OpenShift Container Platform 4.11 Sandboxed Containers Support for OpenShift

OpenShift sandboxed containers guide
Abstract

OpenShift sandboxed containers support for OpenShift Container Platform provides users with built-in support for running Kata Containers as an additional optional runtime.
Table of Contents

CHAPTER 1. OPENSSHIFT SANDBOXED CONTAINERS 1.3 RELEASE NOTES .......................... 4
  1.1. ABOUT THIS RELEASE .................................................................................. 4
  1.2. NEW FEATURES AND ENHANCEMENTS ...................................................... 4
    1.2.1. Container ID in metrics list .................................................................... 4
    1.2.2. OpenShift sandboxed containers availability on AWS bare metal .......... 4
    1.2.3. Support for OpenShift sandboxed containers on single-node OpenShift .... 4
  1.3. BUG FIXES .................................................................................................... 4
  1.4. KNOWN ISSUES ............................................................................................ 5
  1.5. ASYNCHRONOUS ERRATA UPDATES .......................................................... 6
    1.5.1. RHSA-2022:6072 - OpenShift sandboxed containers 1.3.0 image release, bug fix, and enhancement advisory. ................................................................. 7
    1.5.2. RHSA-2022:7058 - OpenShift sandboxed containers 1.3.1 security fix and bug fix advisory.................. 7

CHAPTER 2. UNDERSTANDING OPENSHELL SANDBOXED CONTAINERS ....................... 8
  2.1. OPENSHELL SANDBOXED CONTAINERS SUPPORTED PLATFORMS ............. 9
  2.2. OPENSHELL SANDBOXED CONTAINERS COMMON TERMS ......................... 9
  2.3. OPENSHELL SANDBOXED CONTAINERS WORKLOAD MANAGEMENT ............ 10
    2.3.1. OpenShift sandboxed containers building blocks .................................... 10
    2.3.2. RHCOS extensions ............................................................................. 10
    2.3.3. Virtualization and OpenShift sandboxed containers .............................. 10
  2.4. UNDERSTANDING COMPLIANCE AND RISK MANAGEMENT .................... 11

CHAPTER 3. DEPLOYING OPENSHELL SANDBOXED CONTAINERS WORKLOADS .......... 12
  3.1. PREREQUISITES ............................................................................................. 12
    3.1.1. Resource requirements for OpenShift sandboxed containers .................. 12
    3.1.2. Checking whether cluster nodes are eligible to run OpenShift sandboxed containers .......................... 14
  3.2. DEPLOYING OPENSHELL SANDBOXED CONTAINERS WORKLOADS USING THE WEB CONSOLE 16
    3.2.1. Installing the OpenShift sandboxed containers Operator using the web console ...................................... 16
    3.2.2. Creating the KataConfig custom resource in the web console ............... 17
    3.2.3. Deploying a workload in a sandboxed container using the web console .... 19
  3.3. DEPLOYING OPENSHELL SANDBOXED CONTAINERS WORKLOADS USING THE CLI 20
    3.3.1. Installing the OpenShift sandboxed containers Operator using the CLI .... 20
    3.3.2. Creating the KataConfig custom resource using the CLI ...................... 22
    3.3.3. Deploying a workload in a sandboxed container using the CLI ............... 24
  3.4. ADDITIONAL RESOURCES ......................................................................... 25

CHAPTER 4. MONITORING OPENSHELL SANDBOXED CONTAINERS .......................... 26
  4.1. ABOUT OPENSHELL SANDBOXED CONTAINERS METRICS ....................... 26
  4.2. VIEWING METRICS FOR OPENSHELL SANDBOXED CONTAINERS .......... 26
  4.3. VIEWING THE OPENSHELL SANDBOXED CONTAINERS DASHBOARD .......... 27
  4.4. ADDITIONAL RESOURCES ......................................................................... 28

CHAPTER 5. UNINSTALLING OPENSHELL SANDBOXED CONTAINERS .......................... 29
  5.1. UNINSTALLING OPENSHELL SANDBOXED CONTAINERS USING THE WEB CONSOLE 29
    5.1.1. Deleting OpenShift sandboxed containers pods using the web console .... 29
    5.1.2. Deleting the KataConfig custom resource using the web console .......... 29
    5.1.3. Deleting the OpenShift sandboxed containers Operator using the web console ........................................... 30
    5.1.4. Deleting the OpenShift sandboxed containers namespace using the web console .................................. 31
    5.1.5. Deleting the KataConfig custom resource definition using the web console ........................................ 31
  5.2. UNINSTALLING OPENSHELL SANDBOXED CONTAINERS USING THE CLI .... 32
    5.2.1. Deleting OpenShift sandboxed containers pods using the CLI .............. 32
    5.2.2. Deleting the KataConfig custom resource using the CLI ..................... 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.3. Deleting the OpenShift sandboxed containers Operator using the CLI</td>
<td>33</td>
</tr>
<tr>
<td>5.2.4. Deleting the KataConfig custom resource definition using the CLI</td>
<td>34</td>
</tr>
<tr>
<td>CHAPTER 6. UPGRADING OPENSIFT SANDBOXED CONTAINERS</td>
<td>36</td>
</tr>
<tr>
<td>6.1. UPGRADING THE OPENSIFT SANDBOXED CONTAINERS RESOURCES</td>
<td>36</td>
</tr>
<tr>
<td>6.2. UPGRADING THE OPENSIFT SANDBOXED CONTAINERS OPERATOR</td>
<td>36</td>
</tr>
<tr>
<td>6.3. UPGRADING THE OPENSIFT SANDBOXED CONTAINERS MONITOR PODS</td>
<td>36</td>
</tr>
<tr>
<td>6.3.1. Upgrading the monitor pods using the web console</td>
<td>37</td>
</tr>
<tr>
<td>6.3.2. Upgrading the monitor pods using the CLI</td>
<td>37</td>
</tr>
<tr>
<td>CHAPTER 7. COLLECTING OPENSIFT SANDBOXED CONTAINERS DATA</td>
<td>38</td>
</tr>
<tr>
<td>7.1. COLLECTING OPENSIFT SANDBOXED CONTAINERS DATA FOR RED HAT SUPPORT</td>
<td>38</td>
</tr>
<tr>
<td>7.1.1. About the must-gather tool</td>
<td>38</td>
</tr>
<tr>
<td>7.2. ABOUT OPENSIFT SANDBOXED CONTAINERS LOG DATA</td>
<td>39</td>
</tr>
<tr>
<td>7.3. ENABLING DEBUG LOGS FOR OPENSIFT SANDBOXED CONTAINERS</td>
<td>39</td>
</tr>
<tr>
<td>7.3.1. Viewing debug logs for OpenShift sandboxed containers</td>
<td>40</td>
</tr>
<tr>
<td>7.4. ADDITIONAL RESOURCES</td>
<td>41</td>
</tr>
</tbody>
</table>
CHAPTER 1. OPENShift SANDBOXED CONTAINERS 1.3
RELEASE NOTES

1.1. ABOUT THIS RELEASE

These release notes track the development of OpenShift sandboxed containers 1.3 alongside Red Hat OpenShift Container Platform 4.11.

This product is fully supported and enabled by default as of OpenShift Container Platform 4.10.

1.2. NEW FEATURES AND ENHANCEMENTS

1.2.1. Container ID in metrics list

The sandbox_id with the ID of the relevant sandboxed container now appears in the metrics list on the Metrics page in the web console.

In addition, the kata-monitor process now adds three new labels to kata-specific metrics: cri_uid, cri_name, and cri_namespace. These labels enable kata-specific metrics to relate to corresponding kubernetes workloads.

For more information about kata-specific metrics, see About OpenShift sandboxed containers metrics.

1.2.2. OpenShift sandboxed containers availability on AWS bare metal

Previously, OpenShift sandboxed containers availability on AWS bare metal was in Technology Preview. With this release, installing OpenShift sandboxed containers on AWS bare-metal clusters is fully supported.

1.2.3. Support for OpenShift sandboxed containers on single-node OpenShift

OpenShift sandboxed containers now work on single-node OpenShift clusters when the OpenShift sandboxed containers Operator is installed by Red Hat Advanced Cluster Management (RHACM).

1.3. BUG FIXES

- Previously, when creating the KataConfig CR and observing the pod status under the openshift-sandboxed-containers-operator namespace, a huge number of restarts for monitor pods was shown. The monitor pods use a specific SELinux policy that was installed as part of the sandboxed-containers extension installation. The monitor pod was created immediately. However, the SELinux policy was not yet available, which resulted in a pod creation error, followed by a pod restart. With this release, the SELinux policy is available when the monitor pod is created, and the monitor pod transitions to a Running state immediately. (KATA-1338)

- Previously, OpenShift sandboxed containers deployed a security context constraint (SCC) on startup which enforced a custom SELinux policy that was not available on Machine Config Operator (MCO) pods. This caused the MCO pod to change to a CrashLoopBackOff state and cluster upgrades to fail. With this release, OpenShift sandboxed containers deploys the SCC when creating the KataConfig CR and no longer enforces using the custom SELinux policy. (KATA-1373)
Previously, when uninstalling the OpenShift sandboxed containers Operator, the `sandboxed-containers-operator-scc` custom resource was not deleted. With this release, the `sandboxed-containers-operator-scc` custom resource is deleted when uninstalling the OpenShift sandboxed containers Operator. (KATA-1569)

1.4. KNOWN ISSUES

- If you are using OpenShift sandboxed containers, you might receive SELinux denials when accessing files or directories mounted from the `hostPath` volume in an OpenShift Container Platform cluster. These denials can occur even when running privileged sandboxed containers because privileged sandboxed containers do not disable SELinux checks.
  
  Following SELinux policy on the host guarantees full isolation of the host file system from the sandboxed workload by default. This also provides stronger protection against potential security flaws in the `virtiofsd` daemon or QEMU.

  If the mounted files or directories do not have specific SELinux requirements on the host, you can use local persistent volumes as an alternative. Files are automatically relabeled to `container_file_t`, following SELinux policy for container runtimes. See Persistent storage using local volumes for more information.

  Automatic relabeling is not an option when mounted files or directories are expected to have specific SELinux labels on the host. Instead, you can set custom SELinux rules on the host to allow the `virtiofsd` daemon to access these specific labels. (BZ#1904609)

- Some OpenShift sandboxed containers Operator pods use container CPU resource limits to increase the number of available CPUs for the pod. These pods might receive fewer CPUs than requested. If the functionality is available inside the container, you can diagnose CPU resource issues by using `oc rsh <pod>` to access a pod and running the `lscpu` command:

```
$ lscpu
```

Example output

```
CPU(s): 16
On-line CPU(s) list: 0-12,14,15
Off-line CPU(s) list: 13
```

The list of offline CPUs will likely change unpredictably from run to run.

As a workaround, you can use a pod annotation to request additional CPUs rather than setting a CPU limit. CPU requests that use pod annotation are not affected by this issue, because the processor allocation method is different. Rather than setting a CPU limit, the following annotation must be added to the metadata of the pod:

```
metadata:
  annotations:
    io.katacontainers.config.hypervisor.default_vcpus: "16"
```

(KATA-1376)

- The progress of the runtime installation is shown in the `status` section of the `kataConfig` custom resource (CR). However, the progress is not shown if all of the following conditions are true:
  - There are no worker nodes defined. You can run `oc get machineconfigpool` to check the
number of worker nodes in the machine config pool.

- No `kataConfigPoolSelector` is specified to select nodes for installation.

In this case, the installation starts on the control plane nodes because the Operator assumes it is a converged cluster where nodes have both control plane and worker roles. The `status` section of the `kataConfig` CR is not updated during the installation. (KATA-1017)

- When using older versions of the Buildah tool in OpenShift sandboxed containers, the build fails with the following error:

```
process exited with error: fork/exec /bin/sh: no such file or directory
subprocess exited with status 1
```

You must use the latest version of Buildah, available at quay.io. (KATA-1278)

- In the `KataConfig` tab in the web console, if you click `Create KataConfig` while in the YAML view, the `KataConfig` YAML is missing the `spec` fields. Toggling to the Form view and then back to the YAML view fixes this issue and displays the full YAML. (KATA-1372)

- In the `KataConfig` tab in the web console, a 404: Not found error message appears whether a `KataConfig` CR already exists or not. To access an existing `KataConfig` CR, go to Home > Search. From the Resources list, select `KataConfig`. (KATA-1605)

- Upgrading OpenShift sandboxed containers does not automatically update the existing `KataConfig` CR. As a result, monitor pods from previous deployments are not restarted and continue to run with an outdated `kataMonitor` image. Upgrade the `kataMonitor` image with the following command:

```
$ oc patch kataconfig example-kataconfig --type merge --patch '{"spec": {
  "kataMonitorImage": "registry.redhat.io/openshift-sandboxed-containers/osc-monitor-rhel8:1.3.0"
}}'
```

You can also upgrade the `kataMonitor` image by editing the `KataConfig` YAML in the web console. (KATA-1650)

1.5. ASYNCHRONOUS ERRATA UPDATES

Security, bug fix, and enhancement updates for OpenShift sandboxed containers 4.11 are released as asynchronous errata through the Red Hat Network. All OpenShift Container Platform 4.11 errata are available on the Red Hat Customer Portal. For more information about asynchronous errata, see the OpenShift Container Platform Life Cycle.

Red Hat Customer Portal users can enable errata notifications in the account settings for Red Hat Subscription Management (RHSM). When errata notifications are enabled, users are notified by email whenever new errata relevant to their registered systems are released.
NOTE

Red Hat Customer Portal user accounts must have systems registered and consuming OpenShift Container Platform entitlements for OpenShift Container Platform errata notification emails to generate.

This section will continue to be updated over time to provide notes on enhancements and bug fixes for future asynchronous errata releases of OpenShift sandboxed containers 1.3.

1.5.1. RHSA-2022:6072 - OpenShift sandboxed containers 1.3.0 image release, bug fix, and enhancement advisory.

Issued: 2022-08-17

OpenShift sandboxed containers release 1.3.0 is now available. This advisory contains an update for OpenShift sandboxed containers with enhancements and bug fixes.

The list of bug fixes included in the update is documented in the RHSA-2022:6072 advisory.

1.5.2. RHSA-2022:7058 - OpenShift sandboxed containers 1.3.1 security fix and bug fix advisory.

Issued: 2022-10-19

OpenShift sandboxed containers release 1.3.1 is now available. This advisory contains an update for OpenShift sandboxed containers with security fixes and a bug fix.

The list of bug fixes included in the update is documented in the RHSA-2022:7058 advisory.
CHAPTER 2. UNDERSTANDING OPENSHIFT SANDBOXED CONTAINERS

OpenShift sandboxed containers support for OpenShift Container Platform provides you with built-in support for running Kata Containers as an additional optional runtime. The new runtime supports containers in dedicated virtual machines (VMs), providing improved workload isolation. This is particularly useful for performing the following tasks:

Run privileged or untrusted workloads

OpenShift sandboxed containers (OSC) makes it possible to safely run workloads that require specific privileges, without having to risk compromising cluster nodes by running privileged containers. Workloads that require special privileges include the following:

- Workloads that require special capabilities from the kernel, beyond the default ones granted by standard container runtimes such as CRI-O, for example to access low-level networking features.

- Workloads that need elevated root privileges, for example to access a specific physical device. With OpenShift sandboxed containers, it is possible to pass only a specific device through to the VM, ensuring that the workload cannot access or misconfigure the rest of the system.

- Workloads for installing or using set-uid root binaries. These binaries grant special privileges and, as such, can present a security risk. With OpenShift sandboxed containers, additional privileges are restricted to the virtual machines, and grant no special access to the cluster nodes.

Some workloads may require privileges specifically for configuring the cluster nodes. Such workloads should still use privileged containers, because running on a virtual machine would prevent them from functioning.

Ensure kernel isolation for each workload

OpenShift sandboxed containers supports workloads that require custom kernel tuning (such as sysctl, scheduler changes, or cache tuning) and the creation of custom kernel modules (such as out of tree or special arguments).

Share the same workload across tenants

OpenShift sandboxed containers enables you to support multiple users (tenants) from different organizations sharing the same OpenShift cluster. The system also lets you run third-party workloads from multiple vendors, such as container network functions (CNFs) and enterprise applications. Third-party CNFs, for example, may not want their custom settings interfering with packet tuning or with sysctl variables set by other applications. Running inside a completely isolated kernel is helpful in preventing “noisy neighbor” configuration problems.

Ensure proper isolation and sandboxing for testing software

You can use OpenShift sandboxed containers to run a containerized workload with known vulnerabilities or to handle an issue in a legacy application. This isolation also enables administrators to give developers administrative control over pods, which is useful when the developer wants to test or validate configurations beyond those an administrator would typically grant. Administrators can, for example, safely and securely delegate kernel packet filtering (eBPF) to developers. Kernel packet filtering requires CAP_ADMIN or CAP_BPF privileges, and is therefore not allowed under a standard CRI-O configuration, as this would grant access to every process on the Container Host worker node. Similarly, administrators can grant access to intrusive tools such as SystemTap, or support the loading of custom kernel modules during their development.

Ensure default resource containment through VM boundaries
By default, resources such as CPU, memory, storage, or networking are managed in a more robust and secure way in OpenShift sandboxed containers. Since OpenShift sandboxed containers are deployed on VMs, additional layers of isolation and security give a finer-grained access control to the resource. For example, an errant container will not be able to allocate more memory than is available to the VM. Conversely, a container that needs dedicated access to a network card or to a disk can take complete control over that device without getting any access to other devices.

2.1. OPENSHIFT SANDBOXED CONTAINERS SUPPORTED PLATFORMS

You can install OpenShift sandboxed containers on a bare-metal server or on an Amazon Web Services (AWS) bare-metal instance. Bare-metal instances offered by other cloud providers are not supported.

Red Hat Enterprise Linux CoreOS (RHCOS) is the only supported operating system for OpenShift sandboxed containers. OpenShift sandboxed containers 1.3 runs on Red Hat Enterprise Linux CoreOS (RHCOS) 8.6.

OpenShift sandboxed containers 1.3 is compatible with OpenShift Container Platform 4.11.

2.2. OPENSHIFT SANDBOXED CONTAINERS COMMON TERMS

The following terms are used throughout the documentation.

**Sandbox**

A sandbox is an isolated environment where programs can run. In a sandbox, you can run untested or untrusted programs without risking harm to the host machine or the operating system.

In the context of OpenShift sandboxed containers, sandboxing is achieved by running workloads in a different kernel using virtualization, providing enhanced control over the interactions between multiple workloads that run on the same host.

**Pod**

A pod is a construct that is inherited from Kubernetes and OpenShift Container Platform. It represents resources where containers can be deployed. Containers run inside of pods, and pods are used to specify resources that can be shared between multiple containers.

In the context of OpenShift sandboxed containers, a pod is implemented as a virtual machine. Several containers can run in the same pod on the same virtual machine.

**OpenShift sandboxed containers Operator**

An Operator is a software component that automates operations, which are actions that a human operator could do on the system.

The OpenShift sandboxed containers Operator is tasked with managing the lifecycle of sandboxed containers on a cluster. You can use the OpenShift sandboxed containers Operator to perform tasks such as the installation and removal of sandboxed containers, software updates, and status monitoring.

**Kata Containers**

Kata Containers is a core upstream project that is used to build OpenShift sandboxed containers. OpenShift sandboxed containers integrate Kata Containers with OpenShift Container Platform.

**KataConfig**

KataConfig objects represent configurations of sandboxed containers. They store information about the state of the cluster, such as the nodes on which the software is deployed.
Runtime class

A `RuntimeClass` object describes which runtime can be used to run a given workload. A runtime class that is named `kata` is installed and deployed by the OpenShift sandboxed containers Operator. The runtime class contains information about the runtime that describes resources that the runtime needs to operate, such as the pod overhead.

2.3. OPENSHIFT SANDBOXED CONTAINERS WORKLOAD MANAGEMENT

OpenShift sandboxed containers provides the following features for enhancing workload management and allocation:

2.3.1. OpenShift sandboxed containers building blocks

The OpenShift sandboxed containers Operator encapsulates all of the components from Kata containers. It manages installation, lifecycle, and configuration tasks.

The OpenShift sandboxed containers Operator is packaged in the Operator bundle format as two container images. The bundle image contains metadata and is required to make the operator OLM-ready. The second container image contains the actual controller that monitors and manages the `KataConfig` resource.

2.3.2. RHCOS extensions

The OpenShift sandboxed containers Operator is based on the Red Hat Enterprise Linux CoreOS (RHCOS) extensions concept. Red Hat Enterprise Linux CoreOS (RHCOS) extensions are a mechanism to install optional OpenShift Container Platform software. The OpenShift sandboxed containers Operator uses this mechanism to deploy sandboxed containers on a cluster.

The sandboxed containers RHCOS extension contains RPMs for Kata, QEMU, and its dependencies. You can enable them by using the `MachineConfig` resources that the Machine Config Operator provides.

Additional resources

- Adding extensions to RHCOS

2.3.3. Virtualization and OpenShift sandboxed containers

You can use OpenShift sandboxed containers on clusters with OpenShift Virtualization.

To run OpenShift Virtualization and OpenShift sandboxed containers at the same time, you must enable VMs to migrate, so that they do not block node reboots. Configure the following parameters on your VM:

- Use `ocs-storagecluster-ceph-rbd` as the storage class.
- Set the `evictionStrategy` parameter to `LiveMigrate` in the VM.

Additional resources

- Configuring local storage for virtual machines
- Configuring virtual machine eviction strategy
2.4. UNDERSTANDING COMPLIANCE AND RISK MANAGEMENT

OpenShift Container Platform is designed for FIPS. When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures.

For more information about the NIST validation program, see Cryptographic Module Validation Program. For the latest NIST status for the individual versions of RHEL cryptographic libraries that have been submitted for validation, see Compliance Activities and Government Standards.

OpenShift sandboxed containers can be used on FIPS enabled clusters.

When running in FIPS mode, OpenShift sandboxed containers components, VMs, and VM images are adapted to comply with FIPS.

FIPS compliance is one of the most critical components required in highly secure environments, to ensure that only supported cryptographic technologies are allowed on nodes.

IMPORTANT

To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see Installing the system in FIPS mode. The use of FIPS validated or Modules In Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the x86_64 architecture.

CHAPTER 3. DEPLOYING OPENSHIFT SANDBOXED CONTAINERS WORKLOADS

You can install the OpenShift sandboxed containers Operator using either the web console or OpenShift CLI (oc). Before installing the OpenShift sandboxed containers Operator, you must prepare your OpenShift Container Platform cluster.

3.1. PREREQUISITES

Before you install OpenShift sandboxed containers, ensure that your OpenShift Container Platform cluster meets the following requirements:

- Your cluster must be installed on on-premise bare-metal infrastructure with Red Hat Enterprise Linux CoreOS (RHCOS) workers. You can use any installation method including user-provisioned, installer-provisioned, or assisted installer to deploy your cluster.

  NOTE

  - OpenShift sandboxed containers only supports RHCOS worker nodes. RHEL nodes are not supported.
  - Nested virtualization is not supported.

- You can install OpenShift sandboxed containers on Amazon Web Services (AWS) bare-metal instances. Bare-metal instances offered by other cloud providers are not supported.

3.1.1. Resource requirements for OpenShift sandboxed containers

OpenShift sandboxed containers lets users run workloads on their OpenShift Container Platform clusters inside a sandboxed runtime (Kata). Each pod is represented by a virtual machine (VM). Each VM runs in a QEMU process and hosts a kata-agent process that acts as a supervisor for managing container workloads, and the processes running in those containers. Two additional processes add more overhead:

- containerd-shim-kata-v2 is used to communicate with the pod.
- virtiofsd handles host file system access on behalf of the guest.

Each VM is configured with a default amount of memory. Additional memory is hot-plugged into the VM for containers that explicitly request memory.

A container running without a memory resource consumes free memory until the total memory used by the VM reaches the default allocation. The guest and its I/O buffers also consume memory.

If a container is given a specific amount of memory, then that memory is hot-plugged into the VM before the container starts.

When a memory limit is specified, the workload is terminated if it consumes more memory than the limit. If no memory limit is specified, the kernel running on the VM might run out of memory. If the kernel runs out of memory, it might terminate other processes on the VM.

Default memory sizes

The following table lists some the default values for resource allocation.
<table>
<thead>
<tr>
<th>Resource</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory allocated by default to a virtual machine</td>
<td>2Gi</td>
</tr>
<tr>
<td>Guest Linux kernel memory usage at boot</td>
<td>~110Mi</td>
</tr>
<tr>
<td>Memory used by the QEMU process (excluding VM memory)</td>
<td>~30Mi</td>
</tr>
<tr>
<td>Memory used by the virtiofsd process (excluding VM I/O buffers)</td>
<td>~10Mi</td>
</tr>
<tr>
<td>Memory used by the containerd-shim-kata-v2 process</td>
<td>~20Mi</td>
</tr>
<tr>
<td>File buffer cache data after running <code>dnf install</code> on Fedora</td>
<td>~300Mi*</td>
</tr>
</tbody>
</table>

File buffers appear and are accounted for in multiple locations:

- In the guest where it appears as file buffer cache.
- In the virtiofsd daemon that maps allowed user-space file I/O operations.
- In the QEMU process as guest memory.

**NOTE**

Total memory usage is properly accounted for by the memory utilization metrics, which only count that memory once.

**Pod overhead** describes the amount of system resources that a pod on a node uses. You can get the current pod overhead for the Kata runtime by using `oc describe runtimeclass kata` as shown below.

**Example**

```
$ oc describe runtimeclass kata
```

**Example output**

```
kind: RuntimeClass
apiVersion: node.k8s.io/v1
metadata:
  name: kata
overhead:
podFixed:
  memory: "500Mi"
cpu: "500m"
```
You can change the pod overhead by changing the `spec.overhead` field for a RuntimeClass. For example, if the configuration that you run for your containers consumes more than 350Mi of memory for the QEMU process and guest kernel data, you can alter the RuntimeClass overhead to suit your needs.

**NOTE**

The specified default overhead values are supported by Red Hat. Changing default overhead values is not supported and can result in technical issues.

When performing any kind of file system I/O in the guest, file buffers are allocated in the guest kernel. The file buffers are also mapped in the QEMU process on the host, as well as in the `virtiofsd` process.

For example, if you use 300Mi of file buffer cache in the guest, both QEMU and `virtiofsd` appear to use 300Mi additional memory. However, the same memory is being used in all three cases. In other words, the total memory usage is only 300Mi, mapped in three different places. This is correctly accounted for when reporting the memory utilization metrics.

**Additional resources**

- Installing a user-provisioned cluster on bare metal

### 3.1.2. Checking whether cluster nodes are eligible to run OpenShift sandboxed containers

Before running OpenShift sandboxed containers, you can check whether the nodes in your cluster are eligible to run Kata containers. Some cluster nodes might not comply with sandboxed containers’ minimum requirements. The most common reason for node ineligibility is the lack of virtualization support on the node. If you attempt to run sandboxed workloads on ineligible nodes, you will experience errors. You can use the Node Feature Discovery (NFD) Operator and a `NodeFeatureDiscovery` resource to automatically check node eligibility.

**NOTE**

If you want to install the Kata runtime on only selected worker nodes that you know are eligible, apply the `feature.node.kubernetes.io/runtime.kata=true` label to the selected nodes and set `checkNodeEligibility: true` in the `KataConfig` resource.

Alternatively, to install the Kata runtime on all worker nodes, set `checkNodeEligibility: false` in the `KataConfig` resource.

In both these scenarios, you do not need to create the `NodeFeatureDiscovery` resource. You should only apply the `feature.node.kubernetes.io/runtime.kata=true` label manually if you are sure that the node is eligible to run Kata containers.

The following procedure applies the `feature.node.kubernetes.io/runtime.kata=true` label to all eligible nodes and configures the `KataConfig` resource to check for node eligibility.

**Prerequisites**

- Install the OpenShift CLI (`oc`).
- Log in as a user with `cluster-admin` privileges.
- Install the Node Feature Discovery (NFD) Operator.
Procedure

1. Create a **NodeFeatureDiscovery** resource to detect node capabilities suitable for running Kata containers:
   
   a. Save the following YAML in the `nfd.yaml` file:

   ```yaml
   apiVersion: nfd.openshift.io/v1
   kind: NodeFeatureDiscovery
   metadata:
     name: nfd-kata
     namespace: openshift-nfd
   spec:
     operand:
       image: quay.io/openshift/origin-node-feature-discovery:4.10
       imagePullPolicy: Always
       servicePort: 12000
     workerConfig:
       configData: |
       sources:
         custom:
           - name: "feature.node.kubernetes.io/runtime.kata"
             matchOn:
               - cpuid: ["SSE4", "VMX"]
                 loadedKMod: ["kvm", "kvm_intel"]
               - cpuid: ["SSE4", "SVM"]
                 loadedKMod: ["kvm", "kvm_amd"]
   
   b. Create the **NodeFeatureDiscovery** custom resource (CR):

   ```bash
   $ oc create -f nfd.yaml
   ```

   **Example output**

   ```bash
   nodefeaturediscovery.nfd.openshift.io/nfd-kata created
   ```

   A `feature.node.kubernetes.io/runtime.kata=true` label is applied to all qualifying worker nodes.

2. Set the **checkNodeEligibility** field to **true** in the **KataConfig** resource to enable the feature, for example:

   a. Save the following YAML in the `kata-config.yaml` file:

   ```yaml
   apiVersion: kataconfiguration.openshift.io/v1
   kind: KataConfig
   metadata:
     name: example-kataconfig
   spec:
     checkNodeEligibility: true
   
   b. Create the **KataConfig** CR:

   ```bash
   $ oc create -f kata-config.yaml
   ```
Example output

```
kataconfig.kataconfiguration.openshift.io/example-kataconfig created
```

**Verification**

- Verify that qualifying nodes in the cluster have the correct label applied:

```
$ oc get nodes --selector='feature.node.kubernetes.io/runtime.kata=true'
```

**Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ROLES</th>
<th>AGE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute-3.example.com</td>
<td>Ready</td>
<td>worker</td>
<td>4h38m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>compute-2.example.com</td>
<td>Ready</td>
<td>worker</td>
<td>4h35m</td>
<td>v1.24.0</td>
</tr>
</tbody>
</table>

**Additional resources**

- For more information about installing the Node Feature Discovery (NFD) Operator, see [Installing NFD](#).

### 3.2. DEPLOYING OPENSFiRT SANDBOXED CONTAINERS WORKLOADS USING THE WEB CONSOLE

You can deploy OpenShift sandboxed containers workloads from the web console. First, you must install the OpenShift sandboxed containers Operator, then create the **KataConfig** custom resource (CR). Once you are ready to deploy a workload in a sandboxed container, you must manually add **kata** as the `runtimeClassName` to the workload YAML file.

#### 3.2.1. Installing the OpenShift sandboxed containers Operator using the web console

You can install the OpenShift sandboxed containers Operator from the OpenShift Container Platform web console.

**Prerequisites**

- You have OpenShift Container Platform 4.11 installed.
- You have access to the cluster as a user with the **cluster-admin** role.

**Procedure**

1. From the **Administrator** perspective in the web console, navigate to **Operators** → **OperatorHub**.
2. In the **Filter by keyword** field, type **OpenShift sandboxed containers**.
3. Select the **OpenShift sandboxed containers** tile.
4. Read the information about the Operator and click **Install**.
5. On the **Install Operator** page:
a. Select **stable-1.3** from the list of available **Update Channel** options.

b. Verify that **Operator recommended Namespace** is selected for **Installed Namespace**. This installs the Operator in the mandatory **openshift-sandboxed-containers-operator** namespace. If this namespace does not yet exist, it is automatically created.

![NOTE]

Attempted to install the OpenShift sandboxed containers Operator in a namespace other than **openshift-sandboxed-containers-operator** causes the installation to fail.

c. Verify that **Automatic** is selected for **Approval Strategy**. **Automatic** is the default value, and enables automatic updates to OpenShift sandboxed containers when a new z-stream release is available.

6. Click **Install**.

The OpenShift sandboxed containers Operator is now installed on your cluster.

**Verification**

1. From the **Administrator** perspective in the web console, navigate to **Operators → Installed Operators**.

2. Verify that the OpenShift sandboxed containers Operator is listed in the in operators list.

### 3.2.2. Creating the KataConfig custom resource in the web console

You must create one **KataConfig** custom resource (CR) to enable installing **kata** as a **RuntimeClass** on your cluster nodes.

![IMPORTANT]

Creating the **KataConfig** CR automatically reboots the worker nodes. The reboot can take from 10 to more than 60 minutes. Factors that impede reboot time are as follows:

- A larger OpenShift Container Platform deployment with a greater number of worker nodes.
- Activation of the BIOS and Diagnostics utility.
- Deployment on a hard drive rather than on an SSD.
- Deployment on physical nodes such as bare metal, rather than on virtual nodes.
- A slow CPU or network.

**Prerequisites**

- You have installed OpenShift Container Platform 4.11 on your cluster.
- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift sandboxed containers Operator.
NOTE

Kata is installed on all worker nodes by default. If you want to install kata as a RuntimeClass only on specific nodes, you can add labels to those nodes, then define the label in the KataConfig CR when you create it.

Procedure

1. From the Administrator perspective in the web console, navigate to Operators → Installed Operators.

2. Select the OpenShift sandboxed containers Operator from the list of operators.

3. In the KataConfig tab, click Create KataConfig.

4. In the Create KataConfig page, select to configure the KataConfig CR via YAML view.

5. Copy and paste the following manifest into the YAML view:

   ```yaml
   apiVersion: kataconfiguration.openshift.io/v1
   kind: KataConfig
   metadata:
     name: cluster-kataconfig
   spec:
     kataMonitorImage: registry.redhat.io/openshift-sandboxed-containers/osc-monitor-rhel8:1.3.0
   ```

   If you want to install kata as a RuntimeClass only on selected nodes, include the label in the manifest:

   ```yaml
   apiVersion: kataconfiguration.openshift.io/v1
   kind: KataConfig
   metadata:
     name: cluster-kataconfig
   spec:
     kataMonitorImage: registry.redhat.io/openshift-sandboxed-containers/osc-monitor-rhel8:1.3.0
     kataConfigPoolSelector:
       matchLabels:
         # <label_key>: '<label_value>'
   ```

   1 Labels in kataConfigPoolSelector only support single values; nodeSelector syntax is not supported.

6. Click Create.

The new KataConfig CR is created and begins to install kata as a RuntimeClass on the worker nodes. Wait for the kata installation to complete and the worker nodes to reboot before continuing to the next step.

IMPORTANT

OpenShift sandboxed containers installs Kata only as a secondary, optional runtime on the cluster and not as the primary runtime.
Verification

1. In the KataConfig tab, select the new KataConfig CR.

2. In the KataConfig page, select the YAML tab.

3. Monitor the installationStatus field in the status. A message appears each time there is an update. Click Reload to view the updated KataConfig CR.

Once the value of Completed nodes equals the number of worker or labeled nodes, the installation is complete. The status also contains a list of nodes where the installation is completed.

3.2.3. Deploying a workload in a sandboxed container using the web console

OpenShift sandboxed containers installs Kata as a secondary, optional runtime on your cluster, and not as the primary runtime.

To deploy a pod-templated workload in a sandboxed container, you must manually add kata as the runtimeClassName to the workload YAML file.

Prerequisites

- You have installed OpenShift Container Platform 4.11 on your cluster.
- You have access to the cluster as a user with the cluster-admin role.
- You have installed the OpenShift sandboxed containers Operator.
- You have created a KataConfig custom resource (CR).

Procedure

1. From the Administrator perspective in the web console, expand Workloads and select the type of workload you want to create.

2. In the workload page, click to create the workload.

3. In the YAML file for the workload, in the spec field where the container is listed, add runtimeClassName: kata.

Example for Pod object

```yaml
apiVersion: v1
class: Pod
metadata:
  name: hello-openshift
spec:
  runtimeClassName: kata
containers:
- name: hello-openshift
  image: quay.io/openshift/origin-hello-openshift
```
Click Save.

OpenShift Container Platform creates the workload and begins scheduling it.

3.3. DEPLOYING OPENSHIFT SANDBOXED CONTAINERS WORKLOADS USING THE CLI

You can deploy OpenShift sandboxed containers workloads using the CLI. First, you must install the OpenShift sandboxed containers Operator, then create the KataConfig custom resource. Once you are ready to deploy a workload in a sandboxed container, you must add kata as the runtimeClassName to the workload YAML file.

3.3.1. Installing the OpenShift sandboxed containers Operator using the CLI

You can install the OpenShift sandboxed containers Operator using the OpenShift Container Platform CLI.

Prerequisites

- You have OpenShift Container Platform 4.11 installed on your cluster.
- You have installed the OpenShift CLI (oc).
- You have access to the cluster as a user with the cluster-admin role.
- You have subscribed to the OpenShift sandboxed containers catalog.

NOTE

Subscribing to the OpenShift sandboxed containers catalog provides openshift-sandboxed-containers-operator namespace access to the OpenShift sandboxed containers Operator.

Procedure

1. Create the Namespace object for the OpenShift sandboxed containers Operator.

   a. Create a Namespace object YAML file that contains the following manifest:

   ```yaml
   apiVersion: v1
   kind: Namespace
   metadata:
   ```
name: openshift-sandboxed-containers-operator

b. Create the **Namespace** object:

```
$ oc create -f Namespace.yaml
```

2. Create the **OperatorGroup** object for the OpenShift sandboxed containers Operator.

a. Create an **OperatorGroup** object YAML file that contains the following manifest:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-sandboxed-containers-operator
  namespace: openshift-sandboxed-containers-operator
spec:
  targetNamespaces:
  - openshift-sandboxed-containers-operator
```

b. Create the **OperatorGroup** object:

```
$ oc create -f OperatorGroup.yaml
```

3. Create the **Subscription** object to subscribe the **Namespace** to the OpenShift sandboxed containers Operator.

a. Create a **Subscription** object YAML file that contains the following manifest:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-sandboxed-containers-operator
  namespace: openshift-sandboxed-containers-operator
spec:
  channel: "stable-1.3"
  installPlanApproval: Automatic
  name: sandboxed-containers-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: sandboxed-containers-operator.v1.3.1
```

b. Create the **Subscription** object:

```
$ oc create -f Subscription.yaml
```

The OpenShift sandboxed containers Operator is now installed on your cluster.

**NOTE**

All the object file names listed above are suggestions. You can create the object YAML files using other names.

**Verification**
Ensure that the Operator is correctly installed:

```
$ oc get csv -n openshift-sandboxed-containers-operator
```

Example output

<table>
<thead>
<tr>
<th>NAME</th>
<th>DISPLAY</th>
<th>VERSION</th>
<th>REPLACES</th>
<th>PHASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>openshift-sandboxed-containers</td>
<td>openshift-sandboxed-containers-operator</td>
<td>1.3.1</td>
<td>1.3.0</td>
<td>Succeeded</td>
</tr>
</tbody>
</table>

Additional resources

- Installing from OperatorHub using the CLI

### 3.3.2. Creating the KataConfig custom resource using the CLI

You must create one KataConfig custom resource (CR) to install kata as a RuntimeClass on your nodes. Creating the KataConfig CR triggers the OpenShift sandboxed containers Operator to do the following:

- Install the needed RHCOS extensions, such as QEMU and kata-containers, on your RHCOS node.
- Ensure that the CRI-O runtime is configured with the correct kata runtime handlers.
- Create a RuntimeClass CR named kata with a default configuration. This enables users to configure workloads to use kata as the runtime by referencing the CR in the RuntimeClassName field. This CR also specifies the resource overhead for the runtime.

**NOTE**

Kata is installed on all worker nodes by default. If you want to install kata as a RuntimeClass only on specific nodes, you can add labels to those nodes, then define the label in the KataConfig CR when you create it.

### Prerequisites

- You have installed OpenShift Container Platform 4.11 on your cluster.
- You have installed the OpenShift CLI (oc).
- You have access to the cluster as a user with the cluster-admin role.
- You have installed the OpenShift sandboxed containers Operator.
**IMPORTANT**

Creating the **KataConfig** CR automatically reboots the worker nodes. The reboot can take from 10 to more than 60 minutes. Factors that impede reboot time are as follows:

- A larger OpenShift Container Platform deployment with a greater number of worker nodes.
- Activation of the BIOS and Diagnostics utility.
- Deployment on a hard drive rather than on an SSD.
- Deployment on physical nodes such as bare metal, rather than on virtual nodes.
- A slow CPU or network.

**Procedure**

1. Create a YAML file with the following manifest:

   ```yaml
   apiVersion: kataconfiguration.openshift.io/v1
   kind: KataConfig
   metadata:
     name: cluster-kataconfig
   spec:
     kataMonitorImage: registry.redhat.io/openshift-sandboxed-containers/osc-monitor-rhel8:1.3.0
   ```

2. (Optional) If you want to install **kata** as a **RuntimeClass** only on selected nodes, create a YAML file that includes the label in the manifest:

   ```yaml
   apiVersion: kataconfiguration.openshift.io/v1
   kind: KataConfig
   metadata:
     name: cluster-kataconfig
   spec:
     kataMonitorImage: registry.redhat.io/openshift-sandboxed-containers/osc-monitor-rhel8:1.3.0
     kataConfigPoolSelector:
       matchLabels:
         `<label_key>`: `<label_value>`
   ```

   **1** Labels in **kataConfigPoolSelector** only support single values; **nodeSelector** syntax is not supported.

3. Create the **KataConfig** resource:

   ```bash
   $ oc create -f <file name>.yaml
   ```

The new **KataConfig** CR is created and begins to install **kata** as a **RuntimeClass** on the worker nodes. Wait for the **kata** installation to complete and the worker nodes to reboot before continuing to the next step.
IMPORTANT

OpenShift sandboxed containers installs Kata only as a secondary, optional runtime on the cluster and not as the primary runtime.

Verification

- Monitor the installation progress:

  ```sh
  $ watch "oc describe kataconfig | sed -n /^Status:/,/^Events/p"
  ```

  Once the value of **Is In Progress** appears as **false**, the installation is complete.

Additional resources

- Understanding how to update labels on nodes

3.3.3. Deploying a workload in a sandboxed container using the CLI

OpenShift sandboxed containers installs Kata as a secondary, optional runtime on your cluster, and not as the primary runtime.

To deploy a pod-templated workload in a sandboxed container, you must add **kata** as the **runtimeClassName** to the workload YAML file.

Prerequisites

- You have installed OpenShift Container Platform 4.11 on your cluster.
- You have installed the OpenShift CLI (**oc**).
- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift sandboxed containers Operator.
- You have created a **KataConfig** custom resource (CR).

Procedure

- Add **runtimeClassName: kata** to any pod-templated object:
  - **Pod** objects
  - **ReplicaSet** objects
  - **ReplicationController** objects
  - **StatefulSet** objects
  - **Deployment** objects
  - **DeploymentConfig** objects

Example for **Pod** object
OpenShift Container Platform creates the workload and begins scheduling it.

**Verification**

- Inspect the `runtimeClassName` field on a pod-templated object. If the `runtimeClassName` is `kata`, then the workload is running on a OpenShift sandboxed containers.

### 3.4. ADDITIONAL RESOURCES

- The OpenShift sandboxed containers Operator is supported in a restricted network environment. For more information, [Using Operator Lifecycle Manager on restricted networks](#).

- When using a disconnected cluster on a restricted network, you must [configure proxy support in Operator Lifecycle Manager](#) to access the OperatorHub. Using a proxy allows the cluster to fetch the OpenShift sandboxed containers Operator.
CHAPTER 4. MONITORING OPENSCHIFT SANDBOXED CONTAINERS

You can use the OpenShift Container Platform web console to monitor metrics related to the health status of your sandboxed workloads and nodes.

OpenShift sandboxed containers has a pre-configured dashboard available in the web console, and administrators can also access and query raw metrics through Prometheus.

4.1. ABOUT OPENSCHIFT SANDBOXED CONTAINERS METRICS

OpenShift sandboxed containers metrics enable administrators to monitor how their sandboxed containers are running. You can query for these metrics in Metrics UI in the web console.

OpenShift sandboxed containers metrics are collected for the following categories:

Kata agent metrics
Kata agent metrics display information about the kata agent process running in the VM embedded in your sandboxed containers. These metrics include data from `/proc/<pid>/[io, stat, status].

Kata guest OS metrics
Kata guest OS metrics display data from the guest OS running in your sandboxed containers. These metrics include data from `/proc/[stats, diskstats, meminfo, vmstats] and `/proc/net/dev.

Hypervisor metrics
Hypervisor metrics display data regarding the hypervisor running the VM embedded in your sandboxed containers. These metrics mainly include data from `/proc/<pid>/[io, stat, status].

Kata monitor metrics
Kata monitor is the process that gathers metric data and makes it available to Prometheus. The kata monitor metrics display detailed information about the resource usage of the kata-monitor process itself. These metrics also include counters from Prometheus data collection.

Kata containerd shim v2 metrics
Kata containerd shim v2 metrics display detailed information about the kata shim process. These metrics include data from `/proc/<pid>/[io, stat, status] and detailed resource usage metrics.

4.2. VIEWING METRICS FOR OPENSCHIFT SANDBOXED CONTAINERS

You can access the metrics for OpenShift sandboxed containers in the Metrics page in the web console.

Prerequisites

- You have OpenShift Container Platform 4.11 installed.
- You have OpenShift sandboxed containers installed.
- You have access to the cluster as a user with the cluster-admin role or with view permissions for all projects.

Procedure

1. From the Administrator perspective in the web console, navigate to Observe → Metrics.
2. In the input field, enter the query for the metric you want to observe. All kata-related metrics begin with kata. Typing kata will display a list with all of the available kata metrics.

The metrics from your query are visualized on the page.

Additional resources

- For more information about creating PromQL queries to view metrics, see Querying metrics.

### 4.3. VIEWING THE OPENSHIFT SANDBOXED CONTAINERS DASHBOARD

You can access the OpenShift sandboxed containers dashboard in the Dashboards page in the web console.

**Prerequisites**

- You have OpenShift Container Platform 4.11 installed.
- You have OpenShift sandboxed containers installed.
- You have access to the cluster as a user with the cluster-admin role or with view permissions for all projects.

**Procedure**

1. From the Administrator perspective in the web console, navigate to Observe → Dashboards.

2. From the Dashboard drop-down list, select the Sandboxed Containers dashboard.

3. Optional: Select a time range for the graphs in the Time Range list.
   - Select a pre-defined time period.
   - Set a custom time range by selecting Custom time range in the Time Range list.
     - Define the date and time range for the data you want to view.
     - Click Save to save the custom time range.

4. Optional: Select a Refresh Interval.

The dashboard appears on the page with the following metrics from the Kata guest OS category:

**Number of running VMs**

Displays the total number of sandboxed containers running on your cluster.

**CPU Usage (per VM)**

Displays the CPU usage for each individual sandboxed container.

**Memory Usage (per VM)**

Displays the memory usage for each individual sandboxed container.

Hover over each of the graphs within a dashboard to display detailed information about specific items.
4.4. ADDITIONAL RESOURCES

- For more information about gathering data for support, see Gathering data about your cluster.
CHAPTER 5. UNINSTALLING OPENSHIFT SANDBOXED CONTAINERS

You can uninstall OpenShift sandboxed containers by using either the OpenShift Container Platform web console or OpenShift CLI (oc). Both procedures are explained below.

5.1. UNINSTALLING OPENSHIFT SANDBOXED CONTAINERS USING THE WEB CONSOLE

Use the OpenShift Container Platform web console to delete the relevant OpenShift sandboxed containers pods, resources, and namespace.

5.1.1. Deleting OpenShift sandboxed containers pods using the web console

To uninstall OpenShift sandboxed containers, you must first delete all running pods that use `kata` as the `runtimeClass`.

**Prerequisites**

- You have OpenShift Container Platform 4.11 installed on your cluster.
- You have access to the cluster as a user with the `cluster-admin` role.
- You have a list of the pods that use `kata` as the `runtimeClass`.

**Procedure**

1. From the Administrator perspective, navigate to **Workloads → Pods**.
2. Search for the pod that you want to delete using the **Search by name** field.
3. Click the pod name to open it.
4. On the **Details** page, check that `kata` is displayed for **Runtime class**.
5. Click the **Actions** menu and select **Delete Pod**.
6. Click **Delete** in the confirmation window.

**Additional resources**

You can retrieve a list of running pods that use `kata` as the `runtimeClass` from the OpenShift CLI. For details, see Deleting OpenShift sandboxed containers pods.

5.1.2. Deleting the KataConfig custom resource using the web console

Deleting the **KataConfig** custom resource (CR) removes and uninstalls the `kata` runtime and its related resources from your cluster.
IMPORTANT

Deleting the KataConfig CR automatically reboots the worker nodes. The reboot can take from 10 to more than 60 minutes. Factors that impede reboot time are as follows:

- A larger OpenShift Container Platform deployment with a greater number of worker nodes.
- Activation of the BIOS and Diagnostics utility.
- Deployment on a hard drive rather than on an SSD.
- Deployment on physical nodes such as bare metal, rather than on virtual nodes.
- A slow CPU or network.

Prerequisites

- You have OpenShift Container Platform 4.11 installed on your cluster.
- You have access to the cluster as a user with the cluster-admin role.
- You have no running pods that use kata as the runtimeClass.

Procedure

1. From the Administrator perspective, navigate to Operators → Installed Operators.
2. Search for the OpenShift sandboxed containers Operator using the Search by name field.
3. Click the Operator to open it, and then select the KataConfig tab.
4. Click the Options menu for the KataConfig resource, and then select Delete KataConfig.
5. Click Delete in the confirmation window.

Wait for the kata runtime and resources to uninstall and for the worker nodes to reboot before continuing to the next step.

5.1.3. Deleting the OpenShift sandboxed containers Operator using the web console

Deleting the OpenShift sandboxed containers Operator removes the catalog subscription, Operator group, and cluster service version (CSV) for that Operator.

Prerequisites

- You have OpenShift Container Platform 4.11 installed on your cluster.
- You have access to the cluster as a user with the cluster-admin role.

Procedure

1. From the Administrator perspective, navigate to Operators → Installed Operators.
2. Search for the OpenShift sandboxed containers Operator using the Search by name field.

3. Click the Options menu for the Operator and select Uninstall Operator.

4. Click Uninstall in the confirmation window.

5.1.4. Deleting the OpenShift sandboxed containers namespace using the web console

After you run the preceding commands, your cluster is restored to the state that it was prior to the installation process. You can now revoke namespace access to the Operator by deleting the openshift-sandboxed-containers-operator namespace.

Prerequisites

- You have OpenShift Container Platform 4.11 installed on your cluster.
- You have access to the cluster as a user with the cluster-admin role.

Procedure

1. From the Administrator perspective, navigate to Administration → Namespaces.

2. Search for the openshift-sandboxed-containers-operator namespace using the Search by name field.

3. Click the Options menu for the namespace and select Delete Namespace.

   NOTE

   If the Delete Namespace option is not available, you do not have permission to delete the namespace.

4. In the Delete Namespace pane, enter openshift-sandboxed-containers-operator and click Delete.

5. Click Delete.

5.1.5. Deleting the KataConfig custom resource definition using the web console

The KataConfig custom resource definition (CRD) lets you define the KataConfig CR. To complete the uninstall process, delete the KataConfig CRD from your cluster.

Prerequisites

- You have OpenShift Container Platform 4.11 installed on your cluster.
- You have access to the cluster as a user with the cluster-admin role.
- You have removed the KataConfig CR from your cluster.
You have removed the OpenShift sandboxed containers Operator from your cluster.

**Procedure**

1. From the Administrator perspective, navigate to Administration → CustomResourceDefinitions.
2. Search for KataConfig using the Search by name field.
3. Click the Options menu for the KataConfig CRD, and then select Delete CustomResourceDefinition.
4. Click Delete in the confirmation window.
5. Wait for the KataConfig CRD to disappear from the list. This can take several minutes.

### 5.2. UNINSTALLING OPENSHIFT SANDBOXED CONTAINERS USING THE CLI

You can uninstall OpenShift sandboxed containers by using the OpenShift Container Platform command-line interface (CLI). Follow the steps below in the order that they are presented.

#### 5.2.1. Deleting OpenShift sandboxed containers pods using the CLI

To uninstall OpenShift sandboxed containers, you must first delete all running pods that use kata as the runtimeClass.

**Prerequisites**

- You have installed the OpenShift CLI (oc).
- You have the command-line JSON processor (jq) installed.

**Procedure**

1. Search for pods that use kata as the runtimeClass by running the following command:
   ```bash
   $ oc get pods -A -o json | jq -r '.items[] | select(.spec.runtimeClassName == "kata").metadata.name'
   ```
2. To delete each pod, run the following command:
   ```bash
   $ oc delete pod <pod-name>
   ```

#### 5.2.2. Deleting the KataConfig custom resource using the CLI

Remove and uninstall the kata runtime and all its related resources, such as CRI-O config and RuntimeClass, from your cluster.

**Prerequisites**
You have OpenShift Container Platform 4.11 installed on your cluster.

You have installed the OpenShift CLI (oc).

You have access to the cluster as a user with the cluster-admin role.

IMPORTANT
Deleting the KataConfig CR automatically reboots the worker nodes. The reboot can take from 10 to more than 60 minutes. Factors that impede reboot time are as follows:

- A larger OpenShift Container Platform deployment with a greater number of worker nodes.
- Activation of the BIOS and Diagnostics utility.
- Deployment on a hard drive rather than on an SSD.
- Deployment on physical nodes such as bare metal, rather than on virtual nodes.
- A slow CPU or network.

Procedure
- Delete the KataConfig custom resource by running the following command:

  $ oc delete kataconfig <KataConfig_CR_Name>

The OpenShift sandboxed containers Operator removes all resources that were initially created to enable the runtime on your cluster.

IMPORTANT
During deletion, the CLI stops responding until all worker nodes reboot. Wait for the process to complete before performing the verification or continuing to the next procedure.

Verification
- To verify that the KataConfig custom resource is deleted, run the following command:

  $ oc get kataconfig <KataConfig_CR_Name>

Example output

No KataConfig instances exist

5.2.3. Deleting the OpenShift sandboxed containers Operator using the CLI

Remove the OpenShift sandboxed containers Operator from your cluster by deleting the Operator subscription, Operator group, cluster service version (CSV), and namespace.

Prerequisites
You have OpenShift Container Platform 4.10 installed on your cluster.

You have installed the OpenShift CLI (``oc``).

You have installed the command-line JSON processor (``jq``).

You have access to the cluster as a user with the cluster-admin role.

**Procedure**

1. Fetch the cluster service version (CSV) name for OpenShift sandboxed containers from the subscription by running the following command:

   ```
   CSV_NAME=$(oc get csv -n openshift-sandboxed-containers-operator -o=custom-columns=:metadata.name)
   ```

2. Delete the OpenShift sandboxed containers Operator subscription from Operator Lifecycle Manager (OLM) by running the following command:

   ```
   $ oc delete subscription sandboxed-containers-operator -n openshift-sandboxed-containers-operator
   ```

3. Delete the CSV name for OpenShift sandboxed containers by running the following command:

   ```
   $ oc delete csv ${CSV_NAME} -n openshift-sandboxed-containers-operator
   ```

4. Fetch the OpenShift sandboxed containers Operator group name by running the following command:

   ```
   $ OG_NAME=$(oc get operatorgroup -n openshift-sandboxed-containers-operator -o=jsonpath={..name})
   ```

5. Delete the OpenShift sandboxed containers Operator group name by running the following command:

   ```
   $ oc delete operatorgroup ${OG_NAME} -n openshift-sandboxed-containers-operator
   ```

6. Delete the OpenShift sandboxed containers namespace by running the following command:

   ```
   $ oc delete namespace openshift-sandboxed-containers-operator
   ```

5.2.4. Deleting the KataConfig custom resource definition using the CLI

The KataConfig custom resource definition (CRD) lets you define the KataConfig CR. Delete the KataConfig CRD from your cluster.

**Prerequisites**

- You have installed the OpenShift CLI (``oc``).
- You have access to the cluster as a user with the cluster-admin role.
- You have removed the KataConfig CR from your cluster.
You have removed the OpenShift sandboxed containers Operator from your cluster.

**Procedure**

- Delete the **KataConfig** CRD by running the following command:

  ```
  $ oc delete crd kataconfigs.kataconfiguration.openshift.io
  ```

**Verification**

- To verify that the **KataConfig** CRD is deleted, run the following command:

  ```
  $ oc get crd kataconfigs.kataconfiguration.openshift.io
  ```

**Example output**

```
Unknown CR KataConfig
```
CHAPTER 6. UPGRADING OPENSHIFT SANDBOXED CONTAINERS

The upgrade of the OpenShift sandboxed containers components consists of the following three steps:

- Upgrading OpenShift Container Platform to update the Kata runtime and its dependencies.
- Upgrading the OpenShift sandboxed containers Operator to update the Operator subscription.
- Manually patching the KataConfig custom resource (CR) to update the monitor pods.

You can upgrade OpenShift Container Platform before or after the OpenShift sandboxed containers Operator upgrade, with the one exception noted below. Always apply the KataConfig patch immediately after upgrading OpenShift sandboxed containers Operator.

IMPORTANT

If you are upgrading to OpenShift Container Platform 4.11 with OpenShift sandboxed containers 1.3, the recommended order is to first upgrade OpenShift sandboxed containers from 1.2 to 1.3, and then upgrade OpenShift Container Platform from 4.10 to 4.11.

6.1. UPGRADING THE OPENSHIFT SANDBOXED CONTAINERS RESOURCES

The OpenShift sandboxed containers resources are deployed onto the cluster using Red Hat Enterprise Linux CoreOS (RHCOS) extensions.

The RHCOS extension sandboxed containers contains the required components to run Kata Containers such as the Kata containers runtime, the hypervisor QEMU, and other dependencies. You upgrade the extension by upgrading the cluster to a new release of OpenShift Container Platform.

For more information about upgrading OpenShift Container Platform, see Updating Clusters.

6.2. UPGRADING THE OPENSHIFT SANDBOXED CONTAINERS OPERATOR

Use Operator Lifecycle Manager (OLM) to upgrade the OpenShift sandboxed containers Operator either manually or automatically. Selecting between manual or automatic upgrade during the initial deployment determines the future upgrade mode. For manual upgrades, the web console shows the available updates that can be installed by the cluster administrator.

For more information about upgrading the OpenShift sandboxed containers Operator in Operator Lifecycle Manager (OLM), see Updating installed Operators.

6.3. UPGRADING THE OPENSHIFT SANDBOXED CONTAINERS MONITOR PODS

After upgrading OpenShift sandboxed containers, you need to update the monitor image in the KataConfig CR to upgrade the monitor pods. Otherwise, the monitor pods will continue running images from the previous version.

You can perform the update using the web console or the CLI.
6.3.1. Upgrading the monitor pods using the web console

The KataConfig YAML file in the OpenShift Container Platform contains the version number for the monitor image. Update the version number with the correct version.

Prerequisites

- You have OpenShift Container Platform 4.11 installed on your cluster.
- You have access to the cluster as a user with the cluster-admin role.

Procedure

1. From the Administrator perspective of OpenShift Container Platform, navigate to Operators → Installed Operators.
2. Select the OpenShift sandboxed containers Operator and go to the KataConfig tab.
3. Search for the KataConfig resource using the Search by name field. The default name for the KataConfig resource is example-kataconfig.
4. Select the KataConfig resource and go to the KataConfig tab.
5. Modify the version number for kataMonitorImage:

   ```yaml
   checkNodeEligibility: false
   kataConfigPoolSelector: null
   kataMonitorImage: 'registry.redhat.io/openshift-sandboxed-containers/osc-monitor-rhel8:1.3.0'
   ```

6. Click Save.

6.3.2. Upgrading the monitor pods using the CLI

You can manually patch the monitor image in the KataConfig CR to update the monitor pods.

Prerequisites

- You have OpenShift Container Platform 4.11 installed on your cluster.
- You have installed the OpenShift CLI (oc).
- You have access to the cluster as a user with the cluster-admin role.

Procedure

- In the OpenShift Container Platform CLI, run the following command:

  ```
  $ oc patch kataconfig <kataconfig_name> --type merge --patch '{"spec":{"kataMonitorImage":"registry.redhat.io/openshift-sandboxed-containers/osc-monitor-rhel8:1.3.0"}}'
  ```

  where: `<kataconfig_name>`: specifies the name of your Kata configuration file, such as example-kataconfig.
CHAPTER 7. COLLECTING OPENSOURCE SANDBOXED CONTAINERS DATA

When troubleshooting OpenShift sandboxed containers, you can open a support case and provide debugging information using the **must-gather** tool.

If you are a cluster administrator, you can also review logs on your own, enabling a more detailed level of logs.

### 7.1. COLLECTING OPENSOURCE SANDBOXED CONTAINERS DATA FOR RED HAT SUPPORT

When opening a support case, it is helpful to provide debugging information about your cluster to Red Hat Support.

The **must-gather** tool enables you to collect diagnostic information about your OpenShift Container Platform cluster, including virtual machines and other data related to OpenShift sandboxed containers.

For prompt support, supply diagnostic information for both OpenShift Container Platform and OpenShift sandboxed containers.

#### 7.1.1. About the must-gather tool

The **oc adm must-gather** CLI command collects the information from your cluster that is most likely needed for debugging issues, including:

- Resource definitions
- Service logs

By default, the **oc adm must-gather** command uses the default plugin image and writes into .*/must-gather.local.

Alternatively, you can collect specific information by running the command with the appropriate arguments as described in the following sections:

- To collect data related to one or more specific features, use the **--image** argument with an image, as listed in a following section.
  
  For example:

  ```bash
  $ oc adm must-gather  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.11.0
  ```

- To collect the audit logs, use the **--/usr/bin/gather_audit_logs** argument, as described in a following section.
  
  For example:

  ```bash
  $ oc adm must-gather -- /usr/bin/gather_audit_logs
  ```

**NOTE**

Audit logs are not collected as part of the default set of information to reduce the size of the files.
When you run `oc adm must-gather`, a new pod with a random name is created in a new project on the cluster. The data is collected on that pod and saved in a new directory that starts with `must-gather.local`. This directory is created in the current working directory.

For example:

```
<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Running</td>
<td></td>
<td></td>
</tr>
<tr>
<td>openshift-must-gather-5drcj</td>
<td>must-gather-bklx4</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>72s</td>
</tr>
<tr>
<td>openshift-must-gather-5drcj</td>
<td>must-gather-s8sdh</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>72s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

To collect OpenShift sandboxed containers data with `must-gather`, you must specify the OpenShift sandboxed containers image:

```
--image=registry.redhat.io/openshift-sandboxed-containers/osc-must-gather-rhel8:1.3.0
```

### 7.2. ABOUT OPENSHIFT SANDBOXED CONTAINERS LOG DATA

When you collect log data about your cluster, the following features and objects are associated with OpenShift sandboxed containers:

- All namespaces and their child objects that belong to any OpenShift sandboxed containers resources
- All OpenShift sandboxed containers custom resource definitions (CRDs)

The following OpenShift sandboxed containers component logs are collected for each pod running with the `kata` runtime:

- Kata agent logs
- Kata runtime logs
- QEMU logs
- Audit logs
- CRI-O logs

### 7.3. ENABLING DEBUG LOGS FOR OPENSHIFT SANDBOXED CONTAINERS

As a cluster administrator, you can collect a more detailed level of logs for OpenShift sandboxed containers. You can also enhance logging by changing the `logLevel` field in the `KataConfig` CR. This changes the `log_level` in the CRI-O runtime for the worker nodes running OpenShift sandboxed containers.

#### Procedure

1. Change the `logLevel` field in your existing `KataConfig` CR to `debug`:

```
$ oc patch kataconfig <name_of_kataconfig_file> --type merge --patch '{"spec":{"logLevel":"debug"}}'
```
NOTE

When running this command, reference the name of your KataConfig CR. This is the name you used to create the CR when setting up OpenShift sandboxed containers.

Verification

1. Monitor the kata-oc machine config pool until the UPDATED field appears as True, meaning all worker nodes are updated:

   $ oc get mcp kata-oc

   **Example output**

   ```
   NAME     CONFIG                 UPDATED  UPDATING  DEGRADED  MACHINECOUNT
   READYMACHINECOUNT  UPDATEDMACHINECOUNT  DEGRADEDMACHINECOUNT
   AGE
   kata-oc  rendered-kata-oc-169   False    True      False     3             1                  1
   0                     9h
   ```

2. Verify that the log_level was updated in CRI-O:

   a. Open an oc debug session to a node in the machine config pool and run chroot /host

   ```
   $ oc debug node/<node_name>
   sh-4.4# chroot /host
   ```

   b. Verify the changes in the crio.conf file:

   ```
   sh-4.4# crio config | egrep 'log_level
   ```

   **Example output**

   ```
   log_level = "debug"
   ```

7.3.1. Viewing debug logs for OpenShift sandboxed containers

Cluster administrators can use the enhanced debug logs for OpenShift sandboxed containers to troubleshoot issues. The logs for each node are printed to the node journal.

You can review the logs for the following OpenShift sandboxed containers components:

- Kata agent
- Kata runtime (containerd-shim-kata-v2)
- virtiofsd

Logs for QEMU do not print to the node journal. However, a QEMU failure is reported to the runtime, and the console of the QEMU guest is printed to the node journal. You can view these logs together with the Kata agent logs.
Prerequisites

- You have installed the OpenShift CLI (`oc`).
- You have access to the cluster as a user with the `cluster-admin` role.

Procedure

- To review the Kata agent logs and guest console logs, run:

  ```
  $ oc debug node/<nodename> -- journalctl -D /host/var/log/journal -t kata -g "reading guest console"
  ```

- To review the kata runtime logs, run:

  ```
  $ oc debug node/<nodename> -- journalctl -D /host/var/log/journal -t kata
  ```

- To review the virtiofsd logs, run:

  ```
  $ oc debug node/<nodename> -- journalctl -D /host/var/log/journal -t virtiofsd
  ```

7.4. ADDITIONAL RESOURCES

- For more information about gathering data for support, see [Gathering data about your cluster](#).