OpenShift Sandboxed Containers 1.4

OpenShift 沙盒容器用户指南

Red Hat OpenShift

Last Updated: 2024-02-02

OpenShift Sandboxed Containers 1.4 OpenShift 沙盒容器用户指南

Red Hat OpenShift

法律通告

摘要

有关 OpenShift 沙盒容器操作器及其用法的信息

目录

前言 3

第 1 章 了解 OPENSHIFT 沙盒容器 4
1.1. OPENSHIFT 沙盒容器支持的平台 4
1.2. OPENSHIFT 沙盒容器常用术语 5
1.3. OPENSHIFT 沙盒容器工作负载管理 5
1.4. 了解合规性及风险管理 6

第 2 章 部署 OPENSHIFT 沙盒容器工作负载 7
2.1. 先决条件 7
2.2. 使用 WEB 控制台部署 OPENSHIFT 沙盒容器工作负载 11
2.3. 使用 CLI 部署 OPENSHIFT 沙盒容器工作负载 14
2.4. 其他资源 19

第 3 章 使用对等 POD 部署 OPENSHIFT 沙盒容器工作负载 20
3.1. 先决条件 20
3.2. 使用带有 WEB 控制台的对等 POD 部署 OPENSHIFT 沙盒容器工作负载 24
3.3. 使用带有 CLI 的对等 POD 部署 OPENSHIFT 沙盒容器工作负载 33
3.4. 其他资源 45

第 4 章 监控 OPENSHIFT 沙盒容器 46
4.1. 关于 OPENSHIFT 沙盒容器指标 46
4.2. 查看 OPENSHIFT 沙盒容器的指标 46
4.3. 查看 OPENSHIFT 沙盒容器仪表板 47
4.4. 其他资源 47

第 5 章 卸载 OPENSHIFT 沙盒容器 48
5.1. 使用 WEB 控制台卸载 OPENSHIFT 沙盒容器 48
5.2. 使用 CLI 卸载 OPENSHIFT 沙盒容器 51

第 6 章 升级 OPENSHIFT 沙盒容器 55
6.1. 升级 OPENSHIFT 沙盒容器资源 55
6.2. 升级 OPENSHIFT 沙盒容器 OPERATOR 55
6.3. 升级 OPENSHIFT 沙盒容器监控 POD 55

第 7 章 收集 OPENSHIFT 沙盒容器数据 57
7.1. 为红帽支持收集 OPENSHIFT 沙盒容器数据 57
7.2. 关于 OPENSHIFT 沙盒容器日志数据 58
7.3. 为 OPENSHIFT 沙盒容器启用调试日志 58
7.4. 其他资源 60

前言

第 1 章 了解 OPENSHIFT 沙盒容器

OpenShift 沙盒容器支持 Red Hat OpenShift 为您提供了将 Kata Containers 作为额外可选运行时运行的内置支持。新的运行时支持专用虚拟机 (VM) 中的容器，从而改进了工作负载隔离。这对执行以下任务特别有用：

运行特权或不受信任的工作负载

OpenShift 沙盒容器 (OSC) 使得可以安全地运行需要特定特权的工作负载，而无需通过运行特权容器来破坏集群节点的风险。需要特殊权限的工作负载包括：

- 需要内核的特殊功能的工作负载，除了标准容器运行时（如 CRI-O）授予的默认功能外，例如访问低级别网络功能。
- 需要提高 root 特权的工作负载，例如访问特定物理设备。使用 OpenShift 沙盒容器时，只能将特定的设备传递给虚拟机，确保工作负载无法访问或错误配置系统的其余部分。
- 用于安装或使用 **set-uid** root 二进制文件的工作负载。这些二进制文件授予特殊权限，因此可能会造成安全风险。使用 OpenShift 沙盒容器时，对虚拟机有额外的权限，不授予对集群节点的特殊访问权限。

有些工作负载可能需要专门用于配置集群节点的权限。此类工作负载应该仍然使用特权容器，因为在虚拟机上运行可能会阻止它们正常工作。

确保每个工作负载的内核隔离

OpenShift 沙盒容器支持需要自定义内核调整（如 **sysctl**、调度程序更改或缓存调整）以及创建自定义内核模块（如 **out of tree** 或特殊参数）的工作负载。

在租户间共享相同的工作负载

OpenShift 沙盒容器允许您支持来自共享同一 OpenShift 集群的不同组织的多个用户（租户）。该系统还允许您从多个供应商运行第三方工作负载，如容器网络功能 (CNF) 和企业应用程序。例如，第三方 CNF 可能不希望其自定义设置与数据包调整或由其他应用程序设置的 **sysctl** 变量干扰。在完全隔离的内核内运行有助于防止"邻居噪音"配置问题。

确保正确隔离和沙盒测试软件

您可以使用 OpenShift 沙盒容器来运行具有已知漏洞的容器化工作负载，或者处理传统应用程序中的问题。通过这种隔离，管理员可以为开发人员提供对 pod 的管理控制，这在开发人员想要测试或验证管理员通常授权的配置时很有用。例如，管理员可以安全地将内核数据包过滤 (eBPF) 委派给开发人员。内核数据包过滤需要 **CAP_ADMIN** 或 **CAP_BPF** 权限，因此不允许在标准 CRI-O 配置下，因为这会授予容器主机 worker 节点上的每个进程的访问权限。类似地，管理员可以授予对 SystemTap 等入侵工具的访问权限，或者支持在开发期间加载自定义内核模块。

确保通过虚拟机边界的默认资源控制

默认情况下，CPU、内存、存储或网络等资源以 OpenShift 沙盒容器中的更加强大和安全的方式进行管理。由于 OpenShift 沙盒容器部署到虚拟机上，因此额外的隔离层和安全性可为资源提供更精细的访问控制。例如，错误容器将无法分配超过虚拟机可用内存更多的内存。相反，需要专用访问网卡或磁盘的容器可以完全控制该设备，而无需访问其他设备。

1.1. OPENSHIFT 沙盒容器支持的平台

您可以在裸机服务器或 Amazon Web Services (AWS) 裸机实例上安装 OpenShift 沙盒容器。不支持由其他云提供商提供的裸机实例。

Red Hat Enterprise Linux CoreOS (RHCOS) 是 OpenShift 沙盒容器唯一支持的操作系统。OpenShift 沙盒容器 1.4 在 Red Hat Enterprise Linux CoreOS (RHCOS) 8.6 上运行。

OpenShift 沙盒容器 1.4 与 Red Hat OpenShift 4.11 兼容。

1.2. 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 是一个继承自 Kubernetes 和 Red Hat OpenShift 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。您可以使用 OpenShift 沙盒容器 Operator 来执行任务，如安装和删除沙盒容器、软件更新和状态监控。

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 Containers 与 Red Hat OpenShift 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。

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。

对等（peer）pod

OpenShift 沙盒容器中的对等 pod 扩展标准 pod 的概念。与标准沙盒容器不同，在 worker 节点本身上创建虚拟机，在对等 pod 中，虚拟机会使用任何支持的虚拟机监控程序或云供应商 API 通过远程 hypervisor 创建。对等 pod 作为 worker 节点上的常规 pod，其对应的虚拟机在其他位置运行。虚拟机的远程位置对用户是透明的，并由 pod 规格中运行时类指定。对等 pod 设计对嵌套虚拟化的需求。

1.3. OPENSHIFT 沙盒容器工作负载管理

OpenShift 沙盒容器提供以下功能以增强工作负载管理和分配：

1.3.1. 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。

1.3.2. RHCOS 扩展

OpenShift 沙盒容器 Operator 基于 Red Hat Enterprise Linux CoreOS（RHCOS）扩展概念。Red Hat Enterprise Linux CoreOS (RHCOS)扩展是安装可选 Red Hat OpenShift 软件的机制。OpenShift 沙盒容器 Operator 使用此机制在集群中部署沙盒容器。

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

1.3.3. 虚拟化和 OpenShift 沙盒容器

您可以在带有 OpenShift Virtualization 的集群上使用 OpenShift 沙盒容器。

要同时运行 OpenShift Virtualization 和 OpenShift 沙盒容器，您必须启用虚拟机迁移的虚拟机，以便不阻止节点重启。在虚拟机上配置以下参数：

- 使用 **ocs-storagecluster-ceph-rbd** 作为存储类。
- 在虚拟机中将 **evictionStrategy** 参数设置为 **LiveMigrate**。

其他资源

- 为虚拟机配置本地存储
- 配置虚拟机驱除策略

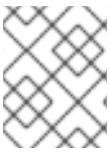
1.4. 了解合规性及风险管理

Red Hat OpenShift 专为 FIPS 设计。当以 FIPS 模式运行 Red Hat Enterprise Linux (RHEL)或 Red Hat Enterprise Linux CoreOS (RHCOS)时，Red Hat OpenShift 核心组件使用 RHEL 加密库，在 **x86_64**、**ppc64le**、**s390x** 架构上提交给 NIST 的 FIPS 140-2/140-3 Validation。

有关 NIST 验证程序的更多信息，请参阅加密模块验证程序。有关为验证提交的 RHEL 加密库的单独版本的最新 NIST 状态，请参阅 Compliance Activities 和 Government Standards。

OpenShift 沙盒容器可以在启用了 FIPS 的集群中使用。

在 FIPS 模式下运行时，OpenShift 沙盒容器组件、虚拟机和虚拟机镜像会根据 FIPS 进行调整。



注意

OpenShift 沙盒容器的 FIPS 合规性只适用于 **kata** 运行时类。新的对等 pod 运行时类 **kata-remote-cc** 尚未被完全支持，且尚未针对 FIPS 合规性进行测试。

FIPS 合规性是高安全性环境中所需的最重要的组件之一，可确保节点上只允许使用支持的加密技术。

重要

只有在 **x86_64** 架构的 Red Hat OpenShift 部署中才支持使用 FIPS 验证的/Modules in Process 加密库。

要了解红帽 OpenShift 合规框架的视图，请参阅 OpenShift 安全指南手册中的风险管理和责任章节。

第 2 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。在安装 OpenShift 沙盒容器 Operator 之前，您必须准备 Red Hat OpenShift 集群。

2.1. 先决条件

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

- 集群必须使用 Red Hat Enterprise Linux CoreOS (RHCOS) worker 在内部裸机基础架构上安装。您可以使用任何安装方法，包括用户置备、安装程序置备或协助的安装程序来部署集群。

注意

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

- 您可以在 Amazon Web Services (AWS) 裸机实例上安装 OpenShift 沙盒容器。不支持由其他云提供商提供的裸机实例。

2.1.1. OpenShift 沙盒容器的资源要求

OpenShift 沙盒容器允许用户在沙盒运行时 (Kata) 的 Red Hat OpenShift 集群上运行工作负载。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，它充当管理容器工作负载的监管程序，以及这些容器中运行的进程。两个额外的进程会增加开销：

- **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。

在没有内存资源的情况下运行的容器会消耗可用内存，直到虚拟机使用的总内存达到默认分配。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。

当指定内存限制时，如果消耗的内存超过限制，工作负载将被终止。如果没有指定内存限制，则虚拟机中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]

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** 获取 Kata 运行时的当前 pod 开销，如下所示。

示例

```
$ oc describe runtimeclass kata
```

输出示例

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

您可以通过更改 **RuntimeClass** 的 **spec.overhead** 字段来更改 pod 开销。例如，如果您为容器运行的配置消耗 QEMU 进程和客户机内核数据的 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。

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。

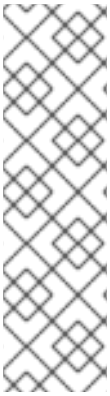
例如，如果您在客户机中使用 300Mi 文件缓冲区缓存，QEMU 和 **virtiofsd** 都显示使用 300Mi 额外内存。但是，所有三种情况下都使用相同的内存。换句话说，内存使用的总量仅为 300Mi，这个值被映射在三个不同的位置。报告内存使用率指标时，会正确计算。

其他资源

- 在裸机上安装用户置备的集群

2.1.2. 检查集群节点是否有资格运行 OpenShift 沙盒容器

在运行 OpenShift 沙盒容器前，您可以检查集群中的节点是否有资格运行 Kata 容器。有些集群节点可能不符合沙盒容器的最低要求。节点不合格的最常见原因是节点上缺少虚拟化支持。如果您试图在不符合节点上运行沙盒工作负载，则会出现错误。您可以使用 Node Feature Discovery (NFD) Operator 和 **NodeFeatureDiscovery** 资源自动检查节点资格。



注意

如果您只想在符合条件的所选 worker 节点上安装 Kata 运行时，请将 **feature.node.kubernetes.io/runtime.kata=true** 标签应用到所选节点，并在 **KataConfig** 资源中设置 **checkNodeEligibility: true**。

另外，要在所有 worker 节点上安装 Kata 运行时，在 **KataConfig** 资源中设置 **checkNodeEligibility: false**。

在这两种场景中，您不需要创建 **NodeFeatureDiscovery** 资源。如果您确定节点有资格运行 Kata 容器，则应该只应用 **feature.node.kubernetes.io/runtime.kata=true** 标签。

以下流程将 **feature.node.kubernetes.io/runtime.kata=true** 标签应用到所有有资格的节点，并将 **KataConfig** 资源配置为检查节点资格。

先决条件

- 安装 OpenShift CLI（**oc**）。
- 以具有 **cluster-admin** 特权的用户身份登录。
- 安装 Node Feature Discovery (NFD) Operator。

流程

1. 创建 **NodeFeatureDiscovery** 资源来检测适合运行 Kata 容器的节点功能：
   a. 将以下 YAML 保存到 **nfd.yaml** 文件中：

```
apiVersion: nfd.openshift.io/v1
kind: NodeFeatureDiscovery
metadata:
  name: nfd-kata
  namespace: openshift-nfd
spec:
  operand:
    image: quay.io/openshift/origin-node-feature-discovery:4.10
    imagePullPolicy: Always
    servicePort: 12000
  workerConfig:
```

```
    configData: |
      sources:
        custom:
          - name: "feature.node.kubernetes.io/runtime.kata"
            matchOn:
              - cpuId: ["SSE4", "VMX"]
                loadedKMod: ["kvm", "kvm_intel"]
              - cpuId: ["SSE4", "SVM"]
                loadedKMod: ["kvm", "kvm_amd"]
```

b. 创建 **NodeFeatureDiscovery** 自定义资源(CR)：

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

输出示例

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

feature.node.kubernetes.io/runtime.kata=true 标签应用到所有合格的 worker 节点。

2. 在 **KataConfig** 资源中将 **checkNodeEligibility** 字段设置为 **true** 来启用这个功能，例如：

a. 将以下 YAML 保存到 **kata-config.yaml** 文件中：

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

b. 创建 **KataConfig** CR：

```
$ oc create -f kata-config.yaml
```

输出示例

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

验证

- 验证集群中是否应用了正确的标签：

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

输出示例

```
NAME                     STATUS   ROLES    AGE     VERSION
compute-3.example.com    Ready    worker   4h38m   v1.25.0
compute-2.example.com    Ready    worker   4h35m   v1.25.0
```

其他资源

- 有关安装 Node Feature Discovery (NFD) Operator 的更多信息，请参阅安装 NFD。

2.2. 使用 WEB 控制台部署 OPENSHIFT 沙盒容器工作负载

您可从 web 控制台部署 OpenShift 沙盒容器工作负载。首先，您必须安装 OpenShift 沙盒容器 Operator，然后创建 **KataConfig** 自定义资源 (CR)。在沙盒容器中部署工作负载后，您必须手动将 **kata** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

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

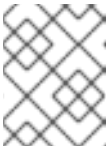
您可从 Red Hat OpenShift Web 控制台安装 OpenShift 沙盒容器 Operator。

先决条件

- 已安装 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 从 Web 控制台中的 **Administrator** 视角，进入到 **Operators → OperatorHub**。
2. 在 **Filter by keyword** 字段中，输入 **OpenShift sandboxed containers**。
3. 选择 **OpenShift sandboxed containers** 标题。
4. 阅读 Operator 信息并单击 **Install**。
5. 在 **Install Operator** 页面中：
   a. 从可用 **Update Channel** 选项列表中选择 **stable**。
   b. 验证为 **Installed Namespace** 选择了 **Operator recommended Namespace**。这会在 **openshift-sandboxed-containers-operator** 命名空间中安装 Operator。如果此命名空间尚不存在，则会自动创建。

   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. 验证是否为 **Approval Strategy** 选择了 **Automatic**。**Automatic** 是默认值，当有新的 z-stream 发行版本可用时，自动启用对 OpenShift 沙盒容器的自动更新。
6. 点 **Install**。

OpenShift 沙盒容器 Operator 现已安装在集群中。

验证

1. 从 Web 控制台中的 **Administrator** 视角，导航到 **Operators → Installed Operators**。
2. 验证 OpenShift 沙盒容器 Operator 是否在 operator 列表中列出。

2.2.2. 在 web 控制台中创建 KataConfig 自定义资源

您必须创建一个 **KataConfig** 自定义资源(CR)，以便在集群节点上启用将 **kata** 作为 **RuntimeClass**。

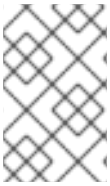
重要

创建 **KataConfig** CR 会自动重启 worker 节点。重启可能需要 10 到 60 分钟。妨碍重启时间的因素如下：

- 带有更多 worker 节点的大型 Red Hat OpenShift 部署。
- 激活 BIOS 和 Diagnostics 实用程序。
- 在硬盘而不是 SSD 上部署。
- 在物理节点上部署，如裸机，而不是在虚拟节点上部署。
- CPU 和网络较慢。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。



注意

Kata 默认安装在所有 worker 节点上。如果要在特定节点上安装 **kata** 作为 **RuntimeClass**，您可以向这些节点添加标签，然后在创建时定义 **KataConfig** CR 中的标签。

流程

1. 从 Web 控制台中的 **Administrator** 视角，导航到 **Operators → Installed Operators**。
2. 从 Operator 列表中选择 OpenShift 沙盒容器 Operator。
3. 在 **KataConfig** 选项卡中，点 **Create KataConfig**。
4. 在 **Create KataConfig** 页面中，输入以下详情：
   - **名称** ：输入 **KataConfig** 资源的名称。默认情况下，名称定义为 **example-kataconfig**。
   - **Labels** （可选）：输入任何相关的、识别到 **KataConfig** 资源的属性。每个标签代表一个键值对。
   - **checkNodeEligibility** （可选，不与对等 pod 相关的）：选择此复选框以使用 Node Feature Discovery Operator (NFD)检测节点资格以作为 **RuntimeClass** 运行 **kata**。如需更多信息，请参阅"检查集群节点是否有资格运行 OpenShift 沙盒容器"。
   - **enablePeerPods** （对于对等 pod）：选择此复选框来启用对等 pod，并在公有云环境中使用 OpenShift 沙盒容器。

- **kataConfigPoolSelector** ：默认情况下，**kata** 在所有节点上作为 **RuntimeClass** 安装。如果要在所选节点上安装 **kata** 作为 **RuntimeClass**，您必须添加一个 matchExpression ：

  a. 展开 **kataConfigPoolSelector** 区域。

  b. 在 **kataConfigPoolSelector** 中，展开 matchExpressions。这是标签选择器要求列表。

  c. 点 Add matchExpressions。

  d. 在 key 字段中，添加选择器应用到的标签键。

  e. 在 operator 字段中，添加键与标签值的关系。有效的运算符为 **In**、**NotIn**、**Exists** 和 **DoesNotExist**。

  f. 展开 values 区域，然后点 Add value。

  g. 在 Value 字段中，为 **key** 标签值输入 **true** 或 **false**。

- **loglevel** ：定义为将 **kata** 作为 **RuntimeClass** 运行的节点检索的日志数据级别。如需更多信息，请参阅"收集 OpenShift 沙盒容器数据"。

5. 点 Create。

新的 **KataConfig** CR 会被创建，并开始在 worker 节点上作为 **RuntimeClass** 安装 **kata**。等待 **kata** 安装完成，以及 worker 节点重启，然后继续下一步。

重要

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

验证

1. 在 KataConfig 选项卡中，选择新的 **KataConfig** CR。
2. 在 KataConfig 页面中，选择 YAML 选项卡。
3. 监控状态中的 **installationStatus** 字段。
   每次有更新时都会出现一条消息。点 Reload 查看更新的 **KataConfig** CR。

   当 **Completed nodes** 的值等于 worker 或已标记的节点的数量，则代表安装已完成。该状态还包含安装完成的节点的列表。

2.2.3. 使用 Web 控制台在沙盒容器中部署工作负载

OpenShift 沙盒容器将 Kata 安装为集群上的辅助、可选运行时，而不是主运行时。

要在沙盒容器中部署 pod 模板工作负载，您必须手动将 **kata** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

- 已安装 OpenShift 沙盒容器 Operator。
- 您已创建了 **KataConfig** 自定义资源 (CR)。

流程

1. 从 web 控制台中的 **Administrator** 视角，展开 **Workloads** 并选择您要创建的工作负载类型。
2. 在工作负载页面中，点击以创建工作负载。
3. 在工作负载的 YAML 文件中，在列出容器的 **spec** 字段中，添加 **runtimeClassName: kata**。

   Pod 对象示例

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-openshift
  labels:
    app: hello-openshift
spec:
  runtimeClassName: kata
  containers:
    - name: hello-openshift
      image: quay.io/openshift/origin-hello-openshift
      ports:
        - containerPort: 8888
      securityContext:
        privileged: false
        allowPrivilegeEscalation: false
        runAsNonRoot: true
        runAsUser: 1001
        capabilities:
          drop:
            - ALL
        seccompProfile:
          type: RuntimeDefault
```

4. 点击 **Save**。

Red Hat OpenShift 创建工作负载并开始调度它。

2.3. 使用 CLI 部署 OPENSHIFT 沙盒容器工作负载

您可以使用 CLI 部署 OpenShift 沙盒容器工作负载。首先，您必须安装 OpenShift 沙盒容器 Operator，然后创建 **KataConfig** 自定义资源。在沙盒容器中部署工作负载后，您必须将 **kata** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

2.3.1. 使用 CLI 安装 OpenShift 沙盒容器 Operator

您可以使用 Red Hat OpenShift CLI 安装 OpenShift 沙盒容器 Operator。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 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. 为 OpenShift 沙盒容器 Operator 创建 **Namespace** 对象。
   a. 创建一个包含以下清单的 **Namespace** 对象 YAML 文件：

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

   b. 创建 **Namespace** 对象：

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

2. 为 OpenShift 沙盒容器 Operator 创建 **OperatorGroup** 对象。
   a. 创建一个包含以下清单的 **OperatorGroup** 对象 YAML 文件：

```
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. 创建 **OperatorGroup** 对象：

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

3. 创建 **Subscription** 对象，以便为 OpenShift 沙盒容器 Operator 订阅 **命名空间**。
   a. 创建一个包含以下内容的 **Subscription** 对象 YAML 文件：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-sandboxed-containers-operator
  namespace: openshift-sandboxed-containers-operator
spec:
```

```
channel: stable
installPlanApproval: Automatic
name: sandboxed-containers-operator
source: redhat-operators
sourceNamespace: openshift-marketplace
startingCSV: sandboxed-containers-operator.v1.4.1
```

b. 创建 **Subscription** 对象：

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

OpenShift 沙盒容器 Operator 现已安装在集群中。

注意

以上列出的所有对象文件名都是建议。您可以使用其他名称创建对象 YAML 文件。

验证

- 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  1.4.1    1.4.0
Succeeded
```

其他资源

- 使用 CLI 从 OperatorHub 安装

2.3.2. 使用 CLI 创建 KataConfig 自定义资源

您必须创建一个 **KataConfig** 自定义资源 (CR)来作为 **RuntimeClass** 在节点上安装 **kata**。创建 **KataConfig** CR 会触发 OpenShift 沙盒容器 Operator 来执行以下操作：

- 在 RHCOS 节点上安装所需的 RHCOS 扩展，如 QEMU 和 **kata-containers**。
- 确保 CRI-O 运行时配置了正确的运行时处理程序。
- 使用默认配置创建一个名为 **kata** 的 **RuntimeClass** CR。这可让用户在 **RuntimeClassName** 字段中引用 CR 将工作负载配置为使用 **kata** 作为运行时。此 CR 也指定运行时的资源开销。

注意

Kata 默认安装在所有 worker 节点上。如果要在特定节点上安装 **kata** 作为 **RuntimeClass**，您可以向这些节点添加标签，然后在创建时在 **KataConfig** CR 中定义标签。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。

重要

创建 **KataConfig** CR 会自动重启 worker 节点。重启可能需要 10 到 60 分钟。妨碍重启时间的因素如下：

- 带有更多 worker 节点的大型 Red Hat OpenShift 部署。
- 激活 BIOS 和 Diagnostics 实用程序。
- 在硬盘而不是 SSD 上部署。
- 在物理节点上部署，如裸机，而不是在虚拟节点上部署。
- CPU 和网络较慢。

流程

1. 使用以下清单创建 YAML 文件：

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
spec:
  checkNodeEligibility: false (1)
  logLevel: info
```

(1) 将'checkNodeEligibility' 设置为 **true**，以检测节点资格作为 **RuntimeClass** 运行 **kata**。如需更多信息，请参阅"检查集群节点是否有资格运行 OpenShift 沙盒容器"。

2. （可选）如果只在所选节点上安装 **kata** 作为 **RuntimeClass**，请创建一个包含清单中的标签的 YAML 文件：

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

(1) **kataConfigPoolSelector** 中的标签只支持单个值；不支持 **nodeSelector** 语法。

3. 创建 **KataConfig** 资源：

```
$ oc create -f cluster-kataconfig.yaml
```

新的 **KataConfig** CR 会被创建，并开始在 worker 节点上作为 **RuntimeClass** 安装 **kata**。等待 **kata** 安装完成，以及 worker 节点重启，然后继续下一步。

重要

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

验证

- 监控安装进度：

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

当 **Is In Progress**的值显示为 **false** 后，安装就已完成。

其他资源

- 了解如何更新节点上的标签

2.3.3. 使用 CLI 在沙盒容器中部署工作负载

OpenShift 沙盒容器将 Kata 安装为集群上的辅助、可选运行时，而不是主运行时。

要在沙盒容器中部署 pod 模板工作负载，您必须将 **kata** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。
- 您已创建了 **KataConfig** 自定义资源 (CR)。

流程

- 将 **runtimeClassName: kata** 添加到任何 pod 模板对象中：
  - **Pod** 对象
  - **ReplicaSet** 对象
  - **ReplicationController** 对象
  - **StatefulSet** 对象

- **Deployment** 对象
- **deploymentConfig** 对象

Pod 对象的 .example

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-openshift
  labels:
    app: hello-openshift
spec:
  runtimeClassName: kata
  containers:
    - name: hello-openshift
      image: quay.io/openshift/origin-hello-openshift
      ports:
        - containerPort: 8888
      securityContext:
        privileged: false
        allowPrivilegeEscalation: false
        runAsNonRoot: true
        runAsUser: 1001
        capabilities:
          drop:
            - ALL
        seccompProfile:
          type: RuntimeDefault
```

Red Hat OpenShift 创建工作负载并开始调度它。

验证

- 检查 pod 模板对象上的 **runtimeClassName** 字段。如果 **runtimeClassName** 是 **kata**，则工作负载在 OpenShift 沙盒容器中运行。

2.4. 其他资源

- OpenShift 沙盒容器 Operator 在受限网络环境中被支持。如需更多信息，请参阅在受限网络中使用 Operator Lifecycle Manager。
- 在受限网络中使用断开连接的集群时，您必须在 Operator Lifecycle Manager 中配置代理支持来访问 OperatorHub。使用代理允许集群获取 OpenShift 沙盒容器 Operator。

第 3 章 使用对等 POD 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。在安装 OpenShift 沙盒容器 Operator 之前，您必须准备 Red Hat OpenShift 集群。

重要

使用对等 pod 部署 OpenShift 沙盒容器工作负载只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

3.1. 先决条件

在安装 OpenShift 沙盒容器并启用对等 pod 之前，您必须满足以下要求：

- 您已在 AWS 或 Azure 上安装 Red Hat OpenShift 4.13。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 cluster-admin 角色的用户访问集群。

3.1.1. 关于 OpenShift 沙盒容器中的对等 pod 资源要求

对等 pod 使用两个位置的资源：

- worker 节点。worker 节点存储元数据、Kata shim 资源(**containerd-shim-kata-v2**)、remote-hypervisor 资源(**cloud-api-adaptor**)，以及 worker 节点和 peer-pod 虚拟机(VM)之间的隧道设置。
- 云实例。这是在云中运行的实际 peer-pod 虚拟机。

Kubernetes worker 节点中使用的 CPU 和内存资源由用于创建对等 pod 的 RuntimeClass (**kata-remote**) 定义中的 pod 开销 处理。

云中运行的 peer-pod 虚拟机总数定义为 Kubernetes 节点扩展资源。这个限制是每个节点，由 **peerpodConfig** 自定义资源(CR)中的 **limit** 属性设置。在创建 **kataConfig** CR 并启用对等 pod 时，会创建名为 **peerpodconfig-openshift** 的 **peerpodConfig** CR，并位于 **openshift-sandboxed-containers-operator** 命名空间中。

以下 **peerpodConfig** CR 示例显示默认的 **spec** 值：

```
apiVersion: confidentialcontainers.org/v1alpha1
kind: PeerPodConfig
metadata:
  name: peerpodconfig-openshift
  namespace: openshift-sandboxed-containers-operator
spec:
  cloudSecretName: peer-pods-secret
  configMapName: peer-pods-cm
  limit: "10" (1)
  nodeSelector:
    node-role.kubernetes.io/kata-oc: ""
```

-

1 默认限制为每个节点 10 个虚拟机。

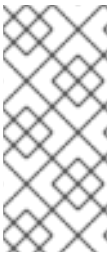
扩展资源名为 **kata.peerpods.io/vm**，并允许 Kubernetes 调度程序处理容量跟踪和核算。

您可以根据环境的要求编辑每个节点的限制。如需更多信息，请参阅修改对等 pod 中的每个节点的虚拟机限制。

变异 Webhook 将扩展的资源 **kata.peerpods.io/vm** 添加到 pod 规格中。如果存在，它还会从 pod 规格中删除任何特定于资源的条目。这可让 Kubernetes 调度程序考虑这些扩展资源，确保仅在资源可用时调度 peer-pod。

变异 Webhook 修改 Kubernetes pod，如下所示：

- 变异 Webhook 会检查 pod 是否有预期的 **RuntimeClassName** 值，在 **TARGET_RUNTIME_CLASS** 环境变量中指定。如果 pod 规格中的值与 **TARGET_RUNTIME_CLASS** 的值不匹配，则 Webhook 会在不修改 pod 的情况下退出。
- 如果 **RuntimeClassName** 值匹配，webhook 会对 pod 规格进行以下更改：
  1. Webhook 从 pod 中所有容器和 init 容器的 **resources** 字段中删除每个资源规格。
  2. Webhook 通过修改 pod 中第一个容器的 resources 字段，将扩展资源 (**kata.peerpods.io/vm**)添加到 spec。Kubernetes 调度程序使用扩展资源 **kata.peerpods.io/vm** 用于核算目的。



注意

变异 Webhook 排除了 Red Hat OpenShift 中的特定系统命名空间。如果在这些系统命名空间中创建了一个 peer-pod，则使用 Kubernetes 扩展资源的资源核算无法正常工作，除非 pod 规格包含扩展的资源。

作为最佳实践，请定义一个集群范围的策略，仅允许在特定命名空间中创建对等 pod。

3.1.1.1. 修改每个节点的对等 pod VM 限制

您可以通过编辑 **peerpodConfig** 自定义资源(CR)来更改每个节点的对等 pod 虚拟机限制。

流程

1. 运行以下命令检查当前的限制：

```
$ oc get peerpodconfig peerpodconfig-openshift -n openshift-sandboxed-containers-operator
\
-o jsonpath='{.spec.limit}{"\n"}'
```

2. 运行以下命令，修改 **peerpodConfig** CR 的 **limit** 属性：

```
$ oc patch peerpodconfig peerpodconfig-openshift -n openshift-sandboxed-containers-
operator \
--type merge --patch '{"spec":{"limit":"<value>"}}'
```

1

1 将 <value> 替换为您要定义的限制。

3.1.2. 使用 AWS 的 peer-pods 的先决条件

如果使用 AWS 创建对等 pod，您必须确保以下要求：

- 您的 Red Hat OpenShift 集群必须安装在 AWS 上，至少有一个 worker 节点。
- 您可以访问 **AWS_ACCESS_KEY_ID** 和 **AWS_SECRET_ACCESS_KEY** 凭据。它们用于在集群的同一虚拟私有云(VPC)中创建额外的云实例。
- 您必须安装并配置 AWS CLI 工具。
- 您必须在端口 15150 和 9000 上启用内部集群通信。
  您可以在 AWS Web 控制台或使用 CLI 中启用这些端口。

3.1.2.1. 为 AWS 启用端口 15150 和 9000

流程

1. 检索实例 ID：

```
$ INSTANCE_ID=$(oc get nodes -l 'node-role.kubernetes.io/worker' -o
jsonpath='{.items[0].spec.providerID}' | sed 's#[^ ]*/##g')
```

2. 检索 AWS 区域：

```
$ AWS_REGION=$(oc get infrastructure/cluster -o
jsonpath='{.status.platformStatus.aws.region}')
```

3. 检索安全组：

```
$ SG=$(aws ec2 describe-instances --instance-ids ${INSTANCE_ID} --query
'Reservations[*].Instances[*].SecurityGroups[*].GroupId' --output text --region
${AWS_REGION})
```

4. 授权 peer-pods shim 并访问 kata-agent 通信。运行以下命令：

```
$ aws ec2 authorize-security-group-ingress --group-id ${SG} --protocol tcp --port 15150 --
source-group ${SG} --region ${AWS_REGION}
```

5. 设置 peer-pods 隧道。运行以下命令：

```
$ aws ec2 authorize-security-group-ingress --group-id ${SG} --protocol tcp --port 9000 --
source-group ${SG} --region ${AWS_REGION}
```

现在启用这些端口。

3.1.3. 使用 Azure 的 peer-pods 的先决条件

如果使用 Microsoft Azure 创建对等 pod，您必须确保以下要求：

- 您的 Red Hat OpenShift 集群必须安装在 Azure 上，至少有一个 worker 节点。
- 您可以访问以下凭证和订阅详情：

- **AZURE_SUBSCRIPTION_ID**
- **AZURE_CLIENT_ID**
- **AZURE_CLIENT_SECRET**
- **AZURE_TENANT_ID**

它们用于在集群的同一虚拟私有云(VPC)中创建额外的云实例。

- 已安装并配置了 Azure CLI 工具。
- 您必须在端口 15150 和 9000 上启用集群通信。
  在 Azure 中，这些端口上允许内部通信。但是，如果通信被阻止，您可以在 Azure web 控制台中或使用 CLI 启用端口。

3.1.3.1. 为 Azure 启用端口 15150 和 9000

流程

1. 检索实例 ID：

```
$ INSTANCE_ID=$(oc get nodes -l 'node-role.kubernetes.io/worker' -o
jsonpath='{.items[0].spec.providerID}' | sed 's#[^ ]*/##g')
```

2. 检索 Azure 资源组：

```
$ AZURE_RESOURCE_GROUP=$(oc get infrastructure/cluster -o
jsonpath='{.status.platformStatus.azure.resourceGroupName}')
```

3. 检索 Azure 网络安全组(NSG)名称：

```
$ AZURE_NSG_NAME=$(az network nsg list --resource-group
${AZURE_RESOURCE_GROUP} --query "[].{Name:name}" --output tsv)
```

4. 检索 Azure VNet 名称：

```
$ AZURE_VNET_NAME=$(az network vnet list --resource-group
${AZURE_RESOURCE_GROUP} --query "[].{Name:name}" --output tsv)
```

5. 检索 Azure 子网名称：

```
$ AZURE_SUBNET_NAME=$(az network vnet subnet list --resource-group
${AZURE_RESOURCE_GROUP} --vnet-name ${AZURE_VNET_NAME}  --query "[].
{Name:name} | [? contains(Name, 'worker')]" --output tsv)
```

6. 检索 Azure 子网前缀：

```
$ AZURE_SUBNET_PREFIX=$(az network vnet subnet show --name
${AZURE_SUBNET_NAME} --vnet-name ${AZURE_VNET_NAME} --resource-group
${AZURE_RESOURCE_GROUP} --query "addressPrefix" --output tsv)
```

7. 授权 peer-pods shim 访问 kata-agent 通信。运行以下命令：

```
$ az network nsg rule create \
  --resource-group $AZURE_RESOURCE_GROUP \
  --nsg-name $AZURE_NSG_NAME \
  --name Allow-Kata-Agent-Internal \
  --access Allow \
  --protocol Tcp \
  --direction Inbound \
  --priority 112 \
  --source-address-prefixes $AZURE_SUBNET_PREFIX \
  --source-port-range "*" \
  --destination-address-prefixes $AZURE_SUBNET_PREFIX  \
  --destination-port-range 15150
```

8. 设置 peer-pods 隧道。运行以下命令：

```
$ az network nsg rule create \
  --resource-group $AZURE_RESOURCE_GROUP \
  --nsg-name $AZURE_NSG_NAME \
  --name Allow-VXLAN-Internal \
  --access Allow \
  --protocol Tcp \
  --direction Inbound \
  --priority 111 \
  --source-address-prefixes $AZURE_SUBNET_PREFIX \
  --source-port-range "*" \
  --destination-address-prefixes $AZURE_SUBNET_PREFIX  \
  --destination-port-range 9000
```

现在启用这些端口。

3.2. 使用带有 WEB 控制台的对等 POD 部署 OPENSHIFT 沙盒容器工作负载

您可从 web 控制台部署 OpenShift 沙盒容器工作负载。首先，您必须安装 OpenShift 沙盒容器 Operator，然后创建一个 secret 对象、虚拟机镜像和 peer-pod ConfigMap。secret 对象和 ConfigMap 是唯一的，具体取决于您的云供应商。最后，您必须创建 **KataConfig** 自定义资源(CR)。在沙盒容器中部署工作负载后，您必须手动将 **kata-remote-cc** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

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

您可从 Red Hat OpenShift Web 控制台安装 OpenShift 沙盒容器 Operator。

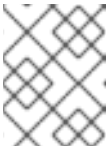
先决条件

- 已安装 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 从 Web 控制台中的 **Administrator** 视角，进入到 **Operators → OperatorHub**。
2. 在 **Filter by keyword** 字段中，输入 **OpenShift sandboxed containers**。

3. 选择 **OpenShift sandboxed containers** 标题。
4. 阅读 Operator 信息并单击 **Install**。
5. 在 **Install Operator** 页面中：
   a. 从可用 **Update Channel** 选项列表中选择 **stable**。
   b. 验证为 **Installed Namespace** 选择了 **Operator recommended Namespace**。这会在 **openshift-sandboxed-containers-operator** 命名空间中安装 Operator。如果此命名空间尚不存在，则会自动创建。



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. 验证是否为 **Approval Strategy** 选择了 **Automatic**。**Automatic** 是默认值，当有新的 z-stream 发行版本可用时，自动启用对 OpenShift 沙盒容器的自动更新。
6. 点 **Install**。

OpenShift 沙盒容器 Operator 现已安装在集群中。

验证

1. 从 Web 控制台中的 **Administrator** 视角，导航到 **Operators → Installed Operators**。
2. 验证 OpenShift 沙盒容器 Operator 是否在 operator 列表中列出。

3.2.2. 使用 Web 控制台为 AWS 配置 peer-pod 参数

您必须创建一个 secret 对象和 ConfigMap，以使用 AWS 上的对等 pod 部署 OpenShift 沙盒容器。

创建 secret 对象后，您仍必须创建 **KataConfig** 自定义资源(CR)来使用对等 pod 部署 OpenShift 沙盒容器。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。

3.2.2.1. 使用 Web 控制台为 AWS 创建 secret 对象

设置 AWS 访问密钥并在 secret 对象中配置网络。secret 对象用于创建 pod 虚拟机镜像，并由对等 pod 使用。

为 AWS 创建 secret 对象时，您必须设置特定的环境值。在创建 secret 对象前，您可以检索其中的一些值。必须使用 CLI 检索这些值。如需更多信息，请参阅使用 CLI 为 AWS 创建 secret 对象 。

另外，在 AWS Web 控制台中，您必须找到并准备以下值：

- **AWS_ACCESS_KEY_ID**
- **AWS_SECRET_ACCESS_KEY**

流程

1. 从 Web 控制台中的 **Administrator** 视角，导航到 **Operators → Installed Operators**。
2. 从 Operator 列表中选择 OpenShift 沙盒容器 Operator。
3. 单击右上角的 Import 图标 (+)。
4. 在 **Import YAML** 窗口中，粘贴以下 YAML 清单：

```
apiVersion: v1
kind: Secret
metadata:
  name: peer-pods-secret
  namespace: openshift-sandboxed-containers-operator
type: Opaque
stringData:
  AWS_ACCESS_KEY_ID: "<enter value>" 1
  AWS_SECRET_ACCESS_KEY: "<enter value>" 2
  AWS_REGION: "<enter value>" 3
  AWS_SUBNET_ID: "<enter value>" 4
  AWS_VPC_ID: "<enter value>" 5
  AWS_SG_IDS: "<enter value>" 6
```

1. 在开始之前，输入您准备的 **AWS_ACCESS_KEY_ID** 值。
2. 在开始之前，输入您准备的 **AWS_SECRET_ACCESS_KEY** 值。
3. 输入您检索到的 **AWS_REGION** 值。
4. 输入您检索到的 **AWS_SUBNET_ID** 值。
5. 输入您检索到的 **AWS_VPC_ID** 值。
6. 输入您检索到的 **AWS_SG_IDS** 值。

5. 点 **Create**。

secret 对象已创建。您可以看到它列在 **Workloads → Secrets** 下。

3.2.2.2. 使用 Web 控制台创建 AWS 虚拟机镜像(AMI)

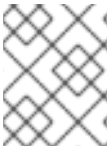
要使用 AWS 上的对等 pod 运行 OpenShift 沙盒容器，您必须首先使用 AWS 帐户和资源创建一个 RHEL AMI。

流程

1. 从 web 控制台中的 **Administrator** 视角，进入到 **Workloads → Jobs**。

2. 在 **Jobs** 窗口中，在左上角的 openshift-sandboxed-containers-operator 项目中验证您是否处于 **openshift-sandboxed-containers-operator** 项目中。
3. 点 **Create Job**。
4. 在 **Create Job** 窗口中，粘贴此完整的 YAML 清单。
5. 点 **Create**。

已创建镜像。



注意

此镜像不由 OpenShift 沙盒容器管理。您可以使用 AWS Web 控制台或 AWS CLI 工具删除它。

创建镜像后，您必须使用 peer-pod ConfigMap 设置镜像。

3.2.2.3. 使用 Web 控制台为 AWS 创建 peer-pod ConfigMap

为 AWS 创建 ConfigMap 时，您必须设置 AMI ID。在创建 ConfigMap 前，您可以检索这个值。必须使用 CLI 检索此值。如需更多信息，请参阅使用 CLI 为 AWS 创建 peer-pod ConfigMap 。

流程

1. 从 Web 控制台中的 **Administrator** 视角，导航到 **Operators → Installed Operators**。
2. 从 Operator 列表中选择 OpenShift 沙盒容器 Operator。
3. 单击右上角的 Import 图标 (+)。
4. 在 **Import YAML** 窗口中，粘贴以下 YAML 清单：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: peer-pods-cm
  namespace: openshift-sandboxed-containers-operator
data:
  CLOUD_PROVIDER: "aws"
  VXLAN_PORT: "9000"
  PODVM_INSTANCE_TYPE: "t3.medium"
  PROXY_TIMEOUT: "5m"
  PODVM_AMI_ID: "<enter value>" (1)
```

(1) 输入您检索到的 **PODVM_AMI_ID** 值。

5. 点 **Create**。

ConfigMap 对象已创建。您可以看到它在 **Workloads → ConfigMaps** 下列出。

创建 **KataConfig** CR 后，您可以使用 AWS 上的对等 pod 运行 OpenShift 沙盒容器。

3.2.3. 使用 Web 控制台为 Azure 配置 peer-pod 参数

您必须创建一个 secret 对象和 ConfigMap，以便使用 Microsoft Azure 上的对等 pod 部署 OpenShift 沙盒容器。

创建 secret 对象后，您仍必须创建 **KataConfig** 自定义资源(CR)来使用对等 pod 部署 OpenShift 沙盒容器。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。

3.2.3.1. 使用 Web 控制台为 Azure 创建 secret 对象

设置 Azure 访问密钥并在 secret 对象中配置网络。secret 对象用于创建 pod 虚拟机镜像，并由对等 pod 使用。

为 Azure 创建 secret 对象时，您必须设置特定的环境值。在创建 secret 对象前，您可以检索其中的一些值。必须使用 CLI 检索这些值。如需更多信息，请参阅使用 CLI 为 Azure 创建 secret 对象 。

另外，在 Azure web 控制台中，您必须找到并准备以下值：

- **AZURE_CLIENT_ID**
- **AZURE_CLIENT_SECRET**
- **AZURE_TENANT_ID**

流程

1. 从 Web 控制台中的 **Administrator** 视角，导航到 **Operators → Installed Operators**。
2. 从 Operator 列表中选择 OpenShift 沙盒容器 Operator。
3. 单击右上角的 Import 图标 (+)。
4. 在 **Import YAML** 窗口中，粘贴以下 YAML 清单：

```
apiVersion: v1
kind: Secret
metadata:
  name: peer-pods-secret
  namespace: openshift-sandboxed-containers-operator
type: Opaque
stringData:
  AZURE_CLIENT_ID: "<enter value>" 1
  AZURE_CLIENT_SECRET: "<enter value>" 2
  AZURE_TENANT_ID: "<enter value>" 3
  AZURE_SUBSCRIPTION_ID: "<enter value>" 4
  AZURE_REGION: "<enter value>" 5
  AZURE_RESOURCE_GROUP: "<enter value>" 6
```

1 在开始之前输入您准备的 **AZURE_CLIENT_ID** 值。

2. 在开始之前，输入您准备的 **AZURE_CLIENT_SECRET** 值。
3. 在开始之前输入您准备的 **AZURE_TENANT_ID** 值。
4. 输入您检索到的 **AZURE_SUBSCRIPTION_ID** 值。
5. 输入您检索到的 **AZURE_REGION** 值。
6. 输入您检索到的 **AZURE_RESOURCE_GROUP** 值。

5. 点 **Create**。

secret 对象已创建。您可以看到它列在 **Workloads → Secrets** 下。

3.2.3.2. 使用 Web 控制台创建 Azure 虚拟机镜像

要使用 Azure 上的对等 pod 运行 OpenShift 沙盒容器，您必须首先使用 Azure 帐户和资源为 Azure 创建 RHEL 镜像。

流程

1. 从 web 控制台中的 **Administrator** 视角，进入到 **Workloads → Jobs**。
2. 在 **Jobs** 窗口中，在左上角的 openshift-sandboxed-containers-operator 项目中验证您是否处于 openshift-sandboxed-containers-operator 项目中。
3. 点 **Create Job**。
4. 在 **Create Job** 窗口中，粘贴此完整的 YAML 清单。
5. 点 **Create**。

已创建镜像。

注意

此镜像不由 OpenShift 沙盒容器管理。如果需要，可以使用 Azure web 控制台或 Azure CLI 工具删除它。

创建镜像后，您必须使用 peer-pod ConfigMap 设置镜像。

3.2.3.3. 使用 Web 控制台为 Azure 创建 peer-pod ConfigMap

为 Azure 创建 ConfigMap 时，您必须设置特定的配置值。必须使用 CLI 检索这些值。如需更多信息，请参阅使用 CLI 为 Azure 创建 peer-pod ConfigMap 。

流程

1. 从 Web 控制台中的 **Administrator** 视角，导航到 **Operators → Installed Operators**。
2. 从 Operator 列表中选择 OpenShift 沙盒容器 Operator。
3. 单击右上角的 Import 图标 (+)。

4. 在 **Import YAML** 窗口中，粘贴以下 YAML 清单：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: peer-pods-cm
  namespace: openshift-sandboxed-containers-operator
data:
  CLOUD_PROVIDER: "azure"
  VXLAN_PORT: "9000"
  AZURE_INSTANCE_SIZE: "Standard_B2als_v2"
  AZURE_SUBNET_ID: "<enter value>" (1)
  AZURE_NSG_ID: "<enter value>" (2)
  AZURE_IMAGE_ID: "<enter value>" (3)
  PROXY_TIMEOUT: "5m"
  DISABLECVM: "true"
```

(1) 输入您检索到的 **AZURE_SUBNET_ID** 值。

(2) 输入您检索到的 **AZURE_NSG_ID** 值。

(3) 输入您检索到的 **AZURE_IMAGE_ID** 值。

5. 点 **Create**。

ConfigMap 对象已创建。您可以看到它在 **Workloads → ConfigMaps** 下列出。

3.2.3.4. 使用 Web 控制台为 Azure 创建 SSH 密钥 secret 对象

您必须创建一个 SSH 密钥 secret 对象才能使用 Azure 的对等 pod。如果您还没有创建对象的 SSH 密钥，则必须使用 CLI 生成对象。更多信息，请参阅

流程

1. 从 web 控制台中的 **Administrator** 视角，进入到 **Workloads → Secrets**。
2. 在左侧的 **Secrets** 窗口中，验证您是否位于 **openshift-sandboxed-containers-operator** 项目中。
3. 点 **Create**，从列表中选择 **Key/value secret**。
4. 在 **Secret name** 字段中，输入 **ssh-key-secret**。
5. 在 **Key** 字段中，输入 **id_rsa.pub**。
6. 在 **Value** 字段中，粘贴您的公共 SSH 密钥。
7. 点 **Create**。

SSH 密钥 secret 对象已创建。您可以看到它列在 **Workloads → Secrets** 下。

创建 **KataConfig** CR 后，您可以使用 Azure 上的对等 pod 运行 OpenShift 沙盒容器。

3.2.4. 在 web 控制台中创建 KataConfig 自定义资源

您必须创建一个 **KataConfig** 自定义资源(CR)，以便在集群节点上启用将 **kata-remote-cc** 安装为 **RuntimeClass**。

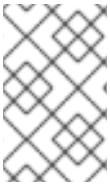
重要

创建 **KataConfig** CR 会自动重启 worker 节点。重启可能需要 10 到 60 分钟。妨碍重启时间的因素如下：

- 带有更多 worker 节点的大型 Red Hat OpenShift 部署。
- 激活 BIOS 和 Diagnostics 实用程序。
- 在硬盘而不是 SSD 上部署。
- 在物理节点上部署，如裸机，而不是在虚拟节点上部署。
- CPU 和网络较慢。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。



注意

默认情况下，对等 pod 的 Kata 安装在所有 worker 节点上。如果要在特定节点上安装 **kata-remote-cc** 作为 **RuntimeClass**，您可以在这些节点上添加标签，然后在创建时在 **KataConfig** CR 中定义标签。

流程

1. 从 Web 控制台中的 **Administrator** 视角，导航到 **Operators → Installed Operators**。
2. 从 Operator 列表中选择 OpenShift 沙盒容器 Operator。
3. 在 **KataConfig** 选项卡中，点 **Create KataConfig**。
4. 在 **Create KataConfig** 页面中，输入以下详情：
   - **名称** ：输入 **KataConfig** 资源的名称。默认情况下，名称定义为 **example-kataconfig**。
   - **Labels** （可选）：输入任何相关的、识别到 **KataConfig** 资源的属性。每个标签代表一个键值对。
   - **checkNodeEligibility** （可选，不与对等 pod 相关的）：选择此复选框以使用 Node Feature Discovery Operator (NFD)检测节点资格以作为 **RuntimeClass** 运行 **kata**。如需更多信息，请参阅"检查集群节点是否有资格运行 OpenShift 沙盒容器"。
   - **enablePeerPods** （对于对等 pod）：选择此复选框来启用对等 pod，并在公有云环境中使用 OpenShift 沙盒容器。

- **KataConfigPoolSelector** ：默认情况下，**kata-remote-cc** 在所有节点上作为 **RuntimeClass** 安装。如果要在所选节点上安装 **kata-remote-cc** 作为 **RuntimeClass**，您必须添加一个 matchExpression ：

  a. 展开 **kataConfigPoolSelector** 区域。

  b. 在 **kataConfigPoolSelector** 中，展开 matchExpressions。这是标签选择器要求列表。

  c. 点 Add matchExpressions。

  d. 在 key 字段中，添加选择器应用到的标签键。

  e. 在 operator 字段中，添加键与标签值的关系。有效的运算符为 **In**、**NotIn**、**Exists** 和 **DoesNotExist**。

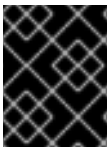
  f. 展开 values 区域，然后点 Add value。

  g. 在 Value 字段中，为 key 标签值输入 **true** 或 **false**。

- **loglevel** ：定义为将 **kata-remote-cc** 运行为 **RuntimeClass** 的节点检索的日志数据级别。如需更多信息，请参阅"收集 OpenShift 沙盒容器数据"。

5. 点 Create。

创建新的 **KataConfig** CR，并开始在 worker 节点上安装 **kata-remote-cc** 作为 **RuntimeClass**。等待安装完成，以及 worker 节点重新引导，然后继续下一步。



重要

OpenShift 沙盒容器仅将 **kata-remote-cc** 安装为集群上的辅助可选运行时，而不是主运行时。

验证

1. 在 KataConfig 选项卡中，选择新的 **KataConfig** CR。
2. 在 KataConfig 页面中，选择 YAML 选项卡。
3. 监控状态中的 **installationStatus** 字段。
   每次有更新时都会出现一条消息。点 Reload 查看更新的 **KataConfig** CR。

   当 **Completed nodes** 的值等于 worker 或已标记的节点的数量，则代表安装已完成。该状态还包含安装完成的节点的列表。

3.2.5. 使用 Web 控制台在沙盒容器中部署对等 pod 的工作负载

OpenShift 沙盒容器将 Kata 安装为集群上的辅助、可选运行时，而不是主运行时。

要使用沙盒容器中的对等 pod 部署 pod 模板工作负载，您必须手动将 **kata-remote-cc** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

- 已安装 OpenShift 沙盒容器 Operator。
- 您已创建了一个 secret 对象和 peer-pod 配置映射，对云供应商是唯一的。
- 您已创建了 **KataConfig** 自定义资源 (CR)。

流程

1. 从 web 控制台中的 **Administrator** 视角，展开 **Workloads** 并选择您要创建的工作负载类型。
2. 在工作负载页面中，点击以创建工作负载。
3. 在工作负载的 YAML 文件中，在列出容器的 **spec** 字段中，添加 **runtimeClassName: kata-remote-cc**。

   Pod 对象示例

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-openshift
  labels:
    app: hello-openshift
spec:
  runtimeClassName: kata-remote-cc
  containers:
    - name: hello-openshift
      image: quay.io/openshift/origin-hello-openshift
      ports:
        - containerPort: 8888
      securityContext:
        privileged: false
        allowPrivilegeEscalation: false
        runAsNonRoot: true
        runAsUser: 1001
        capabilities:
          drop:
            - ALL
        seccompProfile:
          type: RuntimeDefault
```

4. 点击 **Save**。

Red Hat OpenShift 创建工作负载并开始调度它。

3.3. 使用带有 CLI 的对等 POD 部署 OPENSHIFT 沙盒容器工作负载

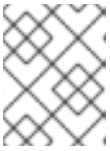
您可以使用 CLI 部署 OpenShift 沙盒容器工作负载。首先，您必须安装 OpenShift 沙盒容器 Operator，然后创建 **KataConfig** 自定义资源。在沙盒容器中部署工作负载后，您必须将 **kata-remote-cc** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

3.3.1. 使用 CLI 安装 OpenShift 沙盒容器 Operator

您可以使用 Red Hat OpenShift CLI 安装 OpenShift 沙盒容器 Operator。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 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. 为 OpenShift 沙盒容器 Operator 创建 **Namespace** 对象。
   a. 创建一个包含以下清单的 **Namespace** 对象 YAML 文件：

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

   b. 创建 **Namespace** 对象：

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

2. 为 OpenShift 沙盒容器 Operator 创建 **OperatorGroup** 对象。
   a. 创建一个包含以下清单的 **OperatorGroup** 对象 YAML 文件：

```
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. 创建 **OperatorGroup** 对象：

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

3. 创建 **Subscription** 对象，以便为 OpenShift 沙盒容器 Operator 订阅 **命名空间**。
   a. 创建一个包含以下内容的 **Subscription** 对象 YAML 文件：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-sandboxed-containers-operator
```

```
  namespace: openshift-sandboxed-containers-operator
spec:
  channel: stable
  installPlanApproval: Automatic
  name: sandboxed-containers-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: sandboxed-containers-operator.v1.4.1
```

b. 创建 **Subscription** 对象：

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

OpenShift 沙盒容器 Operator 现已安装在集群中。

注意

以上列出的所有对象文件名都是建议。您可以使用其他名称创建对象 YAML 文件。

验证

- 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  1.4.1    1.4.0
Succeeded
```

其他资源

- 使用 CLI 从 OperatorHub 安装

3.3.2. 使用 CLI 为 AWS 设置对等 pod

要设置在 AWS 上使用的对等 pod，您必须创建一个 secret 对象、AWS 镜像虚拟机(AMI)和 peer-pod ConfigMap。

为 AWS 设置对等 pod 后，您仍必须创建 **KataConfig** 自定义资源(CR)来使用 peer pod 部署 OpenShift 沙盒容器。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。

3.3.2.1. 使用 CLI 为 AWS 创建 secret 对象

设置 AWS 访问密钥并在 secret 对象中配置网络。secret 对象用于创建 pod 虚拟机镜像，并由对等 pod 使用。

为 AWS 创建 secret 对象时，您必须设置特定的环境值。在创建 secret 对象前，您可以检索其中的一些值。但是，您必须准备以下值：

- **AWS_ACCESS_KEY_ID**
- **AWS_SECRET_ACCESS_KEY**

流程

1. 收集 secret 对象所需的参数值。确保记下每个值。
   a. 检索实例 ID：

   ```
   $ INSTANCE_ID=$(oc get nodes -l 'node-role.kubernetes.io/worker' -o
   jsonpath='{.items[0].spec.providerID}' | sed 's#[^ ]*/##g')
   ```

   secret 对象本身不需要这个值，而是用于检索 secret 对象的其他值。

   b. 检索 AWS 区域：

   ```
   $ AWS_REGION=$(oc get infrastructure/cluster -o
   jsonpath='{.status.platformStatus.aws.region}') && echo "AWS_REGION:
   \"$AWS_REGION\""
   ```

   c. 检索 AWS 子网 ID：

   ```
   $ AWS_SUBNET_ID=$(aws ec2 describe-instances --instance-ids ${INSTANCE_ID} --
   query 'Reservations[*].Instances[*].SubnetId' --region ${AWS_REGION} --output text) &&
   echo "AWS_SUBNET_ID: \"$AWS_SUBNET_ID\""
   ```

   d. 检索 AWS VPC ID：

   ```
   $ AWS_VPC_ID=$(aws ec2 describe-instances --instance-ids ${INSTANCE_ID} --query
   'Reservations[*].Instances[*].VpcId' --region ${AWS_REGION} --output text) && echo
   "AWS_VPC_ID: \"$AWS_VPC_ID\""
   ```

   e. 检索 AWS 安全组 ID：

   ```
   $ AWS_SG_IDS=$(aws ec2 describe-instances --instance-ids ${INSTANCE_ID} --query
   'Reservations[*].Instances[*].SecurityGroups[*].GroupId' --region ${AWS_REGION} --
   output text)
   && echo "AWS_SG_IDS: \"$AWS_SG_IDS\""
   ```

2. 使用以下清单创建 YAML 文件：

```
apiVersion: v1
kind: Secret
metadata:
  name: peer-pods-secret
```

```
  namespace: openshift-sandboxed-containers-operator
type: Opaque
stringData:
  AWS_ACCESS_KEY_ID: "<enter value>" 1
  AWS_SECRET_ACCESS_KEY: "<enter value>" 2
  AWS_REGION: "<enter value>" 3
  AWS_SUBNET_ID: "<enter value>" 4
  AWS_VPC_ID: "<enter value>" 5
  AWS_SG_IDS: "<enter value>" 6
```

1. 在开始之前，输入您准备的 **AWS_ACCESS_KEY_ID** 值。
2. 在开始之前，输入您准备的 **AWS_SECRET_ACCESS_KEY** 值。
3. 输入您检索到的 **AWS_REGION** 值。
4. 输入您检索到的 **AWS_SUBNET_ID** 值。
5. 输入您检索到的 **AWS_VPC_ID** 值。
6. 输入您检索到的 **AWS_SG_IDS** 值。

3. 应用 secret 对象：

```
$ oc apply -f peer-pods-secret.yaml
```

secret 对象被应用。

3.3.2.2. 使用 CLI 创建 AWS 虚拟机镜像(AMI)

要使用 AWS 上的对等 pod 运行 OpenShift 沙盒容器，您必须首先使用 AWS 帐户和资源创建一个 RHEL AMI。

流程

1. 运行以下 K8s 作业以创建镜像：

```
$ oc apply -f https://raw.githubusercontent.com/openshift/sandboxed-containers-operator/peer-pods-tech-preview/hack/aws-image-job.yaml
```

注意

此镜像不由 OpenShift 沙盒容器管理。您可以使用 AWS Web 控制台或 AWS CLI 工具删除它。

2. 等待作业完成：

```
$ oc wait --for=condition=complete job.batch/aws-image-creation --timeout=7m -n openshift-sandboxed-containers-operator
```

创建镜像后，您必须使用 peer-pod ConfigMap 设置镜像。

3.3.2.3. 使用 CLI 为 AWS 创建 peer-pod ConfigMap

为 AWS 创建 ConfigMap 时，您必须设置 AMI ID。在创建 ConfigMap 前，您可以检索这个值。

流程

1. 检索 AMI ID。确保保存该值以便稍后保存。

```
$ PODVM_AMI_ID=$(aws ec2 describe-images --query "Images[*].[ImageId]" --filters
"Name=name,Values=peer-pod-ami" --region ${AWS_REGION} --output text) && echo
"PODVM_AMI_ID: \"$PODVM_AMI_ID\""
```

2. 使用以下清单创建 YAML 文件：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: peer-pods-cm
  namespace: openshift-sandboxed-containers-operator
data:
  CLOUD_PROVIDER: "aws"
  VXLAN_PORT: "9000"
  PODVM_INSTANCE_TYPE: "t3.medium"
  PROXY_TIMEOUT: "5m"
  PODVM_AMI_ID: "<enter value>" (1)
```

(1) 输入您检索到的 AMI ID 值。

3. 部署 ConfigMap：

```
$ oc apply -f peer-pods-cm.yaml
```

ConfigMap 被部署。创建 **KataConfig** CR 后，您可以使用 AWS 上的对等 pod 运行 OpenShift 沙盒容器。

3.3.3. 使用 CLI 为 Azure 设置对等 pod

要设置在 Microsoft Azure 上使用的对等 pod，您必须创建一个 secret 对象、Azure 镜像虚拟机、peer-pod ConfigMap 和 SSH 密钥 secret 对象。

为 Azure 设置对等 pod 后，您仍必须创建 **KataConfig** 自定义资源(CR)来使用 peer pod 部署 OpenShift 沙盒容器。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。

3.3.3.1. 使用 CLI 为 Azure 创建 secret 对象

设置 Azure 访问密钥并在 secret 对象中配置网络。secret 对象用于创建 pod 虚拟机镜像，并由对等 pod 使用。

为 Azure 创建 secret 对象时，您必须设置特定的环境值。在创建 secret 对象前，您可以检索其中的一些值。但是，您必须准备以下值：

- **AZURE_CLIENT_ID**
- **AZURE_CLIENT_SECRET**
- **AZURE_TENANT_ID**

流程

1. 为 secret 对象收集额外的参数值。确保记下每个值。
   a. 检索订阅 ID：

```
$ AZURE_SUBSCRIPTION_ID=$(az account list --query "[?isDefault].id" -o tsv) && echo
"AZURE_SUBSCRIPTION_ID: \"$AZURE_SUBSCRIPTION_ID\""
```

   b. 检索资源组：

```
$ AZURE_RESOURCE_GROUP=$(oc get infrastructure/cluster -o
jsonpath='{.status.platformStatus.azure.resourceGroupName}') && echo
"AZURE_RESOURCE_GROUP: \"$AZURE_RESOURCE_GROUP\""
```

   c. 检索 Azure 区域：

```
$ AZURE_REGION=$(az group show --resource-group
${AZURE_RESOURCE_GROUP} --query "{Location:location}" --output tsv) && echo
"AZURE_REGION: \"$AZURE_REGION\""
```

2. 使用以下清单创建 YAML 文件：

```
apiVersion: v1
kind: Secret
metadata:
  name: peer-pods-secret
  namespace: openshift-sandboxed-containers-operator
type: Opaque
stringData:
  AZURE_CLIENT_ID: "<enter value>" 1
  AZURE_CLIENT_SECRET: "<enter value>" 2
  AZURE_TENANT_ID: "<enter value>" 3
  AZURE_SUBSCRIPTION_ID: "<enter value>" 4
  AZURE_REGION: "<enter value>" 5
  AZURE_RESOURCE_GROUP: "<enter value>" 6
```

1. 在开始之前输入您准备的 **AZURE_CLIENT_ID** 值。
2. 在开始之前，输入您准备的 **AZURE_CLIENT_SECRET** 值。

3 在开始之前输入您准备的 **AZURE_TENANT_ID** 值。

4 输入您检索到的 **AZURE_SUBSCRIPTION_ID** 值。

5 输入您检索到的 **AZURE_REGION** 值。

6 输入您检索到的 **AZURE_RESOURCE_GROUP** 值。

3. 应用 secret 对象：

```
$ oc apply -f peer-pods-secret.yaml
```

secret 对象被应用。

3.3.3.2. 使用 CLI 创建 Azure 虚拟机镜像

要使用 Azure 上的对等 pod 运行 OpenShift 沙盒容器，您必须首先使用 Azure 帐户和资源为 Azure 创建 RHEL 镜像。

流程

1. 运行以下 K8s 作业以创建镜像：

```
$ oc apply -f https://raw.githubusercontent.com/openshift/sandboxed-containers-operator/peer-pods-tech-preview/hack/azure-image-job.yaml
```

注意

此镜像不由 OpenShift 沙盒容器管理。如果需要，可以使用 Azure web 控制台或 Azure CLI 工具删除它。

2. 等待作业完成：

```
$ oc wait --for=condition=complete job.batch/azure-image-creation --timeout=7m -n openshift-sandboxed-containers-operator
```

创建镜像后，您必须使用 peer-pod ConfigMap 设置镜像。

3.3.3.3. 使用 CLI 为 Azure 创建 peer-pod ConfigMap

为 Azure 创建 ConfigMap 时，您必须设置特定的配置值。在创建 ConfigMap 前，您可以检索这些值。

流程

1. 收集 Azure peer-pod ConfigMap 的配置值。确保记下每个值。

   a. 检索 Azure 镜像 ID：

```
$ AZURE_IMAGE_ID=$(az image list --resource-group ${AZURE_RESOURCE_GROUP} --query "[].{Id: id} | [? contains(Id, 'peer-pod-vmimage')]" --output tsv) && echo "AZURE_IMAGE_ID: \"$AZURE_IMAGE_ID\""
```

b. 检索 Azure VNet 名称：

```
$ AZURE_VNET_NAME=$(az network vnet list --resource-group
${AZURE_RESOURCE_GROUP} --query "[].{Name:name}" --output tsv)
```

ConfigMap 不需要这个值，但用于检索 Azure 子网 ID。

c. 检索 Azure 子网 ID：

```
$ AZURE_SUBNET_ID=$(az network vnet subnet list --resource-group
${AZURE_RESOURCE_GROUP} --vnet-name $AZURE_VNET_NAME --query "[].{Id:id}
| [? contains(Id, 'worker')]" --output tsv) && echo "AZURE_SUBNET_ID:
\"$AZURE_SUBNET_ID\""
```

d. 检索 Azure 网络安全组(NSG) ID：

```
$ AZURE_NSG_ID=$(az network nsg list --resource-group
${AZURE_RESOURCE_GROUP} --query "[].{Id:id}" --output tsv) && echo
"AZURE_NSG_ID: \"$AZURE_NSG_ID\""
```

2. 使用以下清单创建 YAML 文件：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: peer-pods-cm
  namespace: openshift-sandboxed-containers-operator
data:
  CLOUD_PROVIDER: "azure"
  VXLAN_PORT: "9000"
  AZURE_INSTANCE_SIZE: "Standard_B2als_v2"
  AZURE_SUBNET_ID: "<enter value>" (1)
  AZURE_NSG_ID: "<enter value>" (2)
  AZURE_IMAGE_ID: "<enter value>" (3)
  PROXY_TIMEOUT: "5m"
  DISABLECVM: "true"
```

1. 输入您检索到的 **AZURE_SUBNET_ID** 值。
2. 输入您检索到的 **AZURE_NSG_ID** 值。
3. 输入您检索到的 **AZURE_IMAGE_ID** 值。

3. 部署 ConfigMap：

```
$ oc apply -f peer-pods-cm.yaml
```

ConfigMap 被部署。

3.3.3.4. 使用 CLI 为 Azure 创建 SSH 密钥 secret 对象

您必须生成 SSH 密钥并创建 SSH 密钥 secret 对象，以便在 Azure 中使用对等 pod。

流程

1. 生成 SSH 密钥：

```
$ ssh-keygen -f ./id_rsa -N ""
```

2. 创建 SHH secret 对象：

```
$ oc create secret generic ssh-key-secret -n openshift-sandboxed-containers-operator --
from-file=id_rsa.pub=./id_rsa.pub --from-file=id_rsa=./id_rsa
```

SSH 密钥已创建，并且创建了 SSH 密钥 secret 对象。创建 **KataConfig** CR 后，您可以使用 Azure 上的对等 pod 运行 OpenShift 沙盒容器。

3.3.4. 使用 CLI 创建 KataConfig 自定义资源

您必须创建一个 **KataConfig** 自定义资源(CR)，以便在节点上安装 **kata-remote-cc** 作为 **RuntimeClass**。创建 **KataConfig** CR 会触发 OpenShift 沙盒容器 Operator 来执行以下操作：

- 在 RHCOS 节点上安装所需的 RHCOS 扩展，如 QEMU 和 **kata-containers**。
- 确保 CRI-O 运行时配置了正确的运行时处理程序。
- 使用默认配置，创建一个名为 **kata-remote-cc** 的 **RuntimeClass** CR。这可让用户在 **RuntimeClassName** 字段中引用 CR 将工作负载配置为使用 **kata-remote-cc** 作为运行时。此 CR 也指定运行时的资源开销。

注意

默认情况下，对等 pod 的 Kata 安装在所有 worker 节点上。如果要在特定节点上安装 **kata-remote-cc** 作为 **RuntimeClass**，您可以在这些节点上添加标签，然后在创建时在 **KataConfig** CR 中定义标签。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。

重要

创建 **KataConfig** CR 会自动重启 worker 节点。重启可能需要 10 到 60 分钟。妨碍重启时间的因素如下：

- 带有更多 worker 节点的大型 Red Hat OpenShift 部署。
- 激活 BIOS 和 Diagnostics 实用程序。
- 在硬盘而不是 SSD 上部署。
- 在物理节点上部署，如裸机，而不是在虚拟节点上部署。
- CPU 和网络较慢。

流程

1. 使用以下清单创建 YAML 文件：

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
spec:
  enablePeerPods: true
  logLevel: info
```

2. （可选）如果您要仅在所选节点上安装 **kata-remote-cc** 作为 **RuntimeClass**，请创建一个包含清单中的标签的 YAML 文件：

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

(1) **kataConfigPoolSelector** 中的标签只支持单个值；不支持 **nodeSelector** 语法。

3. 创建 **KataConfig** 资源：

```
$ oc create -f cluster-kataconfig.yaml
```

创建新的 **KataConfig** CR，并开始在 worker 节点上安装 **kata-remote-cc** 作为 **RuntimeClass**。等待 **kata-remote-cc** 安装完成，以及 worker 节点重新引导，然后继续下一步。

重要

OpenShift 沙盒容器仅将 **kata-remote-cc** 安装为集群上的辅助可选运行时，而不是主运行时。

验证

- 监控安装进度：

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

当 **Is In Progress**的值显示为 **false** 后，安装就已完成。

其他资源

- 了解如何更新节点上的标签

3.3.5. 使用 CLI 在沙盒容器中部署对等 pod 的工作负载

OpenShift 沙盒容器将 Kata 安装为集群上的辅助、可选运行时，而不是主运行时。

要使用沙盒容器中的对等 pod 部署 pod 模板工作负载，您必须将 **kata-remote-cc** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。
- 您已创建了一个 secret 对象和 peer-pod 配置映射，对云供应商是唯一的。
- 您已创建了 **KataConfig** 自定义资源 (CR)。

流程

- 将 **runtimeClassName: kata-remote-cc** 添加到任何 pod 模板对象中：
  - **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: hello-openshift
  labels:
```

```
    app: hello-openshift
spec:
  runtimeClassName: kata-remote-cc
  containers:
    - name: hello-openshift
      image: quay.io/openshift/origin-hello-openshift
      ports:
        - containerPort: 8888
      securityContext:
        privileged: false
        allowPrivilegeEscalation: false
        runAsNonRoot: true
        runAsUser: 1001
        capabilities:
          drop:
            - ALL
        seccompProfile:
          type: RuntimeDefault
```

Red Hat OpenShift 创建工作负载并开始调度它。

验证

- 检查 pod 模板对象上的 **runtimeClassName** 字段。如果 **runtimeClassName** 是 **kata-remote-cc**，则工作负载使用对等 pod 在 OpenShift 沙盒容器上运行。

3.4. 其他资源

- OpenShift 沙盒容器 Operator 在受限网络环境中被支持。如需更多信息，请参阅在受限网络中使用 Operator Lifecycle Manager。
- 在受限网络中使用断开连接的集群时，您必须在 Operator Lifecycle Manager 中配置代理支持来访问 OperatorHub。使用代理允许集群获取 OpenShift 沙盒容器 Operator。

第 4 章 监控 OPENSHIFT 沙盒容器

您可以使用 Red Hat OpenShift Web 控制台监控与沙盒工作负载和节点的健康状态相关的指标。

OpenShift 沙盒容器在 web 控制台中有一个预先配置的仪表板，管理员还可以通过 Prometheus 访问和查询原始指标。

4.1. 关于 OPENSHIFT 沙盒容器指标

OpenShift 沙盒容器指标让管理员能够监控沙盒容器的运行方式。您可以在 web 控制台中的 Metrics UI 中查询这些指标。

OpenShift 沙盒容器指标为以下类别收集：

Kata 代理指标

Kata 代理指标显示有关嵌入在沙盒容器中运行的 kata 代理进程的信息。这些指标包括 **/proc/<pid>/[io, stat, status]** 中的数据。

Kata 客户机操作系统指标

Kata 客户机操作系统指标显示沙盒容器中运行的客户机操作系统中的数据。这些指标包括 **/proc/[stats, diskstats, meminfo, vmstats]** 和 **/proc/net/dev** 中的数据。

hypervisor 指标

hypervisor 指标显示有关运行嵌入在沙盒容器中虚拟机的虚拟机监控程序的数据。这些指标主要包括 **/proc/<pid>/[io, stat, status]** 中的数据。

Kata 监控指标

Kata 监控器是收集指标数据并提供给 Prometheus 的进程。kata 监控指标显示有关 kata-monitor 进程本身的资源使用情况的详细信息。这些指标还包括 Prometheus 数据收集的计数器。

Kata containerd shim v2 指标

Kata containerd shim v2 指标显示有关 kata shim 进程的详细信息。这些指标包括来自 **/proc/<pid>/[io, stat, status]** 和详细的资源使用量指标的数据。

4.2. 查看 OPENSHIFT 沙盒容器的指标

您可以在 web 控制台的 **Metrics** 页面中访问 OpenShift 沙盒容器的指标。

先决条件

- 已安装 Red Hat OpenShift 4.13。
- 已安装 OpenShift 沙盒容器。
- 您可以使用具有 **cluster-admin** 角色或所有项目的查看权限的用户访问集群。

流程

1. 从 web 控制台中的 **Administrator** 视角，进入到 **Observe → Metrics**。
2. 在输入字段中，输入您要观察到的指标的查询。
   所有与 kata 相关的指标都以 **kata** 开头。键入 **kata** 将显示含有所有可用 kata 指标的列表。

在页面中会视觉化查询的指标。

其他资源

其他资源

- 有关创建 PromQL 查询来查看指标的更多信息，请参阅 查询指标。

4.3. 查看 OPENSHIFT 沙盒容器仪表板

您可以在 web 控制台的 **Dashboards** 页面中访问 OpenShift 沙盒容器仪表板。

先决条件

- 已安装 Red Hat OpenShift 4.13。
- 已安装 OpenShift 沙盒容器。
- 您可以使用具有 **cluster-admin** 角色或所有项目的查看权限的用户访问集群。

流程

1. 从 web 控制台中的 **Administrator** 视角，进入到 **Observe → Dashboards**。
2. 从 **Dashboard** 下拉列表中，选择 **Sandboxed Containers** 仪表板。
3. 可选：在 **Time Range** 列表中为图形选择一个时间范围。
   - 选择预定义的时间段。
   - 通过选择 **Time Range** 列表中的 **Custom 时间范围** 来设置自定义时间范围。
     a. 定义您要查看的数据的日期和时间范围。
     b. 单击 **Save** 以保存自定义时间范围。
4. 可选：选择一个 **Refresh Interval**。

仪表板会出现在页面中，其中包含来自 Kata 客户机操作系统类别的以下指标：

正在运行的虚拟机数量
显示集群中运行的沙盒容器总数。

CPU 使用率（每个虚拟机）
显示每个沙盒容器的 CPU 使用量。

内存用量（每个虚拟机）
显示每个沙盒容器的内存用量。

将鼠标悬停在仪表板中的每个图形上，以显示具体项目的详细信息。

4.4. 其他资源

- 有关收集支持数据的更多信息，请参阅收集集群数据。

第 5 章 卸载 OPENSHIFT 沙盒容器

您可以使用 Red Hat OpenShift Web 控制台或 OpenShift CLI (**oc**)卸载 OpenShift 沙盒容器。下面解释这两个程序。

5.1. 使用 WEB 控制台卸载 OPENSHIFT 沙盒容器

使用 Red Hat OpenShift Web 控制台删除相关的 OpenShift 沙盒容器 pod、资源和命名空间。

5.1.1. 使用 Web 控制台删除 OpenShift 沙盒容器 pod

要卸载 OpenShift 沙盒容器，您必须首先删除所有使用 **kata** 作为 **runtimeClass** 的 pod。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您有一个使用 **kata** 作为 **runtimeClass** 的 pod 列表。

流程

1. 从 **Administrator** 视角中，进入到 **Workloads → Pods**。
2. 使用 **Search by name** 字段搜索您要删除的 pod。
3. 点 pod 名称打开它。
4. 在 **Details** 页面中，检查已针对 **Runtime 类** 显示 **kata**。
5. 点 **Actions** 菜单，再选择 **Delete Pod**。
6. 在确认窗口中点击 **Delete**。

其他资源

您可以从 OpenShift CLI 检索使用 **kata** 作为 **runtimeClass** 的运行 pod 的列表。详情请参阅 删除 OpenShift 沙盒容器 pod。

5.1.2. 使用 Web 控制台删除 KataConfig 自定义资源

删除 **KataConfig** 自定义资源 (CR) 会从集群中移除并卸载 **kata** 运行时及其相关资源。

重要

删除 **KataConfig** CR 会自动重启 worker 节点。重启可能需要 10 到 60 分钟。妨碍重启时间的因素如下：

- 带有更多 worker 节点的大型 Red Hat OpenShift 部署。
- 激活 BIOS 和 Diagnostics 实用程序。
- 在硬盘而不是 SSD 上部署。
- 在物理节点上部署，如裸机，而不是在虚拟节点上部署。
- CPU 和网络较慢。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您没有任何正在运行的 pod 使用 **kata** 作为 **runtimeClass**。

流程

1. 从 **Administrator** 视角中，进入到 **Operators → Installed Operators**。
2. 使用 **Search by name** 字段搜索 OpenShift 沙盒容器 Operator。
3. 点 Operator 打开它，然后选择 **KataConfig** 选项卡。
4. 点 **KataConfig** 资源的 **Options** 菜单 ，然后选择 **Delete KataConfig**。
5. 在确认窗口中点击 **Delete**。

等待 **kata** 运行时和资源卸载，并使 worker 节点重启，然后继续下一步。

5.1.3. 使用 Web 控制台删除 OpenShift 沙盒容器 Operator

删除 OpenShift 沙盒容器 Operator 会删除 Operator 的目录订阅、Operator 组和集群服务版本 (CSV)。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 从 **Administrator** 视角中，进入到 **Operators → Installed Operators**。
2. 使用 **Search by name** 字段搜索 OpenShift 沙盒容器 Operator。

3. 点击 Operator 的 **Options** 菜单  并选择 **Uninstall Operator**。
4. 在确认窗口中点 **Uninstall**。

5.1.4. 使用 Web 控制台删除 OpenShift 沙盒容器命名空间

运行上述命令后，集群将恢复到安装过程之前的状态。现在，您可以通过删除 **openshift-sandboxed-containers-operator** 命名空间来撤销对 Operator 的命名空间访问。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 从 **Administrator** 视角中，进入到 **Administration → Namespaces**。
2. 使用 **Search by name** 字段搜索 **openshift-sandboxed-containers-operator** 命名空间。
3. 点命名空间的 **Options** 菜单  并选择 **Delete Namespace**。

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。

4. 在 **Delete Namespace** 窗格中，输入 **openshift-sandboxed-containers-operator** 并点 **Delete**。
5. 点 **Delete**。

5.1.5. 使用 Web 控制台删除 KataConfig 自定义资源定义

KataConfig 自定义资源定义 (CRD) 可让您定义 **KataConfig** CR。要完成卸载过程，请从集群中删除 **KataConfig** CRD。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已从集群中删除 **KataConfig** CR。
- 您已从集群中移除了 OpenShift 沙盒容器 Operator。

流程

1. 从 **Administrator** 视角，进入到 **Administration → CustomResourceDefinitions**。

2. 使用 **Search by name** 字段搜索 **KataConfig**。
3. 点 **KataConfig** CRD  的 **Options** 菜单，然后选择 **Delete CustomResourceDefinition**。
4. 在确认窗口中点击 **Delete**。
5. 等待 **KataConfig** CRD 会从列表中消失。这可能需要几分钟。

5.2. 使用 CLI 卸载 OPENSHIFT 沙盒容器

您可以使用 Red Hat OpenShift 命令行界面(CLI) 卸载 OpenShift 沙盒容器。按照显示它们的顺序按照以下步骤操作。

5.2.1. 使用 CLI 删除 OpenShift 沙盒容器 pod

要卸载 OpenShift 沙盒容器，您必须首先删除所有使用 **kata** 作为 **runtimeClass** 的 pod。

先决条件

- 已安装 OpenShift CLI(**oc**)。
- 已安装命令行 JSON 处理器 (**jq**)。

流程

1. 运行以下命令，搜索使用 **kata** 作为 **runtimeClass** 的 pod：

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

2. 要删除每个 pod，请运行以下命令：

```
$ oc delete pod <pod-name>
```

5.2.2. 使用 CLI 删除 KataConfig 自定义资源

从集群中删除并卸载 **kata** 运行时及其所有相关资源，如 CRI-O 配置和 **RuntimeClass**。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

重要

删除 **KataConfig** CR 会自动重启 worker 节点。重启可能需要 10 到 60 分钟。妨碍重启时间的因素如下：

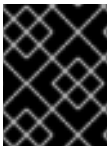
- 带有更多 worker 节点的大型 Red Hat OpenShift 部署。
- 激活 BIOS 和 Diagnostics 实用程序。
- 在硬盘而不是 SSD 上部署。
- 在物理节点上部署，如裸机，而不是在虚拟节点上部署。
- CPU 和网络较慢。

流程

1. 运行以下命令来删除 **KataConfig** 自定义资源：

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

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。



重要

在删除过程中，CLI 会停止响应，直到所有 worker 节点重启为止。等待进程完成，然后执行验证或继续进行后续步骤。

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
```

5.2.3. 使用 CLI 删除 OpenShift 沙盒容器 Operator

通过删除 Operator 订阅、Operator 组、集群服务版本(CSV)和命名空间从集群中删除 OpenShift 沙盒容器 Operator。

先决条件

- 已在集群中安装了 Red Hat OpenShift 4.10。
- 已安装 OpenShift CLI(**oc**)。
- 您已安装了 readand-line JSON 处理器(**jq**)。
- 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，从订阅中获取 OpenShift 沙盒容器的集群服务版本 (CSV) 名称：

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

2. 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 sandboxed-containers-operator -n openshift-sandboxed-containers-operator
```

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
```

4. 运行以下命令来获取 OpenShift 沙盒容器 Operator 组名称：

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

5. 运行以下命令来删除 OpenShift 沙盒容器 Operator 组名称：

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

6. 运行以下命令来删除 OpenShift 沙盒容器命名空间：

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

5.2.4. 使用 CLI 删除 KataConfig 自定义资源定义

KataConfig 自定义资源定义 (CRD) 可让您定义 **KataConfig** CR。从集群中删除 **KataConfig** CRD。

先决条件

- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已从集群中删除 **KataConfig** CR。
- 您已从集群中移除了 OpenShift 沙盒容器 Operator。

流程

1. 运行以下命令来删除 **KataConfig** CRD：

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

验证

- 要验证 **KataConfig** CRD 是否已删除，请运行以下命令：

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

输出示例

```
Unknown CR KataConfig
```

第 6 章 升级 OPENSHIFT 沙盒容器

OpenShift 沙盒容器组件的升级由以下三个步骤组成：

- 升级 Red Hat OpenShift 以更新 **Kata** 运行时及其依赖项。
- 升级 OpenShift 沙盒容器 Operator 以更新 Operator 订阅。
- 手动修补 **KataConfig** 自定义资源 (CR) 以更新监控 pod。

您可以在 OpenShift 沙盒容器 Operator 升级前或之后升级 Red Hat OpenShift，但有以下例外。在升级 OpenShift 沙盒容器 Operator 后，始终立即应用 **KataConfig** 补丁。

重要

如果您要使用 OpenShift 沙盒容器 1.3 升级到 Red Hat OpenShift 4.11，建议的顺序是从 OpenShift 沙盒容器从 1.2 升级到 1.3，然后将 Red Hat OpenShift 从 4.10 升级到 4.11。

6.1. 升级 OPENSHIFT 沙盒容器资源

OpenShift 沙盒容器资源使用 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。您可以通过将集群升级到 Red Hat OpenShift 的新版本来升级扩展。

有关升级 Red Hat OpenShift 的更多信息，请参阅更新集群。

6.2. 升级 OPENSHIFT 沙盒容器 OPERATOR

使用 Operator Lifecycle Manager (OLM) 手动或自动升级 OpenShift 沙盒容器 Operator。在初始部署期间，选择手动或自动升级可决定将来的升级模式。对于手动升级，Web 控制台会显示集群管理员可安装的可用更新。

有关在 Operator Lifecycle Manager (OLM) 中升级 OpenShift 沙盒容器 Operator 的更多信息，请参阅更新已安装的 Operator。

6.3. 升级 OPENSHIFT 沙盒容器监控 POD

升级 OpenShift 沙盒容器后，您需要更新 **KataConfig** CR 中的 monitor 镜像来升级监控 pod。否则，监控器 pod 将继续运行之前版本中的镜像。

您可以使用 Web 控制台或 CLI 执行更新。

6.3.1. 使用 Web 控制台升级 monitor pod

Red Hat OpenShift 中的 **KataConfig** YAML 文件包含监控镜像的版本号。使用正确的版本更新版本号。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 从 Red Hat OpenShift 的 **Administrator** 视角，进入到 **Operators → Installed Operators**。
2. 选择 **OpenShift 沙盒容器 Operator** 并进入 **KataConfig** 选项卡。
3. 使用 **Search by name** 字段搜索 **KataConfig** 资源。**KataConfig** 资源的默认名称为 **example-kataconfig**。
4. 选择 **KataConfig** 资源，再进入 **KataConfig** 选项卡。
5. 修改 **kataMonitorImage** 的版本号：

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

6. 点击 **Save**。

6.3.2. 使用 CLI 升级 monitor pod

您可以手动修补 **KataConfig** CR 中的 monitor 镜像，以更新 monitor pod。

先决条件

- 在集群中安装了 Red Hat OpenShift 4.13。
- 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

- 在 Red Hat OpenShift CLI 中运行以下命令：

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

其中： **<kataconfig_name>** :: 指定您的 Kata 配置文件的名称，如 **example-kataconfig**。

第 7 章 收集 OPENSHIFT 沙盒容器数据

当对 OpenShift 沙盒容器进行故障排除时，您可以创建一个支持问题单，并使用 **must-gather** 工具提供调试信息。

如果您是集群管理员，您还可以自行查看日志，启用更详细的日志级别。

7.1. 为红帽支持收集 OPENSHIFT 沙盒容器数据

在提交问题单时同时提供您的集群信息，可以帮助红帽支持为您进行排除故障。

您可使用 **must-gather** 工具来收集有关 Red Hat OpenShift 集群的诊断信息，包括虚拟机和与 OpenShift 沙盒容器相关的其他数据。

为了获得快速支持，请提供 Red Hat OpenShift 和 OpenShift 沙盒容器的诊断信息。

7.1.1. 关于 must-gather 工具

oc adm must-gather CLI 命令可收集最有助于解决问题的集群信息，包括：

- 资源定义
- 服务日志

默认情况下，**oc adm must-gather** 命令使用默认的插件镜像，并写入 **./must-gather.local**。

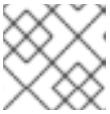
另外，您可以使用适当的参数运行命令来收集具体信息，如以下部分所述：

- 要收集与一个或多个特定功能相关的数据，请使用 **--image** 参数和镜像，如以下部分所述。
  例如：

  ```
  $ oc adm must-gather  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.13.0
  ```

- 要收集审计日志，请使用 **-- /usr/bin/gather_audit_logs** 参数，如以下部分所述。
  例如：

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

  注意

  作为默认信息集合的一部分，不会收集审计日志来减小文件的大小。

当您运行 **oc adm must-gather** 时，集群的新项目中会创建一个带有随机名称的新 pod。在该 pod 上收集数据，并保存至以 **must-gather.local** 开头的一个新目录中。此目录在当前工作目录中创建。

例如：

```
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
...
```

另外，您可以使用 **--run-namespace** 选项在特定命名空间中运行 **oc adm must-gather** 命令。

例如：

```
$ oc adm must-gather --run-namespace <namespace> --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.13.0
```

要使用 **must-gather** 来收集 OpenShift 沙盒容器数据，您必须指定 OpenShift 沙盒容器镜像：

```
--image=registry.redhat.io/openshift-sandboxed-containers/osc-must-gather-rhel8:1.4.0
```

7.2. 关于 OPENSHIFT 沙盒容器日志数据

当您收集集群的日志数据时，以下功能和对象与 OpenShift 沙盒容器相关联：

- 所有属于任何 OpenShift 沙盒容器资源的命名空间及其子对象
- 所有 OpenShift 沙盒容器自定义资源定义 (CRD)

以下 OpenShift 沙盒容器组件日志会针对使用 **kata** 运行时运行的每个 pod 收集：

- Kata 代理日志
- Kata 运行时日志
- QEMU 日志
- 审计日志
- CRI-O 日志

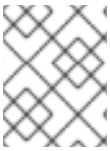
7.3. 为 OPENSHIFT 沙盒容器启用调试日志

作为集群管理员，您可以为 OpenShift 沙盒容器收集更详细的日志级别。您还可以通过更改 **KataConfig** CR 中的 **logLevel** 字段来增强日志记录。这会更改运行 OpenShift 沙盒容器的 worker 节点的 CRI-O 运行时中的 **log_level**。

流程

1. 将现有 **KataConfig** CR 中的 **logLevel** 字段更改为 **debug** ：

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



注意

运行此命令时，引用 **KataConfig** CR 的名称。这是设置 OpenShift 沙盒容器时用于创建 CR 的名称。

验证

1. 监控 **kata-oc** 机器配置池，直到 **UPDATED** 字段显示为 **True**，这意味着所有 worker 节点都已更新：

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

输出示例

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

2. 验证 **log_level** 是否在 CRI-O 中更新：

   a. 打开到机器配置池中节点的 **oc debug** 会话，并运行 **chroot /host**。

   ```
   $ oc debug node/<node_name>
   ```

   ```
   sh-4.4# chroot /host
   ```

   b. 验证 **crio.conf** 文件中的更改：

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

   输出示例

   ```
   log_level = "debug"
   ```

7.3.1. 查看 OpenShift 沙盒容器的调试日志

集群管理员可以使用 OpenShift 沙盒容器增强的调试日志来排除问题。每个节点的日志会输出到节点日志中。

您可以查看以下 OpenShift 沙盒容器组件的日志：

- Kata 代理
- Kata runtime (**containerd-shim-kata-v2**)
- virtiofsd

QEMU 仅生成警告和错误日志。这些警告和错误会在 Kata 运行时日志和带有额外的 **qemuPid** 字段的 CRI-O 日志中打印到节点日志。

QEMU 日志示例

```
Mar 11 11:57:28 openshift-worker-0 kata[2241647]: time="2023-03-11T11:57:28.587116986Z"
level=info msg="Start logging QEMU (qemuPid=2241693)" name=containerd-shim-v2 pid=2241647
sandbox=d1d4d68efc35e5ccb4331af73da459c13f46269b512774aa6bde7da34db48987
source=virtcontainers/hypervisor subsystem=qemu

Mar 11 11:57:28 openshift-worker-0 kata[2241647]: time="2023-03-11T11:57:28.607339014Z"
level=error msg="qemu-kvm: -machine q35,accel=kvm,kernel_irqchip=split,foo: Expected '=' after
parameter 'foo'" name=containerd-shim-v2 pid=2241647 qemuPid=2241693
sandbox=d1d4d68efc35e5ccb4331af73da459c13f46269b512774aa6bde7da34db48987
```

```
source=virtcontainers/hypervisor subsystem=qemu

Mar 11 11:57:28 openshift-worker-0 kata[2241647]: time="2023-03-11T11:57:28.60890737Z"
level=info msg="Stop logging QEMU (qemuPid=2241693)" name=containerd-shim-v2 pid=2241647
sandbox=d1d4d68efc35e5ccb4331af73da459c13f46269b512774aa6bde7da34db48987
source=virtcontainers/hypervisor subsystem=qemu
```

当 QEMU 启动时，Kata 运行时会在 QEMU 启动时打印 **Start logging QEMU**，并在 QEMU 停止时**停止日志记录 QEMU**。使用 **qemuPid** 字段的两个日志消息之间会出现这个错误。QEMU 的实际错误消息以红色显示。

QEMU 客户机的控制台也会输出到节点日志中。您可以查看客户机控制台日志以及 Kata 代理日志。

先决条件

- 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

- 要查看 Kata 代理日志和客户机控制台日志，请运行：

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

- 要查看 kata 运行时日志，请运行：

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

- 要查看 virtiofsd 日志，请运行：

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

- 要查看 QEMU 日志，请运行：

```
$ oc debug node/<nodename> -- journalctl -D /host/var/log/journal -t kata -g "qemuPid=\d+"
```

7.4. 其他资源

- 有关收集支持数据的更多信息，请参阅收集集群数据。