Illumio Core for Kubernetes 5.2

November 6, 2024

Table of Contents

About Illumio Core for Kubernetes 5.2 3
Product Version 4
What's New in Release 5.2.1 5
Updates for Core for Kubernetes 5.2.1 6
    Kubelink 6
        Resolved Issues 6
    C-VEN 6
        Resolved Issues 6
What's New in Release 5.2.0 7
    Resource Allocation Guidelines 8
        Small-scale resource allocation 9
        Medium-scale resource allocation 10
        Large-scale resource allocation 10
    "Wait For Policy" Feature 10
        Behavior 11
        Configuration 11
Updates for Core for Kubernetes 5.2.0 13
    Kubelink 13
        Resolved Issues 13
    C-VEN 13
        Resolved Issues 13

About Illumio Core for Kubernetes 5.2

These release notes describe the resolved issues, known issues, and related information for the 5.2.*x* releases of Illumio Core for Kubernetes, formerly known as Illumio Containerized VEN, or C-VEN. Illumio Core for Kubernetes also includes the related required component, Kubelink. Because of this heritage, many references to this product as "C-VEN" are still used throughout the documentation.

Document Last Revised: November 2024

Product Version

Compatible PCE Versions: 23.5.0+A1 and later releases

Current Illumio Core for Kubernetes Version: 5.2.1, which includes:

- C-VEN version: 23.4.0
- Kubelink version: 5.2.1
- Helm Chart version: 5.2.1

What's New in Release 5.2.1

- **Helm Chart option to Disable NodePort Forwarding**
  A new option was added to Helm Chart for C-VEN that disables NodePort forwarding on host workloads. After setting `enforceNodePortTraffic: never` in the Helm values file, C-VEN behaves like before in its 22.5 version-- that is, the forward chain on Node is open, and custom iptables rules must be used to enforce traffic in this chain.

Updates for Core for Kubernetes 5.2.1

Kubelink

Resolved Issues

- **Kubelink can't start on OpenShift because of fsGroup 1001** (E-120425)
  When using Helm Chart 5.2.0 on OpenShift, Kubelink would not start because of fsGroup 1001.

C-VEN

Resolved Issues

- **NodePort access is working when it should be blocked** (E-120655)
  NodePort traffic was being always allowed, with or without a rule allowing the traffic from an external resource to the NodePort service. This issue was fixed by adding missing legacy iptables command line utilities to the UBI9-based C-VEN.
- **C-VEN crashed when restarted, with kernel logs showing segfault errors** (E-119682)
  A 23.4.0 C-VEN sometimes crashed when restarted, with the kernel logs showing a segfault error.
- **C-VEN is unable to send flows if there is a lot of data** (E-119110)
  C-VEN stopped sending flow records because it was trying to send all flow records all at once, without regard to the gRPC message size limit setting. After a few retries the AgentMgr process would crash.
- **Move C-VEN base image to a minimal image** (E-118492)
  C-VEN now uses UBI9 micro image as base, current latest version 9.4-15.

What's New in Release 5.2.0

- **"Wait for Policy" Feature**

  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 the pod. This feature replaces the local policy convergence controller, the Illumio readiness gate. A readiness gate required adding the `readinessGates.conditionType` into the spec YAML file of the Kubernetes Workload. Instead, Wait For Policy uses an automatically injected init container, with no change of the user application needed. When enabled, Wait For Policy synchronizes the benefit of Kubernetes automatic container creation with the protection of proper policy convergence into the new container.

  For more information, see "Wait For Policy" Feature [10].
- **CLAS Flat Network Support**

  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"
  ```
- **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: (Note that the backslash (\) character is included to indicate the continuation of a long command line that will be truncated by the right margin of this document in PDF form.)

  ```
  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.
- **Base OS Upgraded to UBI9**

  The base OS has been upgraded to Red Hat Universal Base Image 9 (micro UBI9 for Kubelink, mini UBI9 for C-VEN).

IMPORTANT

Important Notice: With the base image upgrade for both Kubelink and C-VEN, you must adjust resource allocations according to the guidance described below in the "Resource Allocation Guidelines [8]" section. You must ensure that resources are updated prior to the upgrade to achieve optimal performance, and to avoid any potential degradation in product performance.

- **Enhanced Pod Stability for Kubelink and C-VEN**

  To address the challenge of pod eviction during Kubernetes cluster issues or space shortages, Kubelink was previously the first pod to be evicted, which led to failures in policy enforcement. Recognizing the critical need for stability, Helm Chart version 5.2.0 introduces default priority classes for both Kubelink and C-VEN. Kubelink is now assigned the priority class of `system-cluster-critical`, while C-VENs receive `system-node-critical`. This implementation significantly enhances the resilience of your deployments, ensuring that key components remain operational even under resource constraints.
- **Changes to Supported Orchestration Platforms and Components in 5.2.0**

  The 5.2.0 release contains several changes to supported platforms and components. For full details, see Kubernetes Operator OS Support and Dependencies on the Illumio Support portal (log in required).

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

"Wait For Policy" Feature

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 the spec of the Kubernetes Workload. Instead, Wait For Policy uses an automatically injected init container, which requires no change to the user application.

Behavior

When Wait For Policy is enabled, Kubelink creates a new `MutatingWebhookConfiguration`. This webhook injects an Illumio init container into every new pod. Now a new pod lifecycle consists of the following sequence of actions:

1. Kubernetes creates a pod.
2. The pod creation request is intercepted by a mutating webhook.
3. Kubernetes requests MutatingAdmissionWebhook Controller running in Kubelink.
4. Controller returns with a new pod patched with an Illumio init container.
5. Init container starts in the pod, and periodically checks the policy status of the pod using the Kubelink status server.
6. At the same time, Kubelink is preparing a policy for the new pod, and is sending the policy to the pod's C-VEN.
7. The C-VEN applies policy to the pod, and sends an acknowledgment to Kubelink.
8. Kubelink reports that the policy is now applied to the init container.
9. The Init container exits, and allows the original container to start.
10. If a policy is not applied within the configured time (see Configuration [11] section for Helm Chart `waitForPolicy.timeout` parameter), the init container exits anyway, and allows the original container to start.

The Illumio init container must be available from a public repository. Deploying `imagePull-Secret` for the private repository to every namespace in the cluster is not feasible.

Configuration

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. The following is the default Helm Chart configuration for Wait For Policy:

```
## Wait for Policy - Illumio delays the start of Pods until policy is applied
waitForPolicy:
  ## @param waitForPolicy.enabled Enable Wait for Policy feature
  enabled: false
  ## @param waitForPolicy.ignoredNamespaces List of namespaces where Illumio
  ## doesn't delay start of Pods. kube-system and
  ## illumio-system name are ignored by Kubelink for this feature by default,
  ## even if not specified in this list.
  ignoredNamespaces:
    - kube-system
    - illumio-system
  ## @param waitForPolicy.timeout How long will pods wait for policy, in seconds
  timeout: 130
```

Pods starting in namespaces listed in `ignoredNamespaces` start immediately, without an Illumio init container injected into them. The namespaces `kube-system` and `illumio-system` are always ignored by the MutatingAdmissionWebhook Controller running in Kubelink, even if those are not specified in the configuration. The default value of `ignoredNamespaces` contains `kube-system` and `illumio-system` for reference, and can be extended with custom namespaces.

The `timeout` value is a total allowed run time of the init container. After this time elapses, the init container exits even if policy is not applied, and allows the original container to start.

Updates for Core for Kubernetes 5.2.0

Kubelink

Resolved Issues

- **Helm: pull secret to quay gets created even if no credentials are set** (E-119659)
  Helm chart now creates Illumio pull secret only if credentials are specified and also externally passed secret names are included.
- **Kubelink: error concurrent map read and map write** (E-119626)
  Kubelink was restarted because previous container exited with the message "`fatal error concurrent map read and map write`."
- **Kubelink: Update base image to address vulnerabilities** (E-119429)
  The Unified Base Image was upgraded to address CVE-2023-45288.
- **Kubelink needs to have higher priority assigned to avoid going to evicted state** (E-113920)
  If the Kubernetes cluster encounters problems or runs out of space, Kubelink was the first pod to be put into the evicted state, which caused policy enforcement to fail. To prevent permanent eviction, in Helm chart version 5.2.0 the Kubelink Deployment and C-VEN DaemonSets are assigned priority classes by default -- `system-cluster-critical` for Kubelink and `system-node-critical` for C-VENs.

C-VEN

Resolved Issues

- **CVEN: Update base image to address vulnerabilities** (E-119428)
  The 23.4 C-VEN Unified Base Image was upgraded to the latest UBI9 to address vulnerabilities described in CVE-2014-3566, CVE-2014-3566, CVE-2014-3566, CVE-2022-3358, and CVE-2023-27533.
- **Cannot deploy C-VEN to GKE when using default OS** (E-116506)
  For GKE clusters, when using the default cluster OS (Container-Optimized OS from Google), the node filesystems are read-only. This prevented C-VEN from mounting `/opt/illumio_ven_data` and writing into it for persistent storage.

  To resolve this issue, a new variable `` `cven.hostBasePath `` was added to the 5.2.0 Helm Chart to specify where the C-VEN DaemonSet mounts its data directory. The default value is `/opt`. Use this variable to specify where the C-VEN DaemonSet mounts its data directory. If using a Container-Optimized OS, you can set the directory to `/var`.
- **[CVEN]: Failed to load policy** (E-115231)
  The log message "`Error: Failed to load policy`" was appearing during scenarios that were obvious or expected. The log level for this message has been changed from Error to Info.
- **Re-adding node does not re-pair it** (E-98120)
  When deleting and then re-adding the same node, the node would not reappear, and its policy disappeared.