Illumio Segmentation for Data Centers	Results
IPv4 forwarding enabled	IPv4 is enabled for the workload.	Yellow
iptables rule count	At least 1 iptables filter rule is detected. VEN removes existing iptables rules in the non-Idle policy state.	Yellow
IPv6 global scope enabled	IPv6 is enabled for the workload.	Yellow
ip6tables rule count	At least 1 iptables filter rule is detected. VEN removes existing ip6tables rules in the Visibility policy state	Yellow
IPsec service enabled	IPsec service is already in use. Do not enable SecureConnect for the workload.	Red

AIX Workloads only

Incompatibility Type	Reason for incompatibility with Illumio Segmentation for Data Centers	Results
IPv6 active connection count	Complementary check whether IPv6 global scope is enabled.	

Pairing Script and Package Installation (Linux & Windows)

The following information is provided for your reference so that you understand the process and events that occur when you install a VEN by using a pairing script in the PCE or by installing a package with the CLI.

Linux Pairing Script for VEN Library

The following example shows a typical Linux pairing script. The pairing script works with the VEN Library in the PCE web console:

```
rm -fr /opt/illumio_ven_data/tmp && \
umask 026 && mkdir -p /opt/illumio_ven_data/tmp && \
curl --tlsv1 "https://example.com:8443/api/v18/software/ven/
image?pair_script=pair.sh&profile_id=2" -o /opt/
illumio_ven_data/tmp/pair.sh && \
chmod +x /opt/illumio_ven_data/tmp/pair.sh && \
/opt/illumio_ven_data/tmp/pair.sh \
--management-server example.com:8443 \
--activation-code <code>
```

This pairing script performs the following actions on the workload:

1. Deletes the `/opt/illumio_ven_data/tmp` directory, if it already exists.
2. Changes `umask` to `026` to prevent the `group-write` and `others-read,write` permissions as it creates the `/opt/illumio_ven_data/tmp` directory.
3. Uses `curl` to download the pairing script from the VEN repository and store it in the `/opt/illumio_ven_data/tmp` directory.
4. Changes the script permissions to allow execution.
5. Runs the `opt/illumio_ven_data/tmp` script with the following command line options:
 - `--management-server` to communicate with the PCE
 - `--activation-code` to authenticate the VEN to the PCE and authorize the VEN to pair with the PCE

The pairing script installs the VEN packages on the workload and pairs the VEN with the PCE. The output of pair is captured in `/var/log/illumio_install.log`.

Next, the script performs the following operations:

1. Detects OS release and CPU architecture. Ensure the combination is supported.
2. Downloads the package to `/opt/illumio_ven_data/tmp`.
3. Uses native OS package manager (detected by line 1) to install the package.

Using native package managers is simpler for newer operating systems. For example, Illumio can use `yum` to manage package dependencies for the VEN and workloads. For older operating systems, customers have to manage dependencies by manually installing packages.
4. Verifies installation by invoking installed scripts.
5. Invokes `/opt/illumio_ven/bin/init_Platform start`.
6. Generates the activation file `/opt/illumio_ven_data/etc/agent_activation.cfg`.

7. Invokes `/opt/illumio_ven/bin/agent_status.sh` to activate the VEN.

RPM Installation

RPM installation performs the following operations:

1. Creates the `ilo-ven` user and group, unless a custom username is specified at installation.
2. Prepares and then starts the Illumio Segmentation for Data Centers to perform the following actions:
 - a. Loads the necessary kernel modules: `ip_tables`, `iptables_filter`, `nf_conntrack`, `nf_conntrack_ipv4`, `nf_conntrack_ftp`, `ipt_LOG`, `ip_set`, `ip6_tables`, `ip6table_filter`, `nf_conntrack_ipv6`, `ip6t_LOG`
 - b. Sets `net.netfilter.nf_conntrack_tcp_timeout_established` to 8 hours (28,800 seconds).
 - c. Disables and stops the system firewall service `iptables`.
This action is acceptable because the Illumio services act in place of the `iptables` service.
 - d. Saves existing `iptables` rules if any.
 - e. Loads `iptables` rules computed from PCE firewall policy.
 - f. Starts the VEN components described in "Description of VEN Components" in the VEN Administration Guide .
This step includes monitoring system `iptables` configuration (similar to the service `iptables` performed).

Windows Pairing Script

The following example shows a typical Windows pairing script. The pairing script works with the VEN Library in the PCE web console. *(Line breaks have been added for readability only.)*

```
PowerShell -Command "& {Set-ExecutionPolicy -Scope process
remotesigned -Force;
Start-Sleep -s 3;
Set-Variable -Name ErrorActionPreference -Value
SilentlyContinue;
[System.Net.ServicePointManager]::SecurityProtocol=[Enum]::ToOb
ject([System.Net.SecurityProtocolType], 3072);
Set-Variable -Name ErrorActionPreference -Value Continue;
(New-Object System.Net.WebClient).DownloadFile('https://
example.com:8443/api/v18/software/ven/image?
pair_script=pair.ps1&profile_id=1', (echo
$env:windir\temp\pair.ps1)); & $env:windir\temp\pair.ps1
-management-server example.com:8443 -activation-code
<code> }"
```

This pairing script performs the following actions on the workload:

1. Changes execution policy of the host PowerShell process to Remote-Signed.
2. Configures the .NET framework **System.Net** class to negotiate TLS 1.2. The Windows VEN uses “3072” instead of “Tls12” because the **enum** value is not defined in older Windows operating systems. When **system.net** does not support TLS 1.2, the script fallbacks to using the system default.
3. Using the .NET framework **WebClient** class, downloads **pair.ps1** from the VEN repository and stores it in the **\$env:windir\temp** directory.
4. Runs the **pair.ps1** script with the following command line options:
 - **-management-server**: Used by the VEN to communicate with the PCE
 - **-activation-code**: Used by the PCE to authenticate and authorize the VEN during the pairing process
5. The pairing script installs the VEN packages on the workload and pairs the VEN with the PCE. The output of **pair.ps1** is captured in **\$env:windir\temp\illumio.log** or **\$env:tmp\illumio.log**.

The script performs the following actions:

- a. Downloads the VEN installer from the VEN repository and installs it.
- b. Generates **agent_activation.cfg** file with PCE information
- c. Retrieves agent activation status and displays it.

VEN Compatibility with Workloads Hosting Containers

This topic describes how to enable VEN compatibility with workloads hosting containers. It covers how to configure the PCE as well as VENs that are supporting the underlying hosts.



NOTE

- This topic describes only how to enable VEN compatibility with workloads hosting containers. It doesn't describe Illumio support for containers.
- Standard VENs are not supported for use in managed containerized environments or overlay networks. If you are operating in a managed containerized environment like Kubernetes or OpenShift, you will need to use the Illumio C-VEN. See [Illumio for Kubernetes and OpenShift](#).

How it works

Illumio Core is compatible with using VENs to protect workloads that host containers. The VEN can apply out-of-band policy to traffic on the forwarding path of the firewall.

The VEN provides a capability called **Container Inherits Host Policy (CIHP)** whereby the VEN applies policies, written for a host workload, to the forwarded traffic. The host, in turn, forwards traffic that is inbound to the container to its destination and in this way the container traffic is subjected to the host firewall policy.

Requirements and Limitations

- See the note at the start of this topic.
- If you are using Docker version 27.x or later, make sure to install the **br_netfilter** module before you perform the procedure [Configure the PCE to support containers \[131\]](#) in this topic (See the installation instructions in the knowledge base links below.) Otherwise, traffic you intended to allow may be blocked in Visibility Only and/or Full Enforcement mode. Previously, the **br_netfilter** module was loaded by default, but starting with Docker version 27 this module is loaded only when explicitly needed. The **br_netfilter** module must be installed for the VEN to apply all necessary rules. For more information, see the support Knowledge Base articles [Container traffic gets blocked. br_netfilter is not enabled by default on Docker 27](#) and [Standalone Docker Blocked with VEN in Visibility](#). (Requires logging in to the [Illumio Support Portal](#).)
- The VEN's interaction with workloads hosting containers does not represent the container as a workload (map or policy object), and each container is implicitly part of the Docker host (the workload).
- FQDN-based rules are not supported for any type of container technology with the VEN.
- Containers must share the same policies as the host.
- CIHP is supported on RHEL 8 / Oracle Linux 8 / Ubuntu 22 and later with Illumio Core 22.5 and later.
- In Illumio Core 21.5 and later, if you want to enforce traffic on the container's hosts only, allow all the traffic to containers and bypass CIHP by enabling IP forwarding. See [Enable IP Forwarding](#) in the PCE Administration Guide.
- No support for segmentation between the host and containers on the same host.
- No support for segmentation between containers on the same host.

Verified container engines



NOTE

Make sure that the VEN is compatible on your version of Linux. (See the [Illumio Support Portal](#).)

VEN interaction with workloads hosting containers has been verified to be compatible with the following container engines:

Container Engine	Supported Versions as of July, 2025
Docker	19.03 through 28.x
Podman	3.0 and 4.0

Enable PCE compatibility with workloads hosting containers

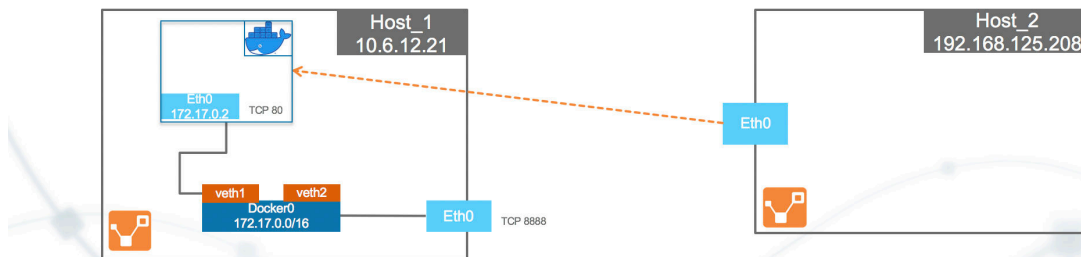
1. If using Docker version 27.x or later, first see the bullet in the [Requirements and Limitations \[130\]](#) section about the `br_netfilter` module.
2. Configure Containers to inherit host policy.
This is disabled by default. Enablement copies the host policy (all Illumio iptables related rules) into the `filter:FORWARD` chain so that packets forwarded to containers are controlled by the Illumio security policy.
Set a scope of Illumio labels for hosts with containers:
 - a. Go to **Settings > Security**.
 - b. Click the **Containers Policy** tab and then click **Edit**.
 - c. Add scope for host with containers.
You can define a narrow scope with specific label values or a broad scope that encompasses all workloads. For more information, see "Ruleset Scope" in the Security Policy Guide.
3. Pair workloads with containers. If they are already paired, go to the Security section of the workload's details page and verify that the workload's containers settings shows **Container Inherit Host Policy: Yes**
4. Write rules to cover the port mapping between the host and the container. Below is an example scenario of container access:

Policy Writing Example 1: Host to Container Traffic



- Host_1 runs 1 container
- Host_2 is a standard host running no containers
- Host_2 needs to reach container_1(docker0). Container_1 port 80 maps to host port 8888.
- The rule must have Host_1 labels providing both the host TCP port (8888) and the container TCP port(80) for consuming Host_2 labels

Provider	Providing Services	Consumer
Host-1 (R-A-E-L Labels)	TCP 8888 TCP 80	Host-2 (R-A-E-L Labels)



To allow access to the container on Host_1 port 80, create an intra-scope rule for access within the application group and extra-scope rule for access from outside of the application group.

The example rule below depicts any workload being allowed to access container port 80 on Host_1. Notice the service includes port 80 of container port and port 8888 of host port.

Rule allowing access to container on Host_1:

Summary **Scopes and Rules** Duplicate Ruleset

Viewing draft version Up to date [View the active version.](#)

Scopes docker_app | docker_env | docker_loc + - ▾

Status	Application	Environment	Location
☐	docker_app	docker_env	docker_loc

Rules + ▾ - ▾ ▾ ▾ 2 Total

1 Intra-Scope Rule + Reorder Rules 1 - 1 of 1 Total

No.	Provision Status	Status	Providers	Providing Service	Consumers	Note
1	Enabled	☒ All Workloads		8888_container_port_name 80 TCP, 8888 TCP	☒ All Workloads	Edit Down Arrow

1 Extra-Scope Rule + Reorder Rules 1 - 1 of 1 Total

No.	Provision Status	Status	Providers	Providing Service	Global Consumers	Note
1	Enabled	☒ All Workloads		8888_container_port_name 80 TCP, 8888 TCP	Any (0.0.0.0/0 and ::/0)	Edit Down Arrow

Docker will have rules to NAT port 8888 to port 80.

5. Verify traffic flows to the containers on the VEN.

Verify in the log at `/opt/illumio_ven_data/log/vtaptopdrop.log` that there is no dropped traffic to the containers. In the `/opt/illumio_ven_data/log/vtapflow.log` file, verify that there are flows to the containers on the VEN when workload is enforced.

The highlighted log entry below shows the flow between host_2 192.168.125.208:54253 to container on Host_1 172.17.0.2:80:

```
==> /opt/illumio_ven_data/log/vtapflow.log <==  
2020-02-03T16:47:15.104-08:00 docker0 0 0 4 192.168.125.208  
172.17.0.2 6 64886 80 12286 C 0 U SWID=3c1b9f96-969a-472e-  
bbbc-9d1c93751ef9 TBI=0 TB0=116  
2020-02-03T16:48:15.157-08:00 docker0 0 0 4 192.168.125.208  
172.17.0.2 6 64885 80 72339 C 0 U SWID=3c1b9f96-969a-472e-  
bbbc-9d1c93751ef9 TBI=0 TB0=116
```

Inbound traffic from Host_2 to Host_1 will not be shown in `vtapflow.log`, only traffic to container IP address from Host_2.

Illumio Legacy Windows VEN

This guide describes how to install and use the Legacy Windows VEN (LW-VEN) with the Illumio Core PCE to enforce security policies on computers running the Windows Server 2003 SP1 & SP2 or Windows Server 2008 SP1 & SP2 operating system.

The LW-VEN is a custom version of the standard Illumio VEN. Like standard VENs, the LW-VEN is a lightweight multi-process application with a minimal footprint that runs on a workload. The LW-VEN programs the native OS firewall that enforces the policy controls you've defined for your legacy Windows machines.

- [About LW-VEN \[134\]](#)
- [Requirements and Limitations \[134\]](#)
- [Installation \[138\]](#)
- [Troubleshooting \[147\]](#)

About the Illumio Legacy Windows VEN

This guide describes how to install and use the Legacy Windows VEN (LW-VEN) with the Illumio Core PCE to enforce security policies on computers running the Windows Server 2003 SP1 & SP2 or Windows Server 2008 SP1 & SP2 operating system.

The LW-VEN is a custom version of the standard Illumio VEN. Like standard VENs, the LW-VEN is a lightweight multi-process application with a minimal footprint that runs on a workload. The LW-VEN programs the native OS firewall that enforces the policy controls you've defined for your legacy Windows machines.

In addition to this guide, you can also refer to the VEN Installation and Upgrade Guide and the VEN Administration Guide for in depth information about working with standard Illumio VENs.

LW-VEN Requirements and Limitations

This section covers the LW-VEN's setup operations, requirements, limitations, and caveats.

The LW-VEN software installs the Illumio Legacy Windows VEN Service on your supported legacy Windows machines. Once installed, the Illumio Legacy Windows VEN Service:

- Enforces policy received from the PCE.
- Consumes CPU as needed to calculate or optimize and apply the firewall while remaining idle in the background as much as possible.

You control the Illumio Legacy Windows VEN Service's operations through the PCE web console or from the command line on the Windows machine on which the LW-VEN is installed.

Set-up Sequence

When run, the Illumio Legacy Windows VEN Service automatically does the following:

1. Checks whether this solution is supported.
2. Installs and pairs an LW-VEN on your legacy Windows Servers.
3. Creates a workload on the PCE to represent your legacy Windows Servers as a managed workload. A secured workload is known as a managed workload.
4. When running, the service:
 - Requests policy from the PCE as follows: after the LW-VEN sends a heartbeat to the PCE every five minutes, if there are any policy updates, the LW-VEN requests them from the PCE. If there are no policy updates, the LW-VEN performs a tamper check on its local policy to ensure that it hasn't been changed.
 - Applies the Illumio firewall rules obtained from the PCE to the Windows workload.

If the Illumio Legacy Windows VEN Service fails, Windows restarts it automatically.

Requirements

- IllumioLWVENInstaller.exe
- Illumio Policy Compute Engine (PCE) release 23.2.20 or later.
- Supported legacy Windows servers
 - Microsoft Windows Server **2003** Service Pack 1 & Service Pack 2, 32-bit or 64-bit
 - Microsoft Windows Server **2008** Service Pack 1 & Service Pack 2, 32-bit or 64-bit
- CPUs:
 - 32-bit Windows Servers: 2x 32-bit CPUs
 - 64-bit Windows Servers: 2x 64-bit CPUs

- RAM:
 - 32-bit Windows Servers: 4GB minimum
 - 64-bit Windows Servers: 8GB minimum
- .NET Framework 4.0.0 (minimum required; versions 5.0 and later are not supported.)
- **Certificate validations:** When sending requests to the PCE, the LW-VEN performs peer certificate validation by validating the certificate against the generally available `cert.pem` file provided in the Illumio LW-VEN Service\certs directory. If you need to add extra certificate validations, **add the appropriate .pem files to the \certs directory before activating the LW-VEN.**
- A dedicated local user account with admin privileges for installing and modifying the Windows firewall, running the service, and issuing the `illumio-lwven-ctl` commands.



IMPORTANT

You must disable the User Access Control (UAC) feature if it is enabled on the legacy Windows Server machines on which you plan to install the Illumio Legacy Windows VEN Service. Otherwise, you will not be able to install the LW-VEN on the machine. UAC is a Windows security feature that prevents unauthorized changes to the operating system.

- Make sure the interface language of the user account used to activate and run the LW-VEN is set to **any version of English**. The LW-VEN requires English to parse the output of the `systeminfo` and `ipconfig /a` commands issued to extract system data about your Windows server (for example, host name, operating system, etc.):
 - Windows Server **2003**: Click **Start**, point to Control Panel, and then click Regional and Language Options.
 - Windows Server **2008**: Click **Start**, point to Control Panel, and then in the section Clock, Language, and Region, click Change display language.

Limitations and Caveats

Take careful note of the following limitations and caveats.

Item	Windows 2003 Server SP1 & SP2	Windows 2008 Server SP1 & SP2
Policy size and server specifications	Changing or removing a large policy on an under-powered server may result in policy failure in some cases. To avoid this and possibly other unexpected policy issues, Illumio recommends that you avoid applying a policy that will generate more than 1000 firewall rules or which affects more than a total of 50000 port/IP combinations on a server with less than 2 x64 CPUs and 8GB RAM.	
Enforcement modes	Support for: • Idle mode • Full Enforcement If you change the Enforcement Mode from Full to Idle, the Illumio Legacy Windows VEN Service removes all Illumio policy from the Windows server. If you switch back to Full enforcement, the policy is reapplied to the workload. Although the Visibility and Selective options are not supported with Win 2003 SP1/SP2 servers, the options still appear in the PCE UI in the Enforcement drop-down menu on each Workload's details page. If you change the Enforcement mode from Full to Visibility or Selective, the LW-VEN ignores the policy and logs an event to the Windows Event Log and the PCE.	Support for: • Idle mode • Visibility mode • Selective Enforcement • Full Enforcement If you change the Enforcement Mode from Full to Visibility or Selective, the PCE creates an Illumio ALLOW ALL rule, effectively allowing all non-blocked traffic.  NOTE In Selective Enforcement mode, the Windows 2008 Server firewall applies all block rules before applying any allow rules. This behavior is opposite to how the standard Illumio VEN works on other Windows systems.
In-bound/Out-bound Rules	Support for: • Inbound rules only	Support for: • Inbound rules • Outbound rules
Policy Rules Limitations	Support for: • Port & protocol rules only • One rule per port/protocol. For example, if you specify a rule that includes a port range, multiple single rules are created, one per port. • Support for programming only a list of IP addresses and/or CIDR blocks per rule. (IP ranges are converted to CIDR blocks.) The LW-VEN always attempts to merge IP addresses into the most com-	Support for: • Port & protocol rules only • Specifying a rule that includes a port range results in a single rule, but ports are shown in a comma-separated list instead of a port range. • IP ranges. (CIDR blocks are converted to an IP range.) The LW-VEN always attempts to merge IP addresses into the most compact IP ranges possible. • If the Illumio Legacy Windows VEN Service is uninstalled, all Illumio rules are removed from the firewall.

Item	Windows 2003 Server SP1 & SP2	Windows 2008 Server SP1 & SP2
	pact CIDR addresses possible. If the Illumio Legacy Windows VEN Service is uninstalled, all Illumio rules are removed from the firewall.	
Matching rules	- Exact Matches (port/protocol and all IPs match) Customer rule remains enforced; Illumio rules are not applied. - Partial Matches (port/protocol match but only some or no IP addresses match) If a customer rule exists for the same port & protocol as an Illumio rule, the Illumio rule is applied and the customer rule is overwritten.	- Exact Matches (port/protocol and all IPs match) Customer rule remains enforced; Illumio rules are not applied. - Partial Matches (port/protocol match but only some or no IP addresses match) If a customer rule exists for the same port & protocol as an Illumio rule, the customer rule is disabled and the Illumio rule applies. If the customer suspends or uninstalls the Illumio LW-VEN Service, their partially matching rules, if any, remain disabled.
Rule character limits	Windows limits the size of rules to approximately 8K characters. Rules that exceed 8K characters will cause the entire policy to be rejected and a message to be logged in the Window's Event Log.	Windows limits the size of rules to approximately 8K characters. Rules that exceed 8k characters are split into multiple rules. No limit on the number of rules is enforced.
Error handling	Log messages are written to local logs; errors and warnings are also written to the Windows Event Log and to the PCE.	
LW-VEN and workload names in the PCE	After you activate an LW-VEN, the LW-VEN workload appears in the PCE UI with the same name as the Server's hostname.	
User interface	- The Upgrade button that appears on the VEN page in the PCE Web Console doesn't apply to this solution. Clicking the button has no effect. - If you unpair the LW-VEN through the PCE UI by clicking Unpair on the LW-VEN's detail page, only the Open All Ports option is supported.	

Install and Configure the Illumio LW-VEN Service

This section details how to install and configure the Illumio LW-VEN Service.

STEP 1: (Recommended) Back Up the Existing Firewall Configuration

Before you install the Illumio LW-VEN Service, Illumio recommends that you back up your legacy Windows's existing firewall configuration in case it becomes necessary to revert back to it. For example, reversion would be necessary if you uninstall the Illumio LW-VEN Service.

STEP 2: Create or Find a Pairing Profile with the Appropriate Settings



IMPORTANT

Note that this solution differs from the standard VEN pairing process in that it doesn't use the pairing script available in the pairing profile. Only a properly-encoded pairing key is required to pair the LW-VEN installed on your legacy Windows server with the PCE.

All pairing keys are generated from a Pairing Profile and are encoded with settings from that profile. The pairing key you obtain or generate for this solution must have been generated from a pairing profile with the appropriate settings for your type of Windows server.

Minimum required pairing profile settings

Operating System	Supported Enforcement Modes	Supported Enforcement Node Type
Win 2003 Server SP1 & SP2	• Idle (recommended) • Full	Server VEN
Win 2008 Server SP1 & SP2	• Idle • Visibility • Selective • Full	Server VEN

Option 2.1 - Create a new Pairing Profile

To create a new pairing profile, go to **Servers & Endpoints > Pairing Profiles** and configure settings using this image and the table above as a guide.

For more information about creating a pairing profile, see "Configure a Pairing Profile" in the VEN Installation and Upgrade Guide.

Enforcement

Key

Win 2003

Win 2008

Idle

VEN does not take control of the host firewall

Visibility Only

No traffic is blocked by policy

Selective

Rules are enforced only for selected inbound services when workload is within scope of a Deny Rule

Full

Rules are enforced for all inbound and outbound services. Traffic not allowed by a Rule is blocked

Enforcement Node Type

Server VEN

Activate server VENs with pairing keys generated via this pairing profile. VENs that do not support server mode cannot be activated using this pairing profile. VENs cannot be activated using a command-line option to specify endpoint mode with this pairing profile.

Endpoint VEN

Activate endpoint VENs with pairing keys generated via this pairing profile. VENs that do not support endpoint mode cannot be activated using this pairing profile.

Specified during VEN activation (deprecated legacy option)

Specify that a VEN should run in endpoint mode via command-line option.

Option 2.2 - Find an existing Pairing Profile with the proper settings

To identify an existing pairing profile with the appropriate settings for your server type, go to **Servers & Endpoints > Pairing Profiles** and find a profile with **Enforcement Node Type: Server VEN** and the Enforcement mode(s) appropriate for your Windows Server.

You can filter the list by Enforcement Node Type.

Pairing Profiles ⓘ				
⊕ Add ⊖ Remove				
Enforcement Node Type: Server VEN x				
☐	Pairing Status	Enforcement Node Type	↕ Name	Enforcement
☐	Running	Server VEN	pp-rhc	Visibility Only
☐	Running	Server VEN	PP5-9713	Idle
☐	Running	Server VEN	PP2-969	Selective
☐	Running	Server VEN	PP1-5942	Full

STEP 3: Obtain or Generate a Pairing Key in the PCE Web Console

Choose one of the following options to obtain a pairing key.

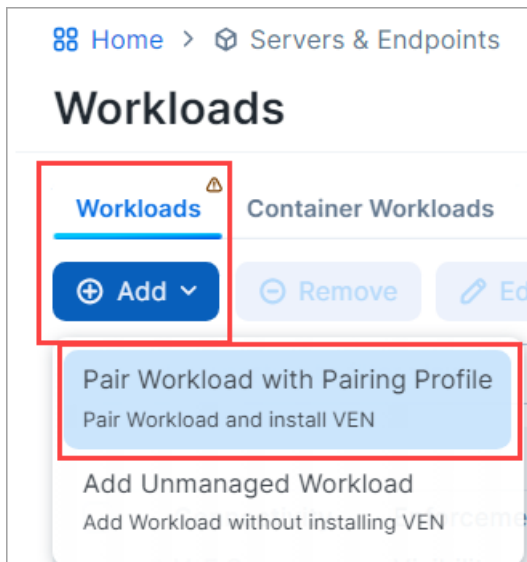


IMPORTANT

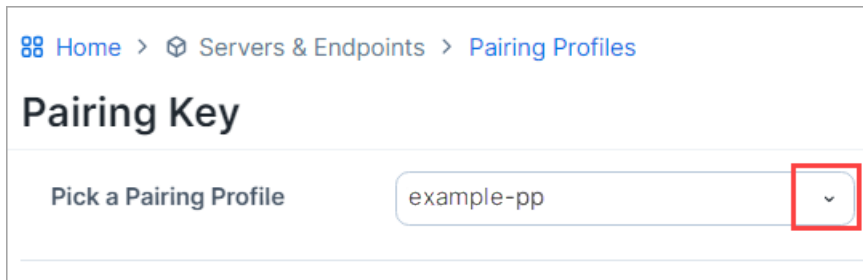
As detailed in [STEP 2: Create or Find a Pairing Profile with the Appropriate Settings \[139\]](#), make sure that the pairing key you obtain or generate for this solution was generated from a pairing profile with the appropriate settings for your type of Windows server.

Option 3.1 - Copy a Pairing Key from an existing Pairing Profile

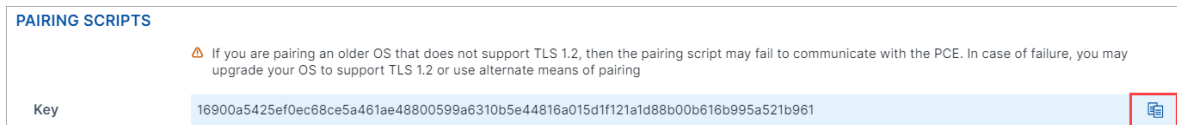
1. Expand the **Servers & Endpoints** section in the left navigation.
2. Click **Workloads**.
3. Click **Add**, and then choose **Pair Workload with Pairing Profile**.



4. In the **Pick a Pairing Profile** drop-down list, select the pairing profile you identified previously that has the appropriate settings for your legacy Windows server (see [STEP 2: Create or Find a Pairing Profile with the Appropriate Settings \[139\]](#)).



5. Scroll down to **Pairing Scripts** and copy and preserve the Key for use in [STEP 4 \[143\]](#).



IMPORTANT

Don't copy the pairing script available in the pairing profile. Pairing scripts are not used with this solution.

Option 3.2 - Generate a new Pairing Key from an existing Pairing Profile

1. Expand the **Servers & Endpoints** section in the left navigation.
2. Click **Pairing Profiles**.
3. Click an existing pairing profile that has the appropriate settings for your legacy Windows server (see [STEP 2: Create or Find a Pairing Profile with the Appropriate Settings \[139\]](#)).
4. Click **Generate Key**.
5. Scroll down to **Pairing Scripts** and copy and preserve the Key for use in [STEP 4 \[143\]](#).

Option 3.3 - Create a new Pairing Profile

1. Expand the **Servers & Endpoints** section in the left navigation.
2. Click **Pairing Profiles**.
3. Click **Add** and configure the settings appropriate for your legacy Windows server as described in [Option 2.1 - Create a new Pairing Profile \[139\]](#).
4. Click **Save**.
5. Open the Pairing Profile you just created.
6. Scroll down to **Pairing Scripts** and copy and preserve the Key for use in [STEP 4 \[143\]](#).



IMPORTANT

Don't copy the Pairing Script. The script is not used in this solution.

STEP 4: Install, configure, and pair the Illumio Legacy Windows VEN Service on a legacy Windows server



IMPORTANT

- You must disable the User Access Control (UAC) feature if it is enabled on the legacy Windows Server machines on which you plan to install the Illumio Legacy Windows VEN Service. Otherwise, you will not be able to install the LW-VEN on the machine. UAC is a Windows security feature that prevents unauthorized changes to the operating system.
- You must install and activate the Illumio Legacy Windows VEN Service from a dedicated local admin account.
- Only the Illumio LW-VEN Service account user can run the LW-VEN service and issue `illumio-lwven-ctl` commands.

1. Obtain the `IllumioLWVENInstaller.exe` file and place it on the Windows server. (Recommended location: `C:\Users\Administrator`). The installer is available on the Illumio Support portal.
2. Perform one of the following installation + activation options:

Option 4.1 - Respond to prompts

Launch the file from a command line or by double-clicking the file.

a. Install

- **Select Destination Location**

- 32-bit machines: (`C:\Program Files\Illumio LW-VEN Service`)
- 64-bit machines: (`C:\Program Files(x86)\Illumio LW-VEN Service`)

- **Select Additional Tasks**

- Make sure the path option is selected, and then click **Next**.

- **Ready to Install**

- Click **Install**.

- **Complete the Setup Wizard**

- Select "Launch illumio-lwven-ctl activate"
- Click **Finish**. The `certs.pem` file is added immediately after you click **Finish**.

b. Activate and configure

- **Complete the Setup Wizard**

- Select **Launch illumio-lwven-ctl activate**
- Click **Finish**. The `certs.pem` file is added immediately after you click **Finish**.

- **Enter hostname: port for PCE address:** Enter the subdomain(s) and domain of your PCE's web address and port (for example, `example.illumio:8443`).

- **Enter pairing key for LW-VEN:** Paste the pairing key that you obtained in [STEP 3 \[141\]](#) and then press Enter.

Messages appear:

- Pairing to `<pcehost:port>`
- Activation Complete
- You are prompted to enter the user account password, which is necessary to run the Illumio LWVEN Service.

Option 4.2 - No prompts (useful for automated processes)

Enter the following command at a command prompt:

```
C:\Users\Administrator > illumio-lwven-ctl activate --management-server <pcehost:port> --activation-code <pairing-key> [--password <account-password>]
```



NOTE

For a fully automated activation process, make sure to include the `--password` option and specify the user account password. Otherwise, you are prompted to enter a password to complete the activation.

Messages appear:

- Resolving: `<pcehost>`

- Pairing to PCE `<pcehost:port>`
 - `POST /org/0/agents/activate`
 - Activation Complete
3. Go to **Servers & Endpoints > Workloads > VENs**
 4. Click the name of the LW-VEN you added.
 5. Confirm the following on the LW-VEN's details page:
 - **NODE** section:
 - Hostname: (your-Windows-Server-Computer-Name)
 - Enforcement Node Type: See [STEP 2: Create or Find a Pairing Profile with the Appropriate Settings \[139\]](#).
 - **HOST** section:
 - OS: LW-VEN 1.0.0
 6. You can perform the following operations on the LW-VEN (For details, see the VEN Administration Guide):
 - Edit the LW-VEN
 - Generate a support bundle (see Support report).
 - Mark the LW-VEN Suspended

**NOTE**

This should be necessary only if you issue the `illumio-lwen-ctl suspend` command and receive a message indicating that the LW-VEN failed to inform the PCE of its suspension.

- Unpair the LW-VEN

**NOTE**

If you unpair the LW-VEN through the PCE UI by clicking Unpair on the LW-VEN's detail page, only the **Open All Ports** option is supported.

STEP 5: Enable Flow Reporting

**NOTICE**

This feature is available in LW-VEN releases 1.1.0 and later.

The LW-VEN can enable the native Windows Firewall log on your legacy Windows server and then send traffic flow information to the PCE. After ingesting the log information, the PCE displays it in its Map and Traffic views to help you gain insights about – and create policy for – your business applications.

How LW-VEN Flow Reporting works

1. Enable Windows Firewall logging on your legacy Windows server by issuing a command on the LW-VEN.
2. The LW-VEN parses the firewall log every 10 minutes and pushes the extracted information to the PCE.
3. The PCE ingests the information from the LW-VEN and makes it available in its Visualization tools (Map, Traffic, and Mesh views).
4. Use the information to create, update, verify, and troubleshoot Illumio policy rules for your legacy Windows server.

Limitations

- Depending on the amount of traffic passing through the firewall, enabling firewall logging may impact the server's performance. If this occurs, try enabling flow reporting for only brief periods of time only as necessary (minimum 1 hr) and then disable it.
- Rules exceeding 1000 ports are split into multiple rules.
- Applying a policy with a large port range may cause the windows firewall to become unresponsive and take a long time to respond to any firewall command.

5.1 Enable, disable, or check status

1. From a command prompt on the Windows server, navigate to;

```
C:\Program Files (x86)\Illumio LW-VEN
```

2. Issue the following command and the appropriate option to enable, disable, or check status:

```
illumio-lwven-ctl flow-reporting [enable|disable|status]
```

5.2 View traffic flows and create policy

1. **Visualize traffic.** The PCE's visualization tools allow you to see the traffic flowing to and from your Windows server so you can configure and troubleshoot the Illumio policies you need to protect your applications. For details, see the Visualization Guide.
2. **Create policy.** Illumio Core relies on security policies to secure communications between workloads. Security policies are configurable sets of rules that protect network assets from threats and disruptions. For guidance

designing an Illumio security policy, including creating rulesets and rules, see the Security Policy Guide.

STEP 6: Create Security Policy

In the PCE web console, create label-based policies for your Windows Server 2003 SP1 & SP2 and Windows Server 2008 SP1 & SP2 workloads. For information on how to create policies, see the Security Policy Guide.

Manage and Troubleshoot the Illumio LW-VEN

This section covers Illumio LW-VEN pairing and activation concepts, Illumio firewall rules, tamper detection, support bundle generation, common commands, and troubleshooting.

About Pairing and Activation

The terms “activation” and “pairing” indicate the same function from different perspectives; namely, putting the workload under managed control by the PCE:

- The LW-VEN sees itself as activated or deactivated.
- The PCE sees an LW-VEN as paired or unpaired.

Pairing and Activating the LW-VEN		
1	The LW-VEN is installed.	The PCE remains unaware the LW-VEN is present.
2	The LW-VEN and the PCE are paired.	The PCE uses a pairing key (activation code) to pair with the LW-VEN. After pairing, the PCE becomes aware of the LW-VEN.
3	The LW-VEN is activated.	The LW-VEN uses an activation code generated by the PCE. After activation, the LW-VEN is ready to function.

Unpairing, deactivating, and uninstalling the LW-VEN

Here’s how these operations work in this solution:

- **Unpairing** the LW-VEN through the PCE UI or by issuing `illumio-lwvenctl unpair` unpairs the LW-VEN from the PCE and uninstalls the LW-VEN software.

- **Deactivating** the LW-VEN by issuing `illumio-lwven-ctl deactivate` unpairs the LW-VEN from the PCE but doesn't remove the LW-VEN software.
- **Uninstalling** the Illumio Legacy Windows VEN Service through the *Windows Control Panel > Programs and Features*:
 - Unpairs the LW-VEN from the PCE
 - Removes the Workload object from the PCE
 - Removes Illumio firewall rules and any working files
 - Uninstalls the LW-VEN software from the Windows server

View the Illumio rules applied to the native firewall

Illumio rules applied to the Windows Server's native firewall begin with `Illumio`. For example: `IllumioInTcp14000Permit`

There are two ways to view Illumio firewall rules:

- Generate a Support Report and look in the **Firewall.txt** file.
- Issue a command on the Windows Server:



NOTE

Using the `findstr` filter shows only the first line of the rule, not the entire rule.

- **Win 2003 SP1/SP2:** `C:\Users\Administrator> netsh firewall show portopening enable | findstr /R "Illumio.*"`
- **Win 2008 SP1/SP2:** `C:\Users\Administrator> netsh advfirewall firewall show rule name=all | findstr /R "Illumio.*"`

Tamper detection

The Illumio Legacy Windows VEN Service performs tamper checking whenever it heartbeats to the PCE (every 5 minutes) and discovers that there is no new policy to apply. Whenever the policy update check occurs, the Illumio Legacy Windows VEN Service checks whether the last-applied Illumio policy on the legacy server differs from the last applied policy from the PCE. If a difference is detected, the Legacy Windows VEN Service reverts the policy to the intended state so that the correct PCE security policy is enforced.

Support report

You can generate the Illumio Legacy Windows VEN Service support report. It includes the following information:

Firewall.txt: Lists all the rules currently programmed in the native Windows Firewall.

- Logs specifying:
 - When policy was last received
 - When policy was last applied and what was applied
 - System information (output of the `systeminfo` command)

Generate a Support Report

Option 1:

This is the simplest way to generate a report.



NOTE

This option assumes that the LW-VEN is in a running state on the Windows Server.

1. Go to Servers & Endpoints > Workloads > VENs
2. Click the name of the LW-VEN you added to go to its details page.
3. Click Generate Support Bundle.

The bundle is uploaded to the PCE (may take up to 10 minutes).

Option 2:

This option is useful if the LW-VEN is stopped due to a major problem.

- Issue `illumio-lwven-ctl support-report`

The location of the report on the Windows server is returned after you issue the command. This report is not sent to the PCE.

Logs

The Illumio LW-VEN Service logs its operations locally on the Windows Server. Logs are rotated from primary to backup when their size reaches 10MB or once every 24 hours at midnight.

Location

- 32-bit: `C:\Program Files\Illumio LW-VEN Service\logs`
- 64-bit: `C:\Program Files (x86)\Illumio LW-VEN Service\logs`

Archive

By default, seven log archives are preserved on the workload.

Commands

You can issue the following commands to interact with the Illumio LW-VEN Service.



NOTE

- Only the Illumio LW-VEN Service account user can issue `illumio-lwven-ctl` commands.
- All commands include the prefix `illumio-lwven-ctl`

- `activate`
- `status`
- `restart`
- `stop`
- `start`
- `unpair`

Removes the Illumio policy from the firewall, removes the LW-VEN from the PCE, and uninstalls the LW-VEN software from the Window's server. You can also uninstall the LW-VEN from the PCE by clicking Unpair for the appropriate LW-VEN on the PCE VEN page. With this unpairing method, it may take up to five minutes for the LW-VEN to be unpaired and uninstalled.

- `deactivate`

Removes the Illumio policy from the firewall; removes the PCE objects from the PCE and from the Illumio LW-VEN Service; does not remove the LW-VEN software from the installation directory (in case you want to later re-activate the LW-VEN without having to install the LW-VEN package).

- **support-report**
- **suspend**

Suspends the Illumio LW-VEN Service and uninstalls Illumio policy from the firewall.

- **unsuspend**

Enables and starts the Illumio LW-VEN Service; retrieves and applies the latest PCE policy.

Troubleshooting

This section describes how to troubleshoot common issues.

Issue	Remediation
The Illumio Legacy Windows VEN Service stops.	Check logs: Windows Event Viewer Log Local Illumio logs
Problem receiving policy from the PCE.	
Problem applying policy to the workload created by the Illumio Legacy Windows VEN Service.	
Problem with the connection between the Illumio Legacy Windows VEN Service and the PCE.	The Illumio Legacy Windows VEN Service tries every five minutes to reconnect to the PCE.
Unable to install, stop, suspend, or unpair the Illumio Legacy Windows VEN Service.	These issues may be caused by the User Access Control (UAC) feature if it is enabled on your legacy Windows Server machines. UAC is a Windows security feature that prevents unauthorized changes to the operating system. Disable the User Access Control (UAC) feature if it is enabled.
Pairing the LW-VEN with the PCE fails; a message indicates that the pairing key was generated from a pairing profile with unsupported settings for this solution, such as the wrong Enforcement mode or Enforcement Node Type.	Obtain a properly-encoded pairing key (see STEP 2 [139]) and repeat STEP 3 [141] and STEP 4 [143] .
LW-VEN fails to activate with the PCE.	Make sure the interface language of the user account used to activate and run the LW-VEN is set to any version of English. The LW-VEN requires English to parse the output of the systeminfo and ipconfig /a commands issued to extract system data about your Windows server (for example, host name, operating system, etc.): • Windows Server 2003: Click Start, point to Control Panel, and then click Regional and Language Options. • Windows Server 2008: Click Start, point to Control Panel, and then in the section Clock, Language, and Region, click Change display language.

Kubernetes and Openshift

Overview of Containers

This section describes the architecture, key concepts, and the integration requirements.

Before You Begin

- ☐ Prepare your environment
- ☐ Create a container cluster in the PCE
- ☐ Deploy Kubelink and C-VEs in your cluster
- ☐ Configure labels for namespaces, pods, and services
- ☐ Configure security policies for containerized environments
- ☐ Upgrade and uninstall the C-VE in your containerized environments
- ☐ Migrate to a Helm Chart deployment from a previously-installed C-VE deployment

Recommended Skills

- Illumio Segmentation (formerly Illumio Core)
- Linux shell (bash)
- TCP/IP networks, including protocols and well-known ports and a familiarity with PKI certificates
- Docker concepts, such as containers, container images, and docker commands
See [Get Started with Docker](#).
- Red Hat OpenShift Container Platform
See [OpenShift Documentation](#).
- Kubernetes concepts, such as clusters, Pod, and services
See [Kubernetes Documentation](#).

Introduction to Illumio Segmentation for Containers

Modern applications increasingly rely on containers and orchestrators like Kubernetes and OpenShift to deliver scalable, resilient services. While these

platforms offer agility, they also introduce security challenges. Because containers share the same underlying OS kernel and communicate frequently, a single compromised container can lead to lateral movement across the environment. Illumio addresses these challenges with a container-native segmentation solution that provides visibility and control across dynamic environments.

Containers and Orchestrators

Containers are lightweight, portable units that package applications with their dependencies. Containers allow large applications to be broken into smaller, more manageable parts. Containers run isolated processes using the host operating system kernel, making them faster and more resource-efficient than virtual machines.

Orchestrators such as Kubernetes and OpenShift manage container lifecycles, health, networking, and scaling. Examples of orchestrators include:

- **Kubernetes**
Open-source, cloud-native container orchestrator developed by Google and common in the public cloud that manages workloads via nodes, pods, services, and namespaces.
- **OpenShift**
Enterprise-ready product and platform often deployed in on-prem data centers and the cloud built on Kubernetes by Red Hat. It adds enhanced Role-Based Access Control (RBAC), developer tools, and CI/CD features.
- **Amazon Elastic Kubernetes Service (EKS)**
- **Azure Kubernetes Service (AKS)**
- **Google Kubernetes Engine (GKE)**

Illumio Segmentation for Containers Components

Illumio Segmentation for Containers extends segmentation to containerized environments. It secures workloads across Kubernetes, OpenShift, virtual machines, and bare-metal servers. Illumio Segmentation for Containers integrates directly with Kubernetes clusters using two key components:

- **C-VEN (Containerized Virtual Enforcement Node)**
C-VEN deploys as a DaemonSet on each node. It enforces security policies at the node and pod levels by using iptables or nftables. Each node requires only one C-VEN.
- **Kubelink**
Kubelink runs as a single pod in each cluster. It synchronizes Kubernetes metadata such as pods, services, and namespaces with Illumio's Policy

Compute Engine (PCE). Kubelink enables real-time traffic mapping and policy delivery. It automatically discovers nodes, network details, and services and sends that information to the PCE.

Benefits of Illumio Segmentation for Containers

- Unify security for containers, securing workloads across Kubernetes, OpenShift, VMs, and bare metal
- Gain visibility and control with real-time application traffic insight
- Work across environments like on-prem, hybrid, or cloud-native container deployments
- Deploy with Helm, performing installation and upgrades via Helm Charts on cloud-managed and self-managed clusters

How It Works

- **Visibility**
C-VEN captures traffic flows, while Kubelink maps them to workloads. This data is visualized in Illumio.
- **Deployment**
C-VEN and Kubelink are deployed via Helm Chart together, for installation and upgrades.
- **Policy Enforcement**
Policies are applied at the pod level using iptables or nftables. Pods are treated as single workload units in the PCE.

Illumio Segmentation for Containers Use Cases

Kubernetes node visibility and control

Illumio provides visualization and inventory for all container nodes, segmenting at the node level.

Application tier-to-tier segmentation

Illumio enables controlling communication between services. For example, you can control communication flow from web, to application, to database.

Namespace isolation

Illumio helps prevent cross-talk between namespaces and enforces environment boundaries. For example, you can control flow from production environments to development environments.

Ingress traffic control

Illumio secures access through ingress controllers or load balancers, to only allow sanctioned entry points.

Egress traffic control

Illumio enables restricting pods from reaching external or internet destinations by enforcing Fully Qualified Domain Names (FQDNs) and IP-based allowlists, with label-based policies that clearly define intent. Illumio enforces on both pod egress and external server ingress. The pod gets IPs of the external server in egress rules. The external server gets the IPs of the Kubernetes nodes in ingress rule.

Kubernetes infrastructure protection

Illumio can limit access to Kubernetes API and control plane components to harden clusters against lateral movement attacks.

Intra-cluster communication

Illumio provides policy granularity, controlling communication between pods in the same namespace or across namespaces.

Concepts

This section describes some key concepts of the Illumio Core for Kubernetes solution, which includes, as its main components, the Containerized VEN (C-VEN) and Kubelink.

Containerized VEN

Containerized VEN (C-VEN) is an Illumio-provided software component that provides visibility and enforcement on nodes and Pods. In a standard Illumio deployment, the Virtual Enforcement Node (VEN) is installed on the host as a package. The C-VEN is not installed on the host but runs as a Pod on the Kubernetes nodes. The C-VEN functions in the same manner as a standard VEN. However, to program iptables on the node and Pod namespaces, the C-VEN requires privileged access to the host.

The C-VEs are delivered as a DaemonSet with one replica per host in the Kubernetes cluster. A C-VEN Pod instance is required on each node in the

cluster to ensure proper segmentation in your environment. In self-managed deployments, C-VEs are deployed on all nodes in the cluster. In cloud-managed deployments, C-VEs are deployed only on the Worker nodes and not on the Master nodes (Cloud customers do not manage master nodes).

Container Cluster

A container cluster object is used to store all the information about a Kubernetes cluster in the PCE by collecting telemetry from Kubelink. Each Kubernetes cluster is mapped to a single container cluster object in the PCE. Each Pod network(s) that exists on a container cluster is uniquely identified on the PCE to handle overlapping subnets. This helps the PCE in differentiating between container workloads that may have the same IP address but are running on two different container clusters. This differentiation is required for both illumination and policy enforcement.

You can view the workloads associated with a container cluster in the PCE Web Console. This mapping between the host workload and the container cluster is done using **machine-ids** reported by Kubelink and C-VE.

Container Workloads

Container workloads are basic containers (as with Docker) or the smallest resource that can be assimilated within a container in an orchestration system (as with Kubernetes). In the context of Kubernetes and OpenShift, a Pod is a container workload. Similar to workloads, these container workloads (managed Pods) can have labels assigned to them. Container workloads with their associated Illumio labels are also displayed in Illumination. In Illumio Core, containers are differentiated based on whether they are on the Pod network or the host network:

- Containers on the Pod network are considered container workloads and can be managed similarly to workloads.
- Containers sharing the host network stack (i.e., Pods that are host-networked) are not considered container workloads and therefore inherit the labels and policies of the host.

To manage container workloads, you can define the policy state (Idle, Visibility Only, Selective, and Full enforcement) in container workload profiles.

Container Workload Profiles

A container workload profile maps to a Kubernetes namespace and defines:

- Policy state (Idle, Visibility Only, Selective, and Full enforcement) for the Pods and services that belong to the namespace.
- Labels are assigned to the Pods and services. Standard predefined label types were Role, Application, Environment, and Location. Newer releases of Core allow you to define your own custom label types and label values for these types.

Once Illumio Core is installed on a container cluster, all namespaces that exist on the cluster are reported by Kubelink to the PCE and made visible via Container Workload Profiles. Each time Kubelink detects the creation of a namespace from Kubernetes, a corresponding Container Workload Profile object gets dynamically created in the PCE.

Each profile can either be in a managed or unmanaged state. The default state for a profile is unmanaged. The main difference between the two states:

- Unmanaged: no policy applied to Pods by the PCE and no visibility
- Managed: The policy is controlled by the PCE and has full visibility through Illumination and Traffic Explorer.

A container workload profile is a convenient way to dynamically secure new applications with Illumio Core just by inheriting security policies associated with the scope of that profile.

Kubelink

Kubelink is a software component provided by Illumio to facilitate the integration between the PCE and Kubernetes. Kubelink queries Kubernetes APIs to discover nodes, networking details, and services, and synchronizes them between the Kubernetes cluster and the PCE.

Kubelink reports network information to the PCE, enabling the PCE to understand the cluster network for both the hosts and the Pods in the cluster. This enables the PCE to both accurately visualize the communication flow and create the correct policies for the C-VEs to implement in the iptables of the host and the Pods. It provides flexibility in the type of networking used with the cluster. Kubelink also associates C-VEs with the particular container cluster by matching a unique identifier of the underlying OS called **machine-id** reported by each C-VE with the one reported by the Kubernetes cluster.

Kubelink is not in the critical path for normal Pod creation. When Pods start, stop, scale up or down, or nodes reboot, these events are all detected by the C-VE, and the C-VE provides sufficient information to the PCE, allowing it to receive the security policy immediately. There is no dependency on Kubelink to keep that information in sync.

Kubelink is delivered as a Deployment with only one replica within the Kubernetes cluster. One Kubelink Pod instance is required per cluster. There is no node affinity required for Kubelink so that the Kubelink Pod can be spun up on either a Master or Worker node.

Virtual Services

Virtual services are labeled objects and can be utilized to write policies for the respective services and the member Pods they represent.

Kubernetes services are represented as virtual services in the Illumio policy model. Kubelink creates a virtual service in the PCE for services in the Kubernetes cluster. Kubelink reports the list of Replication Controllers, DaemonSets, and ReplicaSets that are responsible for managing the Pods supporting that service.

Workloads

A workload is commonly referred to as a host OS in Illumio Core. In the context of container clusters, a workload is referred to as a node. Usually, a Kubernetes cluster is composed of two types of nodes:

- One or more Master Node(s) - In the control plane of the cluster, these nodes control and manage the cluster.
- One or more Worker Node(s) - In the data plane of the cluster, these nodes run the application (containers).

In Illumio, Master and Worker nodes are referred to as workloads and are part of a container cluster. Labels and policies can be applied to these workloads, similar to any other workload that does not run containers. For a managed Kubernetes solution, only the Worker nodes are visible to the administrator, and the Master nodes are not displayed in the list of Workloads.

Architecture

With the increased adoption of containers, the threat of unauthorized lateral movement from vulnerabilities and exploits increases considerably in the east-west attack surface. In addition, **Destinations** and **Sources** may be other containers, bare-metal servers, or virtual machines running on-premises or in the cloud. Multiple disparate solutions create complexity in management and operational workflow, leaving your organization more open to attack.

Illumio Segmentation provides a homogenous segmentation solution for your applications regardless of where they are running - bare-metal servers, virtual

machines, or containers. It is a single unified solution with many points of integration, including how you can easily and quickly secure your applications regardless of their location or form.

A container is a loosely defined construct that abstracts a group of processes into an addressable entity, which can run application instances inside it. Containers are implemented using Linux namespaces and cgroups, allowing you to virtualize and limit system resources. Since containers operate at a process-level and share the host OS, they require fewer resources than virtual machines. The isolation mechanism provided through Linux namespaces allows containers to have unique IP addresses. Illumio Segmentation uses these mechanisms to program iptables or nftables in the network namespace.

Kubernetes-based orchestration platforms such as native Kubernetes and Red Hat OpenShift integrate with Illumio Segmentation by using the following two components in the cluster:

- Kubelink - An Illumio software component that listens to events stream on the Kubernetes API server.
CLAS - (Cluster Local Actor Store) A new architecture introduced in Core for Kubernetes 5.0.0, When Kubelink is enabled with CLAS, it tracks Pods at the Kubernetes Workload level, and dispenses any existing policy for them, reducing the load on and interaction with the Policy Compute Engine (PCE), which improves scalability, responsiveness, and overall system performance.
- Containerized VEN (C-VEN) - An Illumio software component that provides visibility and enforcement on the nodes and the Pods.

Learn about key concepts of the Kubernetes solution including the C-VEN and Kubelink.

Containerized VEN (C-VEN)

The C-VEN provides visibility and enforcement on nodes and Pods. In a standard Illumio deployment the Virtual Enforcement Node (VEN) is installed on the host as a package. In contrast, the C-VEN is not installed on the host but runs as a Pod on the Kubernetes nodes. The C-VEN functions in the same manner as a standard Illumio VEN. However, in order to program iptables or nftables on the node and Pods namespaces, the C-VEN requires privileged access to the host. For details on the privileges required by the C-VEN, see [Privileges \[189\]](#).

The C-VEs are delivered as a DaemonSet, with one replica per host in the Kubernetes cluster. A C-VEN Pod instance is required on each node in the

cluster to ensure proper segmentation in your environment. In self-managed deployments, C-VEs are deployed on all nodes in the cluster. In cloud-managed deployments, C-VEs are deployed only on the Worker nodes and not on the Master nodes (Master nodes are not managed by Cloud customers).

Kubelink

Kubelink is a software component provided by Illumio to make the integration between the PCE and Kubernetes easier. Starting in Illumio Segmentation (formerly Illumio Core) for Kubernetes 5.0.0, Kubelink is enhanced with a Cluster Local Actor Store (CLAS) module, that handles the workload-to-Pod relationship via C-VE communication. See Cluster Local Actor Store (CLAS) for details on how Kubelink in CLAS mode operates. The remainder of this description of Kubelink describes its basic, non-CLAS behavior.

- Kubelink queries Kubernetes APIs to discover nodes, networking details, and services and synchronizes them between the Kubernetes cluster and the PCE.
- Kubelink reports network information to the PCE, enabling the PCE to understand the cluster network for both the hosts and the Pods in the cluster. This enables the PCE to accurately visualize the communication flow and create the correct policies for the C-VEs to implement in the iptables or nftables of the host and the Pods. It provides flexibility in the type of networking used with the cluster. Kubelink also associates C-VEs with the particular container cluster by matching a unique identifier of the underlying OS called machine-id reported by each C-VE with the one reported by the Kubernetes cluster.

Kubelink is delivered as a deployment with only one replica within the Kubernetes cluster. One Kubelink Pod instance is required per cluster. There is no node affinity required for Kubelink so the Kubelink Pod can be spun up on either a Master or Worker node.

Cluster Local Actor Store (CLAS)

A Cluster Local Actor Store (CLAS) mode is introduced into the architecture of Kubernetes 5.0.0. When this mode is enabled, Kubelink still interacts with the Kubernetes API to track and manage Kubernetes components, and their interaction with PCE and C-VEs. This includes policy flowing from PCE to C-VEs and traffic flowing from C-VEs to PCE.

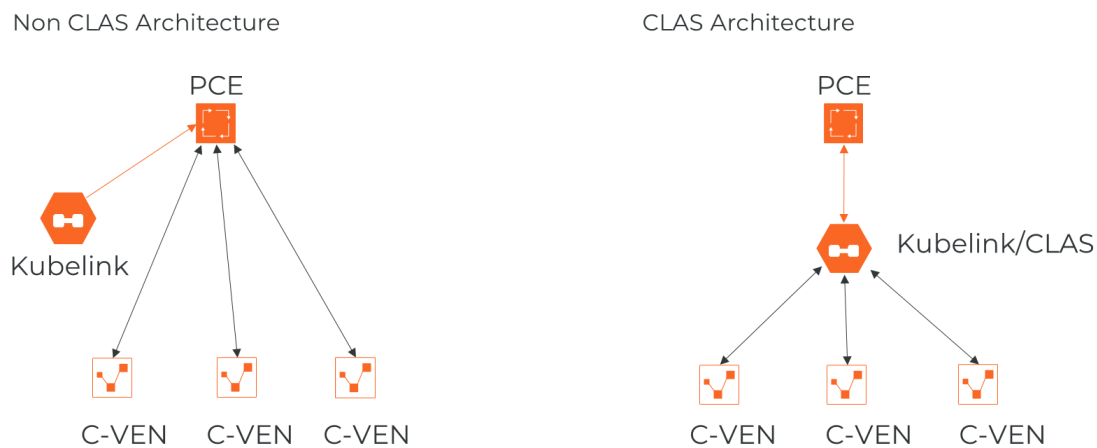
Within the CLAS architecture, Kubelink provides greater scalability, faster responsiveness, and streamlined policy convergence with several key improvements.

- Kubelink now discovers that a new Pod is being created directly from a Kubernetes API event. While Kubernetes (via Kubelet) continues with the process of downloading the proper images, and starting the Pod, Kubelink in CLAS mode is in parallel delivering policy for the emerging Pod to the proper C-VEN to apply.

Because CLAS stores (caches) all existing policies that have been calculated, C-VEs can get matching policies directly from the CLAS cache without needing to communicate with the PCE, which also improves convergence times.

- With Kubelink now a full intermediary between the PCE and the C-VEs, and maintaining a store of workload data, the C-VEs report traffic flow not to the PCE directly, but now to Kubelink, which "decorates" the flows with the proper Workload IDs based on IP addresses on either end, and then sends this information to the PCE.

This graphic illustrates the difference between the new CLAS architecture and the legacy non-CLAS architecture.



CLAS Degraded Mode

To ensure robustness of policy enforcement and traffic flow in the CLAS architecture, Kubelink and C-VEs can operate in *degraded mode*. If a CLAS-enabled Kubelink detects that its connection with the PCE becomes unavailable (for example, due to connectivity problems or an upgrade), Kubelink by default enters this degraded mode.

In degraded mode, new Pods of existing Kubernetes Workloads get the latest policy version cached in CLAS storage. When Kubelink detects a new Kubernetes Workload labeled the same way and in the same namespace as an existing Kubernetes Workload, Kubelink delivers the existing, cached policy to Pods of this new Workload.

If Kubelink cannot find a cached policy (that is, when labels of a new Workload do not match those of any existing Workload in the same namespace), Kubelink delivers a "fail open" or "fail closed" policy to the new Workload based on the Helm Chart parameter **degradedModePolicyFail**. The degraded mode can also be turned on or off by Helm Chart parameter as well -- **disableDegradedMode**. For more details on degraded mode, see the section on "disableDegradedMode and degradedModePolicyFail" in [Deploy with Helm Chart \[207\]](#)

Kubernetes Workloads

Starting in Kubernetes 5.0.0, the concept of Kubernetes Workloads is introduced in CLAS-enabled environments as the front-end for the Deployment of an application or service. In contrast to the Container Workload concept used previously (and still used in non-CLAS environments), Kubernetes Workloads now closely match the typical definition of workloads in Kubernetes and similar container orchestration platforms.

Therefore, Kubernetes Workloads as shown in the PCE Web UI are any workloads that have Pods, including but not limited to Deployment or DaemonSet workloads. StatefulSet, DeploymentConfig, ReplicationController, ReplicaSet, CronJob, Job, Pod, and ClusterIP are also modeled as Kubernetes Workloads in CLAS mode. Kubernetes Workloads replace Container Workloads in the non-CLAS mode.

Container Workloads

Container Workloads are reported only in non-CLAS environments. In these environments, Container Workloads are basic containers (as with Docker), or the smallest resource that can be assimilated within a container in an orchestration system (as with Kubernetes). In the context of Kubernetes and OpenShift, a Pod is a container workload. Similar to workloads reported in Illumio Segmentation, these container workloads (managed Pods) can have labels assigned to them. Container workloads with their associated Illumio labels are also displayed in Illumination. In Illumio Segmentation non-CLAS environments, containers are differentiated based on whether they are on the Pod network or the host network:

- Containers on the Pod network are considered container workloads and can be managed similarly to workloads.
- Containers sharing the host network stack (Pods that are host networked) are not considered as container workloads and therefore inherit the labels and policies of the host.

To manage container workloads, you can define the Policy Enforcement mode (Full, Selective, or Visibility Only) in container workload profiles.

**NOTE**

Container Workloads are relevant only in non-CLAS environments. CLAS-enabled environments instead use the concept of Kubernetes Workloads in Illumio Segmentation, which more closely maps to the standard Kubernetes workload concept of an application that is run on any number of dynamically-created (or destroyed) Pods.

Workloads

A workload is commonly referred to as a host OS in Illumio Segmentation. In the context of container clusters, a workload is referred to as a node in a container cluster. Usually, a Kubernetes cluster is composed of two types of nodes:

- One or more Master Node(s) - In the control plane of the cluster, these nodes control and manage the cluster.
- One or more Worker Node(s) - In the data plane of the cluster, these nodes run the application (containers).

In Illumio Segmentation, Master and Worker nodes are called workloads and are part of a container cluster. Labels and policies can be applied to these workloads, similar to any other workload that does not run containers. For a managed Kubernetes solution, only the Worker nodes are visible to the administrator and the Master nodes are not displayed in the list of Workloads.

Virtual Services

Virtual services are labeled objects and can be utilized to write policies for the respective services and the member Pods they represent.

Kubernetes services are represented as virtual services in the Illumio policy model. Kubelink creates a virtual service in the PCE for services in the Kubernetes cluster. Kubelink reports the list of Replication Controllers, DaemonSets, and ReplicaSets that are responsible for managing the Pods supporting that service.

In CLAS mode, only NodePort and LoadBalancer services are reported in the PCE UI as virtual services. Replication Controllers, DaemonSets, and ReplicaSets are no longer reported as virtual service backends in CLAS.

Container Cluster

A container cluster object is used to store all the information about a Kubernetes cluster in the PCE by collecting telemetry from Kubelink. Each Kubernetes cluster maps to one container cluster object in the PCE. Each Pod network(s) that exists on a container cluster is uniquely identified on the PCE in order to handle overlapping subnets. This helps the PCE in differentiating between container workloads that may have the same IP address but are running on two different container clusters. This differentiation is required both for Illumination and for policy enforcement.

You can see the workloads that belong to a container cluster in the PCE Web Console. This mapping between the host workload and the container cluster is done using machine-ids reported by Kubelink and C-VEN.

Container Workload Profiles

A Container Workload Profile maps to a Kubernetes namespace and defines:

- Policy Enforcement state (Full, Selective, or Visibility Only) for the Pods and services that belong to the namespace.
- Labels assigned to the Pods and services. Standard predefined label types were Role, Application, Environment, and Location. Newer releases allow you to define your own custom label types and label values for these types.

After Illumio Segmentation is installed on a container cluster, all namespaces that exist on the clusters are reported by Kubelink to the PCE and made visible using Container Workload Profiles. Each time Kubelink detects the creation of a namespace from Kubernetes, a corresponding Container Workload Profile object gets dynamically created in the PCE.

After creating a Container Workload Profile, copy the pairing key that is automatically generated and save it. Use this key for the **cluster_code** Helm Chart parameter value when installing.

Each profile can either be in a managed or unmanaged state. The default state for a profile is unmanaged. The main difference between both states:

- Unmanaged: No policy applied to Pods by the PCE and no visibility
- Managed: Policy is controlled by the PCE and full visibility through Illumination and traffic explorer

In a CLAS environment, Kubernetes Workloads are displayed only for managed Container Workload Profiles.

A Container Workload Profile is a convenient way to dynamically secure new applications with Illumio Segmentation by inheriting security policies associated with the scope of that profile.

Configure Labels for Namespaces, Pods, and Services

After Kubelink is deployed onto the Kubernetes cluster and it gets synced with the PCE, the namespaces within the cluster appear as Container Workload Profiles.

- By default, all namespaces are unmanaged, which means Illumio does not apply any inbound or outbound controls to the Pods within those namespaces.
- Any Pods or services within unmanaged namespaces do not show up in the PCE inventory or in Illumination.

Use Container Workload Profiles

The Illumio PCE administrator can change a Kubernetes namespace from unmanaged to managed by modifying the Container Workload Profile. Each profile can be modified even if the Illumio C-VEN is not yet installed on the Kubernetes nodes. If the C-VEN is deployed on the cluster nodes and Container Workload Profile is in the managed state, the Pods and services are displayed in Illumination and they inherit the labels assigned to the Kubernetes namespace. The Pods are represented in Illumio Core as Container Workloads. If Kubernetes services exist in the respective namespace, Illumio Core represents each service as an Illumio Core Virtual Service object.

This section describes how to change a namespace from unmanaged to managed and how to use edit labels and use custom annotations to add more context to your applications. This section also describes how to set enforcement boundaries for your containerized workloads.

1. Log in to the PCE UI and navigate to **Infrastructure > Container Clusters**.
2. Select the **Container Cluster** you want to manage.
3. Select the **Container Workload Profiles** tab.
4. You will see a list of all namespaces in the cluster. Select the namespace you want to manage.
5. Click **Edit**:

- a. Enter a Name (optional).
- b. Select a Management state (any state, except unmanaged).
- c. Select an Enforcement mode for how policy rules will be enforced.
- d. Select a Visibility state.
- e. Assign Labels (optional).
- f. Click **Save**.

Configure New Container Workload Profiles

A Container Workload Profile is beneficial when you want to assign labels to resources that are deployed in a namespace and also define the state of the policy created for the scope of labels assigned. A new Container Workload Profile can be created in either of the following ways:

- Dynamically created through the creation of a new namespace in the Kubernetes or OpenShift cluster. This is a *reactive* option in which the Illumio Core Administrator assigns labels and a policy state after the creation of the namespace.
- Manually pre-created to assign labels and a policy state to a namespace that will be created later on. This is a *proactive* option in which the Illumio Core Administrator assigns labels and a policy state before the creation of the namespace. This option offers the best-in-class security mechanism and authenticates each namespace created in the cluster by leveraging the concept of pairing key (same concept that Illumio Core provides in a pairing profile).



TIP

For a best-in-class security deployment, Illumio recommends to *proactively* create pairing profiles and assign labels and a policy state to them. The pairing key for each profile can be provided to the DevOps team for namespaces deployments later on.

When a Container Cluster is created for the first time in the PCE, Kubelink will report the existing namespaces or projects in the cluster. These namespaces will inherit what was defined as part of the Container Workload Profile Template for that cluster.

Dynamic Creation of a Profile

When the team managing Kubernetes or OpenShift clusters creates a namespace in a cluster, this namespace is reported immediately to the PCE via Ku-

belink. The new namespace will be listed under Container Workload Profiles and the following scenarios can occur:

- A Container Workload Profile Template exists for this cluster - The new namespace will inherit what was defined in the template, as far as Policy state and labels are concerned.
- A Container Workload Profile Template does not exist for this cluster - The new namespace will remain blank until further edited by an Illumio Core Administrator.

The example below shows a new namespace "namespace1" created in a cluster where a Container Workload Profile Template exists with a policy state set to "Build" and a partial label assignment as "Development | Cloud":



NOTE

The namespace is created by the Kubernetes or OpenShift administrator (outside the scope of Illumio Core).

For example, to edit the "namespace1" namespace:

1. Click on it and then click **Edit**.
2. Enter a *Name*.
3. Assign missing *Labels* wherever relevant or modify the existing ones. See [Labels Restrictions for Kubernetes Namespaces \[170\]](#).
4. After you are done, click **Save**.

The updates are displayed in the *Container Workload Profiles* list.

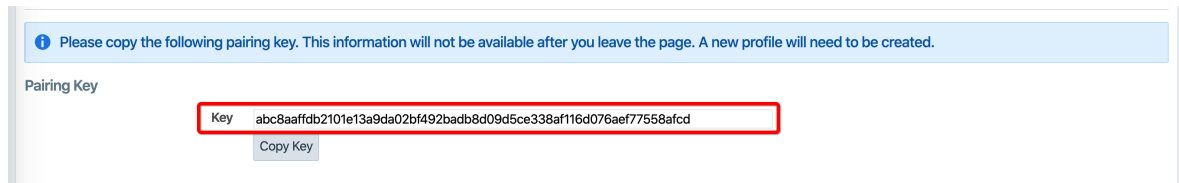
Manual Pre-creation of a Profile

To pre-create a profile:

1. In the Container Workload Profiles page, click **Add**.
2. Enter a *Name*.
3. Select the desired *Management* state.
4. Select the Enforcement state.
5. Choose a **Visibility** state. Note that Enhanced Data Collection is an optional feature that you must contact Illumio Support to enable.
6. Assign *Labels* to the profile.

See [Labels Restrictions for Kubernetes Namespaces \[170\]](#).

7. Click **Save**.



8. Click **Copy Key** and provide this key to the DevOps team, which will be used as an annotation in a namespace manifest file to authenticate this resource with the PCE.

You can view the newly-created Container Workload Profile. The status is in "Pending" state with the hourglass icon displayed next to it.

To edit the namespace configuration file to include the pairing key in order to authenticate this namespace with the PCE:

1. Navigate to `metadata: > annotations:`. If `annotations:` does not exist, create an `annotations:` section under `metadata:`.
2. Add the `com.illumio.pairing_key:` Illumio label key field under the `annotations:` section.
 - Enter the pairing key obtained during the new Container Workload Profile creation.
 - Save the file and exit.
3. Apply the change using `kubectl` commands.

An example is show below.

```
apiVersion: v1
kind: Namespace
metadata:
  name: namespace2
  annotations:
    com.illumio.pairing_key:
abc8aaffdb2101e13a9da02bf492badb8d09d5ce338af116d076aef77558afcd
```

The updates are displayed in the *Container Workload Profiles* list.

Set Enforcement

Set an Enforcement Boundary to establish how policy rules affect traffic to and from namespace workloads. Enforcement boundaries can be one of:

- **Visibility Only** - Rules are enforced on any traffic
- **Selective** - Rules are enforced only for selected traffic
- **Full** - Rules are enforced for all traffic

An enforcement boundary can be applied only to Managed workloads, which means Idle workloads cannot have an enforcement state applied to them.

You can change Enforcement for multiple profiles of the same current Enforcement level by selecting the checkboxes for the desired profiles (or by selecting the checkbox in the table heading row to select all profiles), and then hovering over the Enforcement button, which then shows a list of new Enforcement states and how many profiles will be changed to that state. Note that when you change Enforcement to Selective, then Visibility mode must be Blocked & Allowed, which is automatically done for you.

Labels Restrictions for Kubernetes Namespaces

At a high level, creating policy for containerized applications functions in the same basic way as for other types of applications running on bare-metal servers and virtual machines protected by the Illumio Core. Container workloads are assigned multi-dimensional labels to identify their roles, applications, environments, locations (RAEL), or other custom label types. These labels can then be used to apply security policies to specific parts of the containerized application environment. The PCE converts these label-based policies into rules that can be applied to the container workloads.

In previous releases, the PCE supported two options for assigning labels to container workloads:

- When creating or editing a container workload profile in the PCE web console or by using the Illumio Core REST API, an Illumio administrator assigned labels for the resources in that Kubernetes namespace.
- The Illumio administrator did not assign labels in the container workload profile. The DevOps/SRE team could use custom annotations in the service and deployment manifest files (YAML) to apply labels to the pods and services running in a namespace. On receiving this information from Kubelink, the PCE applied these labels to the container workloads, as long as the labels matched existing labels in the PCE.

These two ways of assigning labels for container workloads are sufficient for most container segmentation use cases; however, this approach lacks the flexibility with label assignment for namespaces requested by Illumio customers. However, there is an alternative in addition to those two options that still allows developers/DevOps teams to assign their own labels for Kubernetes pods and services, but at the same time restricts the list of labels that they

can assign. Illumio administrators now have a way to control which labels can be assigned by the developers managing their Kubernetes environments.

Options for Assigning Labels with a Container Workload Profile

You assign labels with container workload profiles in a number of ways:

- By creating a new container workload profile; see [Manual Pre-creation of a Profile \[168\]](#).
- By editing a container workload profile that was dynamically created in the PCE when Kubelink imported a new Kubernetes namespace; see [Dynamic Creation of a Profile \[167\]](#).
- By specifying label assignments in the default settings for the container workload profile template; see [Configure New Container Workload Profiles \[167\]](#).

Previously, four standard label types were predefined (Role, Application, Environment, and Location) for setting labels with a container workload profile. Now you can define custom label types and values in addition to these four predefined labels. You also have the following options:

- Do not allow a label for a specific label type (the “None” option).
- Allow developers to assign any label from Kubernetes for a specified label type (the “Use Container Annotations” option); so long as the labels match ones in the PCE.

In previous releases, when the PCE administrator left the labels unassigned in the GUI or through the REST API, labels specified in annotations were used. Now the “Use Container Annotations” option is selected by default for all labels in a container workload profile (provided the default settings for the cluster are not configured).

- Specify a list of labels that are allowed for that label type.
- Fix a label to a specific label for that label type (the “Assign Label” option).

Example: Assigning Labels with a Container Workload Profile

The following example shows how you can use each of the four standard predefined options:

Labels

Any container annotation label is accepted by default. You can choose to restrict container annotations to one or more labels if needed, or assign your own label [?](#)

Role ☐ Use Container Annotations ☐ Assign Label ☒ None

No label allowed

Application ☒ Use Container Annotations ☐ Assign Label ☐ None

Environment ☒ Use Container Annotations ☐ Assign Label ☐ None

Location ☐ Use Container Annotations ☒ Assign Label ☐ None

The Role label annotation (com.illumio.role) is ignored when passed at run-time and reported by Kubelink to the PCE when "Role" label is set to "None".

Adding, Editing, or Removing Labels

To add one or more labels:

1. Click a profile name, then click **Edit**.
To apply the same label edits to multiple profiles, click the checkboxes for the desired profiles (or click the topmost checkbox in the table heading to select all profiles), then click **Edit Labels**.
2. Under the Labels heading, add a label type by clicking the field under the Label Type heading and then choose a label type from the list.
3. Choose a Label Assign Type:
 - **Use Container Annotations** - Use label values for container annotations. See the topic "Using Annotations" for more information.
 - **Assign Label** - Explicitly set labels from the values configured for this label type.
 - **No Label Allowed** - Prevent this label type from being used in this profile.
4. Specify labels for this label type by clicking the field under Labels Allowed/Label Assign, and choosing a label from the list.
Any label type defined from annotations or explicit assignments must also have a label value specified in order to add the label definitions to the profile.
5. Click **Save** when finished.

To remove label types (and their associated label values):

1. Click the profile name.

2. Click **Edit**.
3. Under the Labels heading, choose one or more types to delete from this profile by clicking the checkboxes in front of the Label Type name.
4. Click **Remove**.

To remove or change label values:

1. Click the desired profile name.
2. Click **Edit**.
3. In the Labels table, remove a label value by clicking the small "x" near the label name under the Labels Allowed/Label Assign column.
You can replace or add label values by clicking the **Select Label** or **Select Labels** field under Labels Allowed/Label Assign column, and then choosing the new label (or in the case of Annotations, multiple labels).
4. Click **Save**.

Possible Labels for the Example

- Developers can specify any label for Applications, so long as the label matches a preexisting label in the PCE.
- For Environment, a list of two labels (env1 and env2) is available. Developers can set either of these labels in Kubernetes. If a developer sets another value for the Environment label as a Kubernetes annotation, the PCE considers it invalid and, as a result, a label is not assigned to that label key. Because the wrong label is assigned, the policy will not allow expected traffic from other services or applications with the Environment label env1.
- The Location label is fixed as the loc1 label. If a developer assigns another Location label (for example, loc3, which is a label in the PCE) or the developer leaves the Location label empty, the PCE overrides what the developer has specified in the annotation and the PCE assigns loc1 for the Location label.

The label assignments for that namespace appears in the Container Clusters list in the PCE web console.

For this example, you can see the label assignments mirrored in the Kubernetes annotation for the namespace:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app-1
  namespace: demo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: app1
  template:
    metadata:
      annotations:
        com.illumio.role: role1
        com.illumio.app: app1
        com.illumio.env: env1
        com.illumio.loc: loc100
    labels:
      app: app1
    spec:
      containers:
        - name: app1
          image: kodekloud/webapp-conntest
          ports:
            - containerPort: 8080
```

Kubernetes Annotation for Namespaces

where developers set **role1**, **app1**, **env1**, and **loc100** for the labels in the annotations.

Kubelink passes this data to the PCE at runtime. The PCE ignores the Role label because it's not allowed in the profile. The PCE accepts the Application and Environment labels. It ignores the **loc100** label and uses **loc1** instead.

In the Container Workloads tab, you can see how the label assignments are applied for the pod in this example.

**NOTE**

If a developer sets another value for the Environment label as a Kubernetes annotation, the PCE considers it invalid and, as a result, a label is not assigned to that label key. Because the wrong label is assigned, the policy will not allow expected traffic from other services or applications.

For example, if developers leave the Environment label empty or specify env100 in the Kubernetes annotations, the following labels are used for the namespace, and there are no policy for applications or services with the Environment label env1.

Effect of Upgrading the PCE to Core 21.1.0 or Later

After upgrading your PCE to Core 21.1.0 or later, the labels assignments for your Kubernetes namespaces are not impacted operationally. However, you will see changes in the PCE web console and in the REST API.

- The values set in the PCE in the previous Core release are unchanged and the “Assign Label” option is selected in the PCE web console and through the REST API.
- The values left open so that container annotations were used for label assignments are updated to the “Use Container Annotations” option and the label assignments won’t be restricted by any settings in the PCE web console or through the REST API.

Container Workload Profile Restriction**WARNING**

There's a 1:1 mapping between the pairing key of a container workload profile and the Kubernetes namespace that uses this pairing key.

Two namespaces in Kubernetes that use the same pairing key for a given container profile are not supported or accepted. Such use would cause an error in the container cluster status.

Using Annotations



NOTE

Illumio annotations operate differently in CLAS-mode clusters (optionally available starting in Illumio Core for Kubernetes version 5.0.0) than in previous legacy (non-CLAS) environments.

The initial portion of this topic describes how to use annotations in legacy non-CLAS clusters. After this initial portion, in the latter part of this topic, you can find information about using annotations in CLAS-mode clusters, described in the section [Using Annotations in CLAS \[185\]](#).

When assigning labels, you can assign no labels, some labels, or all labels to the namespace. If there is a label that is not assigned, then you can insert annotations in the Deployment configuration (or application configuration) to assign labels. If there is a conflict between a label assigned via the Container Workload Profile and the annotations in the deployment configuration, the label from the Container Workload Profile overrides the Deployment configuration file. This security mechanism ensures that a malicious actor cannot spoof labels and get a preferential security policy based on a different scope. Regardless of how you assign labels, it is not required for Pods or services to have all labels in order for the PCE to manage them.

To manually annotate the different resources created in a Kubernetes namespace or OpenShift project, use the steps described in the sections below.

Deployments

1. Edit the Deployment configuration file:
 - a. Navigate to `spec: > template: > metadata: > annotations:`. If `annotations:` does not exist, create an `annotations:` section underneath `metadata:`.
 - b. The annotation can support any Illumio label key fields, including user-defined label types, as well as the standard set of predefined Illumio labels:
 - `com.illumio.role:`
 - `com.illumio.app:`
 - `com.illumio.env:`

- `com.illumio.loc:`
 - c. Fill in the appropriate labels.
 - d. Save the file and exit.
2. Propagate your changes to all Pods.

Services

1. Edit the Deployment configuration file:
 - a. Navigate to `metadata: > annotations:`. If `annotations:` does not exist, create an `annotations:` section underneath `metadata:`.
 - b. The following Illumio label key fields can be under the `annotations:` section.
 - `com.illumio.role:`
 - `com.illumio.app:`
 - `com.illumio.env:`
 - `com.illumio.loc:`
 - c. Fill in the appropriate labels.
 - d. Save the file and exit.
2. Propagate your changes to all Pods.



IMPORTANT

When using the annotations method, you should redeploy the Pods or services after saving the changes to the configuration files by using the `kubectl apply` command.

Annotation Examples

Below are examples of namespaces, Pods, and services that use label assignments using either Container Workload Profiles or Container Workload Profiles with annotation insertion.

In the example shown below:

- Kubernetes default services or control plane Pods exist within namespaces such as, `kube-system`. They will inherit the Application, Environment, and Location labels from what has been configured in the Container Workload Profile(s). Kubelink is part of the `illumio-system` namespace, and because the Role label is left blank on the `illumio-system` namespace, you should assign a Role to Kubelink using annotations in the manifest file.
- A new `app1` namespace that contains two different Deployments or a two-tier application (Web and Database) is deployed. To achieve tier-to-tier

segmentation across the application they will need different Role labels. Therefore, a Role label should be inserted into the annotations of each Deployment configuration.

A snippet of the **illumio-kubelink** Deployment configuration file is shown below, and the "Kubelink" Role label is inserted under the **spec: > template: > metadata: > annotations:** section:

illumio-kubelink-kubernetes.yml

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: illumio-kubelink
  template:
    metadata:
      annotations:
        com.illumio.role: Kubelink
      labels:
        app: illumio-kubelink
    spec:
      nodeSelector:
#       node-role.kubernetes.io/master: ""
#       serviceAccountName: illumio-kubelink
      tolerations:
        - key: node-role.kubernetes.io/master
          effect: NoSchedule
```

A snippet of the **app1**'s Web Deployment configuration file is shown below, and the "Web" Role label is inserted under the **spec: > template: > metadata: > annotations:** section:

shopping-cart-web.yml


```
spec:
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: webapp1
      tier: frontend
  strategy:
    activeDeadlineSeconds: 21600
    resources: {}
    rollingParams:
      intervalSeconds: 1
      maxSurge: 25%
      maxUnavailable: 25%
      timeoutSeconds: 600
      updatePeriodSeconds: 1
    type: Rolling
  template:
    metadata:
      annotations:
        com.illumio.role: Web
      creationTimestamp: null
      labels:
```

A snippet of the **app1**'s Database Deployment configuration file is shown below and the "Database" Role label is inserted under the **spec: > template: > metadata: > annotations:** section:

shopping-cart-db.yml

```
spec:
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: redis
      role: slave
      tier: backend
  strategy:
    activeDeadlineSeconds: 21600
    recreateParams:
      timeoutSeconds: 600
    resources: {}
    type: Recreate
  template:
    metadata:
      annotations:
        com.illumio.role: Database
      creationTimestamp: null
      labels:
```

Below is the final outcome of the label assignment from the example.

Container Workloads - MyK8sCluster							
Summary Container Workload Profiles Workloads Container Workloads Service Backends							
Refresh							
Select properties to filter view							
Policy State	Policy Sync	Namespace/Project	Name	Role	Application	Environment	Location
Build	Syncing	app1	web-frontend-655857999f-862gt	Web	App1	Development	Cloud
Build	Syncing	app1	redis-slave-6bb554dbcc-9759s	Database	App1	Development	Cloud
Build	Active	app1	redis-slave-6bb554dbcc-8mpv9	Database	App1	Development	Cloud
Build	Active	app1	redis-master-c67979c8c-fhwpl	Database	App1	Development	Cloud
Build	Syncing	app1	web-frontend-655857999f-sxqpt	Web	App1	Development	Cloud
Build	Syncing	app1	web-frontend-655857999f-tp5gq	Web	App1	Development	Cloud
Build	Active	kube-system	coredns-58687784f9-znn9j		Kube-System	Development	Cloud
Build	Active	kube-system	dns-autoscaler-79599df498-m55mg		Kube-System	Development	Cloud
Build	Active	kube-system	coredns-58687784f9-h4pp2		Kube-System	Development	Cloud
Build	Active	illumio-system	illumio-kubelink-67d8d9f6-7jvc6	Kubelink	System	Development	Cloud
Build	Active	kubernetes-dashboard	kubernetes-dashboard-7b5bf5d559-znnvq		Dashboard	Development	Cloud
Build	Active	kubernetes-dashboard	dashboard-metrics-scraper-566cddb686-vmxv2		Dashboard	Development	Cloud

In Illumination Map, the application groups will appear differently if you've assigned labels on resources in the cluster.

DaemonSets and ReplicaSets

The steps described in the above section apply only to services in Kubernetes and OpenShift which are bound to Deployment or DeploymentConfig (existing deployments). This is because Kubelink depends on the Pod hash tem-

plates to map resources together, templates that DaemonSet and ReplicaSet configurations do not have. If you discover Pods derived from DaemonSet or ReplicaSet configurations and also discover services bound to those Pods, then Kubelink will **not** automatically bind the virtual service and service backends for the PCE. The absence of this binding will create limitations with Illumio policies written against the virtual service.

To work around this limitation for DaemonSets and ReplicaSets follow the steps below.

1. Generate a random uuid using the **uuidgen** command (on any Kubernetes or OpenShift node, or your laptop).
2. Copy the output of the **uuidgen** command.
3. Edit the DaemonSet or ReplicaSet YAML configuration file.
4. Locate the **spec: > template: > metadata: > labels:** field in the YAML file and create the **Pod-template-hash:** field under the **labels:** section.
5. Paste the new uuid as the value of the **Pod-template-hash:** field.
6. Save the changes.

Repeat steps 1 through 6 for each DaemonSet or ReplicaSet configuration.

The examples below generate a random **pod-template-hash** value and applies it to a DaemonSet configuration.

```
$ uuidgen
9e6f8753-d8ac-11e8-9999-0050568b6a18
```

```
$ cat nginx-ds.yml
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: nginx-webserver
spec:
  template:
    metadata:
      labels:
        app: nginx-webserver
        pod-template-hash: 9e6f8753-d8ac-11e8-9999-0050568b6a18
    spec:
      containers:
        - name: webserver
          image: rstarmer/nginx-curl
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 80
```

Static Pods

Another way of deploying Pods without Deployments or ReplicaSet is by using 'Static Pods'. In this case, a Pod is spun up by not depending on the API server and is managed by an individual node's Kubelet. Static Pods are used to spin up control-plane components such as kube-apiserver, controller-manager, and scheduler. Static Pods are useful if you want a Pod to be running even if the Kubernetes control-plane components fail. Unlike Naked Pods, if a Static Pod is not functional, kubelet spins up a new Static Pod automatically by looking at the manifest file in the `/etc/kubernetes/manifests` directory.

Services for such Pods can also be created without any selectors. In which case, you need to manually create the **EndPoint** resources for such services without a selector. For example, the default 'kubernetes' service in the default namespace which binds to the API-Server Pod running on HostNetwork.

If you create Static Pods on an overlay network, you need to create a service without selectors and manually create **EndPoint** resource to map the Pod to see the Container Workload and the Virtual Service on the PCE. You will not see any bindings or backends for this Virtual Service. In order to bind the Static Pods to the Virtual Service, use the '**com.illumio.service_uids**' annotation in the Static Pods manifest and configure the service without selectors and manually create the EndPoints. Once the '**com.illumio.service_uids**' annotation is used, you can bind the Container Workloads to its Virtual Service.

Sample code: Place the Static Pod manifest in the `/etc/kubernetes/manifests` directory

```
[root@qvc-k8s-027-master01 manifests]# pwd
/etc/kubernetes/manifests

[root@qvc-k8s-027-master01 manifests]# cat network-tool.yml
apiVersion: v1
kind: Pod
metadata:
  name: nw-tool1
  annotations:
    com.illumio.service_uids: <numerical-value>
spec:
  containers:
  - name: nw-tool1
    image: praqma/network-multitool
    args: [/bin/sh, -c, 'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 10; done']
    imagePullPolicy: IfNotPresent
    restartPolicy: Always

[root@qvc-k8s-027-master01 ~]# cat nw-tool-endpoint.yaml
apiVersion: v1
kind: Endpoints
metadata:
  name: nw-tool-svc
  namespace: default
subsets:
- addresses:
  - ip: <ip-value>
  ports:
  - name: http
    port: 80
    protocol: TCP

[root@qvc-k8s-027-master01 ~]# cat nw-tool-svc.yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2020-05-18T18:39:19Z"
  labels:
    app: nw-tool
  name: nw-tool-svc
  namespace: default
  resourceVersion: "29308511"
  selfLink: /api/v1/namespaces/default/services/nw-tool-svc
  uid: <numerical-value>
spec:
```

```

clusterIP: <ip-value>
ports:
- name: http
  port: 80
  protocol: TCP
  targetPort: 80
sessionAffinity: None
type: ClusterIP
status:
  loadBalancer: {}
[root@qvc-k8s-027-master01 ~]#

```



IMPORTANT

In the above code sample, you need to modify the following two values based on your configuration:

- uid: <numerical-value>
- clusterIP: <ip-value>

Using Annotations in CLAS

Illumio annotations in CLAS-mode environments are specified on the Kubernetes Workload, and not on a Pod's template, as is done in legacy non-CLAS environments. This distinction follows from the concept of the Kubernetes Workload in the PCE UI introduced with CLAS-mode, which maps directly to the native Kubernetes concept of a workload resource (that is, Deployments, ReplicaSets, and the like).

Therefore, Kubernetes Workloads on the PCE should be labelled based on the corresponding workload annotations in Kubernetes, instead of on individual pod template annotations in Kubernetes.

This labelling distinction prevents confusion, because Pods from a single Deployment can have different annotations:

```
# kubectl get pod azure-vote-front-6fd8b9b657-6pv8t -n voting-
app -o jsonpath='{.metadata.annotations}' | tr ',' '\n' | grep
com.illumio
"com.illumio.app":"A-VotingApp"
"com.illumio.env":"E-Production"
"com.illumio.loc":"Azure"

# kubectl get pod azure-vote-front-6fd8b9b657-npppz -n voting-
app -o jsonpath='{.metadata.annotations}' | tr ',' '\n' | grep
com.illumio
"com.illumio.app":"A-VotingApp"
"com.illumio.env":"Development"
"com.illumio.loc":"Amazon"
"com.illumio.role":"R-Frontend"}
```

Migration

Workloads reporting supports both: Pod template annotations and workload annotations. However, the priority is put on workload, if it contains at least one annotation with a **com.illumio.** prefix.

In the following example, annotations are specified in **metadata.annotations:** and **spec.template.metadata.annotations:**. Annotations specified in **metadata.annotations:** are prioritized.

The resulting annotations mapped to labels are: **app=A-VotingApp** and **env=E-Test** (no merging between the sets of annotations occurs).


```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    com.illumio.app: A-VotingApp
    com.illumio.env: E-Test
  name: test-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test-pod
  template:
    metadata:
      annotations:
        com.illumio.loc: Amazon
        com.illumio.env: test-env
      labels:
        app: test-pod
    spec:
      containers:
        - name: test-pod
          image: nginx:1.14.2
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 80
```

Deployment with Helm Chart (Kubernetes 3.0.0 and Later)

After you set up your clusters, perform these steps.

**NOTE**

Illumio Segmentation for Kubernetes 3.0.0 and later is a combined release of C-VEN and Kubelink. Starting with C-VEN 21.5.17 and Kubelink 3.0, C-VEN and Kubelink 3.0 will be only used through the combined release. A Helm Chart (via quay.io) is used to deploy all necessary product components. If you are deploying C-VEN 21.5.15 or earlier, follow the deployment instructions in [Deployment for C-VEN Versions 21.5.15 or Earlier \[226\]](#).

The installation process is mostly the same for Kubernetes and OpenShift, with a few exceptions. A dedicated section is created for Kubernetes or OpenShift wherever required.

You can manually deploy components with YAML manifests that are first generated by Helm but are not actually deployed with a Helm chart. See [Generating YAML Manifests for Manual Deployment \[224\]](#).

Helm Chart Deployment Overview

Starting with the Illumio Core for Kubernetes 3.0.0 release and later, the product (including C-VEN and Kubelink) is now deployed by using a Helm Chart. The product components and the Helm Chart are downloaded from a public container repository: <https://quay.io/repository/illumio/illumio>.

Use these steps to deploy Helm Chart:

1. Deploy and configure your PCE. See [Install the PCE and UI](#).
2. Create a container cluster. See [Create a Container Cluster in the PCE \[201\]](#).
3. Create a pairing profile. See [Create a Pairing Profile for Your Cluster Nodes \[203\]](#).
4. Deploy Helm Chart. See [Deploy with Helm Chart \[207\]](#). At this stage you can optionally map existing Kubernetes labels to Illumio labels.

Follow the sections in this order, including the requirements and environment preparations described next.

Host and Cluster Requirements

To deploy Illumio containers into your environment, you must meet the following requirements.

Supported Configurations for On-premises and IaaS

For full details on all supported configurations for Illumio Core for Kubernetes version 3.0.0 and later, see the [Kubernetes Operator OS Support and Dependencies](#) page on the Illumio Support Portal (under **Software** > **OS Support**).

Privileges

The Helm Chart deployment process automatically sets all necessary privileges. The privileges listed below must be provided on host-level and cluster-level for the respective components. They are listed here for reference.

Host-Level

C-VEN

C-VEN requires the following privileges on the host:

- C-VEN is a privileged container and requires access to the following system calls:
 - NET_ADMIN
 - SYS_MODULE
 - SYS_ADMIN
- C-VEN requires persistent storage on the host to write iptables or nftables rules and logs.
- C-VEN mounts volumes on the local host to be able to operate (mount points may differ depending on the orchestration platform).

Kubelink

Kubelink does not require specific privileges on the host because Kubelink:

- is not a privileged container
- is a stateless container
- does not require persistent storage

Cluster-Level

Namespace

C-VEs and Kubelink are deployed in the `illumio-system` namespace.

C-VEN

C-VEN requires the following privileges on the cluster:

- C-VEN uses the `illumio-ven` ServiceAccount.

Kubelink

Kubelink requires the following privileges on the cluster:

- Kubelink creates a new Cluster Role to list and watch events occurring on the Kubernetes API server for the following elements:
 - `nodes`
 - `hostsubnets`
 - `replicationcontrollers`
 - `services`
 - `replicasets`
 - `daemonsets`
 - `namespaces`
 - `statefulsets`
- Kubelink uses the `illumio-kubelink` ServiceAccount.

Prepare Your Environment

You need to do these steps before creating clusters or pairing profiles in the PCE, or subsequent deployment.



CAUTION

If the prerequisite steps are not done before deployment, then containerized environments and Kubelink can get disrupted.

Unique Machine ID

Some of the functionality and services provided by the Illumio C-VEN and Kubelink depend on the Linux machine ID of each Kubernetes cluster node. Each machine ID must be unique in order to take advantage of the functionality. By default, the Linux operating system generates a random machine ID to give each Linux host uniqueness. However, there are cases when machine IDs can be duplicated across machines. This is common across deployments that clone machines from a golden image, for example, spinning up virtual

machines from VMware templates, creating compute instances from a reference image, or from a template from a Public Cloud source.



IMPORTANT

Illumio Core requires a unique machine ID on all nodes. This issue is more likely to occur with on-premises or IaaS deployments, rather than with Managed Kubernetes Services (from Cloud Service Providers). For more information, see "Troubleshooting".

Create Labels

For details on creating labels, see "Labels and Label Groups" in Security Policy Guide. The labels shown below are used in examples throughout this document. You are not required to use the same labels.

Name	Label Type
Kubernetes Cluster	Application
OpenShift Cluster	Application
Production	Environment
Development	Environment
Data Center	Location
Cloud	Location
Kubelink	Role
Node	Role
Control Plane Node (formerly Master)	Role
Worker	Role

**NOTE**

Starting in Illumio Core for Kubernetes 4.2.0, you can map Kubernetes labels to Illumio labels by using a Container Resource Definition in your **illumio-values.yaml** with the Helm Chart deployment. See [Map Kubernetes Labels to Illumio Labels \[204\]](#).

Create a ConfigMap to Store Your Root CA Certificate

This section describes how to implement Kubelink with a PCE using a certificate signed by a private PKI. It describes how to configure Kubelink and C-VEN to accept the certificate from the PCE signed by a private root or intermediate Certificate Authority (CA), and ensure that Kubelink can communicate in a secure way with the PCE.

Prerequisites

- Access to the root CA to download the root CA certificate
- Access to your Kubernetes cluster and can run **kubectl** commands
- Correct privileges in your Kubernetes cluster to create resources like ConfigMaps, secrets, and Pods
- Access to the PCE web console as a Global Organization Owner

Resource Allocation Guidelines

New resource allocation guidelines have been developed to help configure deployments to achieve optimal performance and cost-efficiency.

These guidelines are grouped into the following general deployment sizes:

- **Small-scale:** Customers with limited Kubernetes deployments and moderate workloads.
- **Medium-scale:** Customers with moderate-sized Kubernetes environments and growing workloads.
- **Large-scale:** Customers with extensive Kubernetes deployments and high-performance requirements.

The following variables determine the deployment sizes listed above:

- Number of nodes per cluster
- Total number of workloads per cluster
- Total policy size per cluster

Set the **resources** values in the appropriate pod spec (Kubelink or C-VEN) **yaml** file under the **storage** section, as shown in the following example:

```
storage:
  sizeGi: 1
  resources:
    limits:
      memory: 600Mi
    requests:
      memory: 500Mi
      cpu: 500m
```

If you have two parameters that match one category, and a third parameter that matches another, it's important to select the category based on the highest value among them.

For instance, if the number of nodes per cluster is 8, and the total number of Kubernetes workloads is 500, but the average size of the policy is 1 Gi, the resource allocation should align with the large-scale resource allocation. This ensures that your resources are appropriately scaled to meet the demands of your workloads, optimizing performance and stability.

In practice, monitor these resources, and if usage is at 80% of these limits, then consider increasing.

NOTE that amounts are expressed in mebibytes (Mi) and gibibytes (Gi) and not in megabytes (MB) or gigabytes (GB).

Small-scale resource allocation

Customer Category	Nodes per Cluster	Total K8s Workloads	Total Policy Size	
Small-scale	1 - 10	0 - 1000	0 - 1.5 Mi	
Resources		C-VEN	Kubelink	Storage
Requests	CPU	0.5	0.5	0.5
Requests	memory	600 Mi	500 Mi	500 Mi
Limits	CPU	1	1	1
Limits	memory	700 Mi	600 Mi	600 Mi
Volumes	size limits	n/a	n/a	1 Gi

Medium-scale resource allocation

Customer Category	Nodes per Cluster	Total K8s Workloads	Total Policy Size	
Medium-scale	10 - 20	1000 - 5000	1.5 Mi - 500 Mi	
Resources		C-VEN	Kubelink	Storage
Requests	CPU	2	2	1
Requests	memory	3 Gi	5 Gi	5 Gi
Limits	CPU	3	2	2
Limits	memory	5 Gi	7 Gi	7 Gi
Volumes	size limits	n/a	n/a	5 Gi

Large-scale resource allocation

Customer Category	Nodes per Cluster	Total K8s Workloads	Total Policy Size	
Large-scale	20+	5000 - 8000	500 Mi - 1.5 Gi	
Resources		C-VEN	Kubelink	Storage
Requests	CPU	2	3	1
Requests	memory	6 Gi	10 Gi	10 Gi
Limits	CPU	3	4	2
Limits	memory	8 Gi	12 Gi	12 Gi
Volumes	size limits	n/a	n/a	10 Gi

Download the Root CA Certificate

Before you begin, ensure that you have access to the root CA certificate. The root CA certificate is a file that can be exported from the root CA without compromising the security of the company. It is usually made available to external entities to ensure a proper SSL handshake between a server and its clients.

You can download the root CA certificate in the CRT format on your local machine. Below is an example of a root CA certificate:

```
$ cat root.democsa.illumio-demo.com.crt
-----BEGIN CERTIFICATE-----
MIIGSzCCBD0gAwIBAgIUAPw0NfPAivJW4YmKZ499eHZH3S8wDQYJKoZIhvcNAQEL
---output suppressed---
wPG0lug46K1EPQqMA7YshmrwOd6ESy6RGNFFZdhk9Q==
-----END CERTIFICATE-----
```

You can also get the content of your root CA certificate in a readable output format by using the following command:

```

$ openssl x509 -text -noout -in ./root.democa.illumio-
demo.com.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
fc:34:35:f3:c0:8a:f2:56:e1:89:8a:67:8f:7d:78:76:47:dd:2f
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=California, L=Sunnyvale, O=Illumio,
        OU=Technical Marketing, CN=Illumio Demo Root
        CA 1/emailAddress=tme-team@illumio.com
        Validity
            Not Before: Jan 20 00:05:36 2020 GMT
            Not After : Jan 17 00:05:36 2030 GMT
        Subject: C=US, ST=California, L=Sunnyvale, O=Illumio,
        OU=Technical Marketing, CN=Illumio Demo Root
        CA 1/emailAddress=tme-team@illumio.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (4096 bit)
            Modulus:
00:c0:e5:48:7d:97:f8:5b:8c:ef:ac:16:a8:8c:aa:
68:b8:48:af:28:cd:17:8f:02:c8:82:e9:69:62:e2:
89:2b:be:bd:34:fc:e3:4d:3f:86:5e:d7:e6:89:34:
71:60:e6:54:61:ac:0f:26:1c:99:6f:80:89:3f:36:
b3:ad:78:d1:6c:3f:d7:23:1e:ea:51:14:48:74:c3:
e8:6e:a2:79:b1:60:4c:65:14:2a:f1:a0:97:6c:97:
50:43:67:07:b7:51:5d:2c:12:49:81:dc:01:c9:d1:
57:48:32:2e:87:a8:d2:c0:b9:f8:43:b2:58:10:af:
54:59:09:05:cb:3e:f0:d7:ef:70:cc:fc:53:48:ee:
a4:a4:61:f1:d7:5b:7c:a9:a8:92:dc:77:74:f4:4a:
c0:4a:90:71:0f:6d:9e:e7:4f:11:ab:a5:3d:cd:4b:
8b:79:fe:82:1b:16:27:94:8e:35:37:db:dd:b8:fe:

```

```

fa:6d:d9:be:57:f3:ca:f3:56:aa:be:c8:57:a1:a8:
c9:83:dd:5a:96:5a:6b:32:2d:5e:ae:da:fc:85:76:
bb:77:d5:c2:53:f3:5b:61:74:e7:f3:3e:4e:ad:10:
7d:4f:ff:90:69:7c:1c:41:2f:67:e4:13:5b:e6:3a:
a3:2f:93:61:3b:07:56:59:5a:d9:bc:34:4d:b3:54:
b5:c6:e5:0a:88:e9:62:7b:4b:85:d2:9e:4c:ee:0b:
0d:f4:72:b1:1b:44:04:93:cf:cc:bb:18:31:3a:d4:
83:4a:ff:15:42:2d:91:ca:d0:cb:36:d9:8d:62:c0:
41:59:1a:93:c7:27:79:08:94:b2:a2:50:3c:57:27:
33:af:f0:b6:92:44:49:c5:09:15:a7:43:2a:0f:a9:
02:61:b3:66:4f:c3:de:d3:63:1e:08:b1:23:ea:69:
90:db:e8:e9:1e:21:84:e0:56:e1:8e:a1:fa:3f:7a:
08:0f:54:0a:82:41:08:6b:6e:bb:cf:d6:5b:80:c6:
ea:0c:80:92:96:ab:95:5d:38:6d:4d:da:38:6b:42:
ef:7c:88:58:83:88:6d:da:28:62:62:1f:e5:a7:0d:
04:9f:0d:d9:52:39:46:ba:56:7c:1d:77:38:26:7c:
86:69:58:4d:b0:47:3a:e2:be:ee:1a:fc:4c:de:67:
f3:d5:fe:e6:27:a2:ef:26:86:19:5b:05:85:9c:4c:
02:24:76:58:42:1a:f8:e0:e0:ed:78:f2:8f:c8:5a:
20:a9:2d:0b:d4:01:fa:57:d4:6f:1c:0a:31:30:8c:
32:7f:b0:01:1e:fe:94:96:03:ee:01:d7:f4:4a:83:
f5:06:fa:60:43:15:05:9a:ca:88:59:5c:f5:13:09:
      82:69:7f
Exponent: 65537 (0x10001)

```

X509v3 extensions:

X509v3 Subject Key Identifier:

3D:3D:3D:61:E6:88:09:FE:34:0F:1D:5E:5E:52:72:71:C7:DE:15:92

X509v3 Authority Key Identifier:

keyid:3D:3D:3D:61:E6:88:09:FE:34:0F:1D:5E:5E:52:72:71:C7:
DE:15:92

X509v3 Basic Constraints: critical

CA:TRUE

X509v3 Key Usage: critical

Digital Signature, Certificate Sign, CRL Sign

Signature Algorithm: sha256WithRSAEncryption

28:24:86:91:a6:4a:88:e4:8d:6b:fc:67:2a:68:08:67:35:e5:
a6:77:ff:07:4b:89:53:99:2e:6d:95:df:12:81:28:6a:8e:6f:
5a:98:95:5b:4a:21:ae:f0:20:a4:4e:06:b2:4e:5a:67:c1:6a:
06:f1:0f:c1:f7:7e:f2:e0:b3:9d:d8:54:26:6a:b2:1c:19:b8:
b5:5c:c7:03:6b:f7:70:9e:72:85:c9:29:55:f9:f4:a4:f2:b4:
3b:3d:ce:25:96:67:32:1e:8d:e2:00:22:55:4b:05:4f:ee:0e:
67:ac:db:1b:61:da:5f:9c:10:1c:0c:05:66:c0:5b:5f:b9:95:
59:a9:58:5b:e7:69:ac:b0:bd:b3:c2:a3:35:58:01:a4:ff:c0:
8d:ac:1c:19:21:41:50:fb:8e:e0:f5:a9:ad:ec:de:cb:53:04:
a9:d8:ac:76:8a:09:0d:7c:c6:1a:bc:06:74:bb:10:1c:aa:07:
f6:cb:b2:1b:0c:0c:65:03:45:2b:51:d5:6e:a0:4d:91:ce:c5:
ed:8d:a9:e7:f6:37:7d:ab:1b:a4:a2:a3:3b:76:17:5b:d9:3a:
9c:c1:df:cc:cd:a0:b0:a9:5c:74:61:d7:a0:1d:04:67:68:ee:
a6:7b:1e:41:a4:02:fc:65:9e:e3:c1:c2:57:b2:2e:b0:ff:a9:
86:82:35:4d:29:b2:fe:74:2e:b8:37:5d:2b:e8:69:f2:80:29:
19:f1:1e:7a:5d:e3:d2:51:50:46:30:54:7e:b8:ad:59:61:24:
45:a8:5a:fe:19:ff:09:31:d0:50:8b:e2:15:c0:a2:f1:20:95:
63:55:18:a7:a2:ad:16:25:c7:a3:d1:f2:e5:be:6d:c0:50:4b:
15:ac:e0:10:5e:f3:7b:90:9c:75:1a:6b:e3:fb:39:88:e4:e6:
9f:4c:85:60:67:e8:7d:2e:85:3d:87:ed:06:1d:13:0b:76:d7:
97:a5:b8:05:76:67:d6:41:06:c5:c0:7a:bd:f4:c6:5b:b2:fd:
23:6f:1f:57:2e:df:95:3f:26:a5:13:4d:6d:96:12:56:98:db:
2e:7d:fd:56:f5:71:b7:19:2b:c9:de:2d:b9:c8:17:cc:20:de:
7c:19:7a:aa:12:97:1c:80:b7:d3:67:d3:b7:a7:96:f0:c9:4d:
f5:8b:0e:10:3b:b9:4e:09:90:5a:3b:51:c9:48:a2:ca:9f:db:
72:44:87:59:db:49:fa:75:44:b5:f6:7f:c5:26:e1:01:ae:7b:
6f:4a:75:d1:b5:b3:68:c0:31:48:f8:5c:06:c0:f1:b4:96:e8:
38:e8:ad:44:3d:0a:8c:03:b6:2c:86:6a:f0:39:de:84:4b:2e:
91:18:d1:45:65:d8:64:f5

Create a ConfigMap in the Kubernetes Cluster

After downloading the certificate locally on your machine, create a ConfigMap in the Kubernetes cluster that will copy the root CA certificate on your local machine into the Kubernetes cluster.

To create a ConfigMap, use the following command:

```
$ kubectl -n illumio-system create configmap root-ca-config \
  --from-file=./certs/root.democa.illumio-demo.com.crt
```

The **--from-file** option points to the path where the root CA certificate is stored on your local machine.

To verify that ConfigMap was created correctly, use the following command:

```
$ kubectl -n illumio-system create configmap root-ca-config \
> --from-file=./certs/root.democa.illumio-demo.com.crt
configmap/root-ca-config created
$
$ kubectl -n illumio-system get configmap
NAME                                DATA    AGE
root-ca-config                     1        12s
$
$ kubectl -n illumio-system describe configmap root-ca-config
Name:                             root-ca-config
Namespace:                         illumio-system
Labels:                           <none>
Annotations:                       <none>

Data
===
root.democa.illumio-demo.com.crt:
----
-----BEGIN CERTIFICATE-----
MIIGSzCCBD0gAwIBAgIUAPw0NfPAivJW4YmKZ499eHZH3S8wDQYJKoZIhvcNAQEL
L
---output suppressed---
wPG0lug46K1EPQqMA7YshmrwOd6ESy6RGNFFZdhk9Q==
-----END CERTIFICATE-----

Events:                            <none>
$
```

root-ca-config is the name used to designate the ConfigMap. You can modify it according to your naming convention.

Configure Calico in Append Mode

In case your cluster is configured with Calico as the network plugin (usually for Kubernetes and not for OpenShift), both Calico and Illumio Core will write iptables or nftables rules on the cluster nodes.

- Calico - Needs to write iptables or nftables rules to instruct the host how to forward packets (overlay, IPIP, NAT, and so on).
- Illumio Core - Needs to write iptables or nftables rules to secure communications between nodes and/or Pods.

You should establish a hierarchy to make the firewall coexistence work smoothly because Illumio Core and Calico will write rules at the same time. By default, both solutions are configured to insert rules first in the iptables or nftables chains/tables and Illumio Core will remove other rules added by a third-party software (in the Exclusive mode).

To allow Calico to write rules along with Illumio without flushing rules from one another, you should:

- Configure Illumio to work in Firewall Coexistence mode (default for workloads that are part of a container cluster).
- Configure Calico to work in Append mode (default is Insert mode).

To configure Calico to work in Append mode with iptables or nftables:

1. Edit the Calico DaemonSet:

```
kubectl -n kube-system edit ds calico-node
```

2. Locate the **spec:** > **template:** > **spec:** > **containers:** section inside the **YAML** file and change ChainInsertMode by adding the following code block:

```
- name: FELIX_CHAININSERTMODE
  value: Append
```

3. Save your changes and exit.
4. Kubernetes will restart all Calico Pods in a rolling update.

For more information on changing Calico ChainInsertMode, see [Calico documentation](#).

Create a Container Cluster in the PCE

To provide visibility and enforcement to your containerized environment, you first need to create a container cluster in the PCE. Each container cluster maps to an existing Kubernetes or OpenShift cluster.

Create a Container Cluster

To create a new container cluster:

1. Log into the PCE web console as a user with Global Organization Owner privileges.
2. From the PCE web console menu, navigate to **Infrastructure > Container Clusters**.
3. Click **Add**.
 - a. Add a Name.
 - b. **Save** the Container Cluster.
4. You will see a summary page of the new Container Cluster. From the Cluster Pairing Token section, copy the values of the Cluster ID and Cluster Token.
5. After copying and saving the values (in a text editor or similar tool), open the Container Workload Profiles page.

Container Clusters - MyK8sCluster

Summary Container Workload Profiles Workloads Container Workloads Service Backends

Edit Remove

General

Name MyK8sCluster

Status ☒ Not yet connected

Heartbeat Last Received

Platform Version

Kubelink Version

Description

Please copy the following token and ID. This information will not be available after you leave the page. A new token will need to be generated.

Cluster Pairing Token

Cluster ID 42083a4d-dd90-43a6-b495-df8a2a40073c

Copy ID

Cluster Token 10_d1ea040af1fb0c-100a208070a093c-1009a445452a330a87c109a0e3e3bac1a95d2

Copy Token

Configure a Container Workload Profile Template

When configuring a new Container Cluster, it is recommended to set the default settings shared by all the Container Workload Profiles. Illumio provides a Container Workload Profile template that can be used for that purpose. By defining the default Policy State and minimum set of labels common to all namespaces in the cluster, you will save time later on when new namespaces are discovered by Kubelink. Each new profile created will inherit what was defined in the template.



IMPORTANT

Illumio does not provide a method to redefine at once all the labels associated with each profile. Hence, it is **strongly recommended** to use the provided template to define the default values for all profiles that are part of the same cluster.

To define the default parameters for all profiles using a template, under *Container Workload Profiles*, click **Edit default settings** and select values for all the fields.

After you click OK, the following information is displayed:

Create a Pairing Profile for Your Cluster Nodes



IMPORTANT

Before deploying the C-VEN, ensure that either of the following two requirements has been met:

- Kubelink is deployed on the Kubernetes cluster and is in sync with the PCE, or
- Firewall coexistence is enabled.

Before deploying, you should create a pairing profile to pair the cluster nodes with the PCE. You only need to create one pairing profile for all your nodes.



NOTE

You only need to create pairing profiles for Kubernetes or OpenShift nodes and not for container workloads.

For ease of configuration and management, consider applying the same Application, Environment, and Location labels across all nodes of the same Kubernetes or OpenShift cluster. The screenshot below shows an example of a pairing profile for a Kubernetes cluster.

Pairing Status	Name	Policy State	Role	Application	Environment	Location	Last Modified On	Last Modified By	Last Used On	Description
Running	Kubernetes Nodes	Build	Node	Kubernetes Cluster	Development	Cloud	03/22/2020, 22:39:13		02/04/2020, 03:42:52	

**TIP**

Illumio recommends all pairing profiles for Kubernetes nodes to *not* use Full Enforcement policy state. Use Idle or Visibility Only mode for initial configuration.

You should only move them into Full enforcement state after you have completed all other configuration steps in this guide.

Map Kubernetes Node or Workload Labels to Illumio Labels

Label mapping is a method of mapping some or all existing Kubernetes labels to Illumio labels. Label maps are an additional way to assign Illumio labels to container hosts or Kubernetes workloads in addition to existing methods, such as with container workload profiles and pairing profiles. Labels assigned through label maps take precedence; that is, they overwrite any labels assigned using other methods.

A label map is defined by a Kubernetes Custom Resource Definition (CRD) within a YAML file, which is typically installed via a Helm Chart. Installing the Helm Chart then applies the defined labels.

The label type must be created and exist in PCE first before new labels can be created through label mapping.

Kubernetes Node Labels or Kubernetes Workload Labels

You can map labels on Kubernetes nodes (also called host workloads) to Illumio labels, or map labels on Kubernetes Workloads to Illumio labels. Define labels for Kubernetes nodes in a **nodeLabelMap** section of your CRD, and labels for Kubernetes Workloads in a **workloadLabelMap** section.

**NOTE**

Note that Kubernetes Workloads is a term used only in CLAS-enabled deployments of Illumio Core for Kubernetes (contrasted to Container Workloads). Currently, there is no support for labeling non-CLAS Container Workloads.

**IMPORTANT**

You can map Kubernetes Workload labels only to deployments running PCE version 24.5.0 or later.

Label Mapping CRD

The CRD is defined in the YAML file with a **kind: LabelMap** declaration, which in turn contains a **nodeLabelMap** section that applies to nodes (host workloads) or a **workloadLabelMap** section that applies to Kubernetes Workloads. The declaration can contain both sections.

Within the **nodeLabelMap** or **workloadLabelMap** section, Illumio label types are mapped with **fromKey** and **toKey** key-value pairs, where the **fromKey** value specifies a source Kubernetes label, and the **toKey** value paired with it defines the destination Illumio label type.

If an optional **allowCreate: true** is within a **fromKey** and **toKey** pair, the Illumio label value defined in that mapping is created if it does not already exist on the PCE.

An optional **valuesMap:** within a **fromKey** and **toKey** pair specifies one or more label value mappings for that label type, with **from:** value identifying the source Kubernetes label and the **to:** value following it specifying the destination Illumio label value. If no **valuesMap:** is specified, then label values for the mapped label type are not changed. Only the label type is changed in the PCE.

Example Label Maps

The following example label map for Kubernetes node labels performs these mapping functions:

- The **nodeLabelMap** item creates a new Illumio **loc** label of **Amazon** (if it does not exist, per the **allowCreate: true** declaration). It maps this label to all nodes with the Kubernetes label **topology.kubernetes.io/region** with either value of **eu-west-1** or **eu-west-2**.
- With the second item under **nodeLabelMap**, for every **node-type** Kubernetes label, the map creates Illumio **k8s-node** labels with values based on the existing Kubernetes label values (because there is no associated **valuesMap** mapping definition).

```

kind: LabelMap
apiVersion: ic4k.illumio.com/v1alpha1
metadata:
  name: default
nodeLabelMap:
  - allowCreate: true
    fromKey: topology.kubernetes.io/region
    toKey: loc
    valuesMap:
      - from: eu-west-1
        to: Amazon
      - from: eu-west-2
        to: Amazon
  - allowCreate: true
    fromKey: node-type
    toKey: k8s-node

```

The following is a similar YAML file code excerpt that defines a label map for a Kubernetes Workload.

- In the first declaration under **workloadLabelMap**, for every **environ** Kubernetes label, the map creates an Illumio **env** label type, and maps **EKS** values to **AmazonK8SService** label values for this type.
- The next **fromKey** section maps workloads with the Kubernetes label **stage** to the Illumio label type **role**.

```

kind: LabelMap
apiVersion: ic4k.illumio.com/v1alpha1
metadata:
  name: default

workloadLabelMap:
  - fromKey: environ
    toKey: env
    allowCreate: false
    valuesMap:
      - from: EKS
        to: AmazonK8SService

  - fromKey: stage
    toKey: role
    allowCreate: true

```

Show the Source of a PCE Label

Because a Kubernetes Workload can have its label assigned to it in any of three different ways (Container Workload Profile, Kubernetes annotations,

or the label map CRD), the PCE now shows how a Kubernetes workload was labeled, that is, what is the source of the PCE label. The label source is indicated by an annotation that begins with the string `com.ilo-result.<label_type>` which is paired with a label source indicator.

The label source indicator can be one of these values:

- **container-workload-profile** - Container Workload Profile
- **annotations** - Kubernetes workload template annotation
- **label-map** - LabelMap CRD

This is shown in both the PCE web UI (in the workload details page in the Kubernetes Attributes section) and also in the command line output produced by the `kubectl get deploy` command.

Deploy with Helm Chart

To deploy via Helm Chart:

1. Install Helm. See <https://helm.sh/docs/> for a quick start guide and other relevant information.
According to official Helm documentation, if your version of Helm is lower than 3.8.0, the following command must be executed in the installation environment:

```
$ export HELM_EXPERIMENTAL_OCI=1
```

2. Prepare an `illumio-values.yaml` file with the following mandatory parameters set with values that describe this deployment:

```
pce_url: URL_PORT # PCE URL with port, e.g.
mypce.example.com:8443
cluster_id: ILO_CLUSTER_UUID # Cluster ID from PCE, e.g.
cc4997c1-40...
cluster_token: ILO_CLUSTER_TOKEN # Cluster Token from PCE,
e.g. 1_170b...
cluster_code: ILO_CODE # Pairing Profile key from PCE, e.g.
1391c...
containerRuntime: containerd # Container runtime engine
used in cluster, allowed values are [containerd, cri-o,
k3s-containerd]
containerManager: kubernetes # Container manager used in
cluster, allowed values are [kubernetes, openshift]
```

where `URL_PORT`, `ILO_CLUSTER_UUID`, `ILO_CLUSTER_TOKEN`, and `ILO_CODE` are placeholders for customer provided variables.

If you are using a private PKI, you need to add these additional lines to your `illumio_values.yaml`:

```
extraVolumeMounts:
  - name: root-ca
    mountPath: /etc/pki/tls/ilo_certs/
    readOnly: false
extraVolumes:
  - name: root-ca
    configMap:
      name: root-ca-config
```

You may also want to include selected optional parameters when installing, for example, with `clusterMode: clas` to deploy with a CLAS-enhanced Kubelink component. For more information, see Important Optional Parameters.



NOTE

If you want to deploy with CLAS enabled, you must explicitly set the `clusterMode` Helm Chart parameter. The default is to deploy in legacy (non-CLAS) mode.

3. Install the Helm Chart:

```
helm install illumio -f illumio-values.yaml oci://quay.io/
illumio/illumio --version <ver#> --namespace illumio-system
--create-namespace
```



IMPORTANT

Be sure to specify the version to install with the `--version <ver#>` option (for example, `--version 5.1.0`), after confirming that the Illumio Kubernetes Operator version you want to install is supported with your PCE version.

Verify which PCE versions support the Illumio Core for the Kubernetes version you want to deploy on the [Kubernetes Operator OS Support and Dependencies](#) page on the Illumio Support Portal.

Optionally, you can deploy into a custom namespace of your choice instead of the default namespace of `illumio-system`. The default name-

space is overridden for backward compatibility by using the variable `namespaceOverride: illumio-system`.

For example, to install into the `ilo` namespace, specify the namespace with the `--namespace` option and the `--set` option specifying `namespaceOverride` to `null`:

```
helm install illumio -f illumio-values.yaml oci://quay.io/illumio/illumio --version 5.3.0 --namespace ilo --create-namespace --set namespaceOverride=null
```

Alternatively, specify the namespace with the `--namespace` option but also use `--set` to explicitly set `namespaceOverride` to `ilo`:

```
helm install illumio -f illumio-values.yaml oci://quay.io/illumio/illumio --version 5.3.0 --namespace ilo --create-namespace --set namespaceOverride=ilo
```



NOTE

Kubelink version labeling has changed. Prior to version 3.3.0, Kubelink used a 6-hexit suffix for its release version, like 3.2.1.445a83. In Kubelink 3.3.0 and later, the version suffix is now changed to a numeric build number, like 3.3.0-56.

Optional Parameters

Refer to the **README** file included with the Helm Chart for important deployment information, including parameters you can specify in the Helm Chart before installing it. To view the readme, execute:

```
helm show readme oci://quay.io/illumio/illumio
```

The following list describes a few important optional parameters to consider using in your `illumio-values.yaml` file.

CLAS Mode: `clusterMode`

Starting in Illumio Core for Kubernetes versions 5.0.0 and later, a Cluster Local Actor Store (CLAS) mode is introduced into the Kubelink architecture. Use the optional `clusterMode` parameter to configure Kubelink when first installing a new cluster, or when migrating an existing cluster.

When installing, set `clusterMode` to `clas` or `legacy` in your `illumio-values.yaml` file to turn on (or leave off) CLAS mode in the cluster, respectively.

The default setting for `clusterMode` is `legacy` (non-CLAS). To enable CLAS in a new cluster, you must explicitly include `clusterMode:clas` in the `illumio-values.yaml` file when installing.

When upgrading an existing non-CLAS cluster to CLAS, set `clusterMode` to `migrateLegacyToClas`. When reverting (or downgrading) CLAS to non-CLAS, set `clusterMode` to `migrateClasToLegacy`.



IMPORTANT

To upgrade to CLAS, follow the procedure described in Upgrade to CLAS Architecture.

Illumio recommends enabling (or migrating to) CLAS-enabled clusters to take advantage of this architecture's benefits. For more information about CLAS, see the Cluster Local Actor Store (CLAS) section.

Cluster mode supports these values:

- `legacy` - the default, PCE manages individual Pods as Container Workloads
- `clas` - Cluster Local Actor Store, PCE manages cluster workloads as Kubernetes Workloads
- `migrateLegacyToClas` - use for upgrade to CLAS mode, see Product Guide
- `migrateClasToLegacy` - use for downgrade from CLAS mode, see Product Guide
- `unpair` - unpair C-VEs from PCE before deletion

CLAS Degraded Mode: `disableDegradedMode` and `degradedMode-PolicyFail`

If the connection between Kubelink and the PCE becomes unavailable, a CLAS-enabled Kubelink can still serve policies to C-VEs (and therefore to its Kubernetes Workloads and pods). When a PCE interruption is detected, a CLAS-enabled Kubelink enters a **degraded mode**.

By default, degraded mode is enabled in CLAS clusters. You can disable degraded mode by explicitly setting the parameter/value pair `disableDegradedMode: true` in `illumio-values.yaml`, and performing a `helm upgrade`.

In degraded mode, new Pods of existing Kubernetes Workloads get the latest policy version cached in CLAS storage. When Kubelink detects a new Kubernetes Workload labeled the same way and in the same namespace as an existing Kubernetes Workload, Kubelink delivers the existing, cached policy to Pods of this new Workload.

If Kubelink cannot find a cached policy (that is, when labels of a new Workload do not match the labels of any existing Workload in the same namespace), Kubelink delivers a "fail open" or "fail closed" policy based on the Helm Chart parameter **degradedModePolicyFail** setting, as specified in the **illumio-values.yaml** file when installing (or upgrading).

The default parameter value of **degradedModePolicyFail** is **open**, which opens the firewall of new Pods. The **closed** value means the firewall of new Pods is programmed to block all network connectivity.

The precise behavior of **closed** depends on the Cluster Workload Profile's Enforcement setting: all connectivity is blocked only if the Enforcement of the namespace is set to Full.

By default, degraded mode is enabled in CLAS clusters. You can disable degraded mode by explicitly setting the parameter/value pair **disableDegradedMode: true** in **illumio-values.yaml**, and performing a **helm upgrade**.

When degraded mode is disabled, Kubelink/CLAS does not deliver policy based on matching labels. Kubelink continues to run, and delivers the cached policy to existing Kubernetes Workloads, but does not deliver policy to new Workloads. Kubelink continues to attempt re-establishing communication with the PCE.

After the PCE becomes available again, it restarts, synchronizes policy and labels, and then continues normal operation.



NOTE

If the PCE becomes inaccessible due to database restoration or maintenance, and Kubelink has disabled degraded mode, you are advised to restart Kubelink by deleting its Pod to synchronize the current state.

Flat Networks: `networkType`

To add support for flat network CNIs in addition to the default (where pods run on an overlay network), an optional **`networkType`** parameter is now available in the Helm Chart where you can specify **`flat`** or **`overlay`** type. The default value is **`overlay`**.

Starting in version 5.2.0, the Kubelink Operator supports flat network CNIs in CLAS mode, a feature that was previously only available in non-CLAS mode. This update includes compatibility with flat network types such as Azure CNI Pod Subnet and Amazon VPC CNI. To enable a flat network CNI, set the **`networkType`** parameter to **`flat`** in the Helm Chart's **`illumio-values.yaml`** file during installation.

Also note that in CLAS-enabled flat networks, if a pod communicates with a virtual machine outside the cluster using private IP addresses, you must enable the annotation **`meta.illumio.podIPObservability`**. This is a scenario in which the virtual machine is in a private network and has an IP address from the same range as cluster nodes and pods. In this case, the PCE needs to know the private IP address of the pod to be able to open a connection on the virtual machine. The main benefit of CLAS is that the PCE no longer directly manages individual pods, so the implementation expects a specific annotation on such pods. Traffic between such private IPs will be blocked without this annotation, and will appear in the UI as blocked.

In this case, when the application communicates through private IPs, add the following annotation so that Kubelink can then report the private IPs of Kubernetes Workloads to the PCE:

```
metadata:
  annotations:
    meta.illumio.podIPObservability: "true"
```

CLAS etcd Internal Storage Size: `sizeGi`

Kubelink in CLAS mode uses etcd as a local cache for policy and runtime data. The Helm Chart parameter **`storage.sizeGi`** sets the size in GB of this ephemeral storage. Set the parameter under **`storage`** in the **`illumio-values.yaml`** for a cluster, as shown in the following example:

```
storage:
  registry: "docker.io/bitnami"
  repo: "etcd"
  imageTag: "3.5.7"
  imagePullPolicy: "IfNotPresent"
  sizeGi: 1
```

The default value is 1, for 1 GB, which should be enough for a cluster with under 1000 Kubernetes workloads. If a cluster is bigger and you increase memory limits for C-VEN and Kubelink, then increase the etcd internal storage size with this parameter.

Label Mapping

You can map existing Kubernetes labels to desired Illumio labels by adding a Kubernetes *Custom Resource Definition* (CRD) label map to your **illumio-values.yaml** file.

Pod priority

Kubelink Pod and CVEN Pods have **priorityClassName** specified by default. This can be disabled by setting **priorityClasses.enabled** to **false**. Default values of **priorityClassName** can be changed by setting **kubelink** and **cven** values.

NodePort traffic enforcement disabling

Starting from C-VEN 22.5, NodePort ingress traffic is enforced by default. To allow NodePort traffic, additional policy rules are needed. If this behavior is undesired, it can be controlled by setting **enforceNodePortTraffic: never**.

Default values:

```
enforceNodePortTraffic: always
```

Common Labels and Common Annotations

There is an option to specify Common Labels and Common Annotations in **illumio-values.yaml**:

```
commonLabels: { } # Labels applied to all deployed objects
commonAnnotations: { } # Annotations applied to all
deployed objects
```

Root CA Certificate

If you are using a private PKI to sign the PCE certificate, make sure you add to **illumio-values.yaml** the references to the root CA certificate that signed the PCE certificate, as shown below:

```
extraVolumeMounts:
  - name: root-ca
    mountPath: /etc/pki/tls/ilo_certs/
    readOnly: false
extraVolumes:
  - name: root-ca
    configMap:
      name: root-ca-config
```

Kubelink nodeSelector

If the infrastructure requires Kubelink to run on the control-plane node, add the below configuration to `illumio-values.yaml` (the selector label or name may differ). Keep in mind that this only applies to Kubelink deployment.

```
nodeSelector:
  node-role.kubernetes.io/control-plane: ""
```

Kubelink deployment also supports two more settings:

1. Option to ignore certificate verification of PCE
2. The verbosity level of logging

```
ignore_cert: true
verbosity : 0
```

Image Pull Policy

Illumio Core for Kubernetes and Openshift provides an option to specify default image pull policy for each of its components.

```
kubelink:
  imagePullPolicy: "Always"
cven:
  imagePullPolicy: "Always"
```

Resources

Resources limits and requests are defined as chart values for Kubelink Deployment and CVEN DaemonSet. Values for both components can be changed in `illumio-values.yaml`, and each **resources** section is used as a whole for the corresponding container, effectively replacing the default **resources** definition.

See Resource Allocation Guidelines for deployment sizes.

```
kubelink:
  resources:
    limits:
      memory: 500Mi
    requests:
      memory: 200Mi
      cpu: 200m

cven:
  resources:
    limits:
      memory: 300Mi
    requests:
      memory: 100Mi
      cpu: 250m

storage:
  resources:
    limits:
      memory: 500Mi
    requests:
      memory: 200Mi
      cpu: 100m
```

Metrics

Illumio Core for Kubernetes & OpenShift provides tools for collecting and visualization of runtime metrics since IC4K chart version 4.4. The chart contains InfluxDB as a dependency. To enable metrics use:

```
metrics:
  enabled: true
```

Please refer to <https://github.com/bitnami/charts/tree/main/bitnami/influxdb/> for a complete list of available parameters. To set a value in the `illumio-values.yaml` for InfluxDB subchart the `influxdb` section must be used.



WARNING

Secrets for the InfluxDB must be specified during the upgrade.

First installation with metrics

Use the standard command, secrets will be generated.

```
$ helm install illumio -f illumio-values.yaml oci://quay.io/
illumio/illumio --version 5.3.0 --namespace illumio-system --
create -namespace
```

Upgrade and enable metrics

To upgrade Illumio Core for Kubernetes & OpenShift deployed with the previous chart version or without metrics enabled, the user has to generate secrets for the version with enabled metrics.

```
export ADMIN_USER_PASSWORD = `LC_ALL=LC tr -dc A-Za-z0-9
</dev/urandom | head -c 20 ; echo`

export ADMIN_USER_TOKEN = `LC_ALL=LC tr -dc A-Za-z0-9 </dev/
urandom | head -c 20 ; echo`

helm upgrade illumio -f illumio-values.yaml oci://quay.io/
illumio/illumio --version 5.3.0 --namespace illumio-system --
set influxdb.auth.admin.password = $ADMIN_USER_PASSWORD --
set influxdb.auth.admin.token = $ADMIN_USER_TOKEN
```

Upgrade with enabled metrics

To upgrade Illumio Core for Kubernetes & OpenShift deployed with enabled metrics, the user has to pass existing secrets.

```
export ADMIN_USER_PASSWORD = $(kubectl get secret --
namespace illumio-system illumio-influxdb -o jsonpath=
"{.data.admin-user-password}" | base64 -d)

export ADMIN_USER_TOKEN = $(kubectl get secret --namespace
illumio-system illumio-influxdb -o jsonpath= "{.data.admin-
user-token}" | base64 -d)

helm upgrade illumio -f illumio-values.yaml oci://quay.io/
illumio/illumio --version 5.3.0 --namespace illumio-system --
set influxdb.auth.admin.password = $ADMIN_USER_PASSWORD --
set influxdb.auth.admin.token = $ADMIN_USER_TOKEN
```

InfluxDB parameters

The InfluxDB deployed with default options will create a persistent volume with a capacity 8Gi and use the default storage class from the cluster. The current Illumio metrics data size for 1 day is approximately 55Mi, and the

retention policy for the Illumio bucket is set to 30 days. 4Gi capacity should be enough room for Illumio metrics. Example setting:

```
influxdb:
  persistence:
    size: 4Gi
    storageClass: kops-csi-1-21
```

Setting for disabling the metrics persistence:

```
influxdb:
  persistence:
    enabled: false
```

Wait for policy

With a new Wait For Policy feature, CLAS-enabled Kubelink can be configured to automatically and transparently delay the start of an application container in a pod until a policy is properly applied to that container. This synchronizes the benefit of automatic container creation with the protection of proper policy convergence into the new container.

This Wait For Policy feature replaces the existing local policy convergence controller, also known as a readiness gate. A readiness gate required manually adding the readinessGate condition into spec of Kubernetes workload. Wait For Policy uses an automatically injected init container, which requires no change to the user application.

The Wait For Policy feature is disabled by default. To enable it, change the **waitForPolicy: enabled** value to true in the Helm Chart `illumio-values.yaml` file.

Support bundle

To assist the Illumio Support team with more details for troubleshooting, Kubelink 5.2.0 and higher provide a support bundle that collects up to 2 GB of logs, metrics, and other data inside its pod. Kubelink version 5.7.0 and later offer the option to upload these support bundles to the PCE. For Kubelink 5.2.0 through 5.6.1, you must copy this support bundle by running the script `/support_bundle.sh` inside the Kubelink pod. The script generates debug data, creates a gzipped tar archive using **stdout** as output, and encodes this data using Base64.

Use the following command to generate and transfer the Kubelink support bundle from its pod:

```
kubectl -n illumio-system exec deploy/illumio-kubelink -- /
support_bundle.sh | base64 -d > kubelink_support.tgz
```

Kubelink Report Directory Size

For kubelink 5.7.0 and higher, the support report directory size is configurable from Helm Chart using the `kubelink.supportReportVolumeSize` value. The default size is 2000 MB. When the data size reaches 75% of the configured size (1500 MB for the default), Kubelink removes older files from the support report directory to prevent filling the storage volume.

The default PCE upload limit is 100MB. Smaller support reports can be uploaded to PCE and then downloaded from the PCE UI in the **Summary** tab for the container cluster in the **Support Reports** section.

Support reports larger than the upload limit can be downloaded using `kubectl` command:

```
kubectl -n illumio-system exec deploy/illumio-kubelink -- /
support_bundle.sh | base64 -d > kubelink_support.tgz
```

Kubernetes Workload metadata

Kubelink reports Kubernetes workloads with its labels and annotations. Starting from Kubernetes Operator 5.4.0, only Illumio annotations are reported. To include other annotation keys, use `userAllowedAnnotations` list:

```
userAllowedAnnotations:
- "deployment.kubernetes.io/revision"
```

Workloads in host network

Starting from Kubernetes Operator 5.4.0, Kubelink does not create a Kubernetes Workload on PCE if the workload is in the host network. Pods of these workloads don't have separate network namespaces. Policy for these workloads must be part of the policy for Nodes. This was always the case, legacy non-CLAS mode works the same way.

Label-set optimized policy

This optimization can reduce the size of the policy sent from PCE. Instead of sending policy for each Kubernetes workload, PCE groups policies using Kubernetes Workload PCE labels. Kubernetes Workloads with a unique set of PCE labels will not benefit from this optimization.

```
policyLabelSetEnable: true
```


C-VEN DaemonSet Tolerations

Starting from Kubernetes Operator 5.7.0, configure tolerations for the C-VEN DaemonSet via Helm chart values to deploy to nodes with custom taints without needing to patch and republish the chart. This enables the ability to add, update, or override tolerations for the C-VEN DaemonSet.

Taints are applied to nodes to prevent pods from being scheduled on them unless the pods explicitly tolerate the taint.

This helps isolate workloads or reserve nodes for specific purposes.

Tolerations are added to pod specifications to let pods bypass taints. Tolerations don't force scheduling onto tainted nodes. They only allow scheduling if other conditions are met.

Together, taints and tolerations control pod placement.

Within the Helm Chart `values.yaml` file, the `cven` field supports the `tolerations` field with the following syntax:

```
tolerations:
  - key: <keyname>
    value: <value>
    effect: <effect>
```

The tolerations field requires these keys to ensure OpenShift support and backwards compatibility.

```
cven:
  ...
  tolerations:
    - key: node-role.kubernetes.io/master
      effect: NoSchedule
    - key: node-role.kubernetes.io/control-plane
      effect: NoSchedule
```

To append to existing tolerations, copy all of the existing tolerations, then provide an additional key, value, and effect.

For example, to allow C-VEN pods to run on nodes tainted with `dedicated=monitoring`, the tolerations field includes the key `dedicated`, the value `monitoring`, and the effect `NoSchedule`.

```
cven:
...
  tolerations:
    - key: node-role.kubernetes.io/master
      effect: NoSchedule
    - key: node-role.kubernetes.io/control-plane
      effect: NoSchedule
    - key: dedicated
      value: monitoring
      effect: NoSchedule
```

After updating the Helm chart yaml file to include the tolerations, run `helm install`.

To view the tolerations for a node, run the `kubectl describe node` command for a node.

```
kubectl describe node <node-name>
...
Taints:                dedicated=monitoring:NoSchedule
```

For Kubernetes documentation on Taints and Tolerations, see <https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/>

Helm Chart Parameters

Name	Description	Default Value	Accepted Values
<code>pce_url</code>	URL connection to PCE with port	<code>""</code>	
<code>cluster_id</code>	Cluster ID for created cluster in PCE	<code>""</code>	
<code>cluster_token</code>	Cluster Token for created cluster in PCE	<code>""</code>	
<code>cluster_code</code>	Cluster Code from Pairing Profile	<code>""</code>	
<code>namespaceOverride</code>	Override the namespace for Illumio resources	<code>"illumio-system"</code>	
<code>containerRuntime</code>	Container Runtime used in cluster, used for unix sockets directory path	<code>"containerd"</code>	<code>[containerd, crio, k3s_containerd]</code>
<code>containerManager</code>	Container Manager used in cluster, used for specific deployments	<code>"kubernetes"</code>	<code>[kubernetes, openshift]</code>
<code>clusterMode</code>	See "Cluster Mode" section	<code>"legacy"</code>	<code>[legacy, migrateClasToLegacy, migrateLegacyToClas, clas]</code>
<code>enforceNodePortTraffic</code>	See "Cluster Mode" section	<code>"always"</code>	<code>[always, never]</code>
<code>priorityClasses.enabled</code>	Enables priorityClassName	<code>true</code>	<code>[true, false]</code>
<code>priorityClasses.kubelink</code>	Kubelink Deployment priorityClassName	<code>system-cluster-critical</code>	
<code>priorityClasses.cven</code>	CVEN DaemonSet priorityClassName	<code>system-node-critical</code>	
<code>degradedModePolicyFail</code>	See "CLAS Degraded Mode" section	<code>"open"</code>	<code>[open, closed]</code>
<code>disableDegradedMode</code>	Disable CLAS Degraded Mode	<code>false</code>	<code>[true, false]</code>

Name	Description	Default Value	Accepted Values
<code>reportHostNetwork-KubernetesWorkloads</code>	See "Workloads in host network" section	false	[true, false]
<code>imagePullSecret.username</code>	Username for connecting to quay.io registry	""	
<code>imagePullSecret.password</code>	Password for connecting to quay.io registry	""	
<code>commonLabels</code>	Add common labels to all deployed objects	{}	
<code>commonAnnotations</code>	Add common annotations to all deployed objects	{}	
<code>extraVolumeMounts</code>	Used to mount directories to Kubelink and C-VEN containing certificates	{}	
<code>extraVolumes</code>	Used to mount directories to Kubelink and C-VEN containing certificates	{}	
<code>ignore_cert</code>	Skip certificate verification of <code>pce_url</code> in Kubelink	false	[true, false]
<code>networkType</code>	Cluster network type	"overlay"	[overlay, flat]
<code>verbosity</code>	Verbosity level for logs	0	[0, 1, 2]
<code>kubelink.imagePullPolicy</code>	Kubelink imagePullPolicy level	IfNotPresent	[IfNotPresent, Always, Never]
<code>cven.imagePullPolicy</code>	C-VEN imagePullPolicy level	IfNotPresent	[IfNotPresent, Always, Never]
<code>cven.hostBasePath</code>	Path for C-VEN data directory on Nodes (/opt/illumio_ven_data)	/opt	
<code>cven.tolerations</code>	C-VEN Daemonset tolerations	array	array
<code>cven.firewallBackend</code>	C-VEN firewall backend	"auto"	["auto","iptables","nftables"]
<code>kubelink.pceApiTimeout</code>	HTTP request timeout, in seconds	1200	60 - 7200

Name	Description	Default Value	Accepted Values
<code>kubelink.pceApi-MinBackoff</code>	backoff delay before request retry, in seconds	60	5 - 600
<code>kubelink.pceApi-MaxBackoff</code>	max backoff delay, in seconds	240	60 - 1200
<code>kubelink.pceApiPolicyInterval</code>	how often should Kubelink ask if there is new policy, in seconds	30	10 - 3600
<code>kubelink.lastAppliedPolicyUpdateInterval</code>	how often should Kubelink update last applied policy timestamp, in seconds	120	5 - inf
<code>kubelink.resources</code>	See "Resources" section	object	
<code>cven.resources</code>	See "Resources" section	object	
<code>storage.resources</code>	See "Resources" section	object	
<code>metrics.enabled</code>	Enable metrics	false	[true, false]
<code>policyLabelSetEnable</code>	Enable label-set policy optimization	true	[true, false]
<code>argoCD</code>	Chart is deployed with ArgoCD	false	[true, false]
<code>httpProxy</code>	HTTP proxy URL to be used for Kubelink and C-VEN requests	""	

Re-Label Your Cluster Nodes



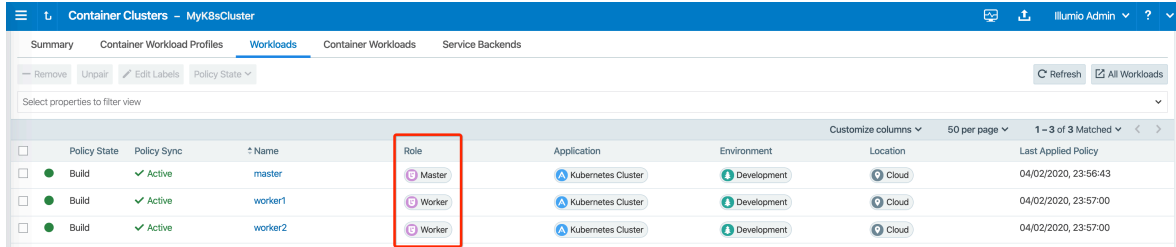
NOTE

Re-labeling the cluster nodes is optional.

In the case of self-managed deployments in which both Master and Worker nodes are managed, you may want to re-label your nodes to differentiate Master nodes from Worker nodes. Doing this helps when you are writing different policies for the Worker and Master nodes, or if you want to segment these nodes differently.

To re-label your cluster nodes:

1. In the PCE UI, go to **Infrastructure > Container Clusters > YourCluster-Name > Workloads**.
2. Select the workloads you want to re-label.
3. Click **Edit Labels** to assign the new labels (for example, Master and Worker).



Policy State	Policy Sync	Name	Role	Application	Environment	Location	Last Applied Policy
Build	Active	master	Master	Kubernetes Cluster	Development	Cloud	04/02/2020, 23:56:43
Build	Active	worker1	Worker	Kubernetes Cluster	Development	Cloud	04/02/2020, 23:57:00
Build	Active	worker2	Worker	Kubernetes Cluster	Development	Cloud	04/02/2020, 23:57:00

4. After re-labeling your cluster nodes, the nodes part of the cluster reflect the updated label(s).

Generating YAML Manifests for Manual Deployment

In addition to the typical deployment with a Helm Chart, alternatively you can manually deploy Illumio Core for Kubernetes and OpenShift using customized YAML manifests that you have changed to suit your specific needs.

The procedure consists of the following steps, which are described in the following sections:

1. Install Helm tool.
2. Generate files.
3. Remove unpair DaemonSet and Job commands.

Install Helm Tool

There are several options for installing the Helm tool, depending on the operating system you are running. For complete details on all options, see <https://helm.sh/docs/intro/install/>. A few common installation commands are shown below:

```
brew install helm
```

```
sudo snap install helm --classic
```

```
export HELM_LATEST=$(curl -s https://api.github.com/repos/helm/helm/releases/latest | grep tag_name | cut -d '"' -f 4)
curl -LJO https://get.helm.sh/helm-$HELM_LATEST-linux-amd64.tar.gz
tar -zxvf helm-$HELM_LATEST-linux-amd64.tar.gz
mv linux-amd64/helm /usr/local/bin/helm
```

Generate Files

Prepare `values.yaml` in advance. The file must set at least the following minimally required parameters:

```
pce_url: URL_PORT
cluster_id: ILO_CLUSTER_UUID
cluster_token: ILO_CLUSTER_TOKEN
cluster_code: ILO_CODE
containerRuntime: RUNTIME # supported values: [containerd
(default), docker, cri-o, k3s_containerd]
containerManager: MANAGER # supported values: [kubernetes,
openshift]
networkType: flat # CNI type, allowed values are [overlay,
flat]
clusterMode: clas #
```

Generate templates and redirect output into a file, for example, into `illumio.yaml`:

```
helm template oci://quay.io/illumio/illumio -f values.yaml --
version <ver#> > illumio.yaml
```



IMPORTANT

Be sure to explicitly specify the version you want to install with the `--version <ver#>` option (for example, `--version 5.1.0`), after confirming that the product version you want to install is supported with your PCE version. Verify which PCE versions support the Illumio Core for Kubernetes version you want to deploy at the [Kubernetes Operator OS Support and Dependencies](#) page on the Illumio Support Portal.

Remove Unpair DaemonSet and Job Objects

In the generated YAML file `illumio.yaml`, search for and remove the DaemonSet and Job objects. Remove only these two objects; they are only used for the removal of Illumio product:

```

...
kind: Job
metadata:
name: illumio-ven-unpair-job
...
kind: DaemonSet
metadata:
name: illumio-ven
...

```

Note that the DaemonSet name changed in Illumio Core for Kubernetes version 5.2.0 from `illumio-ven-unpair` to `illumio-ven`.

Deployment for C-VEN Versions 21.5.15 or Earlier

After you set up your clusters, perform these steps.



NOTE

Follow these instructions if you are deploying Kubernetes (C-VEN) versions 21.5.15 or earlier.

If you are deploying Kubernetes 3.0.0 release (or later), do not follow these instructions, but instead refer to [Deployment with Helm Chart \(Kubernetes 3.0.0 and Later\) \[187\]](#), which describes how to use a Helm Chart to deploy all necessary product components.

The installation process is mostly the same for Kubernetes and OpenShift, with a few exceptions. A dedicated section is created for Kubernetes or OpenShift wherever required.

Host and Cluster Requirements

To deploy Illumio containers into your environment, you must meet the following requirements.

Supported Configurations for On-premises and IaaS

For full details on all supported configurations for Containerized VEN release 21.5.15 and earlier, see the [C-VEN/Kubelink OS Support and Dependencies page](#) on the Illumio Support Portal (under Software > OS Support).

Privileges

The privileges listed below should be provided on host-level and cluster-level for the respective components.

Host-Level

C-VEN

C-VEN requires the following privileges on the host:

- C-VEN is a privileged container and requires access to the following system calls:
 - `NET_ADMIN`
 - `SYS_MODULE`
 - `SYS_ADMIN`
- C-VEN requires persistent storage on the host to write iptables rules and logs.
- C-VEN mounts volumes on the local host to be able to operate (mount points may differ depending on the orchestration platform).

Optionally, you can set the Priority Class to **system-node-critical**. This option is only supported in Kubernetes 1.17 and later, in a namespace other than **kube-system**. For more details, see the Kubernetes documentation.

Kubelink

Kubelink does not require specific privileges on the host because Kubelink:

- Is not a privileged container.
- Is a stateless container.
- Does not require persistent storage.

Cluster-Level

Namespace

C-VEs and Kubelink are deployed in the **illumio-system** namespace. You can modify this namespace name according to your deployment (manifest file modification).

C-VEN

C-VEN requires the following privileges on the cluster:

- C-VEN uses the `illumio-ven` ServiceAccount.

Kubelink

Kubelink requires the following privileges on the cluster:

- Kubelink creates a new Cluster Role to list and watch events occurring on the Kubernetes API server for the following elements:
 - `nodes`
 - `hostsubnets`
 - `replicationcontrollers`
 - `services`
 - `replicasets`
 - `daemonsets`
 - `namespaces`
 - `statefulsets`
- Kubelink uses the `illumio-kubelink` ServiceAccount.

Optionally, you can set the Priority Class to `system-cluster-critical`. This option is only supported in Kubernetes 1.17 and later, in a namespace other than `kube-system`. For more details, see the Kubernetes documentation.

Prepare Your Environment



IMPORTANT

The following steps for preparing your environment are no longer needed when deploying Illumio Core for Kubernetes version 3.0.0 and beyond, which now uses Helm Chart for deploying C-VEN and Kubelink. This section is included here for backwards compatibility and historical purposes. If you are deploying using Helm Chart, skip this section and now follow the instructions in [Create a Container Cluster in the PCE \[240\]](#).

You need to do these steps before C-VEN installation and pairing.

**CAUTION**

If the prerequisite steps are not done before C-VEN and Kubelink installation, then containerized environments and Kubelink can get disrupted.

Unique Machine ID

Some of the functionality and services provided by the Illumio C-VEN and Kubelink depend on the Linux machine-id of each Kubernetes cluster node. Each machine-id must be unique in order to take advantage of the functionality. By default, the Linux operating system generates a random machine-id to give each Linux host uniqueness. However, there are cases when machine-id's can be duplicated across machines. This is common across deployments that clone machines from a golden image, for example, spinning up virtual machines from VMware templates, creating compute instances from a reference image, or from a template from a Public Cloud source.

**IMPORTANT**

Illumio Core requires a unique machine-id on all nodes. This issue is more likely to occur with on-premises or IaaS deployments, rather than with Managed Kubernetes Services (from Cloud Service Providers). For more information on how to create a new unique machine-id, see [Troubleshooting \[295\]](#).

Create Labels

For details on creating labels, see "Labels and Label Groups" in Security Policy Guide. The labels shown below are used in examples throughout this document. You are not required to use the same labels

Name	Label Type
Kubernetes Cluster	Application
OpenShift Cluster	Application
Production	Environment
Development	Environment
Data Center	Location
Cloud	Location
Kubelink	Role
Node	Role
Control Plane Node (formerly Master)	Role
Worker	Role

Push Kubelink and C-VEN Images to Your Container Registry

In order to install Illumio Core for containers, you first need to upload (or push) Kubelink and C-VEN container images to your container registry. The files in the C-VEN and Kubelink packages you've downloaded are as follows:

C-VEN `illumio-ven-xx.x.x-xxxx.k8s.x86_64.tgz` package includes:

- A Docker image
 - `illumio-ven-xx.x.x-xxxx.tgz`
- Configuration files:
 - `illumio-ven-secret.yml`
 - `illumio-ven-kubernetes.yml`
 - `illumio-ven-openshift.yml`

Kubelink `illumio-kubelink-x.x.x.tar.gz` package includes:

- A docker image
 - `kubelink-image.tar.gz`
- Configuration files in kube-yaml
 - `illumio-kubelink-secret.yml`
 - `illumio-kubelink-kubernetes.yml`
 - `illumio-kubelink-openshift.yml`

- `illumio-kubelink-namespace.yml`



CAUTION

These images are not publicly available and should **not** be posted on a publicly open container registry without Illumio's consent.

In a self-managed deployment, Kubelink and C-VEN images can be pushed to a private container registry. In OpenShift, a container registry is provided as part of the platform, and images can be pushed to this registry for simplicity and better authentication. In the case of Kubernetes, there is no container registry provided by default and must be provided as an external component.

In a cloud-managed deployment, Cloud Service Providers (CSPs) provide integration of private container registries such as, Amazon ECR, Microsoft ACR, and so on. These registries can securely be used to host Illumio's container images for Kubelink and C-VEN. Refer to the documentation provided by the respective CSPs to learn how to push images to those registries.

To push Kubelink and C-VEN container images to your private container registry, use the following commands (based on docker):

1. Log in to your private container registry.

```
docker login <docker-registry>
```

2. Load Kubelink and C-VEN container images on your local computer.

```
docker load -i kubelink-image.tar.gz
docker load -i illumio-ven-21.5.x-xxxx.tgz
```

Verify that docker images are loaded on your computer.

```
docker image ls
```

3. Tag the Kubelink and C-VEN container image IDs with the name of your container registry.

```
docker tag <illumio-kubelink-image-id> <docker-registry>/illumio-kubelink:2.1.x.xxxxxx
docker tag <illumio-ven-image-id> <docker-registry>/illumio-ven:21.5.x-xxxx
```

Verify that images are tagged on your computer and ready to be pushed to your private container registry.

```
docker image ls
```

4. Push Kubelink and C-VEN container images on your private container registry.

```
docker push <docker-registry>/illumio-kubelink:2.1.x.xxxxxx  
docker push <docker-registry>/illumio-ven:21.5.x-xxxx
```

After pushing images to your private container registry, proceed to the next section.

Create Illumio Namespace

Illumio Core for containers is deployed in a dedicated namespace `illumio-system`, by default. This namespace has the minimum privileges in the cluster required to run Illumio Segmentation for Data Centers and can tie into the Kubernetes and OpenShift RBAC models.

To create the `illumio-system` namespace for Kubernetes, use the following command:

```
kubectl create namespace illumio-system
```



NOTE

Illumio provides a yaml manifest file to create the namespace in the Kubelink tarball `illumio-kubelink-namespace.yml`. You can create this namespace by applying this manifest file to your Kubernetes cluster, using the following command:

```
kubectl apply -f illumio-kubelink-namespace.yml
```

To create the `illumio-system` project for OpenShift, use the following command:

```
oc new-project illumio-system
```

Authenticate Kubernetes Cluster with Container Registry



NOTE

Depending on your deployment, the steps in the [Authenticate Kubernetes Cluster with Container Registry \[233\]](#), [Create a ConfigMap to Store Your Root CA Certificate \[233\]](#), and [Configure Calico in Append Mode \[239\]](#) topics are optional.

When storing container images in a private container registry, it is often required and strongly recommended to authenticate against the registry to be able to pull an image from it. In order to do this, the Kubernetes or OpenShift cluster must have the credentials configured and stored in a secret file to be able to pull container images.

To configure a secret to store your container registry credentials, use the following command:

```
kubectl create secret docker-registry <container-registry-secret-name>
-n illumio-system --docker-server=<container-registry>
--docker-username=<username> --docker-password=<password>
```

To verify that the secret has been created, use the following command:

```
kubectl get secret -n illumio-system |
grep <container-registry-secret-name>
```



IMPORTANT

The above commands are valid for deployments with your own private container registry, but may not be valid for a cloud-managed private container registry. For more information, refer to your Cloud Service Provider documentation.

Create a ConfigMap to Store Your Root CA Certificate

This section describes how to implement Kubelink with a PCE using a certificate signed by a private Public Key Infrastructure (PKI). It describes how to

configure Kubelink and C-VEN to accept certificates from the PCE signed by a private root or intermediate Certificate Authority (CA), ensuring that Kubelink can communicate securely with the PCE.

Prerequisites

- Access to the root CA to download the root CA certificate.
- Access to your Kubernetes cluster and can run **kubect l** commands.
- Correct privileges in your Kubernetes cluster to create resources like ConfigMaps, secrets, and Pods.
- Access to the PCE web console as a Global Organization Owner.

Download the Root CA Certificate

Before you begin, ensure that you have access to the root CA certificate. The root CA certificate is a file that can be exported from the root CA without compromising the company's security. It is usually made available to external entities to ensure a proper SSL handshake between a server and its clients.

You can download the root CA cert in the CRT format on your local machine. Below is an example of a root CA certificate:

```
$ cat root.democa.illumio-demo.com.crt
-----BEGIN CERTIFICATE-----
MIIGSzCCBD0gAwIBAgIUAPw0NfPAivJW4YmKZ499eHZH3S8wDQYJKoZIhvcNAQEL
---output suppressed---
wPG0lug46K1EPQqMA7YshmrwOd6ESy6RGNFFZdhk9Q==
-----END CERTIFICATE-----
```

You can also get the content of your root CA certificate in a readable output format by using the following command:


```

$ openssl x509 -text -noout -in ./root.democa.illumio-
demo.com.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
fc:34:35:f3:c0:8a:f2:56:e1:89:8a:67:8f:7d:78:76:47:dd:2f
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=California, L=Sunnyvale, O=Illumio,
        OU=Technical Marketing, CN=Illumio Demo Root
        CA 1/emailAddress=tme-team@illumio.com
        Validity
            Not Before: Jan 20 00:05:36 2020 GMT
            Not After : Jan 17 00:05:36 2030 GMT
        Subject: C=US, ST=California, L=Sunnyvale, O=Illumio,
        OU=Technical Marketing, CN=Illumio Demo Root
        CA 1/emailAddress=tme-team@illumio.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
00:c0:e5:48:7d:97:f8:5b:8c:ef:ac:16:a8:8c:aa:
68:b8:48:af:28:cd:17:8f:02:c8:82:e9:69:62:e2:
89:2b:be:bd:34:fc:e3:4d:3f:86:5e:d7:e6:89:34:
71:60:e6:54:61:ac:0f:26:1c:99:6f:80:89:3f:36:
b3:ad:78:d1:6c:3f:d7:23:1e:ea:51:14:48:74:c3:
e8:6e:a2:79:b1:60:4c:65:14:2a:f1:a0:97:6c:97:
50:43:67:07:b7:51:5d:2c:12:49:81:dc:01:c9:d1:
57:48:32:2e:87:a8:d2:c0:b9:f8:43:b2:58:10:af:
54:59:09:05:cb:3e:f0:d7:ef:70:cc:fc:53:48:ee:
a4:a4:61:f1:d7:5b:7c:a9:a8:92:dc:77:74:f4:4a:
c0:4a:90:71:0f:6d:9e:e7:4f:11:ab:a5:3d:cd:4b:
8b:79:fe:82:1b:16:27:94:8e:35:37:db:dd:b8:fe:

```

```

fa:6d:d9:be:57:f3:ca:f3:56:aa:be:c8:57:a1:a8:
c9:83:dd:5a:96:5a:6b:32:2d:5e:ae:da:fc:85:76:
bb:77:d5:c2:53:f3:5b:61:74:e7:f3:3e:4e:ad:10:
7d:4f:ff:90:69:7c:1c:41:2f:67:e4:13:5b:e6:3a:
a3:2f:93:61:3b:07:56:59:5a:d9:bc:34:4d:b3:54:
b5:c6:e5:0a:88:e9:62:7b:4b:85:d2:9e:4c:ee:0b:
0d:f4:72:b1:1b:44:04:93:cf:cc:bb:18:31:3a:d4:
83:4a:ff:15:42:2d:91:ca:d0:cb:36:d9:8d:62:c0:
41:59:1a:93:c7:27:79:08:94:b2:a2:50:3c:57:27:
33:af:f0:b6:92:44:49:c5:09:15:a7:43:2a:0f:a9:
02:61:b3:66:4f:c3:de:d3:63:1e:08:b1:23:ea:69:
90:db:e8:e9:1e:21:84:e0:56:e1:8e:a1:fa:3f:7a:
08:0f:54:0a:82:41:08:6b:6e:bb:cf:d6:5b:80:c6:
ea:0c:80:92:96:ab:95:5d:38:6d:4d:da:38:6b:42:
ef:7c:88:58:83:88:6d:da:28:62:62:1f:e5:a7:0d:
04:9f:0d:d9:52:39:46:ba:56:7c:1d:77:38:26:7c:
86:69:58:4d:b0:47:3a:e2:be:ee:1a:fc:4c:de:67:
f3:d5:fe:e6:27:a2:ef:26:86:19:5b:05:85:9c:4c:
02:24:76:58:42:1a:f8:e0:e0:ed:78:f2:8f:c8:5a:
20:a9:2d:0b:d4:01:fa:57:d4:6f:1c:0a:31:30:8c:
32:7f:b0:01:1e:fe:94:96:03:ee:01:d7:f4:4a:83:
f5:06:fa:60:43:15:05:9a:ca:88:59:5c:f5:13:09:
      82:69:7f
Exponent: 65537 (0x10001)

```

X509v3 extensions:

X509v3 Subject Key Identifier:

3D:3D:3D:61:E6:88:09:FE:34:0F:1D:5E:5E:52:72:71:C7:
DE:15:92

X509v3 Authority Key Identifier:

keyid:3D:3D:3D:61:E6:88:09:FE:34:0F:1D:5E:5E:52:72:71:
C7:DE:15:92

X509v3 Basic Constraints: critical

CA:TRUE

X509v3 Key Usage: critical

Digital Signature, Certificate Sign, CRL Sign
Signature Algorithm: sha256WithRSAEncryption

28:24:86:91:a6:4a:88:e4:8d:6b:fc:67:2a:68:08:67:35:e5:
a6:77:ff:07:4b:89:53:99:2e:6d:95:df:12:81:28:6a:8e:6f:
5a:98:95:5b:4a:21:ae:f0:20:a4:4e:06:b2:4e:5a:67:c1:6a:
06:f1:0f:c1:f7:7e:f2:e0:b3:9d:d8:54:26:6a:b2:1c:19:b8:
b5:5c:c7:03:6b:f7:70:9e:72:85:c9:29:55:f9:f4:a4:f2:b4:
3b:3d:ce:25:96:67:32:1e:8d:e2:00:22:55:4b:05:4f:ee:0e:
67:ac:db:1b:61:da:5f:9c:10:1c:0c:05:66:c0:5b:5f:b9:95:
59:a9:58:5b:e7:69:ac:b0:bd:b3:c2:a3:35:58:01:a4:ff:c0:
8d:ac:1c:19:21:41:50:fb:8e:e0:f5:a9:ad:ec:de:cb:53:04:
a9:d8:ac:76:8a:09:0d:7c:c6:1a:bc:06:74:bb:10:1c:aa:07:
f6:cb:b2:1b:0c:0c:65:03:45:2b:51:d5:6e:a0:4d:91:ce:c5:
ed:8d:a9:e7:f6:37:7d:ab:1b:a4:a2:a3:3b:76:17:5b:d9:3a:
9c:c1:df:cc:cd:a0:b0:a9:5c:74:61:d7:a0:1d:04:67:68:ee:
a6:7b:1e:41:a4:02:fc:65:9e:e3:c1:c2:57:b2:2e:b0:ff:a9:
86:82:35:4d:29:b2:fe:74:2e:b8:37:5d:2b:e8:69:f2:80:29:
19:f1:1e:7a:5d:e3:d2:51:50:46:30:54:7e:b8:ad:59:61:24:
45:a8:5a:fe:19:ff:09:31:d0:50:8b:e2:15:c0:a2:f1:20:95:
63:55:18:a7:a2:ad:16:25:c7:a3:d1:f2:e5:be:6d:c0:50:4b:
15:ac:e0:10:5e:f3:7b:90:9c:75:1a:6b:e3:fb:39:88:e4:e6:
9f:4c:85:60:67:e8:7d:2e:85:3d:87:ed:06:1d:13:0b:76:d7:
97:a5:b8:05:76:67:d6:41:06:c5:c0:7a:bd:f4:c6:5b:b2:fd:
23:6f:1f:57:2e:df:95:3f:26:a5:13:4d:6d:96:12:56:98:db:
2e:7d:fd:56:f5:71:b7:19:2b:c9:de:2d:b9:c8:17:cc:20:de:
7c:19:7a:aa:12:97:1c:80:b7:d3:67:d3:b7:a7:96:f0:c9:4d:
f5:8b:0e:10:3b:b9:4e:09:90:5a:3b:51:c9:48:a2:ca:9f:db:
72:44:87:59:db:49:fa:75:44:b5:f6:7f:c5:26:e1:01:ae:7b:
6f:4a:75:d1:b5:b3:68:c0:31:48:f8:5c:06:c0:f1:b4:96:e8:
38:e8:ad:44:3d:0a:8c:03:b6:2c:86:6a:f0:39:de:84:4b:2e:
91:18:d1:45:65:d8:64:f5

Create a configmap in the Kubernetes Cluster

After downloading the certificate locally on your machine, create a configmap in the Kubernetes cluster that will copy the root CA certificate on your local machine into the Kubernetes cluster.



IMPORTANT

When running the command to create a configmap, the C-VEN and Kubelink require the file to have the **.crt** extension not work.

To create configmap, use the following command:

```
$ kubectl -n illumio-system create configmap root-ca-config \
  --from-file=./certs/root.democa.illumio-demo.com.crt
```

The **--from-file** option points to the path where the root CA certificate is stored on your local machine.

To verify that configmap was created correctly, use the following command:

```

$ kubectl -n illumio-system create configmap root-ca-config \
> --from-file=./certs/root.democa.illumio-demo.com.crt
configmap/root-ca-config created
$
$ kubectl -n illumio-system get configmap
NAME                                DATA    AGE
root-ca-config                     1        12s
$
$ kubectl -n illumio-system describe configmap root-ca-config
Name:                             root-ca-config
Namespace:                        illumio-system
Labels:                           <none>
Annotations:                       <none>

Data
====
root.democa.illumio-demo.com.crt:
----
-----BEGIN CERTIFICATE-----
MIIGSzCCBD0gAwIBAgIUAPw0NfPAivJW4YmKZ499eHZH3S8wDQYJKoZIhvcNAQEL
L
---output suppressed---
wPG0lug46K1EPQqMA7YshmrwOd6ESy6RGNFFZdhk9Q==
-----END CERTIFICATE-----

Events:                            <none>
$

```

`root-ca-config` is the name used to designate configmap. You can modify it according to your naming convention.

Configure Calico in Append Mode

In case your cluster is configured with Calico as the network plugin (usually for Kubernetes and not for OpenShift), both Calico and Illumio Core will write iptables rules on the cluster nodes.

- Calico - Needs to write iptables rules to instruct the host how to forward packets (overlay, IPsec, NAT, and so on).
- Illumio Core - Needs to write iptables rules to secure communications between nodes and/or Pods.

You should establish a hierarchy to ensure the firewall coexistence works smoothly, as Illumio and Calico will write rules simultaneously. By default, both solutions are configured to insert rules first in the iptables chains and tables, and Illumio will remove other rules added by third-party software (in Exclusive mode).

To allow Calico to write rules along with Illumio without flushing rules from one another, you should:

- Configure Illumio to operate in Firewall Coexistence mode (the default for workloads that are part of a container cluster).
- Configure Calico to operate in Append mode (the default is Insert mode).

To configure Calico to work in Append mode with iptables:

1. Edit the Calico DaemonSet.

```
kubectl -n kube-system edit ds calico-node
```

2. Locate the `spec: > template: > spec: > containers:` section inside the **YAML** file and change `ChainInsertMode` by adding the following code block:

```
- name: FELIX_CHAININSERTMODE
  value: Append
```

3. Save your changes and exit.
4. Kubernetes will restart all Calico Pods in a rolling update.

For more information on changing Calico `ChainInsertMode`, see [Calico documentation](#).

Create a Container Cluster in the PCE

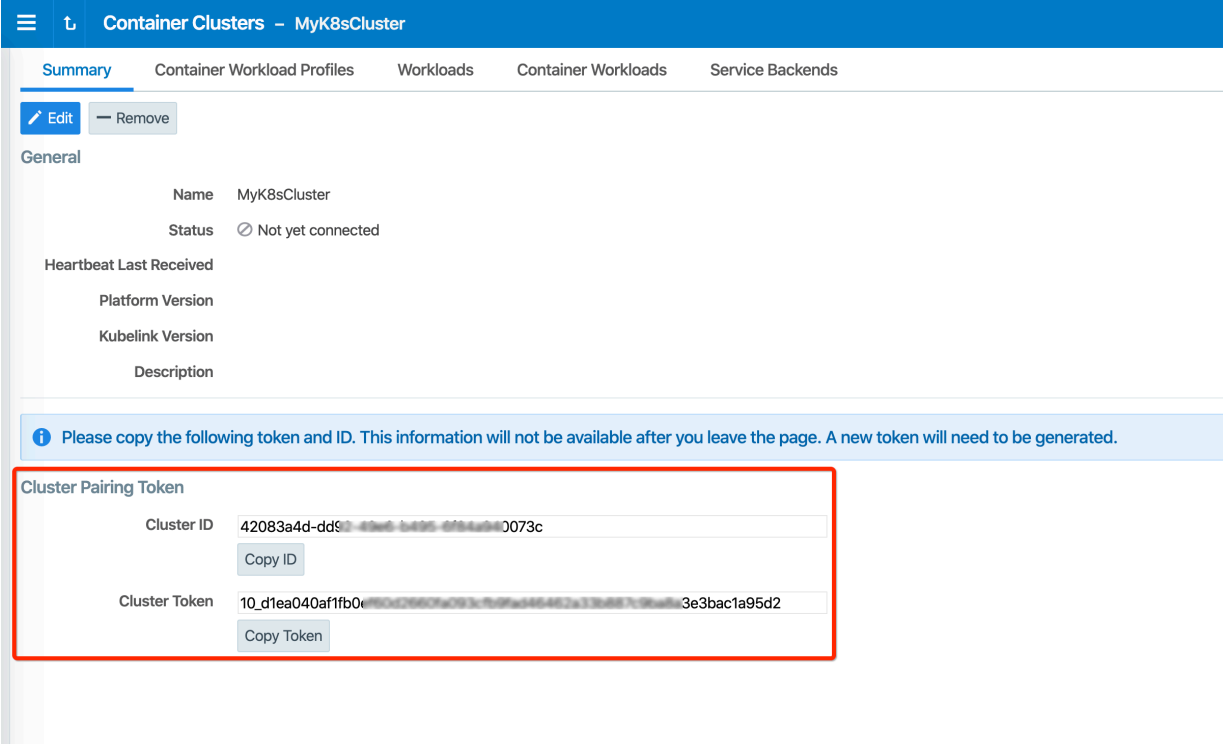
To provide visibility and enforcement to your containerized environment, you first need to create a container cluster in the PCE. Each container cluster maps to an existing Kubernetes or OpenShift cluster.

Create a Container Cluster

To create a new container cluster:

1. Log into the PCE web console as a user with Global Organization Owner privileges.
2. From the PCE web console menu, navigate to **Infrastructure > Container Clusters**.
3. Click **Add**.
 - a. Add a Name.
 - b. **Save** the Container Cluster.
4. You will see a summary page of the new Container Cluster. From the Cluster Pairing Token section, copy the values of the Cluster ID and Cluster Token.

- After copying and saving the values (in a text editor or similar tool), open the Container Workload Profiles page.



The screenshot shows the 'Container Clusters - MyK8sCluster' page. The 'Summary' tab is selected. Below the 'Edit' and 'Remove' buttons, the 'General' section displays the cluster's name, status, and various version details. A blue information banner states: 'Please copy the following token and ID. This information will not be available after you leave the page. A new token will need to be generated.' Below this, the 'Cluster Pairing Token' section is highlighted with a red box. It contains two rows: 'Cluster ID' with the value '42083a4d-dd9e-43a5-b495-df84a240073c' and a 'Copy ID' button, and 'Cluster Token' with the value '10_d1ea040af1fb0c-8804236c7ac093a7b9f9a4454622a135a887c99a3e3bac1a95d2' and a 'Copy Token' button.

Configure a Container Workload Profile Template

When configuring a new Container Cluster, it is recommended to set the default settings shared by all the Container Workload Profiles. Illumio provides a Container Workload Profile template that can be used for that purpose. By defining the default Policy State and minimum set of labels common to all namespaces in the cluster, you will save time later on when new namespaces are discovered by Kubelink. Each new profile created will inherit what was defined in the template.



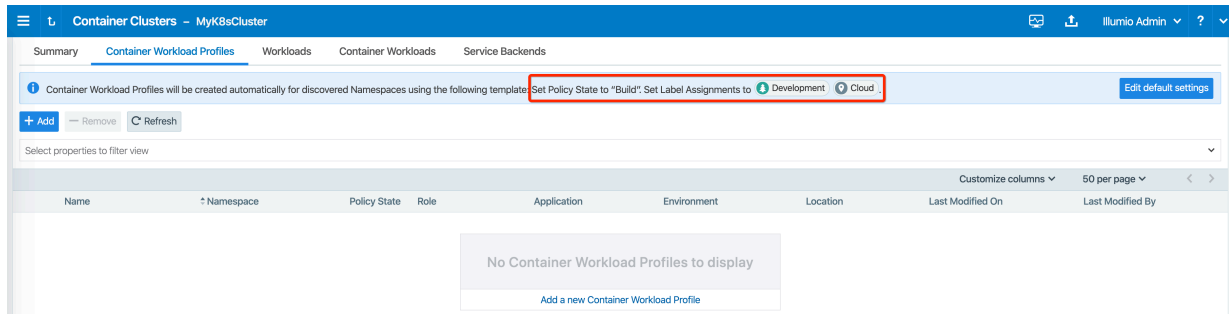
IMPORTANT

Illumio does not provide a method to redefine at once all the labels associated with each profile. Hence, it is **strongly recommended** to use the provided template to define the default values for all profiles that are part of the same cluster.

To define the default parameters for all profiles using a template, under *Container Workload Profiles*, click **Edit** default settings and select values for all the fields.

For information about assigning default labels in the template, see the "Labels Restrictions for Kubernetes Namespaces" topic.

After you click OK, the following information is displayed:



Deploy C-VEs in Your Cluster



IMPORTANT

Before deploying the C-VE, ensure that either of the following two requirements has been met:

- Kubelink is deployed on the Kubernetes cluster and is in sync with the PCE, or
- Firewall coexistence is enabled.

Prerequisites

- VEN deployment file provided by Illumio.
- VEN secret file provided by Illumio.
- Illumio's C-VE docker image is uploaded to a private container registry.
- In OpenShift, create the 'illumio-ven' service account in the 'illumio-system' project and add this account to the privileged Security Context Constraint (SCC):
 - `oc create sa illumio-ven`
 - `oc adm policy add-scc-to-user privileged -z illumio-ven -n illumio-system`

Create a Pairing Profile for Your Cluster Nodes

Before deploying the C-VEN in your cluster, you should create a pairing profile to pair the cluster nodes with the PCE. You only need to create one pairing profile for all your nodes.



NOTE

You only need to create pairing profiles for Kubernetes or OpenShift nodes and not for container workloads.

For ease of configuration and management, consider applying the same Application, Environment, and Location labels across all nodes of the same Kubernetes or OpenShift cluster. The screenshot below shows an example of a pairing profile for a Kubernetes cluster.

Pairing Status	Name	Policy State	Role	Application	Environment	Location	Last Modified On	Last Modified By	Last Used On	Description
Running	Kubernetes Nodes	Build	Node	Kubernetes Cluster	Development	Cloud	03/22/2020, 22:39:13		02/04/2020, 03:42:52	



TIP

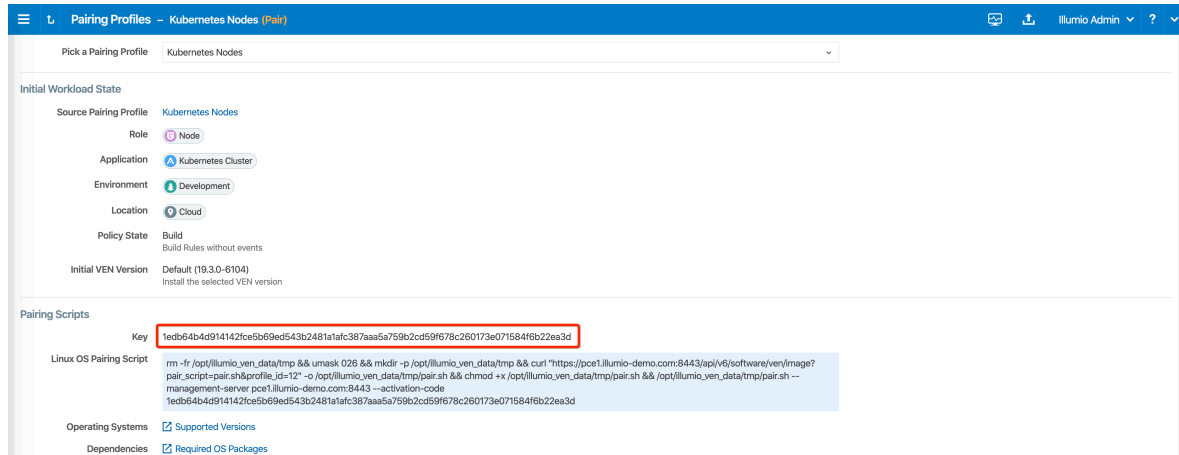
Illumio recommends that all pairing profiles for Kubernetes nodes not use the Full enforcement policy state. Use Idle or Visibility Only mode for initial configuration.

You should only move them into Full enforcement state after completing all other configuration steps in this guide, such as setting up Kubelink, discovering services, and writing rules.

Configure C-VEN Secret

This section assumes that you have already created a Pairing Profile in the PCE. You will need the activation code for the C-VEN secret.

1. To retrieve the activation code from the pairing profile, go to **Policy Objects > Pairing Profiles**, open the pairing profile created for your cluster nodes, and click **Generate Key**.



2. After copying and saving the **Key** (in a text editor or similar tool), you can exit the page.
3. Open the C-VEN secret YAML file and modify the following keys (under **stringData**):
 - **ilo_server** = PCE URL and port. Example: mypce.example.com:8443
 - **ilo_code** = Activation code value from Step 1. Example:
1edb64b4d914142fce5b69ed543b2481a1afc387aaa5a759b2cd59f678c260173e071584f6b22ea3d

Contents of a modified **illumio-ven-secret.yml** file are shown below.

```
#
# Copyright 2013-2021 Illumio, Inc. All Rights Reserved.
#
# VEN 21.5.x-xxxx

apiVersion: v1
kind: Secret
metadata:
  name: illumio-ven-config
  namespace: illumio-system
type: Opaque
stringData:
  ilo_server: mypce.example.com:8443 # Example:
  mypce.example.com:8443
  ilo_code:
  1edb64b4d914142fce5b69ed543b2481a1afc387aaa5a759b2cd59f678c26
  0173e071584f6b22ea3d # activation-code
```

**CAUTION**

Do not use 'https://' for the value associated with the **ilo_server:** key. This is a known issue and will be fixed in a future release.

4. Save the changes.
5. Create the C-VEN secret using the file.
 - To create the secret for Kubernetes:

```
kubectl apply -f illumio-ven-secret.yml
```

- To create the secret for OpenShift:

```
oc apply -f illumio-ven-secret.yml
```

6. Verify the C-VEN secret creation in your cluster.
 - To verify the creation of the secret for Kubernetes:

```
kubectl get secret -n illumio-system
```

```
oc get secret -n illumio-system
```

Deploy C-VEs

Modify the C-VEN configuration file to point to the correct Docker image. The example in this document has **illumio-ven:21.5.x-xxxx** uploaded to **registry.example.com:443**, so the image link in this example is: **registry.example.com:443/illumio-ven:21.5.x-xxxx**

1. Edit the C-VEN configuration YAML file. The file name is **illumio-ven-kubernetes.yml** for a Kubernetes cluster and **illumio-ven-open-shift.yml** for an OpenShift cluster.
 - Locate the **spec: > template: > spec: > containers:** section inside the YAML file. Modify the image link in the **image:** attribute.

2. Save the changes.

Below is a snippet from an example of the C-VEN configuration for Kubernetes or OpenShift to illustrate the image location.

```

#
# Copyright 2013-2021 Illumio, Inc. All Rights Reserved.
#
# VEN 21.5.x-xxxx

---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: illumio-ven
  namespace: illumio-system
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: illumio-ven
  namespace: illumio-system
  labels:
    k8s-app: illumio-ven
spec:
  selector:
    matchLabels:
      name: illumio-ven
  template:
    metadata:
      labels:
        name: illumio-ven
    spec:
      priorityClassName: system-node-critical
      serviceAccountName: illumio-ven
      hostNetwork: true
      hostPID: true
      tolerations:
        - key: node-role.kubernetes.io/master
          effect: NoSchedule
      containers:
        - name: illumio-ven
          env:
            - name: ILO_SERVER
              valueFrom:
                secretKeyRef:
                  name: illumio-ven-config
                  key: ilo_server
            - name: ILO_CODE
              valueFrom:
                secretKeyRef:

```

```

        name: illumio-ven-config
        key: ilo_code
    command: [ "/ven-init", "activate" ]
    image: registry.example.com/illumio-ven:21.5.x-xxxx
    imagePullPolicy: IfNotPresent
<...>

```

3. (Optional) Reference your root CA certificate.

If you are using a private PKI to sign your PCE certificate, ensure that you include the references to the root CA certificate that signed the PCE certificate. If Kubelink is already deployed in the cluster, the ConfigMap used to store the root CA certificate should already be created in the cluster.

Add the following sections to the C-VEN manifest file to reference the ConfigMap containing the root CA certificate:

- **volumeMounts** (under **spec.template.spec.containers**)
- **volumes** (under **spec.template.spec**)

root-ca is the name used to designate the new volume mounted in the container. You can modify it according to your naming convention.

```

    volumeMounts:
    - name: root-ca
      mountPath: /etc/pki/tls/ilo_certs/
      readOnly: false
    volumes:
    - name: root-ca
      configMap:
        name: root-ca-config

```

4. (Optional) Reference your container registry secret. See the [Create Illumio Namespace \[232\]](#) section.

In case you need to authenticate against your container registry when you pull an image from your cluster, you must make reference to the secret previously created for the container registry. Locate the **spec: > template: > spec:** section inside the YAML file and add the following lines:

```

imagePullSecrets:
- name: <container-registry-secret-name>

```



IMPORTANT

Indentation matters in a YAML file. Make sure there are 6 spaces to the left before inserting the 'imagePullSecrets' keyword and align the '-' character below it with the 'i' of the 'imagePullSecrets' keyword.

5.

**NOTE**

From the 20.2.0 and later releases, the container runtime detection is done automatically. You do not need to manually modify the container runtime socket path. You should do this 'Modify the container runtime socket path' step only if you are using a customized configuration for your container runtime.

(Optional) Modify the container runtime socket path.

In some cases, you have to modify the default socket path that C-VEN relies on to get information about the containers due to the following reasons:

- A non-conventional or customized container runtime socket path
- Two concurrent container runtimes

In this case, you may need to modify the default mount path for the unixsocks volume in the C-VEN configuration file.

For example, you want to listen on the 'containerd' container runtime, however, docker is also used on the nodes. You should modify the file as shown below, so that the C-VEN listens to events on 'containerd':

```

    volumeMounts:
      - name: unixsocks
        mountPath: /var/run/containerd/
    <...>
  volumes:
    - name: unixsocks
      hostPath:
        path: /var/run/containerd/
        type: Directory
    <...>

```

6. Save the changes.

7. Deploy C-VEN.

- For Kubernetes:

```
kubectl apply -f illumio-ven-kubernetes.yml
```

- For OpenShift:

```
oc apply -f illumio-ven-openshift.yml
```

8. Verify the deployment.

- For Kubernetes:

```
kubectl get pods -n illumio-system
```

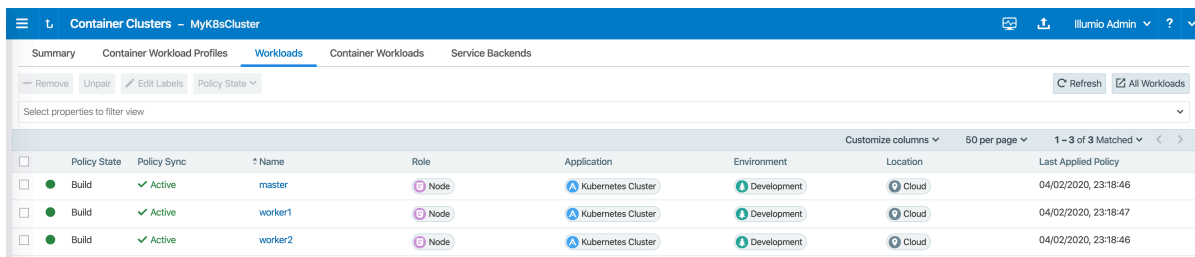
```
oc get pods -n illumio-system
```

The `illumio-ven-xxxxxxxxxx-xxxxx` Pods should be in the "Running" state.

After C-VEs are successfully deployed, you can check the cluster information in the Illumio PCE UI. From the main menu, navigate to **Infrastructure > Container Clusters**.

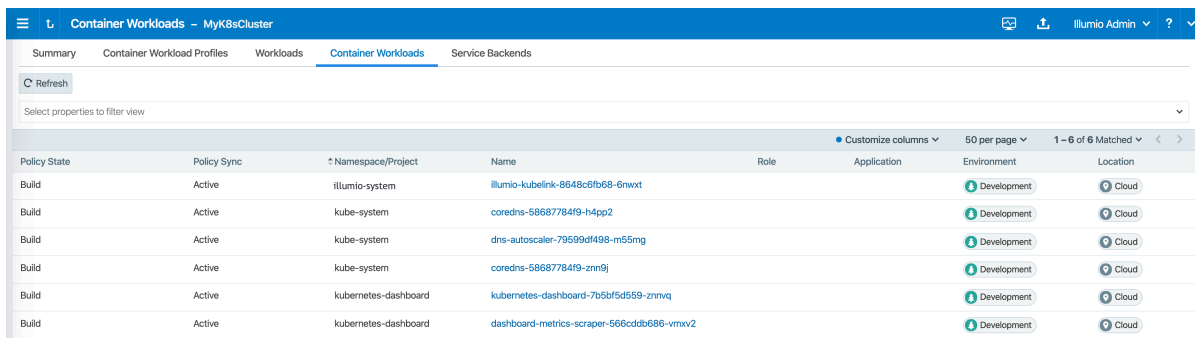
You can also verify in the PCE UI that the C-VEs were successfully deployed by checking the following:

- Under the **Workload** tab, nodes that are part of your Kubernetes or OpenShift cluster should be listed. An example is shown below.



Policy State	Policy Sync	Name	Role	Application	Environment	Location	Last Applied Policy
Build	Active	master	Node	Kubernetes Cluster	Development	Cloud	04/02/2020, 23:18:46
Build	Active	worker1	Node	Kubernetes Cluster	Development	Cloud	04/02/2020, 23:18:47
Build	Active	worker2	Node	Kubernetes Cluster	Development	Cloud	04/02/2020, 23:18:46

- Under the **Container Workloads** tab, Pods deployed in your Kubernetes or OpenShift cluster should be listed. An example is shown below.



Policy State	Policy Sync	Namespace/Project	Name	Role	Application	Environment	Location
Build	Active	illumio-system	illumio-kubelink-8648c6fb68-6nwxt			Development	Cloud
Build	Active	kube-system	coredns-58687784f9-h4pp2			Development	Cloud
Build	Active	kube-system	dns-autoscaler-79599df498-m55mg			Development	Cloud
Build	Active	kube-system	coredns-58687784f9-znn0j			Development	Cloud
Build	Active	kubernetes-dashboard	kubernetes-dashboard-7b5bf5d559-znnvq			Development	Cloud
Build	Active	kubernetes-dashboard	dashboard-metrics-scraper-566cddb686-vmxv2			Development	Cloud

- Illumination Map now displays system and application Pods running in your cluster.

Re-Label Your Cluster Nodes



NOTE

Re-labeling the cluster nodes is optional.

In the case of self-managed deployments in which both Master and Worker nodes are managed, you may want to re-label your nodes to differentiate

Master nodes from Worker nodes. Doing this helps when you are writing different policies for the Worker and Master nodes or if you want to segment these nodes differently.

To re-label your cluster nodes:

1. In the PCE UI, go to **Infrastructure > Container Clusters > YourCluster-Name > Workloads**.
2. Select the workloads you want to re-label.
3. Click **Edit Labels** to assign the new labels (for example, Master and Worker).

Policy State	Policy Sync	Name	Role	Application	Environment	Location	Last Applied Policy
Build	Active	master	Master	Kubernetes Cluster	Development	Cloud	04/02/2020, 23:56:43
Build	Active	worker1	Worker	Kubernetes Cluster	Development	Cloud	04/02/2020, 23:57:00
Build	Active	worker2	Worker	Kubernetes Cluster	Development	Cloud	04/02/2020, 23:57:00

4. After re-labeling your cluster nodes, the nodes part of the cluster reflect the updated label(s).

Configure Security Policies for Containerized Environments

Security policies are a set of rules that you can configure to secure your Kubernetes or OpenShift environment. You can follow the guidelines and examples described in this section to write rules for your Kubernetes or OpenShift clusters and containerized applications, which you can then modify incrementally.

IP and FQDN Lists

FQDN Services for Kubernetes

There are some basic services that need to be defined as IP lists, such as docker.io or the Kubernetes API server. These FQDNs will be used later in the ring-fence policy for the Kubernetes cluster. The following FQDNs are commonly found to be dependencies for Kubernetes and should be defined inside Illumio Core's IP list policy objects:

- docker.io

- myregistry.example.com

The PCE FQDN is required for Kubelink for example, mypce.example.com.

IP Lists for Kubernetes

Additionally, the following subnets or IP addresses should be defined in the IP list policy objects:

- **Kubernetes Pod Network:** Locate subnet in master node's `/etc/kubernetes/kubeadm-config.yaml` file (Ubuntu) under `networking: > pod-Subnet:` section, for example, `10.200.0.0/16`
- **Kubernetes Service Network:** Locate subnet in master node's `/etc/kubernetes/kubeadm-config.yaml` file (Ubuntu) under `networking > serviceSubnet` section, for example, `10.100.0.0/16`

The screenshot below displays IP lists created for Kubernetes Infrastructure dependencies.

IP Lists		
+ Add ↑ Provision ↶ Revert — Remove 📄 Reports ↻ Refresh		
Select properties to filter view		
☐	Provision Status	Name
		Addresses
☐		Any (0.0.0.0/0 and ::/0)
		0.0.0.0/0 +1 more
☐		Docker Registry
		myregistry.example.com
☐		docker.io
		docker.io
☐		Kubernetes Pod Network
		10.200.0.0/16
☐		Kubernetes Service Network
		10.100.0.0/16
☐		PCE
		mypce.example.com

FQDN Services for OpenShift

There are some basic services that should be defined as IP lists such as docker.io or the Kubernetes API server. These FQDNs will be used later in the ring fence policy for the OpenShift cluster. The following FQDNs are commonly found to be dependencies for OpenShift and should be defined in Illumio IP list policy objects:

- docker.io
- registry.access.redhat.com
- access.redhat.com
- subscription.rhsm.redhat.com
- github.com

The PCE FQDN is required for Kubelink, for example, mypce.example.com.

IP Lists for OpenShift

Additionally, the following subnets or IP addresses should be defined in IP list policy objects:

- **OpenShift Pod Network:** Find subnet in master node's `/etc/origin/master/master-config.yaml` file under `networkConfig > clusterNetworkCIDR` section, for example, `10.128.0.0/14`
- **OpenShift Service Network:** Find subnet in master node's `/etc/origin/master/master-config.yaml` file under `networkConfig > serviceNetworkCIDR` section, for example, `172.30.0.0/16`

The screenshot below displays IP lists created for OpenShift Infrastructure dependencies. It references the IP lists which automatically come with the Illumio Segmentation Template.

+ Add Provision Revert Remove Reports Refresh Select properties to filter view		
Customize		
☐	Provision Status	Name
☐		access.redhat.com
		Any (0.0.0.0/0 and ::/0)
☐		docker.io
☐		Openshift Pod Network
☐		Openshift Service Network
☐		PCE
☐		registry.access.redhat.com
☐		subscription.rhsm.redhat.com

**NOTE**

The IP lists mentioned above are for FQDNs and IP addresses that Illumio has found to be necessary for basic Kubernetes or OpenShift deployments. Each deployment varies and may have dependencies on additional FQDNs or IP addresses that are not mentioned in this document.

If your Kubernetes or OpenShift infrastructure needs to communicate with external services that are not mentioned here, then make sure you describe those in the IP lists.

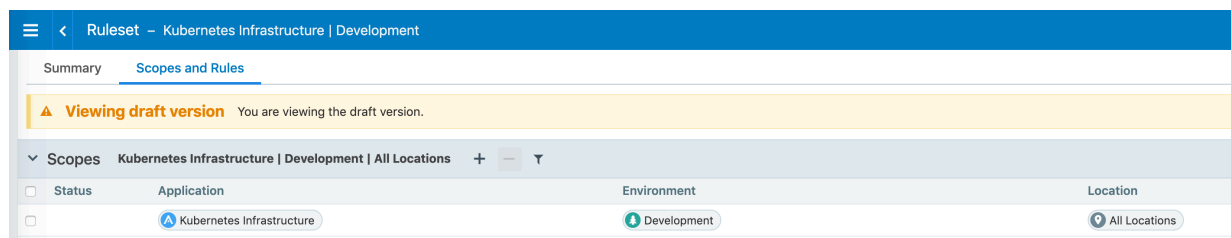
Rules for Kubernetes or OpenShift Clusters

This section assumes the following:

- Kubernetes or OpenShift cluster nodes and infrastructure Pods are activated and managed.
- Labels have been assigned to each workload and container workload.
- All cluster nodes and infrastructure Pods are in the same application group, which means they have been assigned the same application, environment, and location labels.

Kubernetes

Create a ruleset for the Kubernetes cluster and control plane Pods. The labels assigned to all of the Kubernetes nodes and control Pod workloads should fall within the scope.



Add the following lines of policy to the ruleset.

Intra-Scope Rules

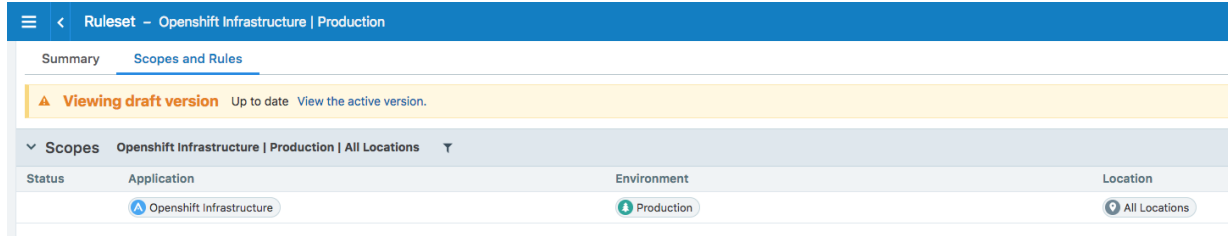
Sources	Serv-ices	Destinations	Notes
All Work-loads	All Serv-ices	docker.io (IP List) myregis-try.exam-ple.com (IP List)	Containerized environments depend on various external resources to perform basic operations such as pulling a docker image. Illumio has determined that the listed FQDNs are essential to Kubernetes deployments. Each deployment varies and may have dependencies on additional resources. If your container infrastructure has requirements for FQDNs not mentioned in this document, then you should include those FQDNs in this policy line.
Kubelink	8443 TCP	All Work-loads	Kubelink sends context about the Kubernetes cluster to the PCE over TCP 8443 port.
Kuber-netes Pod Net-work (IP List)	53 TCP 53 UDP	All Work-loads	The Kubernetes cluster provides internal DNS services to the pods (using coreDNS in this example). This policy enables internal DNS resolution for these tasks.
All Work-loads	All Serv-ices	All Work-loads (Uses Virtual Services and Workloads)	Any communication across all managed Kubernetes nodes or managed infrastructure pods which will be permitted by this policy.
All Work-load	All Serv-ices	Kubernetes Pod Network (IP List)	Communications across initiated by any workload which pass through service front ends will be allowed by this policy. It also covers other IP addresses on the Kubernetes pod network which are not discovered by the PCE. Critical for infrastructure functions including but not limited to liveness probes and infrastructure service front ends (Kubernetes).

Extra-scope Rules

Sources	Serv-ices	Desti-nations	Notes
Any 0.0.0.0/0 (IP List)	6443 TCP 22 TCP	All Work-loads	Optional: Opens ports used for remote manage-ment. For example, TCP 22 is used to provide SSH services to Kubernetes admins. TCP 6443 pro-vides Kubernetes admins with dashboard services. The Dashboard may vary across Kubernetes deploy-ments. The ports can be modified to match your environment, and the consuming IP list can be up-dated to include corporate subnets or jump servers.
Any 0.0.0.0/0 (IP List)	80 TCP 443 TCP	Worker	This policy assumes Ingress Controllers exist on Worker nodes. If the ingress controllers exist on oth-er nodes, modify the source to point to the host where the Ingress controllers reside. This rule opens default front-end ports, which are used to access containerized applications from external IP address-es.

OpenShift

Create a ruleset for the OpenShift cluster and control plane Pods. The labels assigned to all of the OpenShift nodes and control Pod workloads should fall within the scope.



Add the following lines of policy to the ruleset.



NOTE

The IP lists referenced in this ruleset are commonly used public registries (e.g., docker.io) for container environments. If you have confirmed that your OpenShift environment does not depend on a public registry shown below, then it is recommended that you remove the IP lists from the ruleset.

Intra-scope Rules

Sources	Serv-ices	Destinations	Notes
All Workloads	All Serv-ices	docker.io (IP List) registry.access.redhat.com (IP List) registry.webscaleone.info (IP List) access.redhat.com (IP List) subscription.rhsm.redhat.com (IP List)	Containerized environments depend on various external resources to perform basic operations such as pulling a Docker image. Illumio has determined that the listed Fully Qualified Domain Names (FQDNs) are essential to OpenShift deployments. Each deployment varies and may have dependencies on additional resources. If your container infrastructure has requirements for FQDNs not mentioned in this doc, then you should include those FQDNs in this policy line.
Kubelink	8443 TCP	Illumio PCE (IP List)	Kubelink sends context about the OpenShift cluster to the PCE over TCP port 8443.
OpenShift Pod Network (IP List)	53 TCP 53 UDP	All Workloads	The OpenShift cluster in this example uses DNSmasq, meaning each cluster node listens on port 53 and provides internal DNS services to the pods. This policy enables internal DNS resolution for these tasks.
All Workloads	All Serv-ices	All Workloads (Uses Virtual Services and Workloads)	This policy permits communication across all managed OpenShift nodes and managed infrastructure pods.
All Workloads	All Serv-ices	OpenShift Pod Network (IP List) OpenShift Service Network (IP List)	This policy will permit communications initiated by any workload that passes through service frontends. It also covers other IP addresses on the OpenShift pod network, which the PCE does not discover. Critical for infrastructure functions, including but not limited to liveness probes and infrastructure service front ends (Kubernetes).

Extra-Scope Rules

Sources	Serv-ices	Desti-nations	Notes
Any 0.0.0.0/0 (IP List)	8443 TCP 22 TCP	All Work-loads	Optional: Opens ports used for remote management. For example, TCP 22 is used to provide SSH services to OpenShift admins. TCP 8443 provides OpenShift admins with web console services. Webconsole may vary across OpenShift deployments. The ports can be modified to serve other remote management services, and the IP list consumed can be changed to corporate subnets or jump servers.
Any 0.0.0.0/0 (IP List)	TCP 80 TCP 443	Infra (Role)	This policy assumes the router exists only on dedicated Infra nodes. If the router exists on other nodes, then modify the Source to the host where the router resides. This rule opens default front-end router ports, which are used to access containerized applications from external IP addresses. As you start to open up application pods to the outside world, you will need to add the application's exposed port to this policy's list of services. For example, you spin up an HTTP server and expose it on TCP port 8080. The first step to allow access to the httpd server from outside is to add TCP 8080 to this line of policy.

**NOTE**

The IP lists referenced in the rulesets are commonly used public registries (for example, docker.io) for container environments. If you have confirmed that your Kubernetes or OpenShift environment does not depend on the public registries mentioned above, then it is recommended that you remove the IP lists from the ruleset.

Rules and Traffic Considerations with CLAS

In Container Local Actor Store (CLAS) deployments, be sure to take into account the following special traffic and policy rules considerations that differ from legacy non-CLAS environments that are described in other sections of this chapter.

Mandatory Rules

The CLAS architecture requires mandatory infrastructure rules to be in place for the cluster to work properly. Do not upgrade to the CLAS mode, or move the cluster to Full Enforcement until the following infrastructure rules are configured:

- Node -> Service CIDR IP list (9000/TCP)
- Any (0.0.0.0/0 and ::0) -> Node (2379-2380/TCP, 2379-2380/UDP)

Mandatory rules for namespace enforcement

The following rules must be set on the PCE to ensure the illumio-system namespace is fully enforced (or the custom namespace that is used instead of the default of illumio-system):

- illumio-kubelink-* → Any; 2379 TCP (pod to arbitrary IP address)
This rule is necessary for connecting a illumio-kubelink-* pod to a illumio-storage-* pod. All destination IP addresses are allowed, because the IP address of illumio-storage-* pod changes after it is restarted.
- Any → illumio-storage-*; 2379 TCP (arbitrary IP address to pod)
The same explanation as for the above rule, but the ingress on illumio-storage-* pod is allowed here.
- Node → illumio-kubelink-*; 8080 TCP, 8081 TCP, 9000 TCP (node to pod)
This rule is needed for successfully connecting C-VEN pods to illumio-kubelink-* pods.
- illumio-kubelink-* → PCE FQDN; 8443 TCP (pod to FQDN IP list)
The illumio-kubelink-* pod must be able to connect to PCE. Use the FQDN IP list, which contains the URL of the PCE, to do this. The outgoing connections using the DNS port are always allowed by an implicit rule installed on C-VEN pods. Therefore you need not list any rules for DNS.
- illumio-kubelink-* → Kubernetes API; 443 TCP (for example, pod to IP list of all possible ClusterIPs of Kubernetes Services)
This rule ensures the illumio-kubelink-* pod can connect to the Kubernetes API, which is required.

ClusterIP Rules

In the CLAS environment, ClusterIP ports are now represented as Kubernetes Workloads when viewing traffic, rather than as Virtual Services, as previously.

If you want to migrate from a legacy (non-CLAS) to a CLAS environment, ensure that all rules applicable to ClusterIP Services are changed to "Use Workloads" at a specific point during the CLAS upgrade process. For complete details, see [Upgrade to CLAS Architecture \[290\]](#).

Because Services are now Kubernetes Workloads, the "All Workloads" flag in a rule will include all Services. Do not use "All Workloads" as a Destination in a rule. Use a more specific label instead that targets the Service.

All rules that include a label of at least one ClusterIP service will have specified ports internally replaced. However - this is not reflected in PCE UI, where the rule still displays ports.

NodePort and LoadBalancer services remain shown as Virtual Services in the PCE. However, the ClusterIP part of a NodePort or LoadBalancer service now exists as a Kubernetes Workload and is linked with the Virtual Service in the PCE.

General Traffic View Changes

The following is a summary of general changes to traffic views in a CLAS-enabled cluster:

- Kubernetes workloads (for example, Deployments) are now displayed in the UI as Kubernetes Workloads, rather than as Container Workloads. Container Workloads (Pods) are still shown in non-CLAS clusters.
- ClusterIP Virtual Services are now shown as Kubernetes Workloads.
- NodePort and LoadBalancer services remain Virtual Services in the PCE.
- Traffic from other Virtual Services to Kubernetes Workloads is not shown.
- Traffic between Kubernetes Workloads within a cluster is shown.

CLAS Traffic Limitations

Consider the following differences and limitations in these scenarios when viewing traffic and writing rules in a CLAS environment:

- **Pod to Host**

When a Pod is on a different Node than the target Node, additional traffic is shown occurring from the Pod's Node to the target Node. This traffic cannot be selectively hidden with filters because it behaves similarly to traffic from host to host that should not be hidden. (This behavior occurs only when Calico is used as the CNI.)

- **Pod to ClusterIP**

Additional direct traffic occurs from a source Pod to a source Target, regardless of whether it is destined for the same node or a different target node.

- **Managed Workload to NodePort**

Additional traffic is shown for a Client's direct access to a target Pod, and from a Used Node to a target Pod.

Additionally, crucial traffic destined for a NodePort is directed to the Node with the NodePort's port.

- **Unmanaged Workload (or Internet) to NodePort**

Additional traffic is shown for a Client's direct access to a target Pod, and from a Used Node to a target Pod.

Additionally, in the traffic from the Client to the NodePort virtual service, we are missing crucial traffic associated with the NodePort's port.

• **Draft Traffic to Virtual Services**

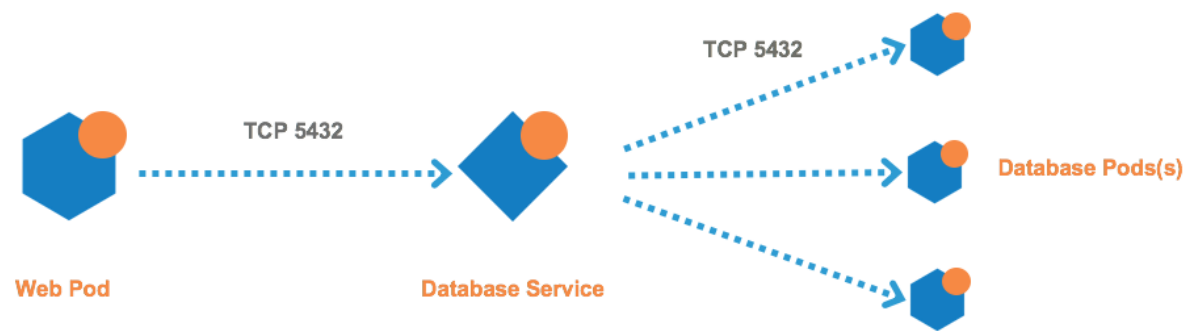
Traffic in Draft mode, where the destination is a Virtual Service, is marked as Potentially Blocked.

Rules for Containerized Applications

This section covers different scenarios on writing rules for containerized applications.

Access Services from within the Cluster

For connections to a service from within the cluster, the Pods connect to a Service IP and the connections get distributed to the Pods.



Kubernetes

The rules you need to write are:

Example Ruleset

Scope

Application	Environment	Location
Risk Assessment	Development	Cloud

Intra-Scope Rule

Source	Destination	Destination Service	Notes
Web (Role for Web pods)	Database (Virtual Service Role label for database service) + Use Virtual Services Only	Derived from Destination Virtual Service	After the database service gets discovered by the PCE it becomes a virtual service object in the PCE -- not a container workload. The Destination should be the role label of the virtual service plus the "Use Virtual Service Only" option. The Source in this example is the Web pod. Use the Web Role label which describes the pod. Leave the Providing Service empty. Once the rule is saved, it will automatically populate with <i>Derived from Destination Virtual Service</i> .



NOTE
This does not allow Web pods to directly access Database pods through the pod IP. This only allows traffic through the service.

OpenShift

The rules you need to write are:

Example Ruleset

Scope

Application	Environment	Location
Risk Assessment	Production	HQ

Intra-Scope Rule

Source	Destination	Destination Service	Notes
Web (Role for Web pods)	Database (Virtual Service Role label for database service) + Use Virtual Services Only	Derived from Destination Virtual Service	After the database service gets discovered by the PCE it becomes a virtual service object in the PCE -- not a container workload. The Destination should be the role label of the virtual service plus the "Use Virtual Service Only" option. The Source in this example is the Web pod. Use the Web Role label which describes the pod. Leave the Providing Service empty. Once the rule is saved, it will automatically populate with <i>Derived from Destination Virtual Service</i> .



NOTE
This does not allow Web pods to directly access Database pods through the pod IP. This only allows traffic through the service.

Access Services from Outside the Cluster

Kubernetes

With Kubernetes, connections to a containerized application from the outside world can be handled in many different ways. In this release, Illumio supports only configurations which expose applications via the Kubernetes NGINX ingress controller (HostNetwork type only).

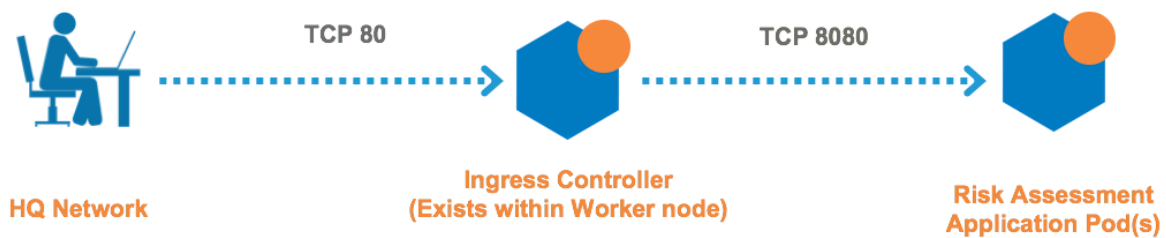
Exposing applications using HostPort is supported for deployments running PCE versions 24.5.0 and higher. HostPort is only supported by explicitly set ports. Setting port ranges or all services will not work.

In typical Kubernetes deployments, connections to a containerized application from the outside world go through the ingress controllers, then the connection goes directly from controllers to the pods - not the service. Example of scenario and rule coverage are shown below.

Scenario:

- The Kubernetes cluster and containerized applications are in the Development environment

- The containerized application is called RiskAssessment and each Pod within the application listens on TCP 8080
- The RiskAssessment application is exposed to the outside world via the ingress controller. The controller listens on TCP port 80 for the RiskAssessment application
- In Illumio, the RiskAssessment workloads (Pods) provide to the controller on TCP 8080. The controller provides TCP 80 to the outside world.



The rules you need to write are:

Example Ruleset 1

Scope

Application	Environment	Location
Risk Assessment	Development	Cloud

Intra-Scope Rule

Source	Destination	Destination Service
All Workloads	All Workloads	All Services

Extra-Scope Rule

Source	Destination	Destination Service	Notes
Worker	Risk Assessment	TCP 8080	The source should be the role label of the nodes which nest the Ingress controllers.

Example Ruleset 2

The second ruleset opens the ingress controller to the external network. The rule and ruleset below should have been created from the [Rules for Kubernetes or OpenShift Cluster \[253\]](#) section of this guide. You can modify the ruleset as needed.

Scope

Application	Environment	Location	Notes
Kubernetes Infrastructure	Development	Cloud	The scope of the ruleset should match the Kubernetes infrastructure scope.

Intra-Scope Rule

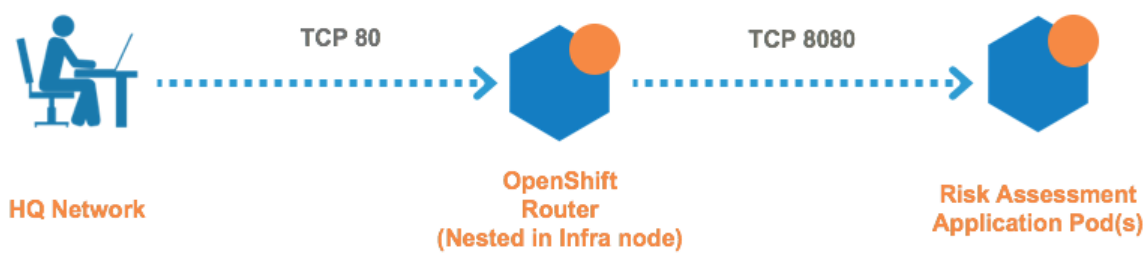
Source	Destination	Destination Service	Notes
External Network	Worker Node(s)	TCP 80	This rule should exist from the Rules for Kubernetes or OpenShift Cluster [253] section. The source should be the Kubernetes node(s) which contain the ingress controller. The destination can be an IP List such as 0.0.0.0/0 (any), HQ, Corporate, or employee subnet that requires connectivity into the exposed container workloads.

OpenShift

Connections to a containerized application from the outside world go through the OpenShift Router, then the connection goes directly from router to the Pods - not the service. Example of scenario and rule coverage are shown below.

Scenario:

- The OpenShift cluster and containerized applications are in the development environment
- The containerized application is called RiskAssessment and each Pod within the application listens on TCP 8080
- The RiskAssessment application is exposed to the outside world via the router. The router listens on TCP port 80 for the RiskAssessment application
- In Illumio, the RiskAssessment workloads (Pods) provide to the router on TCP 8080. The router provides TCP 80 to the outside world.



The rules you need to write are:

Example Ruleset 1

Scope

Application	Environment	Location
Risk Assessment	Production	HQ

Intra-Scope Rule

Source	Destination	Destination Service
All Workloads	All Workloads	All Services

Extra-Scope Rule

Source	Destination	Destination Service	Notes
IST Infra (Role)	Risk Assessment	TCP 8080	Source refers to the Illumio Segmentation Template. The source should be the role label of the node(s) that nest the OpenShift Router.

Example Ruleset 2

The following Ruleset is from the Segmentation Template and you can modify it as needed.

Scope

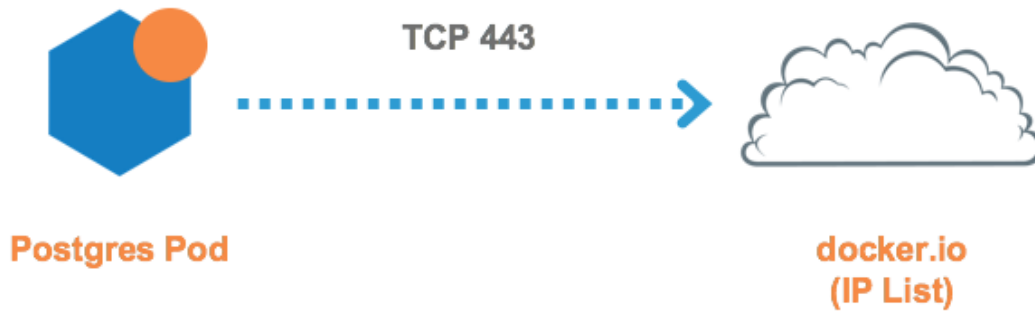
Application	Environment	Location	Notes
IST OpenShift Infrastructure	IST Production	IST HQ	Ruleset is derived from Illumio Segmentation Template. The scope should match the OpenShift cluster.

Intra-Scope Rule

Source	Destination	Destination Service	Notes
External Network	IST Infra (Role)	TCP 80	This rule is included in Illumio Segmentation Template. The destination should be the OpenShift cluster node(s) that nest the router. The source can be an IP list such as 0.0.0.0/0 (any), HQ, Corporate, or employee subnet. The IST default includes 0.0.0.0/0 (any) IP list.

Outbound Connections

The outbound connections are required to access repositories.



Kubernetes and OpenShift

The rules you need to write are:

Example Ruleset

Scope

Application	Environment	Location
Risk Assessment	Development	Cloud

Intra-Scope Rule

Source	Destination	Destination Service	Notes
All Workloads	docker.io (IP List)	All Services	Containerized environments depend on various external resources to perform basic operations such as pulling a docker image. In this rule, All pods in the Risk Assessment application are allowed to pull the image from the docker.io repository..

Liveness Probes

Containerized applications may require periodic health checks known as liveness probes and readiness probes. Each application includes a health check YAML file that contains configurations for liveness and readiness probes. The health checks between the container node and the local container workload may rely on TCP ports. Illumio has included a source object called Container

Host for this use case. The Container Host object represents the container node or nodes that host the Pod(s). The example below uses the Container Host object as a source for Liveness and Readiness Probes.

**NOTE**

The Container Host must always fall under an Extra-Scope rule.

The rules you need to write are:



Kubernetes and OpenShift

The rules you need to write are:

Example Ruleset

Scope

Application	Environment	Location
Risk Assessment	Development	Cloud

Extra-Scope Rule

Source	Destination	Destination Service	Notes
Container Host (built-in Illumio object)	All Workloads	TCP 9090	In this example, the Risk Assessment health check configuration indicates that the liveness probe occurs on TCP 9090. Liveness probe ports/protocols may vary across applications. Container Host is an object built into the PCE by default and represents any node that hosts the respective Pod(s).

NodePort Support on Kubernetes and OpenShift

Kubernetes and OpenShift provide a mechanism to access cluster services from the outside world, of type NodePort. This service exposes a port on all nodes in the cluster on which traffic will be forwarded to any of the backing pods that match the service's selector.

Scenario:

- The Kubernetes cluster and containerized applications are in the Production environment.
- The containerized application is called RiskAssessment, and each Pod within the application listens on TCP 8080.
- The RiskAssessment application is exposed to the outside world via a FrontEnd service with type NodePort.
- The exact NodePort in use is not specified, but is automatically allocated by Kubernetes.
- There may be clients to the FrontEnd service within the cluster or outside the cluster - in both cases, they are labeled as Client.

The rules you need to write are:

Example Ruleset 1: Internal and External Access to Service

Scope

Application	Environment	Location
Risk Assessment	Production	Cloud

Extra-Scope Rule

Source	Destination	Destination Service	Notes
Client (Role label for Web pods and external workloads)	FrontEnd (Virtual Service Role label for Risk Assessment service) + Use Virtual Services Only	Derived from Destination Virtual Service	After the Risk Assessment service gets discovered by the PCE it becomes a virtual service object in the PCE. The destination here should be the role label of the virtual service plus the "Use Virtual Service Only" option.

Rules for Persistent Storage

This section only applies to deployments which require communication with external storage nodes over NFS, iSCSI, and others for persistent storage. If the cluster or Pods have external storage dependencies, then you need a policy to allow outbound communications to the storage node. The storage node can be represented as an unmanaged workload or IP list.

The following is an example of outbound policy to a NFS node, which is represented by an IP list.

Kubernetes

The following is an example of an outbound policy to an NFS node, which is represented by an IP list:

Example Ruleset 1

Scope

Application	Environment	Location	Notes
Kubernetes Infrastructure	Development	Cloud	Kubernetes cluster

Intra-Scope Rule

Source	Source Service	Destination	Notes
NFS Storage (IP List)	TCP 2049	All Workloads	All Kubernetes nodes and infrastructure Pods can communicate outbound to NFS over the NFS TCP port.

Example Ruleset 2

Scope

Application	Environment	Location	Notes
ERP	Development	Cloud	From httpd example

Intra-Scope Rule

Source	Source Service	Destination	Notes
NFS Storage (IP List)	TCP 2049	All Workloads	All Pods can talk outbound to NFS over the NFS TCP port.

OpenShift

The following is an example of an outbound policy to an NFS node, which is represented by an IP list:

Example Ruleset 1

Scope

Application	Environment	Location	Notes
OpenShift Infrastructure	Development	Cloud	OpenShift cluster

Intra-Scope Rule

Source	Source Service	Destination	Notes
NFS Storage (IP List)	TCP 2049	All Workloads	All OpenShift nodes and infrastructure Pods can communicate outbound to NFS over the NFS TCP port.

Example Ruleset 2

Scope

Application	Environment	Location	Notes
ERP	Development	Cloud	From httpd example

Intra-Scope Rule

Source	Source Service	Destination	Notes
NFS Storage (IP List)	TCP 2049	All Workloads	All Pods can talk outbound to NFS over the NFS TCP port.

Local Policy Convergence Controller

The local policy convergence controller provides a deterministic way of setting the readiness state of pods in your cluster after local policy has converged. By controlling the readiness state of pods, you can prevent them from receiving and sending traffic through Kubernetes until they are ready. Using a controller ensures that the network and security infrastructure is ready for a multi-microservice application.

In this release, the Kubernetes Custom Pod Conditions feature introduced in v1.14 is available for containerized VENS.

About the Controller Behavior

By default, the readiness gate is not specified on a pod spec and the C-VEN does not affect the readiness state of the pod regardless of annotations or Illumio managed state.

When the Illumio readiness gate is specified on a pod spec, the PCE completes the following actions when a new pod is created:

1. Sends the C-VEN policy for the new pod P.
2. When pod P is managed, the C-VEN applies local policy for the new pod P.
3. The C-VEN waits for a timer to expire to allow peers to apply policy on their end (such as, updating the new pod P IP address).

By default, the timer uses the following values:

- If the pod is managed by Illumio, the timer is set to 15 seconds.
- If the pod is not managed by Illumio, the timer is set to 0 seconds.



TIP

To configure a custom value for the timer duration, see [Timer Customization \[274\]](#).

4. The C-VEN sets the readiness gate pod condition to “True.”
The pod is now considered “Ready” by Kubernetes.

Configure the Illumio Readiness Gate

To use a local policy convergence controller, specify the Illumio readiness gate under `readinessGates.conditionType` in the pod spec YAML.

See the following example pod spec YAML file:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deploy
spec:
  selector:
    matchLabels:
      app: my-pod
  replicas: 2
  template:
    metadata:
      labels:
        app: my-pod
    spec:
      readinessGates:
# <----- declare readiness gates
  - conditionType: "com.illumio.policy-ready"
# <----- Illumio policy convergence readiness gate
    containers:
      - name: my-pod-web
        image: nginx
        ports:
          - containerPort: 80

```

Timer Customization

You can customize the timer cluster-wide or pre-pod.



NOTE

When configuring a custom timer by using the DaemonSet environment variable or an annotation, you are limited to specifying 0-300 seconds.

Cluster Wide Timer Customization

To customize the timer duration on a cluster-wide basis, set the readiness gate timer variable in the C-VEN DaemonSet YAML.

See the following YAML file:


```

...
containers:
  - name: illumio-ven
    env:
      - name: ILO_SERVER
        valueFrom:
          secretKeyRef:
            name: illumio-ven-config
            key: ilo_server
      - name: ILO_CODE
        valueFrom:
          secretKeyRef:
            name: illumio-ven-config
            key: ilo_code
      - name: ILO_K8S_NODE_NAME
        valueFrom:
          fieldRef:
            fieldPath: spec.nodeName
      - name: ILO_K8S_READINESS_TIMER
# <--- custom readiness gate timer across the cluster
  value: "20"
# <--- timer value
...

```

Pre-pod Timer Customization

To customize the timer duration for specific pods, set the Illumio readiness gate timer annotation on the pod spec.

See the following example deployment:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deploy
spec:
  selector:
    matchLabels:
      app: my-pod
  replicas: 2
  template:
    metadata:
      labels:
        app: my-pod
      annotations:
        com.illumio.readiness-gate-timer: "20"
# <----- custom readiness gate timer for all pods in this
deployment
  spec:
    readinessGates:
      - conditionType: "com.illumio.policy-ready"
    containers:
      - name: my-pod-web
        image: nginx
        ports:
          - containerPort: 80

```

Track the State of the Readiness Gate

You can track the state of the readiness gate by running either of the following commands:

- `kubectl get pod -o wide`
- `kubectl get ep -o wide`

Example: State of the Readiness Gate

This example shows a cluster with Kubelink and the C-VEN deployed and running. When you initially deploy or scaled up the Illumio Readiness Gate, you see the following values:



NOTE

The state of gate readiness appears in the "READINESS GATES" column.

```
$ kubectl get pod,ep -o wide
```

NAME			READY	STATUS	RESTARTS
AGE	IP	NODE	NOMINATED	NODE	READINESS
GATES					
pod/my-deploy-855dfbf94f-gwz7c			1/1	Running	1
4d20h	172.17.0.7	ubuntu20	<none>		0/1
pod/my-deploy-855dfbf94f-p7czp			1/1	Running	1
4d20h	172.17.0.6	ubuntu20	<none>		0/1
NAME		ENDPOINTS			AGE
endpoints/kubernetes		10.0.2.15:8443			19d
endpoints/my-service					4d22h

In this example, the readiness gates are marked as 0/1 for both pods and my-service does not have any available endpoints. After the VEN has processed the policy for the new pods and the timer expires, it sets the readiness gate to “True” for each pod and you see the following output:

```
$ kubectl get pod,ep -o wide
```

NAME			READY	STATUS	RESTARTS
AGE	IP	NODE	NOMINATED	NODE	READINESS
GATES					
pod/my-deploy-855dfbf94f-gwz7c			1/1	Running	1
4d20h	172.17.0.7	ubuntu20	<none>		1/1
pod/my-deploy-855dfbf94f-p7czp			1/1	Running	1
4d20h	172.17.0.6	ubuntu20	<none>		1/1
NAME		ENDPOINTS			AGE
endpoints/kubernetes		10.0.2.15:8443			19d
endpoints/my-service		172.17.0.6:9376,172.17.0.7:9376			4d22h

To view greater detail about the pod conditions, run the command `kubectl get pod <pod name> -o yaml`:

```
$ kubectl get pod my-deploy-855dfbf94f-gwz7c -o yaml
...
status:
  conditions:
    - lastProbeTime: null                // <--
      lastTransitionTime: "2021-05-18T20:26:26Z" // <--
      message: Pod Policy Ready              // <-- this
pod condition is set by VEN
      reason: PolicyReady                  // <--
      status: "True"                      // <--
      type: illumio.com/policy-ready       // <--
    - lastProbeTime: null
      lastTransitionTime: "2021-05-18T20:25:51Z"
      status: "True"
      type: Initialized
    - lastProbeTime: null
      lastTransitionTime: "2021-05-19T19:56:24Z"
      status: "True"
      type: Ready
// <-- this is only set to True after all readiness gates are
set to True
    - lastProbeTime: null
      lastTransitionTime: "2021-05-19T19:56:24Z"
      status: "True"
      type: ContainersReady
    - lastProbeTime: null
      lastTransitionTime: "2021-05-18T20:25:51Z"
      status: "True"
      type: PodScheduled
  ...
```

Firewall Coexistence on Pods

The Illumio C-VEN configures iptables on each host and each Pod (in a managed namespace). By default, Illumio Core coexistence mode is set to **Exclusive** meaning the C-VEN will take full control of iptables and flush any rules or chains that are not created by Illumio Core. In containerized environments, this may affect communications to/from container components (Docker, Kubernetes, Illumio Kubelink). Therefore, Illumio Core must allow firewall coexistence in order to achieve non-disruptive installation and deployment.

**NOTE**

For Workloads part of a Container cluster (Kubernetes or OpenShift nodes), firewall coexistence is enabled by default if Kubelink was deployed and is "In Sync" with the PCE (prior to the C-VEN installation).

In some cases, there may be some Pods that implement iptables rules inside the Pod namespace for the containerized application to work (VPN, NAT, and others). In order to support such requirements from containerized applications, you should enable firewall coexistence for these Pods.

In order to allow firewall coexistence, you must set a scope of Illumio labels in the firewall coexistence configuration. Once you provision a firewall coexistence scope, the PCE will enable firewall coexistence configuration on all the Pods whose labels fall within the scope.

**NOTE**

Labels assigned to Kubernetes cluster nodes must fall within the firewall coexistence scope. This is not a requirement for the labels assigned to container workloads.

To configure firewall coexistence:

1. In the PCE UI, navigate to **Settings > Security**.
2. On the Security page, select the **Manage Firewall Coexistence** tab.
3. Click **Edit**.
4. In the edit wizard, click **Add**. The **Add Firewall Coexistence Labels and Policy State** wizard will pop-up.
5. Select a scope of Illumio labels. The scope must include the labels you intend to use for your Kubernetes cluster nodes.
 - a. Select **All** for Policy State.
 - b. Illumio Core is Primary Firewall - Select your preference.
 - i. **Yes** = (Recommended) Illumio iptable chains will be at the top of iptables at all times. Non-Illumio iptable chains can coexist, but will follow after Illumio chains.
 - ii. **No** = (Not Recommended) Non-Illumio iptable chains may coexist and can be placed before Illumio chains.

**NOTE**

For deployments using Calico, Illumio recommends setting the Calico ChainInsertMode to **Append** and set Illumio Core as Primary Firewall value to **Yes**. If the Kubernetes cluster requires Calico Insert mode, then set Illumio Core as Primary Firewall value to **No**.

c. Click **OK**.

6. Click **Save**.

7. Provision the changes.

Be sure to provision the saved changes or else firewall coexistence will not take effect.

The following example is of a firewall coexistence scope for a Kubernetes or OpenShift cluster which has the following labels:

- Role: All
- Application: Kube-System
- Environment: Development
- Location: Cloud

The firewall coexistence scope in the example uses the 'All Roles' objects to cover future Pods spun up in the kube-system namespace that may require additional iptables rules to forward packets.

Role	Application	Environment	Location	Policy State	Illumio ASP is Primary Firewall
All Roles	Kube-System	Development	Cloud	All	Yes

Upgrade and Uninstallation

Follow the steps to upgrade or uninstall Kubernetes components. This section also describes the procedure for migrating from a deployment of C-VEN version 21.5.15 or earlier (which did not use Helm Charts) to a current Helm Chart deployment.

Learn how to upgrade a non-CLAS deployment to a CLAS-enabled one (which is the default mode starting in version 5.0.0 of Kubernetes).



IMPORTANT

Use the upgrade and uninstallation procedures according to the method that was first used to deploy the product. For deployments made with a Helm Chart (typically with Kubernetes 3.0.0 or later), follow the steps in [Upgrade and Uninstall Helm Chart Deployments \[285\]](#).

For deployments made without using a Helm Chart (for installations of C-VEN 21.5.15 or earlier), follow the steps in [Upgrade and Uninstall Non-Helm Chart Deployments. \[287\]](#)

Migrate from Previous C-VEN Versions (21.5.15 or Earlier)

This section outlines the steps to migrate a manually deployed Illumio installation to a Helm-managed deployment. Manually deployed (or non-Helm) deployments were used to configure and deploy C-VEN versions 21.5.15 and earlier, as well as Kubelink versions prior to 3.0.

To upgrade an existing Helm installation to a newer version, follow standard Helm practice with **helm upgrade** command.

Follow these general steps to migrate from a manually-deployed Illumio Core for Kubernetes to a Helm Chart deployment:

1. Annotate and label resources.
2. Delete C-VEN DaemonSet.
3. Install Helm and the Helm Chart.

Annotate and Label Resources

From Helm version 3.0.0 on, Helm supports adopting already-deployed resources with the correct name, annotations, and labels.

Required annotations and labels are:

```
annotations:  
  meta.helm.sh/release-name: illumio  
  meta.helm.sh/release-namespace: illumio-system  
labels:  
  app.kubernetes.io/managed-by: Helm
```

To annotate and label all Illumio resources, use the commands below (provided the names of resources match your deployment). Note the **--overwrite** flag which replaces any existing ownership annotations that might be already assigned.


```

kubectl -n illumio-system annotate secret illumio-ven-config
meta.helm.sh/
release-name=illumio --overwrite
kubectl -n illumio-system annotate secret illumio-ven-config
meta.helm.sh/
release-namespace=illumio-system --overwrite
kubectl -n illumio-system label secret illumio-ven-config
app.kubernetes.io/
managed-by=Helm --overwrite
kubectl -n illumio-system annotate secret illumio-kubelink-
config meta.helm.sh/
release-name=illumio --overwrite
kubectl -n illumio-system annotate secret illumio-kubelink-
config meta.helm.sh/
release-namespace=illumio-system --overwrite
kubectl -n illumio-system label secret illumio-kubelink-config
app.kubernetes.io/
managed-by=Helm --overwrite
kubectl -n illumio-system annotate serviceaccount illumio-ven
meta.helm.sh/
release-name=illumio --overwrite
kubectl -n illumio-system annotate serviceaccount illumio-ven
meta.helm.sh/
release-namespace=illumio-system --overwrite
kubectl -n illumio-system label serviceaccount illumio-ven
app.kubernetes.io/
managed-by=Helm --overwrite
kubectl -n illumio-system annotate clusterrole illumio-
kubelink meta.helm.sh/
release-name=illumio --overwrite
kubectl -n illumio-system annotate clusterrole illumio-
kubelink meta.helm.sh/
release-namespace=illumio-system --overwrite
kubectl -n illumio-system label clusterrole illumio-kubelink
app.kubernetes.io/
managed-by=Helm --overwrite
kubectl -n illumio-system annotate clusterrolebinding illumio-
ven meta.helm.sh/
release-name=illumio --overwrite
kubectl -n illumio-system annotate clusterrolebinding illumio-
ven meta.helm.sh/
release-namespace=illumio-system --overwrite
kubectl -n illumio-system label clusterrolebinding illumio-ven
app.kubernetes.io/
managed-by=Helm --overwrite
kubectl -n illumio-system annotate clusterrole illumio-ven

```

```

meta.helm.sh/
release-name=illumio --overwrite
kubectl -n illumio-system annotate clusterrole illumio-ven
meta.helm.sh/
release-namespace=illumio-system --overwrite
kubectl -n illumio-system label clusterrole illumio-ven
app.kubernetes.io/
managed-by=Helm --overwrite
kubectl -n illumio-system annotate serviceaccount illumio-
kubelink meta.helm.sh/
release-name=illumio --overwrite
kubectl -n illumio-system annotate serviceaccount illumio-
kubelink meta.helm.sh/
release-namespace=illumio-system --overwrite
kubectl -n illumio-system label serviceaccount illumio-
kubelink app.kubernetes.io/
managed-by=Helm --overwrite
kubectl -n illumio-system annotate deployment illumio-kubelink
meta.helm.sh/
release-name=illumio --overwrite
kubectl -n illumio-system annotate deployment illumio-kubelink
meta.helm.sh/
release-namespace=illumio-system --overwrite
kubectl -n illumio-system label deployment illumio-kubelink
app.kubernetes.io/
managed-by=Helm --overwrite
kubectl -n illumio-system annotate clusterrolebinding
illumio-kubelink meta.helm.sh/release-name=illumio --overwrite
kubectl -n illumio-system annotate clusterrolebinding
illumio-kubelink meta.helm.sh/release-namespace=illumio-system
--overwrite
kubectl -n illumio-system label clusterrolebinding
illumio-kubelink app.kubernetes.io/managed-by=Helm --overwrite

```

The output should look similar to this:

```

...
clusterrolebinding.rbac.authorization.k8s.io/illumio-kubelink
annotated
clusterrolebinding.rbac.authorization.k8s.io/illumio-kubelink
annotated
clusterrolebinding.rbac.authorization.k8s.io/illumio-kubelink
labeled

```

Delete C-VEN DaemonSet

The next step is removing the C-VEN DaemonSet. Save any custom labels and validations included in the DaemonSet and reapply them later.

```
kubectl delete daemonset illumio-ven -n illumio-system
```

Install Helm

The final step is to install Helm and the Helm Chart for Illumio Core on Kubernetes. Follow the steps in [Deploy with Helm Chart \[207\]](#). Filling in the fields in **illumio-values.yaml** is still mandatory.

Upgrade and Uninstall Helm Chart Deployments

Deployments of Illumio Core for Kubernetes 3.0.0 or later are performed with Helm Charts. Upgrades and uninstallations are also performed with Helm commands.

Upgrade Helm Chart Deployments

To upgrade an existing installation to a newer version after it had been initially deployed with a Helm Chart, follow standard Helm practice with the **helm upgrade** command.

For example, if you install the Helm Chart for Core for Kubernetes 4.2.0 initially with this command:

```
helm install illumio -f values.yaml oci://quay.io/illumio/illumio --version 4.2.0 --namespace illumio-system
```

Then use the following command to upgrade to version 4.3.0:

```
helm upgrade illumio -f values.yaml oci://quay.io/illumio/illumio --version 4.3.0
```

Use the same **values.yaml** file for the upgrade that was used for the original install command.

**IMPORTANT**

Be sure to explicitly specify the version to upgrade to with the `--version <ver#>` option (for example, `--version 4.3.0`), after confirming that the product version you want to install is supported with your PCE version. Verify which PCE versions support the Illumio Core for Kubernetes version you want to deploy at the [Kubernetes Operator OS Support and Dependencies](#) page on the Illumio Support Portal.

Uninstall Helm Chart Deployments

To completely uninstall an existing installation that had been initially deployed with a Helm Chart:

```
$ helm uninstall illumio --namespace illumio-system
$ kubectl delete namespace illumio-system
```

The uninstallation process also unpairs the C-VEs from the PCE.

Uninstalling the Helm Chart takes around two minutes to complete.

Unpair C-VEs from PCE before Helm Chart uninstallation for large clusters

For large clusters, it is recommended to initiate a c-ven unpair prior to a helm uninstall.

clusterMode unpair C-VEs from PCE before Helm Chart uninstallation

For Helm Chart version 5.4.0 and higher, you can change the cluster mode to unpair with `clusterMode:unpair` within the `values.yaml` file and perform `helm upgrade` or `ArgoCD sync` or the equivalent.

Updating daemonset to unpair C-VEs from PCE before Helm Chart uninstallation

For large clusters that would take longer than two minutes to uninstall Helm Chart, you can use a command like the following to first unpair with `kubectl`.

**CAUTION**

If you run this command, verify your command uses straight single ' and double " quotes, not curly quotes: “ , ” , ‘ , ’ . Copying and pasting into some text editors or word processing programs automatically converts straight quotes to curly quotes.

```
kubectl patch daemonset illumio-ven -n illumio-system
--type=json -p='[{"op":"replace","path":"/spec/template/spec/containers/0/command","value":["/ven-init","unpair"]}]'
```

Once the command completes, run the helm uninstall and kubectl delete commands.

Upgrade and Uninstall Non-Helm Chart Deployments

This section describes how deployments that were not installed with Helm can be upgraded or uninstalled.

Upgrade Illumio Components

Illumio Core for Kubernetes and OpenShift is a flexible and modular solution that can be upgraded piece by piece.

For minor upgrades, Kubelink can be upgraded independently from the C-VEN and vice versa unless explicitly mentioned in the release notes.

For major upgrades, including PCE, Kubelink, and C-VEN, Illumio recommends the following process:

- Upgrade the PCE to the new desired version.
- Review the compatibility matrix between PCE, Kubelink, and C-VEN on the Illumio support website.
- Upgrade Kubelink.
- Upgrade C-VEN.

Upgrade Kubelink

The supported process to upgrade Kubelink is as follows:

1. Upload the new image to your private container registry.
2. Change the manifest file to point to the latest Kubelink image in the registry. You do not need to change the previously created secret for Kubelink.
3. Apply this new manifest file to the cluster. **illumio-kubelink** follows the default update behavior of Kubernetes. For more information, see [Kubernetes Documentation](#).

You can verify that the upgrade was successful in the PCE UI on the **Container Clusters > Summary** page and checking for the new Kubelink version.

Upgrade C-VEN

The supported process to upgrade C-VEs is as follows:

1. Upload the new image to your private container registry.
2. Change the manifest file to point to the latest C-VEN image in the registry. You do not need to change the previously created secret for C-VEN.
3. Apply this new manifest file to the cluster. **illumio-ven daemonset** follows the default rolling update behavior of Kubernetes. For more information, see [Kubernetes Documentation](#).

You can verify that the upgrade was successful in the PCE UI on the **Container Clusters > Workloads** page and clicking on any workload and checking for the new C-VEN version.

Uninstall Illumio from Your Cluster

To uninstall the Illumio components, you need to contact Illumio Professional Services to unpair the C-VEs and then delete the Illumio resources from your cluster.

Unpair C-VEs



IMPORTANT

Contact Illumio Professional Services to unpair the C-VEs in your Kubernetes or OpenShift clusters.

Deleting C-VEs or DaemonSet will not properly unpair them from the PCE and can cause the following issues:

- Workloads will go offline in the PCE UI after 5 minutes (defined by the default Offline Timers configured in the PCE).
- Workloads will be left in the PCE UI as offline with the button to unpair them grayed out (this action is not supported by Illumio).
- Firewall rules configured on the Host and Pods namespaces will remain untouched and active.

The current way to properly delete these workloads created in the PCE UI by C-VEs is by deleting the entire cluster in the PCE UI.



IMPORTANT

Unpairing an individual C-VE is not supported. It has to be done at the cluster level (through the DaemonSet), because the cluster is considered as a single entity from a security point of view.

If a node unjoins the cluster for any reason or due to the `kubectl delete node <node_name>` command, the PCE automatically unpairs the C-VE and deletes the workload and Container workloads associated with the C-VE that was running on the deleted node.

Delete Illumio Resources

To delete the existing Illumio resources created in your Kubernetes or OpenShift cluster, follow these steps:

Delete C-VE Resources

1. Contact Illumio Professional Services to unpair the C-VEs and clean up existing iptables or nftables rules created by Illumio.
2. Check the Workloads and Container Workloads tabs under **Infrastructure > Container Clusters > YourClusterName** and validate that your nodes and Pods are no longer visible.
3. Delete the resources created during the C-VE installation by using the following command:

```
kubectl delete -f illumio-ven-kubernetes.yml
kubectl delete -f illumio-ven-secret.yml
```

```
oc delete -f illumio-ven-openshift.yml
oc delete -f illumio-ven-secret.yml
```

Delete Kubelink Resources

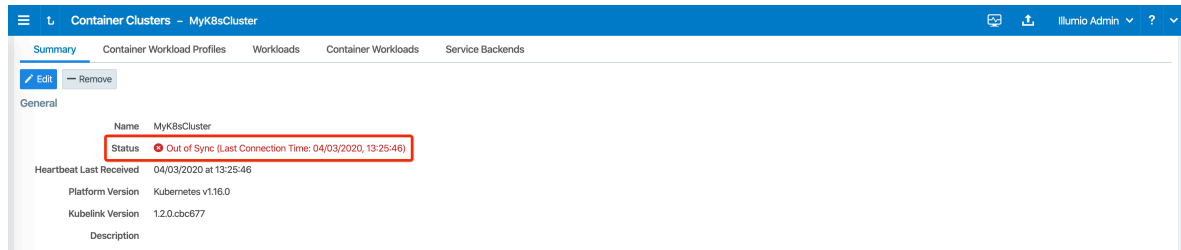
1. Delete the resources created during the Kubelink installation.
2. Delete Kubelink resources from Kubernetes:

```
kubectl delete -f illumio-kubelink-kubernetes.yml
kubectl delete -f illumio-kubelink-secret.yml
```

3. Delete Kubelink resources from OpenShift:

```
oc delete -f illumio-kubelink-openshift.yml
oc delete -f illumio-kubelink-secret.yml
```

4. Check the Summary tab under **Infrastructure > Container Clusters > YourClusterName** and validate that your cluster is "Out of Sync". It takes approximately 10 minutes for the cluster Status to change from "In Sync" to "Out-of-Sync".



5. Finally, delete the container cluster from the PCE UI and verify that there are no resources left in your cluster such as, ConfigMap, Secrets, and others.

Delete Illumio Namespace

- To delete the Illumio namespace in Kubernetes, use the following command:

```
kubectl delete ns illumio-system
```

- To delete the Illumio namespace in OpenShift, use the following command:

```
oc delete project illumio-system
```

Upgrade to CLAS Architecture

A Cluster Local Actor Store (CLAS) mode is introduced into the architecture of Illumio Core for Kubernetes 5.0.0.



IMPORTANT

To use CLAS, your PCE must be upgraded to Core 23.5.10 or later.

The CLAS architecture brings two major changes to the typical Illumio Core policy model:

- The definition of a workload is fundamentally changed. In a legacy, non-CLAS environment, a container workload is a Pod. In CLAS, a workload is now the Kubernetes Workload resource (such as Deployment, StatefulSet, ReplicaSet, DaemonSet, and so on), which typically includes multiple Pods that can change in amount during the lifetime of the workload. As such, CLAS workloads are called *Kubernetes Workloads*, to distinguish them from non-CLAS *Container Workloads*.
- ClusterIP services change from Virtual Services to workloads. NodePort and LoadBalancer services remain as Virtual Services in the PCE. The ClusterIP part of a NodePort or LoadBalancer service also exists as a Kubernetes Workload and is linked with the Virtual Service.

Illumio recommends writing a policy using labels. In addition to being impractical, it is not even possible to write a policy for individual Pods. It was (and still is) possible to use Virtual Services in the policy explicitly in rule writing, but Illumio nevertheless recommends using labels.



IMPORTANT

The CLAS architecture is supported only in Illumio Core for Kubernetes versions 5.0.0 and later

Pre-upgrade Policy Check

All policies for your Kubernetes environments must be expressed using labels. In the rare case that policies are using Virtual Service objects, those policies must be changed to label-based policies.

ClusterIP Services as Kubernetes Workloads

ClusterIP services are modeled as workloads in the CLAS environment. If you had a policy written with "Virtual Services Only," that policy will not apply to Kubernetes Workloads (including ClusterIP Services) after the upgrade to CLAS. All rules that apply to ClusterIP Services must be changed to "Use Workloads" before upgrading to CLAS, which needs Destination Services to be specified. This setting also causes ports to be populated from Virtual Services to the rule. So, at least one port number must be filled in when writing this rule.

To keep the old functionality of PCE synchronizing the ports of the ClusterIP Service, the CLAS now performs this operation. When the rule arrives at

Kubelink/CLAS, ports will be replaced by the current ports of the ClusterIP Services. The port replacement includes all ports from the Service. If the Service has two ports, it is not possible to include one without including the other.

Because Services are now Kubernetes Workloads, the "All Workloads" flag in a rule will include all Services. Do not use "All Workloads" as a Destination in a rule. Use a more specific label instead that targets the Service.

All rules that include a label of at least one ClusterIP service will have specified ports internally replaced. However, this is not reflected in PCE UI, where the rule still displays ports.

Upgrade Strategy

Illumio Core for Kubernetes 5.x is backward-compatible and supports both CLAS and legacy non-CLAS mode operation.

This is controlled by the **clusterMode** parameter specified in the Helm Chart installation yaml file. The default value is **legacy**, meaning that after the upgrade, the software operates in the legacy, non-CLAS mode.

The PCE supporting Illumio Core for Kubernetes 5.0.0 and later (PCE version 23.5.0+A1 and later) also supports both CLAS and non-CLAS modes of operation. Illumio recommends that, after the software upgrade, the migration to CLAS be performed one cluster at a time.

The CLAS implementation uses the configuration parameter **clusterMode** set to **clas** or **legacy** to turn on (or off) CLAS mode in the cluster, respectively, when installing. When upgrading an existing non-CLAS cluster to CLAS, set **clusterMode**, to **migrateLegacyToClas**. When reverting (or downgrading) CLAS to non-CLAS, set **clusterMode** to **migrateClasToLegacy**.

Upgrade Steps (on Each Kubernetes Cluster)

Be sure to perform all steps in this procedure on each existing Kubernetes cluster that you want to upgrade to CLAS mode.

1. Prepare the **values.yaml** file with all required parameters. Refer to [Deploy with Helm Chart \[207\]](#).
2. Upgrade Illumio Core for Kubernetes to version 5.1.0 or later. Refer to [Upgrade and Uninstall Helm Chart Deployments \[285\]](#).

3. Verify the upgrade was successful.
4. Perform a pre-upgrade policy check (see [Pre-upgrade Policy Check \[291\]](#) above).
5. Set the "illumio-system" namespace to Visibility Only enforcement mode.
6. Consider setting all cluster nodes into Visibility Only enforcement mode. This step compromises security and will open traffic to/from your protected applications. On the other hand, any policy errors will not result in an application outage.
7. Migrate the cluster to CLAS mode:
 - a. Add `clusterMode: migrateLegacyToClas` parameter-value pair to your `values.yaml`.
 - b. Perform `helm upgrade` command:

```
helm upgrade <nameofyourhelmdployment> -n illumio-system
-f <yourvalues.yaml> oci://quay.io/illumio/illumio --
version 5.1.0
```



IMPORTANT

Be sure to explicitly specify the version to upgrade to with the `--version <ver#>` option (for example, `--version 5.1.0`), after confirming that the product version you want to upgrade to is supported with your PCE version. Verify which PCE versions support the Illumio Core for Kubernetes version you want to deploy at the [Kubernetes Operator OS Support and Dependencies](#) page on the Illumio Support Portal.

8. Refresh the Container Cluster page so that the Kubernetes Workloads tab now appears along with the Container Workloads tab.
9. Check that all C-VEN pods and the Kubelink pod have restarted.
10. This cluster is now running in migration mode, Container Workloads are still present, and new Kubernetes Workloads (CLAS-enabled) are populated.
11. Check if the policy sync status of all Kubernetes Workloads and Peer Workloads is "Active." Some container workloads may be in an Active state, while others are Syncing — this is expected. Check if traffic still works. If something goes wrong, revert the cluster to non-CLAS mode with the following procedure, otherwise, go to the next Step:
 - a. Specify `clusterMode: migrateClasToLegacy` parameter-value pair in your `values.yaml`.
 - b. Perform `helm upgrade` command:

```
helm upgrade <nameofyourhelmdeployment> -n illumio-system
-f <yourvalues.yaml> oci://quay.io/illumio/illumio --
version 5.1.0
```

- c. Refresh the Container Cluster page so the Container Workloads tab appears along with the Kubernetes Workloads tab
- d. Wait until Container Workloads and peers are Active, and traffic is working as expected.
- e. Change the **clusterMode** parameter to **legacy** in **values.yaml** -- or delete the variable (because the default parameter value is **legacy**).
- f. Perform the **helm upgrade** command:

```
helm upgrade <nameofyourhelmdeployment> -n illumio-system
-f <yourvalues.yaml> oci://quay.io/illumio/illumio --
version 5.1.0
```

- g. Verify that Kubernetes Workloads were deleted, that Container Workloads are in a Synced state, and that traffic is working as expected.
12. Set the cluster to CLAS mode:
 - a. Change **clusterMode: clas** in **values.yaml**.
 - b. Perform the **helm upgrade** command:

```
helm upgrade <nameofyourhelmdeployment> -n illumio-system
-f <yourvalues.yaml> oci://quay.io/illumio/illumio --
version 5.1.0
```

13. Check that the Kubelink Pod restarted.
14. The cluster is now running in the CLAS mode. All Container Workloads from this cluster will no longer be visible on the PCE. Instead, the PCE will display only a list of Kubernetes Workloads (Deployments, etc.).
15. Set all nodes into original enforcement mode if they were previously changed to visibility only.



IMPORTANT

Before using your CLAS cluster, ensure that you have written mandatory infrastructure rules to enable proper operation. See [Rules and Traffic Considerations with CLAS \[257\]](#) for details on these mandatory rules.

Reference: General

This section lists a few known limitation of this release and how to troubleshoot issues that may occur during the installation process.

For more information see these additional topics.

- Troubleshooting
- Troubleshooting CLAS Mode Architecture
- Known Limitations
- Kubelink Monitoring and Troubleshooting
- Aggregating Logs from Kubelink and C-VEN Pods

Troubleshooting

Troubleshoot common issues when installing Illumio on Kubernetes or OpenShift deployments.

Kubelink Support Bundle

To assist the Illumio Support team with more details for troubleshooting, Kubelink now provides a support bundle that collects up to 2 GB of logs, metrics, and other data inside its pod. Future versions will add the option to upload these support bundles to the PCE. Currently, you must copy this support bundle by running the script `/support_bundle.sh` inside the Kubelink pod. The script generates debug data, creates a gzipped tar archive using `stdout` as output, and encodes this data using Base64.

Use the following command to generate and transfer the Kubelink support bundle from its pod:

```
kubectl --namespace illumio-system exec deploy/illumio-kubelink -- /support_bundle.sh | base64 --decode > /tmp/kubelink_support.tgz
```

Send the resulting compressed archive file to Illumio Support when requested.

Helm deployment (and uninstall) fails with C-VEN stuck in ContainerCreating state

During a deployment with Helm, if C-VEN pods do not start, and instead continually show a status of **ContainerCreating**, check that you have the correct runtime set in your `illumio-values.yaml` file. If, for example, the `containerRuntime` value is set to `containerd` but you are now using a Docker runtime (parameter value of `docker`), then the C-VEN will become stuck in a **ContainerCreating** state. If you later attempt to uninstall, the

unpair action for the C-VEN will also become stuck in a **ContainerCreating** state.

Confirm that a C-VEN exhibiting these persistent **ContainerCreating** symptoms is set to the proper **containerRuntime** value in its **illumio-values.yaml**. Another clue when troubleshooting is to check output of the **describe** command for the affected pod:

```
kubectl -n illumio-system describe pod/<your_pod_name>
```

Check the output under the **Containers** section, and, within that section, under the **Mounts** section, to confirm the pod is attempting to mount to a location appropriate for your container runtime.

```
# kubectl -n illumio-system describe pod/illumio-ven-unpair-
cwj2f
Name:          illumio-ven-unpair-cwj2f
Namespace:     illumio-system

[. . .]

Mounts:
  /var/run/containerd/containerd.sock from unixsocks (rw)
```

This problem is also shown under the **Events** section of this output, with a **Warning** event for that mount location due to the mismatched container runtime values.

```
Events:
  Type           Reason          Age           From
Message
  ----           -
[. . .]
Warning FailedMount 96s (x11 over 7m48s) kubelet
MountVolume.SetUp failed for volume "unixsocks" : hostPath
type check failed: /var/run/containerd/containerd.sock is not
a socket file
```

Also check for any other mistakes in the **illumio_values** file. For example, use the following **kubectl get nodes -o wide** commands to verify the proper OS node versions, Kubernetes/Openshift version, and the like:

```

root@Master:~# kubectl get nodes -o wide
NAME          STATUS  ROLES          AGE
VERSION      INTERNAL-IP  EXTERNAL-IP  OS-IMAGE  KERNEL-
VERSION                                CONTAINER-RUNTIME
master    Ready    control-plane  100d    v1.29.9
10.2.85.63    <none>      CentOS      Linux 8
4.18.0-348.7.1.el8_5.x86_64    containerd://1.6.32
node0      Ready    <none>          100d    v1.29.9
10.2.85.65    <none>      CentOS      Linux 8
4.18.0-348.7.1.el8_5.x86_64    containerd://1.6.32
node1      Ready    <none>          100d    v1.29.9
10.2.85.66    <none>      CentOS      Linux 8
4.18.0-348.7.1.el8_5.x86_64    containerd://1.6.32

root@Master:~# ssh root@ubuntu-1 kubectl get nodes -o wide
NAME          STATUS  ROLES          AGE
VERSION      INTERNAL-IP  EXTERNAL-IP  OS-IMAGE  KERNEL-
VERSION                                CONTAINER-RUNTIME
ubuntu-1    Ready    control-plane,master  94d    v1.30.5+k3s1
10.2.144.1    <none>      Ubuntu      20.04.6 LTS 5.4.0-196-
generic      containerd://1.7.21-k3s2
ubuntu-2    Ready    <none>          94d    v1.30.5+k3s1
10.2.197.128  <none>      Ubuntu      20.04.6 LTS 5.4.0-196-
generic      containerd://1.7.21-k3s2
ubuntu-3    Ready    <none>          93d    v1.30.5+k3s1
10.2.197.129  <none>      Ubuntu      20.04.6 LTS 5.4.0-196-
generic      containerd://1.7.21-k3s2

```

Failed Authentication with the Container Registry

In some cases, your Pods are in `ImagePullBackOff` state after the deployment:

```

$ kubectl -n kube-system get Pods
NAME                                READY  STATUS
RESTARTS  AGE
coredns-58687784f9-h4pp2            1/1    Running
8                                  175d
coredns-58687784f9-znn9j            1/1    Running
9                                  175d
dns-autoscaler-79599df498-m55mg      1/1    Running
9                                  175d
illumio-kubelink-87fd8d9f6-nmh25    0/1    ImagePullBackOff
0                                  28s

```

In this case, check the description of your Pods using the following command:

```

$ kubectl -n kube-system describe Pods illumio-
kubelink-87fd8d9f6-nmh25
Name:          illumio-kubelink-87fd8d9f6-nmh25
Namespace:     kube-system
Priority:       0
Node:          node2/10.0.0.12
Start Time:    Fri, 03 Apr 2020 21:05:07 +0000
Labels:        app=illumio-kubelink
               Pod-template-hash=87fd8d9f6
Annotations:   com.illumio.role: Kubelink
Status:        Pending
IP:            10.10.65.55
IPs:
  IP:          10.10.65.55
Controlled By: ReplicaSet/illumio-kubelink-87fd8d9f6
Containers:
  illumio-kubelink:
    Container ID:
    Image:        registry.poc.segmentationpov.com/illumio-
kubelink:2.0.x.xxxxxx
    Image ID:
    Port:         <none>
    Host Port:    <none>
    State:        Waiting
      Reason:     ImagePullBackOff
    Ready:        False
    Restart Count: 0
    Environment:
      ILO_SERVER:  <set to the key 'ilo_server' in
secret 'illumio-kubelink-config'> Optional: false
      ILO_CLUSTER_UUID: <set to the key 'ilo_cluster_uuid'
in secret 'illumio-kubelink-config'> Optional: false
      ILO_CLUSTER_TOKEN: <set to the key 'ilo_cluster_token'
in secret 'illumio-kubelink-config'> Optional: false
      CLUSTER_TYPE:  Kubernetes
      IGNORE_CERT:   <set to the key 'ignore_cert' in
secret 'illumio-kubelink-config'> Optional: true
      DEBUG_LEVEL:   <set to the key 'log_level' in
secret 'illumio-kubelink-config'> Optional: true
    Mounts:
      /etc/pki/tls/ilo_certs/ from root-ca (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from
illumio-kubelink-token-7mvgk (ro)
Conditions:
  Type          Status
  Initialized    True

```



```

    Ready                False
    ContainersReady      False
    PodScheduled         True
Volumes:
  root-ca:
    Type:                ConfigMap (a volume populated by a ConfigMap)
    Name:                root-ca-config
    Optional:            false
  illumio-kubelink-token-7mvgk:
    Type:                Secret (a volume populated by a Secret)
    SecretName:          illumio-kubelink-token-7mvgk
    Optional:            false
QoS Class:               BestEffort
Node-Selectors:          <none>
Tolerations:             node-role.kubernetes.io/master:NoSchedule
                        node.kubernetes.io/not-ready:NoExecute for
300s
                        node.kubernetes.io/unreachable:NoExecute for
300s
Events:
  Type                Reason                  Age
  From                Message
  ----                -
  ----                -
  Normal              Scheduled                  <unknown>          default-
scheduler            Successfully assigned kube-system/illumio-
kubelink-87fd8d9f6-nmh25 to node2
  Normal              SandboxChanged             45s              kubelet,
node2                Pod sandbox changed, it will be killed and re-
created.
  Normal              BackOff                   14s (x4
over 45s)            kubelet, node2    Back-off
pulling image "registry.poc.segmentationpov.com/illumio-
kubelink:2.0.x.xxxxxx"
  Warning             Failed                     14s (x4 over 45s) kubelet,
node2                Error: ImagePullBackOff
  Normal              Pulling                   1s (x3 over 46s)  kubelet,
node2                Pulling image "registry.poc.segmentationpov.com/
illumio-kubelink:2.0.x.xxxxxx"
  Warning             Failed                     1s (x3 over
46s)                kubelet, node2    Failed to
pull image "registry.poc.segmentationpov.com/illumio-
kubelink:2.0.x.xxxxxx": rpc error: code = Unknown desc = Error
response from daemon: unauthorized: authentication required
  Warning             Failed                     1s (x3 over 46s)  kubelet,
node2                Error: ErrImagePull

```

The messages at the end of the output above are self-explanatory that there is a problem with the authentication against the container registry. Verify the credentials you entered in the secret for your private container registry and reapply it after fixing the issue.

Kubelink Pod in CrashLoopBackOff State

In some cases, your Kubelink Pod is in `CrashLoopBackOff` state after the deployment:

```
$ kubectl -n kube-system get Pods
```

NAME	READY	STATUS
coredns-58687784f9-h4pp2	1/1	Running
coredns-58687784f9-znn9j	1/1	Running
dns-autoscaler-79599df498-m55mg	1/1	Running
illumio-kubelink-8648c6fb68-mdh8p	0/1	CrashLoopBackOff

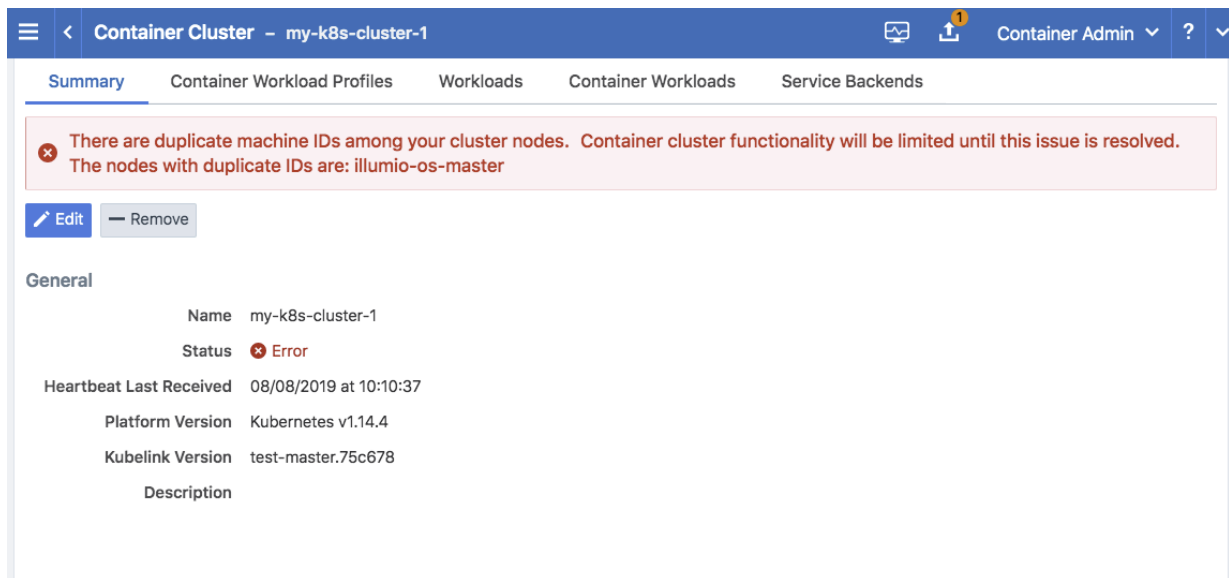
In this case, check the logs of your Pods using the following command:

```
$ kubectl -n kube-system logs illumio-kubelink-8648c6fb68-mdh8p
I, [2020-04-03T01:46:33.587761 #19] INFO -- : Starting
Kubelink for PCE
https://mypce.example.com:8443
I, [2020-04-03T01:46:33.587915 #19] INFO -- : Found 1 custom
certs
I, [2020-04-03T01:46:33.594212 #19] INFO -- : Installed
custom certs to
/etc/pki/tls/certs/ca-bundle.crt
I, [2020-04-03T01:46:33.619976 #19] INFO -- : Connecting to
PCE
https://mypce.example.com:8443
E, [2020-04-03T01:46:33.651410 #19] ERROR -- : Received a non
retriable error
401
/illumio/kubelink.rb:163:in `update_pce_resource': HTTP status
code 401 uri:
https://mypce.example.com:8443/api/v2/orgs/10/
container_clusters/
42083a4d-dd92-49e6-b495-6f84a940073c/put_from_cluster,
request_id:
21bdfc05-7b02-442d-a778-e6f2da2a462b response: request_body:
{"kubelink_version":"2.0.x.xxxxxx","errors":
[], "manager_type":"Kubernetes v1.16.0"}
(Illumio::PCEHttpException)
  from /illumio/kubelink.rb:113:in `initialize'
  from /illumio/main.rb:39:in `new'
  from /illumio/main.rb:39:in `block in main'
  from /external/lib/ruby/gems/2.4.0/gems/em-synchrony-1.0.6/
lib/em-synchrony.rb:39:in `block (2 levels) in synchrony'
```

In the example above, the request is rejected by the PCE because of a wrong identifier. Open your secret file for Kubelink, verify your cluster UUID and token, and make sure you copy-pasted the same string provided by the PCE during cluster creation.

Container Cluster in Error

In some cases, the container cluster page displays an error indicating that duplicate machine IDs were detected and functionality will be limited. See the screenshot below.



To resolve this error, follow the steps in the section below. After following those steps, restart the C-VEN Pod on each of the affected Kubernetes cluster node.

Verify Machine IDs on All Nodes

To verify machine-ids and resolve any duplicate IDs across nodes:

1. Check the machineID of all your cluster nodes with the following command:

```
kubectl get node -o yaml | grep machineID
```

```
$ kubectl get node -o yaml | grep machineID
    machineID: ec2eefcfc1bdfa9d38218812405a27d9
    machineID: ec2bcf3d167630bc587132ee83c9a7ad
    machineID: ec2bf11109b243671147b53abe1fcfc0
```

2. As an alternative, you can also to check content of the `/etc/machine-id` file on all cluster nodes. The output should be a single newline-terminated, hexadecimal, 32-character, and lowercase ID.
3. If the machine-id string is unique for each node, then the environment is OK. If the machine-id is duplicated across any of the nodes, then you must generate a machine-id for each node which has the same machine-id.
4. Running the following command displays the output of the machine-id:

```
cat /etc/machine-id
```

```
root@k8s-c2-node1:~# cat /etc/machine-id
2581d13362cd4220b20020ff728efff8
```

Generate a New Machine ID

If the machineID is duplicated on some or all of the Kubernetes nodes, use the following steps to generate a new machine-id.

- For CentOS or Red Hat:

```
rm -rf /etc/machine-id; systemd-machine-id-setup;  
systemctl restart kubelet
```

- For Ubuntu:

```
rm -rf /etc/machine-id; rm /var/lib/dbus/machine-id; systemd-  
machine-id-setup;  
systemctl restart kubelet
```



NOTE

Check the machine-id again after doing the above steps to verify that each Kubernetes cluster node has a unique machine-id.

Pods and Services Not Detected

In some cases, the Container Workloads page under **Infrastructure > Container Clusters > MyClusterName** is empty although the Workloads page has all the cluster nodes in it. This issue typically occurs when the wrong container runtime is monitored by Illumio. To resolve this issue:

1. Validate which container runtime is used in your Kubernetes or OpenShift cluster.
2. Open your configuration file for the C-VEN DaemonSet.
3. Modify the **unixsocks** mount configuration to point to the right socket path on your hosts.



NOTE

This issue typically occurs when containerd or cri-o is the primary container runtime on Kubernetes or OpenShift nodes and there is an existing docker container runtime on the nodes that is not "active" (the socket still present on the nodes and process still running, mostly some leftover from the staging phase of the servers).

Pods Stuck in Terminating State

In a Kubernetes cluster running containerd 1.2.6-10 as the container runtime, on deleting a Pod while the C-VEN is deployed may result in the Pod being stuck in a terminating state. If you see this error, redeploy the C-VEN and modify the socket path as follows:

Change the `volumeMount` and `hostPath` from `/var/run` to `/var/run/containerd` in the `illumio-ven.yaml` file

Kubelink and Abandoned or Errored C-VEs

For disconnected or abandoned C-VEs:

- Click the Unpair button in version 25.3 or newer.
- For older versions, upgrade to version 25.3 and click the Unpair button.

For errored C-VEs, if deleting a pod does not resolve the problem, first run the unpair command.

```
kubectl -n illumio-system exec illumio-ven-kcr7g - /ven-init
unpair &
```

The `&` unpairs as a background process.

If running the unpair command doesn't resolve the issue, delete the `/opt/illumio_ven_data/` directory and unpair on the PCE.

For unpairing all C-VEs at once, either:

- Use `helm uninstall` for smaller cluster sizes, which times out in 2 minutes.
- Use the following daemonset command and change the container command from `activate` to `unpair`.

```
kubectl -n illumio-system edit daemonsets.apps illumio-ven
```

This command provides more control and the ability to handle much larger cluster sizes that would take longer than 2 minutes to unpair.

If abandoned C-VEs cannot be removed and unpaired, reach out to support. Generate logs to provide support by using the `kubectl logs -p` command. The `-p` parameter retrieves the previous logs.

To increase verbosity for Helm chart during troubleshooting, edit the verbosity value in the `illumio-values.yaml` file for kubelink:

- **verbosity: 0** the default, includes errors and warnings.
- **verbosity: 1** includes errors, warnings, and debugging information.
- **verbosity: 2** includes errors, warnings, debugging information, and more details.



IMPORTANT

Verbosity levels 1 and 2 may rotate logs quickly to prevent excessive storage growth. After you collect logs for troubleshooting, reduce verbosity.

For the most current list of Helm commands and readme, see <https://quay.io/repository/illumio/illumio>.

Troubleshooting CLAS Mode Architecture

If your upgrade or installation of CLAS-enabled clusters exhibits unusual behavior, follow these steps to troubleshoot the issue:

1. Check that the Kubelinks and C-VEs are operating. Verify at the Container Clusters page, in the Summary and VEs tabs.

[Home](#) > [Infrastructure](#) > [Container Clusters](#)

Demo

[Summary](#) [VEs](#) [Container Workload Profiles](#) [Workloads](#) [Kubernetes Workloads](#) [Service Backends](#)

[Edit](#) [Remove](#)

GENERAL

Name	Demo
Status	● In Sync
Heartbeat Last Received	01/08/2024 at 09:32:37
Platform Version	Kubernetes v1.27.8+IKS
Kubelink Version	5.0.0-150
Container Runtime	containerd
Cluster Local Actor Store	Enabled
Description	

Home > Infrastructure > Container Clusters

Demo

Summary **VENs** Container Workload Profiles Workloads Kubernetes Workloads Service Backends

Status	Health	Name	Version	Labels	OS	Enforcement Node Type
Active	✓	kube-clu637ud0mmlvajo7a0-myclusterus-jmazag2-000002af	23.3.0-59-dev		centos-x86_64-7.0	Containerized VEN (C-VEN)
Active	✓	kube-clu637ud0mmlvajo7a0-myclusterus-jmazag2-000003d2	23.3.0-59-dev		centos-x86_64-7.0	Containerized VEN (C-VEN)
Active	✓	kube-clu637ud0mmlvajo7a0-myclusterus-jmazag2-00000165	23.3.0-59-dev		centos-x86_64-7.0	Containerized VEN (C-VEN)

- Check that Kubernetes Namespace and Kubernetes Workload objects are created, and are present in the PCE. Verify from the Kubernetes Workloads tab.

Home > Infrastructure > Container Clusters

Demo

Summary VENs Container Workload Profiles Workloads **Kubernetes Workloads** Service Backends

Refresh

Namespace/Project: votingapp

Namespace/Project	Name	Policy Sync	Enforcement	Visibility	Kind	Labels	Last Applied Policy
votingapp	azure-vote-back	Active	Visibility Only	Blocked + Allowed	Deployment	Backend votingapp	01/08/2024, 09:27:20
votingapp	azure-vote-back	Active	Visibility Only	Blocked + Allowed	Service	BackendService votingapp	01/08/2024, 09:27:20
votingapp	azure-vote-front	Active	Visibility Only	Blocked + Allowed	Deployment	Frontend votingapp	01/08/2024, 09:27:20
votingapp	azure-vote-front	Active	Visibility Only	Blocked + Allowed	Service	FrontendService votingapp	01/08/2024, 09:27:20
votingapp	azure-vote-front-lb	Active	Visibility Only	Blocked + Allowed	Service	FrontendServiceLoadBalancer votingapp	01/08/2024, 09:27:20
votingapp	azure-vote-front-nodeport	Active	Visibility Only	Blocked + Allowed	Service	FrontendServiceNodePort votingapp	01/08/2024, 09:27:20
votingapp	azure-vote-front2	Active	Visibility Only	Blocked + Allowed	Service		01/08/2024, 09:27:20

- Check that labeling is done correctly by examining each Kubernetes Workload:

Home > Servers & Endpoints > ...

azure-vote-front

Summary Rules

GENERAL

Name: azure-vote-front

Container Cluster: Demo

Namespace/Project: votingapp

Enforcement: Visibility Only
No traffic is blocked by policy

Visibility: Blocked + Allowed
VEN logs connection information for allowed, blocked and potentially blocked traffic

Policy Sync: Active

Firewall Coexistence Mode: Exclusive

Created At: 01/08/2024 at 08:46:12

Last Modified At: 01/08/2024 at 16:59:20

LABELS

Labels: Frontend Votingapp

KUBERNETES ATTRIBUTES

Kind: Deployment

my-app-name: front3

com.illumio.app: Votingapp

com.illumio.env: Development

com.illumio.loc: Amazon

com.illumio.role: Frontend

deployment.kubernetes.io/revision: 1

KUBERNETES LABELS

app: azure-vote-front

4. Check that NodePort Services have correct IPs. Find NodePort IPs on the Service Backends tab, under the Virtual Services table column.

Home > Infrastructure > Container Clusters

Demo

Summary VENS Container Workload Profiles Workloads Kubernetes Workloads Service Backends

Refresh

Name	Resource Type	Namespace	Virtual Service	Last Modified On
NodePort:0ce061810255881a85019b453955b15e	nodeport	votingapp	azure-vote-front-lb-upg-votingapp	01/08/2024, 08:57:46
NodePort:932e90a61c7321e865529fb0c446c8e5	nodeport	votingapp	azure-vote-front-nodeport-upg-votingapp	01/08/2024, 08:46:08

Home > Policy Objects > Virtual Services

azure-vote-front-nodeport-upg-votingapp

Summary Workloads

This Virtual Service is managed by a Container Cluster

Edit Remove

GENERAL

Name: azure-vote-front-nodeport-upg-votingapp

Description:

Created: 01/08/2024 at 08:46:07 by Container Cluster

Last Modified: 01/08/2024 at 08:46:07 by Container Cluster

CONNECTION SERVICE OR PORTS

Service or Ports: 80 TCP

LABELS

Labels: FrontendServiceNodeport Votingapp

ADDRESS POOL

IP Addresses and FQDNs: 172.21.20.146 "upg container network"

ADVANCED

Pool Target: Host Only (Default)

- Check that traffic is flowing properly. Find the pod-to-pod, and pod-to-application flows with IP information from the Traffic view. Select "Individual Connections" to see the name of the Kubernetes Workloads and the IPs of the Pods sending and receiving the traffic.

Home > Explore

Traffic

Search All Categories: Votingapp

Time: Last Month More

Clear Query Save Query EDIT Source: Votingapp TL Run Load Results

Individual Connections

Allow Selected Connections Resolve Unknown FQDNs Export

Timestamp: 01/09/2024, 07:12:15 Connections: 1 - 6 of 6

Reported Policy Decision	Source	Source Labels	Source Port/Process User	Destination	Destination Labels	Destination Port Process User	Flows/Bytes	First Detected	Last Detected
Allowed by Rule by Destination	azure-vote-front 172.17.182.161 Visibility Only	Frontend Votingapp		azure-vote-back 172.17.182.160 Visibility Only	Backend Votingapp	6379 TCP	950 Flows Corporate	01/08/2024, 11:02:20	01/09/2024, 06:57:57
Allowed by Rule by Source	azure-vote-front 172.17.182.161 Visibility Only	Frontend Votingapp		azure-vote-back 172.21.56.49 Visibility Only	BackendService Votingapp	6379 TCP	954 Flows Corporate	01/08/2024, 11:02:20	01/09/2024, 06:57:57
Allowed by Rule by Destination	azure-vote-front 172.17.182.161 Visibility Only	Frontend Votingapp		azure-vote-back 172.17.182.160 Visibility Only	Backend Votingapp	6379 TCP	34 Flows upg container network	01/08/2024, 08:52:19	01/08/2024, 10:57:19
Allowed by Rule by Source	azure-vote-front 172.17.182.161 Visibility Only	Frontend Votingapp		azure-vote-back 172.21.56.49 Visibility Only	BackendService Votingapp	6379 TCP	30 Flows upg container network	01/08/2024, 09:57:19	01/08/2024, 10:57:19
Allowed by Rule by Source	azure-vote-front 172.17.182.161 Visibility Only	Frontend Votingapp		azure-vote-back 172.21.56.49 Visibility Only	BackendService Votingapp	6379 TCP	4 Flows upg container network	01/08/2024, 08:52:19	01/08/2024, 09:51:35
Allowed by Rule by Source	azure-vote-front 172.17.182.161 Visibility Only	Frontend Votingapp		172.21.0.10		53 UDP	4 Flows upg container network	01/08/2024, 08:52:19	01/08/2024, 08:52:19

Src Pod IPs

Dst Pod IPs

Dst Cluster IPs

Known Limitations

The known limitations are:

- C-VEN on server hosting containers and Enforced policy state limitation

If a C-VEN on a server hosting containers is paired directly into the Enforced policy state, other nodes may lose connectivity with the master node until policy is synchronized across all the nodes.

- **Pods running on host network stack limitation**

Pods which run on the host network stack (inherit the host IP address) are not reported to the PCE. Any rules written for the host will also be inherited by any *hostNetworked Pods* on the host.

- **Kubernetes stateless rules limitation**

The use of stateless rules is not recommended because Kubernetes uses NAT tables, which depend on traffic being tracked and stateful.

- **Kubernetes service ports and pod firewall in Non-CLAS limitation**

If a Kubernetes service has both port 1234/TCP and port 2345/UDP configured, a rule configured with the Pod as source and the virtual service as Provider will open up both ports 1234/TCP and 2345/TCP, and 1234/UDP and 2345/UDP on the Pod's firewall (outbound rule).

In case of a Kubernetes service configured with a **port** and **targetPort** statement in the manifest file as shown in the example below:

```
apiVersion: v1
kind: Service
metadata:
  name: web-frontend-svc
  namespace: app1
  labels:
    app: app1
    tier: web-frontend
  annotations:
    com.illumio.role: Web
spec:
  type: ClusterIP
  ports:
    - port: 8080
      targetPort: 80
      protocol: TCP
    - port: 8081
      targetPort: 81
      protocol: UDP
  selector:
    app: app1
    tier: web-frontend
```

This configuration is supported with Illumio Core. In this case, only the port number associated to the **port** statement will show this issue, the port number associated to the **targetPort** statement will not show this issue and will use the **protocol** specified in the Service yaml file.

- **NodePort services in OpenShift with OVN limitation**

NodePort services in OpenShift with OVN are not enforceable at the service or node level by Illumio; policy is enforced only on the destination workloads.

In OpenShift clusters using OVN CNI, NodePort traffic routes through a virtualized network infrastructure comprising OVS routers, switches, and load balancers. Unlike traditional CNIs like Calico, where traffic passes through the host's IP stack and is subject to DNAT and filtering, OVN handles NodePort traffic at the virtual network layer. This means traffic reaches the target pod without traversing the host workload's IP stack, complicating visibility and enforcement.

Because NodePort traffic in OVN bypasses the host IP stack, Illumio cannot natively observe or enforce policies on it. The only observable point is at the pod level, where traffic appears to originate from the Gateway router IP (e.g., 100.64.x.x), obscuring the original source. This lack of visibility makes it difficult to apply precise security rules, as policies must allow traffic from all Gateway router IPs, potentially exposing the pod to unintended sources.

- **NodePort namespace traffic enforcement and container hosts limitation**

NodePort namespace traffic does not get enforced unless container hosts are enforced, **even with full enforcement enabled**.

The container hosts control what accesses the namespace. If you want to use NodePort and want to enforce NodePort traffic, you must enforce container hosts traffic, first.

Kubelink Monitoring and Troubleshooting

If you deployed Illumio Core for Kubernetes 3.0.0 or later, Kubelink is deployed as part of the overall Helm Chart deployment, as described in [Deployment with Helm Chart \(Core for Kubernetes 3.0.0 and Higher\)](#) [187].

Kubelink Process

Kubelink uses a single Ruby process which runs as: **ruby /illumio/init.rb**.

Kubelink Startup Log Messages

After deploying Kubelink (whether by Helm Chart or manually), verify your deployment with the **kubectl get pods -n illumio-system** command. The **kubelinkpod** should be shown with the Running status. In addition, you can review the log file entries after the deployment with the **kubectl logs** command pointing to the Kubelink pod name.

```
kubectl logs <kubelink_pod_name> -n illumio-system
```

A typical successful Kubelink deployment produces log entries similar to these:

```
I, [2022-05-23T14:36:53.847248 #10] INFO -- : Starting
Kubelink for PCE https://192.168.88.127:10443
I, [2022-05-23T14:36:53.847502 #10] INFO -- : Metrics
reporting enabled; reporting window 30
I, [2022-05-23T14:36:53.847520 #10] INFO -- : PCE fqdn
https://192.168.88.127:10443
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:36:53.893048 #10] INFO -- : Successfully
connected to PCE
I, [2022-05-23T14:36:53.893170 #10] INFO -- : begin sync on
resource namespaces
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:36:53.904369 #10] INFO -- : Synchronized 6
namespaces.
I, [2022-05-23T14:36:53.904424 #10] INFO -- : sync on
resource namespaces successful, setting up resource version to
184232
I, [2022-05-23T14:36:53.904522 #10] INFO -- : Start watch on
namespaces with version 184232
I, [2022-05-23T14:36:53.905678 #10] INFO -- : begin sync on
resource nodes
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:36:53.918093 #10] INFO -- : Synchronized 1
nodes.
I, [2022-05-23T14:36:53.918143 #10] INFO -- : sync on
resource nodes successful, setting up resource version to
184232
I, [2022-05-23T14:36:53.918175 #10] INFO -- : Start watch on
nodes with version 184232
I, [2022-05-23T14:36:53.919265 #10] INFO -- : begin sync on
resource pods
I, [2022-05-23T14:36:53.935536 #10] INFO -- : sync on
resource pods successful, setting up resource version to 184232
```

```

I, [2022-05-23T14:36:53.935601 #10] INFO -- : Start watch on
pods with version 184232
I, [2022-05-23T14:36:53.936938 #10] INFO -- : begin sync on
resource services
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:36:54.029965 #10] INFO -- : Synchronized 3
services, full=true, force=false
I, [2022-05-23T14:36:54.030013 #10] INFO -- : sync on
resource services successful, setting up resource version to
184232
I, [2022-05-23T14:36:54.030046 #10] INFO -- : Start watch on
services with version 184232
I, [2022-05-23T14:36:54.031042 #10] INFO -- : begin sync on
resource replica_sets
I, [2022-05-23T14:36:54.100090 #10] INFO -- : Nothing to sync
I, [2022-05-23T14:36:54.100237 #10] INFO -- : sync on
resource replica_sets successful, setting up resource version
to 184232
I, [2022-05-23T14:36:54.100281 #10] INFO -- : Start watch on
replica_sets with version 184232
I, [2022-05-23T14:36:54.101226 #10] INFO -- : begin sync on
resource stateful_sets
I, [2022-05-23T14:36:54.170175 #10] INFO -- : Nothing to sync
I, [2022-05-23T14:36:54.170220 #10] INFO -- : sync on
resource stateful_sets successful, setting up resource version
to 184232
I, [2022-05-23T14:36:54.170267 #10] INFO -- : Start watch on
stateful_sets with version 184232
I, [2022-05-23T14:36:54.171159 #10] INFO -- : begin sync on
resource daemon_sets
I, [2022-05-23T14:36:54.245866 #10] INFO -- : Nothing to sync
I, [2022-05-23T14:36:54.246025 #10] INFO -- : sync on
resource daemon_sets successful, setting up resource version
to 184232
I, [2022-05-23T14:36:54.246210 #10] INFO -- : Start watch on

```

```

daemon_sets with version 184232
I, [2022-05-23T14:36:54.247946 #10] INFO -- : begin sync on
resource replication_controllers
I, [2022-05-23T14:36:54.324925 #10] INFO -- : Nothing to sync
I, [2022-05-23T14:36:54.324977 #10] INFO -- : sync on
resource replication_controllers successful, setting up
resource version to 184232
I, [2022-05-23T14:36:54.325032 #10] INFO -- : Start watch on
replication_controllers with version 184232
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:36:54.505403 #10] INFO -- : replica_sets
MODIFIED
I, [2022-05-23T14:37:24.312086 #10] INFO -- : Heart beating
to PCE
I, [2022-05-23T14:37:24.312191 #10] INFO --
: Attaching metrics report to heartbeat:
{:pod_changes=>[{:namespace=>"illumio-system", "added"=>0,
"modified"=>0, "deleted"=>1}], :service_changes=>[],
:duration_seconds=>30}
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:37:54.343467 #10] INFO -- : Heart beating
to PCE
I, [2022-05-23T14:37:54.343874 #10] INFO -- :
Attaching metrics report to heartbeat: {:pod_changes=>[],

```



```

:service_changes=>[], :duration_seconds=>30}
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:38:24.373847 #10] INFO -- : Heart beating
to PCE
I, [2022-05-23T14:38:24.373924 #10] INFO -- :
Attaching metrics report to heartbeat: {:pod_changes=>[],
:service_changes=>[], :duration_seconds=>30}
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:38:54.380933 #10] INFO -- : Heart beating
to PCE
I, [2022-05-23T14:38:54.381009 #10] INFO -- :
Attaching metrics report to heartbeat: {:pod_changes=>[],
:service_changes=>[], :duration_seconds=>30}
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:39:24.401636 #10] INFO -- : Heart beating
to PCE
I, [2022-05-23T14:39:24.401748 #10] INFO -- :
Attaching metrics report to heartbeat: {:pod_changes=>[],
:service_changes=>[], :duration_seconds=>30}
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:39:54.422494 #10] INFO -- : Heart beating
to PCE
I, [2022-05-23T14:39:54.422595 #10] INFO -- :
Attaching metrics report to heartbeat: {:pod_changes=>[],
:service_changes=>[], :duration_seconds=>30}
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:40:24.453077 #10] INFO -- : Heart beating
to PCE
I, [2022-05-23T14:40:24.453217 #10] INFO -- :
Attaching metrics report to heartbeat: {:pod_changes=>[],
:service_changes=>[], :duration_seconds=>30}

```

```

[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:40:54.466210 #10] INFO -- : Heart beating
to PCE
I, [2022-05-23T14:40:54.466455 #10] INFO -- :
Attaching metrics report to heartbeat: {:pod_changes=>[],
:service_changes=>[], :duration_seconds=>30}
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:41:24.296410 #10] INFO -- :
Verify watches for ["namespaces", "nodes", "pods",
"services", "replica_sets", "stateful_sets", "daemon_sets",
"replication_controllers"]
I, [2022-05-23T14:41:24.296468 #10] INFO -- : Watch client
namespaces Connection Idle: 270.3355407714844s
I, [2022-05-23T14:41:24.296485 #10] INFO -- : Watch client
nodes Connection Idle: 179.93679809570312s
I, [2022-05-23T14:41:24.296499 #10] INFO -- : Watch client
pods Connection Idle: 240.5237274169922s
I, [2022-05-23T14:41:24.296513 #10] INFO -- : Watch client
services Connection Idle: 270.0260314941406s
I, [2022-05-23T14:41:24.296526 #10] INFO -- : Watch client
replica_sets Connection Idle: 269.85888671875s
I, [2022-05-23T14:41:24.296542 #10] INFO -- : Watch client
stateful_sets Connection Idle: 270.0269775390625s
I, [2022-05-23T14:41:24.296573 #10] INFO -- : Watch client
daemon_sets Connection Idle: 270.02490234375s
I, [2022-05-23T14:41:24.296731 #10] INFO -- : Watch client
replication_controllers Connection Idle: 270.02490234375s
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:41:24.300532 #10] INFO -- : Synchronized 3
services, full=true, force=true
I, [2022-05-23T14:41:24.452846 #10] INFO -- : Heart beating
to PCE
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
W, [2022-05-23T14:41:54.186807 #10] WARN -- : watch client

```

```

for stateful_sets error callback invoked. Resetting watch ...
W, [2022-05-23T14:41:54.186863 #10] WARN -- : Watch on
stateful_sets ended. Resetting it after 3 seconds
I, [2022-05-23T14:41:54.441880 #10] INFO -- : Heart beating
to PCE
I, [2022-05-23T14:41:54.441991 #10] INFO -- :
Attaching metrics report to heartbeat: {:pod_changes=>[],
:service_changes=>[], :duration_seconds=>60}
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:41:57.193339 #10] INFO -- : begin sync on
resource stateful_sets
I, [2022-05-23T14:41:57.267375 #10] INFO -- : Nothing to sync
I, [2022-05-23T14:41:57.267411 #10] INFO -- : sync on
resource stateful_sets successful, setting up resource version
to 184451
I, [2022-05-23T14:41:57.267424 #10] INFO -- : Start watch on
stateful_sets with version 184451
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details
I, [2022-05-23T14:42:24.483142 #10] INFO -- : Heart beating
to PCE
I, [2022-05-23T14:42:24.483224 #10] INFO -- :
Attaching metrics report to heartbeat: {:pod_changes=>[],
:service_changes=>[], :duration_seconds=>30}
[WARNING; em-http-request] TLS hostname validation is disabled
(use 'tls: {verify_peer: true}'), see CVE-2020-13482 and
https://github.com/igrigorik/em-http-request/issues/339 for
details

```

Verify Kubelink Deployment

To verify your Kubelink deployment.

- To check the Kubelink Pod status for Kubernetes:

```
kubectl get pods -n illumio-system
```

- To check the Kubelink Pod status for OpenShift:

```
oc get pods -n illumio-system
```

The `illumio-kubelink-xxxxxxxxxx-xxxxx` Pod should be in the "Running" state. If the either `get pods -n illumio-system` command shows the ku-

belink pod is not successfully running, check the log file for any ERROR messages.

After Kubelink is successfully deployed, you can check the cluster information in the Illumio PCE UI. From the main menu, navigate to **Infrastructure > Container Clusters**.

Below is an example of a healthy container cluster state reported by Kubelink, where Status is "In Sync".

Container Clusters - MyK8sCluster

Summary | Container Workload Profiles | Workloads | Container Workloads | Service Backends

[Edit](#) [Remove](#)

General

Name	MyK8sCluster
Status	● In Sync
Heartbeat Last Received	04/02/2020 at 19:03:53
Platform Version	Kubernetes v1.17.0
Kubelink Version	1.3.0.742af6
Description	

You can also verify in the PCE UI that Kubelink was successfully deployed by checking the following:

- Under the **Container Workload Profiles** tab, namespaces created in your Kubernetes or OpenShift cluster should be listed. An example is shown below.

Container Clusters - MyK8sCluster

Summary | **Container Workload Profiles** | Workloads | Container Workloads | Service Backends

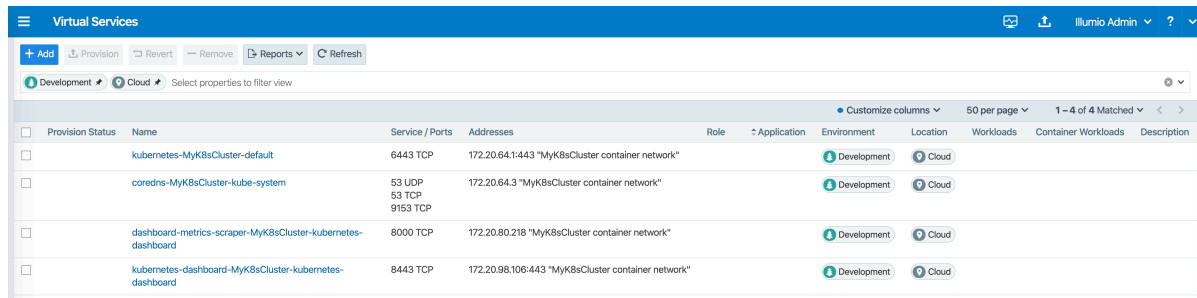
Container Workload Profiles will be created automatically for discovered Namespaces using the following template: Set Policy State to "Build". Set Label Assignments to ● Development ● Cloud. [Edit default settings](#)

[Add](#) [Remove](#) [Refresh](#)

Select properties to filter view

Name	Namespace	Policy State	Role	Application	Environment	Location	Last Modified On	Last Modified By
	default	Build			Development	Cloud	04/02/2020, 19:03:53	Container Cluster
	ingress-nginx	Build			Development	Cloud	04/02/2020, 19:03:53	Container Cluster
	kube-node-lease	Build			Development	Cloud	04/02/2020, 19:03:53	Container Cluster
	kube-public	Build			Development	Cloud	04/02/2020, 19:03:53	Container Cluster
	illumio-system	Build			Development	Cloud	04/02/2020, 19:03:53	Container Cluster
	kubernetes-dashboard	Build			Development	Cloud	04/02/2020, 19:03:53	Container Cluster

- Under **Policy Objects > Virtual Services**, services created in your Kubernetes or OpenShift cluster should be listed. An example is shown below.



The screenshot shows the 'Virtual Services' interface with a table of services. The table has columns for Provision Status, Name, Service / Ports, Addresses, Role, Application, Environment, Location, Workloads, Container Workloads, and Description. There are four rows of services listed.

Provision Status	Name	Service / Ports	Addresses	Role	Application	Environment	Location	Workloads	Container Workloads	Description
☐	kubernetes-MyK8sCluster-default	6443 TCP	172.20.64.1:443 "MyK8sCluster container network"			Development	Cloud			
☐	coredns-MyK8sCluster-kube-system	53 UDP 9153 TCP	172.20.64.3 "MyK8sCluster container network"			Development	Cloud			
☐	dashboard-metrics-scraper-MyK8sCluster-kubernetes-dashboard	8000 TCP	172.20.80.218 "MyK8sCluster container network"			Development	Cloud			
☐	kubernetes-dashboard-MyK8sCluster-kubernetes-dashboard	8443 TCP	172.20.98.106:443 "MyK8sCluster container network"			Development	Cloud			

PCE-Kubelink Connection and Heartbeat

The Kubelink heartbeat to the PCE is logged in its log file. Use the **kubect1 logs** command, and search for the **Heart beating to PCE** string to confirm. To confirm PCE-Kubelink connectivity, check the PCE UI, which will show the Kubelink pod as being offline if the heartbeat is missing 2-3 times (about 10 minutes).

Additional Kubelink Monitoring

Other Kubelink actions that can be confirmed in the Kubelink log file include:

API request succeeds

When Kubelink successfully sets up a watch with the Kubernetes API, the related log entry is:

```
sync on resource <RESOURCE> successful, setting up resource
version to <RESOURCE VERSION>
```

Information sent to PCE

When Kubelink successfully sends information to the PCE, the related log entry is:

```
Synchronized 2 <RESOURCE>, full=..., force=...
```

Setting Log Verbosity

The log verbosity level is set by default to include INFO, WARNING, and ERROR messages in the log. If your log appears to be extremely small (showing only ERRORS, for example), or is extremely large (which could indicate being set at the DEBUG level), you can check the **log_level** setting in the **illumio-kubelink-secret.yml** file. Values for this setting are:

log_level Setting	Description
0	Debug
1	Info (default)
2	Warn
3	Error

Values are cumulative, meaning that a setting includes all other settings greater than it. For example, the default setting of '1' includes all INFO, WARNING, and ERROR messages in the log file, but a setting of '3' would only include ERROR messages.

Aggregating Logs from Kubelink and C-VEN Pods

There are many log aggregation solutions; this topic describes one example of using Fluent Bit to aggregate our logs. Fluent Bit is a lightweight version of Fluentd with many outputs. See <https://docs.fluentbit.io/manual/pipeline/outputs> for official details about supported Fluent Bit output plugins.

Loki is used as storage in this example. Change the output section of your Fluent Bit yaml file to suit your needs.

Loki and Grafana

As an example installation for testing, Loki and Grafana are installed in the `illumio-system` namespace. Loki is installed in monolithic mode to use file system storage. For more details, see <https://grafana.com/docs/loki/latest/setup/install/helm/install-monolithic/>.

```
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update
helm upgrade --install loki grafana/loki --values loki-values.yaml -n illumio-system
```

Example contents of `loki-values.yaml`:

```

loki:
  commonConfig:
    replication_factor: 1
  storage:
    type: 'filesystem'
  auth_enabled: false

singleBinary:
  replicas: 1

# lokiCanary:
#   enabled: false
# gateway:
#   enabled: false
# grafanaAgent:
#   installOperator: true

```

```

helm upgrade --install --wait -n illumio-system --set
admin.username=admin --set admin.password=UseYourPassword --
set persistence.enabled=false -f grafana-values.yaml grafana
oci://registry-1.docker.io/bitnamicharts/grafana
kubectl -n illumio-system expose deployment grafana --
type=NodePort --name=grafana-service
kubectl -n illumio-system get svc grafana-service -o
go-template='{{range.spec.ports}}{{if .nodePort}}{{.nodePort}}
{{"\n"}}{{end}}{{end}}'

```

Example contents of `grafana-values.yaml`:

```

dashboardsProvider:
  enabled: true

```

Fluent Bit

The following procedure shows one way of downloading and installing Fluent Bit:

```

helm repo add fluent https://fluent.github.io/helm-charts
helm repo update
helm upgrade --install fluent-bit fluent/fluent-bit --version
0.40.0 --values fluentbit-values.yaml -n illumio-system
kubectl --namespace illumio-system patch daemonsets.apps
fluent-bit --patch-file fluentbit-patch-nodename.yaml

```

Example contents of `fluentbit-values.yaml`:

```

labels
  app: IllumioFluentBit

image:
  pullPolicy: IfNotPresent

extraVolumes:
  - name: illumio-ven-data
    hostPath:
      path: /opt/illumio_ven_data
      type: Directory

extraVolumeMounts:
  - name: illumio-ven-data
    mountPath: /opt/illumio_ven_data

config:
  service: |
    [SERVICE]
      daemon Off
      flush {{ .Values.flush }}
      log_level debug
      parsers_file parsers.conf
      parsers_file custom_parsers.conf
      http_server On
      http_listen 0.0.0.0
      http_port {{ .Values.metricsPort }}
      health_check On

  inputs: |
    [INPUT]
      Name tail
      Path /var/log/containers/illumio-kubelink*.log
      Tag kubelink.*
      Multiline.parser docker,cri
      Read_From_Head true
      Buffer_Chunk_Size 3MB
      Buffer_Max_Size 10MB
      Mem_Buf_Limit 10MB
      Skip_Long_Lines Off
    [INPUT]
      Name tail
      Path /opt/illumio_ven_data/log/*.log
      Tag cven.*
      Read_From_Head true
      Buffer_Chunk_Size 3MB

```



```

    Buffer_Max_Size 10MB
    Mem_Buf_Limit 10MB
    Skip_Long_Lines Off

filters: |
  [FILTER]
    Name kubernetes
    Match kubelink.*
    Merge_Log On
    Kube_Tag_Prefix kubelink.var.log.containers.
    Merge_Log_Key log_processed
  [FILTER]
    Name parser
    Parser cvenparser
    Match cven.*
    Key_name log
    Preserve_key false
    Reserve_data true
  [FILTER]
    Name record_modifier
    Match cven.*
    Record nodename ${K8S_NODE_NAME}

upstream: {}

outputs: |
  [OUTPUT]
    #for debugging only should be turned off in PROD
    #PLEASE TURN OFF IN PROD
    Name stdout
    Match *

    [OUTPUT]
      Name                                loki
      Match                               *
      Host                                loki.illumio-
system.svc.cluster.local
      Port                                3100
      Labels                              job=fluentbit

customParsers: |
  [PARSER]
    Name      cvenparser
    Format     regex
    Regex      ^(?<time>[^\ ]+) (?<message>.+)$
    Time_Key   time

```

```

    Time_Format %Y-%m-%dT%H:%M:%S.%L

    extraFiles {}

logLevel: info

```

Example contents of `fluentbit-patch-nodeport.yaml`:

```

spec:
  template:
    spec:
      containers:
      - name: fluent-bit
        env:
          - name: K8S_NODE_NAME
            valueFrom:
              fieldRef:
                fieldPath: spec.nodeName

```

Reference: OpenShift Deployment

Prepare OpenShift

If you don't perform the prerequisite steps before installing VEN and Kubelink, containerized environments and Kubelink may be disrupted.

Unique Machine ID

Some of the functionalities and services provided by the Illumio VEN and Kubelink depend on the Linux machine-id of each OpenShift cluster node. Each machine-id must be unique. By default, the Linux OS generates a random machine-id to give each Linux host uniqueness. However, there are cases when machine-ids can be duplicated across machines. This is common across deployments that clone machines from a golden image, for example, spinning up virtual machines from VMware templates or creating Amazon EC2 instances from an AMI.

To verify machine-ids and resolve any duplicate machine-ids across nodes:

1. **ssh** into every node of the OpenShift cluster (master, infra, and worker) as the root user.

2. Check the contents of the `/etc/machine-id` file. The output is a string of letters and numbers.
3. If the machine-id string is unique for each node, then the environment is ok. If the machine-id is duplicated across any of the nodes, you must generate a machine-id for each node which has the same machine-id.

Run the following command to view the output of machine-id:

```
cat /etc/machine-id
```

If the machine-id is duplicated, then run the command listed below to generate a new machine-id. You will also need to restart the **atomic-OpenShift-node** service on each node. If the machine-id is not duplicated, go to the next section.

```
rm -rf /etc/machine-id; touch /etc/machine-id; systemd-machine-id-setup;  
service atomic-OpenShift-node restart
```



NOTE

Check the machine-id to verify that each machine has a unique machine-id.

Create Labels

For details on creating labels, see "Labels and Label Groups" in the Security Policy Guide.

The labels listed below are used in examples throughout this document. You are not required to use the same labels.

Name	Label type
Openshift Infrastructure	Application
Development	Environment
HQ	Location
Kubelink	Role
Master	Role
Infra	Role
Compute	Role

Create Pairing Profiles

After creating labels for your OpenShift cluster nodes, you can use those labels to create pairing profiles. You do not need to create pairing profiles for container workloads.

For ease of configuration and management, consider applying the same Application, Environment, and Location labels across all nodes of the same OpenShift cluster. The screenshot below show examples of three pairing profiles for one OpenShift Enterprise cluster. The pairing profiles are used for pairing either master, compute, or infrastructure nodes of an OpenShift cluster.



TIP

It is recommended that all pairing profiles for OpenShift nodes **not** use Enforced policy state.

Move into Enforced state after you have completed all other configuration steps (setup Kubelink, discover services, and write rules).

Customize columns 50 per page 1 – 3 of 3 Matched						
☐ Pairing Status	Name	Policy State	Role	Application	Environment	Location
☐ Running	Openshift Compute	Build	Compute	Openshift Infrastructure	Development	HQ
☐ Running	Openshift Infra	Build	Infra	Openshift Infrastructure	Development	HQ
☐ Running	Openshift Master	Build	Master	Openshift Infrastructure	Development	HQ

Deploy Kubelink

Download the required resources, such as the Kubelink Docker image, secret file, and deployment file, from the Illumio Support portal (login required).

Prerequisites

- Kubelink deployment file provided by Illumio. For OpenShift deployments, the file name is `illumio-kubelink-openshift.yml`.
- Kubelink secret file provided by Illumio. This file name is `illumio-kubelink-secret.yml`.
- Illumio's Kubelink Docker image was uploaded to your private Docker registry.

Create Container Cluster

1. Log in to the PCE as a user with Global Organization Owner privileges.
2. From the PCE web console menu, choose **Infrastructure > Container Clusters**.
3. Click **Add**.
 - a. Enter a Name.
 - b. **Save** the Container Cluster.
4. You will see a summary page of the new Container Cluster. Copy the values of the Cluster ID and Cluster Token found under the Cluster Pairing Token section.
5. Once you have the values, you can exit the Container Cluster page.

Container Cluster - my-k8s-cluster-1

Summary Container Workload Profiles Workloads Container Workloads Service Backends

Edit Remove

General

Name my-k8s-cluster-1

Status ☒ Not yet connected

Platform Version

Kubelink Version

Description

Please copy the following token and ID. This information will not be available after you leave the page. A new token will need to be generated.

Cluster Pairing Token

Cluster ID dc1ecbf9-f481-44a7-a4b7-fb028b1b4ece

Copy ID

Cluster Token 1_d37ea3dcd34ae8ae2a78fb33f4e159cc4003e95cc4babe0d992062127a21dab4

Copy Token

Configure Container Workload Profile

When configuring a new Container Cluster, it is recommended to set the default settings shared by all the Container Workload Profiles. Illumio provides a Container Workload Profile template that can be used for that purpose. By defining the default Policy State and minimum set of labels common to all namespaces in the cluster, you will save time later on when new namespaces are discovered by Kubelink. Each new profile created will inherit what was defined in the template.

SSL Verification

Illumio does not provide a simple way to redefine all at once the labels associated to each profile all at once in this release, so it is strongly recommended to use this template to define the default values for all profiles part of the same cluster.

To define the default parameters for all profiles using a template, under Container Workload Profiles, click on Edit default settings and fill in the different fields. An example is shown below:

Container Workload Profile Template

i Editing the template does not affect existing Container Workload Profiles.

Policy State

Build

Label Assignments for container workloads

Role

Select a Role

Application

Select an Application

Environment

Development
Select an Environment

Location

HQ
Select a Location

Cancel

OK

Once you validate, you should see something like the following:

Summary
Container Workload Profiles
Workloads
Container Workloads
Service Backends

i Container Workload Profiles will be created automatically for discovered Namespaces using the following template: Set Policy State to "Build". Set Label Assignments to Development HQ.

Edit default settings

+ Add
Remove

Refresh

Select properties to filter view

Configure Kubelink Secret

This step assumes that you have created a Container Cluster object in the PCE. You will need the Cluster ID and Cluster Token values for the Kubelink secret.

1. `ssh` to the master node.
2. Open the Kubelink secret YAML file and modify the `stringData`.
 - a. `ilo_server` = the PCE URL and port. Example: `https://mypce.example.com:8443`
 - b. `ilo_cluster_uuid` = Cluster ID value from the previous step. Example: `dc1ecbf9-f481-44a7-a4b7-fb028b1b4ece`
 - c. `ilo_cluster_token` = Cluster Token from the previous step. Example: `1_d37ea3dcd34ae8ae2a78fb33f4e159cc4003e95cc4babe0d992062127a21dab4`

- d. **ignore_cert** = SSL verification. The value is boolean and is recommended to be set to false so that Kubelink requires PCE certificate verification. Example: **'false'**
- e. **log_level** = Log level where **'0'** for debug, **'1'** for info, **'2'** for warn, or **'3'** for error. Example: **'1'**

SSL Verification

Illumio does not recommend turning off SSL verification (**ignore_cert: 'true'**); however, this is an option for deployments in which the PCE uses a self-signed certificate.

Contents of a modified `illumio-kubelink-secret.yml` file are shown below.

```
#
# Copyright 2013-2020 Illumio, Inc. All Rights Reserved.
#

apiVersion: v2
kind: Secret
metadata:
  name: illumio-kubelink-config
  namespace: kube-system
type: Opaque
stringData:
  ilo_server: https://mypce.example.com:8443 # Example:
https://mypce.example.com:8443
  ilo_cluster_uuid: dc1ecbf9-f481-44a7-a4b7-fb028b1b4ece #
Example: cc4997c1-408b-4f1d-a72b-91495c24c6a0
  ilo_cluster_token:
1_d37ea3dcd34ae8ae2a78fb33f4e159cc4003e95cc4babe0d992062127a2
1dab4 # Example:
170b8aa3dd6d8aa3c284e9ea016e8653f7b51cb4b0431d8cbdba11508763f
3a3
  ignore_cert: 'false' # Set to 'true' to ignore the PCE
certificate
  log_level: '1' # Default log level is info
```



NOTE

If you are going to use a private PKI to sign the PCE certificate, see [Implement Kubelink with a Private PKI \[333\]](#) before deploying Kubelink.

3. Save the changes.
4. Create the Kubelink secret using the file.

```
oc create -f illumio-kubelink-secret.yml
```


Deploy Kubelink

Modify the Kubelink configuration file to point to the correct Docker image. The example in this document has **kubelink:<version#>** uploaded to **registry.example.com:443/illumio**, which means the image link in this example is **registry.example.com:443/illumio/kubelink:<version#>**

1. Edit the Kubelink configuration YAML file. For OpenShift clusters, the file name will be **illumio-kubelink-openshift.yml**.
 - Inside the YAML, you will find the **spec: > template: > spec: > containers:** section. Paste the image link in the **image:** attribute. The string should be wrapped in single quotes as shown in the example below.
2. Save the changes.

Below is a snippet from an example of the Kubelink configuration for OpenShift to illustrate the image location.

```

apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: illumio-kubelink
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: illumio-kubelink
  template:
    metadata:
      labels:
        app: illumio-kubelink
    spec:
      nodeSelector:
#       node-role.kubernetes.io/master: ""
      serviceAccountName: illumio-kubelink
      tolerations:
        - key: node-role.kubernetes.io/master
          effect: NoSchedule
      containers:
        - name: illumio-kubelink
          image: 'registry.example.com:443/illumio/illumio-
kubelink:<version#>'
          imagePullPolicy: Always
          env:
            - name: ILO_SERVER
              valueFrom:
                secretKeyRef:
                  name: illumio-kubelink-config
                  key: ilo_server

```

3. (Optional) If you're using a private PKI to sign the PCE certificate, ensure that you include the references to the root CA certificate that signed the PCE certificate. For more details, see [Implement Kubelink with a Private PKI \[333\]](#).
4. To deploy Kubelink, run the following command:

```
oc apply -f illumio-kubelink-openshift.yml
```

After Kubelink is successfully installed, you can check the cluster information by using the Illumio PCE web console. From the main menu, navigate to **Infrastructure > Container Clusters**.

Below is an example of a healthy container cluster state reported by Kubelink.

☰

<

Container Cluster – my-k8s-cluster-1

Summary

Container Workload Profiles

Workloads

Edit

— Remove

General

Name

my-k8s-cluster-1

Status

● In Sync

Platform Version

Openshift v3.9.65

Kubelink Version

test-master.75c678

Description

Implement Kubelink with a Private PKI

This section describes how to configure Kubelink with a PCE using a certificate signed by a private PKI. You can configure Kubelink to accept certificates from the PCE signed by a private root or intermediate Certificate Authority (CA) to ensure that Kubelink can communicate securely with the PCE.



NOTE

The steps described below do not apply to a PCE using a self-signed certificate.

Prerequisites

- Access to the root CA to download the root CA certificate.
- Access to your Kubernetes cluster and can run **kubectl** commands.
- Correct privileges in your Kubernetes cluster to create resources like configmaps, secrets, and pods.
- Access to the PCE UI as a Global Organization Owner.

Download the Root CA Certificate

Before you begin, ensure that you have access to the root CA certificate. The root CA certificate is a file that can be exported from the root CA without compromising the company's security. It is usually made available to external entities to ensure a proper SSL handshake between a server and its clients.

You can download the root CA certificate in CRT format to your local machine. Below is an example of a root CA certificate:

```
$ cat root.democa.illumio-demo.com.crt
-----BEGIN CERTIFICATE-----
MIIGSzCCBD0gAwIBAgIUAPw0NfPAivJW4YmKZ499eHZH3S8wDQYJKoZIhvcNAQEL
---output suppressed---
wPG0lug46K1EPQqMA7Yshmrw0d6ESy6RGNFFZdhk9Q==
-----END CERTIFICATE-----
```

You can also get the content of your root CA certificate in a readable output format by running the following command:

```

$ openssl x509 -text -noout -in ./root.democa.illumio-
demo.com.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:

fc:34:35:f3:c0:8a:f2:56:e1:89:8a:67:8f:7d:78:76:47:dd:2f
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=California, L=Sunnyvale, O=Illumio,
OU=Technical Marketing, CN=Illumio Demo Root CA 1/
emailAddress=tme-team@illumio.com
        Validity
            Not Before: Jan 20 00:05:36 2020 GMT
            Not After : Jan 17 00:05:36 2030 GMT
        Subject: C=US, ST=California, L=Sunnyvale, O=Illumio,
OU=Technical Marketing, CN=Illumio Demo Root CA 1/
emailAddress=tme-team@illumio.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:

00:c0:e5:48:7d:97:f8:5b:8c:ef:ac:16:a8:8c:aa:
68:b8:48:af:28:cd:17:8f:02:c8:82:e9:69:62:e2:
89:2b:be:bd:34:fc:e3:4d:3f:86:5e:d7:e6:89:34:
71:60:e6:54:61:ac:0f:26:1c:99:6f:80:89:3f:36:
b3:ad:78:d1:6c:3f:d7:23:1e:ea:51:14:48:74:c3:
e8:6e:a2:79:b1:60:4c:65:14:2a:f1:a0:97:6c:97:
50:43:67:07:b7:51:5d:2c:12:49:81:dc:01:c9:d1:
57:48:32:2e:87:a8:d2:c0:b9:f8:43:b2:58:10:af:
54:59:09:05:cb:3e:f0:d7:ef:70:cc:fc:53:48:ee:
a4:a4:61:f1:d7:5b:7c:a9:a8:92:dc:77:74:f4:4a:
c0:4a:90:71:0f:6d:9e:e7:4f:11:ab:a5:3d:cd:4b:
8b:79:fe:82:1b:16:27:94:8e:35:37:db:dd:b8:fe:

```

```

fa:6d:d9:be:57:f3:ca:f3:56:aa:be:c8:57:a1:a8:
c9:83:dd:5a:96:5a:6b:32:2d:5e:ae:da:fc:85:76:
bb:77:d5:c2:53:f3:5b:61:74:e7:f3:3e:4e:ad:10:
7d:4f:ff:90:69:7c:1c:41:2f:67:e4:13:5b:e6:3a:
a3:2f:93:61:3b:07:56:59:5a:d9:bc:34:4d:b3:54:
b5:c6:e5:0a:88:e9:62:7b:4b:85:d2:9e:4c:ee:0b:
0d:f4:72:b1:1b:44:04:93:cf:cc:bb:18:31:3a:d4:
83:4a:ff:15:42:2d:91:ca:d0:cb:36:d9:8d:62:c0:
41:59:1a:93:c7:27:79:08:94:b2:a2:50:3c:57:27:
33:af:f0:b6:92:44:49:c5:09:15:a7:43:2a:0f:a9:
02:61:b3:66:4f:c3:de:d3:63:1e:08:b1:23:ea:69:
90:db:e8:e9:1e:21:84:e0:56:e1:8e:a1:fa:3f:7a:
08:0f:54:0a:82:41:08:6b:6e:bb:cf:d6:5b:80:c6:
ea:0c:80:92:96:ab:95:5d:38:6d:4d:da:38:6b:42:
ef:7c:88:58:83:88:6d:da:28:62:62:1f:e5:a7:0d:
04:9f:0d:d9:52:39:46:ba:56:7c:1d:77:38:26:7c:
86:69:58:4d:b0:47:3a:e2:be:ee:1a:fc:4c:de:67:
f3:d5:fe:e6:27:a2:ef:26:86:19:5b:05:85:9c:4c:
02:24:76:58:42:1a:f8:e0:e0:ed:78:f2:8f:c8:5a:
20:a9:2d:0b:d4:01:fa:57:d4:6f:1c:0a:31:30:8c:
32:7f:b0:01:1e:fe:94:96:03:ee:01:d7:f4:4a:83:
f5:06:fa:60:43:15:05:9a:ca:88:59:5c:f5:13:09:
      82:69:7f
Exponent: 65537 (0x10001)

```

X509v3 extensions:

X509v3 Subject Key Identifier:

3D:3D:3D:61:E6:88:09:FE:34:0F:1D:5E:5E:52:72:71:C7:DE:15:92

X509v3 Authority Key Identifier:

keyid:3D:3D:3D:61:E6:88:09:FE:34:0F:1D:5E:5E:52:72:71:C7:DE:15:92

X509v3 Basic Constraints: critical

CA:TRUE

X509v3 Key Usage: critical

Digital Signature, Certificate Sign, CRL Sign

Signature Algorithm: sha256WithRSAEncryption

28:24:86:91:a6:4a:88:e4:8d:6b:fc:67:2a:68:08:67:35:e5:
a6:77:ff:07:4b:89:53:99:2e:6d:95:df:12:81:28:6a:8e:6f:
5a:98:95:5b:4a:21:ae:f0:20:a4:4e:06:b2:4e:5a:67:c1:6a:
06:f1:0f:c1:f7:7e:f2:e0:b3:9d:d8:54:26:6a:b2:1c:19:b8:
b5:5c:c7:03:6b:f7:70:9e:72:85:c9:29:55:f9:f4:a4:f2:b4:
3b:3d:ce:25:96:67:32:1e:8d:e2:00:22:55:4b:05:4f:ee:0e:
67:ac:db:1b:61:da:5f:9c:10:1c:0c:05:66:c0:5b:5f:b9:95:
59:a9:58:5b:e7:69:ac:b0:bd:b3:c2:a3:35:58:01:a4:ff:c0:
8d:ac:1c:19:21:41:50:fb:8e:e0:f5:a9:ad:ec:de:cb:53:04:
a9:d8:ac:76:8a:09:0d:7c:c6:1a:bc:06:74:bb:10:1c:aa:07:
f6:cb:b2:1b:0c:0c:65:03:45:2b:51:d5:6e:a0:4d:91:ce:c5:
ed:8d:a9:e7:f6:37:7d:ab:1b:a4:a2:a3:3b:76:17:5b:d9:3a:
9c:c1:df:cc:cd:a0:b0:a9:5c:74:61:d7:a0:1d:04:67:68:ee:
a6:7b:1e:41:a4:02:fc:65:9e:e3:c1:c2:57:b2:2e:b0:ff:a9:
86:82:35:4d:29:b2:fe:74:2e:b8:37:5d:2b:e8:69:f2:80:29:
19:f1:1e:7a:5d:e3:d2:51:50:46:30:54:7e:b8:ad:59:61:24:
45:a8:5a:fe:19:ff:09:31:d0:50:8b:e2:15:c0:a2:f1:20:95:
63:55:18:a7:a2:ad:16:25:c7:a3:d1:f2:e5:be:6d:c0:50:4b:
15:ac:e0:10:5e:f3:7b:90:9c:75:1a:6b:e3:fb:39:88:e4:e6:
9f:4c:85:60:67:e8:7d:2e:85:3d:87:ed:06:1d:13:0b:76:d7:
97:a5:b8:05:76:67:d6:41:06:c5:c0:7a:bd:f4:c6:5b:b2:fd:
23:6f:1f:57:2e:df:95:3f:26:a5:13:4d:6d:96:12:56:98:db:
2e:7d:fd:56:f5:71:b7:19:2b:c9:de:2d:b9:c8:17:cc:20:de:
7c:19:7a:aa:12:97:1c:80:b7:d3:67:d3:b7:a7:96:f0:c9:4d:
f5:8b:0e:10:3b:b9:4e:09:90:5a:3b:51:c9:48:a2:ca:9f:db:
72:44:87:59:db:49:fa:75:44:b5:f6:7f:c5:26:e1:01:ae:7b:
6f:4a:75:d1:b5:b3:68:c0:31:48:f8:5c:06:c0:f1:b4:96:e8:
38:e8:ad:44:3d:0a:8c:03:b6:2c:86:6a:f0:39:de:84:4b:2e:
91:18:d1:45:65:d8:64:f5

Create a configmap in the Kubernetes Cluster

After downloading the certificate locally on your machine, create a configmap in the Kubernetes cluster that will copy the root CA certificate on your local machine into the Kubernetes cluster. Only .crt and .pem files are supported.

To create a configmap, run the following command:

For .crt:

```
$ oc -n kube-system create configmap root-ca-config \
  --from-file=./certs/root.democa.illumio-demo.com.crt
```

For .pem:

```
$ oc -n kube-system create configmap root-ca-config \
  --from-file=./certs/root.democa.illumio-demo.com.pem
```

The **--from-file** option points to the path where the root CA certificate is stored on your local machine.

To verify that the configmap was created correctly, run the following command. Substitute .pem for .crt for the file extension if you use a .pem file:


```

$ oc -n kube-system create configmap root-ca-config \
> --from-file=./certs/root.democa.illumio-demo.com.crt
configmap/root-ca-config created
$
$ oc -n kube-system get configmap
NAME                                DATA    AGE
calico-config                      8        142d
cluster-info                      4        142d
coredns                           1        142d
coredns-autoscaler                1        142d
crn-info-ibmc                     6        142d
extension-apiserver-authentication 6        142d
iaas-subnet-config                1        142d
ibm-cloud-cluster-ingress-info    2        142d
ibm-cloud-provider-data           1        142d
ibm-cloud-provider-ingress-cm     6        142d
ibm-master-proxy-config           1        142d
ibm-network-interfaces            1        142d
kube-dns                          0        142d
kubernetes-dashboard-settings     1        44d
metrics-server-config             1        142d
node-local-dns                   1        142d
root-ca-config                    1        12s
subnet-config                     1        142d
$
$ oc -n kube-system describe configmap root-ca-config
Name:                             root-ca-config
Namespace:                       kube-system
Labels:                           <none>
Annotations:                      <none>

Data
====
root.democa.illumio-demo.com.crt:
----
-----BEGIN CERTIFICATE-----
MIIGSzCCBD0gAwIBAgIUAPw0NfPAivJW4YmKZ499eHZH3S8wDQYJKoZIhvcNAQEL
---output suppressed---
wPG0lug46K1EPQqMA7YshmrwOd6ESy6RGNFFZdhk9Q==
-----END CERTIFICATE-----

Events:                           <none>
$

```

`root-ca-config` is the name used to designate a configmap. You can modify it according to your naming convention.

Modify the Kubelink Manifest File to Use a Certificate

After creating the configmap in your Kubernetes cluster, modify the YAML file that describes Kubelink.

The current manifest file provided by Illumio does not include this modification by default. Open the .yaml file and add the following code blocks:

- `volumeMounts` (under `spec.template.spec.containers`)
- `volumes` (under `spec.template.spec`)

```
    volumeMounts:
      - name: root-ca
        mountPath: /etc/pki/tls/ilo_certs/
        readOnly: false
    volumes:
      - name: root-ca
        configMap:
          name: root-ca-config
```



NOTE

In a YAML file, indentation is important. Ensure that the indentation in the file matches the specified format.

`root-ca` is the name used to designate the new volume mounted in the container. You can modify it according to your naming convention.

After successfully modifying the manifest file, deploy Kubelink. For more details, see [Deploy Kubelink \[327\]](#).

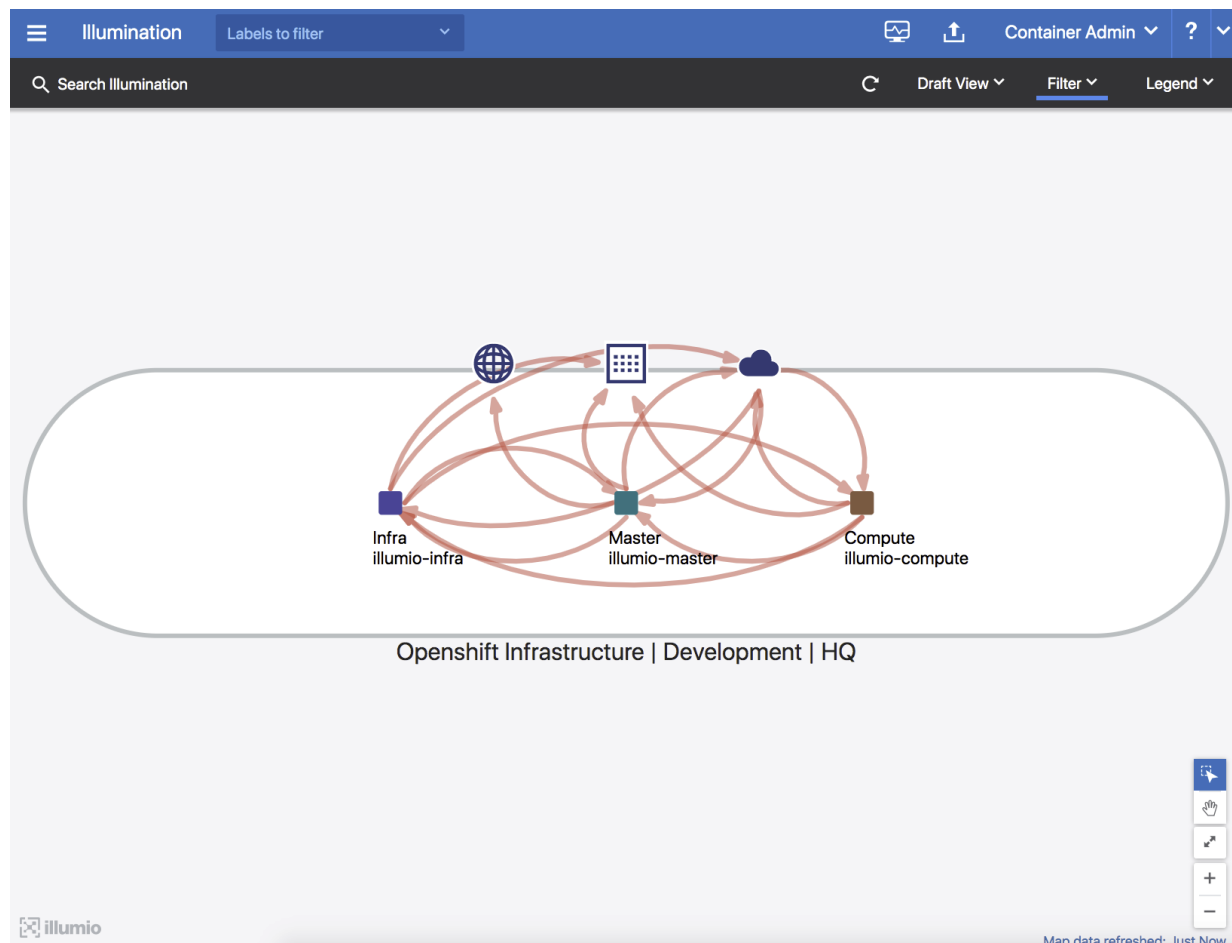
Install and Pair VENs for Containers

Using the pairing profiles mentioned earlier in this guide to install the VEN on each node of your OpenShift cluster. For more information about installing VENs, see the VEN Installation and Upgrade Guide.

Ensure that either of the two requirements below have been met prior to installing the VEN:

- Kubelink is deployed on the OpenShift cluster and in sync with the PCE
- Firewall coexistence is enabled

Below is a screenshot of Illumination with a master, compute, and infra node after deploying and pairing the Illumio VEN.



Manage OpenShift Namespaces

After activating the VENs on the OpenShift cluster nodes and Kubelink is in sync with the PCE, you can start managing the OpenShift projects (or namespaces). By default, all namespaces are unmanaged, which means Illumio Core does not apply any inbound or outbound controls to the pods within those namespaces. Any pods or services within unmanaged namespaces do not show up in the PCE inventory and Illumination.

After an Illumio Core PCE administrator changes an OpenShift namespace from unmanaged to managed, the pods and services will show up in Illumina-

tion and inherit the labels of each OpenShift namespace. The pods are represented in Illumio Core as Container Workloads. If there are frontend services, then Illumio Core represents each one as a Virtual Service.

The following section describes how to change a namespace from unmanaged to managed.

Container Workload Profiles

Log into the PCE Web Console.

1. From the PCE web console menu, choose **Infrastructure > Container Clusters**.
2. Select the **Container Cluster** you want to manage.
3. Select the **Container Workload Profiles** tab.
4. You will see a list of all namespaces in the cluster. Select the namespace you want to manage.
5. Click **Edit**:
 - a. Name is optional.
 - b. Select a Container Workload Policy State (anything other than unmanaged).
 - c. Assign Labels (optional).
 - d. Click **Save**.

When assigning labels, you can assign no labels, some labels, or all labels to the namespace. If there is a label which is not assigned, then you can insert annotations into the deployment configuration (or application configuration) to assign labels. If there is a conflict between a label assigned via the Container Workload Profile and the annotations in the deployment configuration, then the label from the Container Workload Profile will override the deployment configuration. Regardless of how you assign labels, it is not required for pods or services to have all labels in order for the PCE to manage them. Below are instructions on how to assign labels via the deployment configuration.

Using Annotations

For Deployment Configurations (Pods)

1. Open the OpenShift Web Console.
2. Navigate to the desired deployment/daemon set and click **Edit YAML**.
 - a. Inside the configuration YAML navigate to **spec: > template: > metadata: > annotations:**. If **annotations:** does not exist, then create an **annotations:** section underneath **metadata:**.
 - b. The following Illumio label key fields which can go under the the **annotations:** section.

- `com.illumio.role:`
 - `com.illumio.app:`
 - `com.illumio.env:`
 - `com.illumio.loc:`
- c. Fill in the appropriate labels.
 - d. Save the file and exit.

For Service Configurations (Services)

1. Open the OpenShift Web Console.
2. Navigate to the desired service and click **Edit YAML**.
 - a. Inside the configuration YAML navigate to `metadata: > annotations:`. If `annotations:` does not exist, then create an `annotations:` section underneath `metadata:`.
 - b. The following Illumio label key fields which can go under the the `annotations:` section.
 - `com.illumio.role:`
 - `com.illumio.app:`
 - `com.illumio.env:`
 - `com.illumio.loc:`
 - c. Fill in the appropriate labels.
 - d. Save the file and exit.

When using the annotations method, you may need to restart the pods or service after saving the changes to the YAML for the labels to get assigned.

Below are examples of pods and namespaces which use label assignments via either Container Workload Profiles or a mix of Container Workload Profiles plus annotation insertion.

This example changes unmanaged namespaces of Openshift infrastructure services (such as `apiserver`, `registry-console`, etc.) into managed namespaces.

Things to notice about the example shown below:

- There are Openshift infrastructure services, or control plane pods, that exist within namespaces like **default**, **kube-service-catalog**, etc. They will inherit all four R-A-E-L labels, including a Role label called "Control", from what has been configured in the Container Workload Profile(s). The Application, Environment, and Location labels are the same as the Openshift cluster nodes. This will minimize the complexity of writing policy which is mentioned later in this guide.
- The Kubelink pod exists in the `kube-system`. This pod will get the same application, environment, and location labels as the Openshift cluster nodes.

The role label is left blank and will be specified later using the annotations. These labels are assigned to the Kubelink pod through the Container Workload Profile associated to the kube-system namespace.

- There is a namespace called **openshift** which contains two different deployments or a two-tier shopping cart application (Web and Database). To achieve tier-to-tier segmentation across the application they would need different Role labels; therefore, a Role label will be inserted into the annotations of each deployment configuration.

Summary Container Workload Profiles Workloads Container Workloads Service Backends							
<i>i</i> Container Workload Profiles will be created automatically for discovered Projects using the following template: Set Policy State to "Unmanaged". Edit default settings.							
+ Add - Remove Refresh							
Select properties to filter view							
Customize columns 50 per page 1 - 12 of 12 Total							
☐	Name	Project	Policy State	Role	Application	Environment	Location
☐		default	Build				
☐		kube-public	Unmanaged				
☐		kube-service-catalog	Build				
☐		kube-system	Build				
☐		logging	Unmanaged				
☐		management-infra	Unmanaged				
☐		openshift	Build				
☐		openshift-ansible-service-broker	Unmanaged				
☐		openshift-infra	Unmanaged				
☐		openshift-node	Unmanaged				

Snippet of **illumio-kubelink** deployment configuration file shown here. Role label of "Kubelink" inserted under **spec: > template: > metadata: > annotations:** section.

illumio-kubelink-openshift.yml

```

apiVersion: apps/v2
kind: Deployment
metadata:
  name: illumio-kubelink
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: illumio-kubelink
  template:
    metadata:
      labels:
        app: illumio-kubelink
      annotations:
        com.illumio.role: Kubelink

```

Snippet of the Shopping-Cart Web deployment configuration file shown here. Role label of "Web" inserted under **spec: > template: > metadata: > annotations:** section.

shopping-cart-web.yml

```

spec:
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    name: shopping-cart-web
  strategy:
    activeDeadlineSeconds: 21600
    resources: {}
    rollingParams:
      intervalSeconds: 1
      maxSurge: 25%
      maxUnavailable: 25%
      timeoutSeconds: 600
      updatePeriodSeconds: 1
    type: Rolling
  template:
    metadata:
      annotations:
        com.illumio.role: Web
        openshift.io/generated-by: OpenShiftNewApp
      creationTimestamp: null
      labels:

```

Snippet of the Shopping-Cart Database deployment configuration file shown here. Role label of "Database" inserted under **spec: > template: > meta-data: > annotations:** section.

shopping-cart-db.yml

```
spec:
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    name: postgresql
  strategy:
    activeDeadlineSeconds: 21600
    recreateParams:
      timeoutSeconds: 600
    resources: {}
    type: Recreate
  template:
    metadata:
      annotations:
        com.illumio.role: Database
        openshift.io/generated-by: OpenShiftNewApp
      creationTimestamp: null
      labels:
```

Below is the final outcome of the label assignment from the example.

Policy State	Policy Sync	Namespace	Name	Role	Application	Environment	Location
Build	Active	default	registry-console-1-r85jf	Control	OpenShift Infrastructure	Development	HQ
Build	Active	kube-service-catalog	apiserver-bqf5g	Control	OpenShift Infrastructure	Development	HQ
Build	Active	kube-service-catalog	controller-manager-x4vb2	Control	OpenShift Infrastructure	Development	HQ
Build	Active	default	docker-registry-6-2jfcn	Control	OpenShift Infrastructure	Development	HQ
Build	Active	openshift-template-service-broker	apiserver-x8bx7	Control	OpenShift Infrastructure	Development	HQ
Build	Active	kube-system	illumio-kubelink-554688b759-k2mxt	Kubelink	OpenShift Infrastructure	Development	HQ
Build	Active	openshift	postgresql-1-2v99p	Database	ShoppingCart	Development	HQ
Build	Active	openshift	shopping-cart-web-1-shgbc	Web	ShoppingCart	Development	HQ
Build	Active	openshift	shopping-cart-web-1-ljq78	Web	ShoppingCart	Development	HQ
Build	Active	openshift	shopping-cart-web-1-ghv2t	Web	ShoppingCart	Development	HQ
Build	Active	openshift	postgresql-1-qbl6z	Database	ShoppingCart	Development	HQ

Daemonsets and Replicasets

The steps above apply only to services in OpenShift which are bound to **deployment** or **deploymentconfig**. This is due to the Kubelink's dependency on pod hash templates which daemonset and replicaset configurations do not have. If you discover pods derived from daemonset or replicaset configurations and also discover services bound to those pods, then Kubelink will **not** automatically bind the virtual service and service backends for the PCE. The absence of this binding will create limitations with Illumio policies written against the virtual service. To get around this limitation for daemonsets and replicasets follow the steps below.

1. Log into the CLI of any OpenShift node and generate a random uuid using the **uuidgen** command.
2. Copy the output of the **uuidgen** command.
3. In the OpenShift web console, navigate to the configuration of the daemonset or replicaset and edit the YAML file.
4. Find the **spec: > template: > metadata: > labels:** field in the YAML and create field called **pod-template-hash:** under the **labels:** section.
5. Paste the new uuid to the value of the **pod-template-hash:** field.
6. Save the changes.

Repeat steps 1 through 6 for each daemonset or replicaset configuration.

See screenshots below for DaemonSet or ReplicaSet reference.



```

[root@master ~]#
[root@master ~]#
[root@master ~]#
[root@master ~]# uuidgen
be85a690-613a-4b24-a7f7-5765befbe11d
[root@master ~]#
template:
  metadata:
    annotations:
      com.illumio.pairing_key: 4229fb4a718c628
    labels:
      app: nginx-webserver
      pod-template-hash: be85a690-613a-4b24-a7f7-5765befbe11d
  spec:
    containers:
      - name: webserver
        image: rstarnier/nginx-curl
        imagePullPolicy: IfNotPresent
        ports:

```

```
[root@master ~]# cat nginx-ds.yaml
apiVersion: extensions/v1
kind: DaemonSet
metadata:
  name: nginx-webserver
spec:
  template:
    metadata:
      annotations:
        com.illumio.pairing_key: 4229fb4a718c62861e11139749580112068b35394639954eac02a1395c87888e307d72676fc0
      labels:
        app: nginx-webserver
        pod-template-hash: be85a690-613a-4b24-a7f7-5765befbe11d
    spec:
      containers:
        - name: webserver
          image: rstarmer/nginx-curl
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 80
```



NEN Installation and Usage Guide

This guide describes how to install the Illumio Network Enforcement Node (NEN), configure Server Load Balancers (SLBs) and switches to work with it, and use the NEN to secure workloads attached to those network devices. It also describes how to configure data flows for non-default ports and protocols for NEN flow collection.

- [Overview of the NEN \[349\]](#)
- [NEN Installation and Configuration \[358\]](#)
- [NEN Integration with Load Balancers \[383\]](#)
- [NEN Integration with Switches \[399\]](#)

This guide also details how to set up the NEN for use in the following use cases:

- [Enhanced Network Security for Top of Rack Routers \[422\]](#)
- [Integrate NEN with NVIDIA Bluefield DPU featuring OVS \[423\]](#)
- [IBM i Integration \[426\]](#)

Introducing the Illumio Network Enforcement Node

This section provides an overview of how the NEN integrates with network devices and presents the new features across several releases.

Overview of the NEN



NOTE

In the NEN Guide, the term "switch" refers to both switches and routers.

This section describes the situations where installing an agent (the Illumio VEN) on a device is not possible and how to workaround it by using the NEN.

When Installing a VEN Isn't Possible

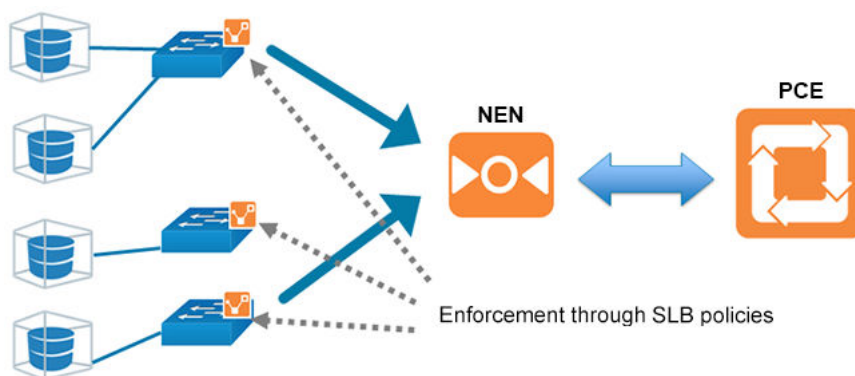
Visibility of communication across applications is critical for segmentation. The optimal method of getting visibility is to use a lightweight agent or a VEN, to report all inbound and outbound communications for each workload.

However, in certain cases a VEN cannot be installed on special purpose systems that provide services to application workloads; for example, IBM Mainframes, NetAPP Filers, legacy Windows machines, or appliances. In other cases, the VEN could be installed on a workload but customers choose not to; for example, installing a VEN might void the vendor's support agreement or the workload is sensitive to latency because it is a high transaction server.

How the NEN Integrates with Network Devices

In cases where a VEN cannot be installed, the NEN extends visualization capabilities to agentless workloads via the network. The NEN is installed as part of an Illumio Core deployment and paired with a PCE. Every IP address associated with the network endpoints managed by the NEN has one workload or virtual server associated with it. The NEN can manage multiple endpoints and enforce policy for those endpoints.

This guide describes how to integrate the NEN with supported load balancers (SLBs) and switches.



Using the NEN, Illumio Core enforces policy on the nearest point to the workload, either:

- A virtual server on a load balancer in front of the workload
- A switch port on a router in front of the workload

The NEN receives generic policy from PCE and generates policy appropriate to the managed network devices:

- **SLBs:** Firewall policy; for example, the F5 load balancer has two variants of applying policy: AFM and LTM
- **Switches:** ACLs

Until a NEN is paired to the PCE, the switch and load balancer features are deactivated. Using the PCE web console, Illumio users associate unmanaged

workloads to the network device endpoints. The NEN syncs its configuration with the PCE every 1 minute. For switch devices, the NEN can be configured to receive traffic flow information from the managed network devices and provide Illumination data to the PCE.

Currently, the NEN does not configure network devices automatically. Network device management has to be done by the user. This process includes applying generated policy by using the Illumio REST API. For information about applying policy to switches, see [Apply Policy for Switches \[416\]](#) and [NEN Switch Configuration Using REST API \[409\]](#). For information about applying policy for SLBs, see [Write SLB Policy \[393\]](#).

IP Lists, FQDNs, and the NEN

The PCE does not calculate IP lists that include FQDNs in the policies it delivers through the NEN. If the PCE detects FQDNs in the IP list, the entire IP list is omitted from policy, not just the FQDN entries.

What's New in NEN 2.7.x Releases

This topic describes the new features introduced in the following NEN releases.

NEN 2.7.10 New Features

- **Client Auth EKU no longer required in TLS certificates**

Prior to release 2.7.10, the NEN relied on TLS certificates with the Client Auth EKU for some internal services. Certificates issued without this EKU could've resulted in operational disruptions. However, with the release of NEN 2.7.10, Client Auth EKU is no longer required.

- **Support for Postgres 16.11**

NEN 2.7.10 adds support for Postgres 16.11.

- **Discontinued dependency on net-tools**

Beginning with release 2.7.10, Illumio has discontinued dependency on net-tools. The **iproute(2)** suite of tools, especially the **ip** command, replaces the deprecated net-tools package.

- **Support for Non-default Ports and Protocols for NEN Flow Collection**

Beginning with release 2.7.10, the NEN supports configurable flow-transport settings. Instead of using UDP and fixed destination ports, you can now configure TCP for flow collection and send flows to ports you define. This provides greater flexibility and support for diverse network and security requirements. For details, see [Configure Non-default Ports and Protocols for NEN Flow Collection](#).

NEN 2.7.0 New Features

This release adds support for the following:

- **Top-of-rack Cisco IOS XR series routers**

This release supports integrating the NEN with Cisco IOS series routers. (Illumio Core PCE 25.3.0 or later, SaaS only.)

- **Support for CIDR block interfaces**

Allows you to assign CIDR blocks to unmanaged workloads. Each unmanaged workload can represent a subnet, a Layer 3 interface, or a group of workloads instead of just a single workload. (Illumio Core PCE 25.3.0 or later, SaaS only.) See [Enhance network security for Top Of Rack routers using Illumio NEN 2.7.0 and Cisco IOS XR](#).

- **Support for NVIDIA BlueField DPU (with OVS)**

OVS is a software-based network technology that enhances virtual machine (VM) communication within internal and external networks. It functions as a virtual switch, allowing VMs to communicate within a host and across different hosts. Typically installed on a NIC (for example NVIDIA's BlueField-3 Data Processing Unit; support for other cards may also be available), OVS' software-based approach for packet switching relieves the strain on CPU resources that can impact system performance and network bandwidth. See [Integrate the NEN with the NVIDIA BlueField®-3 DPU featuring OVS](#).

- **Illumio NEN + OVS Use Case**

Integrating the NEN with OVS enables visibility and policy enforcement for traffic within and between IT and OT layers, allowing you to visualize all traffic to and from OT systems. Illumio's flexible labeling architecture helps you understand how your assets communicate. The NEN converts your segmentation policies into ACLs that are then installed on the OVS to secure your OT/IT infrastructure.

- **Streamlined integration through the Illumio API**

Integrating the NEN with OVS through the PCE web console is straightforward enough, but integration through the PCE API is even easier: enter the IP address and credentials for the OVS switch (see note below) and the NEN automatically discovers the switch configuration, programs flow monitoring on the switch, discovers and creates workloads in the PCE, and programs the ACLs on the OVS.

- **Support for NetFlow and IPFIX flow data monitoring protocols**

These protocols are added to the NEN's existing support for sFlow.

- **Support for IPv6 Access Control Lists (ACLs)**

Provided in addition to existing support for IPv4.

What's New in NEN 2.6.x Releases

This section describes new features introduced in the following NEN releases.

NEN 2.6.40 New Feature

JSON Format Change

Beginning with this release, generic workload JSON files are uploaded as a single, parseable object. This new format allows a program to use the JSON file to apply policy to a device customers want to protect.

```

1 [
2   {
3     "$schema": "http://json-schema.org/draft-04/schema#",
4     "definitions": {
5       "rules": {
6         "description": "Array of rule objects",
7         "type": "array",
8         "items": {
9           "description": "A single rule",
10          "type": "object",
11          "required": ["action", "port", "protocol", "ips"],
12          "properties": {
13            "action": {
14              "description": "Action for the rule either permit or deny",
15              "type": "string",
16              "enum": ["permit", "deny"]
17            },
18            "port": {
19              "description": "Inbound or Outbound port(s) bound to rule. Either a port, port range or *",
20              "type": "string"
21            },
22            "protocol": {
23              "description": "Protocol for rule. Either a protocol number or *",
24              "type": "string"
25            },
26            "ips": {
27              "description": "An array of inbound or outbound IP addresses bound to rule",
28              "type": "array",
29              "items": {
30                "description": "IP address associated to rule. Either IP address, CIDR block, IP address range or *",
31                "type": "string"
32              }
33            }
34          }
35        }
36      },
37      "description": "An array of objects, one per workload",
38      "type": "array",
39      "items": {
40        "type": "object",
41        "required": ["name", "href", "rules"],
42        "properties": {
43          "name": {
44            "description": "Name of workload",
45            "type": "string"
46          },
47          "href": {
48            "description": "href of workload",
49            "type": "string"
50          },
51          "rules": {
52            "description": "Object containing Inbound and Outbound rules",
53            "type": "object",
54            "properties": {
55              "Inbound": {
56                "description": "Array of Inbound rule objects",
57                "$ref": "#/definitions/rules"
58              },
59              "Outbound": {
60                "description": "Array of Outbound rule objects",
61                "$ref": "#/definitions/rules"
62              }
63            }
64          }
65        }
66      }
67    }
68  ]
69 ]
70

```


NEN 2.6.30 New Features



IMPORTANT

Before installing NEN release 2.6.30

Installing this release upgrades the existing database on the NEN to a newer version of the database software. Illumio recommends that you back up the existing NEN database before you install NEN 2.6.30 so that you can revert the installation if necessary.

To back up the existing NEN database, issue the following commands on the NEN primary node:

```
illumio-nen-ctl set-runlevel 1 -svw
```

```
illumio-nen-db-management dump --file <outputfile-name>
```

```
illumio-nen-ctl stop
```

Support for CentOS Stream 9

This release includes support for installing NENs on nodes running CentOS Stream 9.

Switch ACL generation now supports all protocols

With this release, the NEN now recognizes all PCE-supported protocols, ensuring that the NEN can translate switch policy into ACLs when such policy references any PCE-supported protocol.

Support for VMware NSX Advanced Load Balancer AVI 22.1.6

With this release, the NEN now supports VMware NSX Advanced Load Balancer AVI version 22.1.6.

NEN 2.6.20 New Features

Support for RHEL 9

This release includes support for running standalone NENs on Red Hat Enterprise Linux (RHEL) 9 where the version of **openssl-lib** is **3.1 or earlier**.

To determine the openssl-lib version, issue `rpm -qa | grep openssl-lib`.

NEN 2.6.10 New Features

Support for Verifying NEN RPM Signature

Beginning with NEN release 2.6.10, you can verify the signature of the NEN RPM package before installation. This allows you to ensure that the package hasn't been modified since it was signed. For details, see [Verify the NEN RPM digital signature](#).

Support for NEN Proxy Communication

Beginning with NEN release 2.6.10, there is now `runtime_env` support for defining an HTTP/HTTPS proxy for communication between the NEN and the PCE or between the NEN and managed devices (such as Server Load Balancers (SLBs)). You can also specify a list of IP address that are not allowed to communicate via a proxy server. For details, see [Configure Proxy Support for NENs](#).

Ruby updated to version 3.1.2

Ruby was upgraded from version 2.7.1 to 3.1.2.

NEN 2.6.1 New Features

Support for all Citrix ADC (Netscaler) Load Balancer-supported protocols

With this release, the NEN now supports all the protocols that Citrix (NetScaler) 13.1 lists in the **Load Balancing > Virtual Servers > Add > Protocol** menu.

NEN 2.6.0 New Features

Support for Citrix ADC (Netscaler) Load Balancer

With this release, the NEN now supports Citrix ADC (Netscaler) Load Balancers and their associated virtual servers that have only a single IPv4 address.

To add a Citrix Software Load Balancer, see the section *Configure Load Balancers* in the "Load Balancers and Virtual Servers for the NEN" topic.

Support for allowing customers to specify whether disabled VIPs are reported to the PCE

Prior to the release of NEN 2.6.0, if VIP filtering was disabled, all VIPs – including disabled VIPs – were reported to the PCE. You can now disable this reporting using the following new option in the `illumio-nen-ctl slb-enable` command:

```
--disabled-virtual-server-reporting enabled|disabled
```

To ensure backwards compatibility, the default value is **enabled**.

PCE-provided rule IP addresses and ports now combined into CIDR blocks

NENs now combine rule IP addresses and ports provided by the PCE into CIDR blocks and port ranges. This reduces the number of ACLs that NENs need to generate for switches.

Benefits include:

- Fewer ACLs that the NEN generates for switches.
- Fewer ACLs generated for the IBM iSeries integration with Precisely (current limit: 10k ACLs) allows for optimization of IP addresses into ranges larger than can be covered by a single CIDR block.
- Lower demand on switch TCAM where ACLs are stored.

Support for Rocky Linux 8.7

This release includes support for running standalone NENs on Rocky Linux 8.7.

Support for configuring a PCE policy request timeout

Beginning with NEN 2.5.2.A1, you can configure a PCE policy request timeout. This may be needed if your NEN SLB implementation will involve large policy calculations. The timeout ensures that the NEN doesn't wait too long for the PCE to respond to policy requests in scenarios involving large policy calculations.

To configure the timeout, use the following runtime environment variable:

`pce_policy_request_timeout_minutes`

- Default value: 10 minutes
- Minimum value: 3 minutes

NEN Installation and Configuration

This section provides an overview of supported NEN architectures and describes how to install the NEN RPM package standalone hosts as well as upgrade the NEN to the latest version.

About NEN Installation and Architecture

This topic explains how the NEN is installed and the supported architectures.

PCE-based versus Standalone NEN Installation



IMPORTANT

Beginning in NEN 2.3, the NEN is deployed as a standalone NEN installation only. New PCE-based installations are not supported.

In NEN 2.1.x, two types of NEN installations were supported:

- **PCE-based installation**

You installed the NEN on one of the PCE data nodes so that the NEN ran as a service on the PCE. When you installed the NEN as a service on a PCE data node, you had the option of installing it on both data nodes (data node 0 and data node 1) so that the NEN operated as a high availability (HA) pair.

- **Standalone NEN installation**

You installed the NEN on a separate Linux host. When you installed a standalone NEN in NEN 2.1.x, you did not have the option to configure the NEN deployment as an HA pair.

Beginning in NEN 2.3, you must install the NEN on a separate Linux host (standalone installation). Installing the NEN on a PCE data node isn't supported beginning with NEN 2.3. The new standalone installation has the following benefits:

- Provides full (optional) HA support for Illumio On-Premises customers and Illumio Cloud customers.
- Allows you to deploy NENs closer to your network devices, namely load balancers and switches.
- Supports higher scale with multiple NEN HA pairs paired to a single PCE cluster.

**IMPORTANT**

Because NEN releases from 2.3 and later don't support a PCE-based installation, customers with existing installations (NEN 1.0.1 through NEN 2.1.0) must upgrade to NEN 2.3 or later. For information, see [Upgrade Standalone NEN 2.1.0 to Standalone NEN 2.3.x or Later \[374\]](#).

NEN High Availability Support

Prior to NEN 2.1.0, when NENs had to be installed on a PCE data node, High Availability (HA) on NENs was achieved by using the PCE's HA capabilities. Beginning with the move to a standalone NEN installation in NEN 2.2.0, the NEN now features full HA support independently of the PCE.

The following diagram illustrates how to plan your NEN installation to provide full HA support by installing it on two Linux hosts (node 1 and node 2). In an HA configuration, the primary NEN performs the following actions:

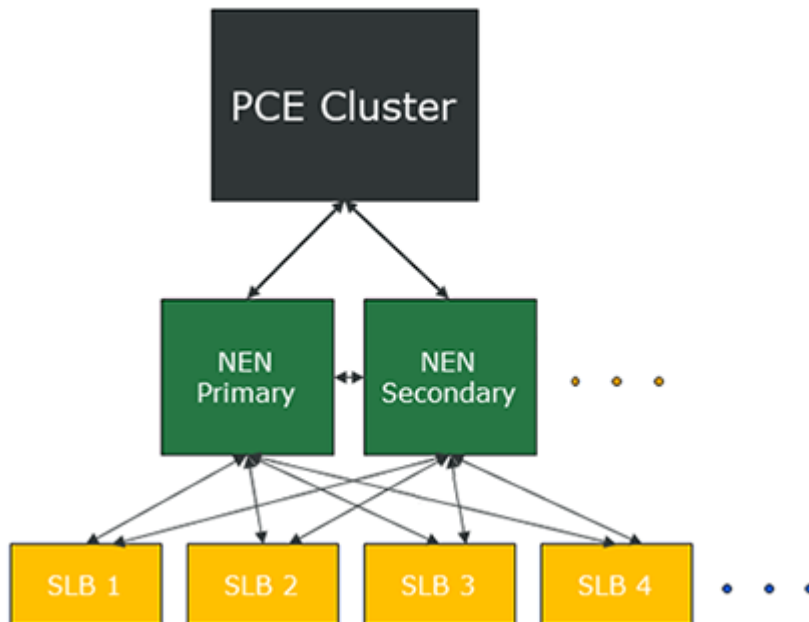
- Retrieves configuration information from the PCE and reconciles it with the PCE database.
- Determines what work needs to go into the work queue for the NEN HA pair.

If the primary NEN (on node 1) loses connectivity to the PCE, the secondary NEN (on node 2) becomes the primary NEN until the NEN on node 1 re-establishes connectivity with the PCE.

**NOTE**

For hardware requirements in an HA Pair implementation, see [CPU, Memory, and Storage Requirements \[363\]](#) in this topic.

When using the NEN for SLB integration, both NENs (primary and secondary) can program any load balancer because they share the work queue. Either NEN can accept the next job from the work queue depending on their available capacity. This capability is available when the primary NEN has connectivity with the PCE.



A PCE cluster supports multiple NENs per PCE, which can consist of multiple single node NENs, multiple NEN HA pairs, or a combination of both.

NEN Supercluster Support

In NEN 2.1.x (when installed as part of Illumio Segmentation for Data Centers 20.2.0, 21.1.0, or 21.2.x), Illumio provided limited support for the NEN with PCE Supercluster deployments. NEN releases prior to 2.1.0 did not include Supercluster support.

NEN 2.3.10 extended support for installing a NEN within a PCE Supercluster as follows:

- **NEN Installation on Supercluster Members**

You can pair the NEN to the other regions in the Supercluster; referred to as Supercluster “members.” Prior to NEN 2.3.10, you could only install the NEN on the Supercluster leader. For more information about PCE Supercluster deployment architecture, see “Design Supercluster Deployment” in the PCE Supercluster Deployment Guide.

**CAUTION**

Plan your NEN installation carefully when you install it as part of a PCE Supercluster deployment. Once installed, you cannot move NENs from one PCE Supercluster member to another member.

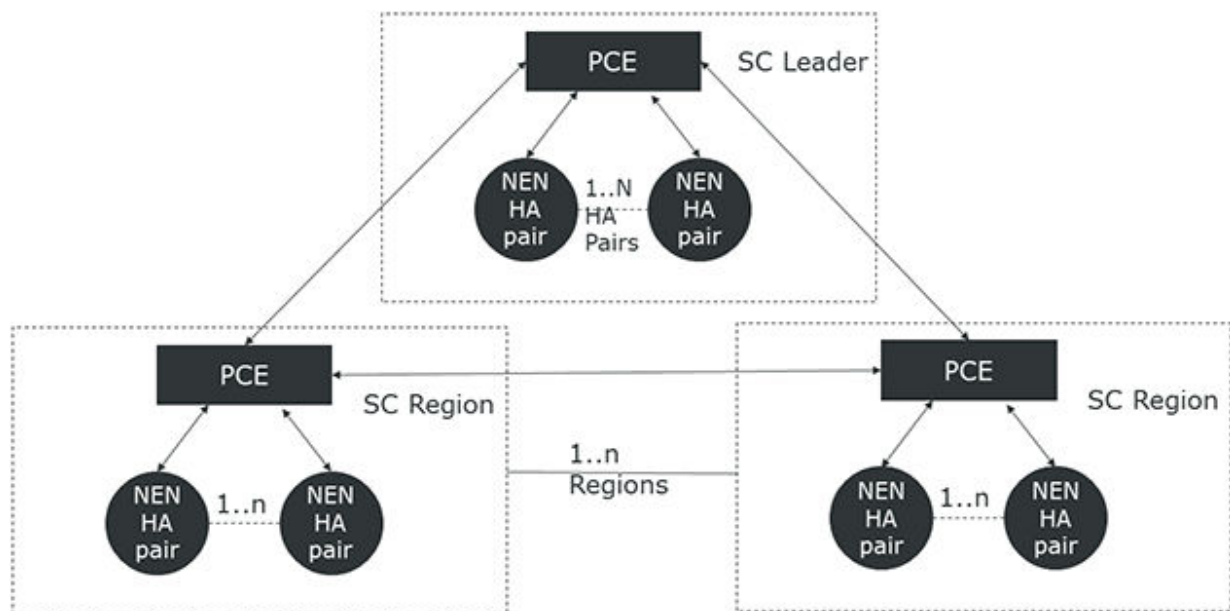
- **Multiple NEN HA Pairs in a Supercluster Member**

Depending on your scale requirements and the location of your network devices (such as SLBs), you can connect multiple NEN HA pairs to any cluster in a PCE Supercluster deployment (not just the PCE Supercluster leader). This enhancement is necessary to support environments with large numbers of SLBs and virtual servers that are geographically distributed.

**NOTE**

At a minimum, you must install a primary and secondary NEN HA pair in one of the Supercluster regions.

The following diagram illustrates how to plan your NEN installation in a PCE Supercluster deployment:



Workload JSON File Upload

Beginning in NEN release 2.6.40, generic workload JSON files are uploaded as a single, parseable object. This format allows a program to use the JSON file to apply policy to a device customers want to protect.

```

1 [
2   {
3     "$schema": "http://json-schema.org/draft-04/schema#",
4     "definitions": {
5       "rules": {
6         "description": "Array of rule objects",
7         "type": "array",
8         "items": {
9           "description": "A single rule",
10          "type": "object",
11          "required": ["action", "port", "protocol", "ips"],
12          "properties": {
13            "action": {
14              "description": "Action for the rule either permit or deny",
15              "type": "string",
16              "enum": ["permit", "deny"]
17            },
18            "port": {
19              "description": "Inbound or Outbound port(s) bound to rule. Either a port, port range or *",
20              "type": "string"
21            },
22            "protocol": {
23              "description": "Protocol for rule. Either a protocol number or *",
24              "type": "string"
25            },
26            "ips": {
27              "description": "An array of inbound or outbound IP addresses bound to rule",
28              "type": "array",
29              "items": {
30                "description": "IP address associated to rule. Either IP address, CIDR block, IP address range or *",
31                "type": "string"
32              }
33            }
34          }
35        }
36      },
37      "description": "An array of objects, one per workload",
38      "type": "array",
39      "items": {
40        "type": "object",
41        "required": ["name", "href", "rules"],
42        "properties": {
43          "name": {
44            "description": "Name of workload",
45            "type": "string"
46          },
47          "href": {
48            "description": "href of workload",
49            "type": "string"
50          },
51          "rules": {
52            "description": "Object containing Inbound and Outbound rules",
53            "type": "object",
54            "properties": {
55              "Inbound": {
56                "description": "Array of Inbound rule objects",
57                "$ref": "#/definitions/rules"
58              },
59              "Outbound": {
60                "description": "Array of Outbound rule objects",
61                "$ref": "#/definitions/rules"
62              }
63            }
64          }
65        }
66      }
67    }
68  ]
69 ]
70

```


CPU, Memory, and Storage Requirements

This section presents hardware requirements for supporting SLBs and switches.

Hardware requirements to support SLBs and VIPs

To install NEN(s) to support a given number of server load balancers and Virtual IPs, your hardware must meet the hardware requirements detailed in this section.

Server Load Balancers (SLBs)	Virtual IPs (VIPs)	Cores/Clock Speed ¹	RAM per Node ²	Storage Device Size ³ and IOPS ⁴	Network
Up to 6 SLBs	• Max 1,000 VIPs per SLB • Max 3,000 VIPs across all SLBs	• 2 cores • Intel® Xeon(R) CPU E5-2695 v4 at 2.10GHz or equivalent	8 GB	A single node including both core and data: • 1 x 50 GB • 100 IOPS per device	1 Gb Ethernet
Up to 50 SLBs	• Max 1,000 VIPs per SLB • Max 12,000 VIPs across all SLBs	• 4 cores • Intel® Xeon(R) CPU E5-2695 v4 at 2.10GHz or equivalent	16 GB	A single node including both core and data: • 1 x 50 GB • 100 IOPS per device	1 Gb Ethernet
NEN HA implementation considerations: The two nodes in an HA pair must match the size specified in this table. This is necessary because in the event of a failover, a single node must be able to manage the entire load. • For an HA Pair for up to 3000 VIPs, use the sizing detailed in the first row. • For an HA pair for 3000+ VIPs, use the sizing detailed in the second row.					

Footnotes:

¹ CPUs:

- The recommended number of cores is based only on physical cores from allocated CPUs, irrespective of hyper-threading or virtual cores. For example, in AWS one vCPU is only a single hyper-thread running on a physical core, which is half a core. 16 physical cores equates to 32 vCPUs in AWS.
- Full reservations for vCPU. No overcommit.

² Full reservations for vRAM. No overcommit.

³ Additional disk notes:

- Storage requirements for network traffic data can increase rapidly as the amount of network traffic increases. Allocating a separate, large storage device for traffic data can accommodate these rapid changes without potentially interrupting the service.
- Network File Systems (NFS) is not supported.

⁴ Input/output operations per second (IOPS) are based on 8K random write operations. IOPS specified for an average of 300 flow summaries (80% unique src_ip, dest_ip, dest_port, proto) per workload every 10 minutes. Different traffic profiles might require higher IOPS.

Hardware requirements to support switches

To install NEN(s) to support a given number of switches, your hardware must meet the hardware requirements detailed in this table.

Switches	Cores/Clock Speed ¹	RAM per Node ²	Storage Device Size ³ and IOPS ⁴	Network
Up to 30 switches	• 2 cores • Intel® Xeon(R) CPU E5-2695 v4 at 2.10GHz or equivalent	8 GB	A single node including both core and data: • 1 x 50 GB • 100 IOPS per device	1 Gb Ethernet
More than 30 switches	• 4 cores • Intel® Xeon(R) CPU E5-2695 v4 at 2.10GHz or equivalent	16 GB	A single node including both core and data: • 1 x 50 GB • 100 IOPS per device	1 Gb Ethernet

Footnotes:

¹ CPUs:

- The recommended number of cores is based only on physical cores from allocated CPUs, irrespective of hyper-threading or virtual cores. For example, in AWS one vCPU is only a single hyper-thread running on a physical core, which is half a core. 16 physical cores equates to 32 vCPUs in AWS.
- Full reservations for vCPU. No overcommit.

² Full reservations for vRAM. No overcommit.

³ Additional disk notes:

- Storage requirements for network traffic data can increase rapidly as the amount of network traffic increases. Allocating a separate, large storage device for traffic data can accommodate these rapid changes without potentially interrupting the service.
- Network File Systems (NFS) is not supported.

⁴ Input/output operations per second (IOPS) are based on 8K random write operations. IOPS specified for an average of 300 flow summaries (80% unique src_ip, dest_ip, dest_port, proto) per workload every 10 minutes. Different traffic profiles might require higher IOPS.

Machine Resource Requirements for NEN VMs

Storage Device	Partition mount point	Size to Allocate
Device 1, Partition A	/	8 GB
Device 1, Partition B	/var/log	16 GB ¹
Device 1, Partition C	/var/lib/illumio-nen	Balance of Device ¹

Footnote:

¹ The size of this partition assumes that NEN application logs and system logs are both stored in /var/log/illumio-nen.

Install and Activate the NEN



IMPORTANT

Before installing NEN release 2.6.30

Installing this release upgrades the existing database on the NEN to a newer version of the database software. Illumio recommends that you back up the existing NEN database before you install NEN 2.6.30 so that you can revert the installation if necessary.

To back up the existing NEN database, issue the following commands on the NEN primary node:

```
illumio-nen-ctl set-runlevel 1 -svw
```

```
illumio-nen-db-management dump --file <outputfile-name>
```

```
illumio-nen-ctl stop
```

This section describes how to:

- Install and activate a new standalone NEN deployment
- Upgrade a PCE-based NEN installation to the standalone NEN installation required for NEN 2.3.x and later.

Illumio recommends that you have the following knowledge before installing and administering the NEN:

- A thorough understanding of our organization's security goals.
- A thorough understanding of Illumio Core.
- When integrating the NEN with your organization's load balancers and switches, know how to configure and manage these network devices.

NEN Software

For the complete list of OS support for the NEN, see [NEN OS Support and Package Dependencies](#) on the Illumio Support portal.

To download the NEN software:

1. Log into the Illumio Support portal and go to **Software > NEN**.
2. From the **Download NEN Software** page, select the latest version.
3. Click the filename in the table to download the software locally.

Optional Configurations

Consider configuring the following optional functionality when you install NEN software.

Verify the NEN RPM digital signature

You can verify the signature of the NEN RPM package before installation to ensure that the package hasn't been modified since it was signed.

1. Download the NEN RPM.
 - a. Go to [Illumio Support software download page](#).
 - b. Select the NEN version you want to verify.
 - c. Click the RPM package you plan to install.
2. Import the Illumio NEN Public Key.

```
% gpg --import illumio_nen_pub.key
```

The imported file is placed in your `/home` directory.

3. List the keys in the imported file.

```
% gpg --list-keys illumio
```

In the output, locate the last 16 digits in the signature line.

```
pub rsaXXXX 2022-06-31 [SC]
```

```
8C34J70E2D13F9332AD1F49Dxxxxxxxxxxxxxyyy! Last 16 digits of the  
public key
```

4. List signatures in the RPM

```
% rpm -qpi illumio-nen-xxx-x.xx.x86_64.rpm |  
grep ^Signature
```

where `xxx-x.xx` is the version number of the package.

5. Visually compare the last 16 digits in the RPM with the last 16 digits in the imported Public Key.

Signature : RSA/SHA256, Key ID xxxxxxxxxxxxyyyy! Last 16 digits of the RPM.

If the signatures don't match, don't install the package. Contact Illumio Support.

Configure Proxy Support for NENs

Beginning with NEN release 2.6.10, you can configure proxy support for NENs by adding environment variables to the `runtime_env` file. This support defines an HTTP/HTTPS proxy for communication between the NEN and the PCE or between the NEN and managed devices (such as Server Load Balancers (SLB)). There's also support for specifying a list of IP address that are not allowed to communicate via a proxy server. You can configure these options by adding a field to the `runtime_env.yml` file.

Modify the template `runtime_env` file



NOTE

The NEN will honor the environment variables `http_proxy`, `https_proxy`, and `no_proxy` if they are present. However, you can override these variable values by setting appropriate values in the `proxy_config` variables in the NEN `runtime_env.yml` file.

The NEN will honor the environment variables `http_proxy`, `https_proxy`, and `no_proxy` if they are present. However, you can override these variable values by setting appropriate values in the `proxy_config` variables in the NEN `runtime_env.yml` file.

You can modify the `runtime_env.yml` file either during an interactive installation or later by copying and modifying the template runtime file.

- Modify during an interactive installation
- Modify post-installation.

For details, see [Install a New Standalone NEN](#).

Configuration scenarios

Under the `proxy_config` option, configure proxy support for any of the following scenarios in the `runtime_env.yml` file.

PCE	Managed Devices (SLBs)	Scenario	Proxy Environment Variable
The PCE is proxied.	No SLBs are installed.	Configure the NEN to communicate with the proxied PCE.	<code>pce_https_proxy:</code>
The PCE is proxied.	SLBs are installed but not proxied.		
The PCE is proxied.	SLBs are installed and proxied.	Configure the NEN to communicate with the proxied PCE and proxied SLBs.	<code>pce_https_proxy:</code> and <code>device_http_proxy:</code> or <code>device_https_proxy:</code>
The PCE is not proxied.		Configure the NEN to communicate with proxied SLBs.	<code>device_http_proxy:</code>
N/A		Specify a list of IP address that are not allowed to communicate via a proxy server.	<code>no_proxy:</code>

Configure a PCE policy request timeout

Beginning with NEN 2.5.2.A1, you can configure a PCE policy request timeout. This may be needed if your NEN SLB implementation will involve large policy calculations. The timeout ensures that the NEN doesn't wait too long for the PCE to respond to policy requests in scenarios involving large policy calculations.

To configure the timeout, use the following runtime environment variable:

`pce_policy_request_timeout_minutes`

- Default value: 10 minutes
- Minimum value: 3 minutes

Configure a PCE connect timeout

Beginning with NEN 2.6.1, you can configure a PCE policy connect timeout. This may be necessary because the shortness of the default connect timeout in the CURL library (5 minutes) may make the NEN susceptible to timing out when trying to connect to the PCE. This in turn will prevent the NEN from updating policy on the SLB.

To configure the timeout, use the following runtime environment variable:

`pce_policy_connect_timeout_minutes`

- Default value: 10 minutes
- Minimum value: 3 minutes

Install a New Standalone NEN



NOTE

This procedure describes how to perform a **new** NEN standalone installation where you have **not** previously installed the NEN as a service on a PCE data node or you have not installed the NEN 2.1.0 standalone service on your own host.

- For the steps to upgrade standalone NEN 2.1.0 to standalone NEN 2.3.x or later, see [Upgrade Standalone NEN 2.1.0 to Standalone NEN 2.3.x or Later \[374\]](#).

To install a NEN as a standalone NEN:



NOTE

For standalone NEN hardware requirements, see [CPU, Memory, and Storage Requirements \[363\]](#).

1. Download the NEN software from the Illumio Support portal.
2. Run the following command to install the NEN RPM on the host:

```
sudo yum install -y <path_to_Illumio_NEN_rpm>/illumio-nen-  
<release_number>  
-<build_number>.x86_64.rpm
```

3. Configure the NEN runtime environment settings using **one** of the following options:
 - **Option 1: Perform an interactive installation**

Run the **setup** command to launch an interactive installation and answer the prompts to configure the NEN runtime environment. (This method creates the NEN runtime environment file and saves it in the correct NEN directory.)

```
sudo /opt/illumio-nen/illumio-nen-env setupf
```

- **Option 2: Modify the template runtime environment file**

Copy a template of the NEN runtime environment file to the required location and then modify that file.

a. Copy from:

```
/opt/illumio-nen/illumio/config/templates
```

b. Paste to:

```
/etc/illumio-nen/runtime_env.yml
```

c. Update the file with the host FQDNs and service discovery certificate information.



IMPORTANT

Take careful note of the following:

- A standalone NEN cannot communicate with the PCE by using a self-signed service discovery certificate. The NEN requires an X.509 public certificate in PEM format for TLS communication with the PCE.
- When modifying the template runtime environment file (Option 2), you create your own value. However, if you are using the interactive NEN installation method (Option 1), the NEN CTL **setup** command automatically creates this value in the file.
- If you are implementing an HA cluster on NENs, specify **2node_v0** as the cluster type instead of **snc0**.

```
# Configuration generated <timestamp>
install_root: "/opt/illumio-nen"
runtime_data_root: "/var/lib/illumio-nen/runtime"
persistent_data_root: "/var/lib/illumio-nen/data"
ephemeral_data_root: "/var/lib/illumio-nen/tmp"
log_dir: "/var/log/illumio-nen"
private_key_cache_dir: "/var/lib/illumio-nen/keys"
nen_fqdn: <example.com>
service_discovery_fqdn: <example.com>
cluster_type: snc0
service_discovery_private_key: "/var/lib/illumio-nen/cert/server.key"
service_discovery_certificate: "/var/lib/illumio-nen/cert/server.crt"
service_discovery_encryption_key: <key>
```

Where:

- **nen_fqdn** is the hostname of the node where the NEN is installed.
- **service_discovery_fqdn** is the hostname of the NEN FQDN.
- **service_discovery_private_key** is the directory path of the RSA private key file.
- **service_discovery_certificate** is the directory path of the certificate file.
- **service_discovery_encryption_key** is a 16 byte hexadecimal base-64 encoded value



NOTE

Beginning in NEN release 2.4.10, you can add a field to the runtime environment file to configure how frequently the NEN polls Server Load Balancers (SLBs) to discover new virtual servers (VS). For details, see [Load Balancers and Virtual Servers for the NEN \[383\]](#).

4. Start the NEN and set the runlevel to 5. The option **-svw** shows the status of the start operation.

```
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl start --runlevel 5 -svw
```

NEXT STEPS

1. Activate the NEN with a pairing key from the PCE. See [Obtain Pairing Key and Activate the NEN \[373\]](#).

2. To enable the NEN to integrate with a load balancer, see [Enable Load Balancer Support \[373\]](#).
3. (Optional) To configure the NEN as an HA pair, perform the steps in [Configure HA Support for the NEN \[374\]](#).

Obtain a Pairing Key and Activate the NEN

When the NEN is installed as part of a NEN HA pair, you only pair the NEN primary node with the PCE.

1. Log into the PCE web console.
2. From the left navigation menu, choose **Servers & Endpoints > Workloads**.
3. Click **Add > Pair Workload with Pairing Profile**.
4. Select any existing pairing profile from the “Pick a Pairing Profile” drop-down menu.
5. Copy the pairing **Key** value (alphanumeric).
6. Log in to the NEN host and run the `illumio-nen-ctl activate` command:

```
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl activate
<pairing_key_value>
--host <pce-address>:<pce-port>
```

Enable load balancer support

After installing the NEN RPM and activating it with the PCE, enable load balancer support by running the following command on the NEN node:



NOTE

If the NEN is configured as an HA pair, run this command on the primary node.

```
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl slb-enable
```

Move a NEN from one PCE to another PCE

You can move a NEN from one PCE to another PCE in the same supercluster. When a NEN is moved in this way, associated Server Load Balancers maintain policy for managed virtual servers. After the PCE database is restored, the moved NEN remains connected to the new PCE. The command for moving a NEN is:

```
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl pce-host-  
update  
<pce-host-addr>:<port>
```

Upgrade Standalone NEN 2.1.0 to Standalone NEN 2.3.x or Later

Upgrading from NEN 2.1.0 to NEN 2.3.x or later

If you perform this procedure, you **don't** need to:

- Restore the NEN database because the NEN upgrade doesn't impact it.
- Activate the NEN with the PCE if you are upgrading an existing NEN installation; that is, if you are upgrading a NEN 2.1.0 standalone installation to NEN 2.3.x or later standalone installation.

1. Run the following upgrade command:

```
sudo yum update -y <path to Illumio NEN rpm>/illumio-nen-  
<release_number>  
-<build-number>.x86_64.rpm
```

2. Enable load balancer support by running the following command on the NEN node:



NOTE

If the NEN is configured as an HA pair, run the command on the primary node.

```
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl slb-enable
```

3. (Optional) To configure an HA pair for the NEN in the PCE cluster, see [Configure HA Support for the NEN \[374\]](#). (The steps are the same whether you are installing a new standalone NEN or upgrading an existing NEN.)

Configure HA Support for the NEN

This optional procedure describes how to install the NEN on a secondary node to provide HA support for the NEN in a PCE cluster. For information about running the NEN as an HA pair, see [NEN HA Support. \[359\]](#) For information about upgrading nodes in an HA pair, see [Upgrade a NEN HA pair \[376\]](#).

Prerequisites for HA support



IMPORTANT

`service_discovery_fqdn` must be the hostname or IP address of the primary node.

- You have already installed the NEN on the primary node.
- Network latency between DB nodes must not exceed 10ms.
- `nen_fqdn` in the `runtime_env.yml` file:
 - Both nodes must have the same `nen_fqdn` so that the PCE knows they are part of the same NEN HA pair.
 - The `nen_fqdn` can be anything you choose as long as it is unique among NEN clusters paired to the PCE.
 - The `nen_fqdn` doesn't need to match the actual hostname of either node nor be resolvable via DNS.
- Each NEN node's actual hostname must be resolvable from the actual hostname of the other node in the pair.



NOTE

You cannot change the `nen_fqdn` once the NEN has been paired to the PCE.

To set up a NEN HA Pair

1. Install the NEN on the secondary node.

```
sudo yum install -y <path_to_Illumio_NEN_rpm>/illumio-nen-
<release_number>
-<build_number>.x86_64.rpm
```

2. Set up NEN runtime environment on the secondary node using one of the following methods:
 - Copy `/etc/illumio-nen/runtime_env.yml` file from primary node to `/etc/illumio-nen/runtime_env.yml` on the secondary node, and change the `node_type` value to `network_enforcement1`
 - `sudo /opt/illumio-nen/illumio-nen-env setup`.
3. Start the NEN on the secondary node.

```
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl start
```

Upgrade a NEN HA pair

To upgrade the nodes in a NEN pair, you must do so in the proper sequence when the nodes are in the proper state.

Before you begin:

- A rolling upgrade is not supported. Perform the upgrade in the order described in the steps below.
- Make sure that the nodes can communicate with each other (that is, that the network connection between them is up). The nodes need to be able to share the same database information. This is to avoid a "split brain" state where both nodes can communicate with the PCE but not with each other.

To upgrade a NEN HA pair

1. Stop the secondary NEN node.
2. Stop the primary node.
3. Upgrade the primary NEN node.
4. Wait for the primary NEN node to be online (in the RUNNING state).
5. Upgrade the secondary NEN node.

Upgrade a PCE-based NEN 2.1.0 to Standalone NEN 2.3.x or later

If you are upgrading Illumio Segmentation for Data Centers to 21.5.0-PCE or later, you must upgrade the NEN to 2.3.x or later. Illumio Segmentation for Data Centers 21.3.0-PCE is not backwards compatible with NEN 2.1.0 and earlier releases.

Upgrade prerequisites

Before taking the NEN database back up, ensure that no asynchronous jobs have been submitted right before you begin the upgrade. As a best practice, wait until all asynchronous jobs have finished before upgrading the PCEs and associated NENs.

**NOTE**

When to back up the NEN database and uninstall the NEN software from the PCE

You must back up the NEN database and uninstall the NEN RPM from your PCE-based NEN installation before you upgrade to Illumio Segmentation for Data Centers 21.3.0-PCE and later. Be aware that you must set the PCE to runlevel 1 before backing up the NEN database on the PCE primary data node and uninstalling the NEN RPM from both PCE data nodes.

About the upgrade

- Upgrading the NEN in a single PCE cluster versus a PCE Supercluster deployment

The steps to install and configure a NEN in a PCE Supercluster deployment are the same as for a single PCE cluster. You perform the procedure to install a NEN in each individual region (PCE Supercluster members).

- Restoring the NEN database in a Supercluster deployment

When upgrading the NEN that is part of a PCE Supercluster deployment, restore the NEN database from the PCE-based installation; you must restore the NEN database on the NEN paired to the PCE Supercluster leader. You do not need to restore the database for the NENs paired with the PCE Supercluster members.

- When to back up the NEN database and uninstall the NEN software from the PCE

You must back up the NEN database and uninstall the NEN RPM from your PCE-based NEN installation before you upgrade to Illumio Segmentation for Data Centers 21.3.0-PCE and later. You must set the PCE to runlevel 1 before backing up the NEN database on the PCE primary data node and uninstalling the NEN RPM from both PCE data nodes.

Upgrade NEN 2.1.0 PCE-based installation to NEN 2.2.0 and later standalone installation

1. Back up the NEN database on the PCE primary data node.

For the requirements and syntax to run PCE commands, see “PCE Control Interface and Commands” in the PCE Administration Guide.

```
sudo -u ilo-pce illumio-pce-ctl set-runlevel 1
sudo -u ilo-pce /opt/illumio-pce/illumio-nen-db-management
dump --file
<filename>
sudo -u ilo-pce /opt/illumio-pce/illumio-pce-ctl stop
```

2. Uninstall the NEN from the PCE data node(s).

```
sudo rpm -e illumio-nen
```

3. Upgrade the PCE to Illumio Segmentation for Data Centers 21.3.0-PCE and later.

For the steps to upgrade a single PCE cluster, see “Upgrade the PCE” in the PCE Installation and Upgrade Guide.

For the steps to upgrade the PCEs in a PCE Supercluster deployment, see “Upgrade Supercluster” in the PCE Supercluster Deployment Guide.

After upgrading your single PCE cluster or PCE Supercluster, ensure that all PCEs are started at runlevel 5 before installing the NEN RPM package.

4. Install and configure the NEN software. See [Install a New Standalone NEN \[370\]](#). (This is the procedure for installing a new NEN standalone installation, but the steps are the same whether you are installing a new standalone NEN or upgrading an existing NEN. In the NEN upgrade procedure, you will have uninstalled the previous NEN by this step and must install the new NEN release.)
5. Set the NEN to **runlevel 1** and restore the NEN database that you copied from the PCE primary data node:

```
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl set-
runlevel 1
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-db-management
restore --file
<path to file to restore>
```

If you are performing this step for a NEN that is part of a PCE Supercluster deployment, restore the NEN database for the NEN node paired with the PCE Supercluster leader. You don't need to restore the NEN database in each Supercluster member region.

6. Set the NEN to runlevel 5 and activate the NEN with a pairing key from the PCE by using the --repair option:

```
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl set-
runlevel 5
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl activate
<pairing-key>
--host <PCE_host address:port> --repair
```

If you are performing this step for a NEN that is part of a PCE Supercluster deployment, repair the NEN with the Supercluster leader PCE. Additional-

ly, pair any new NEN installations in each region with the Supercluster member in that region.

For the steps to obtain a pairing key from the PCE, see [Obtain Pairing Key and Activate the NEN \[373\]](#).

7. To configure an HA pair for the NEN in the PCE cluster, see [Configure HA Support for the NEN \[374\]](#). (The steps are the same whether you are installing a new standalone NEN or upgrading an existing NEN.)

Configure Non-default Ports and Protocols for NEN Flow Collection

Beginning with release 2.7.10, the NEN supports configurable flow-transport settings. Instead of using UDP and fixed destination ports, you can now configure TCP for flow collection and send flows to ports you define. This update provides greater flexibility and support for diverse network and security requirements.

Default schema

The following schema defines `flow_data_consumers` and identifies the default protocol and ports:

- **sFlow** : UDP/6343
- **IPFIX**: UDP/4739
- **NetFlow**: UDP/2055

```

"flow_data_consumers"=>{
  "description"=> "An array of properties defining traffic
flow data consumers that override the default consumers
that use udp and standard port (ipfix(4739), netflow(2055),
sflow(6343)).\nNOTE: All connectors must be defined for the
parser being specified, even if using the default connector
for the parser.",
  "help"=> "An array of properties defining traffic
flow data consumers that override the default consumers
that use udp and standard port (ipfix(4739), netflow(2055),
sflow(6343)).\nNOTE: All connectors must be defined for the
parser being specified, even if using the default connector
for the parser.",
  "expose_to"=>"public_experimental",
  "type"=>"array",
  "items"=>{
    "type"=>"object",
    "additionalProperties"=>"false",
    "required"=>["parser", "connectors"],
    "properties"=>{
      "parser"=>{
        "description"=>"Type of parser.",
        "help"=>"Type of parser.",
        "expose_to"=>"public_experimental",
        "type"=>"string",
        "enum"=>["ipfix", "netflow", "sflow"]
      },
      "connectors"=>{
        "description"=>"An array of transport and ports
used to receive data for this parser.",
        "help"=>"An array of transport and ports used to
receive data for this parser.",
        "expose_to"=>"public_experimental",
        "type"=>"array",
        "items"=>{
          "type"=>"object",
          "additionalProperties"=>"false",
          "required"=>["transport", "ports"],
          "minItems"=>1,
          "properties"=>{
            "transport"=>{
              "description"=>"The protocol used to send the
flow data. Default: udp",
              "help"=>"The protocol used to send the flow
data. Default: udp",
              "expose_to"=>"public_experimental",

```

```
        "type"=>"string",
        "enum"=>["udp","tcp"],
        "default"=>"udp"
    },
    "ports"=>{
        "description"=>"An array of integers defining
the port(s) the Network Enforcement Node should listen on for
the flow data.",
        "help"=>"An array of integers defining the
port(s) the Network Enforcement Node should listen on for the
flow data.",
        "expose_to"=>"public_experimental",
        "type"=>"array",
        "minItems"=>1,
        "uniqueItems"=>true,
        "items"=>{"type"=>"integer", "format"=>"port"}
    }
}
}
```

Custom schema example

This example shows a customized `flow_data_consumers` schema with a mix of default and custom settings:

Protocol	Ports
sFlow	• Custom transport protocol and ports: TCP [16353, 16354] • Default transport protocol and port: UDP [6343]
IPFIX	• Default transport protocol and custom port: UDP [14739]

```

flow_data_consumers:
- parser: sflow
  connectors:
    - transport: tcp
      ports: [16353,16354]
    - transport: udp
      ports: [6343]
- parser: ipfix
  connectors:
    - transport: udp
      ports: [14739]

```

Generate NEN Reports

You can generate NEN health and support reports as well as enable/disable debug mode logging. These provide information useful for troubleshooting issues with your NENs.

Health Report

- Appears onscreen only
- Includes the following information

Cluster Mode: HA Pair			
Cluster Status: Normal			
Available Time: 0d 4h 27m 24s			
PCE FQDN: [REDACTED]			
Nodes:			

Hostname	:	[REDACTED]	[REDACTED]
Ip Addr	:	[REDACTED]	[REDACTED]
Node Type	:	network_enforcement0	network_enforcement1
Runlevel	:	5	5
Report Time	:	2021-07-30T18:10:04+00:00	2021-07-30T18:10:00+00:00
Node Available Time	:	2021-07-30T13:43:00+00:00	
Uptime	:	21 days 13:09 hours	21 days 13:08 hours
Service Status	:	RUNNING	RUNNING
Database Master	:	true	
Cpu	:	0% (Normal)	5% (Normal)
Disk	:	8% (Normal)	6% (Normal)
Memory	:	31% (Normal)	27% (Normal)
Services:			

Database Service	:	RUNNING	
Database Slave Service	:		
Network Enforcement Service	:	RUNNING	RUNNING

To generate a NEN health report only:

1. Establish a secure shell connection (SSH) to the NEN you want to investigate
2. Issue the following command:

```
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl health
```

Support Report

A Support Report is a unique generated file saved to the /tmp directory. It includes the following information and data:

- A Health Report (see the image above)
- NEN logs

To generate a NEN support report:

1. Establish a secure shell connection (SSH) to the NEN you want to investigate.
2. Run the following command:

```
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl support-report
```

3. On successful completion, the command indicates where you can find the file so that you can copy the support report off the NEN.

Debug Mode Logging

You can turn debug mode logging on or off. When enabled, debug mode logging provides detail for the `network_enforcement_service`. The following command allows you to show the current debug mode node status or turn debug logging mode on or off dynamically:

```
sudo -u ilo-nen /opt/illumio-nen/illumio-nen-ctl debug-mode status/on/off [--all-nodes]
```

NEN Integration with Load Balancers

This section describes how to create security policy and apply those policies on the load balancers for use with the NEN.

Load Balancers and Virtual Servers for the NEN

Illumio Core supports activation of enforcement on a number of load balancers as listed below.

Supported Load Balancers

- F5 BIG-IP 11.5x or later
- AVI Vantage 18.23 or later
- Citrix ADC (NetScaler) 13.1 or later

Load Balancer and Virtual Server Concepts

- **Load balancer (SLB):** Either a physical machine or a virtual machine performing load balancing functions. An SLB object represents a standalone device or an HA Pair and includes management of IP/port, user/password, and so on. These values are used by an Illumio NEN to read information from and manage the device. In case of HA, it may include multiple SLB devices.
- **Illumio Virtual Server:** The same as a load balancer Virtual Server.
- **Discovered Virtual Server:** An Illumio NEN queries the load balancer for VIPs and specifies the client-facing VIP with port + protocol combination.
- **Created Virtual Server:** Is a provisionable policy object with labels used in policy writing. In the UI, the Virtual Server creation process is called VIP Management. Virtual Server providers (backend servers) are specified using labels and can optionally specify backend port independently of the port used by the VIP.
 - **VIP:** Is a virtual IP or a local IP (a front-end IP that clients can connect to).
- **SNAP pool:** Is a group of IPs that the Virtual Servers use to connect to the backend servers. A Virtual Server can only have a single VIP connected to it, on a single port. It can also be accessed by the SLBs local IPs.

About Load Balancers

Illumio Segmentation for Data Centers supports activation of enforcement on a number of load balancers listed below.



IMPORTANT

Beginning with Illumio Segmentation for Data Centers 19.3.0 release, the Network Function Controller (NFC) is no longer supported. The F5 interface has been moved from the PCE in to the Network Enforcement Node (NEN). Because the NFC has been discontinued, you need to use the NEN to interface with Load Balancers.

By applying labels to your load balancer's virtual servers, you can write rules that allow client workloads in front of the load balancer to communicate with

the virtual IP address of the load balancer's virtual servers. By adding labels to the pool members behind a virtual sever, you can allow communication from the load balancer to the members of the pool. The source for this communication is determined by the load balancer. The Illumio Segmentation for Data Centers programs policies on the load balancer to enforce security policy. The NEN uses the load balancer's REST APIs to read and write security policies to configure security rules.

The PCE supports configuration of two load balancer units if they are configured in Active/Standby or Active/Active modes. The PCE needs to be configured with the user name and password of an administrative user who has read-write access to all configurations on the load balancer.

The NEN configures new objects on the load balancer and does not alter any existing configurations. When an Illumio-created object in the load balancer configuration is modified, the NEN detects it as tampering and modifies the configuration back to the intended state so that the correct security policy is enforced.

The Illumio Segmentation for Data Centers dynamically adjusts policies on the load balancer based on application and topology changes in the datacenter so that the Illumio Segmentation for Data Centers can enforce consistent security policy on load balancers across the datacenter and cloud environments, as well as show the application traffic in Illumination. The Illumio Segmentation for Data Centers keeps track of the policy it programmed and reconfigures policy if it was altered on the load balancer manually or by other means.

The NEN makes use of the following constructs on load balancers:

- **F5 LTM:** iRules or LTP policies on the LTM provide capability to restrict application access. When the LTM is used as enforcement mechanism, the NEN uses virtual-server based iRules/LTP policies and Datagroup Lists.
- **F5 AFM:** AFM provides stateful firewalling on BIG-IP. When AFM is used as an enforcement mechanism, the NEN uses Network Firewall policies in the virtual server section and address-lists in the network firewall. The NEN also supports the F5 BIG-IP Application Services 3 Extension (referred to as BIG-IP AS3) when it is used to define virtual servers on the F56 AFM.
- **AVI:** The NEN uses the Network Security Policy rules to program AVI Vantage.
- **Citrix ADC (NetScaler):** The NEN uses responder policies to control access to the Virtual Servers.

**NOTE**

Configuring two SLB units in Active/Standby mode is supported. However, clustering is **not supported**.

F5 BIG-IP Requirements

The NEN uses its REST API to program F5 load balancers, which means that F5 needs to be running a software version that supports REST-API. The requirements include:

- BIG-IP 11.5.x or higher
- Utilize SNAT or Auto-map mode

AVI Vantage Requirements

- AVI Vantage 18.2.3 or higher

Citrix ADC (NetScaler) Requirements

- Citrix ADC 13.1 or higher

Configure Load Balancers

You can add a load balancer using the PCE web console. However, before you add a load balancer, you need to pair the NEN with the load balancer functionality enabled with the PCE.

**NOTE**

A load balancer does not need to be provisioned to work. However, the virtual servers you associate with this load balancer do need to be provisioned.

Add an SLB from the PCE Web Console

You can add a load balancer using the PCE Web Console. However, before you add a load balancer, you need to pair the NEN with the load balancer functionality enabled with the PCE.

1. From the PCE Web Console menu, choose **Infrastructure > Load Balancers**.

2. Click **Add**.
3. Specify a name for the load balancer and provide a description.
4. From NEN hostname, select the NEN that you want to manage policy programming for this particular SLB.
5. From Device Type, select appropriate load balancer device type.
6. From number of devices, select **(1) Standard** or **(2) HA Pair**.
The load balancer details are displayed.
7. Specify the following settings to enable the PCE to connect to the load balancer:
 - Management IP address or FQDN of the load balancer
 - Port on which to connect
 - Username
 - Password
8. Select **Verify TLS** to verify the trust of the TLS certificate provided by the load balancer before connecting to it.
9. Click **Save**.

Move an SLB to a different NEN host (single and super cluster)



NOTE

Requires Core PCE version 22.2.0 or later.

You can move a Server Load Balancer to a different NEN host from the PCE Web Console. This capability preserves – on the moved SLB – all policies already assigned to the managed virtual server.

1. From the PCE Web Console, go to **Infrastructure > Server Load Balancers**.
2. In the **Name** column, click the link for the SLB you want to move.
3. On the **Summary** tab for the selected SLB, click **Edit**.
4. In **NEN hostname**, click the drop down list to select the destination NEN host where you want to move the SLB.
5. Click **Save**. The PCE recognizes that the SLB has been moved to the chosen NEN host.

About Virtual Servers

Virtual servers in the Illumio Segmentation for Data Centers contain two parts:

- A virtual IP address (VIP) and port through which the service is exposed
- Local IP address(es) used to communicate with backend servers (pool members).

A virtual server is similar to a workload. It can be assigned labels and has IP addresses, but does not report traffic to the Illumio Segmentation for Data Centers. Each virtual server has only one VIP. The local IP addresses are used as a source IP address for connections to the pool members (backend servers) when the virtual server is operating in SNAT mode or Auto mode. These IP addresses are likely to be shared by multiple virtual servers on the server load balancer.

A virtual server is identified by a set of labels. The sources (aka destinations) and destinations (aka providers) for a virtual server can be assigned different labels, which could place them in the same group or a different group in Illumination. See **Groups in Illumination** in the Visualization Guide for information.

Providers are allowed to have an incomplete label set (for example, only a Location label), so the providers can be in all groups within the specified location. As a result, a single virtual server can have providers in any group or in any number of groups in Illumination.

Virtual Server Members and Labels

The Illumio Segmentation for Data Centers allows you to write rules to allow communication with workloads managed by a load balancer using labels.

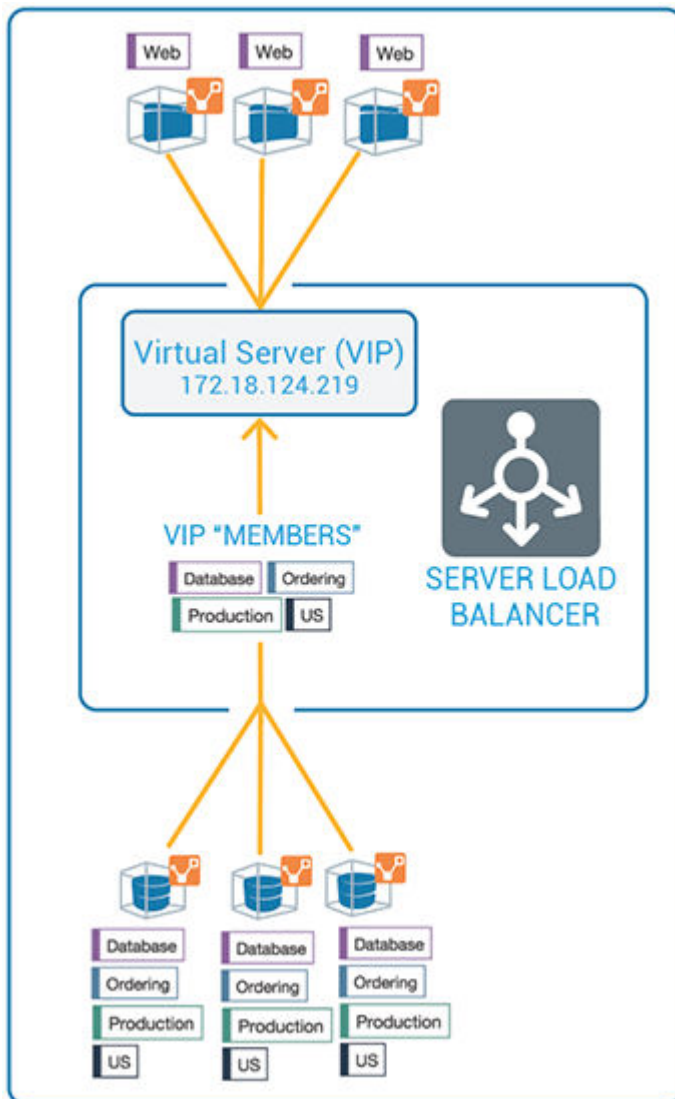
Virtual Server Members

When you configure load balancers in the PCE, it connects to the load balancer using the Illumio Segmentation for Data Centers REST API. The PCE reads all the load balancer virtual servers configurations and populates the Discovered Virtual Servers tab of a load balancer's details page. Any virtual servers associated with the load balancer can be converted to a managed virtual server for use with the PCE. When you configure the virtual server in the PCE web console, you can apply labels to the virtual servers. After configuring a virtual server, you can write a rule that allows other clients to communicate with it.

The members behind a virtual server are specified by configuring a set of labels in the virtual server configuration. A set of four Illumio labels can be applied on the Virtual Server Members tab, which is used to match the same set of labels on workloads in the virtual server's pool. If any of the workloads in the virtual server pool share the same set of four labels specified under the

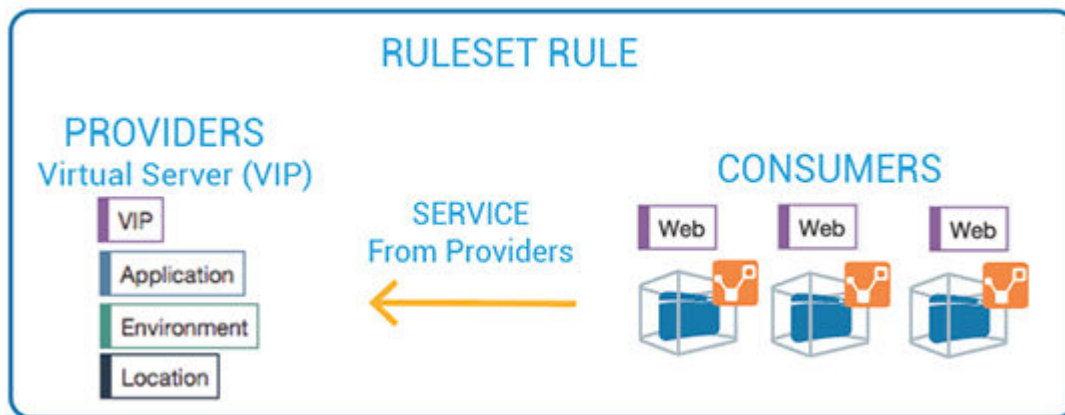
Virtual Server Members tab, then any rule you write with the virtual server also applies to the workload members.

In this diagram, you can see how the workloads that belong to the virtual server pool have the same labels specified on the Virtual Server Members tab:



Ruleset Rule for Virtual Server

This diagram illustrates the rule you can write after you label a virtual server and its members:



Configure Virtual Servers

After adding a load balancer to the PCE, you can manage its virtual servers. To each virtual server, you can apply labels through the Virtual Server details page. Applying labels to a virtual server allows you to add the virtual server to a rule.

When the policy is enforced on the virtual server by the NEN, access from any IPs/Workloads to the Virtual Server is controlled according to the rules defined by the policy. The NEN removes the rules from the virtual server when the policy is no longer enforced.

Configuring a load balancer's virtual servers consists of these three settings:

- **Enforced or Not Enforced:** When you select Enforced, any rules you write using the labels associated with the virtual servers and any of its members are enacted. Selecting Not Enforced disables the labels and any policy written that affects the virtual server or its members is disabled.
- **Service:** Select the service to use for the rules that allow access to the virtual server. For example, HTTPD 80 TCP.
- **Labels:** You must apply one each of the four Illumio labels to the virtual server: Role, Application, Environment, and Location. Assigning labels enables the virtual server to be used in rules.



NOTE

Virtual servers are considered a security policy item, so any changes to a virtual server configuration must be provisioned before any of those changes take effect and become active.

Virtual Server Limitations

- Illumination does not support location-level and application-level maps.
- If a single SNAT pool is shared between multiple virtual servers, the Illumination map does not render correctly.
- SNAT and Auto-map modes of F5 virtual servers are supported. Transparent mode is not supported.



NOTE

Before any virtual server configuration can go into effect, you need to provision your changes. See the Security Policy Guide for information.

Configure the polling interval for discovering new virtual servers

Beginning in release NEN 2.4.10, you can configure how frequently the NEN polls Server Load Balancers (SLBs) to discover new virtual servers (VS). You do this by adding a field to the `runtime_env.yml` file. In previous releases the timeout value was fixed at 5 minutes, which was too long for some use cases. SLB discovery events are customer-configurable as follows:

- **Default** = 5 minutes. You don't have to modify the runtime environment file if you want to keep the default setting.
- **Minimum** = 2 minutes
- **Maximum** = none

The NEN reads the timeout value at startup and polls SLBs accordingly. If you add this field and/or update the timeout value in the field, you must restart the NEN for the change to take effect.

Procedure

You can modify the runtime environment file on an already-running NEN or when installing a NEN. For details, see the "Install a New Standalone NEN" topic.

1. Locate the NEN runtime environment file in the following directory:

```
/etc/illumio-nen/runtime_env.yml
```

2. If it's not already present, add the line `slb_discovery_timeout_minutes` to the file.

3. Add a space, a colon (:), and value of 2 or higher at the end of the line. For example, to configure the SLB discovery timeout to **3 minutes**, you'd enter:

```
slb_discovery_timeout_minutes: 3
```

4. Restart the NEN for the new setting to take effect.

If you've updated the timeout value on an already-running NEN, you're done at this point. If you've configured the timeout value as part of a new NEN installation, continue to [NEXT STEPS \[392\]](#) below.

NEXT STEPS

1. Activate the NEN with a pairing key from the PCE. See the "Obtain Pairing Key and Activate the NEN" section.
2. To enable the NEN to integrate with a load balancer, see the "Enable Load Balancer Support" section.
3. (Optional) To configure the NEN as an HA pair, perform the steps in "Configure HA Support" for the NEN.

Filter the Virtual Server List

You can filter the Virtual Servers list by using the properties filter at the top of the list. For example, you can filter and search by label. You can also filter and search by the following objects:

- Virtual server mode
- Virtual IP address, the VIP port number, or VIP Protocol
- Server Load Balancer

Configure a Load Balancer's Virtual Servers

1. From the PCE web console menu, choose **Infrastructure > Load Balancers**.
2. Select the load balancer for which you want to configure virtual servers.
3. Select the **Virtual Servers** tab.
4. Select one of the load balancer's virtual servers and click **Manage**.
5. Select one of the virtual servers and click **Edit**.
6. Enter a name and description for the virtual server.
7. To enable the virtual server's policy, select **Enforced**.
8. Select a service to associate with the virtual server. The service selected enables that service to be used in rules you write for this virtual server.
9. Select one each of the four labels to assign to the virtual server.
- 10 Click **Save**.

11. Before any virtual servers can go into effect, they must be provisioned.

Write SLB Policy

Writing a policy for a load balancer is similar to writing a policy for a workload, except for the following differences:

- Leave the service as unspecified and the port and protocol of the discovered VIP will determine the service automatically.
- Specify “Uses Virtual Services” in the rule.

A rule that is provided between a virtual server (or its labels) and a set of sources (aka destinations) implicitly programs two sets of rules:

- Rules between the consuming workloads or labels and the frontend VIP of the F5 on the discovered VIP port and protocol: Traffic flows between consuming workloads and the VIP are enforced on both ends if the virtual server is managed and enforced.
- Rules between the F5 pool and the virtual server providers on the service specified in the virtual server object (usually All Services): These rules are enforced for inbound traffic to the virtual server source if the virtual server source workloads are enforced.

SLB Methods

The SLB APIs are used to enable automation for F5 policy management.

Functionality	HTTP	URI
Get the list of SLBs	GET	[api_version][org_href]/slbs
Get a specified SLB	GET	[api_version][org_href]/slbs/:uuid
Create an SLB object	POST	[api_version][org_href]/slbs

SLB Parameters

The parameters for the SLB methods are:

Parameter	Description	Type
name	The short friendly name of the server load balancer	String
nfc	Network Function Controller managing this SLB	String
device_type	Device type of the server load balancer	String
devices	Configuration and runtime state of the devices backing this SLB Network VF.	String

Configure an SLB Object

Step 1. Create an SLB object and instruct the NEN to sync with it.

POST /api/v2/orgs/{org id}/slbs

```
{
  "devices" : [
    {
      "config" : {
        "username" : "admin",
        "port" : 443,
        "credential" : "admin", # never replayed in
northbound API
        "host" : "10.2.32.6",
        "credential_type" : "password",
        "check_certificate" : false
      }
    }
  ],
  "device_type" : "F5 Big-IP LTM"
  "name" : "Illumio Test SLB"
}
```

Step 2. GET an SLB response.

GET /orgs/{org id}/slbs/{UUID of SLB object}


```
{
  "name" : "Illumio Test SLB",
  "devices" : [
    {
      "status" : {"connection_state" : "pending"},      # will
become successful when NEN
              syncs w/ device
      "href" : "/orgs/1/slb_devices/9349ff36-ab38-42bf-909a-
eb5aa3baf187",
      "config" : {
        "username" : "admin",
        "check_certificate" : false,
        "credential_type" : "password",
        "host" : "10.2.32.6",
        "credential" : null,
        "port" : 443
      }
    }
  ],
  "href" : "/orgs/1/slbs/8a82a1b0-c2ce-43ec-
abf7-77bd8a3fd22c",
  "device_type" : "f5_bigip_afm"
  [ ... ] # created_at, updated_at, etc.
}
```

Step 3. GET a list of Discovered Virtual Servers.

```
GET /orgs/1/discovered_virtual_servers
```

```
{
  "snat_type" : "snat_pool",
  "dvs_identifier" : "d3b784c2fd24ad364c5adb3319169bd2",
  "mode" : "snat",
  "vip_port" : {"port" : 8080, "protocol" : 6, "vip" :
"172.16.27.88" },
  "service_checks" : [{"protocol" : 1}],
  "name" : "Common/QL_VIP_1",
  "slb" : {
    "href" : "/orgs/1/slbs/8a82a1b0-c2ce-43ec-
abf7-77bd8a3fd22c"
  },
  "snat_pool_ips" : ["172.16.26.27", "172.16.26.18",
"172.16.27.18"],
  "local_ips" : ["172.16.26.18", "172.16.27.18"],
  "href" : "/orgs/1/discovered_virtual_servers/
2c460b98-2176-4a44-9ba4-e77f3eacd0f1"
  [ ... ] # created_at, updated_at, etc.
}
```

Step 4. Manage a VIP by creating a Virtual Server object.

POST /orgs/1/sec_policy/draft/virtual_servers

```
{
  "name" : "Common/chris-VIP1",
  "service" : {
    "href" : "/orgs/1/sec_policy/draft/services/1"
  },
  "labels" : [],
  "providers" : [],
  "mode" : "unmanaged", # enforced
  "discovered_virtual_server" : {
    "href" : "/orgs/1/discovered_virtual_servers/
23338ceb-7580-466a-bbcf-a645b82ce97b"
  }
}
```

Step 5. Modify the enforcement mode, labels, and backend/source labels of the Virtual Server.

PUT /orgs/1/sec_policy/draft/virtual_servers/84bae9dd-
f1f6-4322-bffc-f07354b0622a

```
{
  "mode" : "enforced",
  "labels" : [{"href" : "/orgs/1/labels/448"}, {"href" : "/orgs/1/labels/444"}],
  # any RAEL tuple
  "providers" : [{"label":{"href":"/orgs/1/labels/449"}}]
  # note: providers may have different labels
}
```

Step 6. Provision the Virtual Server into an active policy.

POST /orgs/1/sec_policy

```
{
  "update_description" : "Provision my first VS",
  "change_subset" : {
    "virtual_servers" : [{"href" : "/orgs/1/sec_policy/draft/virtual_servers/84bae9dd-f1f6-4322-bffc-f07354b0622a"}]
  }
}
/orgs/1/sec_policy/draft/virtual_servers/84bae9dd-f1f6-4322-bffc-f07354b0622a
/orgs/1/sec_policy/active/virtual_servers/84bae9dd-f1f6-4322-bffc-f07354b0622a
```

Step 7. Write rules that apply to the Virtual Server.

POST /orgs/1/sec_policy/draft/rule_sets/1480/sec_rules

```

{
  "enabled" : true,
  "providers" : [
    {"label" : {"href" : "/orgs/1/labels/444"}},
    {"label" : {"href" : "/orgs/1/labels/448"}}
  ],
  "resolve_labels_as" : {
    "destinations" : ["workloads"],
    "providers" : ["virtual_services"]    # NOTE: Must be
virtual_services
  },
  "destinations" : [
    {"actors" : "ams"}    # All Workloads
  ]
}
{
  "destinations" : [
    {"label" : {"href" : "/orgs/1/labels/444"}}
  ],
  "providers" : [
    {
      "virtual_server" :
      {"href" : "/orgs/1/sec_policy/draft/virtual_servers/
      84bae9dd-f1f6-4322-bffc-f07354b0622a"}
    }
  ],
  "enabled" : true,
  "resolve_labels_as" : {
    "destinations" : ["workloads"],
    "providers" : ["virtual_services"]
  }
}

```

Remove Filtering

Some types of virtual servers are not visible, such as those without default server pools. From the NEN 2.1.0 release onwards, you can do filtering related to such virtual servers. You can see VIPs that do not have a pool associated with them or are not SNAT/Auto-SNAT.

To view all types of virtual servers configured on the F5 load balancers, you must enter specific commands during the NEN installation (on a NEN by NEN basis). These commands disable (enabled by default) the built-in filter running on the NEN on the Leader PCE cluster.

1. Enter the following command:

```
sudo -su ilo-nen /opt/illumio-nen/illumio-nen-ctl slb-enable  
--virtual-server-filtering disabled
```

2. Restart the NEN on both db0 and db1 nodes:

```
sudo -u ilo-nen /opt/illumio-nen/illumio-pce-ctl restart
```

NEN Integration with Switches



NOTE

In the NEN Guide, the term "switch" refers to both switches and routers.

This section describes how to create security policy and apply those policies on the switches for use with the NEN.

Overview of Switch Integration



NOTE

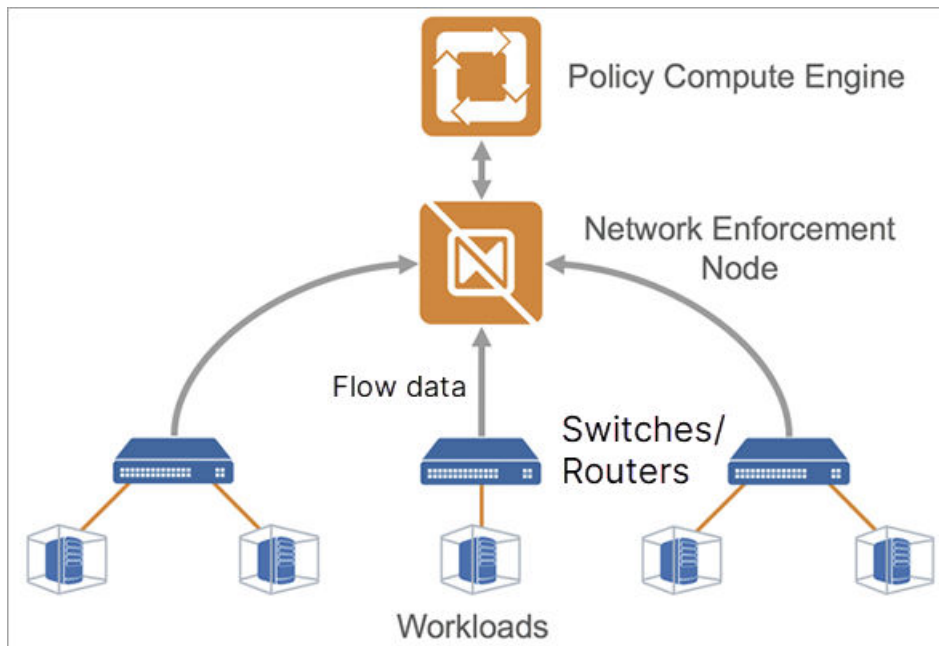
In the NEN Guide, the term "switch" refers to both switches and routers.

The Illumio Network Enforcement Node (NEN) is the Illumio Core switch interface, which allows you to get visibility and enforcement on switches. Using the NEN, you can secure workloads that are attached to network switches. You can use the NEN to generate access control lists (ACLs) and load those on your switches to protect the ports to which your workloads are attached.

How the NEN Receives Switch Data

You can configure your switches to send flow data to a flow data collector, such as the NEN. An Illumio Core administrator can configure the NEN to listen for flow data from switches and associate workloads to those switches. The NEN receives flow data directly from the switches, summarizes it, and

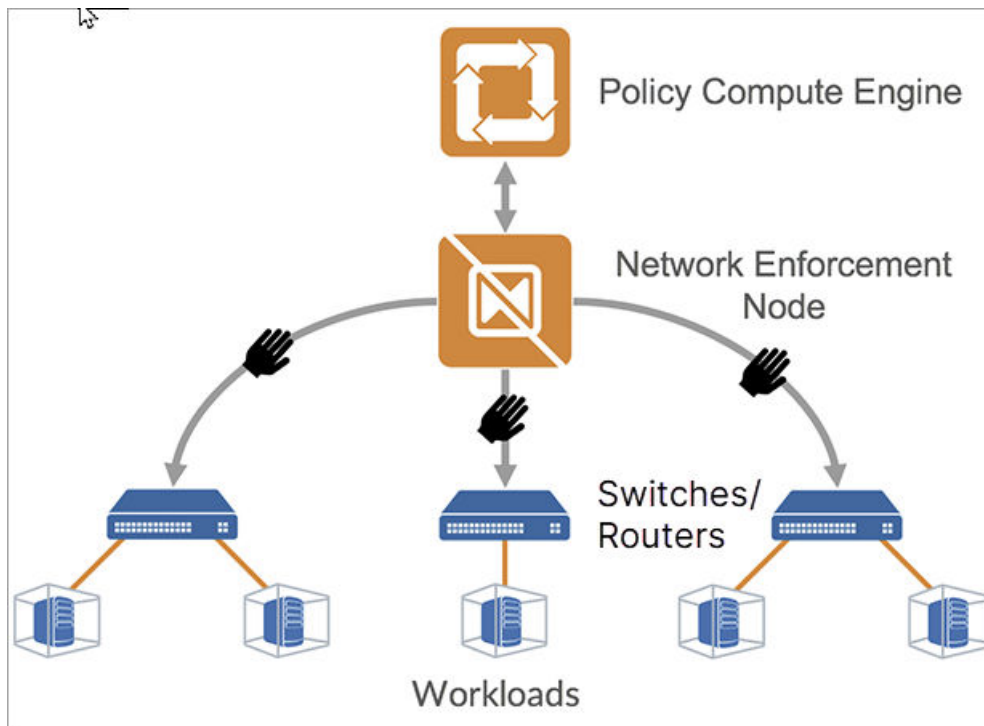
uploads it to the PCE. You can view this traffic flow in the Illumination® map and stream it out of the PCE through UDP in Splunk, CEF, or LEEF formats.



Extended Policy Model

The Illumio policy model encompasses workloads with native stateful firewalls built-in, such as Linux iptables or Windows Filtering Platform. Although all systems might not have a firewall built in, they still have segmentation requirements. To solve this use case, Illumio has extended its policy model to switches.

Illumio administrators can use the NEN to convert natural language policies into ACLs, which the switches understand natively. Your organization's teams that use Illumio Core can download ACLs from the PCE and provide them to the networking team for review before applying new policies to the switches.



Limitations for Switch Integration



NOTE

Refer to your vendor-provided documentation for information about where port and protocol addresses can be applied.

This release is subject to the following limitations:

- You must provide a switch IP address and an interface traffic flow ID for interfaces that need to be monitored for flow data.
- The NEN discards flow data from an interface that it does not monitor.
- The Illumio Segmentation for Data Centers generates IPv4 and (beginning in NEN release 2.7.0) IPv6 ACLs that can be applied to either the L3/Routed interfaces or Switch Virtual Interface (SVI) for L2 interfaces when they are a member of a VLAN. Whenever ACLs are applied to the SVI, workloads within the same VLAN can freely communicate regardless of policy.

Requirements for Switch Integration

- Cisco Nexus 9200 or 9300 or Arista 7000 series switches (requires NEN release 2.1.0 or later and PCE release 20.2.0 or later)

- Cisco IOS XR series routers (requires NEN release 2.7.0 or later and PCE release 25.3 SaaS or later)
- Workloads that are directly attached to the switch on L2 or L3 ports or on port channels



NOTE

NEN releases 2.6.4 and earlier

The NEN targets top-of-rack (TOR) switches that are directly attached to the workload and not the core switches. For example, Cisco Nexus 9200 and 9300 switches are supported, but the Cisco 9500 series switches are not supported.

NEN releases 2.7.0 and later

The NEN targets top-of-rack (TOR) switches. For example, Cisco Nexus 9200 and Cisco 9300 switches and Cisco IOS XR Series routers are supported, but the Cisco 9500 series switches are not supported.

Workflow for Setting up NEN Switch Integration

The following is an overview of the steps required for working with the NEN for switch integration:

1. In the PCE web console:
 - a. Define the switches.
 - b. Create unmanaged workloads.
 - c. Assign those unmanaged workloads to switch interfaces.
 - d. Create security policy rules to protect the workloads attached to the switches.
2. Use the PCE REST API or the PCE web console to generate switch ACLs based on your organization's security policies.
3. Get the generated ACLs on to the switch either manually or automatically, depending on your integration method (UI or API) and the switch model:
 - **Cisco Nexus 9200 and 9300 series switches; Arista 7000 series switches:** Use the switch's command line to manually copy and paste the generated ACLs to configure the switch.

- **Streamlined OVS integration through the PCE API:** If you are integrating with OVS via the PCE API, you can enter the IP address and credentials for the OVS switch and the NEN will automatically:
 - Discover the switch configuration
 - Program flow monitoring on the switch
 - Discover and creates workloads in the PCE
 - Program the ACLs on the OVS
4. Using the PCE REST API or the PCE web console, inform the PCE that the ACLs have been loaded.

Result: The PCE-generated ACLs on the switch will protect the target workloads.

Supported switches and configurations



NOTE

In the NEN Guide, the term "switch" refers to both switches and routers.



NOTE

This topic has been updated to include information about the Cisco IOS XR router, the NetFlow and IPFIX flow data monitoring protocols, and IPv6 address support. Illumio's support for these items began in NEN release 2.7.0 (Requires PCE 25.3 SaaS and later).

The following switches are supported in this release:

- Cisco Nexus 9200 and 9300 series
- Cisco IOS XR series routers (requires NEN release 2.7.0 or later and PCE release 25.3 SaaS or later)
- Arista 7000 series

Switch configuration

The following ACL and interface configurations are supported for the Illumio NEN integration:

ACL Implementation	Switch Interfaces	ACL Type
Router ACL (RACL)	- VLAN interface (SVI) - Layer 3 physical interface - Layer 3 port-channel interface	- IPv4 - IPv6 (beginning in NEN release 2.7.0)
RACLs support both inbound and outbound enforcement.		



IMPORTANT

Unsupported interface and ACL configurations



NOTE

Refer to your vendor-provided documentation for information about where port and protocol addresses can be applied.

NEN 2.6.4 and earlier does not support:

- VLAN ACL (VACL) or Virtual Teletype (VTY) ACL as the ACL implementation
- VLAN trunk port (switchport mode trunk) or sub-interface as the switch interface
- MAC ACL type
- IPv6 ACL type
- PACLs for Layer 2 interfaces

NEN 2.7.0 and later does not support:

- MAC ACL type
- PACLs for Layer 2 interfaces

Administrative access to the switch

You or your network administrators need administrative access to your switches to configure them and load the NEN-generated ACLs.



NOTE

The PCE and the NEN do not send any communication to the switch and never log into the switch. The PCE and the NEN do not require root or admin privileges on the switch.

Sufficient TCAM

Your switch's ternary content-addressable memory (TCAM) must be sufficient to store the IPv4 RACLs generated by the NEN.



NOTE

Illumio does not provide a mechanism to check the TCAM depth or available memory for each platform. Your network or security administrators need to check whether the generated IP ACLs can be handled by the switch.

Enable network flow monitoring protocol

In order for the NEN to provide network traffic flow data to the PCE for Illumination, your switch must be configured to work with a network flow monitoring protocol such as sFlow, NetFlow, and IPFIX. See your vendor documentation for information.



NOTE

If you are using either the IPFIX or NetFlow flow data monitoring protocols, configure the protocol to send one of the following fields along with the flow data information (see the IBM document [IPFIX Information Elements](#)):

- **interfaceName** (ElementID 82), which will be matched against the interface name for the switch specified in the NEN
- **ingressInterface** (ElementID 10) which is matched against the **ifindex** of the interface
- **egressInterface** (ElementID 14) which is matched against the **ifindex** of the interface

Configure ports for flow output

The flow output from the switch must be sent to the PCE so it can be monitored. The standard ports for supported flow monitoring protocols are:

- **sFlow**: 6343
- **NetFlow**: 2055 (Illumio's support began in NEN release 2.7.0)
- **IPFIX**: 4739 (Illumio's support began in NEN release 2.7.0)

Network connectivity between switches and the NEN

The NEN listens for sFlow, NetFlow, and IPFIX from the switches.



IMPORTANT

Ensure that your network is configured to allow communication between your switches and the NEN.

Switch information

You need to enter information about the switch in the PCE web console. See the table listed in Add Unmanaged Workloads and Switch Definitions in the PCE Web Console for information.

Configure Switches for the NEN



NOTE

In the NEN Guide, the term "switch" refers to both switches and routers.



NOTE

This topic has been updated to include information about the Cisco IOS XR router, the NetFlow and IPFIX flow data monitoring protocols, and IPv6 address support. Illumio's support for these items began in NEN release 2.7.0 (Requires PCE 25.3 SaaS and later).

You must configure the flow data monitoring protocol on the switch to send its output to the NEN. Additionally, from the PCE web console you must configure the switch's IP address and monitored-interfaces in the NEN service. If the NEN service receives flow data from an unrecognized or undefined network endpoint (or interface), it will reject that information. The NEN service continually aggregates the flow data and sends the aggregated information to the PCE traffic collector every 10 minutes.

**NOTE**

If you are using either the IPFIX or NetFlow flow data monitoring protocols, configure the protocol to send one of the following fields along with the flow data information (see the IBM document [IPFIX Information Elements](#)):

- **interfaceName** (ElementID 82), which will be matched against the interface name for the switch specified in the NEN
- **ingressInterface** (ElementID 10) which is matched against the **ifindex** of the interface
- **egressInterface** (ElementID 14) which is matched against the **ifindex** of the interface

Configure your switch or router

Refer to your switch or router documentation for how to configure your preferred flow data monitoring protocol.

1. Enable a flow data monitoring protocol on the flow exporter.
2. Configure the NEN as a flow collector. The flow collector address is the IP address of the NEN primary node.
3. Configure the flow exporter address as the IP address of the switch, which you will also enter in the PCE web console.
4. Configure the interfaces that you want to monitor. These should match the name you specified in the PCE web console for the switch.

For more information, see [Add Unmanaged Workloads and Switch Definitions in the PCE Web Console \[408\]](#).

Collect the SNMP ifindex value for your switch or router

When the switch or router reports flow data to the NEN, it includes interface index (ifindex) details in the flow records. When the NEN receives flow data,

it parses the records and retains records only for the interfaces you specify in the NEN configuration. You need to collect the ifindex IDs and add them to the NEN configuration later. See your switch documentation for the ifindex of the interfaces you want to monitor.

Add Unmanaged Workloads and Switch Definitions in the PCE Web Console

To create a security policy, the switches and the workloads attached to them should be defined in the PCE web console as follows:

1. Log into the PCE web console.
2. Define the unmanaged workloads that are attached to the switch by going to **Servers & Endpoints > Workloads > Add > Add Unmanaged Workload**. You will associate these unmanaged workloads with their switches later.
See the Security Policy Guide for information on adding unmanaged workloads.
3. Define the switches and associated workloads, by selecting **Infrastructure > Switches**.
4. Click **Add**.
5. Enter the details in the displayed fields as described in the table below.
6. After entering or selecting values for all the required fields, click **Save**.

Fields in the **PCE web console > Infrastructure > Switches > Add Switch** page:

Field Name	Description	Re-quired	Notes
NEN host-name	FQDN of the NEN that runs the NEN service	Yes	This field is populated with the FQDN of your NEN. You cannot edit this field.
Description	Description of the NEN service	Yes	This field is populated with "Illumio Network Enforcement Node" and the FQDN of your NEN. You cannot edit this field.
Switch Name	A free-form, mnemonic name of your choice for the switch	Yes	Make this name easy to remember and distinguishable from other switch names.
Switch IP	IP address of the switch	Optional	IP address of the switch that is configured to send flow data to the NEN.
Manufacturer	Name of the switch manufacturer	Yes	Select the manufacturer.
Model	Model number of the switch	Yes	Select your model.
Interfaces	Defined interfaces on the switch	No	If you're monitoring flows, this corresponds to the interface name you defined in the switch flow configuration. That is, it's the name of the interface on the switch which you want to generate ACLs or monitor traffic. You can also add interfaces that are not being flow-monitored.
Workloads	Names of workloads connected to the switch's defined interfaces	Yes	Only those workloads assigned to the switch interfaces are secured. You can attach one or more workloads to an interface.
Monitor Traffic	SNMP ifIndex of the switch interface See the documentation for your switch.	Yes/No	This field is required if the interface is being flow-monitored. It is the ifindex of the interface in the switch's configuration.

NEN Switch Configuration Using REST API

To manage network switches reporting data flows to the NEN and to get the generated ACLs to enforce policies based on what's been defined in the PCE, you need to complete these tasks:

1. Get the list of switches and their details.
2. Generate the ACLs for one or all switches.
3. Print the ACLs in the desired format.

The sections below describe the manual steps you can insert into any script to automate this process.

List, Update, and Create Switches

Use these commands as needed to:

- Get a list of switches registered against the NEN
- Get a list of switches
- Update the credentials of an existing NVIDIA OVS switch
- Create a new NVIDIA OVS switch and configure credentials

cURL Command to Get a List of Switches and Details

To get the list of all the network switches registered against the NEN, run the following cURL command. It should return a list of switches with all the reported endpoints (ports, workloads) to the NEN.

```
cURL -u api_xxx:xxx -H "Accept: application/json" -X GET
https://mypce.domain.io:8443/api/v2/orgs/1/network_devices
```

cURL Command to a Get List of Switches

```
cURL -u
api_1853ebfcb1187acb4:9c2a381773a44e3a609448109278c02c4ec1fe597
f9643af71a832c0a8b0c0d0
-H "Accept: application/json"
-X GET https://mypce.domain.io:8443/api/v2/orgs
/1/network_devices
```

Response


```
[
  {
    "network_enforcement_node" : {
      "href" : "/orgs/1/network_enforcement_nodes/
f64e78b7-2917-409f-9093-
9d6ddaa35799"
    },
    "href" : "/orgs/1/network_devices/f07a077a-70ad-4b57-
b82a-f1d204fcfd99",
    "configure" : false,
    "network_endpoints" : [
      {
        "href" : "/orgs/1/network_devices/
f07a077a-70ad-4b57-b82a-
f1d204fcfd99/network_endpoints/1ff6f037-d644-438e-
ab32-019a45a7d8d5"
      },
      {
        "href" : "/orgs/1/network_devices/
f07a077a-70ad-4b57-b82a-f1d204fcfd99/
network_endpoints/dd687e16-6998-4a39-8bde-
a7fb445f18d9"
      },
      {
        "href" : "/orgs/1/network_devices/
f07a077a-70ad-4b57-b82a-f1d204fcfd99/
network_endpoints/7345aed3-1fbd-4596-ada9-
f6cbfb361dfe"
      },
      {
        "href" : "/orgs/1/network_devices/
f07a077a-70ad-4b57-b82a-f1d204fcfd99/
network_endpoints/be58f614-7cc7-4132-
a409-97ea8334dfef"
      }
    ],
    "enforcement_instructions_data_timestamp" :
"2019-05-06T15:45:02Z",
    "enforcement_instructions_data_href" : "/orgs/1/
datafiles/
49b11cf6-d6f9-4efc-8cb2-c1a444cb9c02",
    "supported_endpoint_type" : "switch_port",
    "config" : {
      "model" : "9000",
      "name" : "cisco-n9k",
      "rules_format" : "cli",

```

```

        "ip_address" : "10.1.2.3",
        "device_type" : "switch",
        "manufacturer" : "Cisco"
    },
    "status" : "unmonitored"
}
]

```

cURL Command to Update the Credentials of an Existing NVIDIA OVS Switch

(Note: Applicable for Illumio NEN release 2.7.0 and later. For more information, see [this integration topic](#).)

```

curl -X PUT /orgs/{org_id}/network_devices/{uuid}
-u $KEY:$TOKEN
-H 'Content-Type: application/json'
-d @- <<EOF
{
    "port": 22,
    "credentials": {
        "username": "network.admin",
        "password": "mysecretpassword"
        "type": "cli"
    }
}
EOF

```

cURL Command to Set Credentials of a New NVIDIA BlueField®-3 DPU featuring OVS

(Note: Applicable for Illumio NEN release 2.7.0 and later. For more information, see [this integration topic](#).)

Use this POST command only if you want to create a new NVIDIA OVS switch and configure credentials. Otherwise, use the [PUT command above \[412\]](#) to update an existing switch.

```
cURL -X POST /orgs/{org_id}/network_enforcement_nodes/{uuid}/
network_devices/{uuid}
-u $KEY:$TOKEN
-H 'Content-Type: application/json'
-d @- <<EOF
{
  "name": "NVIDIA BlueField-3 DPU",
  "description": "Example Output of Illumio REST API to
integrate with NVIDIA BlueField-3 DPU",
  "device_type": "switch",
  "manufacturer": "NVIDIA",
  "model": "OVS",
  "ip_address": "192.168.100.1",
  "credentials": {
    "type": "cli",
    "port": 22,
    "username": "network.admin",
    "password": "mysecretpassword"
  },
  "rules_format": "cli"
}
EOF
```

Generate ACLs for Switches

cURL Command to Generate ACLs for a Specific Switch

To generate ACLs for a specific switch registered against the NEN, run the following cURL command:

```
cURL -u api_xxx:xxx -H "Content-Type: application/json" -d {}
-X POST https://mypce.domain.io:8443/api/v2/orgs/1/
network_devices/
xxxxxxx-xxxx-xxxx-xxxx-xxxxxxx/
enforcement_instructions_request
```



NOTE

Replace the xxx-...-xxx value with the value of the switch for which you intend to generate ACLs.

cURL Command Using Generate ACLs

```
cURL -u api_1853ebfcb1187acb4:9c2a381773a44e3a609448109278c02c4
ec1fe597f9643af71a832c0a8b0c0d0
-H "Content-Type: application/json" -d {}
-X POST https://mypce.domain.io:8443 -d {}
-X POST https://mypce.domain.io:8443/api/v2/orgs/1/
network_devices/
f07a077a-70ad-4b57-b82a-f1d204fcfd99/
enforcement_instructions_request
```

Result: Response with a 202 status code = Accepted.

The ACLs are generated on the NEN and are ready for use in a few minutes.



IMPORTANT

API POST Requirements

While sending a POST request, you must include the header (-H) flag and the data (-d) flag. Even if you do not have any data to send, you must insert an empty data flag, as shown in the above example.



NOTE

Illumio recommends that you insert a pause in any script to allow the NEN to generate the new ACLs for the specific switch. It takes approximately 30 seconds to generate all the ACLs.

The PCE will not send any update or acknowledgment to the REST client once it is finished generating the new ACLs for the switch.

Alternatively, you might want to generate ACLs for all the switches in your inventory to deliver them to your network team, by using the `all_devices: true` key-value pair in your JSON payload while sending the POST request.

To generate ACLs for all the switches registered against the NEN, run the following cURL command:

```
cURL -u api_xxx:xxx -H "Content-Type: application/json" -d
'{"all_devices": true}'
-X POST https://mypce.domain.io:8443/api/v2/orgs/1/
network_devices/multi_enforcement_instructions_request
```

Get List of ACLs



NOTE

An error log may be generated if the NEN encountered an issue generating ACLs or the PCE had an issue sending ACLs to the switch. If this occurs, send the error report to Illumio as instructed by Illumio Support.

To download ACLs for a specific switch registered against the NEN, get the updated value of the `enforcement_instructions_data_href` key. This value keeps changing because each time the NEN generates new ACLs for a switch, it is considered to be a new datafile.

1. To get the updated `enforcement_instructions_data_href` value for a network switch, run the following cURL command:

```
cURL -u api_xxx:xxx -H "Accept: application/json" -X
GET https://mypce.domain.io:8443/api/v2
/orgs/1/network_devices/
enforcement_instructions_data_href'
```

The above command returns a list of switches. You have to then parse the JSON output and filter on the `enforcement_instructions_data_href` key to get the updated value. You can use the [JQ tool](#) to filter outputs on any JSON file.

2. After you retrieve the updated value, use it in the following cURL command to get the generated ACLs:

```
cURL -u api_xxx:xxx -H "Accept: application/json" -X GET
https://mypce.domain.io:8443/api/v2
/orgs/1/datafiles/xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Replace the xxx-...-xxx value with the value of the `enforcement_instructions_data_href` key that you got by running the previous GET request.

Example of Get List of ACLs

```
cURL -u api_1853ebfcb1187acb4:9c2a381773a44e3a609448109278
c02c4ec1fe597f9643af71a832c0a8b0c0d0
-H "Accept: application/json" -X GET https://
mypce.domain.io:8443/
api/v2/orgs/1/network_devices/
enforcement_instructions_data_href' "/orgs/1/datafiles/
d1bdbb23-60c4-439e-bd74-ca0d03d959a7"
```

Output:

```
no ip access-list ILLUMIO_ACLS-Ethernet1-21-Inbound
p access-list ILLUMIO_ACLS-Ethernet1-21-Inbound
!---Inbound ACL Rules ---
    permit ip host 10.10.100.201 host 10.10.100.202
    permit ip host 10.10.100.201 host 10.10.100.203
    permit ip host 10.10.100.201 host 10.10.100.204
    permit tcp any any established
    permit udp any eq 68 any eq 67
    permit udp any range 1024 65535 any eq 53
    exit
    ...
no ip access-list ILLUMIO_ACLS-VLAN-20-Outbound
ip access-list ILLUMIO_ACLS-VLAN-20-Outbound
!---Outbound ACL Rules ---
    permit ip host 10.10.100.201 host 10.10.100.204
    permit ip host 10.10.100.202 host 10.10.100.204
    permit ip host 10.10.100.203 host 10.10.100.204
    permit tcp any any established
    permit udp any eq 67 any eq 68
    permit udp any eq 53 any range 1024 65535
    exit
```

Apply Policy for Switches



NOTE

In the NEN Guide, the term "switch" refers to both switches and routers.

To apply the security policy for switches, you need to:

1. [Create security policy \[417\]](#).
2. [Generate and download ACLs \[417\]](#).
3. [Apply ACLs on the switch \[418\]](#).
4. [Mark ACLs as applied \[421\]](#).

STEP 1: Create Security Policy

In the PCE web console, create label-based policies for the workloads that are bound to the switch ports. For information on how to create policies, see the Security Policy Guide.



NOTE

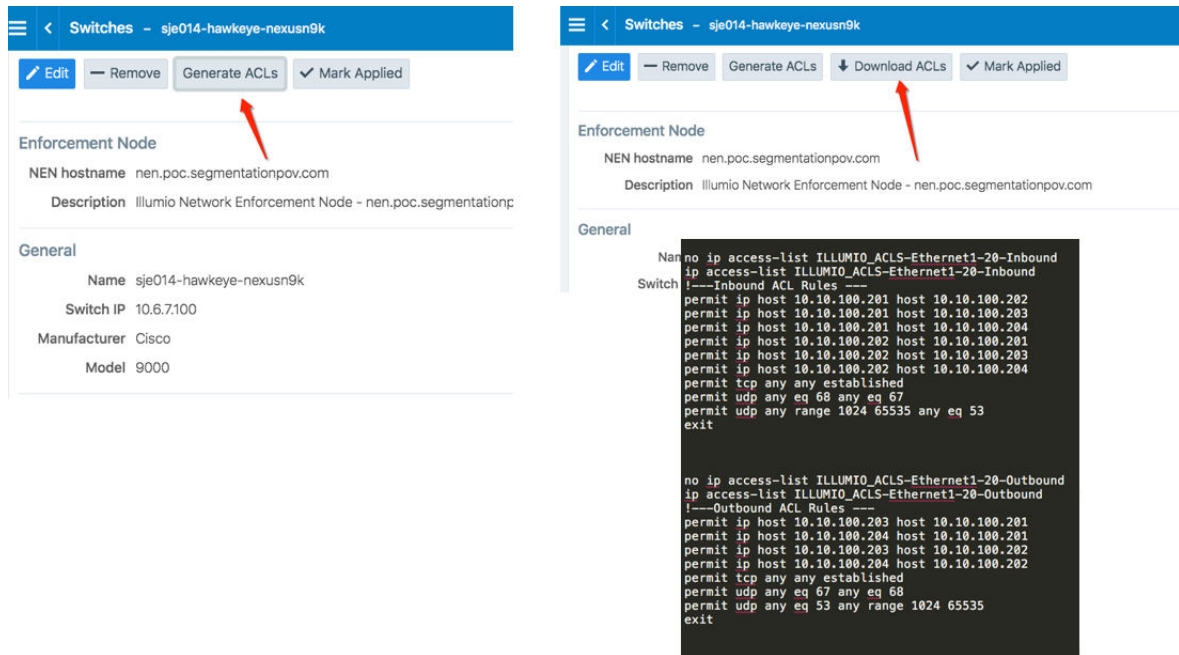
Make sure you provision the policies before generating the ACLs.

STEP 2: Generate and Download ACLs

After you have created new policies for the NEN-managed workloads and provisioned them in the PCE, you can generate the ACLs. The PCE writes those ACLs to its local files.

Create the associated switch ACLs as follows:

1. Log into the PCE web console.
2. From the PCE web console menu, choose **Infrastructure > Switches**.
3. Select the switch.
4. On the Switches page, click **Generate ACLs**. It takes a moment for the ACL generation to complete.
5. After the ACLs have been generated, click **Download ACLs** to download a .txt file of the updated ACLs from your web browser.
6. Go to the Downloads folder on your computer and open the .txt file. This file contains a list of inbound and outbound ACLs.

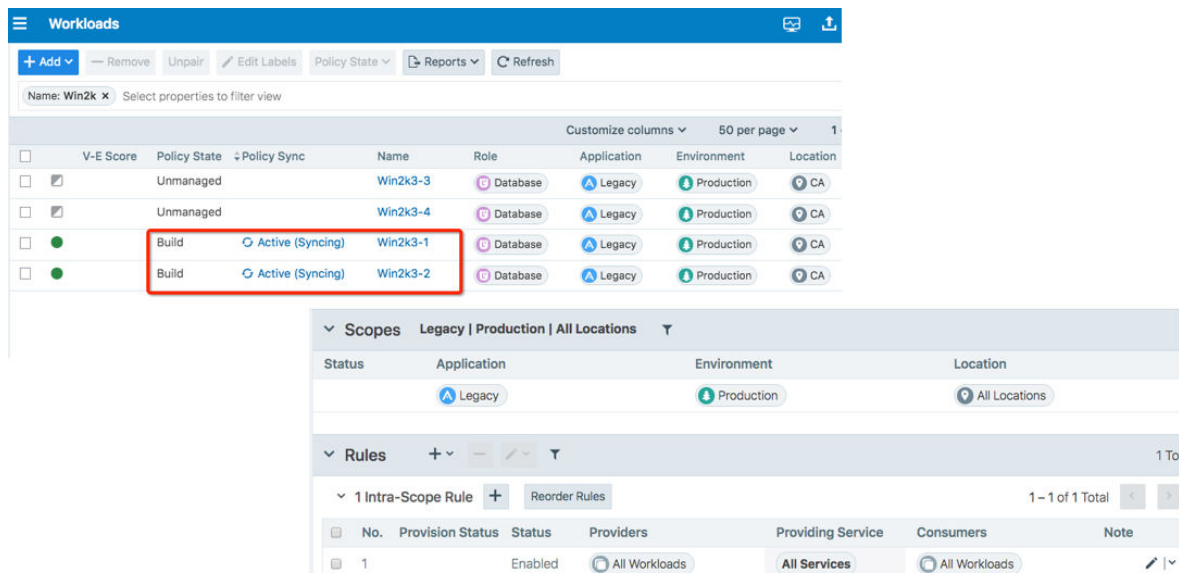


The workloads associated with the switch remain in an **Active (Syncing)** policy sync state until you click the **Mark Applied** button on the Switches page.



NOTE

You might see a “Syncing” notification appear in the PCE web console until you mark the ACLs as **Applied**.



STEP 3: Apply ACLs on the Switch

After generating the ACLs from the NEN, you can copy the text of the ACLs from the PCE-generated files and paste them into the switch's command line to configure the ACLs on the switch.

Example of Inbound and Outbound ACLs



NOTE

Although the NEN supports several switches, this topic provides ACL examples only for the **Cisco Nexus 9000**. For detailed information about your switch, consult vendor documentation.

Cisco Nexus 9000 example: The following ACLs are generated for only Ethernet 1/7 interface of a Cisco Nexus 9000:

```
no ip access-list ILLUMIO_ACLS-Eth1-7-Inbound
ip access-list ILLUMIO_ACLS-Eth1-7-Inbound
!---Inbound ACL Rules ---
permit tcp any any established
permit udp any eq 68 any eq 67
permit udp any range 1024 65535 any eq 53
exit

no ip access-list ILLUMIO_ACLS-Eth1-7-Outbound
ip access-list ILLUMIO_ACLS-Eth1-7-Outbound
!---Outbound ACL Rules ---
permit tcp host 10.6.4.94 host 10.0.17.17 eq 5432
permit tcp any any established
permit udp any eq 67 any eq 68
permit udp any eq 53 any range 1024 65535
exit
```



NOTE

By default, the NEN generates ACLs to allow basic infrastructure services such as DHCP and DNS from all IP addresses in addition to the policies defined on the PCE. You cannot prevent DNS and DHCP ACLs from being generated by the NEN. If your network administrator does not want DHCP or DNS rules added to the switch, you can remove those ACL lines while copying the ACLs over to the switch.

See your switch documentation for steps on how to apply ACLs to your particular switch.

Cisco Nexus 9000 example: To configure the ACLs on to the switch, the network administrator must log in to the command line and go in to configuration mode. Once in configuration mode, you can copy the ACLs from the NEN into the switch command line as shown in this example:

```
nexus9000# conf
Enter configuration commands, one per line. End with CNTL/Z.
nexus9000(config)# no ip access-list ILLUMIO_ACLS-Eth1-7-
Inbound
nexus9000(config)# ip access-list ILLUMIO_ACLS-Eth1-7-Inbound
nexus9000(config-acl)# !---Inbound ACL Rules ---
nexus9000(config-acl)# permit icmp host 10.0.17.17 0.0.0.0/0
nexus9000(config-acl)# permit tcp any any established
nexus9000(config-acl)# permit udp any eq 68 any eq 67
nexus9000(config-acl)# permit udp any range 1024 65535 any eq
53
nexus9000(config-acl)# exit
nexus9000(config)#
nexus9000(config)#
nexus9000(config)#
nexus9000(config)# no ip access-list ILLUMIO_ACLS-Eth1-7-
Outbound
nexus9000(config)# ip access-list ILLUMIO_ACLS-Eth1-7-Outbound
nexus9000(config-acl)# !---Outbound ACL Rules ---
nexus9000(config-acl)# permit tcp host 10.6.4.94 host
10.0.17.17 eq 5432
nexus9000(config-acl)# permit tcp 0.0.0.0/0 host 10.0.17.17 eq
22
nexus9000(config-acl)# permit tcp any any established
nexus9000(config-acl)# permit udp any eq 67 any eq 68
nexus9000(config-acl)# permit udp any eq 53 any range 1024
65535
nexus9000(config-acl)# exit
```

After you have added the ACLs to the switch, you must apply them to an interface on the switch as either a PACL or a RACL. To take advantage of both inbound and outbound ACLs, this example uses RACLs. If the workload is directly attached to a Layer 2 interface (switchport mode access), you must apply the RACL to the VLAN/SVI interface. If the workload is directly attached to a Layer 3 interface (no switchport), you can apply ACL directly to the physical interface (or port).

Optional command to verify the interface configuration

```
# show run interface ethernet x/y
```

Example of Command Usage for Ethernet 1/7

Cisco Nexus 9000 example:

```
nexus9000(config)# show run int eth1/7

!Command: show running-config interface Ethernet1/7
!Time: Tue Oct 16 17:32:52 2018

version 7.0(3)I5(1)

interface Ethernet1/7
switchport
switchport access vlan 17
no shutdown
```

This interface is a Layer 2 interface and is part of VLAN 17. You must apply this to VLAN 17 interface (SVI).

**NOTE**

For L2 interfaces, the VLAN must have a Switch Virtual Interface (SVI).

```
nexus9000(config)# interface vlan 17
nexus9000(config-if)# ip access-group ILLUMIO_ACLS-Eth1-7-
Inbound in
nexus9000(config-if)# ip access-group ILLUMIO_ACLS-Eth1-7-
Outbound out
```

Result: The ACLs are now configured on the switch and any communication through VLAN 17 is denied if it is not permitted in the ACLs.

STEP 4: Mark ACLs as Applied

You have to inform the PCE after you have loaded the ACLs because the NEN service does not configure the generated ACLs on the switch.

1. Log into the PCE web console.
2. From the PCE web console menu, choose **Infrastructure > Switches**.
3. Select the switch.
4. On the Switch page, click **Mark Applied**.

Enhance network security for Top Of Rack routers using Illumio NEN 2.7.0 and Cisco IOS XR



NOTE

In the NEN Guide, the term "switch" refers to both switches and routers.

Starting with NEN release 2.7.0, you can assign subnets or CIDR blocks to unmanaged workloads, expanding beyond the previous limitation of single IP addresses. In this context, an unmanaged workload may represent a subnet, a Layer 3 interface, or a collection of workloads. As before, the Network Enforcement Node (NEN) generates access control lists (ACLs) which are eventually installed on your network appliance—specifically, an IOS XR series router—to secure the ports connected to these workloads.

How it works

1. Deploy the Illumio Network Enforcement Node (NEN) on an RHEL or CentOS workload near your Cisco IOS XR series router and pair it with your Policy Compute Engine (PCE).
2. Configure the NEN to receive flow data from the router. The NEN forwards this data to the PCE, where you can visualize it using built-in tools.
3. Use the PCE's visualization tools to analyze router traffic and define your security policy. The NEN translates this policy into Access Control Lists (ACLs) for distribution to the router for enforcement.
4. ACLs generated by the NEN are loaded on to the router to control traffic at the configured interface(s).

Prerequisites

- Cisco IOS XR series router
- Illumio PCE release 25.3.0 (or later SaaS PCE)
- Illumio NEN release 2.7.0 (or later)

Specifications

Scale

- 70-80 of the following Cisco IOS-XR router models:
 - Cisco 8201-32FH - IOS-XR 7.7.2
 - Cisco NCS 55A1 - IOS-XR 7.7.2
 - Cisco ASR9010 - IOS-XR 6.8.2
 - Cisco ASR9922 - IOS-XR 6.8.2
- 2, 4, 10, 20 Interfaces
- 200 entries per router interface

Supported address space

- IPv4 & IPv6 subnets and single addresses

Flow data monitoring protocols

- sFlow
- NetFlow (beginning in NEN 2.7.0)
- IPFIX (beginning in NEN 2.7.0)

Configuration steps

1. Install the NEN on a CentOS or RHEL workload in your environment. See [Install and Activate the NEN \[366\]](#).
2. Pair and activate the NEN. See [Obtain a Pairing Key and Activate the NEN \[373\]](#).
3. Configure the router to communicate with the NEN. See [Configure Switches for the NEN \[406\]](#) or [NEN Switch Configuration Using REST API \[409\]](#).
4. Define security policy, generate ACLs, and send ACLs to the router. See [Apply Policy for Switches \[416\]](#).

Integrate the NEN with the NVIDIA BlueField®-3 DPU featuring OVS



NOTE

In the NEN Guide, the term "switch" refers to both switches and routers.

This solution integrates the NVIDIA® BlueField®-3 Data Processing Unit (DPU), featuring Open vSwitch (OVS), with Illumio's visualization tools and policy engine to prevent lateral movement and contain threats in real time to and from managed host VMs or within operational technology (OT) environments.

How it works

For configuration details, see [OVS integration steps \[425\]](#).

1. Deploy the Illumio Network Enforcement Node (NEN) on a CentOS or RHEL workload within the supervisory layer of your OT environment and pair it with your Policy Compute Engine (PCE).
2. Configure the NEN to receive flow data from the OVS running on the DPU. The NEN forwards this data to the PCE where you can visualize it using built-in tools.
3. Use the PCE's visualization tools to analyze traffic flowing through the switch and define your security policy. The NEN translates this policy into Access Control Lists (ACLs) for distribution to the OVS for enforcement.
4. After the NEN-generated ACLs are loaded onto the OVS, enforcement occurs at the DPU to control traffic at the switch interface.
5. The DPU's hardware offload capabilities accelerate policy enforcement, enabling the OVS to apply security rules with high efficiency and minimal latency.

Key benefits

- Integrating the NEN with OVS enables visibility and policy enforcement for traffic within and between IT and OT layers, allowing you to visualize all traffic to and from OT systems. Illumio's flexible labeling architecture helps you understand how your assets communicate. NEN-generated ACLs convert your segmentation policies into ACLs that you install on the OVS to secure your OT/IT infrastructure.
- Integrating the NEN with OVS through the PCE API is simple: enter the IP address and credentials for the OVS switch (see note below) and the NEN automatically discovers the switch configuration, programs flow monitoring on the switch, discovers and creates workloads in the PCE, and programs the ACLs on the OVS.



IMPORTANT

The user credentials you provide for the OVS must allow access to the **ovs-vsctl** and **ovs-ofctl** commands either through the user login or password-less **sudo** access.

- Microsegmentation and policy enforcement tasks are offloaded from the CPU to the DPU to minimize the impact on application performance.

Limitation

- Only generates IPv4 port, protocol, and IP address rules.

Prerequisites

- Illumio NEN release 2.7.0 (or later)
- NVIDIA® BlueField®-3 DPU with OVS configured and running
- Review the topics [Overview of Switch Integration \[399\]](#) and [Supported Switches and Configurations \[403\]](#).

Specifications

Supported flow data monitoring protocols:

- sFlow
- NetFlow (beginning in NEN 2.7.0)
- IPFIX (beginning in NEN 2.7.0)

OVS integration steps

1. Install the NEN on a Centos or RHEL workload in your OT environment. See [Install and Activate the NEN \[366\]](#).
2. Pair and activate the NEN. See [Obtain a Pairing Key and Activate the NEN \[373\]](#).
3. Configure the OVS to communicate with the NEN.



IMPORTANT

The user credentials you provide for the OVS must allow access to the **ovs-vsctl** and **ovs-ofctl** commands either through the user login or password-less **sudo** access.

See: [Configure Switches for the NEN \[406\]](#) or [NEN Switch Configuration Using REST API \[409\]](#).

4. Define security policy, generate ACLs, and send ACLs to the OVS. See [Apply Policy for Switches \[416\]](#).

Integrate the Illumio PCE with IBM i running Precisely Assure Security



NOTE

For the most up-to-date information about configuring Precisely Assure Security for this integration, see the Precisely document [Commands for the Assure SAM integration with Illumio](#).

This topic describes how to integrate **IBM i** system(s) running Precisely Assure Security with your Illumio PCE. This integration differs from the typical switch integration in the following ways:

- Although the **IBM i** is not a switch, this integration uses the PCE switch integration user interface to perform the integration.
- Instead of generating ACLs as you would do when integrating a switch, you'll generate a Precisely-formatted CSV file to configure relevant policy on your **IBM i** system that is running Precisely.
- Flow information is collected by Assure Security and sent to the Illumio Flowlink server (see [Illumio Flowlink Configuration and Usage Guide](#)).

Add Unmanaged Workloads and IBM i Definitions

To create a security policy, add unmanaged workloads representing each **IBM i** system included in the PCE policy. A set of csv data is generated for each configured **IBM i** unmanaged workload. To define the **IBM i** system and the workloads attached to them as unmanaged workloads in the PCE web console, complete the following steps:

1. Log into the PCE web console.
2. Define the **IBM i** system as unmanaged workloads by adding an unmanaged workload through the PCE. You will associate these unmanaged workloads with their **IBM i** Precisely integration later. (For more about adding unmanaged workloads, see [Adding Unmanaged Workloads](#) in the Security Policy Guide.



NOTE

The workload name must be the IBM i system name in upper-case.

- Illumio Core **release 22.5** and earlier: Go to **Workloads and VENs > Workloads** . . . click **Add > Add Unmanaged Workload**
 - Illumio Core **release 23.2** and later: Go to **Servers & Endpoints > Workloads** . . . click **Add > Add Unmanaged Workload**
3. Define the **IBM i** integration and associated workloads in the PCE by going to **Infrastructure > Switches**.
 4. Click **+Add**.
 5. Enter details:
 - **NEN hostname:** This field is populated with the FQDNs of the NENs paired with your organization's PCE. Select the appropriate NEN.
 - **Description:** This field is populated with "Illumio Network Enforcement Node" and the FQDN of the NEN. You cannot edit this field.
 - **Switch Name:** Enter a unique name that's easy to remember.
 - **Switch IP:** IP address of the **IBM i** system.
 - **Manufacturer:** Select IBM.
 - **Model:** Select Precisely.
 6. Click **Save**.
 7. Click **Interfaces**.
 8. Click **Edit** and then enter details:
 - **Total Interfaces:** 1
 - **Interface 1:** Enter a name, such as interface 1.
 - **Workloads:** Select the unmanaged workload representing the appropriate **IBM i** system. Only workloads assigned to the **IBM i** system interfaces are secured. You can attach one or more workloads to an interface.
 - **Monitor Traffic:** Ignore this setting. It doesn't apply to this integration.
 9. Click **Save**.



NOTE

If your unmanaged **IBM i** system has two or more network interfaces, the generated ACL file will include duplicate entries for Inbound Rules, one pair of entries for each interface. This is expected behavior.

Fields in the **PCE web console > Infrastructure > Switches > Add Switch** page:

Summary

Interfaces

Edit

Remove

Generate ACLs

Mark Applied

Enforcement Node

NEN hostname

Description

Illumio Network Enforcement Node

General

Name

Test AS400

Switch IP

10.

Manufacturer

IBM

Model

Precisely

Flowlink 2.0.0 Configuration and Usage

This guide describes how to install, configure, and use Flowlink, an Illumio standalone application that can be used with the Illumio Core to collect network flow data from different types of network sources, such as switches, routers, F5 load balancers, cloud monitoring tools, and syslog exporters.

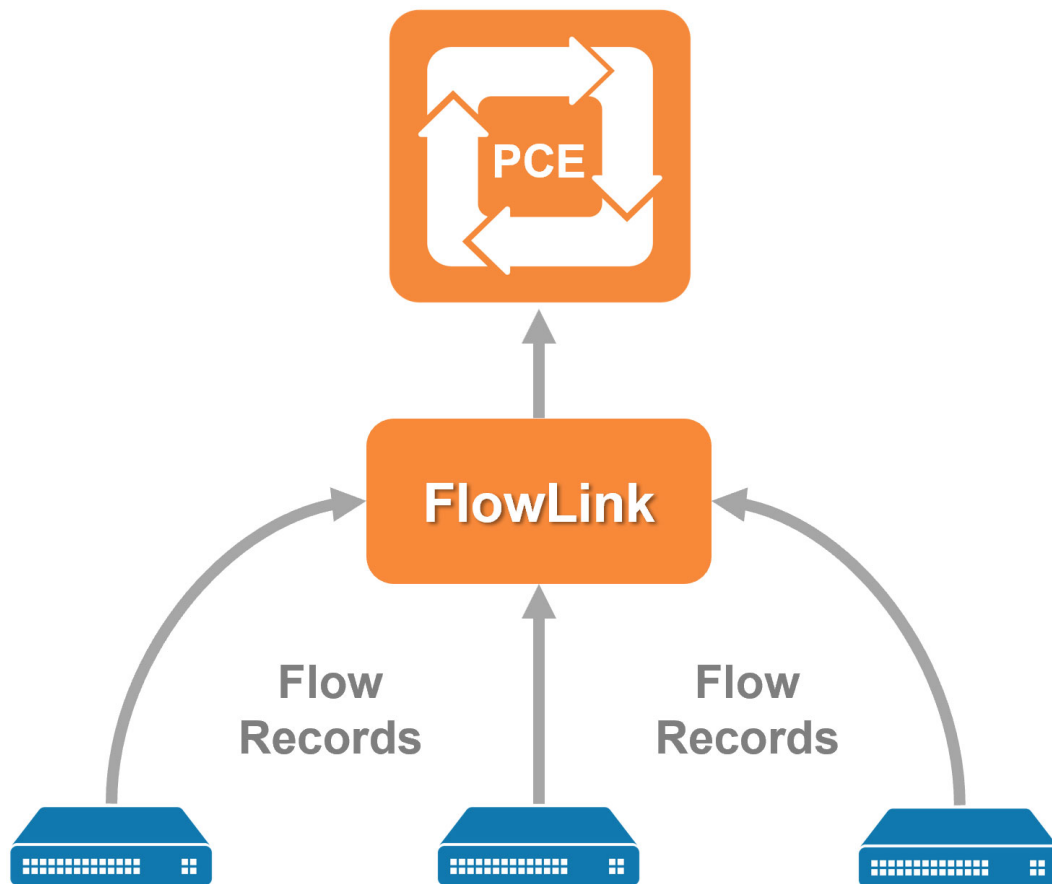
- [Overview of Flowlink \[429\]](#)
- [Scale and Limitations \[431\]](#)
- [Configure Flowlink \[434\]](#)
- [Flowlink Usage \[451\]](#)
- [Troubleshooting \[460\]](#)

Overview

The Flowlink application normalizes and aggregates the network flow data that it collects from different types of network sources into a format that can be ingested by the PCE for use by traffic data applications. It does not resolve any flow data source and destination IP addresses in to the PCE workloads. The PCE displays the flow in Illumination and marks the policy decision as 'unknown'. Flowlink is supported on standard PCE clusters and also on Supercluster.

How Flowlink Works

Flowlink can receive the flow data by connecting to a data source provided by you and adheres to your organizations' data format. It may consume flows at a rate that is slower than the source speed. Therefore, the flow sender caches the flow data for 48 hours or more. If the PCE is unable to accept flow data because of the rate of flow or availability issues, Flowlink caches the data locally to a disk for a configurable period of time or disk space and retries periodically (user-configurable number of minutes). It aggregates data flows and sends them to the PCE once every configurable number of minutes. It does not have access to the PCE data and therefore no knowledge of workloads, virtual services, and other objects.



NOTE

Flowlink version 1.1.0 does not support a High Availability (HA) configuration. You will have to monitor Flowlink and ensure that you restart it on failure.

Supported Flow Record Formats

The following types of flow records are supported:

- AWS VPC flows
- IPFIX v10
- NetFlow v5, v7, v9, and v10

- sFlow v5
- Text (customizable parser configured by user, for example, Syslog or Kafka)

What's New in Flowlink 2.0.0

This section describes Flowlink 2.0.0 support for the following:

Support for Azure Event Hub (managed Kafka)



NOTE

Currently, this feature is accessible to only a limited number of organizations.

Flowlink 2.0.0 adds support for streaming flows to Azure Event Hub, enabling the PCE to ingest network flow data at higher throughput while unifying device identity and flow visibility within a single control plane. For details, see `pce_kafka` in [Flowlink Key-Value Parameters](#).

Scale and Limitations

This section lists the supported scale and known limitations to be considered while using Flowlink.

PCE

- The PCE processes up to 10K unique flows/second. This is the total number of Flowlink and VEN flows received by the PCE.
- The PCE handles up to 20 concurrent POSTs.
- The PCE allows a maximum file size of 100MB per POST.
- For each IP address that exists in your data flows, you need to create corresponding unmanaged workloads in the PCE if you want to see those traffic flows in Illumination. Else, those flows will not be displayed.

Flowlink

- Flowlink supports multiple flow data sources.
- The maximum number of sources per Flowlink is not reported. As a best practice, consider one source per Flowlink.

- Flows with Class D addresses are ignored.
- Flowlink is not installed as a service, nor does it support a High Availability (HA) configuration. As such, it doesn't restart automatically if the host fails or is rebooted. In those cases, you need to restart Flowlink manually.
- The following two limitations are generic traffic limitations with Illumination and are not specific to Flowlink:
 - At least one IP address in the reported flow must match an IP address of a workload object (managed or unmanaged).
 - If a virtual service object and workload object have the same IP address, flow lines will always be drawn to the virtual service.

Flowlink Requirements

This topic details the prerequisites and hardware capacity requirements for configuring and using Flowlink.

Prerequisites

Make sure you possess and/or have configured the following:

- CentOS or RHEL server
- Root privileges to the server
- Flowlink RPM downloaded from the [Illumio Support](#) site
- PCE with Service Account API Key and Secret



IMPORTANT

You must have the Org User role to configure Flowlink.

CPU, Memory, and Storage Requirements

To use Flowlink, your hardware must meet the capacity requirements detailed in this section.

Machine Type	Cores/Clock Speed ¹	RAM per Node ²	Storage Device Size ³ and IOPS ⁴
Flowlink 2500 workloads	• 2 cores • Intel® Xeon(R) CPU E5-2695 v4 at 2.10GHz or equivalent	8 GB	• 1 x 20 GB • 100 IOPS per device

Footnotes:

¹ CPUs:

- The recommended number of cores is based only on physical cores from allocated CPUs, irrespective of hyper-threading or virtual cores. For example, in AWS one vCPU is only a single hyper-thread running on a physical core, which is half a core. 16 physical cores equates to 32 vCPUs in AWS.
- Full reservations for vCPU. No overcommit.

² Full reservations for vRAM. No overcommit.

³ Additional disk notes:

- Storage requirements for network traffic data can increase rapidly as the amount of network traffic increases. Allocating a separate, large storage device for traffic data can accommodate these rapid changes without potentially interrupting the service.
- Network File Systems (NFS) is not supported.

⁴ Input/output operations per second (IOPS) are based on 8K random write operations. IOPS specified for an average of 300 flow summaries (80% unique **src_ip**, **dest_ip**, **dest_port**, **proto**) per workload every 10 minutes. Different traffic profiles might require higher IOPS.

Flowlink Storage Partitioning

Storage Device	Partition mount point	Size to Allocate	Notes
Device 1, Partition A	/	20 GB	Logrotate must be configured to limit the disk consumption of Flow & System Logs.

Configure Flowlink

This topic describes how to configure Flowlink.

You configure Flowlink using a YAML file that defines its runtime behavior, while an included JSON schema validates the configuration to ensure all parameters are complete, well-formed, and supported. When FlowLink starts (or restarts), it automatically validates the YAML against the JSON schema.

If validation succeeds:

- FlowLink parses flows according to the configured consumers.
- Aggregates them on the defined interval.
- Posts them to the PCE using the provided credentials.

If validation fails:

- FlowLink logs explicit configuration errors.
- No data is ingested or sent.

STEP 1: Install the Flowlink RPM

1. Log in as a root user.
2. Install the RPM.

The default install location is: `/usr/local/bin/`

Standard installation:

```
sudo su
rpm -ivh illumio-flowlink-x.x.x-yy.x86_64.rpm
```

For FIPS compliance (applies only to Flowlink version 1.2 and later; see [\[434\]](#) for more information):

```
sudo rpm -ivh --nodigest illumio-
flowlink-1.2.0-104.x86_64.rpm
```


**IMPORTANT**

Only the [Install Flowlink RPM \[434\]](#) step requires root user login.

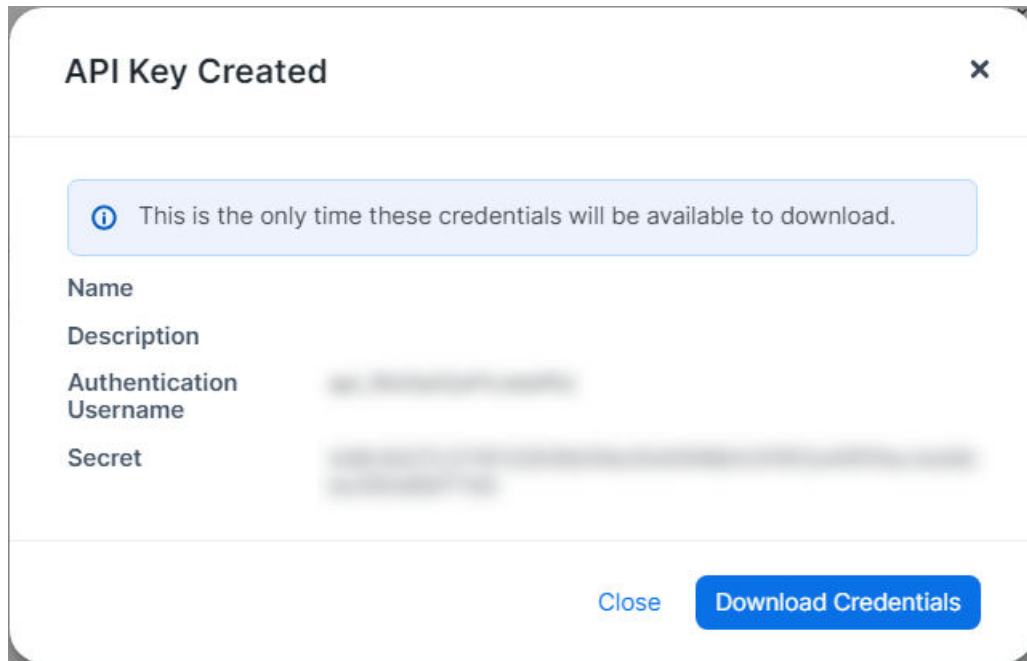
Illumio users logged in with any role can perform the steps in [STEP 2: Create a Service Account API Key \[435\]](#), Create YAML Configuration File, and [Run Flowlink \[441\]](#).

In the following sections, `/home/employee` directory is used as an example. The `api_info` file should be in a directory writable by the user, for example in the `/home/employee` directory.

STEP 2: Create a Service Account API Key

- There are two ways to create a Service Account API key for Flowlink:
 - Through the API. See [API Keys](#).
 - Through the PCE Web Console (described in the procedure below).
- The Org ID value is not shown when you create a Service Account API key.
- Service accounts are always organization-based and specific to a PCE. While creating a service account, users create their permissions and an `api_key` is created implicitly. Deleting a service account removes its permissions and all associated API keys.

1. In the PCE UI, go to **Access > Service Accounts**.
2. Click **Add** and configure settings.
 - **Name**
 - **Description** (optional)
 - **Access Restriction:** None.
 - **API Key expiration:** Keep the default or choose a different option.
 - **Roles and Scopes:** Select Global Administrator. The **All** is chosen automatically and cannot be changed.
3. Click **Save**.
4. When the *API Key Created* dialog appears, preserve the credentials (make a note or download them).



5. Copy the values of the **Authentication Username** and **Secret** into to a text file on the Flowlink server.
Use a space to separate the key and secret. For example:

```
api_XXXXXXXXXXXXXXXXX
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
```
6. Copy the absolute path of the file PCE API file `/home/employee/api_info`. You will need it in the Flowlink configuration file.

STEP 3: Configure HTTP/HTTPS Proxy (if needed)



IMPORTANT

Perform this step only when FlowLink is isolated from the internet and needs to use a proxy to access the PCE.

Supported on Flowlink 1.3.0 and later.

When Flowlink is running behind a proxy or in a corporate network and the PCE is in the cloud, Flowlink can access the PCE via HTTP/HTTPS proxy configurations.

The following configuration parameters are available to define an HTTP/HTTPS proxy:

```
proxy_config:
  https_proxy: <HTTPS_PROXY>
  http_proxy: {} <HTTPS_PROXY>{}
```

The following is an example of a Flowlink YAML configuration file:

```
proxy_config:
  https_proxy: http://proxy.corporate.com:3128
  http_proxy: http://proxy.corporate.com:3128
```

In the example above, the HTTP/HTTPS proxy is running on FQDN `proxy.corporate.com`{ `port: 3128`}.

STEP 4: Configure a Flowlink YAML File

Configure Flowlink by defining its runtime parameters in a YAML file. The included JSON schema validates the configuration to ensure all parameters are complete, well-formed, and supported.



NOTE

Refer to the `/usr/local/illumio/flowlink_config_schema.json` file provided with the Flowlink RPM for definitions of all the fields supported by the Flowlink YAML configuration file.

1. In the `/home/employee` directory, create a YAML configuration file. You can find an example yml file at `/usr/local/illumio/config.yml.example`.
2. Enter the parameters. (See [Flowlink Key-Value Parameters \[438\]](#) for details).

Example configuration

The following configuration listens for NetFlow on UDP 2055 from any data source. The absolute path is: `/home/employee/config.yml.netflow`

```
pce_addr: mypce.example.com:8443
api_key: $cat /home/employee/api_info
data_directory: /home/employee
aggregation_minutes: 10
consumers:
  - name: netflow
    parser:
      type: netflow
    connectors:
      - type: udp
        properties:
          ports: '2055'
```

Flowlink Key-Value Parameters

This table details the key-value parameters in Flowlink's YAML configuration file.

Parameter	Required/ Optional	Description
<code>aggregation_minutes</code>	Optional	The interval (in minutes) in which flows are aggregated and sent to the PCE. Default interval: 10 Minimum allowed interval: 5 Maximum allowed interval: 60 For example: <code>aggregation_minutes: 10</code>
<code>api_key</code>	Required	API key and secret of the PCE. This allows Flowlink to POST flows to the PCE. The API key and secret can be copied into a file. You can run a script to <i>cat</i> the contents of that file. In the example below, a file called <i>api_info</i> is created which contains the PCE API key and secret. For example: <code>api_key: \$cat /home/employee/api_info</code>
<code>consumers</code>	Required	A list of dictionaries. It requires a name, parser, and connector. Flowlink configuration supports one or many consumers (flow types). For more details about configuring the ingested flow types, see Ingested Flow Types [443].
<code>data_directory</code>	Required	The pathname of a directory where Flowlink can store any unsent data flow files or any restart information. For example: <code>data_directory: /home/employee/</code>
<code>data_directory_size_mb</code>	Optional	The maximum size (in Megabytes) of data that can be stored in the data directory before being pruned. Default: 500 Minimum value: 100 For example:

Parameter	Required/ Optional	Description
<code>file_retention_hours</code>	Optional	<code>data_directory_size_mb: 200</code> The maximum number of hours unsent data flow files will be stored before being pruned. Default: 24 Minimum: 4 For example: <code>file_retention_hours: 8</code>
<code>metrics_print_seconds</code>	Optional	The frequency (in seconds) at which the metrics information is printed. Default: 60 Minimum: 15 For example: <code>metrics_print_seconds: 60</code>
<code>org_id</code>	Required for SaaS	The org id to which the flow data will be posted. The default id is 1. For example: <code>org_id: 1</code>
<code>pce_addr</code>	Required	FQDN of the PCE and port. For example: <code>pce_addr: https://mypce.example.com:8443</code>
<code>pce_kafka</code>	Optional (unless you're configuring certain integrations (for example, Armis and the PCE)).	Allows you to stream flows to Azure Event Hub (instead of directly to the PCE) for use in Insights and Segmentation, and to configure the parameters included in flow headers.


NOTE

Currently, this feature is accessible to only a limited number of organizations.

Parameter	Required/ Optional	Description
		Applies to Flowlink 2.0.0 and later: If you are configuring Flowlink as part of an integration with the PCE and Armis, make sure to add the following to your Flowlink YAML file at the root level: • <code>pce_kafka:</code> • <code>installation_id:</code> armis-site For details, see About the Illumio and Netflow/sFlow Integration.

STEP 5: Run Flowlink

1. To manage Flowlink, use the following commands:

```
illumio-flowlink-ctl start --config <path to config file> [--log-file <path to log file>]
illumio-flowlink-ctl stop
illumio-flowlink-ctl status
```

The default path for the log file is <data_directory specified in config file>/**flowlink.log**

2. To start Flowlink, use the `illumio-flowlink-ctl start` command. Make sure that you include the `--config` option in the start command, which will begin running the program in the background. Example with expected output:

```
illumio-flowlink-ctl start --config /home/employee/
config.yaml.netflow
```

OUTPUT TO CONSOLE

```
Checking Flowlink started successfully.
OK.
```

Output logs can be found at: /home/employee/flowlink.log

OUTPUT IN LOG FILE (/home/employee/flowlink.log)

```
2020-03-11T09:58:51.173203-07:00 Waiting for signal
2020-03-11T09:58:51.330757-07:00 Starting Data source:
netflow
2020-03-11T09:58:51.331162-07:00 Listening for netflow
messages on udp port: 2055
2020-03-11T09:58:51.332929-07:00 Reporting flows every 10
minutes
```

3. To stop Flowlink, use the `illumio-flowlink-ctl stop` command.

Example with expected output:

```
illumio-flowlink-ctl stop
```

OUTPUT ON CONSOLE

```
/illumio-flowlink-ctl stop
```

```
Stopping Flowlink: ..... Stopped.
```

OUTPUT IN LOG FILE (/home/employee/flowlink.log)

```
2020-03-11T09:58:57.097817-07:00 Got signal
2020-03-11T09:58:57.097835-07:00 Telling connectors to stop
2020-03-11T09:58:57.097856-07:00 Allowing parsers to drain
2020-03-11T09:58:57.098766-07:00 udp exiting
2020-03-11T09:58:57.098800-07:00 udp exiting
2020-03-11T09:58:57.101361-07:00 udp exiting
2020-03-11T09:58:57.101400-07:00 udp exiting
2020-03-11T09:58:57.103881-07:00 udp exiting
2020-03-11T09:58:57.103905-07:00 udp exiting
2020-03-11T09:58:57.106527-07:00 udp exiting
2020-03-11T09:58:57.106579-07:00 udp exiting
2020-03-11T09:58:57.109120-07:00 udp exiting
2020-03-11T09:58:57.109145-07:00 udp exiting
2020-03-11T09:58:57.111790-07:00 udp exiting
2020-03-11T09:58:57.111837-07:00 udp exiting
2020-03-11T09:58:57.113853-07:00 udp exiting
2020-03-11T09:58:57.113912-07:00 udp exiting
2020-03-11T09:58:57.116262-07:00 udp exiting
2020-03-11T09:58:57.116397-07:00 udp exiting
2020-03-11T09:58:57.118365-07:00 udp exiting
2020-03-11T09:58:57.119002-07:00 udp exiting
2020-03-11T09:58:57.120865-07:00 udp exiting
2020-03-11T09:58:57.121108-07:00 udp exiting
2020-03-11T09:58:57.123517-07:00 udp exiting
2020-03-11T09:58:57.123552-07:00 udp exiting
2020-03-11T09:58:57.126043-07:00 udp exiting
2020-03-11T09:58:57.126079-07:00 udp exiting
2020-03-11T09:59:02.100923-07:00 Writing flows
2020-03-11T09:59:02.100969-07:00 Flow count: 48468
2020-03-11T09:59:02.417261-07:00 Waiting for file senders to
drain
2020-03-11T09:59:02.418564-07:00 Sending file: /home/
employee/traffic_flows_1583945942416835.pb.gz
2020-03-11T09:59:07.390307-07:00 Response Code 204
```

4. To check the status of Flowlink, use the `illumio-flowlink-ctl status` command.

Example with expected output:

```
illumio-flowlink-ctl status
```

OUTPUT ON CONSOLE

```
/illumio-flowlink-ctl status
```

```
Flowlink: RUNNING
```

Ingested Flow Types

This section provides the source Syntax available for use with various supported parsers and connectors.

IPFIX, NetFlow, and sFlow Parsers

destinations:

- name: # Required. An array of properties defining the data destinations configured for Flowlink. For example: netflow
- parser:
 - type: #Required. Information describing the parser associated with the data destination.
 - List of supported values: 'netflow', 'ipfix', 'sflow', 'aws', or 'text'
- connectors:
 - type: #Required. Information describing the data source connector associated with the data destination. Supported values: 'udp', 'tcp', 'kafka', or 'aws'
 - properties:
 - ports: #Required parameter to describe tcp or udp port. For example: '2055'
 - remote_addrs: #Optional parameter. String or list of IP address(es) to listen for as trusted data sources. Default is allow all IPs.
 - CIDRs are not supported. For example: '192.168.1.10,192.168.1.15'.

AWS Parser and Connector

destinations:

- name: # Required. An array of properties defining the data destinations configured for Flowlink. For example: aws
 - parser:
 - type: #Required. Information describing the parser associated with the data destination. Supported value: aws
 - connectors:
 - type: #Required. Information describing the data source connector associated with the data destination. Supported value: aws
 - properties:
 - region: #Required. Configures the AWS region of where the VPC flow logs are stored. Value not wrapped in quotes. Examples: us-west-2 or us-east-1
 - credentials: #Required. This is the AWS Access Key ID and AWS Access Key Secret created by IAM. The IAM user must have privileges to read Cloud Watch logs. You can put the contents into a file and run a script to cat the file. Value not wrapped in quotes. For example: \$cat /home/employee/aws_info
 - log_groupname: #Required. The name of the AWS Log Group. Value not wrapped in quotes. For example: myVPCFlowLogs



NOTE

The Access Key ID and Key Secret format should be the same as defined in YAML Configuration.

Text Parser with TCP or UDP Connector

destinations:

- name: # Required. An array of properties defining the data destinations configured for

Flowlink. For example: syslog

parser:

type: #Required. Information describing the parser associated with the data

destination. Supported value: 'text'

properties:

src_ip: #Required. Attribute tag or field number (starting at 1) used to

extract source IP. For example: sip

dst_ip: #Required. Attribute tag or field number (starting at 1) used to

extract destination IP. For example: dip

dst_port: #Required. Attribute tag or field number (starting at 1) used to

extract destination port. For example: dport

protocol: #Required. Attribute tag or field number (starting at 1) used to

extract protocol. For example: prot

icmp_type: #Optional. Attribute tag or field number (starting at 1) used to

extract icmp type. For example: type

icmp_code: #Optional. Attribute tag or field number (starting at 1) used to

extract icmp code. For example: code

timestamp: #Optional. Attribute tag or field number (starting at 1) used to

extract timestamp. Default: 1. For example:

"date_time, 1"

timestamp_format: #Optional. A string used to describe the timestamp format

field(s) in a record. The following values can be used year: yy[yy],

month(Jan[uary] etc): mmm[mmm], dayOfMonth:

dd or _d, dayOfWeek(Mon[day], etc): ddd[ddd], hour: HH, minutes: MM,

seconds(with optional precision): SS[.0{1 or more}],

timeZone: ZZZ, -HH[:MM],

-HHMM, ZHH[:MM], ZHHMM, unix timestamp: unix. For example: "mm dd yyyy HH:MM:SS"

connectors:

- type: #Required. Information describing the data

```
source connector associated
    with the data destination. List of supported values:
'tcp', 'udp', or 'sctp'
    properties:
        ports: #Required parameter to describe tcp or udp
port. For example: '514'
        remote_addrs: #Optional. A comma separated list of
remote host addresses
        from which to accept flows. For example:
'192.168.200.13'
```

Text Parser with Kafka Connector

```

destinations:
  - name: # Required. An array of properties defining the data
    destinations configured
      for Flowlink. For example: syslog
      parser:
        type: #Required. Information describing the parser
        associated with the data
          destination. Supported value: 'text'
          properties:
            src_ip: #Required. Attribute tag or field number used
            to extract source IP.
              For example: sip
            dst_ip: #Required. Attribute tag or field number used
            to extract destination IP.
              For example: dip
            dst_port: #Required. Attribute tag or field number
            used to extract destination
            port. For example: dport
            protocol: #Required. Attribute tag or field number
            used to extract protocol.
              For example: prot
            icmp_type: #Optional. Attribute tag or field number
            used to extract icmp type.
              For example: type
            icmp_code: #Optional. Attribute tag or field number
            used to extract icmp code.
              For example: code
            timestamp: #Optional. Attribute tag or field number
            used to extract timestamp.
              For example: "date_time, 1"
            timestamp_format: #Optional. A string used to describe
            the timestamp format
              field(s) in a record. The following values can be used
            year: yy[yy],
              month(Jan[uary] etc): mmm[mmm], dayOfMonth: dd or _d,
            dayOfWeek(Mon[day], etc):
              ddd[ddd], hour: HH, minutes: MM,
              seconds(with optional precision): SS[.0{1 or more}],
            timeZone: ZZZ, -HH[:MM],
              -HHMM,
              ZHH[:MM], ZHHMM, unix timestamp: unix. For example:
            "mm dd yyyy HH:MM:SS"
            connectors:
              - type: kafka

```

```

    properties:
      version: #Required. The version of the kafka
broker(s). For example: 1.2.0
      brokers: #Required. A comma separated list of kafka
brokers using FQDN and
      port: For example: example.com:9092
      group: test
      topics: test
      client_id: flowlink

```

Ingested Flow Examples

This section provides flow examples while using the supported parsers and connectors.

IPFIX

This example shows a destination that listens for IPFIX on UDP 4739 coming only from an IPFIX exporter whose IP address is 192.168.11.5. The flows from other IPFIX exporters will be discarded.

```

destinations:
- name: ipfix
  parser:
    type: ipfix
  connectors:
    - type: udp
      properties:
        ports: '4739'
        remote_addrs: '192.168.11.5'

```

NetFlow

This example uses NetFlow in which Flowlink will parse NetFlow records via UDP 6500 and listen for any data source IP address.

```

destinations:
- name: netflow
  parser:
    type: netflow
  connectors:
    - type: udp
      properties:
        ports: '6500'

```

AWS

This example shows an AWS destination in which the CloudWatch Log Group name is myVPCFlowLogs and is configured in the AWS Oregon region.

```
destinations:
- name: aws
  parser:
    type: aws
  connectors:
  - type: aws
    properties:
      region: us-west-2
      credentials: $cat /home/employee/aws_info
      log_groupname: myVPCFlowLogs
```

Text

This example shows a text destination using Syslog and listening on UDP 6514. The syslog format uses sip attribute to extract the source IP of the flow.

```
destinations:
- name: syslog
  parser:
    type: text
    properties:
      src_ip: sip
      dst_ip: dip
      dst_port: dport
      protocol: prot
      timestamp: "date_time, 1"
      timestamp_format: "mmm dd yyyy HH:MM:SS"
  connectors:
  - type: udp
    properties:
      ports: "6514"
```

YAML

```
pce_addr: 2x2mypce.example.com:8443
api_key: $cat api_info
data_directory: /home/employee/
aggregation_minutes: 5
destinations:
  - name: netflow
    parser:
      type: netflow
    connectors:
      - type: udp
        properties:
          ports: '6500'
  - name: ipfix
    parser:
      type: ipfix
    connectors:
      - type: udp
        properties:
          ports: '6514'
```

FIPS Compliance for Flowlink

This section describes the operational requirements for compliance with Federal Information Processing Standard (FIPS) 140-2 and 140-3 for Illumio Flowlink.

The Federal Information Processing Standard Publication (FIPS PUB) 140-x is a U.S. government computer security standard used to approve cryptographic modules. An authorized cryptographic equipment assessment laboratory has tested and verified that Flowlink faithfully incorporates the use of cryptographic functions provided by the FIPS 140-x validated modules as it applies to data in transit.

FIPS Prerequisites

The server on which Flowlink is installed must be running a FIPS-validated version of RHEL in FIPS mode and satisfy the Security Policy as stated in the relevant Red Hat Enterprise Linux OpenSSL Cryptographic Module document.

- **RHEL 8.2:** [Red Hat Enterprise Linux 8 OpenSSL Cryptographic Module version rhel8.20200305.1](#)

- **RHEL 9:** [Red Hat Enterprise Linux 9 OpenSSL FIPS Provider](#)

Enable Flowlink FIPS Compliance

1. After installing RHEL8.x or RHEL9, follow the required steps in the "Crypto Officer Guidance" section of the Red Hat Enterprise Linux OpenSSL documentation.
2. Reboot the system.
3. After the system starts, check that FIPS mode is enabled:

```
$ fips-mode-setup --check  
FIPS mode is enabled
```

4. Install the Flowlink RPM using this command:

```
sudo rpm -ivh --nodigest illumio-flowlink-<add-version-info>.rpm
```

5. To configure Flowlink, see [Configure Flowlink \[434\]](#).

When you've completed this procedure, Flowlink is FIPS compliant.

Check FIPS Mode Readiness

You can use a third-party tool to detect whether your system/container and your Golang binary are ready to run in FIPS mode. For details, see [fips-detect](#).

Flowlink Usage

This section describes how to export IPFIX or NetFlow v9 flow records from F5 BIG-IP to an external flow collector and some solutions while troubleshooting.

Collect Flow Records from F5

The example listed in the following steps uses a virtual edition of the F5 BIG-IP appliance in AWS and the Illumio Flowlink application to gather and parse flow data.



IMPORTANT

IPFIX and NetFlow have slightly different configuration steps depending on which flow record standard you choose.

Requirements

- Flowlink (flow collector)
- F5 BIG-IP system with LTM
- A virtual server configured on F5 box



NOTE

F5 must have a self-IP interface. The flows are sent out of this interface. When Flowlink is not in the same subnet as the self-IP, you must know the default gateway IP of the self-IP interface.

Create a Pool for Flow Collector

To create a pool of flow collectors to receive the flow record messages from the F5 system:

1. In the F5 UI, click **Main > Local Traffic > Pools > Pool Lists > Create**.
2. Enter a unique name in the **Name** field, which represents the flow collector.
3. A *Health Monitor* is not required. If you want to see if the F5 system can reach the flow collector, select **gateway_icmp** and move it to the Active box.
4. In the **New Member** section, configure the collector IP address.
5. Click **Add**.

If you are using **IPFIX**, use the following configuration:

Field	Value
Node Name	Enter the Collector IP address
Service Port	4739

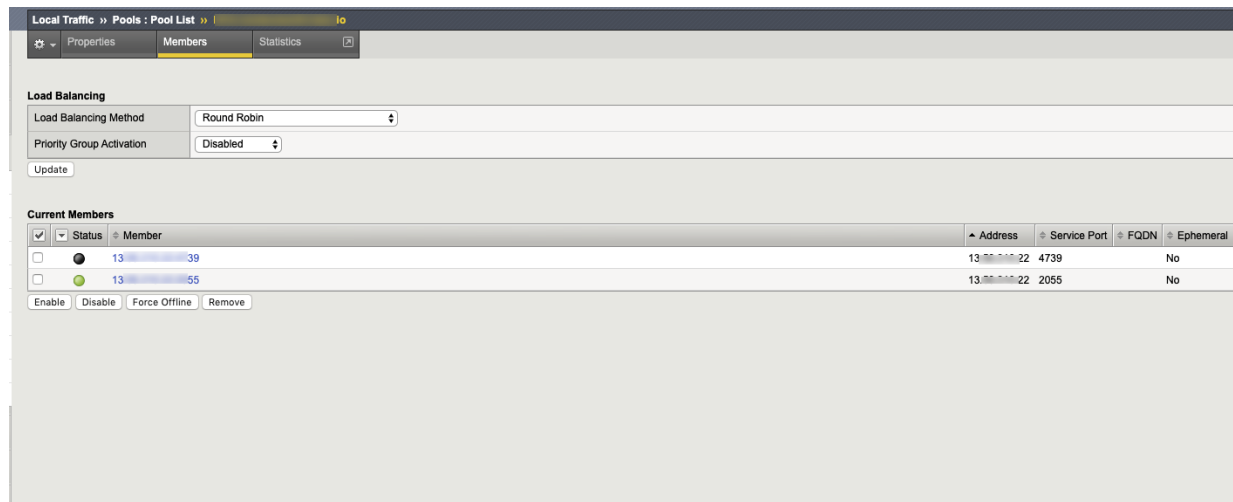
If you are using **NetFlow**, use the following configuration:

Field	Value
Node Name	Enter the Collector IP address
Service Port	2055

6. Click **Finished**.

The below example shows two (2) different nodes configured in one pool. Both nodes have the IP address. However, one is for IPFIX and one is for NetFlow. Even though F5 allows two nodes in the pool, it is recommended to only have one node enabled (either 2055 or 4739).

Example with NetFlow enabled and IPFIX disabled:



Create a Log Destination

To create a log destination to stream the logs in either IPFIX or NetFlow V9 format to the Pool:

1. In the F5 UI, click **Main > System > Logs > Configuration > Log Destinations > Create**.
2. Enter a unique name in the **Name** field, which represents the flow collector.
3. In the **Type** field, select IPFIX.
4. Configure the IPFIX Settings.

If you are using **IPFIX**, use the following configuration:

Field	Value
Protocol	Select IPFIX
Pool Name	Select the pool created earlier
Transport Profile	UDP

If you are using **NetFlow**, use the following configuration:

Field	Value
Protocol	Select NetFlow V9
Pool Name	Select the pool created earlier
Transport Profile	UDP

5. Click **Finished**.

Example of a Log Destination configuration with NetFlow:

The screenshot shows the F5 UI configuration page for a Log Destination. The breadcrumb trail is: System » Logs : Configuration : Log Destinations » ipfix-destination. The 'Properties' tab is selected. The 'General' section contains the following fields:

Name	ipfix-destination
Partition / Path	Common
Description	
Type	IPFIX

The 'IPFIX Settings' section contains the following fields:

Protocol	Netflow V9
Pool Name	NFIA-2 .io
Transport Profile	udp
Template Retransmit Interval	30
Template Delete Delay	5
Server SSL Profile	None

At the bottom of the settings section are two buttons: 'Update' and 'Delete...'.

Create a Log Publisher

To create a log publisher to send logs to the specified log destination:

1. In the F5 UI, click **Main > System > Logs > Configuration > Log Publishers > Create**.
2. Enter a unique name in the **Name** field, which represents the flow collector.
3. In the **Destination** field, move your log destination from *Available* to *Selected*.

4. Click **Finished**.

System » Logs : Configuration : Log Publishers » ipfix-publisher

⚙️ Properties

General

Name	ipfix-publisher
Partition / Path	Common
Description	

Log Destinations

	Selected		Available
Destinations	/Common ipfix-destination	<< >>	/Common alertd local-db local-syslog

Update Delete...

Create an iRule

To create an iRule to which it parses network traffic and sends flow records to the specified log publisher:

1. Go to **Main > iRules > iRule List > Create**.
2. Enter a unique name in the **Name** field, which represents the flow collector.
3. In the **Definition** text field, enter the rules for parsing traffic. Ensure the iRule points to the *log publisher* created earlier.
4. Click **Finished**.



IMPORTANT

In the iRule example shown below, replace *<insert_log_publisher_name_here>* with the name of the log publisher.

```

when RULE_INIT {
    set static::http_rule1_dest ""
    set static::http_rule1_tmplt ""
}

# CLIENT_ACCEPTED event to initiate IPFIX destination and
template
when CLIENT_ACCEPTED {
    set start [clock clicks -milliseconds]
    if { $static::http_rule1_dest == "" } {
        # open the logging destination if it has not been opened
yet
        set static::http_rule1_dest [IPFIX::destination open
-publisher /Common/
<insert_log_publisher_name_here>]
    }
    if { $static::http_rule1_tmplt == "" } {
        # if the template has not been created yet, create the
template
        set static::illumio_flowlink_POC_tmplt [IPFIX::template
create "flowStartMilliseconds
sourceIPv4Address sourceIPv6Address destinationIPv4Address
destinationIPv6Address
sourceTransportPort destinationTransportPort
protocolIdentifier octetTotalCount
packetTotalCount octetDeltaCount packetDeltaCount
postNATSourceIPv4Address
postNATSourceIPv6Address postNATDestinationIPv4Address
postNATDestinationIPv6Address
postNAPTSourceTransportPort
postNAPTDestinationTransportPort postOctetTotalCount
postPacketTotalCount postOctetDeltaCount
postPacketDeltaCount flowEndMilliseconds "]
    }
    set rule1_msg1 [IPFIX::msg create $static::http_rule1_tmplt]
}

# SERVER_CONNECTED event to initiate flow data to specified
log publisher and populate 5
tuples
when SERVER_CONNECTED {
    set client_closed_flag 0
    set server_closed_flag 0
    IPFIX::msg set $rule1_msg1 flowStartMilliseconds $start

```

```

IPFIX::msg set $rule1_msg1 protocolIdentifier [IP::protocol]

# Clientside
if { [clientside {IP::version}] equals "4" } {
    # Client IPv4 address
    IPFIX::msg set $rule1_msg1 sourceIPv4Address
[IP::client_addr]
    # BIG-IP IPv4 VIP address
    IPFIX::msg set $rule1_msg1 destinationIPv4Address
[clientside {IP::local_addr}]
} else {
    # Client IPv6 address
    IPFIX::msg set $rule1_msg1 sourceIPv6Address
[IP::client_addr]
    # BIG-IP IPv6 VIP address
    IPFIX::msg set $rule1_msg1 destinationIPv6Address
[clientside {IP::local_addr}]
}
# Client port
IPFIX::msg set $rule1_msg1 sourceTransportPort
[TCP::client_port]
# BIG-IP VIP port
IPFIX::msg set $rule1_msg1 destinationTransportPort
[clientside {TCP::local_port}]

# Serverside
if { [serverside {IP::version}] equals "4" } {
    # BIG-IP IPv4 self IP address
    IPFIX::msg set $rule1_msg1 postNATSourceIPv4Address
[IP::local_addr]
    # Server IPv4 IP address
    IPFIX::msg set $rule1_msg1 postNATDestinationIPv4Address
[IP::server_addr]
} else {
    # BIG-IP IPv6 self IP address
    IPFIX::msg set $rule1_msg1 postNATSourceIPv6Address
[IP::local_addr]
    # Server IPv6 IP address
    IPFIX::msg set $rule1_msg1 postNATDestinationIPv6Address
[IP::server_addr]
}
# BIG-IP self IP port
IPFIX::msg set $rule1_msg1 postNAPTSourceTransportPort
[TCP::local_port]
# Server port
IPFIX::msg set $rule1_msg1 postNAPTDestinationTransportPort

```

```

[TCP::server_port]
}

# SERVER_CLOSED event to collect IP pkts and bytes count on
serverside
when SERVER_CLOSED {
    set server_closed_flag 1
    # when flow is completed, BIG-IP to server REQUEST pkts and
    bytes count
    IPFIX::msg set $rule1_msg1 octetTotalCount [IP::stats bytes
out]
    IPFIX::msg set $rule1_msg1 packetTotalCount [IP::stats pkts
out]
    # when flow is completed, server to BIG-IP RESPONSE pkts and
    bytes count
    IPFIX::msg set $rule1_msg1 octetDeltaCount [IP::stats bytes
in]
    IPFIX::msg set $rule1_msg1 packetDeltaCount [IP::stats pkts
in]
    IPFIX::destination send $static::http_rule1_dest
$rule1_msg1
}

# CLIENT_CLOSED event to collect IP pkts and bytes count on
clientside
when CLIENT_CLOSED {
    set client_closed_flag 1
    # when flow is completed, client to BIG-IP REQUEST pkts and
    bytes octetDeltaCount
    IPFIX::msg set $rule1_msg1 postOctetTotalCount [IP::stats
bytes in]
    IPFIX::msg set $rule1_msg1 postPacketTotalCount [IP::stats
pkts in]
    # when flow is completed, BIG-IP to client RESPONSE pkts and
    bytes count
    IPFIX::msg set $rule1_msg1 postOctetDeltaCount [IP::stats
bytes out]
    IPFIX::msg set $rule1_msg1 postPacketDeltaCount [IP::stats
pkts out]
    # record the client closed time in ms
    IPFIX::msg set $rule1_msg1 flowEndMilliseconds [clock click
-milliseconds]
    # send the IPFIX log
    IPFIX::destination send $static::http_rule1_dest
$rule1_msg1
}

```


Apply the iRule to a Virtual Server

To apply the iRule to a virtual server whose traffic you want to parse:

1. Go to **Main > Virtual Server > Virtual Server List**.
2. Select the virtual server you want to monitor.
3. Click the **Resources** tab. In the iRule section, click **Manage**.
4. Select the **iRule** that you previously created and move the iRule from *Available* to *Enable*.
5. Click **Finished**.

Example of a Virtual Server Resources page with the new iRule applied:

Local Traffic » Virtual Servers : Virtual Server List » HRM-VirtualServer

Properties Resources Security Statistics

Load Balancing

Default Pool	HRM-Webserver
Default Persistence Profile	source_addr
Fallback Persistence Profile	None

Update

iRules Manage...

Name
IPFIX

Policies Manage...

Name
No records to display.

Create a Route Entry

By default, all traffic is sent out of the management interface. However, F5 does not support flow exports via the management NIC. You must add a route to force traffic, which is destined to the flow collector to leave a self-IP interface.

To create a route entry, if the F5 self-IP is unable to reach the flow collector:

1. In the F5 UI, click **Main > Network > Routes > Add**.
2. In the **Properties** section, create a route entry to send the flow records from F5 to the external flow collector IP address.
For Resource, select the *Use Gateway* option.

The screenshot shows the 'FlowLink' configuration page in a web interface. The breadcrumb trail is 'Network » Routes » FlowLink'. The 'Properties' tab is selected. The 'Properties' section contains a table with the following fields:

Name	FlowLink
Partition / Path	Common
Description	
Destination	13... 22
Netmask	255... 255
Resource	Use Gateway...
Gateway Address	IP Address 10.1.3.1
MTU	0

At the bottom of the form are 'Update' and 'Delete' buttons. Two red arrows point to the 'Resource' and 'Gateway Address' fields, with a red text label 'Self-IP's Default Gateway' next to them.

Troubleshooting

This section describes how to troubleshoot some issues when configuring or using Flowlink.

Flowlink not Receiving Data

1. Make sure iptables is turned *Off* on Flowlink, or make sure iptables is not blocking the ports that Flowlink is listening on.
2. Use `netstat -a` to make sure Flowlink is listening on the correct ports.



NOTE

netstat has a bug, which shows that applications are only listening with IPv6 on listed ports, when they are actually listening on those ports with IPv4.

Unable to Ping or TCPdump on the F5 Self-IP Interface

1. SSH to F5 as an administrator.

2. List the interfaces to see the interface names.

```
admin@(ip-10-1-1-197)(cfg-sync Standalone)(Active)(/Common)
(tmos)# show net interface
```

```
-----
---
Net::Interface
Name  Status    Bits      Bits      Pkts      Pkts      Drops  Errs
Media
              In       Out       In       Out
-----
---
1.1      up      1.3G      1.1G      2.6M      2.6M        0      0
none
1.2      up    177.7M    301.4M    298.9K    310.4K        0      0
none
mgmt     up    310.9G    876.6G    298.8M    325.5M        0      0
none
```

3. Run TCPdump to listen for traffic between Self-IP interface and flow collector IP.
4. Generate traffic while the TCPdump is running by either opening another SSH session and doing PING test or by sending normal traffic through the virtual server. If you turned on **health monitoring** with `gateway_icmp` enabled from the [Create a Pool for Flow Collector \[452\]](#) section, then F5 should already generate ICMP traffic.

The example shown below uses interface name *1.2* with flow collector IP *13.56.210.22*. Health monitoring with *gateway_icmp* is enabled.

```
admin@(ip-10-1-1-197)(cfg-sync Standalone)(Active)(/Common)
(tmos)# tcpdump -ni 1.2 host 13.56.210.22
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on 1.2, link-type EN10MB (Ethernet), capture size
65535 bytes
09:08:47.855318 IP 10.1.3.223 > 13.56.210.22: ICMP echo
request, id 54351, seq 37906, length 20 out slot1/tmm3 lis=
09:08:47.857694 IP 13.56.210.22 > 10.1.3.223: ICMP echo
reply, id 54351, seq 37906, length 20 in slot1/tmm3 lis=
09:08:52.864852 IP 10.1.3.223 > 13.56.210.22: ICMP echo
request, id 54354, seq 37906, length 20 out slot1/tmm2 lis=
09:08:52.867091 IP 13.56.210.22 > 10.1.3.223: ICMP echo
reply, id 54354, seq 37906, length 20 in slot1/tmm2 lis=
```

Network Connectivity

The flow to test network connectivity is:

- Network device > Flowlink
- Flowlink > PCE

TCPdump

To use TCPdump:

- Run on a network device to verify flow records are sent out.
- Run on Flowlink to verify flow records are coming in.

Debug Option

Flowlink has a debug option that displays:

- Incoming flow records
- IP, port, and protocol recorded for flow records
- Each time flows are aggregated and uploaded to the PCE
- PCE response code to **POST**

To debug Flowlink in the session, add the **--debug** flag to your Flowlink command.

Example with the debug option enabled:

```
CONFIG_FILE=/home/employee/config.yaml.netflow /usr/local/bin/illumio/flowlink --debug
```



IMPORTANT

Using the debug flag, generates a large amount of data to the console. Enable this option only if needed.

Illumio Core PCE CLI Tool Guide 1.4.3

Overview of the CLI Tool

Learn about the CLI Tool, become familiar with the general syntax of the CLI Tool commands, and learn the environment variables you can use to customize the CLI Tool.



IMPORTANT

See the [Illumio Core CLI Tool 1.4.3 What's New document](#).

Before You Begin Using the CLI Tool

Before you start using the CLI Tool, review these prerequisites:

- The CLI Tool interacts with the PCEs. Become familiar with PCE concepts such as core and data nodes, workloads, and traffic. See the [PCE Administration Guide](#).
- The CLI Tool is often used to upload vulnerability data. Learn how vulnerability data is used in the PCE web console. See "Vulnerability Maps" in the [Visualization Guide](#).
- The CLI Tool can be used with workload data. Learn about workloads. See the "VEN Architecture and Components" topic in the [VEN Administration Guide](#).
- The CLI Tool can be used with security policy rules, rulesets, labels, and similar resources. Become familiar with these concepts. See "The Illumio Policy Model" in the [Security Policy Guide](#).

CLI Tool Versioning

Illumio Segmentation for Data Centers CLI Tool 1.4.3 is compatible with Illumio Segmentation for Data Centers PCE versions:

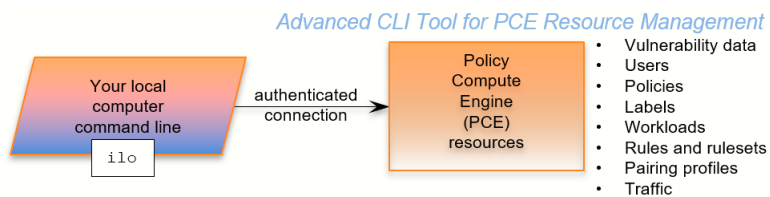
- PCE 24.5
- PCE 24.2.10

- PCE 23.5
- PCE 23.2
- PCE 22.5
- PCE 22.2.0 (Standard)
- PCE 21.5.20 (LTS)
- PCE 21.2.4 (LTS)
- PCE 22.1.1 (Standard)

The CLI Tool version numbering is independent of the release and version numbering of Illumio Segmentation for Data Centers PCE and VEN. The CLI Tool works with multiple versions of the PCE and VEN and does not necessarily need software changes in parallel with releases of the PCE or the VEN.

CLI Tool and PCE Resource Management

The Illumio CLI Tool allows you to manage many of your PCE resources directly from your local computer.



Use the CLI Tool to:

- Import vulnerability data for analysis with Illumination.
- Help with tasks such as directly importing workload information to create workloads in bulk.
- Create, view, and manage your organization's security policy rules, rulesets, labels, and other resources.



CAUTION

The CLI Tool is a tool that you can use to work with your PCE resources. Test your CLI Tool commands against a non-production system before using them on your production PCEs.

The CLI Tool is named **ilo**. It is a wrapper around the Illumio Segmentation for Data Centers REST API. No knowledge of the REST API is required.

The ilo Command

Learn about the general syntax of the CLI Tool command, **ilo**, and how to use the command-line help to get more specific syntax information.

CLI Tool Formal Syntax

The formal syntax for the **ilo** command is:

```
ilo resource_or_specialCommand argument options
```

Where:

- **resource_or_specialCommand** represents either a resource managed by the PCE or a command unrelated to a particular resource.
A resource is an object that the PCE manages, such as a workload, label, or pairing profile.
Example resource command on Linux (create a workload):

```
ilo workload create --name FriendlyWorkloadName --hostname  
myWorkload.BigCo.com
```

A special command is a command that is not related to a specific resource. Special commands include **user**, **login**, **use_api_key**, and **node_available**.

Example special command on Windows (log out of PCE):

```
ilo user logout --id 6
```

- The **argument** represents an operation on the resource or special command.
- The **options** are allowed options for the **resource_or_specialCommand**. The specific option depends on the type of resource or special command.

CLI Tool Help

To get a complete list of all the available CLI Tool commands, use the **ilo** command without options. This command displays the high-level syntax of special commands, resources, and their allowable options.

For details about a resource's or special command's arguments, specify the resource's name followed by the argument followed by the **--help** option. For example:

```
ilo workload create --help
```

HTTP Response Codes and Error Messages

Learn about the response codes and error messages that are returned using CLI Tool commands.

REST API HTTP Response Codes

At the end of its output, the `ilo` command displays the REST API HTTP response code from the command. For example, a successful operation shows the following output:

```
...
200, OK
```

Error Messages

For many syntactical or other types of errors, the CLI Tool displays a general message encouraging you to verify your syntax with the CLI Tool help:

The `ilo` command has encountered an error. Check your syntax with either of the following commands:

```
- ilo
- ilo <command> --help
```

In some circumstances, the CLI Tool writes a detailed log of errors:

For detailed error messages, see the file:
`location-of-local-temp-directory/illumio-cli-error.log`

Where `location-of-local-temp-directory` is:

- Linux: `/tmp`
- Windows: `C:\Windows\Temp`

Environment Variables

Illumio provides Linux environment variables to allow you to customize the operation of the CLI tool.

Environment Variable	Purpose
ILO_API_KEY_ID	API key for non-password-based authentication and cookie-less session with PCE. See "Authenticate with an API Key".
ILO_API_KEY_SECRET	API key secret for non-password-based authentication and cookie-less session with PCE. See "Authenticate with an API Key".
ILO_API_VERSION	API version to be used to execute CLI commands. Set this to override the default API version. See "Set the Illumio ASP REST API Version." Default: v2. Example: <code>\$ export ILO_API_VERSION=v1</code>
ILO_CA_DIR	Directory that contains certificates. See "TLS/SSL Certificate for Access to the PCE".
ILO_CA_FILE	Absolute path to the certificate file. See "TLS/SSL Certificate for Access to the PCE".
ILO_DISPLAY_CONFIG	An absolute path to the display configuration file is to be used with the list command. See "Linux Save Specific Fields to File For Reuse".
ILO_INSECURE_PASSWORD	Provide a password for login. If this variable is set, the login password prompt does not appear, and this password is used instead. Do not use in a production system when authentication security is desired. Example: <code>\$ export ILO_INSECURE_PASSWORD=myInsecurePassword</code>
ILO_KERBEROS_SPN	Kerberos service principal name (SPN). Specify this variable when using Kerberos authentication.
ILO_LOGIN_SERVER	PCE login server FQDN. Use this variable when the login server FQDN is not the same as the PCE FQDN. See "Explicit Log into the PCE".
ILO_ORG_ID	Organization identifier for certificate-authenticated session with PCE. Value is always 1. Does not need to be explicitly set The environment variable is set by the system and should not be explicitly set. See "Authentication to PCE with API Key or Explicit Login".
ILO_PCE_VERSION	PCE version for the CLI to use. Default: 19.1.0 Example: <code>\$ export ILO_PCE_VERSION=18.2.5</code>
ILO_PREVIEW	Enable any preview features that are included in this release. To disable preview features, remove this variable from the environment.
ILO_SERVER	FQDN of PCE for login and authentication with PCE. See "Authentication to PCE with API Key or Explicit Login".
TSC_ACCESS_KEY	These two ENV variables have been added in the release 1.4.2 to set up the Tenable SC API keys, which are used for authentication.
TSC_SECRET_KEY	
TSC_HOST	The variable that specifies the target host for Tenable

Environment Variable	Purpose
QAP_HOST	The variable that specifies the target host for Qualys

Installation and Authentication

Learn how to install the CLI Tool, set up authentication, upgrade the tool, and uninstall it.

Review the prerequisites before you install the PCE CLI Tool.

Prerequisite Checklist

- ☐ License for vulnerability data upload
- ☐ Vulnerability data for upload
- ☐ Functional PCE
- ☐ Supported operating systems
- ☐ TLS/SSL certificate for authenticating to the PCE
- ☐ API version set in configuration
- ☐ The CLI Tool installation program

CLI Tool Installation Prerequisites

Review the prerequisites before you install the CLI Tool.

License for Vulnerability Data

The Illumio Segmentation for Data Centers Vulnerability Maps license is required to import vulnerability data into the Illumio PCE. For information about obtaining a license, contact Illumio Customer Support. For information on activating the license, see [Add the License for Vulnerability Data Upload \[490\]](#).

Upload Vulnerability Data

When you plan on using the CLI Tool to upload vulnerability data, make sure you have the data to upload in advance. See [Supported Vulnerability Data Sources \[493\]](#).

Install Functional PCE

Because the CLI Tool is for managing resources on your PCE, you must already have installed a fully functional PCE.

Supported Computer Operating Systems

The following operating systems support the CLI Tool:

Linux

- Ubuntu 18.04
- Ubuntu 20.04
- Centos/RHEL 7.9
- Centos/RHEKL 8.4

Microsoft Windows



NOTE

The CLI Tool is not supported on Windows 32-bit CPU architecture. Ensure that you run it on Windows 64-bit CPU architecture.

- Windows 2012 64 bit
- Windows 2016 64 bit
- Windows 10 64 bit

TLS/SSL Certificate for Access to the PCE

You need a TLS/SSL certificate to connect to the PCE securely. Requirements for this certificate are provided in the PCE Installation and Upgrade Guide.

Alternative Trusted Certificate Store

To secure the connection to the PCE, by default, the CLI Tool relies on your computer's trusted certificate store to verify the PCE's TLS certificate. You can specify a different trusted store. When you have installed a self-signed certificate on the PCE, an alternative trusted store might be necessary.

Example: Set envvar for alternative trusted certificate store z

```
export ILO_CA_FILE=~/.self-signed-cert.pem
```

Set the Illumio Segmentation for Data Centers REST API Version

The CLI Tool uses v2 of the Illumio Segmentation for Data Centers REST API by default.

Install, Upgrade, and Uninstall the CLI Tool

This section explains how to install, upgrade, or uninstall the CLI Tool on Linux or Windows.

Download the Installation Package

Download the CLI Tool installation package from the [Tools Catalog](#) page (login required) to a convenient location on your local computer.

Install Linux CLI Tool

The CLI Tool installer for Linux is delivered as an RPM for RedHat/CentOS and DEB for Debian/Ubuntu.

The CLI Tool is installed in the local binaries directory `/usr/local/bin`.

Log into your local Linux computer as a normal user and then use **sudo** to run one of the following commands.

RedHat/CentOS:

```
sudo rpm -ivh /path_to/nameOfCliRpmFile.rpm
```

Debian/Ubuntu:

```
sudo dpkg -i / path_to / nameOfCliDebFile .deb
```

Upgrade Linux CLI Tool

Log into your local Linux computer as a normal user and then use **sudo** to run one of the following commands.

RedHat/CentOS:

```
sudo rpm -Uvh /path_to/nameOfCliRpmFile.rpm
```

Debian/Ubuntu:

```
sudo dpkg -i / path_to / nameOfCliDebFile .deb
```

The same option, `-i`, is used for installation or upgrade.

Uninstall the Linux CLI Tool

Log into your local Linux computer as a normal user and then use **sudo** to run one of the following commands.

RedHat/CentOS:

```
sudo rpm -e nameOfCliRpmFile
```

Debian/Ubuntu:

```
sudo dpkg -r nameOfCliDebFile
```

Install Windows CLI Tool

The CLI Tool installer for Windows is delivered as an **.exe** file.

Log into your local Windows computer as an administrator and start the installation program in the following ways.

- In the Windows GUI, double-click the .exe file.
- In a cmd window, run the .exe.
- In a PowerShell window, run the .exe.

After starting the installation program, follow the leading prompts.

A successful installation ends with the "Installation Successfully Completed" message, and the help text for the CLI Tool is displayed.

Upgrade Windows CLI Tool

The CLI Tool cannot be directly upgraded from an existing CLI Tool installation.

If you have already installed a previous version of the CLI Tool, manually uninstall it using the Windows Control Panel's Add/Remove Programs.

After uninstalling the previous version of the CLI Tool, install the new version of the CLI Tool as described in [Install Windows CLI Tool \[471\]](#).

Uninstall the Windows CLI Tool

Log into your local Windows computer as an administrator, and from the Windows Control Panel, launch Add/Remove Programs.

Select Illumio CLI from the list and click the **Uninstall** button.

Authenticate with the PCE

When using the CLI Tool, you can authenticate to your PCE in the following ways:

- **With an API key and key secret:**

This is the easiest way. Before you create the API key and secret, you need to log in to authenticate to the PCE. After creating and using the key, you do not have to specify your username and password again.

- **With the explicit command to log in:**

This always requires a username and password.

This method also requires you to log out with a user ID displayed at login. The explicit login times out after ten minutes of inactivity, after which you must log in again.

For both authentication mechanisms, on the command line, you always need to specify the FQDN and port of your PCE. The default port for the PCE is 8443. However, your system administrator can change this default. Check with your system administrator to verify the port you need.

Authenticate with an API Key

To authenticate to the PCE with an API key, you must first explicitly log into the PCE, create the API key, and then use the key to authenticate.

1. Authenticate via explicit login:

```
ilo login --server yourPCEfqdn:itsPort
```

2. Create the API key:

```
ilo api_key create --name someLabel
```

someLabel is an identifier for the key.

3. Use the API key to authenticate:

```
ilo use_api_key --server yourOwnPCEandPort --key-id
yourOwnKeyId --org-id --key-secret yourOwnKeySecret
```

Create an API Key

On Linux, for later ease of use, with the `api_key --create-env-output` option, you can store the API key, API secret, and the PCE server name and port as environment variables in a file that you source in future Linux sessions.

Linux Example

This example creates the API key and secret and stores them as environment variables in a file named `ilo_key_MY_SESSION_KEY`.

```
# ilo api_key create --name MY_SESSION_KEY --create-env-output
# Created file ilo_key_MY_SESSION_KEY with the following
contents:

export ILO_API_KEY_ID=14ea453b6f8b4d509
export
ILO_API_KEY_SECRET=e1fa1262461ca2859fcf9d91a0546478d10a1bcc4c57
9d888
a4e1cace71f9787
export ILO_SERVER=myPCE.BigCo.com:8443
export ILO_ORG_ID=1

# To export these variables:
# $ source ilo_key_MY_SESSION_KEY
```

Log Into the PCE

Without an API key, you must explicitly log into the PCE.

For on-premises PCE deployments, the login syntax is the FQDN and port of the PCE:

```
ilo login --server yourPCEfqdn:itsPort
```

For `yourPCEfqdn:itsPort`, do not specify a URL instead of the PCE's FQDN and port. If you do, an error message is displayed.

For the Illumio Secure Cloud customers, the login syntax is:

```
ilo login --server URL_or_bare_PCEfqdn:itsPort --login-server
login.illum.io:443
```

See the explanation above about the argument to the **--server** option.

- After login, the output of the command shows a user ID value. Make a note of this value. You need it when you log out.
- The session with the PCE remains in effect as long as you keep using the CLI Tool. After 10 minutes of inactivity, the session times out, and you must log in again.

Example

In this example, the user ID is 6.

```
C:\Users\marie.curie> ilo login --server myPCE.BigCo.com:8443
Enter User Name: albert.einstein@BigCo.com
Enter Password: Welcome Albert!
User ID = 6
Last Login Time 2018-08-10T-09:58:07.000Z from someIPAddress
Access to Orgs:
Albert: (2)
Roles: [3]
Capabilities: {"basic"=>["read", "write"],
"org_user_roles"=>["read", "write"]}
User Time Zone: America/Los_Angeles
Server Time: 2018-08-12T17:58:07.522Z
Product Version: 16.09.0-1635
Internal Version:
48.0.0-255d6983962db54dc7ca627534b9f24b94429bd5
Fri Aug 6 16:11:50 2018 -0800
Done
```

Log Out of the PCE

To end a session with the PCE, use the following command:

```
ilo user logout --id valueOfUserIdFromLogin
```

Where:

- **valueOfUserIdFromLogin** is the user ID associated with your login. See [Log Into the PCE \[473\]](#) for information.

Example

In this example, the user ID is 6.

```
ilo user logout --id 6
```

CLI Tool Commands for Resources

This section describes how to use the CLI Tool with various PCE resources.

View Workload Rules

You can view a specific workload's rules with the following command:

```
ilo workload rule_view --workload-id UUID
```

Where:

- UUID is the workload's UUID. See [About the Workload UUID \[480\]](#) for information.

In the example below, the workload's UUID is as follows:

```
2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```

Example View Workload Rules

```
ilo workload rule_view --workload-id 2ca0715a-b7e3-40e3-ade0-79f2c7adced0
+-----+-----+
| Attribute | Value |
+-----+-----+
| providing | [] |
+-----+-----+
Using
+-----+
+-----+
+-----+
| Ports And Protocols |
Rulesets
+-----+
+-----+
+-----+
| Href |
Name |
+-----+
+-----+
+-----+
| [[-1, -1, nil]] | [{"href"=>"/api/v2/orgs/28/sec_policy/8/rule_sets/1909", "name"=>"Default",
"secure_connect"=>false, "peers"=>[{"type"=>"ip_list",
"href"=>"/api/v2/orgs/28/sec_policy/8/ip_lists/188",
"name"=>"Any (0.0.0.0/0)",
"ip_ranges"=>[{"from_ip"=>"0.0.0.0/0"}]}]}] | /api/v2/orgs/28/sec_policy/8/services/1153 | All Services |
+-----+
+-----+
+-----+
200, OK
```

View Report of Workload Services or Processes

The following command lists all running services or processes on a workload:

```
ilo workload service_reports_latest --workload-id UUID
```

Where:

- *UUID* is the workload's UUID. See [About the Workload UUID \[480\]](#).

In the example, the workload's UUID is as follows:

```
2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```

Example Workload Service Report

```
ilo workload service_reports_latest --workload-id 2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```

```
+-----+-----+
| Attribute          | Value                                |
+-----+-----+
| uptime_seconds     | 1491                                |
| created_at         | 2015-10-20T15:13:00.681Z           |
+-----+-----+
```

Open Service Ports

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| Protocol | Address | Port | Process Name | User
| Package | Win Service Name |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| udp      | 0.0.0.0 | 5355 | svchost.exe  | NETWORK
SERVICE |          | Dnscache          |
...
| tcp      | 0.0.0.0 | 135  | svchost.exe  | NETWORK
SERVICE |          | RpcSs            |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
200, OK
```

Use the `list` Option for Resources

Many resources take the `list` option. This section details some of its uses.

Default List of All Fields

The default `list` command displays all fields associated with the resource:

```
ilo resource list
```

List Only Specific Fields

With the `--field` option, specify the fields to display:

```
ilo resource list --field CSV_list_of_fieldnames
```

For example, to display a list of labels with only the href, key, and value fields, use the **--field** option with those fields as comma-separated arguments.

Example List with Selected Fields

```
ilo label list --fields href,key,value
+-----+-----+-----+
| Href          | Key  | Value          |
+-----+-----+-----+
| /api/v2/2/labels/1 | role | Web            |
| /api/v2/2/labels/2 | role | Database       |
| ...           |     |                |
| /api/v2/2/labels/48 | loc  | Asia           |
+-----+-----+-----+
```

Nested Resource Fields and Wildcards

Some resources have hierarchical, nested fields. For example, the workload resource includes the following hierarchy for the agent field:

agent/config/log_traffic

- A field named agent
 - That has a field named **config**
 - That has a field named **log_traffic**

To list nested fields, separate the hierarchy of the field names with a slash to the depth of the desired field.

To see all nested fields of one of a resource's fields, use the asterisk (*) wildcard.

Examples

The following example displays all fields under the agent/config field.

Example of All Nested Fields with Wildcard (*)

```
ilo workload list --field agent/config/*
```

Log Traffic	Visibility Level	Mode
false	flow_summary	illuminated
false	flow_summary	idle

You can combine individual field names, nested field names, and the * wildcard.

Example: Combination of Individual fields, Nested fields, and Wildcard

```
ilo workload list --fields href,hostname,agent/config/*,agent/status/uid,agent/status/status
```

Href	Hostname	Log Traffic	Visibility Level	Mode	Uid	Status
/api/v2/1/workloads/527b8aca-97aa-43b9-82e1-29b17a947cdd	hrm-web.webscaleone.info	false	flow_summary	illuminated	0ffd2290-e26a-4ec6-b241-9e2205c0b730	active
/api/v2/1/workloads/4a8743a4-14ee-40d0-9ed2-990fe3f0ffb1	hrm-db.webscaleone.info	false	flow_summary	illuminated	145a3cc8-01a8-4a52-97b8-74264ad690e4	active

...

Linux: Save Fields for Reuse

On Linux, to easily reuse specific fields, create a display configuration file in YAML format and set the environment variable **ILO_DISPLAY_CONFIG** to point to that file. You no longer need to specify specific fields on the list command line.

Examples

Configure the workloads list command to display only the href, hostname, all agent configuration fields, and agent version:

Example Command to Save to List Configuration File

```
ilo workload list --fields href,hostname,agent/config/*,agent/status/agent_version
```

Add the field names to a display configuration file in the following YAML format:

Example YAML Layout of Display Configuration File

```
workload:
  fields:
    - href
    - hostname
  agent:
    config:
      fields:
        - '*'
    status:
      fields:
        - agent_version
```

Set the Linux environment variable **ILO_DISPLAY_CONFIG** to the path to the YAML file:

Example ILO_DISPLAY_CONFIG environment variable

```
$ export ILO_DISPLAY_CONFIG=~/ilo_display/display_config.yaml
```

List of All Workloads

To view all details for all workloads, use the following command:

```
ilo workload list
```

About the Workload UUID

To view an individual workload, you need the workload's identifier, called the UUID, or Universal Unique Identifier.

The UUID is shown in the list of all workloads described in [List of All Workloads \[480\]](#). The UUID is the last word of the value of the workload's href field, as shown in bold in the following example:

```
/api/v2/orgs/28/workloads/2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```

View Individual Workload

To see the details about an individual workload, use the following command:

```
ilo workload read -workload-id UUID
```

Where:

- UUID is the workload's UUID. See [About the Workload UUID \[480\]](#) for information.

The details of an individual workload are grouped under major headings:

- Workload > Interfaces
- Workload > Labels
- Workload > Services
- Services > Open Service Ports
- Agent > Status

Example List of Individual Workload

```

ilo workload read --workload-id 2ca0715a-b7e3-40e3-ade0-79f2c7adced0
+-----+
+-----+
+-----+
| Attribute          |
Value
+-----+
+-----+
+-----+
| href              | /orgs/1/workloads/2ca0715a-
b7e3-40e3-ade0-79f2c7adced0
| deleted           |
false
...
Workload -> Interfaces
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| Name | Address          | Cidr Block | Default Gateway
Address | Link State | Network Id | Network Detection Mode
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| eth0 | 10.0.0.16        | 8          |
10.0.0.1 | up          | 1          |
single_private_brn
...
Workload -> Labels
+-----+
| Href          |
+-----+
| /orgs/1/labels/37 |
...
Workload -> Services
+-----+-----+-----+
| Attribute      | Value          |
+-----+-----+-----+
| uptime_seconds | 69016553       |
...
Services -> Open Service Ports
+-----+-----+-----+-----+
+-----+

```



```

| Protocol | Address | Port | Process Name | User | Package |
Win Service Name |
+-----+-----+-----+-----+-----+-----+
+-----+
| 17      | 0.0.0.0 | 123  | ntpd          | root |         |
|         |         |      |               |      |         |
...
Workload -> Agent
+-----+
+-----+
+-----+
| Attribute |
Value
|
+-----+
+-----+
+-----+
| config    | {"log_traffic"=>true,
"visibility_level"=>"flow_summary", "mode"=>"enforced"} |
| href      | /orgs/1/
agents/16
|
...
Agent -> Status
+-----+
+-----+
| Attribute |
Value
|
+-----+
+-----+
| uid       | db482b06-41c6-4297-
a60c-396de13576ad |
| last_heartbeat_on |
2016-12-07T04:07:03.756Z
|
...
200, OK

```

List Draft or Active Version of Rulesets

A security policy includes a ruleset, IP lists, label groups, services, and security settings. Before changes to these items take effect, the policy must be provisioned on the managed workload by setting its state to active with the CLI Tool or provisioning it with the PCE web console.

To view a ruleset and provisioning state, use the following command:

```
ilo rule_set list --pversion state
```

Where **state** is one of the following values:

- Draft: Any policy item that has not yet been provisioned.
- Active: All policy items that have been provisioned and are enabled on workloads.

The provisioning states are listed in the Enabled column:

- True: The policy is provisioned.
- Empty: The policy is a draft.

Example Draft Versions of Rulesets

```
ilo rule_set list --pversion draft
+-----+
+-----+-----+-----+-----+
| Href                                     | Created
By                                     | Name | Description | Enabled |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| /api/v2/orgs/28/sec_policy/draft/rule_sets/2387 |
{"href"=>"/api/v2/users/74"} | foo1      |          | true
| /api/v2/orgs/28/sec_policy/draft/rule_sets/1909
| {"href"=>"/api/v2/users/0"} | Default
|                               | true    ...
200, OK
```

The state of the policy is stored in the agent/status/status field. See [Nested Resource Fields and Wildcards \[478\]](#) for information.

Support for Proxy

Release CLI 1.4.3 includes support for authenticated and unauthenticated proxies.

Type the `ilo login --help` command to see proxy-related options.

Table 4. ilo login --help

Command Options	Description
-v, --verbose	Verbose logging mode
--trace	Enable API Trace Mode
--server SERVER_NAME	Illumio API Access gateway server name
--login-server LOGIN_SERVER	Illumio login server name
--kerberos-spn KERBEROS_SPN	Illumio Kerberos SPN Kerberos authentication is only applicable to --login-server option
--proxy-server PROXY_SERVER	proxy server
--proxy-port PROXY_PORT	proxy port
--proxy-server-username PROXY_SERVER_USERNAME	proxy server username
--proxy-server-password PROXY_SERVER_PASSWORD	proxy server password
--logout	Logout
--username USER	User Name
--username USER	User Name
--auth-token AUTH_TOKEN	authorization token

Connecting via a Proxy

The command for connecting via an unauthenticated proxy:

```
ilo login --server <fqdn:port> --proxy-server <proxy_ip> --
proxy-port <proxy_port> --user-name selfserve@illumio.com
```

An example of connecting via an unauthenticated proxy:

```
ilo login --server 2x2testvc308.ilabs.io:8443 --proxy-
server 10.2.184.62 --proxy-port 3128 --user-name
selfserve@illumio.com
```

An example of connecting via an authenticated proxy:

```
ilo login --server 2x2testvc308.ilabs.io:8443 --proxy-server devtest30.ilabs.io --proxy-port 3128 --user-name selfserve@illumio.com --proxy-server-username proxy_user --proxy-server-password proxy_124
```

After the command is executed, users are prompted to enter the PCE user's password, and then a session will be created in the context of the proxy server.

From this point on, all connections/traffic will use the proxy to send traffic.

Using API Keys and Secrets with a Proxy Server

With the command `ilo use_api_key`, you can use an API Key and a secret with a proxy server:

Table 5. ilo use_api_key --help

Command options	Description
<code>--key-id</code>	API Key ID
<code>--key-secret</code>	API Key Secret
<code>--org-id</code>	Illumio Org ID
<code>--user-id Illumio</code>	User ID
<code>-v, --verbose</code>	Verbose logging mode
<code>--trace</code>	Enable API Trace Mode
<code>--server SERVER_NAME</code>	Illumio API Access gateway server name
<code>--login-server LOGIN_SERVER</code>	Illumio login server name
<code>--kerberos-spn KERBEROS_SPN</code>	proxy server
<code>--proxy-port PROXY_PORT</code>	proxy port
<code>--proxy-server-username PROXY_SERVER_USERNAME</code>	proxy server username
<code>--proxy-server-password PROXY_SERVER_PASSWORD</code>	proxy server password

The command for using an API Key with an unauthenticated proxy:

```
ilo use_api_key --key-id <key_id> --key-secret <secret>
--server <pce_fqdn> --org-id <orgid> --proxy-server
<proxy_server> --proxy-port <proxy_port>
```

The command for using an API Key with an authenticated proxy:

```
ilo use_api_key --key-id <key_id> --key-secret
<secret> --server <pce_fqdn> --org-id <orgid> --proxy-
server <proxy_server> --proxy-port <proxy_port> --proxy-
server-username <proxy_username> --proxy-server-password
<proxy_password>
```

After a command is executed, all connections/traffic from this point on will use the proxy.

Import and Export Security Policy

You can export and import security policy to and from the PCE using the CLI Tool. Importing and exporting security policy is particularly useful for moving policy from one PCE to another to avoid recreating policy from scratch on the target PCE. For example:

- You can test the policy on a staging PCE and then move it to your production PCE.
- You can move the policy from a proof-of-concept PCE deployment to your production PCE.

Export and Import Policy Objects

You can use the CLI Tool to export or import the following objects in the PCE:

- Labels: `labels`
- Label groups: `label_groups`
- Pairing profiles: `pairing_profiles`
- IP lists: `ip_lists`
- Services: `services`
- Rulesets and rules: `rule_sets`

About Exporting Rules

You can export rules for workloads, virtual services, or virtual servers.

Illumio recommends that you base your security policy rules on labels for flexibility. Do not tie the rules to specific individual workloads, virtual services, or virtual servers.

Virtual servers and virtual services are not exported.

The CLI Tool policy export does not include such references. A warning is displayed on export when you have rules tied to individual workloads, virtual services, or virtual servers. Attempts to import such rules fail, and the reason for the failure is displayed.

Example: Failed Attempt to Export Rules for Workload

```
WARNING: rule /orgs/1/sec_policy/active/rule_sets/3/
sec_rules/39 contains non-transferrable providers: workload /
orgs/1/workloads/a51ae67d-472a-44c3-984e-d518a8e95aee
Unable to proceed, please verify input
```

Workflow for Security Policy Export/Import

- Authenticate to the source PCE.
- Export the policy to a file. Syntax summary:

```
ilo sec_policy export --file someExportFilename
```

- Authenticate to the target PCE.
- Import the saved policy. Syntax summary:

```
ilo sec_policy import --file someImportFilename
```

Output Options, Format, and Contents

All exported policy is written to standard output. To write to a file, use the `--file` option.

The exported policy is in JSON format.

By default, all supported policy objects are exported. You can export a subset of policy by specifying one or more resource types with the `-resource` option (`labels`, `label_groups`, `pairing_profiles`, `ip_lists`, `services`, or `rule_sets`).

When a subset of policy items is exported (such as only labels), all referenced resources are also exported.

See also [About Exporting Rules \[487\]](#) for information.

Exported Rulesets

With the `--rule_set` option, you can export multiple rulesets.

By default, only the most recently provisioned, active policy is exported. To export the current draft policy or a previous policy, use the `--pversion` state option. See [List Draft or Active Version of Rulesets \[483\]](#) for information.

For a single ruleset, make sure the `--pversion` state you specify matches the provisioned state of the ruleset. In the following example, the state is draft:

```
ilo sec_policy export --pversion draft --rule_set /orgs/1/
sec_policy/draft/rule_sets/1
```

Effects of Policy Import

All imported policies are read from standard input unless you import from a file with the `--file` option.

You can import policy files multiple times. Each import affects only a single copy of a resource.

All imported policies are set in the draft provisioned state. After the import, you must explicitly provision the active state.

Non-transferrable policy rules (that is, rules tied to specific workloads, virtual servers, and bound services), the import aborts with a warning. See [About Exporting Rules \[487\]](#) for information.

Policy items already on the target PCE are updated by imported resources whose names match existing resources' names. Services do not have to have the same names. Services match if they have the same set of ports and protocols.

An import does not delete resources. For example, if you export policy from PCE-1 to PCE-2, delete a resource "R" from PCE 1, and then export and import again, resource "R" is still present on PCE 2. You must explicitly delete resource "R" from PCE2.

Upload Vulnerability Data

This section describes how to use the **ilo** commands to upload vulnerability data to the PCE for analysis in Illumination.

After uploading the data, you can use Vulnerability Maps in the PCE web console to gain insights into the exposure of vulnerabilities and attack paths across your applications running in data centers and clouds. See the "Vulnerability Maps" topic in the Visualization Guide for information.

Add the License for Vulnerability Data Upload

An Illumio Segmentation for Data Centers Vulnerability Maps license is required to upload vulnerability data into the Illumio PCE. For information about obtaining the license, contact Illumio Customer Support.

You are provided with a license file named **license.json**. After you have obtained your license key, store it in a secure location.



NOTE

Before adding the license, you must first authenticate to the PCE.

To add the license, you must be the organization owner or a be a user who has owner privileges.

Use the following command to inform the PCE of your valid license:

```
ilo license create --license-file "path_to_license_file/  
license.json" --feature "feature_name" [debug [v | verbose]  
trace]
```

Where:

What	Required?	Description
"path_to_license_file/license.json"	Yes	The quoted path to the <code>license.json</code> file from Illumio Example: " <code>~/secretDir/license.json</code> "
"feature_name"	Yes	The quoted string " <code>vulnerability_maps</code> ", which specifies the feature name the license enables
<code>debug</code>	No	Enable debugging
<code>v verbose</code>	No	For verbose logging
<code>trace</code>	No	Enable API trace

Vulnerability Data Upload Process

On upload, the CLI Tool associates a workload's IP addresses with corresponding vulnerabilities identified for that workload.

Using API to Download Vulnerability Data

Starting from the release of CLI 1.4.0, Qualys supports API downloads with some minor differences in options.

For the release CLI 1.4.1, it is suggested that users use an API key instead of a login session while using Qualys API download.

For the release CLI 1.4.2 for Tenable, the most reliable way to provide authentication is through API keys instead of username/password. If customers observe any authentication issues while using Tenable SC API upload, they are advised to use API keys.

There are 2 ENV variables to set up the Tenable SC API keys which are used for authentication:

TSC_ACCESS_KEY

TSC_SECRET_KEY

The API connects directly to the cloud instance of Tenable or Qualys and the vulnerability tool then scans new vulnerabilities and downloads them into the PCE.

Users can also set up cron jobs that run in the desired intervals and check the state of the vulnerability scanner.

Qualys and Tenable scanners work in a similar way, using the username and password and similar options.

Automating Vulnerability Imports from Tenable-SC

Users of Illumio vulnerability maps can automate the import of vulnerabilities from tenable-sc using a script.

Illumio CLI supports the API username and password as environment variables or a cmd line switch (such as `--api-password`).

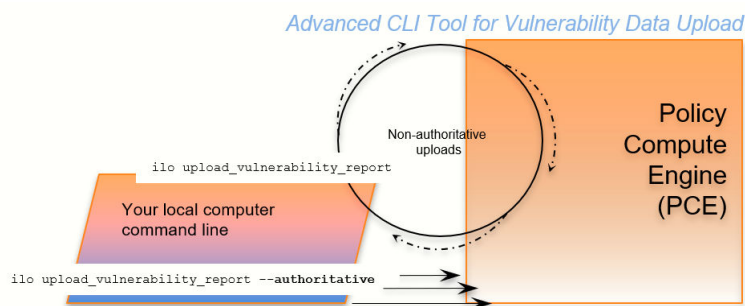
The ILO-CLI tool was updated to add a switch for `--api-user`.

Kinds of Vulnerability Data Uploads

There are two kinds of upload: non-authoritative and authoritative.

- **Non-authoritative:** This is the default. A non-authoritative upload:
 - Appends incoming data to any previously loaded records
 - Accumulates records for the same workloads without regard to duplicates.

You can repeat the non-authoritative upload as many times as you like until you are satisfied with the results.



- **Authoritative:** You indicate authoritative data with the `-authoritative` option. An authoritative upload:
 - Overwrites any previously uploaded records for workloads matched to the incoming records.
 - Eliminates duplicate records.
 - Adds new records not previously written by other uploads.

You can repeat the authoritative upload as many times as you like until you are satisfied with the results.

After either kind of upload, you can examine the uploaded data with the CLI Tool or the PCE web console. See “Vulnerability Maps” in the Visualization Guide for information.

Supported Vulnerability Data Sources

The CLI Tool works with vulnerability data from the following sources.

- Nessus Professional™
- Qualys®
- Tenable Security Center
- Tenable.io
- Rapid7©



NOTE

Before uploading Rapid7 data to the PCE, export the data from Rapid7 to Qualys format with Qualys XML Export.

Vulnerability Data Formats

In the CLI 1.4.0, 1.4.1 and 1.4.2 releases, Illumio supports the following report formats:

- For tenable-io: API, CSV
- For tenable-sc: API, CSV
- For nessus-pro: XML
- For qualys: API, XML

Common Vulnerabilities and Exposures (CVE)

Vulnerabilities are defined by Common Vulnerabilities and Exposures (CVE), with identifiers and descriptive names from the U.S. Department of Homeland Security [National Cybersecurity Center](#).

Vulnerability Scores

Illumio computes a vulnerability score, which measures the vulnerability of your entire organization. The score is displayed by the ilo vulnerability list command for all vulnerabilities or individual vulnerabilities via the vulnerability identifier.

Vulnerability Identifier

An uploaded vulnerability has an identifier, as shown in the example below. The vulnerability identifier is tied to a specific CVE. You use this identifier with **--reference-id** option to examine specific uploaded vulnerabilities. See [Example – List Single Uploaded Vulnerability \[503\]](#) for information.

The following are examples of vulnerability identifiers.

- Nessus Professional: nessus-65432
- Qualys: qualys-23456
- Rapid7: qualys-98765. Because Rapid7 data is first exported from Rapid7 in Qualys format, it is given a Qualys identifier when uploaded to the PCE.

Vulnerabilities for Unmanaged Workloads

You can upload vulnerabilities for unmanaged workloads. However, unmanaged workloads do not have any vulnerability score or associated CVE. This information becomes available if the unmanaged workload is later changed to managed.

Prerequisites for Vulnerability Data Upload

Before uploading vulnerability data, ensure you are ready with the following requirements.

- An Illumio Vulnerability Maps license is required to upload vulnerability data to the PCE. See [Add the License for Vulnerability Data Upload \[490\]](#) for information.
- XML-formatted vulnerability data files from one of the supported sources.
- Authenticated CLI-tool access to the target PCE.
- Authenticated access and necessary permissions in the PCE web console for working with vulnerability maps.

Vulnerability Data Upload CLI Tool Syntax

The key argument and options for uploading vulnerability data are as follows. For readability, this syntax is broken across several lines.

```
ilo upload_vulnerability_report
--input-file path_to_datafile.xml [path_to_datafile.xml]...
--source-scanner [nessus-pro|qualys|tenable-sc|tenable-io]
--format xml
[--authoritative]
[ --api-user ApiServerUserName --api-server
SourceApiServer:port ]
```

Where:

What	Required	Description
<code>--enable-proxy</code>	No	Use this to enable the proxy between tenable and CLI. Use this command to enable the proxy: <pre>ilo upload_vulnerability_report --source-scanner tenable-sc --format api --severities=3 --enable-proxy -v --debug</pre> Use this command if you do not want to enable the proxy: <pre>ilo upload_vulnerability_report --source-scanner tenable-sc --format api --severities=3 -v --debug</pre>
NOTE This is available in CLI Tool 1.4.4.		
<code>--input-file path_to_datafile.xml [path_to_datafile.xml]...</code>	Yes	Location of one or more data files to upload. The path to the data file can be either an absolute path or a relative path.

What	R e q u i r e d	Description
		If more than one data file is listed (bulk upload), separate the file names with space characters.
<code>--debug</code>	No	Enable debugging
<code>--authoritative</code>	No	For uploading authoritative vulnerability data. The default command is without the <code>--authoritative</code> option. See Kinds of Vulnerability Data Uploads [492] for information.
<code>--workload-cache FILE</code>	No	DEBUGGING ONLY: Workload Cache file - use this if available
<code>--source-scanner [nessus-pro qualys tenable-sc]</code>	Yes	Indicates the source of the scan. Note for rapid data: - Vulnerability data from Rapid must have been exported from Rapid in Qualys XML format. - To load the Rapid data, use the 'qualys' argument
<code>--format</code>	Yes	Report format. Allowed values are:
<code>REPORT_FORMAT</code>		xml <code>--source-scanner nessus-pro</code> <code>--source-scanner qualys</code> csv <code>--source-scanner tenable-sc</code> <code>--source-scanner tenable-io</code> api <code>--source-scanner tenable-sc</code> <code>--source-scanner qualys</code> <code>--source-scanner nessus-pro</code> See also <code>--api-server</code> and <code>--api-user</code> .
<code>--api-server SourceApi-Server:port</code>	Yes	API server FQDN. Allowed formats are HOST or HOST:PORT
<code>SERVER_FQDN</code>	Optional	

What	R e q u i r e d	Description
	T e n a b l e w i t h - - f o r m a t a p i	
--api-user ApiServerU- serName USERNAME	Y e s f o r s o u r c e A P I s e r v e r a u t h e n t	The user name for authenticating to the SourceApiServer. You are always prompted to enter your password.

What	R e q u i r e d	Description
	i c a t i o n	
--api-page-size PAGE_SIZE	Y e s f o r Q u a l i t y s a n d T e n a b l e	Appropriate page size if API supports pagination. The default page is 1000.
--skip-cert-verification	Y e s f o r Q u a l i t y s a n d T e n a b	Disable certificate verification for API.

What	Required	Description
<code>--on-premise</code>	Yes only for Tenableable.io	Tenable IO deployment is on-premise.
<code>--mitigated</code>	Yes only for Tenableable scans	Tenable SC input is exported from the mitigated vulnerabilities analysis view.
<code>--scanned-after</code>	Yes for	Qualys users can select scan data to process after a specific date, in ISO 8601 format.
<code>SCANNED_AFTER</code>	or Qualys	When the optional scanned-after option is not provided, the system will pull all the historical vulnerability records from your Qualys account. If your account has historical records, it may take a very long time for the first time. With the scanned-after option, vulnerability

What	Required	Description
	always	data scanned after a specific date will be extracted and uploaded. Including a particular scanned-after time is recommended if you use Qualys API upload option for the first time.
--severities SEVERITIES	No	Qualys API users can select vulnerabilities with defined severity levels to include in their reports. Users can filter based on severity and avoid severity levels 1 and 2, which are often very informational and noisy. Example: --only-include-severity=3,4,5 For Windows, be sure to include quotes around the severity levels: Example: --only-include-severity="3,4,5" NOTE: This option was added in Release 1.4.1
-v, --verbose	No	Verbose logging mode
--trace	No	Enable API trace mode.

Using the ILO Command with Windows Systems

Windows systems take up to four options with the ILO command for the vulnerability data upload. Users who choose to use more optional parameters must set **api-server**, **username**, and **password** as the environmental variables to use other options in the command.

Work with Vulnerability Maps in Illumination

See "Vulnerability Maps" in Visualization Guide for information.

Vulnerability Data Examples

Example - Upload Non-Authoritative Vulnerability Data

In this example, the **--source-scanner nessus-pro** option indicates that the data comes from Nessus Professional. On Windows, provide the absolute

path to the data file. This Windows example is broken across several lines with the PowerShell line continuation character (`).

```
C:\Users\donald.knuth> ilo upload_vulnerability_report `
--input-file
C:\Users\donald.knuth\Desktop\vuln_reports\nessus3.xml `
--source-scanner nessus-pro --format xml

Elapsed Time [0.05 (total : 0.05)] - Data parsing is done.
Elapsed Time [1.08 (total : 1.13)] - Got workloads. Workload
count: 5.
Elapsed Time [0.0 (total : 1.13)] - Built workload interface
mapping. Total interfaces : 11.
Elapsed Time [4.57 (total : 5.7)] - Imported Vulnerabilities..
Elapsed Time [0.0 (total : 5.7)] - Detected Vulnerabilities
are associated with vulnerability and workload data..
Elapsed Time [0.83 (total : 6.53)] - Report Imported.

Summary:
Processed the report with the following details :
Report meta data =>
Name           : Generic
Report Type    : nessus
Authoritative  : false
Scanned IPs    : ["10.1.0.74", "10.1.0.223", "10.1.0.232",
"10.1.0.221", "10.1.0.11", "10.1.0.82", "10.1.0.43",
"10.1.0.91", "10.1.0.8", "10.1.1.250"]

Stats :
  Number of vulnerabilities           => 19
  Number of detected vulnerabilities => 31

Done.
```

Example – Upload of Rapid7 Vulnerability Data

The syntax for uploading vulnerability data from Rapid7 is identical to the syntax for uploading vulnerability data from Qualys. On Windows, you use the **--format qualys** option and the absolute path to the data file. This Windows example is broken across several lines with the PowerShell line continuation character (`).

Rapid7 data exported in Qualys format.

Before uploading to the PCE, Rapid7 vulnerability data must have been exported in Qualys format from Rapid7 with Qualys XML Export.

```
C:\Users\edward.teller> ilo upload_vulnerability_report `
--input-file
C:\Users\edward.teller\Desktop\vuln_reports\rapid7.xml `
--source-scanner qualys --format xml
...
Done.
```

Example - Upload Authoritative Vulnerability Data

In this example, the prompt shows this is an authoritative upload.

To proceed, you must enter the word YES in all capital letters.

```
C:\Users\jrobert.oppenheimer> ilo upload_vulnerability_report
--input-file dataDir/authoritativedata.xml --authoritative --
source-scanner qualys --format xml
```

```
Using /home/centos/.rvm/gems/ruby-2.4.1
Authoritative scan overwrites the previous entries for all the
ips within this scan. There is no ROLLBACK
Are you sure this is an authoritative scan? (YES | NO)
YES
Elapsed Time [11.86 (total : 11.86)] - Data parsing is done.
Elapsed Time [0.27 (total : 12.13)] - Got workloads. Workload
count: 3.
Elapsed Time [0.0 (total : 12.13)] - Built workload interface
mapping. Total interfaces : 6.
Elapsed Time [3.02 (total : 15.15)] - Imported
Vulnerabilities..
Elapsed Time [0.0 (total : 15.15)] - Detected Vulnerabilities
are associated with vulnerability and workload data..
Elapsed Time [0.84 (total : 16.0)] - Report Imported.
Summary:
Processed the report with the following stats -
    Number of vulnerabilities          => 14
    Number of detected vulnerabilities => 48
Done.
```

Example - List Single Uploaded Vulnerability

This example uses a single Qualys vulnerability identifier to show the associated vulnerability. The value passed to the `--reference-id` option is shown as qualys-38173. See [Vulnerability Identifier \[494\]](#) for information.

```
$ ilo vulnerability read --xorg-id=1 --reference-id=qualys-38173
...

| Attribute | Value |
+-----+
+-----+
--+
| href | /orgs/1/vulnerabilities/qualys-38173 |
| name | SSL Certificate - Signature Verification Failed
Vulnerability
| score | 39 |
| cve_ids | [] |
| created_at | 2018-11-05T18:16:56.846Z |
...
```

Example – List All Uploaded Vulnerabilities

This example highlights the vulnerability identifier, the CVE identifiers, and the description of the CVE. See [Common Vulnerabilities and Exposures \(CVE\) \[493\]](#) and [Vulnerability Identifier \[494\]](#) for information. The layout of the output is the same for all supported vulnerability data sources.

Nessus Professional

```
C:\Users\werner.heisenberg> ilo vulnerability list --xorg-id=1
...
| Href | Name | Score | Description | Cve Ids | Created At |
Updated At | Created By | Updated By |
+-----+
+-----+
| /orgs/1/vulnerabilities/nessus-18405 | Microsoft Windows
Remote Desktop Protocol Server Man-in-the-Middle Weakness
| 51 | | ["CVE-2005-1794"]
| 2018-11-07T03:15:39.410Z | 2018-11-07T03:15:39.410Z
| {"href"=>"/users/1"} | {"href"=>"/users/1"} |
...
```

Qualys

```
C:\Users\isaac.newton> ilo vulnerability list --xorg-id=1
...
| Href | Name | Score | Description | Cve Ids | Created At |
Updated At | Created By | Updated By |
-----+-----
+-----+-----+
| /orgs/1/vulnerabilities/qualys-38657 |
Birthday attacks against TLS ciphers with 64bit
block size vulnerability (Sweet32) | 69 | | ["CVE-2016-2183"]
| 2018-07-27T18:16:57.166Z | 2018-08-08T22:30:32.421Z
| {"href"=>"/users/1"} | {"href"=>"/users/16"} |
...
```

Rapid7

Because Rapid7 vulnerability data must be in Qualys format before upload, the output is the same as for Qualys data, including the vulnerability identifier (qualys-38657 in the example above) and CVE. See [Common Vulnerabilities and Exposures \(CVE\) \[493\]](#) and [Vulnerability Identifier \[494\]](#) for information.

Example - View Vulnerability Report

The Report Type column identifies the source of the scan; in this example, Qualys.

```
C:\Users\gracemurphy.hopper> ilo vulnerability_report list --
xorg-id=1
...
| Href | Report Type | Name | Created At | Updated At | Num
Vulnerabilities | Created By | Updated By |
-----+-----
+-----+-----+
+-----+-----+
| /orgs/1/vulnerability_reports/scan_1502310096_09344 |
qualys | NewAuthoritativeScan | 2018-08-08T22:30:34.877Z |
2018-08-08T22:30:34.877Z | 62 | {"href"=>"/users/16"} |
{"href"=>"/users/16"} |
...
```

Example - Upload a Qualys Report Using API

```
upload_vulnerability_report --source-scanner qualys --format
api
--api-server qualysguard.qg3.apps.qualys.com --api-user um3sg
--scanned-after 2021-09-20
```

CLI Tool Tutorials

This section provides several hands-on exercises that demonstrate step-by-step how to perform common tasks using the CLI Tool.

How to Import Traffic Flow Summaries

Static Illumination provides “moment-in-time” visibility of inter-workload traffic. This visibility is useful to model policies, to look for specious traffic flows, and to ensure that metadata for labels is accurate.

Goal

Load workload and traffic data needed for analysis with static Illumination.

Setup

This tutorial relies on the following data to import.

- 1,000 workloads defined in the file **bulkworkloads-1000.csv**, which has the following columns:

```
hostname,ips,os_type
10.14.59.8.netstat,10.14.59.8,linux
10.4.78.178.netstat,10.4.78.178,linux
10.37.134.179.netstat,10.37.134.179,linux
...
```

- 1,000,000 traffic flows defined in the CSV file **traffic.clean-1m.csv**, which has the following columns:

```
src_ip,dst_ip,dst_port,proto
10.40.113.86,10.14.59.8,10050,6
10.14.59.8,10.8.251.138,8080,6
10.40.113.124,10.14.59.8,22,6
...
```

Steps

The workflow is authenticated to the PCE and run two **ilo bulk_upload_csv** commands.

1. Authenticate to the PCE via API key or explicit login.
2. Load the workload data:

```
ilo workload bulk_upload_csv --file bulkworkloads-1000.csv
```


3. Load the traffic flow data:

```
ilo traffic bulk_upload_csv --file traffic.clean-1m.csv
```

Results

The data from the CSV files are uploaded.

How to Create Kerberos-Authenticated Workloads

This tutorial describes how to create workloads that use Kerberos for authentication. The tutorial makes the following assumptions:

- This tutorial assumes that you already have your Kerberos implementation in place.
- As Kerberos requires, the Kerberos realm name is shown in all capital letters as **MYREALM**.
- VEN environment variables must be set *before* VEN installation. Environment variables for Linux are detailed in the VEN Installation and Upgrade Guide.

Goals

- Create two workloads on Linux that are authenticated by Kerberos.
- Set the workloads' modes to idle and illuminated.

Setup

The key data for using the **ilo** command to create these workloads are the name of the Kerberos realm and the Service Principle Name (SPN).

Steps

The workflow is authenticate, run two **workload create** commands that set the workloads' modes, set the VEN environment variables, install the VEN, and run two Kerberos **kinit** commands to get Kerberos tickets for the workloads.

1. Authenticate to the PCE via API key or explicit login.
2. Create Kerberos-authenticated **myWorkload1** and set its **mode** to **idle**:

```
ilo workload create --hostname myPCE.BigCo.com  
--name myWorkload1 --service-principal-name host/  
myKerberosTicketGrantingServer@MYREALM --agent/config/mode  
idle
```

For information about how the mode is a nested field, see [Nested Resource Fields and Wildcards \[478\]](#).

3. Create Kerberos-authenticated `myWorkload2` and set its mode to `illuminated`:

```
ilo workload create --hostname myPCE.BigCo.com
--name myWorkload2 --service-principal-name host/
myKerberosTicketGrantingServer@MYREALM --agent/config/mode
illuminated
```

4. Before installation, set VEN environment variables:

```
# Activate on installation
VEN_INSTALL_ACTION=activate
# FQDN and port PCE to pair with
VEN_MANAGEMENT_SERVER=myPCE.BigCo.com:8443
# Kerberos Service Principal Name
VEN_KERBEROS_MANAGEMENT_SERVER_SPN=host/
myKerberosTicketGrantingServer
# Path to Kerberos shared object library
VEN_KERBEROS_LIBRARY_PATH=/usr/lib/libgssapi_krb5.so
```

5. Install the Linux VEN:

```
rpm -ivh illumio-ven*.rpm
```

6. Run `kinit` to get a Kerberos ticket for `myWorkload1`:

```
kinit -k -t /etc/krb5.keytab host/
myWorkload1.BigCo.com@MYREALM
```

7. Run `kinit` to get a Kerberos ticket for `myWorkload2`:

```
kinit -k -t /etc/krb5.keytab host/
myWorkload2.BigCo.com@MYREALM
```

Results

The Kerberos-authenticated workloads are created, set in the desired modes, and given a Kerberos ticket.

How to Work with Large Datasets

The `--async` option is for working with large data sets without waiting for the results. The option works like “batch job.”

The option can be used with any resource. The workflow is as follows:

1. You issue the desired `ilo` command with the `--async` option, which displays a job ID.

2. You take note of the job ID.
3. Your session is freed up while the job runs.
4. The job creates a data file, which you view with `datafile --read --job-id jobID`.

Goal

Get a report of a large workload data set.

Steps

1. Issue the `--async` request for a workload list. Take note of job ID, which is the final word of the href displayed on the Location line.

```
[kurt.goedel~]$ ilo workload list --async
Using /home/kurt.goedel/.rvm/gems/ruby-2.2.1
Location: /orgs/1/jobs/fe8a1c2b-1674-4b83-8967-eb56c4ffa1e3
202, Accepted
```

2. Check to see if the job completed. Use the job ID from the **Location** output in previous command:

```
[sigmund.freud~]$ ilo job read --job-id
fe8a1c2b-1674-4b83-8967-eb56c4ffa1e
Using /home/sigmund.freud/.rvm/gems/ruby-2.2.1
```

3. Download the resulting data file, specifying the job ID with `-uuid jobID`:

```
[bill.gates ~]$ ilo datafile read --uuid 1e1c1540-8a01-0136-ec14-02f4d6c1190c
Using /home/ bill.gates /.rvm/gems/ruby-2.2.1
+-----+
+-----+-----+
... Many lines not shown
+-----+
+-----+-----+
| Href
| Deleted | Name | Description | Hostname
| Service Principal Name | Public
Ip          | Distinguished Name |
External Data Set | External Data Reference
| Interfaces | Ignored Interface Names
| Service Provider | Data Center
| Data Center Zone | Os Id
| Os Detail | Online | Labels | Services | Agent
| Created At          | Created By
| Updated At          | Updated By
+-----+
+-----+-----+
... More lines not shown
+-----+
| /orgs/1/workloads/50ce441e-75ac-4be8-9201-96169545019c
| false | | | 10.14.59.8.netstat
...
... Many lines not shown
...
```

How to Upload Vulnerability Data

This example tutorial shows how to upload vulnerability data to the PCE. For more information, see [Upload Vulnerability Data \[490\]](#). The source of the vulnerability data in this example comes from Qualys®.

Goal

Upload authoritative vulnerability data for analysis in Illumination.

Steps

1. Do a non-authoritative upload of vulnerability data for examination:

```
ilo upload_vulnerability_report --input-file C:\Users\albert-einstein0.xml --source-scanner qualys --format xml
```

2. Examine a single uploaded vulnerability record identified by its vulnerability identifier, **qualys-38173**. See [Vulnerability Identifier \[494\]](#) for information.

```
ilo vulnerability read --xorg-id=1 --reference-id=qualys-38173
```

3. Do another non-authoritative upload of vulnerability data.

```
ilo upload_vulnerability_report --input-file C:\Users\albert-einstein99.xml --source-scanner qualys --format xml
```

4. Do an authoritative upload of vulnerability data, overwriting any previously uploaded records and adding any new vulnerability records.

```
ilo upload_vulnerability_report --input-file C:\Users\albert.einstein_FINAL.xml --authoritative --source-scanner qualys --format xml
```

Results

The authoritative vulnerability data has been uploaded and is ready for use in Illumination.

Illumio Core PCE CLI Tool Guide 1.4.2

Overview of the CLI Tool

This topic provides an overview of the CLI Tool, describes the general syntax of the CLI Tool command, and lists the environment variables you can use to customize the CLI Tool.



IMPORTANT

See the *Illumio Core CLI Tool 1.4.0 Release Notes* and *Illumio Core CLI Tool 1.4.2 Release Notes* and *Illumio CORE CLI Tool 1.4.2 Release Notes* in your respective Illumio Core Technical Documentation portal for the updates to the CLI Tool for these releases.

About This Guide

The following sections provide useful information to help you get the most out of this guide.

CLI Tool Versioning

Illumio Segmentation for Data Centers CLI Tool version 1.4.2 is compatible with Illumio Segmentation for Data Centers PCE versions:

PCE 19.3.6-H2 (LTS)

PCE 21.2.4 (LTS)

PCE 21.5.20 (LTS)

PCE 22.1.1 (Standard)

PCE 22.2.0 (Standard)

The CLI Tool version numbering is independent of PCE and VEN's release and version numbering. The CLI Tool works with multiple versions of the PCE

and the VEN and does not necessarily need software changes in parallel with releases of the PCE or the VEN.



IMPORTANT

See the *Illumio Core CLI Tool 1.4.0 Release Notes*, *Illumio Core CLI Tool 1.4.1 Release Notes* and *Illumio Core CLI Tool 1.4.2 Release Notes* in your respective Illumio Core Technical Documentation portal for the updates to the CLI Tool for these releases.

How to Use This Guide

This guide includes several major sections:

- Overview of the CLI Tool
- Installation
- The formal syntax of the **ilo** command
- Tutorials for various operations
- Uploading vulnerability data
- Security policy import and export

Before Reading This Guide

Before performing the procedures in this guide, be familiar with the following information:

- The CLI Tool interacts with the PCE; therefore, be familiar with PCE concepts such as core and data nodes, workloads, and traffic. See the PCE Administration Guide.
- The CLI Tool is often used to upload vulnerability data; therefore, understand how vulnerability data is used in the PCE web console. See the "Vulnerability Maps" topic in Visualization Guide.
- The CLI Tool can be used with workload data; therefore, you must understand what workloads are. See the "VEN Architecture and Components" topic in VEN Administration Guide.
- The CLI Tool can be used with security policy rules, rulesets, labels, and similar resources; therefore, be familiar with these concepts. See "The Illumio Policy Model" in Security Policy Guide.

Notational Conventions in This Guide

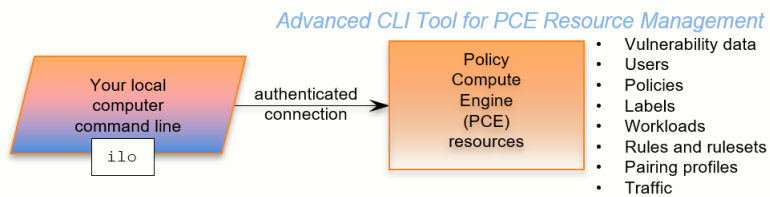
- Newly introduced terminology is italicized. Example: *activation code* (also known as pairing key)

- Command-line examples are monospace. Example: `illumio-ven-ctl --activate`
- Arguments on command lines are monospace italics. Example: `illumio-ven-ctl --activate activation_code`
- In some examples, the output might be shown across several lines but is actually on one single line.
- Command input or output lines not essential to an example are sometimes omitted, as indicated by three periods in a row. Example:

```
...
some command or command output
...
```

CLI Tool and PCE Resource Management

The Illumio CLI Tool allows you to manage many of your PCE resources directly from your local computer.



Use the CLI Tool to:

- Import vulnerability data for analysis with Illumination.
- Help with tasks such as directly importing workload information to create workloads in bulk.
- Create, view, and manage your organization's security policy rules, rulesets, labels, and other resources.



CAUTION

The CLI Tool is a tool that you can use to work with your PCE resources. Test your CLI Tool commands against a non-production system before using them on your production PCEs.

The CLI Tool is named **ilo**. It is a wrapper around the Illumio Segmentation for Data Centers REST API. No knowledge of the REST API is required.

The ilo Command

Learn about the general syntax of the CLI Tool command, **ilo**, and how to use the command-line help to get more specific syntax information.

CLI Tool Formal Syntax

The formal syntax for the **ilo** command is:

```
ilo resource_or_specialCommand argument options
```

Where:

- **resource_or_specialCommand** represents either a resource managed by the PCE or a command unrelated to a particular resource.
A resource is an object that the PCE manages, such as a workload, label, or pairing profile.
Example resource command on Linux (create a workload):

```
ilo workload create --name FriendlyWorkloadName --hostname  
myWorkload.BigCo.com
```

A special command is a command that is not related to a specific resource. Special commands include **user**, **login**, **use_api_key**, and **node_available**.

Example special command on Windows (log out of PCE):

```
ilo user logout --id 6
```

- The **argument** represents an operation on the resource or special command.
- The **options** are allowed options for the **resource_or_specialCommand**. The specific option depends on the type of resource or special command.

CLI Tool Help

To get a complete list of all the available CLI Tool commands, use the **ilo** command without options. This command displays the high-level syntax of special commands, resources, and their allowable options.

For details about a resource's or special command's arguments, specify the resource's name followed by the argument followed by the **--help** option. For example:

```
ilo workload create --help
```

HTTP Response Codes and Error Messages

Learn about the response codes and error messages that are returned using CLI Tool commands.

REST API HTTP Response Codes

At the end of its output, the `ilo` command displays the REST API HTTP response code from the command. For example, a successful operation shows the following output:

```
...
200, OK
```

Error Messages

For many syntactical or other types of errors, the CLI Tool displays a general message encouraging you to verify your syntax with the CLI Tool help:

The `ilo` command has encountered an error. Check your syntax with either of the following commands:

```
- ilo
- ilo <command> --help
```

In some circumstances, the CLI Tool writes a detailed log of errors:

For detailed error messages, see the file:
`location-of-local-temp-directory/illumio-cli-error.log`

Where `location-of-local-temp-directory` is:

- Linux: `/tmp`
- Windows: `C:\Windows\Temp`

Environment Variables

Illumio provides Linux environment variables to allow you to customize the operation of the CLI tool.

Environment Variable	Purpose
ILO_API_KEY_ID	API key for non-password-based authentication and cookie-less session with PCE. See "Authenticate with an API Key".
ILO_API_KEY_SECRET	API key secret for non-password-based authentication and cookie-less session with PCE. See "Authenticate with an API Key".
ILO_API_VERSION	API version to be used to execute CLI commands. Set this to override the default API version. See "Set the Illumio ASP REST API Version." Default: v2. Example: <code>\$ export ILO_API_VERSION=v1</code>
ILO_CA_DIR	Directory that contains certificates. See "TLS/SSL Certificate for Access to the PCE".
ILO_CA_FILE	Absolute path to the certificate file. See "TLS/SSL Certificate for Access to the PCE".
ILO_DISPLAY_CONFIG	An absolute path to the display configuration file is to be used with the list command. See "Linux Save Specific Fields to File For Reuse".
ILO_INSECURE_PASSWORD	Provide a password for login. If this variable is set, the login password prompt does not appear, and this password is used instead. Do not use in a production system when authentication security is desired. Example: <code>\$ export ILO_INSECURE_PASSWORD=myInsecurePassword</code>
ILO_KERBEROS_SPN	Kerberos service principal name (SPN). Specify this variable when using Kerberos authentication.
ILO_LOGIN_SERVER	PCE login server FQDN. Use this variable when the login server FQDN is not the same as the PCE FQDN. See "Explicit Log into the PCE".
ILO_ORG_ID	Organization identifier for certificate-authenticated session with PCE. Value is always 1. Does not need to be explicitly set The environment variable is set by the system and should not be explicitly set. See "Authentication to PCE with API Key or Explicit Login".
ILO_PCE_VERSION	PCE version for the CLI to use. Default: 19.1.0 Example: <code>\$ export ILO_PCE_VERSION=18.2.5</code>
ILO_PREVIEW	Enable any preview features that are included in this release. To disable preview features, remove this variable from the environment.
ILO_SERVER	FQDN of PCE for login and authentication with PCE. See "Authentication to PCE with API Key or Explicit Login".
TSC_ACCESS_KEY	These two ENV variables have been added in the release 1.4.2 to set up the Tenable SC API keys, which are used for authentication.
TSC_SECRET_KEY	
TSC_HOST	The variable that specifies the target host for Tenable

Environment Variable	Purpose
QAP_HOST	The variable that specifies the target host for Qualys

Installation and Authentication

Learn how to install the CLI Tool, set up authentication, upgrade the tool, and uninstall it.

Review the prerequisites before you install the PCE CLI Tool.

Prerequisite Checklist

- ☐ License for vulnerability data upload
- ☐ Vulnerability data for upload
- ☐ Functional PCE
- ☐ Supported operating systems
- ☐ TLS/SSL certificate for authenticating to the PCE
- ☐ API version set in configuration
- ☐ The CLI Tool installation program

Installation Prerequisites

This section details the prerequisites for installing the CLI Tool. Be sure you meet the prerequisites in the checklist.

Prerequisite Checklist

- ☐ License for vulnerability data upload
- ☐ Vulnerability data for upload
- ☐ Functional PCE
- ☐ Supported operating systems
- ☐ TLS/SSL certificate for authenticating to the PCE

- ☐ API version set in configuration
- ☐ The CLI Tool installation program

License for Vulnerability Data

The Illumio Segmentation for Data Centers Vulnerability Maps license is required to import vulnerability data into the Illumio PCE. For information about obtaining a license, contact Illumio Customer Support. For information on activating the license, see [Add the License for Vulnerability Data Upload \[490\]](#).

Upload Vulnerability Data

When you plan on using the CLI Tool to upload vulnerability data, make sure you have the data to upload in advance. See [Supported Vulnerability Data Sources \[493\]](#) for information.

Install Functional PCE

Because the CLI Tool is for managing resources on your PCE, you must already have installed a fully functional PCE.

Supported Computer Operating Systems

The CLI Tool is supported by the following operating systems:

Linux

- Ubuntu 18.04
- Ubuntu 20.04
- Centos/RHEL 7.9
- Centos/RHEL 8.4

Microsoft Windows



NOTE

The CLI Tool is not supported on Windows 32-bit CPU architecture. Ensure that you run it on Windows 64-bit CPU architecture.

- Windows 2012 64 bit
- Windows 2016 64 bit

- Windows 10 64 bit

TLS/SSL Certificate for Access to the PCE

You need a TLS/SSL certificate to connect to the PCE securely. Requirements for this certificate are provided in the PCE Installation and Upgrade Guide.

Alternative Trusted Certificate Store

To secure the connection to the PCE, by default, the CLI Tool relies on your computer's trusted certificate store to verify the PCE's TLS certificate. You can specify a different trusted store. When you have installed a self-signed certificate on the PCE, an alternative trusted store might be necessary.

Example: Set envvar for alternative trusted certificate store z

```
export ILO_CA_FILE=~/.self-signed-cert.pem
```

Set the Illumio Segmentation for Data Centers REST API Version

The CLI Tool uses v2 of the Illumio Segmentation for Data Centers REST API by default.

Install, Upgrade, and Uninstall the CLI Tool

This section explains how to install, upgrade, or uninstall the CLI Tool on Linux or Windows.

Download the Installation Package

Download the CLI Tool installation package from the [Tools Catalog](#) page (login required) to a convenient location on your local computer.

Install Linux CLI Tool

The CLI Tool installer for Linux is delivered as an RPM for RedHat/CentOS and DEB for Debian/Ubuntu.

The CLI Tool is installed in the local binaries directory `/usr/local/bin`.

Log into your local Linux computer as a normal user and then use **sudo** to run one of the following commands.

RedHat/CentOS:

```
sudo rpm -ivh /path_to/nameOfCliRpmFile.rpm
```

Debian/Ubuntu:

```
sudo dpkg -i / path_to / nameOfCliDebFile .deb
```

Upgrade Linux CLI Tool

Log into your local Linux computer as a normal user and then use **sudo** to run one of the following commands.

RedHat/CentOS:

```
sudo rpm -Uvh /path_to/nameOfCliRpmFile.rpm
```

Debian/Ubuntu:

```
sudo dpkg -i / path_to / nameOfCliDebFile .deb
```

The same option, **-i**, is used for installation or upgrade.

Uninstall the Linux CLI Tool

Log into your local Linux computer as a normal user and then use **sudo** to run one of the following commands.

RedHat/CentOS:

```
sudo rpm -e nameOfCliRpmFile
```

Debian/Ubuntu:

```
sudo dpkg -r nameOfCliDebFile
```

Install Windows CLI Tool

The CLI Tool installer for Windows is delivered as an **.exe** file.

Log into your local Windows computer as an administrator and start the installation program in the following ways.

- In the Windows GUI, double-click the .exe file.
- In a cmd window, run the .exe.
- In a PowerShell window, run the .exe.

After starting the installation program, follow the leading prompts.

A successful installation ends with the "Installation Successfully Completed" message, and the help text for the CLI Tool is displayed.

Upgrade Windows CLI Tool

The CLI Tool cannot be directly upgraded from an existing CLI Tool installation.

If you have already installed a previous version of the CLI Tool, manually uninstall it using the Windows Control Panel's Add/Remove Programs.

After uninstalling the previous version of the CLI Tool, install the new version of the CLI Tool as described in [Install Windows CLI Tool \[521\]](#).

Uninstall the Windows CLI Tool

Log into your local Windows computer as an administrator, and from the Windows Control Panel, launch Add/Remove Programs.

Select Illumio CLI from the list and click the **Uninstall** button.

Authenticate with the PCE

When using the CLI Tool, you can authenticate to your PCE in the following ways:

- **With an API key and key secret:**

This is the easiest way. Before you create the API key and secret, you need to log in to authenticate to the PCE. After creating and using the key, you do not have to specify your username and password again.

- **With the explicit command to log in:**

This always requires a username and password.

This method also requires you to log out with a user ID displayed at login. The explicit login times out after ten minutes of inactivity, after which you must log in again.

For both authentication mechanisms, on the command line, you always need to specify the FQDN and port of your PCE. The default port for the PCE is 8443. However, your system administrator can change this default. Check with your system administrator to verify the port you need.

Authenticate with an API Key

To authenticate to the PCE with an API key, you must first explicitly log into the PCE, create the API key, and then use the key to authenticate.

1. Authenticate via explicit login:

```
ilo login --server yourPCEfqdn:itsPort
```

2. Create the API key:

```
ilo api_key create --name someLabel
```

someLabel is an identifier for the key.

3. Use the API key to authenticate:

```
ilo use_api_key --server yourOwnPCEandPort --key-id  
yourOwnKeyId --org-id --key-secret yourOwnKeySecret
```

Create an API Key

On Linux, for later ease of use, with the **api_key --create-env-output** option, you can store the API key, API secret, and the PCE server name and port as environment variables in a file that you source in future Linux sessions.

Linux Example

This example creates the API key and secret and stores them as environment variables in a file named **ilo_key_MY_SESSION_KEY**.

```
# ilo api_key create --name MY_SESSION_KEY --create-env-output
# Created file ilo_key_MY_SESSION_KEY with the following
contents:

export ILO_API_KEY_ID=14ea453b6f8b4d509
export
ILO_API_KEY_SECRET=e1fa1262461ca2859fcf9d91a0546478d10a1bcc4c57
9d888
a4e1cace71f9787
export ILO_SERVER=myPCE.BigCo.com:8443
export ILO_ORG_ID=1

# To export these variables:
# $ source ilo_key_MY_SESSION_KEY
```

Log Into the PCE

Without an API key, you must explicitly log into the PCE.

For on-premises PCE deployments, the login syntax is the FQDN and port of the PCE:

```
ilo login --server yourPCEfqdn:itsPort
```

For *yourPCEfqdn:itsPort*, do not specify a URL instead of the PCE's FQDN and port. If you do, an error message is displayed.

For the Illumio Secure Cloud customers, the login syntax is:

```
ilo login --server URL_or_bare_PCEfqdn:itsPort --login-server
login.illum.io:443
```

See the explanation above about the argument to the **--server** option.

- After login, the output of the command shows a user ID value. Make a note of this value. You need it when you log out.
- The session with the PCE remains in effect as long as you keep using the CLI Tool. After 10 minutes of inactivity, the session times out, and you must log in again.

Example

In this example, the user ID is 6.

```

C:\Users\marie.curie> ilo login --server myPCE.BigCo.com:8443
Enter User Name: albert.einstein@BigCo.com
Enter Password: Welcome Albert!
User ID = 6
Last Login Time 2018-08-10T-09:58:07.000Z from someIPAddress
Access to Orgs:
Albert: (2)
Roles: [3]
Capabilities: {"basic"=>["read", "write"],
"org_user_roles"=>["read", "write"]}
User Time Zone: America/Los_Angeles
Server Time: 2018-08-12T17:58:07.522Z
Product Version: 16.09.0-1635
Internal Version:
48.0.0-255d6983962db54dc7ca627534b9f24b94429bd5
Fri Aug 6 16:11:50 2018 -0800
Done

```

Log Out of the PCE

To end a session with the PCE, use the following command:

```
ilo user logout --id valueOfUserIdFromLogin
```

Where:

- *valueOfUserIdFromLogin* is the user ID associated with your login. See [Log Into the PCE \[524\]](#) for information.

Example

In this example, the user ID is 6.

```
ilo user logout --id 6
```

CLI Tool Commands for Resources

This section describes how to use the CLI Tool with various PCE resources.

View Workload Rules

You can view a specific workload's rules with the following command:

```
ilo workload rule_view --workload-id UUID
```

Where:

- UUID is the workload's UUID. See [About the Workload UUID \[480\]](#) for information.

In the example below, the workload's UUID is as follows:

```
2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```

Example View Workload Rules

```
ilo workload rule_view --workload-id 2ca0715a-b7e3-40e3-ade0-79f2c7adced0
+-----+-----+
| Attribute | Value |
+-----+-----+
| providing | []    |
+-----+-----+
Using
+-----+
+-----+
+-----+
| Ports And Protocols |
Rulesets
+-----+
+-----+
+-----+
| Href |
Name |
+-----+
+-----+
+-----+
| [[-1, -1, nil]] | [{"href"=>"/api/v2/orgs/28/sec_policy/8/rule_sets/1909", "name"=>"Default",
"secure_connect"=>false, "peers"=>[{"type"=>"ip_list",
"href"=>"/api/v2/orgs/28/sec_policy/8/ip_lists/188",
"name"=>"Any (0.0.0.0/0)",
"ip_ranges"=>[{"from_ip"=>"0.0.0.0/0"}]}]}] | /api/v2/orgs/28/sec_policy/8/services/1153 | All Services |
+-----+
+-----+
+-----+
200, OK
```

View Report of Workload Services or Processes

The following command lists all running services or processes on a workload:

```
ilo workload service_reports_latest --workload-id UUID
```

Where:

- *UUID* is the workload's UUID. See [About the Workload UUID \[480\]](#).

In the example, the workload's UUID is as follows:

```
2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```

Example Workload Service Report

```
ilo workload service_reports_latest --workload-id 2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```

```
+-----+-----+
| Attribute          | Value                               |
+-----+-----+
| uptime_seconds     | 1491                               |
| created_at         | 2015-10-20T15:13:00.681Z          |
+-----+-----+
```

Open Service Ports

```
+-----+-----+-----+-----+-----+
+-----+-----+
| Protocol | Address | Port | Process Name | User
| Package | Win Service Name |
+-----+-----+-----+-----+-----+
+-----+-----+
| udp      | 0.0.0.0 | 5355 | svchost.exe | NETWORK
SERVICE |         | Dnscache          |
...
| tcp      | 0.0.0.0 | 135  | svchost.exe | NETWORK
SERVICE |         | RpcSs           |
+-----+-----+-----+-----+-----+
+-----+-----+
200, OK
```

Use the list Option for Resources

Many resources take the `list` option. This section details some of its uses.

Default List of All Fields

The default `list` command displays all fields associated with the resource:

```
ilo resource list
```

List Only Specific Fields

With the `--field` option, specify the fields to display:

```
ilo resource list --field CSV_list_of_fieldnames
```

For example, to display a list of labels with only the href, key, and value fields, use the `--field` option with those fields as comma-separated arguments.

Example List with Selected Fields

```
ilo label list --fields href,key,value
+-----+-----+-----+
| Href           | Key   | Value           |
+-----+-----+-----+
| /api/v2/2/labels/1 | role | Web             |
| /api/v2/2/labels/2 | role | Database        |
| ...            |      |                  |
| /api/v2/2/labels/48 | loc  | Asia            |
+-----+-----+-----+
```

Nested Resource Fields and Wildcards

Some resources have hierarchical, nested fields. For example, the workload resource includes the following hierarchy for the agent field:

```
agent/config/log_traffic
```

- A field named `agent`
 - That has a field named `config`
 - That has a field named `log_traffic`

To list nested fields, separate the hierarchy of the field names with a slash to the depth of the desired field.

To see all nested fields of one of a resource's fields, use the asterisk (*) wildcard.

Examples

The following example displays all fields under the agent/config field.

Example of All Nested Fields with Wildcard (*)

```
ilo workload list --field agent/config/*
```

Log Traffic	Visibility Level	Mode
false	flow_summary	illuminated
false	flow_summary	idle

You can combine individual field names, nested field names, and the * wildcard.

Example: Combination of Individual fields, Nested fields, and Wildcard

```
ilo workload list --fields href,hostname,agent/config/*,agent/status/uid,agent/status/status
```

Href	Hostname	Log Traffic	Visibility Level	Mode	Uid	Status
/api/v2/1/workloads/527b8aca-97aa-43b9-82e1-29b17a947cdd	hrm-web.webscaleone.info	false	flow_summary	illuminated	0ffd2290-e26a-4ec6-b241-9e2205c0b730	active
/api/v2/1/workloads/4a8743a4-14ee-40d0-9ed2-990fe3f0ffb1	hrm-db.webscaleone.info	false	flow_summary	illuminated	145a3cc8-01a8-4a52-97b8-74264ad690e4	active

...

Linux: Save Fields for Reuse

On Linux, to easily reuse specific fields, create a display configuration file in YAML format and set the environment variable **ILO_DISPLAY_CONFIG** to point to that file. You no longer need to specify specific fields on the list command line.

Examples

Configure the workloads list command to display only the href, hostname, all agent configuration fields, and agent version:

Example Command to Save to List Configuration File

```
ilo workload list --fields href,hostname,agent/config/*,agent/status/agent_version
```

Add the field names to a display configuration file in the following YAML format:

Example YAML Layout of Display Configuration File

```
workload:
  fields:
    - href
    - hostname
  agent:
    config:
      fields:
        - '*'
    status:
      fields:
        - agent_version
```

Set the Linux environment variable **ILO_DISPLAY_CONFIG** to the path to the YAML file:

Example ILO_DISPLAY_CONFIG environment variable

```
$ export ILO_DISPLAY_CONFIG=~/.ilo_display/display_config.yaml
```

List of All Workloads

To view all details for all workloads, use the following command:

```
ilo workload list
```

About the Workload UUID

To view an individual workload, you need the workload's identifier, called the UUID, or Universal Unique Identifier.

The UUID is shown in the list of all workloads described in [List of All Workloads \[530\]](#). The UUID is the last word of the value of the workload's href field, as shown in bold in the following example:

```
/api/v2/orgs/28/workloads/2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```

View Individual Workload

To see the details about an individual workload, use the following command:

```
ilo workload read -workload-id UUID
```

Where:

- UUID is the workload's UUID. See [About the Workload UUID \[530\]](#) for information.

The details of an individual workload are grouped under major headings:

- Workload > Interfaces
- Workload > Labels
- Workload > Services
- Services > Open Service Ports
- Agent > Status

Example List of Individual Workload

```

ilo workload read --workload-id 2ca0715a-b7e3-40e3-ade0-79f2c7adced0
+-----+
+-----+
+-----+
| Attribute          |
Value
+-----+
+-----+
+-----+
| href              | /orgs/1/workloads/2ca0715a-
b7e3-40e3-ade0-79f2c7adced0
| deleted           |
false
...
Workload -> Interfaces
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| Name | Address          | Cidr Block | Default Gateway
Address | Link State | Network Id | Network Detection Mode
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| eth0 | 10.0.0.16        | 8          |
10.0.0.1 | up          | 1          |
single_private_brn
...
Workload -> Labels
+-----+
| Href          |
+-----+
| /orgs/1/labels/37 |
...
Workload -> Services
+-----+-----+-----+
| Attribute      | Value          |
+-----+-----+-----+
| uptime_seconds | 69016553       |
...
Services -> Open Service Ports
+-----+-----+-----+-----+
+-----+

```

```

| Protocol | Address | Port | Process Name | User | Package |
Win Service Name |
+-----+-----+-----+-----+-----+-----+
+-----+
| 17      | 0.0.0.0 | 123  | ntpd          | root |         |
|         |         |      |               |      |         |
...
Workload -> Agent
+-----+
+-----+
+-----+
| Attribute |
Value
          |
+-----+
+-----+
+-----+
| config    | {"log_traffic"=>true,
"visibility_level"=>"flow_summary", "mode"=>"enforced"} |
| href      | /orgs/1/
agents/16
          |
...
Agent -> Status
+-----+
+-----+
| Attribute |
Value
          |
+-----+
+-----+
| uid       | db482b06-41c6-4297-
a60c-396de13576ad |
| last_heartbeat_on |
2016-12-07T04:07:03.756Z
          |
...
200, OK

```

List Draft or Active Version of Rulesets

A security policy includes a ruleset, IP lists, label groups, services, and security settings. Before changes to these items take effect, the policy must be provisioned on the managed workload by setting its state to active with the CLI Tool or provisioning it with the PCE web console.

To view a ruleset and provisioning state, use the following command:

```
ilo rule_set list --pversion state
```

Where **state** is one of the following values:

- Draft: Any policy item that has not yet been provisioned.
- Active: All policy items that have been provisioned and are enabled on workloads.

The provisioning states are listed in the Enabled column:

- True: The policy is provisioned.
- Empty: The policy is a draft.

Example Draft Versions of Rulesets

```
ilo rule_set list --pversion draft
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| Href                                     | Created
By                                     | Name | Description | Enabled |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| /api/v2/orgs/28/sec_policy/draft/rule_sets/2387 |
{"href"=>"/api/v2/users/74"} | foo1      |          | true
| /api/v2/orgs/28/sec_policy/draft/rule_sets/1909
| {"href"=>"/api/v2/users/0"} | Default
|                               | true    ...
200, OK
```

The state of the policy is stored in the agent/status/status field. See [Nested Resource Fields and Wildcards \[478\]](#) for information.

Import and Export Security Policy

You can export and import security policy to and from the PCE using the CLI Tool. Importing and exporting security policy is particularly useful for moving policy from one PCE to another to avoid recreating policy from scratch on the target PCE. For example:

- You can test the policy on a staging PCE and then move it to your production PCE.
- You can move the policy from a proof-of-concept PCE deployment to your production PCE.

Export and Import Policy Objects

You can use the CLI Tool to export or import the following objects in the PCE:

- Labels: `labels`
- Label groups: `label_groups`
- Pairing profiles: `pairing_profiles`
- IP lists: `ip_lists`
- Services: `services`
- Rulesets and rules: `rule_sets`

About Exporting Rules

You can export rules for workloads, virtual services, or virtual servers.

Illumio recommends that you base your security policy rules on labels for flexibility. Do not tie the rules to specific individual workloads, virtual services, or virtual servers.

Virtual servers and virtual services are not exported.

The CLI Tool policy export does not include such references. A warning is displayed on export when you have rules tied to individual workloads, virtual services, or virtual servers. Attempts to import such rules fail, and the reason for the failure is displayed.

Example: Failed Attempt to Export Rules for Workload

```
WARNING: rule /orgs/1/sec_policy/active/rule_sets/3/
sec_rules/39 contains non-transferrable providers: workload /
orgs/1/workloads/a51ae67d-472a-44c3-984e-d518a8e95aee
Unable to proceed, please verify input
```

Workflow for Security Policy Export/Import

- Authenticate to the source PCE.
- Export the policy to a file. Syntax summary:

```
ilo sec_policy export --file someExportFilename
```

- Authenticate to the target PCE.
- Import the saved policy. Syntax summary:

```
ilo sec_policy import --file someImportFilename
```

Output Options, Format, and Contents

All exported policy is written to standard output. To write to a file, use the `--file` option.

The exported policy is in JSON format.

By default, all supported policy objects are exported. You can export a subset of policy by specifying one or more resource types with the `--resource` option (`labels`, `label_groups`, `pairing_profiles`, `ip_lists`, `services`, or `rule_sets`).

When a subset of policy items is exported (such as only labels), all referenced resources are also exported.

See also [About Exporting Rules \[535\]](#) for information.

Exported Rulesets

With the `--rule_set` option, you can export multiple rulesets.

By default, only the most recently provisioned, active policy is exported. To export the current draft policy or a previous policy, use the `--pversion` state option. See [List Draft or Active Version of Rulesets \[483\]](#) for information.

For a single ruleset, make sure the `--pversion` state you specify matches the provisioned state of the ruleset. In the following example, the state is draft:

```
ilo sec_policy export --pversion draft --rule_set /orgs/1/
sec_policy/draft/rule_sets/1
```

Effects of Policy Import

All imported policies are read from standard input unless you import from a file with the `--file` option.

You can import policy files multiple times. Each import affects only a single copy of a resource.

All imported policies are set in the draft provisioned state. After the import, you must explicitly provision the active state.

Non-transferrable policy rules (that is, rules tied to specific workloads, virtual servers, and bound services), the import aborts with a warning. See [About Exporting Rules \[535\]](#) for information.

Policy items already on the target PCE are updated by imported resources whose names match existing resources' names. Services do not have to have the same names. Services match if they have the same set of ports and protocols.

An import does not delete resources. For example, if you export policy from PCE-1 to PCE-2, delete a resource "R" from PCE 1, and then export and import again, resource "R" is still present on PCE 2. You must explicitly delete resource "R" from PCE2.

Upload Vulnerability Data

This section describes how to use the **ilo** commands to upload vulnerability data to the PCE for analysis in Illumination.

After uploading the data, you can use Vulnerability Maps in the PCE web console to gain insights into the exposure of vulnerabilities and attack paths across your applications running in data centers and clouds. See the "Vulnerability Maps" topic in the Visualization Guide for information.

Add the License for Vulnerability Data Upload

An Illumio Segmentation for Data Centers Vulnerability Maps license is required to upload vulnerability data into the Illumio PCE. For information about obtaining the license, contact Illumio Customer Support.

You are provided with a license file named **license.json**. After you have obtained your license key, store it in a secure location.



NOTE

Before adding the license, you must first authenticate to the PCE.

To add the license, you must be the organization owner or a be a user who has owner privileges.

Use the following command to inform the PCE of your valid license:

```
ilo license create --license-file "path_to_license_file/
license.json" --feature "feature_name" [debug [v | verbose]
trace]
```

Where:

What	Required?	Description
" <i>path_to_license_file/</i> license.json"	Yes	The quoted path to the <code>license.json</code> file from Illumio Example: " <code>~/secretDir/license.json</code> "
" <i>feature_name</i> "	Yes	The quoted string " <code>vulnerability_maps</code> ", which specifies the feature name the license enables
<code>debug</code>	No	Enable debugging
<code>v verbose</code>	No	For verbose logging
<code>trace</code>	No	Enable API trace

Vulnerability Data Upload Process

On upload, the CLI Tool associates a workload's IP addresses with corresponding vulnerabilities identified for that workload.

Using API to Download Vulnerability Data

Starting from the release of CLI 1.4.0, Qualys supports API downloads with some minor differences in options.

For the release CLI 1.4.1, it is suggested that users use an API key instead of a login session while using Qualys API download.

For the release CLI 1.4.2 for Tenable, the most reliable way to provide authentication is through API keys instead of username/password. If customers observe any authentication issues while using Tenable SC API upload, they are advised to use API keys.

There are 2 ENV variables to set up the Tenable SC API keys which are used for authentication:

TSC_ACCESS_KEY

TSC_SECRET_KEY

The API connects directly to the cloud instance of Tenable or Qualys and the vulnerability tool then scans new vulnerabilities and downloads them into the PCE.

Users can also set up cron jobs that run in the desired intervals and check the state of the vulnerability scanner.

Qualys and Tenable scanners work in a similar way, using the username and password and similar options.

Automating Vulnerability Imports from Tenable-SC

Users of Illumio vulnerability maps can automate the import of vulnerabilities from tenable-sc using a script.

Illumio CLI supports the API username and password as environment variables or a cmd line switch (such as `--api-password`).

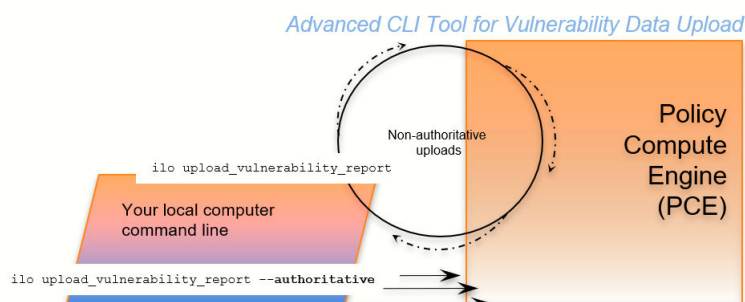
The ILO-CLI tool was updated to add a switch for `--api-user`.

Kinds of Vulnerability Data Uploads

There are two kinds of upload: non-authoritative and authoritative.

- **Non-authoritative:** This is the default. A non-authoritative upload:
 - Appends incoming data to any previously loaded records
 - Accumulates records for the same workloads without regard to duplicates.

You can repeat the non-authoritative upload as many times as you like until you are satisfied with the results.



- **Authoritative:** You indicate authoritative data with the `-authoritative` option. An authoritative upload:
 - Overwrites any previously uploaded records for workloads matched to the incoming records.
 - Eliminates duplicate records.
 - Adds new records not previously written by other uploads.You can repeat the authoritative upload as many times as you like until you are satisfied with the results.

After either kind of upload, you can examine the uploaded data with the CLI Tool or the PCE web console. See “Vulnerability Maps” in the Visualization Guide for information.

Supported Vulnerability Data Sources

The CLI Tool works with vulnerability data from the following sources.

- Nessus Professional™
- Qualys®
- Tenable Security Center
- Tenable.io
- Rapid7©



NOTE

Before uploading Rapid7 data to the PCE, export the data from Rapid7 to Qualys format with Qualys XML Export.

Vulnerability Data Formats

In the CLI 1.4.0, 1.4.1 and 1.4.2 releases, Illumio supports the following report formats:

- For `tenable-io`: API, CSV
- For `tenable-sc`: API, CSV
- For `nessus-pro`: XML
- For `qualys`: API, XML

Common Vulnerabilities and Exposures (CVE)

Vulnerabilities are defined by Common Vulnerabilities and Exposures (CVE), with identifiers and descriptive names from the U.S. Department of Homeland Security [National Cybersecurity Center](#).

Vulnerability Scores

Illumio computes a vulnerability score, which measures the vulnerability of your entire organization. The score is displayed by the `ilo vulnerability list` command for all vulnerabilities or individual vulnerabilities via the vulnerability identifier.

Vulnerability Identifier

An uploaded vulnerability has an identifier, as shown in the example below. The vulnerability identifier is tied to a specific CVE. You use this identifier with `--reference-id` option to examine specific uploaded vulnerabilities. See [Example – List Single Uploaded Vulnerability \[550\]](#) for information.

The following are examples of vulnerability identifiers.

- Nessus Professional: `nessus-65432`
- Qualys: `qualys-23456`
- Rapid7: `qualys-98765`. Because Rapid7 data is first exported from Rapid7 in Qualys format, it is given a Qualys identifier when uploaded to the PCE.

Vulnerabilities for Unmanaged Workloads

You can upload vulnerabilities for unmanaged workloads. However, unmanaged workloads do not have any vulnerability score or associated CVE. This information becomes available if the unmanaged workload is later changed to managed.

Prerequisites for Vulnerability Data Upload

Before uploading vulnerability data, ensure you are ready with the following requirements.

- An Illumio Vulnerability Maps license is required to upload vulnerability data to the PCE. See [Add the License for Vulnerability Data Upload \[537\]](#) for information.
- XML-formatted vulnerability data files from one of the supported sources.
- Authenticated CLI-tool access to the target PCE.
- Authenticated access and necessary permissions in the PCE web console for working with vulnerability maps.

Vulnerability Data Upload CLI Tool Syntax

The key argument and options for uploading vulnerability data are as follows. For readability, this syntax is broken across several lines.

```
ilo upload_vulnerability_report
--input-file path_to_datafile.xml [path_to_datafile.xml]...
--source-scanner [nessus-pro|qualys|tenable-sc|tenable-io]
--format xml
[--authoritative]
[ --api-user ApiServerUserName --api-server
SourceApiServer:port ]
```

Where:

What	R e q u i r e d	Description
<code>--enable-proxy</code>	No	Use this to enable the proxy between tenable and CLI. Use this command to enable the proxy: <pre>ilo upload_vulnerability_report --source-scanner tenable-sc --format api --severities=3 --enable-proxy -v --debug</pre> Use this command if you do not want to enable the proxy: <pre>ilo upload_vulnerability_report --source-scanner tenable-sc --format api --severities=3 -v --debug</pre>
 NOTE This is available in CLI Tool 1.4.4.		
<code>--input-file path_to_datafile.xml [path_to_data-file.xml]...</code>	Yes	Location of one or more data files to upload. The path to the data file can be either an absolute path or a relative path.

What	R e q u i r e d	Description
		If more than one data file is listed (bulk upload), separate the file names with space characters.
<code>--debug</code>	No	Enable debugging
<code>--authoritative</code>	No	For uploading authoritative vulnerability data. The default command is without the <code>--authoritative</code> option. See Kinds of Vulnerability Data Uploads [539] for information.
<code>--workload-cache FILE</code>	No	DEBUGGING ONLY: Workload Cache file - use this if available
<code>--source-scanner [nessus-pro qualys tenable-sc]</code>	Yes	Indicates the source of the scan. Note for rapid data: - Vulnerability data from Rapid must have been exported from Rapid in Qualys XML format. - To load the Rapid data, use the 'qualys' argument
<code>--format</code>	Yes	Report format. Allowed values are:
<code>REPORT_FORMAT</code>		xml <code>--source-scanner nessus-pro</code> <code>--source-scanner qualys</code> csv <code>--source-scanner tenable-sc</code> <code>--source-scanner tenable-io</code> api <code>--source-scanner tenable-sc</code> <code>--source-scanner qualys</code> <code>--source-scanner nessus-pro</code> See also <code>--api-server</code> and <code>--api-user</code> .
<code>--api-server SourceApi-Server:port</code>	Yes	API server FQDN. Allowed formats are HOST or HOST:PORT
<code>SERVER_FQDN</code>	or	

What	R e q u i r e d	Description
	T e n a b l e w i t h - - f o r m a t a p i	
--api-user ApiServerU- serName USERNAME	Y e s f o r s o u r c e A P I s e r v e r a u t h e n t	The user name for authenticating to the SourceApiServer. You are always prompted to enter your password.

What	R e q u i r e d	Description
	i c a t i o n	
--api-page-size PAGE_SIZE	Y e s f o r Q u a l i t y s a n d T e n a b l e	Appropriate page size if API supports pagination. The default page is 1000.
--skip-cert-verification	Y e s f o r Q u a l i t y s a n d T e n a b	Disable certificate verification for API.

What	Required	Description
<code>--on-premise</code>	Yes only for Tenableable.io	Tenable IO deployment is on-premise.
<code>--mitigated</code>	Yes only for Tenableable scans	Tenable SC input is exported from the mitigated vulnerabilities analysis view.
<code>--scanned-after</code> <code>SCANNED_AFTER</code>	Yes for Q	Qualys users can select scan data to process after a specific date, in ISO 8601 format. When the optional scanned-after option is not provided, the system will pull all the historical vulnerability records from your Qualys account. If your account has historical records, it may take a very long time for the first time. With the scanned-after option, vulnerability

What	Required	Description
	always	data scanned after a specific date will be extracted and uploaded. Including a particular scanned-after time is recommended if you use Qualys API upload option for the first time.
--severities SEVERITIES	No	Qualys API users can select vulnerabilities with defined severity levels to include in their reports. Users can filter based on severity and avoid severity levels 1 and 2, which are often very informational and noisy. Example: --only-include-severity=3,4,5 For Windows, be sure to include quotes around the severity levels: Example: --only-include-severity="3,4,5" NOTE: This option was added in Release 1.4.1
-v, --verbose	No	Verbose logging mode
--trace	No	Enable API trace mode.

Using the ILO Command with Windows Systems

Windows systems take up to four options with the ILO command for the vulnerability data upload. Users who choose to use more optional parameters must set **api-server**, **username**, and **password** as the environmental variables to use other options in the command.

Work with Vulnerability Maps in Illumination

See "Vulnerability Maps" in Visualization Guide for information.

Vulnerability Data Examples

Example - Upload Non-Authoritative Vulnerability Data

In this example, the **--source-scanner nessus-pro** option indicates that the data comes from Nessus Professional. On Windows, provide the absolute

path to the data file. This Windows example is broken across several lines with the PowerShell line continuation character (`).

```
C:\Users\donald.knuth> ilo upload_vulnerability_report `
--input-file
C:\Users\donald.knuth\Desktop\vuln_reports\nessus3.xml `
--source-scanner nessus-pro --format xml

Elapsed Time [0.05 (total : 0.05)] - Data parsing is done.
Elapsed Time [1.08 (total : 1.13)] - Got workloads. Workload
count: 5.
Elapsed Time [0.0 (total : 1.13)] - Built workload interface
mapping. Total interfaces : 11.
Elapsed Time [4.57 (total : 5.7)] - Imported Vulnerabilities..
Elapsed Time [0.0 (total : 5.7)] - Detected Vulnerabilities
are associated with vulnerability and workload data..
Elapsed Time [0.83 (total : 6.53)] - Report Imported.

Summary:
Processed the report with the following details :
Report meta data =>
Name           : Generic
Report Type    : nessus
Authoritative  : false
Scanned IPs    : ["10.1.0.74", "10.1.0.223", "10.1.0.232",
"10.1.0.221", "10.1.0.11", "10.1.0.82", "10.1.0.43",
"10.1.0.91", "10.1.0.8", "10.1.1.250"]

Stats :
    Number of vulnerabilities           => 19
    Number of detected vulnerabilities => 31

Done.
```

Example – Upload of Rapid7 Vulnerability Data

The syntax for uploading vulnerability data from Rapid7 is identical to the syntax for uploading vulnerability data from Qualys. On Windows, you use the **--format qualys** option and the absolute path to the data file. This Windows example is broken across several lines with the PowerShell line continuation character (`).

Rapid7 data exported in Qualys format.

Before uploading to the PCE, Rapid7 vulnerability data must have been exported in Qualys format from Rapid7 with Qualys XML Export.

```
C:\Users\edward.teller> ilo upload_vulnerability_report `
--input-file
C:\Users\edward.teller\Desktop\vuln_reports\rapid7.xml `
--source-scanner qualys --format xml
...
Done.
```

Example - Upload Authoritative Vulnerability Data

In this example, the prompt shows this is an authoritative upload.

To proceed, you must enter the word YES in all capital letters.

```
C:\Users\jrobert.oppenheimer> ilo upload_vulnerability_report
--input-file dataDir/authoritativedata.xml --authoritative --
source-scanner qualys --format xml
```

```
Using /home/centos/.rvm/gems/ruby-2.4.1
Authoritative scan overwrites the previous entries for all the
ips within this scan. There is no ROLLBACK
Are you sure this is an authoritative scan? (YES | NO)
YES
Elapsed Time [11.86 (total : 11.86)] - Data parsing is done.
Elapsed Time [0.27 (total : 12.13)] - Got workloads. Workload
count: 3.
Elapsed Time [0.0 (total : 12.13)] - Built workload interface
mapping. Total interfaces : 6.
Elapsed Time [3.02 (total : 15.15)] - Imported
Vulnerabilities..
Elapsed Time [0.0 (total : 15.15)] - Detected Vulnerabilities
are associated with vulnerability and workload data..
Elapsed Time [0.84 (total : 16.0)] - Report Imported.
Summary:
Processed the report with the following stats -
    Number of vulnerabilities          => 14
    Number of detected vulnerabilities => 48
Done.
```

Example - List Single Uploaded Vulnerability

This example uses a single Qualys vulnerability identifier to show the associated vulnerability. The value passed to the `--reference-id` option is shown as qualys-38173. See [Vulnerability Identifier \[541\]](#) for information.

```
$ ilo vulnerability read --xorg-id=1 --reference-id=qualys-38173
...

| Attribute | Value |
+-----+
+-----+
--+
| href | /orgs/1/vulnerabilities/qualys-38173 |
| name | SSL Certificate - Signature Verification Failed
Vulnerability
| score | 39 |
| cve_ids | [] |
| created_at | 2018-11-05T18:16:56.846Z |
...
```

Example – List All Uploaded Vulnerabilities

This example highlights the vulnerability identifier, the CVE identifiers, and the description of the CVE. See [Common Vulnerabilities and Exposures \(CVE\) \[540\]](#) and [Vulnerability Identifier \[541\]](#) for information. The layout of the output is the same for all supported vulnerability data sources.

Nessus Professional

```
C:\Users\werner.heisenberg> ilo vulnerability list --xorg-id=1
...
| Href | Name | Score | Description | Cve Ids | Created At |
Updated At | Created By | Updated By |
+-----+
+-----+
| /orgs/1/vulnerabilities/nessus-18405 | Microsoft Windows
Remote Desktop Protocol Server Man-in-the-Middle Weakness
| 51 | | ["CVE-2005-1794"]
| 2018-11-07T03:15:39.410Z | 2018-11-07T03:15:39.410Z
| {"href"=>"/users/1"} | {"href"=>"/users/1"} |
...
```

Qualys

```
C:\Users\isaac.newton> ilo vulnerability list --xorg-id=1
...
| Href | Name | Score | Description | Cve Ids | Created At |
Updated At | Created By | Updated By |
-----+-----
+-----+-----+
| /orgs/1/vulnerabilities/qualys-38657 |
Birthday attacks against TLS ciphers with 64bit
block size vulnerability (Sweet32) | 69 | | ["CVE-2016-2183"]
| 2018-07-27T18:16:57.166Z | 2018-08-08T22:30:32.421Z
| {"href"=>"/users/1"} | {"href"=>"/users/16"} |
...
```

Rapid7

Because Rapid7 vulnerability data must be in Qualys format before upload, the output is the same as for Qualys data, including the vulnerability identifier (qualys-38657 in the example above) and CVE. See [Common Vulnerabilities and Exposures \(CVE\) \[540\]](#) and [Vulnerability Identifier \[541\]](#) for information.

Example - View Vulnerability Report

The Report Type column identifies the source of the scan; in this example, Qualys.

```
C:\Users\gracemurphy.hopper> ilo vulnerability_report list --
xorg-id=1
...
| Href | Report Type | Name | Created At | Updated At | Num
Vulnerabilities | Created By | Updated By |
-----+-----
+-----+-----+
+-----+-----+
| /orgs/1/vulnerability_reports/scan_1502310096_09344 |
qualys | NewAuthoritativeScan | 2018-08-08T22:30:34.877Z |
2018-08-08T22:30:34.877Z | 62 | {"href"=>"/users/16"} |
{"href"=>"/users/16"} |
...
```

Example - Upload a Qualys Report Using API

```
upload_vulnerability_report --source-scanner qualys --format
api
--api-server qualysguard.qg3.apps.qualys.com --api-user um3sg
--scanned-after 2021-09-20
```

CLI Tool Tutorials

This section provides several hands-on exercises that demonstrate step-by-step how to perform common tasks using the CLI Tool.

How to Import Traffic Flow Summaries

Static Illumination provides “moment-in-time” visibility of inter-workload traffic. This visibility is useful to model policies, to look for specious traffic flows, and to ensure that metadata for labels is accurate.

Goal

Load workload and traffic data needed for analysis with static Illumination.

Setup

This tutorial relies on the following data to import.

- 1,000 workloads defined in the file **bulkworkloads-1000.csv**, which has the following columns:

```
hostname,ips,os_type
10.14.59.8.netstat,10.14.59.8,linux
10.4.78.178.netstat,10.4.78.178,linux
10.37.134.179.netstat,10.37.134.179,linux
...
```

- 1,000,000 traffic flows defined in the CSV file **traffic.clean-1m.csv**, which has the following columns:

```
src_ip,dst_ip,dst_port,proto
10.40.113.86,10.14.59.8,10050,6
10.14.59.8,10.8.251.138,8080,6
10.40.113.124,10.14.59.8,22,6
...
```

Steps

The workflow is authenticated to the PCE and run two **ilo bulk_upload_csv** commands.

1. Authenticate to the PCE via API key or explicit login.
2. Load the workload data:

```
ilo workload bulk_upload_csv --file bulkworkloads-1000.csv
```

3. Load the traffic flow data:

```
ilo traffic bulk_upload_csv --file traffic.clean-1m.csv
```

Results

The data from the CSV files are uploaded.

How to Create Kerberos-Authenticated Workloads

This tutorial describes how to create workloads that use Kerberos for authentication. The tutorial makes the following assumptions:

- This tutorial assumes that you already have your Kerberos implementation in place.
- As Kerberos requires, the Kerberos realm name is shown in all capital letters as **MYREALM**.
- VEN environment variables must be set *before* VEN installation. Environment variables for Linux are detailed in the VEN Installation and Upgrade Guide.

Goals

- Create two workloads on Linux that are authenticated by Kerberos.
- Set the workloads' modes to idle and illuminated.

Setup

The key data for using the **ilo** command to create these workloads are the name of the Kerberos realm and the Service Principle Name (SPN).

Steps

The workflow is authenticate, run two **workload create** commands that set the workloads' modes, set the VEN environment variables, install the VEN, and run two Kerberos **kinit** commands to get Kerberos tickets for the workloads.

- 1.** Authenticate to the PCE via API key or explicit login.
- 2.** Create Kerberos-authenticated **myWorkload1** and set its **mode** to **idle**:

```
ilo workload create --hostname myPCE.BigCo.com
--name myWorkload1 --service-principal-name host/
myKerberosTicketGrantingServer@MYREALM --agent/config/mode
idle
```


For information about how the mode is a nested field, see [Nested Resource Fields and Wildcards \[478\]](#).

3. Create Kerberos-authenticated `myWorkload2` and set its mode to `illuminated`:

```
ilo workload create --hostname myPCE.BigCo.com
--name myWorkload2 --service-principal-name host/
myKerberosTicketGrantingServer@MYREALM --agent/config/mode
illuminated
```

4. Before installation, set VEN environment variables:

```
# Activate on installation
VEN_INSTALL_ACTION=activate
# FQDN and port PCE to pair with
VEN_MANAGEMENT_SERVER=myPCE.BigCo.com:8443
# Kerberos Service Principal Name
VEN_KERBEROS_MANAGEMENT_SERVER_SPN=host/
myKerberosTicketGrantingServer
# Path to Kerberos shared object library
VEN_KERBEROS_LIBRARY_PATH=/usr/lib/libgssapi_krb5.so
```

5. Install the Linux VEN:

```
rpm -ivh illumio-ven*.rpm
```

6. Run `kinit` to get a Kerberos ticket for `myWorkload1`:

```
kinit -k -t /etc/krb5.keytab host/
myWorkload1.BigCo.com@MYREALM
```

7. Run `kinit` to get a Kerberos ticket for `myWorkload2`:

```
kinit -k -t /etc/krb5.keytab host/
myWorkload2.BigCo.com@MYREALM
```

Results

The Kerberos-authenticated workloads are created, set in the desired modes, and given a Kerberos ticket.

How to Work with Large Datasets

The `--async` option is for working with large data sets without waiting for the results. The option works like “batch job.”

The option can be used with any resource. The workflow is as follows:

1. You issue the desired `ilo` command with the `--async` option, which displays a job ID.

2. You take note of the job ID.
3. Your session is freed up while the job runs.
4. The job creates a data file, which you view with `datafile --read --job-id jobID`.

Goal

Get a report of a large workload data set.

Steps

1. Issue the `--async` request for a workload list. Take note of job ID, which is the final word of the href displayed on the Location line.

```
[kurt.goedel~]$ ilo workload list --async
Using /home/kurt.goedel/.rvm/gems/ruby-2.2.1
Location: /orgs/1/jobs/fe8a1c2b-1674-4b83-8967-eb56c4ffa1e3
202, Accepted
```

2. Check to see if the job completed. Use the job ID from the **Location** output in previous command:

```
[sigmund.freud~]$ ilo job read --job-id
fe8a1c2b-1674-4b83-8967-eb56c4ffa1e
Using /home/sigmund.freud/.rvm/gems/ruby-2.2.1
```

3. Download the resulting data file, specifying the job ID with `-uuid jobID`:

```
[bill.gates ~]$ ilo datafile read --uuid 1e1c1540-8a01-0136-ec14-02f4d6c1190c
Using /home/ bill.gates /.rvm/gems/ruby-2.2.1
+-----+
+-----+-----+
... Many lines not shown
+-----+
+-----+-----+
| Href
| Deleted | Name | Description | Hostname
| Service Principal Name | Public
Ip | Distinguished Name |
External Data Set | External Data Reference
| Interfaces | Ignored Interface Names
| Service Provider | Data Center
| Data Center Zone | Os Id
| Os Detail | Online | Labels | Services | Agent
| Created At | Created By
| Updated At | Updated By
+-----+
+-----+-----+
... More lines not shown
+-----+
| /orgs/1/workloads/50ce441e-75ac-4be8-9201-96169545019c
| false | | 10.14.59.8.netstat
...
... Many lines not shown
...
```

How to Upload Vulnerability Data

This example tutorial shows how to upload vulnerability data to the PCE. For more information, see [Upload Vulnerability Data \[490\]](#). The source of the vulnerability data in this example comes from Qualys®.

Goal

Upload authoritative vulnerability data for analysis in Illumination.

Steps

1. Do a non-authoritative upload of vulnerability data for examination:

```
ilo upload_vulnerability_report --input-file C:\Users\albert-einstein0.xml --source-scanner qualys --format xml
```

2. Examine a single uploaded vulnerability record identified by its vulnerability identifier, **qualys-38173**. See [Vulnerability Identifier \[494\]](#) for information.

```
ilo vulnerability read --xorg-id=1 --reference-id=qualys-38173
```

3. Do another non-authoritative upload of vulnerability data.

```
ilo upload_vulnerability_report --input-file C:\Users\albert-einstein99.xml --source-scanner qualys --format xml
```

4. Do an authoritative upload of vulnerability data, overwriting any previously uploaded records and adding any new vulnerability records.

```
ilo upload_vulnerability_report --input-file C:\Users\albert.einstein_FINAL.xml --authoritative --source-scanner qualys --format xml
```

Results

The authoritative vulnerability data has been uploaded and is ready for use in Illumination.

Error Messages

Learn about common error messages and ways you can resolve these errors on your own by performing specific recommended actions.

Containers Error Messages

Here are some common error messages and ways you can resolve them.

Containers

Table 6. Common Containers Error Messages

Action	Error	Description	Recommended Action
Kube-link	pkg.pcelink PCE request failed <pre>{"url": "https://<PCE>:443/api/v2/orgs/<OrgID>/container_clusters/<UUID>/put_from_cluster", "error":</pre> Put \"https://<PCE>:443/api/v2/orgs/<OrgID>/container_clusters/<UUID>/put_from_cluster\": dial tcp: lookup <PCE FQDN> on <IP>:53: server misbehaving\"}	DNS Error	Check the DNS records for the PCE FQDN

NEN, VEN Error Messages

Here are some recommended actions you can take to help you troubleshoot common NEN and VEN errors.

NEN**Table 7. Common NEN Error Messages**

Action	Error	Description	Recommended Action
Add NEN	ERROR -- : The property 'service_discovery_certificate' value='/var/lib/illumio-nen/cert/server.crt' file doesn't exist	NEN requires a separate certificate from the PCE	Create and add a certificate specific to the NEN
Add NEN	ERROR -- : The property 'service_discovery_private_key' value='/var/lib/illumio-nen/cert/server.key' is not a valid private key	NEN certificates require a valid certificate chain as part of the .crt file	Check the syntax of the certificate chain, including the order of the certificates

VEN

Table 8. Common VEN Error Messages

Action	Error	Description	Recommended Action
Operation after upgrade	agentmgr.log: ERROR:: [ApiHelper] Failed to PUT https://<PCE>;8443/api/v26/orgs/1/agents/<AgentID>/heartbeat (internal error 56)platform.log: ERROR:: SSL peer certificate or SSH remote key was not OK , SSL certificate problem: unable to get local issuer certificate	VEN cannot communicate with the PCE after an upgrade	Edit the PCE certificates to be up to date and include all intermediate certificate components. KB article
Install with Pairing Profile	{"error": "profile_id is invalid"}	The profile_id associated with the pairing profile is missing	Create a new pairing profile. KB article
Policy Sync	PCE Reports policy sync errors on VEN, you see two copies of Illumio VEN agent installation path	Multiple VEN binaries are installed in different paths after the VEN upgrade	Remove VEN binaries and rpm link to both directories. Install current VEN binaries in default path. KB article
SUSE Linux 15	failed to create symbolic link '/etc/init.d/rc2.d/S50illumio-ven': No such file or directory	Missing dependency for the VEN to run	insserv-compat is a prerequisite to have deployed before install/upgrade of the VEN KB article

PCE Error Messages

Here are some common error messages and ways you can resolve them.

PCE Error Messages and Recommended Actions

Table 9. Common PCE Error Messages

Action	Error	Description	Recommended Action
Daily operation	ERROR -- : Certificate validation issue with web_service_certificate : host: <PCE>	Expired certificate on the PCE	Run this command: illumio-pce-env setup --test 5 --list
Install compatibility matrix	There was an error (unsupported_ven_release): VEN release <X> is not supported by PCE <Y>	Incorrect/outdated compatibility matrix installed	Get the latest compatibility matrix from the Support website .
Error while generating an inventory report	No such file or directory - sysctl		Get the path where sysctl is located i.e. /sbin or /usr/sbin and add it to the path. See KB article
Log In	DB connection error detected; could not fork new process for connection: Resource temporarily unavailable	File limit settings	Make sure to set the correct file limit and configuration settings of the PCE. See KB article
Log In	500 Internal Server Error / PCE UI Missing Error	This means the UI is not installed or the directory containing /opt/illumio-pce is incorrect.	Check that the UI is installed with: yum list installed grep -i illumio-pce-ui Make sure that you set the correct owner recursively for the folder: /opt/illumio-pce with: chown -R root:ilo-pce /opt/illumio-pce See KB article
Pairing Profile Access	500 Internal Server Error	500 Internal Server Error	VEN Bundle missing from PCE with a Pairing Profile using that VEN version. See KB article
Enable feature for API to SaaS PCE	API call returned error code 403. Errors: forbidden_error: Access denied.		Check the permissions of the API user. Require elevated access to perform API actions.

Action	Error	Description	Recommended Action
			If this is a SaaS PCE, the feature flag may need to be enabled.

Legal Notice

Copyright © 2026 Illumio 920 De Guigne Drive, Sunnyvale, CA 94085. All rights reserved.

The content in this documentation is provided for informational purposes only and is provided "as is," without warranty of any kind, expressed or implied, of Illumio. The content in this documentation is subject to change without notice.

Resources

- [Legal information](#)
- [Trademarks statements](#)
- [Patent statements](#)
- [License statements](#)

Contact Information

- [Contact Illumio](#)
- [Contact Illumio Legal](#)
- [Contact Illumio Documentation](#)