OpenShift Container Platform 4.9

Sandboxed Containers Support for OpenShift

OpenShift sandboxed containers guide

Last Updated: 2023-03-27

OpenShift Container Platform 4.9 Sandboxed Containers Support for OpenShift

OpenShift sandboxed containers guide

Legal Notice

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. {SANDBOXED-CONTAINERS-FIRST} 1.1 RELEASE NOTES

1.1. ABOUT THIS RELEASE

These release notes track the development of OpenShift sandboxed containers 1.1 alongside Red Hat OpenShift Container Platform 4.9.

This product is currently in Technology Preview. OpenShift sandboxed containers is not intended for production use. For more information, see the Red Hat Customer Portal support scope for features in Technology Preview.

1.2. NEW FEATURES AND ENHANCEMENTS

1.2.1. FIPS compatibility

FIPS mode is now automatically enabled for OpenShift sandboxed containers. OpenShift sandboxed containers deployed on an OpenShift Container Platform cluster installed in FIPS mode will not taint the cluster's FIPS support. For more information, see Understanding compliance and risk management .

1.2.2. Collect resources with must-gather

The OpenShift sandboxed containers Operator now includes a must-gather image, allowing you to collect custom resources and log files specific to this Operator and the underlying runtime components for diagnostic purposes. For more information, see Collecting OpenShift sandboxed containers data for Red Hat Support.

1.2.3. Disconnected environments

You can now install the OpenShift sandboxed containers Operator in a disconnected environment. For more information, see the Additional resources for Deploying OpenShift sandboxed containers workloads.

1.3. BUG FIXES

- Previously, when running Fedora on OpenShift sandboxed containers, some packages required file access permission changes that OpenShift Container Platform did not grant to containers by default. With this release, these permissions are granted by default. (BZ#1915377)
- Previously, adding a value to **kataConfgPoolSelector** in the OpenShift Container Platform web console populated **scheduling.nodeSelector** with an empty value. As a result, pods that used a **RuntimeClass** object with the value of **kata** could be scheduled to nodes without the Kata Containers runtime installed. With this release, only nodes labeled with the same label as defined in **kataConfgPoolSelector** will install the Kata Containers runtime. (BZ#2019384)
- Previously, the OpenShift sandboxed containers Operator details page on Operator Hub was missing fields. In this release, these fields are no longer missing. (BZ#2019383)
- Previously, creating multiple **KataConfig** custom resources resulted in a silent failure, with no error from the OpenShift Container Platform web console notifying the user that creating more than one custom resource failed. With this release, the user receives an error when trying to create multiple custom resources. (BZ#2019381)

- Previously, there were instances where the Operator Hub in the OpenShift Container Platform web console did not display icons for an Operator. With this release, icons are always displayed. (BZ#9019380)

1.4. KNOWN ISSUES

- If you are using OpenShift sandboxed containers, you might receive SELinux denials 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, since 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, and provides stronger protection against potential security flaws in **virtiofsd** 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 in order to allow virtiofsd to access these specific labels. (BZ#1904609)

1.5. ASYNCHRONOUS ERRATA UPDATES

Security, bug fix, and enhancement updates for OpenShift sandboxed containers 4.9 are released as asynchronous errata through the Red Hat Network. All OpenShift Container Platform 4.9 errata is available on the Red Hat Customer Portal . See the OpenShift Container Platform Life Cycle for more information about asynchronous errata.

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 via 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 1.1.0.

1.5.1. RHEA-2021:3941 - OpenShift sandboxed containers 1.1.0 image release, bug fix,and enhancement advisory

Issued: 2021-10-21

OpenShift sandboxed containers release 1.1.0 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 RHEA-2021:3941 advisory.

CHAPTER 2. UNDERSTANDING OPENSHIFT SANDBOXED CONTAINERS

OpenShift sandboxed containers support for OpenShift Container Platform provides users with built-in support for running Kata Containers as an additional optional runtime. This is particularly useful for users who are wanting to perform the following tasks:

- Run privileged or untrusted workloads.
- Ensure kernel isolation for each workload.
- Share the same workload across tenants.
- Ensure proper isolation and sandboxing for testing software.
- Ensure default resource containment through VM boundaries.

OpenShift sandboxed containers also provides users the ability to choose from the type of workload that they want to run to cover a wide variety of use cases.

You can use the OpenShift sandboxed containers Operator to perform tasks such as installation and removal, updates, and status monitoring.

Sandboxed containers are only supported on bare metal.

Red Hat Enterprise Linux CoreOS (RHCOS) is the only supported operating system for OpenShift sandboxed containers 1.0.0.

2.1. 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. It deals with operations, such as the installation and removal of sandboxed containers software 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.

RHCOS extensions

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.

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.

2.2. 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. RHCOS EXTENSIONS

The OpenShift sandboxed containers Operator is based on the Red Hat Enterprise Linux CoreOS (RHCOS) extensions concept. 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

2.4. UNDERSTANDING COMPLIANCE AND RISK MANAGEMENT

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

The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture.

To understand Red Hat's view of OpenShift Container Platform compliance frameworks, refer to the Risk Management and Regulatory Readiness chapter of the OpenShift Security Guide Book .

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 bare-metal infrastructure, on premise with Red Hat Enterprise Linux CoreOS (RHCOS) workers.

IMPORTANT

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

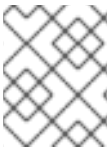
Default memory sizes

The following table lists some the default values for resource allocation.

Resource	Value
Memory allocated by default to a virtual machine	2Gi
Guest Linux kernel memory usage at boot	~110Mi
Memory used by the QEMU process (excluding VM memory)	~30Mi
Memory used by the **virtiofsd** process (excluding VM I/O buffers)	~10Mi
Memory used by the **containerd-shim-kata-v2** process	~20Mi
File buffer cache data after running **dnf install** on Fedora	~300Mi* [1]

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

3.2. DEPLOYING OPENSHIFT 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.

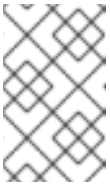
Prerequisites

- You have OpenShift Container Platform 4.9 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 **preview-1.1** 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

Attempting 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

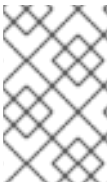
1. From the **Administrator** perspective in the web console, navigate to **Operators → Installed Operators**.
2. Verify that the OpenShift sandboxed containers Operator is listed in the in operators list.

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.

Prerequisites

- You have installed OpenShift Container Platform 4.9 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**:

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
```

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

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
spec:
  kataConfigPoolSelector:
    matchLabels:
      <label_key>: '<label_value>' (1)
```

(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.

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 4.9 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

```
apiVersion: v1
kind: Pod
metadata:
  name: example
  labels:
    app: httpd
  namespace: openshift-sandboxed-containers-operator
spec:
  runtimeClassName: kata
  containers:
    - name: httpd
      image: 'image-registry.openshift-image-registry.svc:5000/openshift/httpd:latest'
      ports:
        - containerPort: 8080
```

Example for Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example
  namespace: openshift-sandboxed-containers-operator
spec:
  selector:
    matchLabels:
      app: httpd
  replicas: 3
  template:
    metadata:
      labels:
        app: httpd
    spec:
      runtimeClassName: kata
      containers:
```

```
      - name: httpd
        image: >-
          image-registry.openshift-image-registry.svc:5000/openshift/httpd:latest
        ports:
          - containerPort: 8080
```

4. 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 4.9 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:

```
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: "preview-1.1"
  installPlanApproval: Automatic
  name: sandboxed-containers-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: sandboxed-containers-operator.v1.1.0
```

   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

```
NAME                         DISPLAY                                VERSION  REPLACES     PHASE
openshift-sandboxed-containers   openshift-sandboxed-containers-operator  1.1.0    1.0.2
Succeeded
```

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 4.9 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.

Procedure

1. Create a YAML file with the following manifest:

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
```

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:

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
```

```
  name: cluster-kataconfig
spec:
  kataConfigPoolSelector:
    matchLabels:
      <label_key>: '<label_value>' 1
```

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

3. Create the **KataConfig** resource:

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

The new **KataConfig** CR is created and begins to install **kata** as a **RuntimeClass** on the worker nodes.

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:

```
$ 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 4.9 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 objects

```
apiVersion: v1
kind: Pod
metadata:
 name: mypod
spec:
  runtimeClassName: kata
```

Example for Deployment objects

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mypod
  labels:
    app: mypod
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mypod
  template:
    metadata:
      labels:
        app: mypod
    spec:
      runtimeClassName: kata
      containers:
      - name: mypod
        image: myImage
```

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. UNINSTALLING OPENSHIFT SANDBOXED CONTAINERS

4.1. UNINSTALLING OPENSHIFT SANDBOXED CONTAINERS USING THE WEB CONSOLE

You can uninstall OpenShift sandboxed containers by using the OpenShift Container Platform web console.

4.1.1. Deleting OpenShift sandboxed containers resources

To uninstall OpenShift sandboxed containers, you must first delete the OpenShift sandboxed containers custom resource **KataConfig**. This removes and uninstalls the **kata** runtime and its related resources from your cluster.

Prerequisites

- You have OpenShift Container Platform 4.9 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 **runtimeClassName**.
    - You have installed the OpenShift CLI (**oc**).
    - You the command-line JSON processor (**jq**) installed.
    - Verify that you have no running pods that use **kata** as the **runtimeClassName** by running the following command:

```
$ oc get pods -A -o json | jq -r '.items[] | select(.spec.runtimeClassName |
test("kata")).metadata.name'
```

Procedure

1. Delete all pods that use **runtimeClassName** with the value of **kata**.
2. From the OpenShift Container Platform web console, select **openshift-sandboxed-containers** from the **Projects** list.
3. Navigate to the **Operators → Installed Operators** page.
4. Click **OpenShift sandboxed containers**.
5. Click the **OpenShift sandboxed containers Operator** tab.
6. Click the scroll-down list in the **Operator Details**, and then click **Delete KataConfig**.
7. Click **Delete** in the confirmation window.

4.1.1.1. Deleting a namespace using the web console

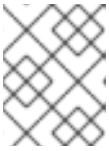
You can delete a namespace by using the OpenShift Container Platform web console.

Prerequisites

- You have OpenShift Container Platform 4.9 installed on your cluster.
- You have access to the cluster as a user with the **cluster-admin** role.

Procedure

1. Navigate to **Administration → Namespaces**.
2. Locate the **openshift-sandboxed-containers-operator** namespace to delete in the list of namespaces.
3. On the rightmost side of the namespace listing, select **Delete Namespace** from the **Options** menu .
4. When the **Delete Namespace** pane opens, enter **openshift-sandboxed-containers-operator** in the field.



NOTE

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

5. Click **Delete**.

4.1.2. Deleting OpenShift sandboxed containers Operator

You can delete the OpenShift sandboxed containers Operator by deleting the catalog subscription and revoking namespace access to the Operator.

Prerequisites

- You have OpenShift Container Platform 4.9 installed on your cluster.
- You have access to the cluster as a user with the **cluster-admin** role.

Procedure

1. Navigate to the **Operators → OperatorHub** page.
2. Search for **OpenShift sandboxed containers** and then select the Operator.
3. Click **Uninstall**.
4. Delete the **openshift-sandboxed-containers-operator** namespace.

4.2. UNINSTALLING KATA RUNTIME FROM THE CLI

You can uninstall OpenShift sandboxed containers by using the OpenShift Container Platform command-line interface (CLI).

4.2.1. Deleting OpenShift sandboxed containers resources

You can 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 4.9 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

1. Delete the **KataConfig** custom resource by running the following command:

```
$ oc delete kataconfig <KataConfig_CR_Name>
```

2. Delete the **KataConfig** custom resource definition by running the following command:

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

The OpenShift sandboxed containers Operator removes all resources that were initially created to enable the runtime on your cluster. After you run the preceding commands, your cluster is restored to the state that it was prior to the installation process. You can now delete the **openshift-sandboxed-containers-operator** namespace.

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
```

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

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

Example output

```
Unknown CR KataConfig
```

4.2.2. Deleting OpenShift sandboxed containers Operator

You can delete the OpenShift sandboxed containers Operator from your cluster.

Prerequisites

- You have OpenShift Container Platform 4.9 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

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

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

2. Set the cluster service version (CSV) name for OpenShift sandboxed containers as an environment variable by running the following command:

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

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

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

CHAPTER 5. UPGRADE OPENSHIFT SANDBOXED CONTAINERS

You can upgrade the components of OpenShift sandboxed containers by upgrading OpenShift sandboxed containers Operator and OpenShift sandboxed containers artifacts.

5.1. UPGRADE OPENSHIFT SANDBOXED CONTAINERS OPERATOR

You can use Operator Lifecycle Manager (OLM) to manually or automatically upgrade the OpenShift sandboxed containers Operator. You can select manual or automatic upgrade during the initial deployment. In the context of manual upgrades, the web console shows the available updates that can be installed by the cluster administrator.

Additional resources

- Upgrading installed Operators

5.2. UPGRADE THE OPENSHIFT SANDBOXED CONTAINERS ARTIFACTS

The OpenShift sandboxed containers artifacts 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. The extension is upgraded when upgrading the cluster to a new release of OpenShift Container Platform.

CHAPTER 6. COLLECTING OPENSHIFT 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.

6.1. 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:

  ```
  $ oc adm must-gather  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.9.0
  ```

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

  ```
  $ 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:

```
NAMESPACE                  NAME            READY   STATUS      RESTARTS      AGE
...
```

```
openshift-must-gather-5drcj    must-gather-bklx4    2/2     Running     0          72s
openshift-must-gather-5drcj    must-gather-s8sdh    2/2     Running     0          72s
...
```

6.2. ABOUT COLLECTING OPENSHIFT SANDBOXED CONTAINERS DATA

You can use the **oc adm must-gather** CLI command to collect information 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 **oc adm must-gather** CLI command collects the following component logs:

- QEMU logs
- Audit logs
- OpenShift sandboxed containers logs
- CRI-O logs

These component logs are collected as long as there is at least one pod running with the **kata** runtime.

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

```
--image=registry.redhat.io/openshift-sandboxed-containers-tech-preview/osc-must-gather-rhel8:1.1.0
```

6.3. ADDITIONAL RESOURCES

- For more information about gathering data for support, see Gathering data about your cluster .