OpenShift Container Platform 4.8

OpenShift 的沙盒容器的支持

OpenShift 沙盒容器指南

Last Updated: 2023-04-19

OpenShift Container Platform 4.8 OpenShift 的沙盒容器的支持

OpenShift 沙盒容器指南

法律通告

摘要

OpenShift 沙盒容器支持 OpenShift Container Platform 为用户提供对将 Kata Containers 作为额外可选运行时运行的内置支持。

目录

第 1 章 {SANDBOXED-CONTAINERS-FIRST} 1.0 发行注记

1.1. 关于此版本

本发行注记介绍了 Red Hat OpenShift Container Platform 中的 OpenShift 沙盒容器的开发。

这个产品目前还只是一个技术预览。OpenShift 沙盒容器并不适用于生产环境。如需更多信息，请参阅红帽客户门户网站中的支持功能的范围。

1.2. 新功能及功能增强

1.2.1. OpenShift 沙盒容器（sandboxed containers）支持 OpenShift Container Platform（技术预览）

OpenShift 沙盒容器 1.0.0 技术预览版本引进了对将 Kata Containers 作为额外运行时运行的内置支持。OpenShift 沙盒容器允许用户选择 Kata Containers 作为额外的运行时，以便为工作负载提供额外的隔离。OpenShift 沙盒容器 Operator 会自动执行安装、删除和更新 Kata 容器的任务。它可以通过描述 **KataConfig** 自定义资源来跟踪这些任务 状态。

OpenShift 沙盒容器仅在裸机上受支持。Red Hat Enterprise Linux CoreOS（RHCOS）是 OpenShift 沙盒容器 1.0.0 唯一支持的操作系统。OpenShift Container Platform 4.8 不支持断开连接的环境。

如需更多信息，请参阅了解 OpenShift 沙盒容器

1.3. 已知问题

- 如果使用 OpenShift 沙盒容器，则无法使用 OpenShift Container Platform 集群中的 **hostPath** 卷将主机节点的文件系统中的文件或目录挂载到 pod 中。另外，您可以使用本地持久性卷。如需更多信息，请参阅 使用本地卷的持久性存储。（BZ#1904609）

- 如果您在 OpenShift 沙盒容器上运行 Fedora，则需要一个临时解决方案来安装一些软件包。有些软件包（如 **iputils** ）需要 OpenShift Container Platform 默认不授予容器的文件访问权限更改。要运行需要此类特殊权限的容器，需要向 YAML 文件添加描述工作负载的注解，该文件告知 **virtiofsd** 接受此类工作负载的文件权限。所需的注解为：

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

  (BZ#1915377)

- 在 4.8 发行版中，使用 OpenShift Container Platform Web 控制台为 **kataConfgPoolSelector** 添加值会导致 **scheduling.nodeSelector** 使用空值填充。使用带有 **kata** 值的 **RuntimeClass** 的 Pod 可能会被调度到没有安装 Kata Containers 运行时的节点。
  要临时解决这个问题，请运行以下命令在 **RuntimeClass kata** 中手动指定 **nodeSelector** 值：

  ```
  $ oc edit runtimeclass kata
  ```

  以下是带有正确 **nodeSelector** 语句的 **RuntimeClass** 示例。

  ```
  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"
```

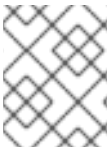
(BZ#2019384)

- Operator Hub 上的 OpenShift 沙盒容器 Operator 详情页面包含一些缺少的字段。缺少的字段不会阻止您在 4.8 中安装 OpenShift 沙盒容器 Operator。(BZ#2019383)
- 创建多个 **KataConfig** 自定义资源会导致静默失败。OpenShift Container Platform Web 控制台不提供通知用户创建多个自定义资源失败的提示。(BZ#2019381)
- 有时，OpenShift Container Platform Web 控制台中的 Operator Hub 不会显示 Operator 的图标。(BZ#2019380)

1.4. 异步勘误更新

OpenShift 沙盒容器 1.0 的安全更新、程序错误修正、功能增强更新将会通过红帽网络以异步勘误的形式发布。所有的 OpenShift Container Platform 4.8 勘误都 可以通过红帽客户门户网站获得。OpenShift Container Platform 生命周期包括了详细的与异步勘误相关的内容。

红帽客户门户网站的用户可以在红帽订阅管理（RHSM）帐户设置中启用勘误通知功能。当勘误通知被启用后，用户会在有与其注册的系统相关的勘误发行时接收到电子邮件通知。



注意

用户的红帽客户门户网站账户需要有注册的系统，以及使用 OpenShift Container Platform 的权限才可以接收到 OpenShift Container Platform 的勘误通知。

本节的内容将会持续更新，以提供以后发行的与 OpenShift 沙盒容器 1.0.0 相关的异步勘误信息。

1.4.1. RHBA-2021:3751 - OpenShift 沙盒容器 1.0.2 程序错误修复更新

发布日期：2021 年 10 月 7 日

OpenShift 沙盒容器版本 1.0.2 现已正式发布。此公告包含 OpenShift 沙盒容器的更新，并包括了相关的程序漏洞修复。

其程序错误修正列表包括在 RHBA-2021:3751 公告中。

1.4.2. RHBA-2021:3552 - OpenShift 沙盒容器 1.0.1 程序错误修复更新

发布日期：2021 年 9 月 16 日

OpenShift 沙盒容器版本 1.0.1 现已正式发布。此公告包含 OpenShift 沙盒容器的更新，并包括了相关的程序漏洞修复。

其程序错误修正列表包括在 RHBA-2021:3552 公告中。

1.4.3. RHEA-2021:2546 - OpenShift 沙盒容器 1.0.0 镜像发行版本、程序错误修正和功能增强公告

发布日期：2021 年 7 月 29 日

OpenShift 沙盒容器发行版本 1.0.0 支持 OpenShift Container Platform 4.8 的组件现在作为技术预览提供。

其程序错误修正列表包括在 RHEA-2021:3941 公告中。

第 2 章 了解 OPENSHIFT 沙盒容器

OpenShift 沙盒容器支持 OpenShift Container Platform 为用户提供对将 Kata Containers 作为额外可选运行时运行的内置支持。这对希望执行以下任务的用户特别有用：

- 运行特权或不受信任的工作负载。
- 确保每个工作负载的内核隔离。
- 在租户之间共享相同的工作负载。
- 确保正确隔离和沙盒测试软件。
- 确保跨越 VM 边界的默认资源控制。

OpenShift 沙盒容器还让用户能够从希望运行的工作负载类型中进行选择，以满足不同的用例。

您可以使用 OpenShift 沙盒容器 Operator 来执行诸如安装和删除、更新和状态监控等任务。

沙盒容器仅在裸机上受支持。

Red Hat Enterprise Linux CoreOS（RHCOS）是 OpenShift 沙盒容器 1.0.0 唯一支持的操作系统。

2.1. OPENSHIFT 沙盒容器常用术语

以下是整个文档中所使用的术语：

Sandbox

沙盒（sandbox）是一种隔离的环境，程序可以在其中运行。在沙盒中，您可以运行未经测试或不受信任的程序，而不影响到主机机器或操作系统。
在 OpenShift 沙盒容器环境中，沙盒通过使用虚拟化在不同的内核中运行工作负载来实现，从而增强了对在同一主机上运行的多个工作负载之间的交互的控制。

Pod

pod 是继承自 Kubernetes 和 OpenShift Container Platform 的构造。它代表了可以部署容器的资源。容器在 pod 内运行，pod 用于指定可以在多个容器之间共享的资源。
在 OpenShift 沙盒容器上下文中，pod 被实施为一个虚拟机。多个容器可以在同一虚拟机上在同一 pod 中运行。

OpenShift 沙盒容器 Operator

Operator 是一个软件组件，可自动执行一般需要人工在系统上执行的操作。
OpenShift 沙盒容器 Operator 的任务是管理集群上沙盒容器的生命周期。它涉及如安装和删除沙盒容器软件以及状态监控等操作。

Kata 容器

Kata 容器是一个上游核心项目，用于构建 OpenShift 沙盒容器。OpenShift 沙盒容器将 Kata 容器与 OpenShift Container Platform 集成。

KataConfig

KataConfig 对象代表沙盒容器的配置。它们存储有关集群状态的信息，如部署软件的节点。

RHCOS 扩展

Red Hat Enterprise Linux CoreOS（RHCOS）扩展是安装可选 OpenShift Container Platform 软件的一种机制。OpenShift 沙盒容器 Operator 使用此机制在集群中部署沙盒容器。

运行时类

RuntimeClass 对象用于描述可以使用哪个运行时来运行给定工作负载。OpenShift 沙盒容器 Operator 安装和部署了名为 **kata** 的运行时类。运行时类包含有关运行时的信息，用于描述运行时需要运行的资源，如 pod 开销。

2.2. OPENSHIFT 沙盒容器构建块

OpenShift 沙盒容器 Operator 封装了来自 Kata 容器的所有组件。它管理安装、生命周期和配置任务。

OpenShift 沙盒容器 Operator 以 Operator 捆绑包格式打包为两个容器镜像。捆绑包镜像包含元数据，这是使 operator OLM 就绪所必需的。第二个容器镜像包含监控和管理 **KataConfig** 资源的实际控制器。

2.3. RHCOS 扩展

OpenShift 沙盒容器 Operator 基于 Red Hat Enterprise Linux CoreOS（RHCOS）扩展概念。沙盒容器 RHCOS 扩展包含用于 Kata、QEMU 及其依赖项的 RPM。您可以使用 Machine Config Operator 提供的 **MachineConfig** 资源启用它们。

其他资源

- 为 RHCOS 添加扩展

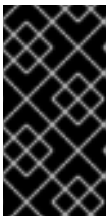
第 3 章 部署 OPENSHIFT 沙盒容器工作负载

您可以使用 Web 控制台或 OpenShift CLI（**oc**）安装 OpenShift 沙盒容器 Operator。安装 OpenShift 沙盒容器 Operator 之前，您必须准备 OpenShift Container Platform 集群。

3.1. 为 OPENSHIFT 沙盒容器准备集群

在安装 OpenShift 沙盒容器前，请确保 OpenShift Container Platform 集群满足以下要求：

- 集群必须使用 Red Hat Enterprise Linux CoreOS（RHCOS）worker 在裸机基础架构上安装。您的集群必须使用安装程序置备的基础架构。



重要

- OpenShift 沙盒容器仅支持 RHCOS worker 节点。不支持 RHEL 7 或 RHEL 8 节点。
- 不支持嵌套虚拟化。

3.1.1. OpenShift 沙盒容器的其他资源要求

OpenShift 沙盒容器是一种产品，它能够在沙盒运行时（如 Kata Containers）中运行工作负载到 OpenShift Container Platform 集群。每个 pod 由一个虚拟机（VM）表示。每个虚拟机都在 **qemu** 进程中运行，并托管一个 **kata-agent** 进程，该进程充当管理这些容器中运行的容器工作负载和进程的主管。另外还有两个额外的进程，这会增加开销：

- **containerd-shim-kata-v2** 用于与 pod 通信。
- **virtiofsd** 代表客户机处理主机文件系统访问。

每个虚拟机都配置有默认内存量。对于明确请求内存的容器，额外的内存会被热插到虚拟机中。

- 如果容器在没有指定内存资源的情况下运行，则它能够消耗可用内存。它将执行此操作，直到虚拟机使用的总内存达到默认的分配量。客户机及其 I/O 缓冲区也消耗内存。
- 如果容器被授予特定数量的内存，那么该内存会在容器启动前热插到虚拟机中。
- 如果指定了内存限制，则在消耗的内存超过限制的内存时，工作负载将被终止。如果没有指定内存限制，则虚拟机上运行的内核可能会耗尽内存。如果内核内存不足，它可能会终止虚拟机上的其他进程。

默认内存大小

下表列出了资源分配的一些默认值。

资源	值
默认为虚拟机分配的内存	2Gi
启动时客户机 Linux 内核内存使用	~110Mi
QEMU 进程使用的内存（虚拟机内存除外）	~30Mi

资源	值
virtiofsd 进程使用的内存（虚拟机 I/O 缓冲区除外）	~10Mi
containerd-shim-kata-v2 进程使用的内存	~20Mi
在 Fedora 上运行 **dnf install** 后的文件缓冲区缓存数据	~300Mi* [1]

1. 文件缓冲区会出现并在多个位置考虑：
   - 在客户机中它被显示为文件缓冲缓存。
   - 在映射允许的用户空间文件 I/O 操作的 **virtiofsd** 守护进程中。
   - 在 QEMU 进程中作为客户机内存。

注意

内存使用率指标正确考虑内存用量总量，该指标仅计算该内存一次。

Pod 开销描述了节点上 pod 使用的系统资源量。您可以使用 **oc describe runtimeclass kata** 获得 **kata** 运行时类的当前 pod 开销，如下所示。

示例

```
$ oc describe runtimeclass kata
```

输出示例

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

您可以通过更改 **RuntimeClass** 的 **spec.overhead** 字段来更改 pod 开销。例如，如果您的容器运行的配置为 QEMU 进程和客户机内核数据消耗超过 350Mi 内存，您可以更改 **RuntimeClass** 开销来满足您的需要。

注意

红帽支持指定的默认开销值。不支持更改默认开销值，这可能会导致技术问题。

示例

```
kind: RuntimeClass
```

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

- 虚拟机的默认分配为 2Gi。
- Linux 内核在引导时使用大约 100Mi 内存。
- QEMU 进程使用大约 30Mi 内存。
- **virtiofsd** 进程使用大约 10Mi 内存。
- **shim-v2** 进程使用大约 20Mi 内存。

在客户机中执行任何类型的文件系统 I/O 时，将在客户机内核中分配文件缓冲区。文件缓冲区也在主机上的 QEMU 进程以及 **virtiofsd** 进程中映射。例如，如果您在客户机中使用 300Mi 文件缓冲区缓存，QEMU 和 **virtiofsd** 都显示使用 300Mi 额外内存。但是，所有三种情况下都使用相同的内存。换句话说，内存使用的总量仅为 300Mi，这个值被映射在三个不同的位置。报告内存使用率指标时，会正确计算。

其他资源

- 在裸机上部署安装程序置备的集群

3.2. 使用 WEB 控制台部署 OPENSHIFT 沙盒容器 OPERATOR

您可以安装 Operator 并从 web 控制台查看工作负载。

3.2.1. 使用 Web 控制台安装 OpenShift 沙盒容器 Operator

您可从 OpenShift Container Platform Web 控制台安装 OpenShift 沙盒容器 Operator。

先决条件

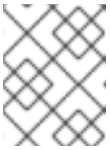
- 已安装 OpenShift Container Platform 4.8。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 打开浏览器窗口并登录 OpenShift Container Platform web 控制台。
2. 从 **Administrator** 视角中，进入 **Operators → OperatorHub**。
3. 在 **Filter by keyword** 字段中，输入 **OpenShift sandboxed containers**。
4. 选择 **OpenShift sandboxed containers** 标题。
5. 阅读 Operator 信息并单击 **Install**。
6. 在 **Install Operator** 页面中：

a. 从可用 **Update Channel** 选项列表中选择 **preview-1.0**。这样可确保安装与 OpenShift Container Platform 版本兼容的 OpenShift 沙盒容器版本。

b. 对于 **Installed Namespace**，请确保选择了 Operator 推荐的命名空间选项。这会在 **openshift-sandboxed-containers-operator** 命名空间中安装 Operator，如果它不存在，会自动创建。



注意

尝试在 **openshift-sandboxed-containers-operator** 以外的命名空间中安装 OpenShift 沙盒容器 Operator 会导致安装失败。

c. 对于 **Approval Strategy**，请确保已选择默认值 **Automatic**。当有新的 z-stream 版本可用时，OpenShift 沙盒容器会自动更新。

7. 点击 **Install** 使 Operator 可供 OpenShift 沙盒容器命名空间使用。

OpenShift 沙盒容器 Operator 现已安装在集群中。您可以通过在集群中启用运行时来触发 Operator。您可以使用 OpenShift CLI（**oc**）创建 **KataConfig** 自定义资源。

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

3.2.2. 从 web 控制台查看 OpenShift 沙盒容器工作负载

在 Web 控制台中查看时，基于 OpenShift 沙盒的容器的外观和行为与正常工作负载相同。两者之间的唯一区别是 **runtimeClassName**。**runtimeClassName** 决定用于工作负载的运行时。在这种情况下，OpenShift 沙盒容器启用的运行时是 **kata**。您可以查看 pod 用于工作负载的 **runtimeClass**。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.8。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 进入 **Administration → Workloads**。
2. 确定您要查看详情的工作负载类型。例如： **Pod**、**Deployment**、**DeploymentConfigs** 对象等。
3. 从列表中选择对应的工作负载。
4. 在 **Details** 页面中，进入 **runtimeClass**。
5. 将鼠标悬停在 **runtimeClass** 上，以查看更多信息。如果将 **kata** 用作运行时，则 **runtimeClass** 的值为 **kata**。

3.3. 使用 CLI 部署 OPENSHIFT 沙盒容器 OPERATOR

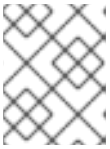
您可以安装和部署 Operator，并通过 CLI 查看工作负载。

3.3.1. 使用 CLI 安装 OpenShift 沙盒容器 Operator

您可以使用 OpenShift Container Platform CLI 安装 OpenShift 沙盒容器 Operator。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.8。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已订阅了 OpenShift 沙盒容器目录。



注意

订阅 OpenShift 沙盒容器目录为 **openshift-sandboxed-containers-operator** 命名空间提供了对 OpenShift 沙盒容器 Operator 的访问权限。

流程

1. 创建一个包含以下清单的 YAML 文件：

```
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"
```

注意

使用 **preview-1.0** 频道可确保安装与 OpenShift Container Platform 版本兼容的 OpenShift 沙盒容器版本。

2. 为 OpenShift 沙盒容器创建所需的 **Namespace**、**OperatorGroup** 和 **Subscription** 对象：

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

3. 确保正确安装 Operator:

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

输出示例

```
NAME                         DISPLAY                          VERSION  REPLACES
PHASE
openshift-sandboxed-containers   openshift-sandboxed-containers-operator  1.0.0    <csv-of-
previous-version>   Succeeded
```

4. 查看可用的部署：

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

输出示例

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

验证

- 验证 Operator 是否正在运行，以便您可以创建 **KataConfig** 资源来触发安装。

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

输出示例

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

其他资源

- 使用 CLI 从 OperatorHub 安装

3.3.1.1. 触发 Kata 运行时安装

您必须创建一个 **KataConfig** 自定义资源（CR）来触发 OpenShift 沙盒容器 Operator 以执行以下操作：

- 在 RHCOS 节点上安装所需的 RHCOS 扩展，如 QEMU 和 **kata-containers**。
- 确保运行时 CRI-O 配置了正确的 Kata 运行时处理程序。
- 创建带有必要配置的 **RuntimeClass** 自定义资源，用于虚拟化和所需额外进程导致的额外开销。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.8。

- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 创建 **KataConfig** 资源：

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

示例

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

2. 监控安装进度。
   - 您可以描述 **KataConfig** 安装：

```
$ oc describe kataconfig
```

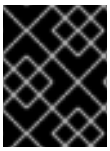
     - 验证状态中的 **Completed nodes** 字段。
     - 如果 **Completed 节点**的值与 worker 节点的数量匹配，则代表安装已完成。该状态还包含安装完成的节点的列表。

   - 您可以通过观察 **KataConfig** 资源来检查安装的进度：

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

     另外，您可以检查 **KataConfig** 资源的状态。这可以通过运行 **oc get KataConfig <name> -oyaml** 并检查输出中的 **status** 字段来完成。

Kata 运行时现在在集群中安装，并可作为辅助运行时使用。验证您是否在集群中为 Kata 创建了 **RuntimeClass**。



重要

OpenShift 沙盒容器仅将 Kata 安装为集群上的辅助可选运行时，而不作为主要运行时安装。

验证

- 您可以通过运行以下命令监控 **KataConfig** 自定义资源的值：

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

其他资源

- RuntimeClass

3.3.1.2. 为 OpenShift 沙盒容器选择节点

您可以选择在特定 worker 上安装 Kata 运行时。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.8。
- 已安装 OpenShift CLI（oc）。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 确定您要用来选择节点的标签。在本例中，使用标签选择作为候选者，以便在 OpenShift 沙盒容器工作负载上运行。如果节点存在，则会选择它们。
   a. 要将标签应用到节点，请运行以下命令：

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

   这会使用 **<label>** 标签标记您的 worker 节点，该标签值为 **<value>**。
2. 要添加标签选择器，请编辑 **KataConfig** 自定义资源（CR）：

   ```
   $ oc edit kataconfig
   ```

   示例

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

验证

- 您可以检查 **machine-config-pool** 对象中的节点是否经历配置更新。
  - 如果使用默认节点，您可以运行以下命令来监控 **machine-config-pool** 资源：

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

  - 如果使用所选节点，您可以运行以下命令监控 **machine-config-pool** 资源：

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

- 您可以运行 **watch oc describe kataconfig cluster-kataconfig** 来显示节点上 **sandboxed-containers** 扩展失败的信息。这些信息从 **machine-config-pool** 对象的状态收集。您可以运行以下命令查看信息：

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

3.3.1.3. 调度 OpenShift 沙盒容器工作负载

您可以调度工作负载在 OpenShift 沙盒容器中运行。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.8。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 将 **runtimeClassName: kata** 添加到任何 pod 模板化的资源：
   - **Pod** 对象
   - **ReplicaSet** 对象
   - **ReplicationController** 对象
   - **StatefulSet** 对象
   - **Deployment** 对象
   - **deploymentConfig** 对象

Pod 对象示例

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

Deployment 对象示例

```
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
```

当创建了带有 **runtimeClassName: kata** 的 pod 模板化的资源后，OpenShift Container Platform 开始将工作负载调度到启用了 OpenShift 沙盒的容器。如果没有使用选择器，则默认设置为所有 worker 节点。您的工作负载在 OpenShift 沙盒容器中运行。

3.3.2. 通过 CLI 查看 OpenShift 沙盒容器工作负载

您可以从 CLI 查看用于工作负载的 pod 的 **runtimeClass**。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.8。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

- 检查 pod 上的 **runtimeClassName** 字段，以查看 OpenShift 沙盒容器中运行的 pod 与普通容器。
  - 在节点上，每个 pod 具有对应的 **qemu** 进程。

验证

- 您可以检查 **openshift-sandboxed-containers-operator** 控制器 pod 的日志，以查看其正在运行的步骤的详细信息。
  - 您可以运行以下命令来检索控制器 pod 的名称：

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

    这可让您监控该 pod 的容器管理器的日志。

第 4 章 卸载 OPENSHIFT 沙盒容器

4.1. 使用 WEB 控制台卸载 OPENSHIFT 沙盒容器

您可以使用 OpenShift Container Platform Web 控制台卸载 OpenShift 沙盒容器。

4.1.1. 删除 OpenShift 沙盒容器资源

要卸载 OpenShift 沙盒容器，必须首先删除 OpenShift 沙盒容器自定义资源 **KataConfig**。这会从集群中移除并卸载 **kata** 运行时及其相关资源。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.8。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您没有正在运行的、使用 **kata** 作为 **runtimeClassName** 的 pod。
  - 已安装 OpenShift CLI(**oc**)。
  - 已安装命令行 JSON 处理器（**jq**）。
  - 运行以下命令，验证您没有运行使用 **kata** 作为 **runtimeClassName** 的 pod：

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

流程

1. 删除所有使用 **runtimeClassName** 的 pod，其值为 **kata**。
2. 在 OpenShift Container Platform web 控制台中，从 **Projects** 列表中选择 **openshift-sandboxed-containers**。
3. 进入到 **Operators → Installed Operators** 页面。
4. 点 **OpenShift 沙盒容器**。
5. 点 **OpenShift 沙盒 containers Operator** 选项卡。
6. 点 **Operator Details** 中的滚动列表，然后单击 **Delete KataConfig**。
7. 在确认窗口中点击 **Delete**。

4.1.1.1. 使用 web 控制台删除命令空间

您可以使用 OpenShift Container Platform web 控制台删除一个命名空间。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.8。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 导航至 **Administration → Namespaces**。
2. 在命名空间列表中找到要删除的 **openshift-sandboxed-containers-operator** 命名空间。
3. 在命名空间列表的最右侧，从 **Options** 菜单中选择 **Delete Namespace**。
4. 当 **Delete Namespace** 窗格打开时，在字段中输入 **openshift-sandboxed-containers-operator**。

注意

如果 **Delete Namespace** 选项不可用，代表您没有删除命名空间的权限。

5. 单击 **Delete**。

4.1.2. 删除 OpenShift 沙盒容器 Operator

您可以通过删除目录订阅并撤销对 Operator 的命名空间访问权限来删除 OpenShift 沙盒容器 Operator。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.8。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 导航到 **Operators → OperatorHub** 页面。
2. 搜索 **OpenShift sandboxed containers**，然后选择 Operator。
3. 点击 **Uninstall**。
4. 删除 **openshift-sandboxed-containers-operator** 命名空间。

4.2. 通过 CLI 卸载 KATA 运行时

您可以使用 OpenShift Container Platform 命令行界面（CLI）卸载 OpenShift 沙盒容器。

4.2.1. 删除 OpenShift 沙盒容器资源

您可以从集群中移除和卸载 **kata** 运行时及其所有相关资源，如 CRI-O 配置和 **RuntimeClass**。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.8。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 运行以下命令来删除 **KataConfig** 自定义资源：

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

2. 运行以下命令来删除 **KataConfig** 自定义资源定义：

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

OpenShift 沙盒容器 Operator 会删除最初为在集群中启用运行时创建的所有资源。运行上述命令后，集群将恢复到安装过程之前的状态。现在，您可以删除 **openshift-sandboxed-containers-operator** 命名空间。

验证

- 要验证 **KataConfig** 自定义资源是否已删除，请运行以下命令：

```
$ oc get kataconfig <KataConfig_CR_Name>
```

输出示例

```
No KataConfig instances exist
```

- 要验证 **KataConfig** 自定义资源的定义已被删除，请运行以下命令：

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

输出示例

```
Unknown CR KataConfig
```

4.2.2. 删除 OpenShift 沙盒容器 Operator

您可以从集群中删除 OpenShift 沙盒容器 Operator。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.8。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 运行以下命令，从 Operator Lifecyle Manager（OLM）中删除 OpenShift 沙盒容器 Operator 订阅：

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

2. 运行以下命令，将 OpenShift 沙盒容器的集群服务版本（CSV）名称设置为环境变量：

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

3. 运行以下命令，删除 OpenShift 沙盒容器的 CSV 名称：

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

第 5 章 升级 OPENSHIFT 沙盒容器

您可以通过升级 OpenShift 沙盒容器 Operator 和 OpenShift 沙盒容器工件来升级 OpenShift 沙盒容器的组件。

5.1. 升级 OPENSHIFT 沙盒容器 OPERATOR

您可以使用 Operator Lifecycle Manager（OLM）手动或自动升级 OpenShift 沙盒容器 Operator。您可以在初始部署过程中选择手动或自动升级。在手动升级的情况下，Web 控制台会显示集群管理员可以安装的可用更新。

其他资源

- 升级安装的 Operator

5.2. 升级 OPENSHIFT 沙盒容器工件

OpenShift 沙盒容器工件使用 Red Hat Enterprise Linux CoreOS（RHCOS）扩展部署到集群中。

RHCOS 扩展**沙盒容器**包含运行 Kata 容器所需的组件，如 Kata 容器运行时、虚拟机监控程序 QEMU 和其他依赖项。当将集群升级到新版 OpenShift Container Platform 时，扩展会被升级。