OpenShift Container Platform 4.8

Sandboxed Containers Support for OpenShift

OpenShift sandboxed containers guide

Last Updated: 2023-02-28

OpenShift Container Platform 4.8 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. {SANDBOXED-CONTAINERS-FIRST} 1.0 RELEASE NOTES

1.1. ABOUT THIS RELEASE

These release notes track the development of OpenShift sandboxed containers in Red Hat OpenShift Container Platform.

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. OpenShift sandboxed containers support on OpenShift Container Platform (Technology Preview)

OpenShift sandboxed containers 1.0.0 Technology Preview release introduces built-in support for running Kata Containers as an additional runtime. OpenShift sandboxed containers enables users to choose Kata Containers as an additional runtime to provide additional isolation for their workloads. The OpenShift sandboxed containers Operator automates the tasks of installing, removing, and updating Kata Containers. It allows for tracking the state of those tasks by describing the **KataConfig** custom resource.

OpenShift 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. Disconnected environments are not supported in OpenShift Container Platform 4.8.

For more information, see Understanding OpenShift sandboxed containers

1.3. KNOWN ISSUES

- If you are using OpenShift sandboxed containers, you cannot use the **hostPath** volume in a OpenShift Container Platform cluster to mount a file or directory from the host node's file system into your pod. As an alternative, you can use local persistent volumes. See Persistent storage using local volumes for more information. (BZ#1904609)

- If you are running Fedora on OpenShift sandboxed containers, you need a workaround to install some packages. Some packages, like **iputils**, require file access permission changes that OpenShift Container Platform does not grant to containers by default. To run containers that require such special permissions, it is necessary to add an annotation to the YAML file describing the workload, which tells **virtiofsd** to accept such file permissions for that workload. The required annotations are:

  ```
  io.katacontainers.config.hypervisor.virtio_fs_extra_args: |
    [ "-o", "modcaps=+sys_admin", "-o", "xattr" ]
  ```

  (BZ#1915377)

- In the 4.8 release, adding a value to **kataConfgPoolSelector** by using the OpenShift Container Platform web console causes **scheduling.nodeSelector** to be populated with an empty value. Pods that use **RuntimeClass** with the value of **kata** might be scheduled to nodes that do not have the Kata Containers runtime installed.

To work around this issue, specify the **nodeSelector** value manually in the **RuntimeClass kata** by running the following command:

```
$ oc edit runtimeclass kata
```

The following is an example of a **RuntimeClass** with the correct **nodeSelector** statement.

```
apiVersion: node.k8s.io/v1
handler: kata
kind: RuntimeClass
metadata:
  creationTimestamp: "2021-06-14T12:54:19Z"
  name: kata
overhead:
  podFixed:
    cpu: 250m
    memory: 350Mi
scheduling:
  nodeSelector:
    custom-kata-pool: "true"
```

(BZ#2019384)

- The OpenShift sandboxed containers Operator details page on Operator Hub contains a few missing fields. The missing fields do not prevent you from installing the OpenShift sandboxed containers Operator in 4.8. (BZ#2019383)
- Creating multiple **KataConfig** custom resources results in a silent failure. The OpenShift Container Platform web console does not provide a prompt to notify the user that creating more than one custom resource has failed. (BZ#2019381)
- Sometimes the Operator Hub in the OpenShift Container Platform web console does not display icons for an Operator. (BZ#2019380)

1.4. ASYNCHRONOUS ERRATA UPDATES

Security, bug fix, and enhancement updates for OpenShift sandboxed containers 1.0 are released as asynchronous errata through the Red Hat Network. All OpenShift Container Platform 4.8 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 1.0.0.

1.4.1. RHBA-2021:3751 - OpenShift sandboxed containers 1.0.2 bug fix advisory

Issued: 2021-10-07

OpenShift sandboxed containers release 1.0.2 is now available. This advisory contains an update for OpenShift sandboxed containers with bug fixes.

The list of bug fixes that are included in the update is documented in the RHBA-2021:3751 advisory.

1.4.2. RHBA-2021:3552 - OpenShift sandboxed containers 1.0.1 bug fix advisory

Issued: 2021-09-16

OpenShift sandboxed containers release 1.0.1 is now available. This advisory contains an update for OpenShift sandboxed containers with bug fixes.

The list of bug fixes that are included in the update is documented in the RHBA-2021:3552 advisory.

1.4.3. RHEA-2021:2546 - OpenShift sandboxed containers 1.0.0 image release, bug fix, and enhancement advisory

Issued: 2021-07-29

The components for OpenShift sandboxed containers release 1.0.0 support for OpenShift Container Platform 4.8 are now available as a technology preview.

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

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. PREPARING YOUR CLUSTER FOR OPENSHIFT SANDBOXED CONTAINERS

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 with Red Hat Enterprise Linux CoreOS (RHCOS) workers. Your cluster must use installer-provisioned infrastructure.

  IMPORTANT

  - OpenShift sandboxed containers only supports RHCOS worker nodes. RHEL 7 or RHEL 8 nodes are not supported.
  - Nested virtualization is not supported.

3.1.1. Additional resource requirements for OpenShift sandboxed containers

OpenShift sandboxed containers is a product that brings the ability to run workloads inside a sandboxed runtime, such as Kata Containers, to your OpenShift Container Platform clusters. 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 processes that are running in those containers. There are two additional processes that 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.

- If a container runs without a given memory resource, it is able to consume free memory. It will do so 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.
- If a memory limit is specified, then the workload is terminated if it consumes more memory than the limit. If no memory limit is specified, the kernel that is running on the virtual machine might run out of memory. If the kernel runs out of memory it might terminate other processes on the virtual machine.

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]

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 class by using **oc describe runtimeclass kata** as shown below.

Example

```
$ oc describe runtimeclass kata
```

Example output

```
Name:         kata
[...]
Metadata:
[...]
Overhead:
  Pod Fixed:
    Cpu:     250m
    Memory:  350Mi
[...]
```

You can change the pod overhead by changing the **spec.overhead** field for a **RuntimeClass**. For instance, 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.

Example

```
kind: RuntimeClass
apiVersion: node.k8s.io/v1
metadata:
  name: kata
overhead:
  podFixed:
    memory: "500Mi"
    cpu: "500m"
```

- The default allocation for virtual machines is 2Gi.
- The Linux kernel uses approximately 100Mi of memory at boot time.
- The QEMU process uses approximately 30Mi of memory.
- The **virtiofsd** process uses approximately 10Mi of memory.
- The **shim-v2** process uses approximately 20Mi of memory.

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 on 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

- Deploying installer-provisioned clusters on bare metal

3.2. DEPLOYING OPENSHIFT SANDBOXED CONTAINERS OPERATOR USING THE WEB CONSOLE

You can install the Operator and view your workloads from the web console.

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

Procedure

1. Open a browser window and log in to the OpenShift Container Platform web console.
2. From the **Administrator** perspective, navigate to **Operators** → **OperatorHub**.
3. In the **Filter by keyword** field, type **OpenShift sandboxed containers**.
4. Select the **OpenShift sandboxed containers** tile.
5. Read the information about the Operator and click **Install**.
6. On the **Install Operator** page:
   a. Select **preview-1.0** from the list of available **Update Channel** options. This ensures that you install the version of OpenShift sandboxed containers that is compatible with your OpenShift Container Platform version.
   b. For **Installed Namespace**, ensure that the Operator recommended namespace option is selected. This installs the Operator in the mandatory **openshift-sandboxed-containers-operator** namespace, which is automatically created if it does not exist.

      NOTE

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

   c. For **Approval Strategy**, ensure that **Automatic**, which is the default value, is selected. OpenShift sandboxed containers automatically updates when a new z-stream release is available.
7. Click **Install** to make the Operator available to the OpenShift sandboxed containers namespace.

The OpenShift sandboxed containers Operator is now installed on your cluster. You can trigger the Operator by enabling the runtime on your cluster. You can do this by creating the **KataConfig** custom resource using the OpenShift CLI (**oc**).

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

3.2.2. Viewing OpenShift sandboxed containers workloads from the web console

OpenShift sandboxed containers based workloads look and feel the same as normal workloads when viewed in the web console. The only difference between the two is the **runtimeClassName**. **runtimeClassName** is what decides the runtime used for workloads. In this context, the runtime enabled by OpenShift sandboxed containers-based is **kata**. You can view the **runtimeClass** that the pods for your workloads use.

Prerequisites

- You have OpenShift Container Platform 4.8 installed on your cluster.
- You have access to the cluster as a user with the **cluster-admin** role.

Procedure

1. Navigate to **Administration** → **Workloads**.
2. Identify the type of workload you want to view details for. For example, **Pod**, **Deployment**, **DeploymentConfigs** objects and so on.
3. Choose the corresponding workload from the list.
4. On the **Details** page, navigate to **runtimeClass**.
5. Hover over **runtimeClass** to view more information. If **kata** was used as the runtime, the value of the **runtimeClass** is **kata**.

3.3. DEPLOYING OPENSHIFT SANDBOXED CONTAINERS OPERATOR USING THE CLI

You can install and deploy the Operator and view workloads from the CLI.

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 4.8 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 a YAML file that contains the following manifest:

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

```
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-sandboxed-containers-kataconfig-group
  namespace: openshift-sandboxed-containers-operator
spec:
  targetNamespaces:
    - openshift-sandboxed-containers-operator
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: sandboxed-containers-operatorhub
  namespace: openshift-sandboxed-containers-operator
spec:
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  name: sandboxed-containers-operator
  startingCSV: sandboxed-containers-operator.v1.0.0
  channel: "preview-1.0"
  approval: "Automatic"
```

NOTE

Using the **preview-1.0** channel ensures that you install the version of OpenShift sandboxed containers that is compatible with your OpenShift Container Platform version.

2. Create the required **Namespace**, **OperatorGroup**, and **Subscription** objects for OpenShift sandboxed containers:

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

3. 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  1.0.0    <csv-of-
previous-version>   Succeeded
```

4. View the available deployments:

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

Example output

```
NAME                                   READY  UP-TO-DATE   AVAILABLE   AGE
openshift-sandboxed-containers-operator                     1/111       9m48s
```

–

Verification

- Verify that the Operator is up and running, so you can create the **KataConfig** resource to trigger the installation.

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

  Example output

  ```
  NAME                                              READY   UP-TO-DATE   AVAILABLE   AGE
  openshift-sandboxed-containers-controller-manager   1/1     1            1           40d
  ```

Additional resources

- Installing from OperatorHub using the CLI

3.3.1.1. Triggering the installation of the Kata runtime

You must create one **KataConfig** custom resource (CR) to trigger 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 runtime, CRI-O, is configured with the correct Kata runtime handlers.
- Create a **RuntimeClass** custom resource with necessary configurations for additional overhead caused by virtualization and the required additional processes.

Prerequisites

- You have OpenShift Container Platform 4.8 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. Create the **KataConfig** resource:

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

   Example

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

2. Monitor the installation progress.

- You can describe the **KataConfig** installation:

  ```
  $ oc describe kataconfig
  ```

  - Verify the **Completed nodes** field in the status.
  - If the value of **Completed nodes** matches the number of worker nodes, then the installation is completed. The status also contains a list of nodes where the installation is completed.

- You can check the progress of the installation by watching the **KataConfig** resource:

  ```
  $ watch -n 10 oc describe kataconfig
  ```

  Alternatively, you can check the status of the **KataConfig** resource. This can be done by running **oc get KataConfig <name> -oyaml** and inspecting the **status** field in the output.

The Kata runtime is now installed on the cluster and ready for use as a secondary runtime. Verify that you see a newly created **RuntimeClass** for Kata on your cluster.

IMPORTANT

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

Verification

- You can monitor the values of the **KataConfig** custom resource by running:

  ```
  $ watch oc describe KataConfig cluster-kataconfig
  ```

Additional resources

- RuntimeClass

3.3.1.2. Selecting nodes for OpenShift sandboxed containers

You can selectively install the Kata runtime on specific workers.

Prerequisites

- You have OpenShift Container Platform 4.8 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. Identify the labels that you want to use for selecting your nodes. For this example, use labels to selects to be chosen as candidates to run on your OpenShift sandboxed containers workloads. If the nodes exist, they are selected.

   a. To apply a label to a node, run the following command:

```
$ oc label node <worker_node_name> <label>=<value>
```

This labels your worker node with the **<label>** label that has a value of **<value>**.

2. To add a label selector, edit the **KataConfig** custom resource (CR):

```
$ oc edit kataconfig
```

Example

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
spec:
  kataConfigPoolSelector:
    matchLabels:
       custom-kata-machine-pool: 'true'
```

Verification

- You can check to see if the nodes in the **machine-config-pool** object are going through a config update.
  - If you are using the default nodes, you can monitor the **machine-config-pool** resource by running:

    ```
    $ watch oc get mcp worker
    ```

  - If you are using selected nodes, you can monitor the **machine-config-pool** resource by running:

    ```
    $ watch oc get mcp kata-oc
    ```

- You can run **watch oc describe kataconfig cluster-kataconfig** to display information about **sandboxed-containers** extension failure on a node. The information is gathered from the status of the **machine-config-pool** object. You can view the information by running:

  ```
  $ oc describe mcp <machine-config-pool>
  ```

3.3.1.3. Scheduling OpenShift sandboxed containers workloads

You can schedule your workloads to run on OpenShift sandboxed containers.

Prerequisites

- You have OpenShift Container Platform 4.8 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. Add **runtimeClassName: kata** to any pod-templated resources:
   - **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
```

After the pod-templated resource is created with **runtimeClassName: kata**, OpenShift Container Platform begins scheduling the workload on OpenShift sandboxed containers enabled nodes. If no selector is used, the default is set to all worker nodes. Your workload runs on OpenShift sandboxed containers.

3.3.2. Viewing OpenShift sandboxed containers workloads from the CLI

You can view the **runtimeClass** that the pods for your workloads use from the CLI.

Prerequisites

- You have OpenShift Container Platform 4.8 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

- Inspect the **runtimeClassName** field on the pod to see a pod running on OpenShift sandboxed containers versus a normal container.
  - On the node, each pod has a corresponding **qemu** process.

Verification

- You can check the logs of the **openshift-sandboxed-containers-operator** controller pod to see detailed messages about the steps it is running.
  - You can retrieve the name of the controller pod by running:

    ```
    $ oc get pods -n openshift-sandboxed-containers-operator | grep openshift-sandboxed-containers-operator-controller-manager
    ```

    This enables you to monitor the logs of the container manager of that pod.

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 4.8 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 4.8 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 4.8 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 4.8 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 4.8 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.