OpenShift Sandboxed Containers 1.4

OpenShift 샌드박스 컨테이너 사용자 가이드

For Red Hat OpenShift

Last Updated: 2024-01-31

OpenShift Sandboxed Containers 1.4 OpenShift 샌드박스 컨테이너 사용자 가이드

For Red Hat OpenShift

법적 공지

초록

OpenShift 샌드박스 컨테이너 Operator 및 사용법에 대한 정보

차례

머리말

1장. OPENSHIFT 샌드박스 컨테이너 이해

Red Hat OpenShift에 대한 OpenShift 샌드박스 컨테이너 지원은 Kata Containers를 추가 선택적 런타임으로 실행하는 데 대한 기본 지원을 제공합니다. 새로운 런타임은 전용 VM(가상 머신)의 컨테이너를 지원하므로 워크로드 격리가 개선되었습니다. 이 기능은 다음 작업을 수행하는 데 특히 유용합니다.

권한 있거나 신뢰할 수 없는 워크로드 실행

OpenShift 샌드박스 컨테이너(OSC)를 사용하면 권한 있는 컨테이너를 실행하여 클러스터 노드를 손상시킬 위험 없이 특정 권한이 필요한 워크로드를 안전하게 실행할 수 있습니다. 특수 권한이 필요한 워크로드에는 다음이 포함됩니다.

- CRI-O와 같은 표준 컨테이너 런타임에서 부여하는 기본 컨테이너 런타임 이외의 커널의 특수 기능이 필요한 워크로드(예: 낮은 수준의 네트워킹 기능에 액세스하는 경우)는 다음과 같습니다.
- 예를 들어 특정 물리적 장치에 액세스하기 위해 승격된 루트 권한이 필요한 워크로드입니다. OpenShift 샌드박스 컨테이너를 사용하면 특정 장치만 VM에 전달하여 워크로드가 나머지 시스템에 액세스하거나 잘못 구성할 수 없습니다.
- **set-uid** 루트 바이너리를 설치하거나 사용하는 워크로드입니다. 이러한 바이너리는 특수 권한을 부여하므로 보안 위험이 발생할 수 있습니다. OpenShift 샌드박스 컨테이너에서는 추가 권한이 가상 머신으로 제한되고 클러스터 노드에 대한 특별한 액세스 권한이 부여되지 않습니다.

일부 워크로드에는 특히 클러스터 노드를 구성하기 위한 권한이 필요할 수 있습니다. 가상 머신에서 실행하면 작동하지 않기 때문에 이러한 워크로드에서는 권한 있는 컨테이너를 계속 사용해야 합니다.

각 워크로드에 대한 커널 격리 보장

OpenShift 샌드박스 컨테이너는 사용자 정의 커널 튜닝(예: **sysctl**, 스케줄러 변경 또는 캐시 튜닝)과 사용자 정의 커널 모듈 생성(예: **out of tree** 또는 special arguments)이 필요한 워크로드를 지원합니다.

테넌트 간에 동일한 워크로드 공유

OpenShift 샌드박스 컨테이너를 사용하면 동일한 OpenShift 클러스터를 공유하는 다른 조직의 여러 사용자(테넌트)를 지원할 수 있습니다. 또한 이 시스템을 사용하면 컨테이너 네트워크 기능(CNF) 및 엔터프라이즈 애플리케이션과 같은 여러 공급업체의 타사 워크로드를 실행할 수 있습니다. 예를 들어 타사 CNF는 패킷 튜닝 또는 다른 애플리케이션에서 설정한 **sysctl** 변수를 사용하는 사용자 지정 설정을 원하지 않을 수 있습니다. 완전히 분리된 커널 내부에서 실행하는 것은 "아마운" 구성 문제를 방지하는 데 도움이 됩니다.

소프트웨어 테스트를 위한 적절한 격리 및 샌드박스 보장

OpenShift 샌드박스 컨테이너를 사용하여 알려진 취약점이 있는 컨테이너화된 워크로드를 실행하거나 레거시 애플리케이션의 문제를 처리할 수 있습니다. 또한 이러한 격리를 통해 관리자는 개발자에게 포드에 대한 제어 권한을 부여할 수 있습니다. 이는 개발자가 해당 관리자 이외의 구성을 테스트하거나 검증하려는 경우 일반적으로 관리자에게 부여되는 경우에 유용합니다. 예를 들어 관리자는 eBPF(커널 패킷 필터링)를 안전하게 개발자에게 위임할 수 있습니다. 커널 패킷 필터링에는 **CAP_ADMIN** 또는 **CAP_BPF** 권한이 필요하므로 표준 CRI-O 구성에서는 컨테이너 호스트 작업자 노드의 모든 프로세스에 대한 액세스 권한을 부여할 수 없습니다. 마찬가지로 관리자는 SystemTap과 같은 개입 도구에 대한 액세스 권한을 부여하거나 개발 중에 사용자 지정 커널 모듈 로드를 지원할 수 있습니다.

VM 경계를 통한 기본 리소스 격리 확인

기본적으로 CPU, 메모리, 스토리지 또는 네트워킹과 같은 리소스는 OpenShift 샌드박스 컨테이너에서 보다 강력하고 안전한 방식으로 관리됩니다. OpenShift 샌드박스 컨테이너는 VM에 배포되므로 추가 격리 및 보안 계층을 통해 리소스에 대한 액세스를 세밀하게 제어할 수 있습니다. 예를 들어 잘못된 컨테이너는 VM에서 사용할 수 있는 메모리보다 더 많은 메모리를 할당할 수 없습니다. 반대로 네트워크 카드 또는 디스크에 대한 전용 액세스 권한이 필요한 컨테이너는 다른 장치에 액세스하지 않고도 해당 장치를 완전히 제어할 수 있습니다.

1.1. OPENSHIFT 샌드박스 컨테이너 지원 플랫폼

베어 메탈 서버 또는 AWS(Amazon Web Services) 베어 메탈 인스턴스에 OpenShift 샌드박스 컨테이너를 설치할 수 있습니다. 다른 클라우드 공급자가 제공하는 베어 메탈 인스턴스는 지원되지 않습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)는 OpenShift 샌드박스 컨테이너에서 지원되는 유일한 운영 체제입니다. OpenShift 샌드박스 컨테이너 1.4는 RHCOS(Red Hat Enterprise Linux CoreOS) 8.6에서 실행됩니다.

OpenShift 샌드박스 컨테이너 1.4는 Red Hat OpenShift 4.11과 호환됩니다.

1.2. OPENSHIFT 샌드박스 컨테이너 일반 용어

설명서 전반에 걸쳐 다음 용어가 사용됩니다.

샌드 박스

샌드박스는 프로그램을 실행할 수 있는 격리된 환경입니다. 샌드박스에서는 호스트 시스템이나 운영 체제에 손상을 주지 않고 테스트되지 않았거나 신뢰할 수 없는 프로그램을 실행할 수 있습니다. OpenShift 샌드박스 컨테이너의 경우 가상화를 사용하여 다른 커널에서 워크로드를 실행하여 동일한 호스트에서 실행되는 여러 워크로드 간의 상호 작용을 보다 효과적으로 제어할 수 있습니다.

Pod

Pod는 Kubernetes 및 Red Hat OpenShift에서 상속된 구성 요소입니다. 컨테이너를 배포할 수 있는 리소스를 나타냅니다. 컨테이너는 Pod 내부에서 실행되며 Pod는 여러 컨테이너 간에 공유할 수 있는 리소스를 지정하는 데 사용됩니다.
OpenShift 샌드박스 컨테이너의 컨텍스트에서 Pod가 가상 시스템으로 구현됩니다. 동일한 가상 시스템의 동일한 Pod에서 여러 컨테이너를 실행할 수 있습니다.

OpenShift 샌드박스 컨테이너 Operator

Operator는 시스템에서 수동으로 수행할 수 있는 작업을 자동화하는 소프트웨어 구성 요소입니다. OpenShift 샌드박스 컨테이너 Operator는 클러스터에서 샌드박스 컨테이너의 라이프사이클을 관리하는 작업을 수행합니다. OpenShift 샌드박스 컨테이너 Operator를 사용하여 샌드박스 컨테이너의 설치 및 제거, 소프트웨어 업데이트, 상태 모니터링과 같은 작업을 수행할 수 있습니다.

Kata 컨테이너

Kata 컨테이너는 OpenShift 샌드박스 컨테이너를 빌드하는 데 사용되는 핵심 업스트림 프로젝트입니다. OpenShift 샌드박스 컨테이너는 Kata Containers와 Red Hat OpenShift를 통합합니다.

KataConfig

KataConfig 오브젝트는 샌드박스 컨테이너의 구성을 나타냅니다. 소프트웨어가 배포된 노드와 같이 클러스터 상태에 대한 정보를 저장합니다.

런타임 클래스

RuntimeClass 오브젝트는 지정된 워크로드를 실행하는 데 사용할 수 있는 런타임에 대해 설명합니다. **kata**라는 런타임 클래스가 OpenShift 샌드박스 컨테이너 Operator에 의해 설치 및 배포됩니다. 런타임 클래스에는 Pod 오버헤드와 같이 런타임에서 작동해야 하는 리소스를 설명하는 런타임에 대한 정보가 포함되어 있습니다.

피어 Pod

OpenShift 샌드박스 컨테이너의 피어 Pod는 표준 Pod의 개념을 확장합니다. 가상 머신이 작업자 노드 자체에서 생성되는 표준 샌드박스 컨테이너와 달리 피어 Pod에서 지원되는 하이퍼바이저 또는 클라우드 공급자 API를 사용하여 원격 하이퍼바이저를 통해 가상 머신이 생성됩니다. 피어 포드는 작업자 노

드에서 일반 pod 역할을 하며 해당 VM이 다른 위치에서 실행됩니다. VM의 원격 위치는 사용자에게 투명하며 Pod 사양의 런타임 클래스에 의해 지정됩니다. 피어 Pod 설계는 중첩된 가상화의 필요성을 우회합니다.

1.3. OPENSHIFT 샌드박스 컨테이너 워크로드 관리

OpenShift 샌드박스 컨테이너는 워크로드 관리 및 할당을 개선하기 위해 다음과 같은 기능을 제공합니다.

1.3.1. OpenShift 샌드박스 컨테이너 빌딩 블록

OpenShift 샌드박스 컨테이너 Operator는 Kata 컨테이너의 모든 구성 요소를 캡슐화합니다. 설치, 라이프사이클 및 구성 작업을 관리합니다.

OpenShift 샌드박스 컨테이너 Operator는 Operator 번들 형식으로 두 개의 컨테이너 이미지로 패키지됩니다. 번들 이미지에는 Operator OLM을 준비하는데 필요한 메타데이터가 포함되어 있습니다. 두 번째 컨테이너 이미지에는 **KataConfig** 리소스를 모니터링하고 관리하는 실제 컨트롤러가 포함되어 있습니다.

1.3.2. RHCOS 확장

OpenShift 샌드박스된 컨테이너 Operator는 RHCOS(Red Hat Enterprise Linux CoreOS) 확장 개념을 기반으로 합니다. RHCOS(Red Hat Enterprise Linux CoreOS) 확장 기능은 선택적 Red Hat OpenShift 소프트웨어를 설치하는 메커니즘입니다. OpenShift 샌드박스된 컨테이너 Operator는 이 메커니즘을 사용하여 클러스터에 샌드박스 컨테이너를 배포합니다.

샌드박스 컨테이너 RHCOS 확장에는 Kata, QEMU 및 종속 항목에 대한 RPM이 포함되어 있습니다. Machine Config Operator에서 제공하는 **MachineConfig** 리소스를 사용하여 활성화할 수 있습니다.

추가 리소스

- RHCOS에 확장 기능 추가

1.3.3. 가상화 및 OpenShift 샌드박스 컨테이너

OpenShift Virtualization과 함께 클러스터에서 OpenShift 샌드박스 컨테이너를 사용할 수 있습니다.

OpenShift Virtualization 및 OpenShift 샌드박스 컨테이너를 동시에 실행하려면 VM이 노드 재부팅을 차단하지 않도록 VM을 활성화해야 합니다. VM에서 다음 매개변수를 구성합니다.

- **ocs-storagecluster-ceph-rbd** 를 스토리지 클래스로 사용합니다.
- VM에서 **evictionStrategy** 매개변수를 **LiveMigrate** 로 설정합니다.

추가 리소스

- 가상 머신 로컬 스토리지 구성
- 가상 머신 제거 전략 구성

1.4. 컴플라이언스 및 위험 관리 이해

Red Hat OpenShift는 FIPS용으로 설계되었습니다. FIPS 모드에서 부팅된 RHEL(Red Hat Enterprise Linux CoreOS) 또는 RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 경우 Red Hat OpenShift 핵심 구성 요소는 **x86_64**,**ppc64le**, **s390x** 아키텍처에서만 FIPS 140-2/140-3 Validation에 대해 NIST에 제출된 RHEL 암호화 라이브러리를 사용합니다.

NIST 검증 프로그램에 대한 자세한 내용은 암호화 모듈 유효성 검사 프로그램을 참조하십시오. 검증을 위해 제출된 개별 RHEL 암호화 라이브러리의 최신 NIST 상태는 규정 준수 활동 및 정부 표준을 참조하십시오.

OpenShift 샌드박스 컨테이너는 FIPS가 활성화된 클러스터에서 사용할 수 있습니다.

FIPS 모드에서 실행되는 경우 OpenShift 샌드박스 컨테이너 구성 요소, VM 및 VM 이미지는 FIPS를 준수하도록 조정됩니다.

참고

OpenShift 샌드박스 컨테이너에 대한 FIPS 컴플라이언스는 **kata** 런타임 클래스에만 적용됩니다. 새로운 피어 Pod 런타임 클래스 **kata-remote-cc** 는 아직 완전히 지원되지 않으며 FIPS 컴플라이언스를 위해 테스트되지 않았습니다.

FIPS 컴플라이언스는 보안 수준이 높은 환경에서 요구되는 가장 중요한 구성요소 중 하나로, 지원되는 암호화 기술만 노드에서 허용합니다.

중요

진행 중인 FIPS 검증 / 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 Red Hat OpenShift 배포에서만 지원됩니다.

Red Hat OpenShift 컴플라이언스 프레임워크에 대한 Red Hat의 의견을 이해하려면 OpenShift 보안 가이드 북 의 위험 관리 및 규제 준비 장을 참조하십시오.

2장. OPENSHIFT 샌드박스 컨테이너 워크로드 배포

웹 콘솔 또는 OpenShift CLI(**oc**)를 사용하여 OpenShift 샌드박스 컨테이너 Operator를 설치할 수 있습니다. OpenShift 샌드박스 컨테이너 Operator를 설치하기 전에 Red Hat OpenShift 클러스터를 준비해야 합니다.

2.1. 사전 요구 사항

OpenShift 샌드박스 컨테이너를 설치하기 전에 Red Hat OpenShift 클러스터가 다음 요구 사항을 충족하는지 확인하십시오.

- RHCOS(Red Hat Enterprise Linux CoreOS) 작업자를 사용하여 온프레미스 베어메탈 인프라에 클러스터가 설치되어 있어야 합니다. 사용자 프로비저닝, 설치 관리자 프로비저닝 또는 지원 설치 프로그램을 포함하여 설치 방법을 사용하여 클러스터를 배포할 수 있습니다.

참고

  - OpenShift 샌드박스 컨테이너는 RHCOS 작업자 노드만 지원합니다. RHEL 노드는 지원되지 않습니다.
  - 중첩된 가상화는 지원되지 않습니다.

- AWS(Amazon Web Services) 베어메탈 인스턴스에 OpenShift 샌드박스 컨테이너를 설치할 수 있습니다. 다른 클라우드 공급자가 제공하는 베어 메탈 인스턴스는 지원되지 않습니다.

2.1.1. OpenShift 샌드박스 컨테이너의 리소스 요구 사항

OpenShift 샌드박스 컨테이너를 사용하면 사용자가 샌드박스 런타임(Kata) 내에서 Red Hat OpenShift 클러스터에서 워크로드를 실행할 수 있습니다. 각 Pod는 VM(가상 머신)으로 표시됩니다. 각 VM은 QEMU 프로세스에서 실행되며 컨테이너 워크로드를 관리하기 위한 감독자 역할을 하는 **kata-agent** 프로세스 및 해당 컨테이너에서 실행되는 프로세스를 호스팅합니다. 두 개의 추가 프로세스는 오버헤드를 더 추가합니다.

- **containerd-shim-kata-v2**는 pod와 통신하는 데 사용됩니다.
- **virtiofsd**는 게스트 대신 호스트 파일 시스템 액세스를 처리합니다.

각 VM은 기본 메모리 양으로 구성됩니다. 메모리를 명시적으로 요청하는 컨테이너의 경우 VM에 추가 메모리가 핫플러그됩니다.

메모리 리소스 없이 실행되는 컨테이너는 VM에서 사용하는 총 메모리가 기본 할당에 도달할 때까지 사용 가능한 메모리를 사용합니다. 게스트와 I/O 버퍼도 메모리를 소비합니다.

컨테이너에 특정 양의 메모리가 제공되면 컨테이너를 시작하기 전에 해당 메모리는 VM에 핫플러그됩니다.

메모리 제한을 지정하면 제한보다 많은 메모리를 사용하는 경우 워크로드가 종료됩니다. 메모리 제한을 지정하지 않으면 VM에서 실행되는 커널이 메모리가 부족해질 수 있습니다. 커널이 메모리가 부족하면 VM의 다른 프로세스를 종료할 수 있습니다.

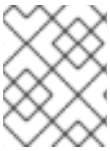
기본 메모리 크기

다음 표에는 리소스 할당에 대한 기본값이 나열되어 있습니다.

리소스	현재의
기본적으로 가상 머신에 할당된 메모리	2Gi
부팅 시 게스트 Linux 커널 메모리 사용량	~110Mi
QEMU 프로세스에서 사용하는 메모리 (VM 메모리 제외)	~30Mi
virtiofsd 프로세스에서 사용하는 메모리(VM I/O 버퍼 제외)	~10Mi
containerd-shim-kata-v2 프로세스에서 사용하는 메모리	~20Mi
Fedora에서 **dnf install**을 실행한 후 파일 버퍼 캐시 데이터	~300Mi* [1]

파일 버퍼가 나타나고 다음과 같은 여러 위치에서 고려됩니다.

- 게스트에서 파일 버퍼 캐시로 표시
- 허용된 사용자 공간 파일 I/O 작업을 매핑된 **virtiofsd** 데몬
- 게스트 메모리로 사용되는 QEMU 프로세스



참고

총 메모리 사용량은 메모리 사용량 통계에 따라 적절히 계산되며, 이 메트릭은 해당 메모리를 한 번만 계산합니다.

Pod 오버헤드는 노드의 Pod에서 사용하는 시스템 리소스의 양을 설명합니다. 아래와 같이 **oc describe runtimeclass kata** 를 사용하여 Kata 런타임에 대한 현재 Pod 오버헤드를 가져올 수 있습니다.

예제

```
$ oc describe runtimeclass kata
```

출력 예

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

RuntimeClass의 **spec.overhead** 필드를 변경하여 Pod 오버헤드를 변경할 수 있습니다. 예를 들어 컨테이너에 대한 구성이 QEMU 프로세스 및 게스트 커널 데이터에 대해 350Mi 이상의 메모리를 사용하는 경우 필요에 맞게 **RuntimeClass** 오버헤드를 변경할 수 있습니다.

참고

Red Hat은 지정된 기본 오버헤드 값을 지원합니다. 기본 오버헤드 값 변경은 지원되지 않으며 값을 변경하면 기술적인 문제가 발생할 수 있습니다.

게스트에서 모든 유형의 파일 시스템 I/O를 수행할 때 게스트 커널에 파일 버퍼가 할당됩니다. 파일 버퍼는 호스트의 QEMU 프로세스 및 **virtiofsd** 프로세스에도 매핑됩니다.

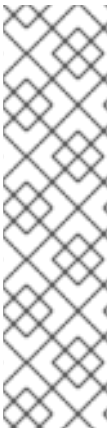
예를 들어 게스트에서 300Mi 파일 버퍼 캐시를 사용하는 경우 QEMU와 **virtiofsd**는 모두 300Mi 추가 메모리를 사용하는 것처럼 나타납니다. 그러나 세 가지 경우 모두 동일한 메모리가 사용됩니다. 즉, 총 메모리 사용량은 300Mi에 불과하며 이 값은 서로 다른 세 위치에 매핑됩니다. 이는 메모리 사용량 메트릭을 보고할 때 올바르게 계산됩니다.

추가 리소스

- 베어 메탈에 사용자 프로비저닝 클러스터 설치

2.1.2. 클러스터 노드가 OpenShift 샌드박스 컨테이너를 실행할 수 있는지 확인

OpenShift 샌드박스 컨테이너를 실행하기 전에 클러스터의 노드가 Kata 컨테이너를 실행할 수 있는지 확인할 수 있습니다. 일부 클러스터 노드는 샌드박스 컨테이너의 최소 요구사항을 준수하지 않을 수 있습니다. 노드 불신성에 대한 가장 일반적인 이유는 노드에서 가상화 지원이 부족하기 때문입니다. 적합하지 않은 노드에서 샌드박스 워크로드를 실행하려고 하면 오류가 발생합니다. NFD(Node Feature Discovery) Operator 및 **NodeFeatureDiscovery** 리소스를 사용하여 노드 자격을 자동으로 확인할 수 있습니다.



참고

적합한 선택한 작업자 노드에 Kata 런타임을 설치하려면 **feature.node.kubernetes.io/runtime.kata=true** 레이블을 선택한 노드에 적용하고 **KataConfig** 리소스에 **checkNodeEligibility: true** 를 설정합니다.

또는 모든 작업자 노드에 Kata 런타임을 설치하려면 **KataConfig** 리소스에서 **checkNodeEligibility: false** 를 설정합니다.

이러한 두 가지 시나리오에서는 **NodeEnatureDiscovery** 리소스를 만들 필요가 없습니다. 노드가 Kata 컨테이너를 실행할 자격이 있는지 확인하는 경우 **feature.node.kubernetes.io/runtime.kata=true** 라벨만 수동으로 적용해야 합니다.

다음 절차에서는 **feature.node.kubernetes.io/runtime.kata=true** 레이블을 모든 적격 노드에 적용하고 **KataConfig** 리소스를 구성하여 노드 자격을 확인합니다.

사전 요구 사항

- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 로그인합니다.
- NFD(Node Feature Discovery) Operator를 설치합니다.

절차

1. **NodeEnatureDiscovery** 리소스를 생성하여 Kata 컨테이너 실행에 적합한 노드 기능을 감지합니다.

   a. 다음 YAML을 **nfd.yaml** 파일에 저장합니다.

```
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. **NodeEnatureDiscovery 사용자** 정의 리소스(CR)를 만듭니다.

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

   출력 예

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

   feature.node.kubernetes.io/runtime.kata=true 레이블이 모든 적격 작업자 노드에 적용됩니다.

2. **KataConfig** 리소스에서 **checkNodeEligibility** 필드를 **true** 로 설정하여 기능을 활성화합니다. 예를 들면 다음과 같습니다.

   a. 다음 YAML을 **kata-config.yaml** 파일에 저장합니다.

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

   b. **KataConfig** CR을 생성합니다.

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

   출력 예

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

검증

- 클러스터의 적격 노드에 올바른 레이블이 적용되었는지 확인합니다.

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

출력 예

```
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
```

추가 리소스

- NFD(Node Feature Discovery) Operator 설치에 대한 자세한 내용은 NFD 설치를 참조하십시오.

2.2. 웹 콘솔을 사용하여 OPENSHIFT 샌드박스 컨테이너 워크로드 배포

웹 콘솔에서 OpenShift 샌드박스 컨테이너 워크로드를 배포할 수 있습니다. 먼저 OpenShift 샌드박스 컨테이너 Operator를 설치한 다음 **KataConfig** CR(사용자 정의 리소스)을 생성해야 합니다. 샌드박스 컨테이너에 워크로드를 배포할 준비가 되면 워크로드 YAML 파일에 **kata** 를 **runtimeClassName** 으로 수동으로 추가해야 합니다.

2.2.1. 웹 콘솔을 사용하여 OpenShift 샌드박스 컨테이너 Operator 설치

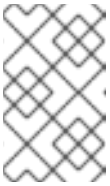
Red Hat OpenShift 웹 콘솔에서 OpenShift 샌드박스 컨테이너 Operator를 설치할 수 있습니다.

사전 요구 사항

- Red Hat OpenShift 4.13이 설치되어 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

절차

1. 웹 콘솔의 **관리자** 화면에서 **Operator → OperatorHub** 로 이동합니다.
2. **키워드로 필터링** 필드에 **OpenShift sandboxed containers**를 입력합니다.
3. **OpenShift 샌드박스 컨테이너** 타일을 선택합니다.
4. Operator에 대한 정보를 확인하고 **Install**을 클릭합니다.
5. **Operator 설치** 페이지에서 다음을 수행합니다.
   a. 사용 가능한 **업데이트 채널** 옵션 목록에서 **stable**을 선택합니다.
   b. 설치된 네임스페이스에 대해 **Operator 권장 네임스페이스** 가 선택되어 **있는지 확인합니다**. 그러면 필수 **openshift-sandboxed-containers-operator** 네임스페이스에 Operator가 설치됩니다. 이 네임스페이스가 아직 존재하지 않으면 자동으로 생성됩니다.



참고

openshift-sandboxed-containers-operator 이외의 네임스페이스에 OpenShift 샌드박스 컨테이너 Operator를 설치하려고 하면 설치가 실패합니다.

c. **승인 전략에 대해 자동 이 선택되어 있는지 확인합니다. 자동** 은 기본값이며 새로운 z-stream 릴리스가 제공되면 OpenShift 샌드박스 컨테이너에 대한 자동 업데이트를 활성화합니다.

6. **설치**를 클릭합니다.

OpenShift 샌드박스 컨테이너 Operator가 클러스터에 설치되었습니다.

검증

1. 웹 콘솔의 **관리자** 화면에서 **Operator → 설치된 Operator 로 이동합니다**.
2. OpenShift 샌드박스 컨테이너 Operator가 Operator 목록에 나열되어 있는지 확인합니다.

2.2.2. 웹 콘솔에서 KataConfig 사용자 지정 리소스 생성

클러스터 노드에서 **RuntimeClass** 로 **kata** 를 설치하려면 하나의 **KataConfig** CR(사용자 정의 리소스)을 생성해야 합니다.

중요

KataConfig CR을 생성하면 작업자 노드가 자동으로 재부팅됩니다. 재부팅하는 데 10분에서 60분 이상 걸릴 수 있습니다. 재부팅 시간을 방해하는 요소는 다음과 같습니다.

- 더 많은 수의 작업자 노드가 있는 대규모 Red Hat OpenShift 배포.
- BIOS 및 iPXE 유틸리티 활성화
- SSD가 아닌 하드 디스크 드라이브에 배포합니다.
- 가상 노드가 아닌 베어 메탈과 같은 물리적 노드에 배포.
- 느린 CPU 및 네트워크

사전 요구 사항

- 클러스터에 Red Hat OpenShift 4.13을 설치했습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 Operator가 설치되어 있습니다.

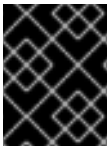
참고

Kata는 기본적으로 모든 작업자 노드에 설치됩니다. **kata** 를 특정 노드에만 **RuntimeClass** 로 설치하려면 해당 노드에 레이블을 추가한 다음 생성할 때 **KataConfig** CR에 레이블을 정의할 수 있습니다.

절차

1. 웹 콘솔의 **관리자** 화면에서 **Operator → 설치된 Operator 로 이동합니다**.
2. Operator 목록에서 OpenShift 샌드박스 컨테이너 Operator를 선택합니다.
3. **KataConfig** 탭에서 **KataConfig 만들기** 를 클릭합니다.
4. **KataConfig 생성** 페이지에서 다음 세부 정보를 입력합니다.
   - name: **KataConfig** 리소스의 이름을 입력합니다. 기본적으로 이름은 **example-kataconfig** 로 정의됩니다.
   - **레이블** (선택 사항): **KataConfig** 리소스에 관련된 식별 특성을 입력합니다. 각 레이블은 키-값 쌍을 나타냅니다.
   - **CheckNodeEligibility** (선택 사항, 피어 Pod와 관련이 없음): NFD(Node Feature Discovery Operator)를 사용하여 **kata** 를 **RuntimeClass** 로 실행할 자격을 감지하려면 이 확인란을 선택합니다. 자세한 내용은 "클러스터 노드가 OpenShift 샌드박스 컨테이너를 실행할 수 있는지 여부"를 참조하십시오.
   - **EnablePeerPods** (피어 Pod의 경우): 이 확인란을 선택하여 피어 Pod를 활성화하고 퍼블릭 클라우드 환경에서 OpenShift 샌드박스 컨테이너를 사용합니다.
   - **kataConfigPoolSelector**: 기본적으로 **kata** 는 모든 노드에 **RuntimeClass** 로 설치됩니다. **kata** 를 선택한 노드에만 **RuntimeClass** 로 설치하려면 **matchExpression** 을 추가해야 합니다.
     a. **kataConfigPoolSelector** 영역을 확장합니다.
     b. **kataConfigPoolSelector** 에서 **matchExpressions** 를 확장합니다. 라벨 선택기 요구 사항 목록입니다.
     c. **Add matchExpressions** 를 클릭합니다.
     d. **키** 필드에서 선택기가 적용되는 라벨 키를 추가합니다.
     e. **operator** 필드에서 레이블 값에 키의 관계를 추가합니다. 유효한 연산자는 **In**,**NotIn**,**Exists** 및 **DoesNotExist** 입니다.
     f. **값** 영역을 확장한 다음 **값 추가** 를 클릭합니다.
     g. **값** 필드에 **키** 레이블 값에 **true** 또는 **false** 를 입력합니다.
   - **loglevel**: **kata** 를 **RuntimeClass** 로 실행하는 노드에 대해 검색된 로그 데이터의 수준을 정의합니다. 자세한 내용은 "OpenShift 샌드박스 컨테이너 데이터 조정"을 참조하십시오.
5. **생성**을 클릭합니다.

새로운 **KataConfig** CR이 생성되고 작업자 노드에 **RuntimeClass** 로 **kata** 를 설치하기 시작합니다. **kata** 설치가 완료되고 작업자 노드가 재부팅될 때까지 기다린 후 다음 단계를 계속합니다.



중요

OpenShift 샌드박스 컨테이너는 **kata** 를 기본 런타임이 아닌 클러스터의 선택적 런타임으로만 설치합니다.

검증

1. **KataConfig** 탭에서 새 **KataConfig** CR을 선택합니다.
2. **KataConfig** 페이지에서 **YAML** 탭을 선택합니다.
3. 상태에서 **installationStatus** 필드를 모니터링합니다.
   업데이트가 있을 때마다 메시지가 나타납니다. **Reload** 를 클릭하여 업데이트된 **KataConfig** CR을 확인합니다.

   완료된 노드 값이 작업자 또는 레이블이 지정된 노드 수와 같으면 설치가 완료됩니다. 상태에는 설치가 완료된 노드 목록도 포함됩니다.

2.2.3. 웹 콘솔을 사용하여 샌드박스 컨테이너에 워크로드 배포

OpenShift 샌드박스 컨테이너는 Kata를 기본 런타임이 아닌 클러스터의 보조 선택적 런타임으로 설치합니다.

샌드박스 컨테이너에 Pod 템플릿 워크로드를 배포하려면 워크로드 YAML 파일에 **kata** 를 **runtimeClassName** 으로 수동으로 추가해야 합니다.

사전 요구 사항

- 클러스터에 Red Hat OpenShift 4.13을 설치했습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 Operator가 설치되어 있습니다.
- **KataConfig** CR(사용자 정의 리소스)을 생성했습니다.

절차

1. 웹 콘솔의 **관리자** 화면에서 **워크로드** 를 확장하고 생성할 워크로드 유형을 선택합니다.
2. 워크로드 페이지에서 워크로드를 생성하려면 클릭합니다.
3. 워크로드의 YAML 파일에서 컨테이너가 나열된 **spec** 필드에서 **runtimeClassName: kata** 를 추가합니다.

   Pod 오브젝트의 예

```
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. **저장**을 클릭합니다.

Red Hat OpenShift는 워크로드를 생성하고 스케줄링을 시작합니다.

2.3. CLI를 사용하여 OPENSHIFT 샌드박스 컨테이너 워크로드 배포

CLI를 사용하여 OpenShift 샌드박스 컨테이너 워크로드를 배포할 수 있습니다. 먼저 OpenShift 샌드박스 컨테이너 Operator를 설치한 다음 **KataConfig** 사용자 지정 리소스를 생성해야 합니다. 샌드박스 컨테이너에 워크로드를 배포할 준비가 되면 워크로드 YAML 파일에 **kata** 를 **runtimeClassName** 으로 추가해야 합니다.

2.3.1. CLI를 사용하여 OpenShift 샌드박스 컨테이너 Operator 설치

Red Hat OpenShift CLI를 사용하여 OpenShift 샌드박스 컨테이너 Operator를 설치할 수 있습니다.

사전 요구 사항

- Red Hat OpenShift 4.13이 클러스터에 설치되어 있습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 카탈로그를 구독하고 있습니다.

참고

OpenShift 샌드박스 컨테이너 카탈로그를 구독하면 **openshift-sandboxed-containers-operator** 네임스페이스에서 OpenShift 샌드박스 컨테이너 Operator에 액세스할 수 있습니다.

절차

1. OpenShift 샌드박스 컨테이너 Operator의 **Namespace** 오브젝트를 생성합니다.
   a. 다음 매니페스트가 포함된 **Namespace** 오브젝트 YAML 파일을 생성합니다.

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

   b. **Namespace** 오브젝트를 생성합니다.

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

2. OpenShift 샌드박스 컨테이너 Operator의 **OperatorGroup** 오브젝트를 생성합니다.

   a. 다음 매니페스트가 포함된 **OperatorGroup** 오브젝트 YAML 파일을 생성합니다.

```
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. **OperatorGroup** 개체를 생성합니다.

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

3. **Subscription** 오브젝트를 생성하여 **네임스페이스** 에서 OpenShift 샌드박스 컨테이너 Operator를 서브스크립션합니다.

   a. 다음 매니페스트가 포함된 **Subscription** 오브젝트 YAML 파일을 생성합니다.

```
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. **Subscription** 오브젝트를 생성합니다.

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

OpenShift 샌드박스 컨테이너 Operator가 클러스터에 설치되었습니다.

참고

위에 나열된 모든 오브젝트 파일 이름은 제안 사항입니다. 다른 이름을 사용하여 오브젝트 YAML 파일을 생성할 수 있습니다.

검증

- Operator가 올바르게 설치되었는지 확인합니다.

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

출력 예

```
NAME                DISPLAY                                VERSION  REPLACES  PHASE
openshift-sandboxed-containers  openshift-sandboxed-containers-operator  1.4.1    1.4.0
Succeeded
```

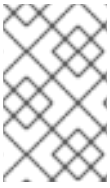
추가 리소스

- CLI를 사용하여 OperatorHub에서 설치

2.3.2. CLI를 사용하여 KataConfig 사용자 지정 리소스 생성

kata 를 노드에 **RuntimeClass** 로 설치하려면 하나의 **KataConfig** CR(사용자 정의 리소스)을 생성해야 합니다. **KataConfig** CR을 생성하면 OpenShift 샌드박스 컨테이너 Operator가 다음을 수행합니다.

- RHCOS 노드에 QEMU 및 **kata-containers** 와 같은 필요한 RHCOS 확장을 설치합니다.
- CRI-O 런타임이 올바른 런타임 처리기로 구성되었는지 확인합니다.
- 기본 구성으로 **kata** 라는 **RuntimeClass** CR을 생성합니다. 이를 통해 사용자는 **RuntimeClassName** 필드에서 CR을 참조하여 **kata** 를 런타임으로 사용하도록 워크로드를 구성할 수 있습니다. 이 CR은 런타임의 리소스 오버헤드도 지정합니다.

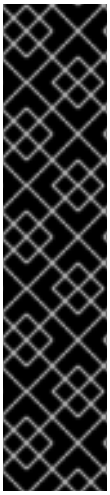


참고

Kata는 기본적으로 모든 작업자 노드에 설치됩니다. 특정 노드에서만 **kata** 를 **RuntimeClass** 로 설치하려면 해당 노드에 레이블을 추가한 다음 **KataConfig** CR에 레이블을 생성할 때 정의할 수 있습니다.

사전 요구 사항

- 클러스터에 Red Hat OpenShift 4.13을 설치했습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 Operator가 설치되어 있습니다.



중요

KataConfig CR을 생성하면 작업자 노드가 자동으로 재부팅됩니다. 재부팅하는 데 10분에서 60분 이상 걸릴 수 있습니다. 재부팅 시간을 방해하는 요소는 다음과 같습니다.

- 더 많은 수의 작업자 노드가 있는 대규모 Red Hat OpenShift 배포.
- BIOS 및 iPXE 유틸리티 활성화
- SSD가 아닌 하드 디스크 드라이브에 배포합니다.
- 가상 노드가 아닌 베어 메탈과 같은 물리적 노드에 배포.
- 느린 CPU 및 네트워크

절차

1. 다음 매니페스트를 사용하여 YAML 파일을 생성합니다.

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

(1) **kata** 를 **RuntimeClass** 로 실행할 노드 자격을 감지하려면 'checkNodeEligibility'를 **true** 로 설정합니다. 자세한 내용은 "클러스터 노드가 OpenShift 샌드박스 컨테이너를 실행할 수 있는지 여부"를 참조하십시오.

2. (선택 사항) **kata** 를 선택한 노드에서만 **RuntimeClass** 로 설치하려면 매니페스트에 라벨이 포함된 YAML 파일을 생성합니다.

```
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** 의 라벨은 단일 값만 지원합니다. **nodeSelector** 구문이 지원되지 않습니다.

3. **KataConfig** 리소스를 생성합니다.

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

새로운 **KataConfig** CR이 생성되고 작업자 노드에 **RuntimeClass** 로 **kata** 를 설치하기 시작합니다. **kata** 설치가 완료되고 작업자 노드가 재부팅될 때까지 기다린 후 다음 단계를 계속합니다.

중요

OpenShift 샌드박스 컨테이너는 **kata** 를 기본 런타임이 아닌 클러스터의 선택적 런타임으로만 설치합니다.

검증

- 설치 진행 상황을 모니터링합니다.

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

Is In Progress의 값이 **false** 로 표시되면 설치가 완료됩니다.

추가 리소스

- 노드에서 라벨을 업데이트하는 방법 이해

2.3.3. CLI를 사용하여 샌드박스 컨테이너에 워크로드 배포

OpenShift 샌드박스 컨테이너는 Kata를 기본 런타임이 아닌 클러스터의 보조 선택적 런타임으로 설치합니다.

샌드박스 컨테이너에 Pod 템플릿 워크로드를 배포하려면 워크로드 YAML 파일에 **kata** 를 **runtimeClassName** 으로 추가해야 합니다.

사전 요구 사항

- 클러스터에 Red Hat OpenShift 4.13을 설치했습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 Operator가 설치되어 있습니다.
- **KataConfig** CR(사용자 정의 리소스)을 생성했습니다.

절차

- pod 템플릿 오브젝트에 **runtimeClassName: kata** 를 추가합니다.
  - **Pod** 오브젝트
  - **ReplicaSet** 오브젝트
  - **ReplicationController** 오브젝트
  - **StatefulSet** 오브젝트
  - **Deployment** 오브젝트
  - **DeploymentConfig** 오브젝트

  Pod 오브젝트의 경우 .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 OpenShift는 워크로드를 생성하고 스케줄링을 시작합니다.

검증

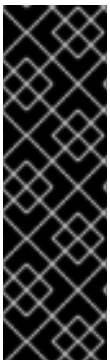
- Pod 템플릿 오브젝트에서 **runtimeClassName** 필드를 검사합니다. **runtimeClassName** 이 **kata** 인 경우 워크로드가 OpenShift 샌드박스 컨테이너에서 실행됩니다.

2.4. 추가 리소스

- OpenShift 샌드박스 컨테이너 Operator는 제한된 네트워크 환경에서 지원됩니다. 자세한 내용은 제한된 네트워크에서 Operator Lifecycle Manager 사용을 참조하십시오.
- 제한된 네트워크에서 연결이 끊긴 클러스터를 사용하는 경우 OperatorHub에 액세스하도록 Operator Lifecycle Manager에서 프록시 지원을 구성해야 합니다. 프록시를 사용하면 클러스터에서 OpenShift 샌드박스 컨테이너 Operator를 가져올 수 있습니다.

3장. 피어 POD를 사용하여 OPENSHIFT 샌드박스 컨테이너 워크로드 배포

웹 콘솔 또는 OpenShift CLI(**oc**)를 사용하여 OpenShift 샌드박스 컨테이너 Operator를 설치할 수 있습니다. OpenShift 샌드박스 컨테이너 Operator를 설치하기 전에 Red Hat OpenShift 클러스터를 준비해야 합니다.



중요

피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너 워크로드를 배포하는 것은 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

3.1. 사전 요구 사항

OpenShift 샌드박스 컨테이너를 설치하고 피어 Pod를 활성화하기 전에 다음 요구 사항을 충족해야 합니다.

- AWS 또는 Azure에 Red Hat OpenShift 4.13이 설치되어 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있습니다.

3.1.1. OpenShift 샌드박스 컨테이너의 피어 Pod 리소스 요구 사항 정보

피어 Pod는 다음 두 위치에서 리소스를 사용합니다.

- 작업자 노드입니다. 작업자 노드는 메타데이터, Kata shim 리소스(**containerd-shim-kata-v2**), remote-hypervisor 리소스(**cloud-api-adaptor**) 및 작업자 노드와 VM(Peer-pod 가상 머신) 간의 터널 설정을 저장합니다.
- 클라우드 인스턴스입니다. 클라우드에서 실행되는 실제 피어 포드 VM입니다.

Kubernetes 작업자 노드에 사용되는 CPU 및 메모리 리소스는 피어 Pod를 생성하는 데 사용되는 RuntimeClass(**kata-remote**) 정의에 포함된 Pod 오버헤드 에 의해 처리됩니다.

클라우드에서 실행되는 총 피어 Pod VM 수는 Kubernetes 노드 확장 리소스로 정의됩니다. 이 제한은 노드당이며 **peerpodConfig** CR(사용자 정의 리소스)의 **limit** 속성으로 설정됩니다. **peerpodconfig-openshift** 라는 **peerpodConfig** CR은 **kataConfig** CR을 생성하고 피어 Pod를 활성화할 때 생성되며 **openshift-sandboxed-containers-operator** 네임스페이스에 있습니다.

다음 **peerpodConfig** CR 예제에서는 기본 **사양** 값을 표시합니다.

```
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 기본 제한은 노드당 VM 10개입니다.

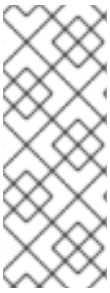
확장된 리소스의 이름은 **kata.peerpods.io/vm** 이며 Kubernetes 스케줄러에서 용량 추적 및 계정을 처리할 수 있습니다.

환경의 요구 사항에 따라 노드당 제한을 편집할 수 있습니다. 자세한 내용은 피어 Pod에서 노드당 VM 제한 수정 을 참조하십시오.

변경 웹 후크 는 확장된 리소스 **kata.peerpods.io/vm** 을 Pod 사양에 추가합니다. 또한 Pod 사양에서 리소스별 항목도 제거합니다(있는 경우). 이를 통해 Kubernetes 스케줄러에서 이러한 확장 리소스를 설명할 수 있으므로 리소스를 사용할 수 있는 경우에만 peer-pod를 예약할 수 있습니다.

변경 웹 후크는 다음과 같이 Kubernetes Pod를 수정합니다.

- 변경 웹 후크는 **TARGET_RUNTIME_CLASS** 환경 변수에 지정된 예상 **RuntimeClassName** 값을 Pod에 확인합니다. Pod 사양의 값이 **TARGET_RUNTIME_CLASS** 의 값과 일치하지 않으면 Pod를 수정하지 않고 웹 후크가 종료됩니다.
- **RuntimeClassName** 값이 일치하는 경우 Webhook에서 Pod 사양을 다음과 같이 변경합니다.
  1. Webhook는 Pod에 있는 모든 컨테이너 및 init 컨테이너의 **resources** 필드에서 모든 리소스 사양을 제거합니다.
  2. Webhook는 Pod의 첫 번째 컨테이너의 resources 필드를 수정하여 확장 리소스 (**kata.peerpods.io/vm**)를 사양에 추가합니다. 확장된 리소스 **kata.peerpods.io/vm** 은 회계 목적으로 Kubernetes 스케줄러에서 사용합니다.



참고

변경 웹 후크에서는 Red Hat OpenShift의 특정 시스템 네임스페이스가 변경되지 않습니다. 해당 시스템 네임스페이스에 피어-포드가 생성되면 Pod 사양에 확장 리소스가 포함되지 않는 한 Kubernetes 확장 리소스를 사용하는 리소스 계정이 작동하지 않습니다.

특정 네임스페이스에서 피어 Pod 생성만 허용하도록 클러스터 전체 정책을 정의하는 것이 좋습니다.

3.1.1.1. 노드당 피어 Pod VM 제한 수정

peerpodConfig CR(사용자 정의 리소스)을 편집하여 노드당 피어 Pod VM 제한을 변경할 수 있습니다.

절차

1. 다음 명령을 실행하여 현재 제한을 확인합니다.

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

2. 다음 명령을 실행하여 **peerpodConfig** CR의 **limit** 속성을 수정합니다.

```
$ oc patch peerpodconfig peerpodconfig-openshift -n openshift-sandboxed-containers-operator \
--type merge --patch '{"spec":{"limit":"<value>"}}' 1
```

1 <value>를 정의할 제한으로 바꿉니다.

3.1.2. AWS를 사용하는 피어 Pod의 사전 요구 사항

AWS를 사용하여 피어 Pod를 생성하는 경우 다음 요구 사항을 확인해야 합니다.

- Red Hat OpenShift 클러스터는 하나 이상의 작업자 노드가 있는 AWS에 설치해야 합니다.
- **AWS_ACCESS_KEY_ID** 및 **AWS_SECRET_ACCESS_KEY** 인증 정보에 액세스할 수 있습니다. 이는 클러스터의 동일한 VPC(가상 프라이빗 클라우드)에 추가 클라우드 인스턴스를 생성하는 데 사용됩니다.
- AWS CLI 툴이 설치 및 구성되어 있어야 합니다.
- 포트 15150 및 9000에서 내부 클러스터 통신을 활성화해야 합니다.
  AWS 웹 콘솔 또는 CLI를 사용하여 이러한 포트를 활성화할 수 있습니다.

3.1.2.1. AWS의 포트 15150 및 9000 활성화

절차

1. 인스턴스 ID를 검색합니다.

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

2. AWS 리전을 검색합니다.

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

3. 보안 그룹을 검색합니다.

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

4. peer-pods shim을 인증하고 kata-agent 통신에 액세스합니다. 다음 명령을 실행합니다.

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

5. peer-pods 터널을 설정합니다. 다음 명령을 실행합니다.

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

이제 포트가 활성화됩니다.

3.1.3. Azure를 사용하는 피어 Pod의 사전 요구 사항

Microsoft Azure를 사용하여 피어 Pod를 생성하는 경우 다음 요구 사항을 확인해야 합니다.

- Red Hat OpenShift 클러스터는 하나 이상의 작업자 노드가 있는 Azure에 설치해야 합니다.
- 다음 인증 정보 및 서브스크립션 세부 정보에 액세스할 수 있습니다.
  - **AZURE_SUBSCRIPTION_ID**
  - **AZURE_CLIENT_ID**
  - **AZURE_CLIENT_SECRET**
  - **AZURE_TENANT_ID**

  이는 클러스터의 동일한 VPC(가상 프라이빗 클라우드)에 추가 클라우드 인스턴스를 생성하는 데 사용됩니다.
- Azure CLI 툴이 설치 및 구성되어 있어야 합니다.
- 포트 15150 및 9000에서 클러스터 통신을 활성화해야 합니다.
  Azure에서는 이러한 포트에서 내부 통신이 허용됩니다. 그러나 통신이 차단되면 Azure 웹 콘솔 또는 CLI에서 포트를 활성화할 수 있습니다.

3.1.3.1. Azure의 포트 15150 및 9000을 활성화합니다.

절차

1. 인스턴스 ID를 검색합니다.

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

2. Azure 리소스 그룹을 검색합니다.

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

3. Azure 네트워크 보안 그룹(NSG) 이름을 검색합니다.

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

4. Azure VNet 이름을 검색합니다.

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

5. Azure 서브넷 이름을 검색합니다.

```
$ 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. Azure 서브넷 접두사를 검색합니다.

```
$ 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. peer-pods shim이 kata-agent 통신에 액세스하도록 권한을 부여합니다. 다음 명령을 실행합니다.

```
$ 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. peer-pods 터널을 설정합니다. 다음 명령을 실행합니다.

```
$ 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
```

이제 포트가 활성화됩니다.

3.2. 웹 콘솔과 함께 피어 POD를 사용하여 OPENSHIFT 샌드박스 컨테이너 워크로드 배포

웹 콘솔에서 OpenShift 샌드박스 컨테이너 워크로드를 배포할 수 있습니다. 먼저 OpenShift 샌드박스 컨테이너 Operator를 설치한 다음 시크릿 오브젝트, VM 이미지 및 피어 포드 ConfigMap을 생성해야 합니다. 보안 오브젝트 및 ConfigMap은 클라우드 공급자에 따라 고유합니다. 마지막으로 **KataConfig** CR(사용자 정의 리소스)을 생성해야 합니다. 샌드박스 컨테이너에 워크로드를 배포할 준비가 되면 워크로드 YAML 파일에 **kata-remote-cc** 를 **runtimeClassName** 으로 수동으로 추가해야 합니다.

3.2.1. 웹 콘솔을 사용하여 OpenShift 샌드박스 컨테이너 Operator 설치

Red Hat OpenShift 웹 콘솔에서 OpenShift 샌드박스 컨테이너 Operator를 설치할 수 있습니다.

사전 요구 사항

- Red Hat OpenShift 4.13이 설치되어 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

절차

1. 웹 콘솔의 **관리자** 화면에서 **Operator → OperatorHub** 로 이동합니다.
2. **키워드로 필터링** 필드에 **OpenShift sandboxed containers**를 입력합니다.
3. **OpenShift 샌드박스 컨테이너** 타일을 선택합니다.
4. Operator에 대한 정보를 확인하고 **Install**을 클릭합니다.
5. **Operator 설치** 페이지에서 다음을 수행합니다.
   a. 사용 가능한 **업데이트 채널** 옵션 목록에서 **stable**을 선택합니다.
   b. 설치된 네임스페이스에 대해 **Operator 권장 네임스페이스** 가 선택되어 **있는지 확인합니다**. 그러면 필수 **openshift-sandboxed-containers-operator** 네임스페이스에 Operator가 설치됩니다. 이 네임스페이스가 아직 존재하지 않으면 자동으로 생성됩니다.

참고

openshift-sandboxed-containers-operator 이외의 네임스페이스에 OpenShift 샌드박스 컨테이너 Operator를 설치하려고 하면 설치가 실패합니다.

   c. **승인 전략에 대해 자동 이 선택되어 있는지 확인합니다**. **자동** 은 기본값이며 새로운 z-stream 릴리스가 제공되면 OpenShift 샌드박스 컨테이너에 대한 자동 업데이트를 활성화합니다.
6. **설치**를 클릭합니다.

OpenShift 샌드박스 컨테이너 Operator가 클러스터에 설치되었습니다.

검증

1. 웹 콘솔의 **관리자** 화면에서 **Operator → 설치된 Operator 로 이동합니다**.
2. OpenShift 샌드박스 컨테이너 Operator가 Operator 목록에 나열되어 있는지 확인합니다.

3.2.2. 웹 콘솔을 사용하여 AWS의 피어 Pod 매개변수 구성

AWS에서 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너를 배포하려면 보안 오브젝트 및 ConfigMap을 생성해야 합니다.

보안 오브젝트를 생성한 후에도 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너를 배포하기 위해 **KataConfig** CR(사용자 정의 리소스)을 생성해야 합니다.

사전 요구 사항

- 클러스터에 Red Hat OpenShift 4.13을 설치했습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 Operator가 설치되어 있습니다.

3.2.2.1. 웹 콘솔을 사용하여 AWS의 보안 오브젝트 생성

AWS 액세스 키를 설정하고 secret 오브젝트에서 네트워크를 구성합니다. secret 오브젝트는 Pod VM 이미지를 생성하고 피어 Pod에서 사용하는 데 사용됩니다.

AWS의 보안 오브젝트를 생성할 때 특정 환경 값을 설정해야 합니다. 보안 오브젝트를 생성하기 전에 이러한 값 중 일부를 검색할 수 있습니다. 이러한 값 검색은 CLI를 사용하여 수행해야 합니다. 자세한 내용은 CLI를 사용하여 AWS의 보안 오브젝트 생성을 참조하십시오.

또한 AWS 웹 콘솔에서 다음 값을 찾아서 준비해야 합니다.

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

절차

1. 웹 콘솔의 **관리자** 화면에서 **Operator → 설치된 Operator 로 이동합니다.**
2. Operator 목록에서 OpenShift 샌드박스 컨테이너 Operator를 선택합니다.
3. 오른쪽 상단에 있는 가져오기 아이콘(+)을 클릭합니다.
4. **YAML 가져오기 창에서 다음 YAML** 매니페스트를 붙여넣습니다.

```
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. 시작하기 전에 준비한 **AWS_ACCESS_KEY_ID** 값을 입력합니다.
2. 시작하기 전에 준비한 **AWS_SECRET_ACCESS_KEY** 값을 입력합니다.
3. 검색한 **AWS_REGION** 값을 입력합니다.
4. 검색한 **AWS_SUBNET_ID** 값을 입력합니다.
5. 검색한 **AWS_VPC_ID** 값을 입력합니다.

6 검색한 **AWS_SG_IDS** 값을 입력합니다.

5. **생성**을 클릭합니다.

보안 오브젝트가 생성됩니다. **워크로드 → 시크릿** 아래에 나열되는 것을 확인할 수 있습니다.

3.2.2.2. 웹 콘솔을 사용하여 AWS VM 이미지(AMI) 생성

AWS에서 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너를 실행하려면 먼저 AWS 계정 및 리소스를 사용하여 RHEL AMI를 생성해야 합니다.

절차

1. 웹 콘솔의 **관리자** 화면에서 **워크로드 → 작업으로 이동합니다**.
2. **작업** 창에서 왼쪽 상단 모서리에서 **openshift-sandboxed-containers-operator** 프로젝트에 있는지 확인합니다.
3. **Create Job**을 클릭합니다.
4. **Create Job** 창에서 이 전체 YAML 매니페스트 를 붙여넣습니다.
5. **생성**을 클릭합니다.

이미지가 생성됩니다.

참고

이 이미지는 OpenShift 샌드박스 컨테이너에서 관리되지 않습니다. 필요한 경우 AWS 웹 콘솔 또는 AWS CLI 툴을 사용하여 삭제할 수 있습니다.

이미지가 생성되면 피어-pod ConfigMap을 사용하여 이미지를 설정해야 합니다.

3.2.2.3. 웹 콘솔을 사용하여 AWS의 피어 Pod ConfigMap 생성

AWS용 ConfigMap을 생성할 때 AMI ID를 설정해야 합니다. ConfigMap을 생성하기 전에 이 값을 검색할 수 있습니다. 이 값을 검색하는 작업은 CLI를 사용하여 수행해야 합니다. 자세한 내용은 CLI를 사용하여 AWS의 피어 Pod ConfigMap 생성을 참조하십시오.

절차

1. 웹 콘솔의 **관리자** 화면에서 **Operator → 설치된 Operator 로 이동합니다**.
2. Operator 목록에서 OpenShift 샌드박스 컨테이너 Operator를 선택합니다.
3. 오른쪽 상단에 있는 가져오기 아이콘(+)을 클릭합니다.
4. **YAML 가져오기 창에서 다음 YAML** 매니페스트를 붙여넣습니다.

```
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 검색한 **PODVM_AMI_ID** 값을 입력합니다.

5. **생성**을 클릭합니다.

ConfigMap 오브젝트가 생성됩니다. **워크로드 → ConfigMaps** 에서 나열되는 것을 확인할 수 있습니다.

KataConfig CR을 생성하면 AWS에서 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너를 실행할 수 있습니다.

3.2.3. 웹 콘솔을 사용하여 Azure의 피어 Pod 매개변수 구성

Microsoft Azure의 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너를 배포하려면 시크릿 오브젝트 및 ConfigMap을 생성해야 합니다.

보안 오브젝트를 생성한 후에도 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너를 배포하기 위해 **KataConfig** CR(사용자 정의 리소스)을 생성해야 합니다.

사전 요구 사항

- 클러스터에 Red Hat OpenShift 4.13을 설치했습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 Operator가 설치되어 있습니다.

3.2.3.1. 웹 콘솔을 사용하여 Azure의 보안 오브젝트 생성

Azure 액세스 키를 설정하고 시크릿 오브젝트에서 네트워크를 구성합니다. secret 오브젝트는 Pod VM 이미지를 생성하고 피어 Pod에서 사용하는 데 사용됩니다.

Azure의 보안 오브젝트를 생성할 때 특정 환경 값을 설정해야 합니다. 보안 오브젝트를 생성하기 전에 이러한 값 중 일부를 검색할 수 있습니다. 이러한 값 검색은 CLI를 사용하여 수행해야 합니다. 자세한 내용은 CLI를 사용하여 Azure의 보안 오브젝트 생성을 참조하십시오.

또한 Azure 웹 콘솔에서 다음 값을 찾아서 준비해야 합니다.

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

절차

1. 웹 콘솔의 **관리자** 화면에서 **Operator → 설치된 Operator 로 이동합니다**.
2. Operator 목록에서 OpenShift 샌드박스 컨테이너 Operator를 선택합니다.

3. 오른쪽 상단에 있는 가져오기 아이콘(+)을 클릭합니다.

4. **YAML 가져오기 창에서 다음 YAML** 매니페스트를 붙여넣습니다.

```
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. 시작하기 전에 준비한 **AZURE_CLIENT_ID** 값을 입력합니다.
2. 시작하기 전에 준비한 **AZURE_CLIENT_SECRET** 값을 입력합니다.
3. 시작하기 전에 준비한 **AZURE_TENANT_ID** 값을 입력합니다.
4. 검색한 **AZURE_SUBSCRIPTION_ID** 값을 입력합니다.
5. 검색한 **AZURE_REGION** 값을 입력합니다.
6. 검색한 **AZURE_RESOURCE_GROUP** 값을 입력합니다.

5. **생성**을 클릭합니다.

보안 오브젝트가 생성됩니다. **워크로드 → 시크릿** 아래에 나열되는 것을 확인할 수 있습니다.

3.2.3.2. 웹 콘솔을 사용하여 Azure VM 이미지 생성

Azure에서 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너를 실행하려면 먼저 Azure 계정 및 리소스를 사용하여 Azure용 RHEL 이미지를 생성해야 합니다.

절차

1. 웹 콘솔의 **관리자** 화면에서 **워크로드 → 작업으로 이동합니다**.
2. **작업** 창에서 왼쪽 상단 모서리에서 openshift-sandboxed-containers-operator 프로젝트에 있는지 확인합니다.
3. **Create Job**을 클릭합니다.
4. **Create Job** 창에서 이 전체 YAML 매니페스트 를 붙여넣습니다.
5. **생성**을 클릭합니다.

이미지가 생성됩니다.

참고

이 이미지는 OpenShift 샌드박스 컨테이너에서 관리되지 않습니다. 필요한 경우 Azure 웹 콘솔 또는 Azure CLI 도구를 사용하여 삭제할 수 있습니다.

이미지가 생성되면 피어-pod ConfigMap을 사용하여 이미지를 설정해야 합니다.

3.2.3.3. 웹 콘솔을 사용하여 Azure용 피어 Pod ConfigMap 생성

Azure용 ConfigMap을 생성할 때 특정 구성 값을 설정해야 합니다. 이러한 값 검색은 CLI를 사용하여 수행해야 합니다. 자세한 내용은 CLI를 사용하여 Azure의 피어 Pod ConfigMap 생성을 참조하십시오 .

절차

1. 웹 콘솔의 **관리자** 화면에서 **Operator → 설치된 Operator 로 이동합니다**.
2. Operator 목록에서 OpenShift 샌드박스 컨테이너 Operator를 선택합니다.
3. 오른쪽 상단에 있는 가져오기 아이콘(+)을 클릭합니다.
4. **YAML 가져오기 창에서 다음 YAML** 매니페스트를 붙여넣습니다.

```
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) 검색한 **AZURE_SUBNET_ID** 값을 입력합니다.

(2) 검색한 **AZURE_NSG_ID** 값을 입력합니다.

(3) 검색한 **AZURE_IMAGE_ID** 값을 입력합니다.

5. **생성**을 클릭합니다.

ConfigMap 오브젝트가 생성됩니다. **워크로드 → ConfigMaps** 에서 나열되는 것을 확인할 수 있습니다.

3.2.3.4. 웹 콘솔을 사용하여 Azure용 SSH 키 시크릿 오브젝트 생성

Azure에서 피어 Pod를 사용하려면 SSH 키 시크릿 오브젝트를 생성해야 합니다. 오브젝트를 생성할 SSH 키가 없는 경우 CLI를 사용하여 생성해야 합니다. 자세한 내용은 다음을 참조하십시오.

절차

1. 웹 콘솔의 **관리자** 화면에서 **워크로드 → 시크릿** 으로 이동합니다.
2. **시크릿** 창에서 왼쪽 상단 모서리에서 **openshift-sandboxed-containers-operator** 프로젝트에 있는지 확인합니다.
3. **생성** 을 클릭하고 목록에서 **키/값 시크릿** 을 선택합니다.
4. **시크릿 이름** 필드에 **ssh-key-secret** 을 입력합니다.
5. **키** 필드에 **id_rsa.pub** 를 입력합니다.
6. **값** 필드에 공개 SSH 키를 붙여넣습니다.
7. **생성**을 클릭합니다.

SSH 키 시크릿 오브젝트가 생성됩니다. **워크로드 → 시크릿** 아래에 나열되는 것을 확인할 수 있습니다.

KataConfig CR을 생성하면 Azure에서 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너를 실행할 수 있습니다.

3.2.4. 웹 콘솔에서 KataConfig 사용자 지정 리소스 생성

클러스터 노드에서 **kata-remote-cc** 를 **RuntimeClass** 로 설치하려면 하나의 **KataConfig** CR(사용자 정의 리소스)을 생성해야 합니다.

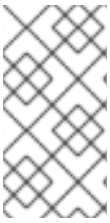
중요

KataConfig CR을 생성하면 작업자 노드가 자동으로 재부팅됩니다. 재부팅하는 데 10분에서 60분 이상 걸릴 수 있습니다. 재부팅 시간을 방해하는 요소는 다음과 같습니다.

- 더 많은 수의 작업자 노드가 있는 대규모 Red Hat OpenShift 배포.
- BIOS 및 iPXE 유틸리티 활성화
- SSD가 아닌 하드 디스크 드라이브에 배포합니다.
- 가상 노드가 아닌 베어 메탈과 같은 물리적 노드에 배포.
- 느린 CPU 및 네트워크

사전 요구 사항

- 클러스터에 Red Hat OpenShift 4.13을 설치했습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 Operator가 설치되어 있습니다.



참고

피어 Pod의 Kata는 기본적으로 모든 작업자 노드에 설치됩니다. 특정 노드에서만 **kata-remote-cc** 를 설치하려면 해당 노드에 라벨을 추가한 다음, **KataConfig** CR을 생성할 때 레이블을 정의할 수 있습니다.

절차

1. **웹 콘솔의 관리자 화면에서 Operator → 설치된 Operator 로 이동합니다.**

2. **Operator 목록에서 OpenShift 샌드박스 컨테이너 Operator를 선택합니다.**

3. **KataConfig 탭에서 KataConfig 만들기 를 클릭합니다.**

4. **KataConfig 생성 페이지에서 다음 세부 정보를 입력합니다.**

   - **name: KataConfig 리소스의 이름을 입력합니다. 기본적으로 이름은 example-kataconfig 로 정의됩니다.**

   - **레이블 (선택 사항): KataConfig 리소스에 관련된 식별 특성을 입력합니다. 각 레이블은 키-값 쌍을 나타냅니다.**

   - **CheckNodeEligibility (선택 사항, 피어 Pod와 관련이 없음): NFD(Node Feature Discovery Operator)를 사용하여 kata 를 RuntimeClass 로 실행할 자격을 감지하려면 이 확인란을 선택합니다. 자세한 내용은 "클러스터 노드가 OpenShift 샌드박스 컨테이너를 실행할 수 있는지 여부"를 참조하십시오.**

   - **EnablePeerPods (피어 Pod의 경우): 이 확인란을 선택하여 피어 Pod를 활성화하고 퍼블릭 클라우드 환경에서 OpenShift 샌드박스 컨테이너를 사용합니다.**

   - **kataConfigPoolSelector: 기본적으로 kata-remote-cc 가 모든 노드에 RuntimeClass 로 설치됩니다. 선택한 노드에서만 kata-remote-cc 를 RuntimeClass 로 설치하려면 matchExpression:을 추가해야 합니다.**

     a. **kataConfigPoolSelector 영역을 확장합니다.**

     b. **kataConfigPoolSelector 에서 matchExpressions 를 확장합니다. 라벨 선택기 요구 사항 목록입니다.**

c. **Add matchExpressions** 를 클릭합니다.

d. **키** 필드에서 선택기가 적용되는 라벨 키를 추가합니다.

e. **operator** 필드에서 레이블 값에 키의 관계를 추가합니다. 유효한 연산자는 **In,NotIn,Exists** 및 **DoesNotExist** 입니다.

f. **값** 영역을 확장한 다음 **값 추가** 를 클릭합니다.

g. **값** 필드에 키 레이블 값에 **true** 또는 **false** 를 입력합니다.

- **loglevel: kata-remote-cc** 를 **RuntimeClass** 로 실행하는 노드에 대해 검색된 로그 데이터의 수준을 정의합니다. 자세한 내용은 **"OpenShift** 샌드박스 컨테이너 데이터 조정**"**을 참조하십시오.

5. **생성**을 클릭합니다.

새로운 **KataConfig CR**이 생성되고 작업자 노드에 **RuntimeClass** 로 **kata-remote-cc** 를 설치하기 시작합니다. 설치가 완료되고 작업자 노드가 재부팅될 때까지 기다린 후 다음 단계를 진행합니다.

중요

OpenShift 샌드박스 컨테이너는 **kata-remote-cc** 를 기본 런타임이 아닌 클러스터의 선택적 런타임으로만 설치합니다.

검증

1. **KataConfig** 탭에서 새 **KataConfig CR**을 선택합니다.

2. **KataConfig** 페이지에서 **YAML** 탭을 선택합니다.

3. 상태에서 **installationStatus** 필드를 모니터링합니다.

업데이트가 있을 때마다 메시지가 나타납니다. **Reload** 를 클릭하여 업데이트된 **KataConfig CR**을 확인합니다.

완료된 노드 값이 작업자 또는 레이블이 지정된 노드 수와 같으면 설치가 완료됩니다. 상태에는 설치가 완료된 노드 목록도 포함됩니다.

3.2.5. 웹 콘솔을 사용하여 샌드박스 컨테이너에 피어 Pod로 워크로드 배포

OpenShift 샌드박스 컨테이너는 **Kata**를 기본 런타임이 아닌 클러스터의 보조 선택적 런타임으로 설치합니다.

샌드박스 컨테이너에서 피어 **Pod**를 사용하여 **Pod** 템플릿 워크로드를 배포하려면 워크로드 **YAML** 파일에 **kata-remote-cc** 를 **runtimeClassName** 으로 수동으로 추가해야 합니다.

사전 요구 사항

- 클러스터에 **Red Hat OpenShift 4.13**을 설치했습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- **OpenShift** 샌드박스 컨테이너 **Operator**가 설치되어 있습니다.
- 클라우드 공급자에 고유한 보안 오브젝트 및 피어**-pod** 구성 맵을 생성했습니다.
- **KataConfig CR**(사용자 정의 리소스)을 생성했습니다.

절차

1. 웹 콘솔의 관리자 화면에서 워크로드 를 확장하고 생성할 워크로드 유형을 선택합니다.
2. 워크로드 페이지에서 워크로드를 생성하려면 클릭합니다.
3. 워크로드에 대한 **YAML** 파일의 컨테이너가 나열된 **spec** 필드에 **runtimeClassName: kata-**

remote-cc 를 추가합니다.

Pod 오브젝트의 예

```
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. 저장을 클릭합니다.

Red Hat OpenShift는 워크로드를 생성하고 스케줄링을 시작합니다.

3.3. CLI와 함께 피어 POD를 사용하여 OPENSHIFT 샌드박스 컨테이너 워크로드 배포

CLI를 사용하여 **OpenShift** 샌드박스 컨테이너 워크로드를 배포할 수 있습니다. 먼저 **OpenShift** 샌드박스 컨테이너 **Operator**를 설치한 다음 **KataConfig** 사용자 지정 리소스를 생성해야 합니다. 샌드박스 컨테이너에 워크로드를 배포할 준비가 되면 워크로드 **YAML** 파일에 **kata-remote-cc** 를 **runtimeClassName** 으로 추가해야 합니다.

3.3.1. CLI를 사용하여 OpenShift 샌드박스 컨테이너 Operator 설치

Red Hat OpenShift CLI를 사용하여 **OpenShift** 샌드박스 컨테이너 **Operator**를 설치할 수 있습니다.

사전 요구 사항

- **Red Hat OpenShift 4.13**이 클러스터에 설치되어 있습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- **OpenShift** 샌드박스 컨테이너 카탈로그를 구독하고 있습니다.

참고

OpenShift 샌드박스 컨테이너 카탈로그를 구독하면 **openshift-sandboxed-containers-operator** 네임스페이스에서 **OpenShift** 샌드박스 컨테이너 **Operator**에 액세스할 수 있습니다.

절차

1. **OpenShift** 샌드박스 컨테이너 **Operator**의 **Namespace** 오브젝트를 생성합니다.
   a. 다음 매니페스트가 포함된 **Namespace** 오브젝트 **YAML** 파일을 생성합니다.

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

   b. **Namespace** 오브젝트를 생성합니다.

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

2. **OpenShift** 샌드박스 컨테이너 **Operator**의 **OperatorGroup** 오브젝트를 생성합니다.

a. 다음 매니페스트가 포함된 **OperatorGroup** 오브젝트 **YAML** 파일을 생성합니다.

```
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. **OperatorGroup** 개체를 생성합니다.

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

3. **Subscription** 오브젝트를 생성하여 네임스페이스 에서 **OpenShift** 샌드박스 컨테이너 **Operator**를 서브스크립션합니다.

a. 다음 매니페스트가 포함된 **Subscription** 오브젝트 **YAML** 파일을 생성합니다.

```
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. **Subscription** 오브젝트를 생성합니다.

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

OpenShift 샌드박스 컨테이너 **Operator**가 클러스터에 설치되었습니다.

참고

위에 나열된 모든 오브젝트 파일 이름은 제안 사항입니다. 다른 이름을 사용하여 오브젝트 YAML 파일을 생성할 수 있습니다.

검증

- Operator가 올바르게 설치되었는지 확인합니다.

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

출력 예

```
NAME                                DISPLAY                                VERSION  REPLACES  PHASE
openshift-sandboxed-containers.v1.4.1  openshift-sandboxed-containers-operator  1.4.1
1.4.0      Succeeded
```

추가 리소스

- CLI를 사용하여 OperatorHub에서 설치

3.3.2. CLI를 사용하여 AWS의 피어 Pod 설정

AWS에서 사용할 피어 Pod를 설정하려면 시크릿 오브젝트, AMI(AWS 이미지 VM) 및 피어 포드 ConfigMap을 생성해야 합니다.

AWS의 피어 Pod를 설정한 후에도 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너를 배포할 KataConfig CR(사용자 정의 리소스)을 계속 생성해야 합니다.

사전 요구 사항

- 클러스터에 Red Hat OpenShift 4.13을 설치했습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.

- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 Operator가 설치되어 있습니다.

3.3.2.1. CLI를 사용하여 AWS의 보안 오브젝트 생성

AWS 액세스 키를 설정하고 secret 오브젝트에서 네트워크를 구성합니다. secret 오브젝트는 Pod VM 이미지를 생성하고 피어 Pod에서 사용하는 데 사용됩니다.

AWS의 보안 오브젝트를 생성할 때 특정 환경 값을 설정해야 합니다. 보안 오브젝트를 생성하기 전에 이러한 값 중 일부를 검색할 수 있습니다. 그러나 다음과 같은 값을 준비해야 합니다.

- AWS_ACCESS_KEY_ID
- AWS_SECRET_ACCESS_KEY

절차

1. secret 오브젝트에 필요한 매개변수 값을 수집합니다. 각 값을 적어 둡니다.

   a. 인스턴스 ID를 검색합니다.

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

   이 값은 보안 오브젝트 자체에 필요하지 않지만 secret 오브젝트의 다른 값을 검색하는 데 사용됩니다.

   b. AWS 리전을 검색합니다.

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

c. AWS 서브넷 ID를 검색합니다.

```
$ 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. AWS VPC ID를 검색합니다.

```
$ 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. AWS 보안 그룹 ID를 검색합니다.

```
$ 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. 다음 매니페스트를 사용하여 YAML 파일을 생성합니다.

```
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. 시작하기 전에 준비한 AWS_ACCESS_KEY_ID 값을 입력합니다.

2. 시작하기 전에 준비한 AWS_SECRET_ACCESS_KEY 값을 입력합니다.

3.

4

검색한 **AWS_SUBNET_ID** 값을 입력합니다.

5

검색한 **AWS_VPC_ID** 값을 입력합니다.

6

검색한 **AWS_SG_IDS** 값을 입력합니다.

3. 보안 오브젝트를 적용합니다.

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

secret 오브젝트가 적용됩니다.

3.3.2.2. CLI를 사용하여 AWS VM 이미지(AMI) 생성

AWS에서 피어 **Pod**를 사용하여 **OpenShift** 샌드박스 컨테이너를 실행하려면 먼저 **AWS** 계정 및 리소스를 사용하여 **RHEL AMI**를 생성해야 합니다.

절차

1. 다음 **K8s** 작업을 실행하여 이미지를 생성합니다.

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

참고

이 이미지는 **OpenShift** 샌드박스 컨테이너에서 관리되지 않습니다. 필요한 경우 **AWS** 웹 콘솔 또는 **AWS CLI** 툴을 사용하여 삭제할 수 있습니다.

2. 작업이 완료될 때까지 기다립니다.

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

이미지가 생성되면 피어-pod ConfigMap을 사용하여 이미지를 설정해야 합니다.

3.3.2.3. CLI를 사용하여 AWS의 피어 Pod ConfigMap 생성

AWS용 ConfigMap을 생성할 때 AMI ID를 설정해야 합니다. ConfigMap을 생성하기 전에 이 값을 검색할 수 있습니다.

절차

1. AMI ID를 검색합니다. 나중에 값을 저장하기 위해 해당 값을 저장해야 합니다.

```
$ 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. 다음 매니페스트를 사용하여 YAML 파일을 생성합니다.

```
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) 검색한 AMI ID 값을 입력합니다.

3. ConfigMap을 배포합니다.

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

ConfigMap이 배포되었습니다. KataConfig CR을 생성하면 AWS에서 피어 Pod를 사용하여

OpenShift 샌드박스 컨테이너를 실행할 수 있습니다.

3.3.3. CLI를 사용하여 Azure의 피어 Pod 설정

Microsoft Azure에서 사용할 피어 Pod를 설정하려면 시크릿 오브젝트, Azure 이미지 VM, 피어 포드 ConfigMap 및 SSH 키 시크릿 오브젝트를 생성해야 합니다.

Azure의 피어 Pod를 설정한 후에도 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너를 배포하려면 KataConfig CR(사용자 정의 리소스)을 생성해야 합니다.

사전 요구 사항

- 클러스터에 Red Hat OpenShift 4.13을 설치했습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.
- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 Operator가 설치되어 있습니다.

3.3.3.1. CLI를 사용하여 Azure의 보안 오브젝트 생성

Azure 액세스 키를 설정하고 시크릿 오브젝트에서 네트워크를 구성합니다. secret 오브젝트는 Pod VM 이미지를 생성하고 피어 Pod에서 사용하는 데 사용됩니다.

Azure의 보안 오브젝트를 생성할 때 특정 환경 값을 설정해야 합니다. 보안 오브젝트를 생성하기 전에 이러한 값 중 일부를 검색할 수 있습니다. 그러나 다음과 같은 값을 준비해야 합니다.

- AZURE_CLIENT_ID
- AZURE_CLIENT_SECRET
- AZURE_TENANT_ID

절차

1. 보안 오브젝트에 대한 추가 매개변수 값을 수집합니다. 각 값을 적어 둡니다.

   a. 서브스크립션 ID를 검색합니다.

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

   b. 리소스 그룹을 검색합니다.

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

   c. Azure 리전을 검색합니다.

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

2. 다음 매니페스트를 사용하여 YAML 파일을 생성합니다.

```
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 시작하기 전에 준비한 AZURE_CLIENT_ID 값을 입력합니다.

2

시작하기 전에 준비한 **AZURE_CLIENT_SECRET** 값을 입력합니다.

3

시작하기 전에 준비한 **AZURE_TENANT_ID** 값을 입력합니다.

4

검색한 **AZURE_SUBSCRIPTION_ID** 값을 입력합니다.

5

검색한 **AZURE_REGION** 값을 입력합니다.

6

검색한 **AZURE_RESOURCE_GROUP** 값을 입력합니다.

3. 보안 오브젝트를 적용합니다.

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

secret 오브젝트가 적용됩니다.

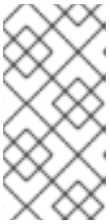
3.3.3.2. CLI를 사용하여 Azure VM 이미지 생성

Azure에서 피어 **Pod**를 사용하여 **OpenShift** 샌드박스 컨테이너를 실행하려면 먼저 **Azure** 계정 및 리소스를 사용하여 **Azure**용 **RHEL** 이미지를 생성해야 합니다.

절차

1. 다음 **K8s** 작업을 실행하여 이미지를 생성합니다.

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



참고

이 이미지는 **OpenShift** 샌드박스 컨테이너에서 관리되지 않습니다. 필요한 경우 **Azure** 웹 콘솔 또는 **Azure CLI** 도구를 사용하여 삭제할 수 있습니다.

2. 작업이 완료될 때까지 기다립니다.

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

이미지가 생성되면 피어**-pod ConfigMap**을 사용하여 이미지를 설정해야 합니다.

3.3.3.3. CLI를 사용하여 Azure용 피어 Pod ConfigMap 생성

Azure용 **ConfigMap**을 생성할 때 특정 구성 값을 설정해야 합니다. **ConfigMap**을 생성하기 전에 이러한 값을 검색할 수 있습니다.

절차

1. **Azure** 피어 포드 **ConfigMap**의 구성 값을 수집합니다. 각 값을 적어 둡니다.

   a. **Azure** 이미지 **ID**를 검색합니다.

   ```
   $ 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. **Azure VNet** 이름을 검색합니다.

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

   이 값은 **ConfigMap**에 필요하지 않지만 **Azure** 서브넷 **ID**를 검색하는 데 사용됩니다.

   c. **Azure** 서브넷 **ID**를 검색합니다.

```
$ 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. Azure 네트워크 보안 그룹(NSG) ID를 검색합니다.

```
$ 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. 다음 매니페스트를 사용하여 YAML 파일을 생성합니다.

```
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. 검색한 AZURE_SUBNET_ID 값을 입력합니다.

2. 검색한 AZURE_NSG_ID 값을 입력합니다.

3. 검색한 AZURE_IMAGE_ID 값을 입력합니다.

3. ConfigMap을 배포합니다.

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

ConfigMap이 배포되었습니다.

3.3.3.4. CLI를 사용하여 Azure용 SSH 키 시크릿 오브젝트 생성

Azure에서 피어 Pod를 사용하려면 SSH 키를 생성하고 SSH 키 시크릿 오브젝트를 생성해야 합니다.

절차

1. SSH 키를 생성합니다.

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

2. SHH 시크릿 오브젝트를 생성합니다.

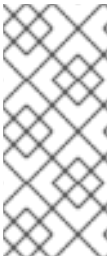
```
$ oc create secret generic ssh-key-secret -n openshift-sandboxed-containers-operator
--from-file=id_rsa.pub=./id_rsa.pub --from-file=id_rsa=./id_rsa
```

SSH 키가 생성되고 SSH 키 시크릿 오브젝트가 생성됩니다. KataConfig CR을 생성하면 Azure에서 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너를 실행할 수 있습니다.

3.3.4. CLI를 사용하여 KataConfig 사용자 지정 리소스 생성

노드에 kata-remote-cc 를 설치하려면 하나의 KataConfig CR(사용자 정의 리소스)을 생성해야 합니다. KataConfig CR을 생성하면 OpenShift 샌드박스 컨테이너 Operator가 다음을 수행합니다.

- RHCOS 노드에 QEMU 및 kata-containers 와 같은 필요한 RHCOS 확장을 설치합니다.
- CRI-O 런타임이 올바른 런타임 처리기로 구성되었는지 확인합니다.
- 기본 구성을 사용하여 kata-remote-cc 라는 RuntimeClass CR을 생성합니다. 이를 통해 사용자는 RuntimeClassName 필드에서 CR을 참조하여 kata-remote-cc 를 런타임으로 사용하도록 워크로드를 구성할 수 있습니다. 이 CR은 런타임의 리소스 오버헤드도 지정합니다.

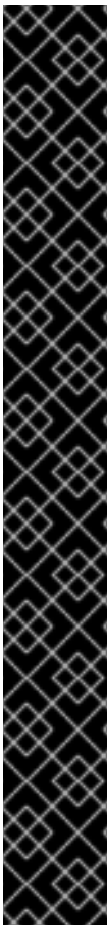


참고

피어 Pod의 Kata는 기본적으로 모든 작업자 노드에 설치됩니다. 특정 노드에서만 kata-remote-cc 를 설치하려면 해당 노드에 라벨을 추가한 다음 KataConfig CR에 레이블을 생성할 때 정의할 수 있습니다.

사전 요구 사항

- 클러스터에 Red Hat OpenShift 4.13을 설치했습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.
- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 Operator가 설치되어 있습니다.



중요

KataConfig CR을 생성하면 작업자 노드가 자동으로 재부팅됩니다. 재부팅하는 데 10분에서 60분 이상 걸릴 수 있습니다. 재부팅 시간을 방해하는 요소는 다음과 같습니다.

- 더 많은 수의 작업자 노드가 있는 대규모 Red Hat OpenShift 배포.
- BIOS 및 iPXE 유틸리티 활성화
- SSD가 아닌 하드 디스크 드라이브에 배포합니다.
- 가상 노드가 아닌 베어 메탈과 같은 물리적 노드에 배포.
- 느린 CPU 및 네트워크

절차

1. 다음 매니페스트를 사용하여 **YAML** 파일을 생성합니다.

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

2. (선택 사항) 선택한 노드에서만 **kata-remote-cc** 를 **RuntimeClass** 로 설치하려면 매니페스트에 라벨이 포함된 **YAML** 파일을 생성합니다.

```
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** 의 라벨은 단일 값만 지원합니다. **nodeSelector** 구문이 지원되지 않습니다.

3. **KataConfig** 리소스를 생성합니다.

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

새로운 **KataConfig CR**이 생성되고 작업자 노드에 **RuntimeClass** 로 **kata-remote-cc** 를 설치하기 시작합니다. **kata-remote-cc** 설치가 완료되고 작업자 노드가 재부팅될 때까지 기다린 후 다음 단계를 진행합니다.

중요

OpenShift 샌드박스 컨테이너는 **kata-remote-cc** 를 기본 런타임이 아닌 클러스터의 선택적 런타임으로만 설치합니다.

검증

- 설치 진행 상황을 모니터링합니다.

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

  Is In Progress 의 값이 false 로 표시되면 설치가 완료됩니다.

추가 리소스

- 노드에서 라벨을 업데이트하는 방법 이해

3.3.5. CLI를 사용하여 샌드박스 컨테이너에 피어 Pod를 사용하여 워크로드 배포

OpenShift 샌드박스 컨테이너는 Kata를 기본 런타임이 아닌 클러스터의 보조 선택적 런타임으로 설치합니다.

샌드박스 컨테이너에서 피어 Pod를 사용하여 Pod 템플릿 워크로드를 배포하려면 워크로드 YAML 파일에 kata-remote-cc 를 runtimeClassName 으로 추가해야 합니다.

사전 요구 사항

- 클러스터에 Red Hat OpenShift 4.13을 설치했습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.
- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift 샌드박스 컨테이너 Operator가 설치되어 있습니다.
- 클라우드 공급자에 고유한 보안 오브젝트 및 피어-pod 구성 맵을 생성했습니다.
- KataConfig CR(사용자 정의 리소스)을 생성했습니다.

절차

- pod 템플릿 오브젝트에 runtimeClassName: kata-remote-cc 를 추가합니다.
  - Pod 오브젝트
  - ReplicaSet 오브젝트
  - ReplicationController 오브젝트
  - StatefulSet 오브젝트
  - Deployment 오브젝트
  - DeploymentConfig 오브젝트

Pod 오브젝트의 예

```
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 OpenShift는 워크로드를 생성하고 스케줄링을 시작합니다.

검증

- Pod 템플릿 오브젝트에서 runtimeClassName 필드를 검사합니다. runtimeClassName 이 kata-remote-cc 인 경우 피어 Pod를 사용하여 OpenShift 샌드박스 컨테이너에서 워크로드가 실행됩니다.

3.4. 추가 리소스

- OpenShift 샌드박스 컨테이너 Operator는 제한된 네트워크 환경에서 지원됩니다. 자세한 내용은 제한된 네트워크에서 Operator Lifecycle Manager 사용을 참조하십시오.

- 제한된 네트워크에서 연결이 끊긴 클러스터를 사용하는 경우 OperatorHub에 액세스하도록 Operator Lifecycle Manager에서 프록시 지원을 구성해야 합니다. 프록시를 사용하면 클러스터에서 OpenShift 샌드박스 컨테이너 Operator를 가져올 수 있습니다.

4장. OPENSHIFT 샌드박스 컨테이너 모니터링

Red Hat OpenShift 웹 콘솔을 사용하여 샌드박스 워크로드 및 노드의 상태와 관련된 메트릭을 모니터링할 수 있습니다.

OpenShift 샌드박스 컨테이너에는 웹 콘솔에서 사용할 수 있는 사전 구성된 대시보드가 있으며 관리자는 Prometheus를 통해 원시 메트릭에 액세스하여 쿼리할 수도 있습니다.

4.1. OPENSHIFT 샌드박스 컨테이너 지표 정보

OpenShift 샌드박스 컨테이너 메트릭을 통해 관리자는 샌드박스 컨테이너 실행 방법을 모니터링할 수 있습니다. 웹 콘솔의 Metrics UI에서 이러한 메트릭을 쿼리할 수 있습니다.

다음 카테고리에 대해 OpenShift 샌드박스 컨테이너 메트릭이 수집됩니다.

Kata 에이전트 지표

Kata 에이전트 지표에는 샌드박스 컨테이너에 포함된 VM에서 실행 중인 kata 에이전트 프로세스에 대한 정보가 표시됩니다. 이러한 메트릭에는 /proc/<pid>/[io, stat, status] 의 데이터가 포함됩니다.

Kata 게스트 OS 메트릭

Kata 게스트 OS 메트릭은 샌드박스 컨테이너에서 실행 중인 게스트 OS의 데이터를 표시합니다. 이러한 메트릭에는 /proc/[stats, diskstats, meminfo, vmstats] 및 /proc/net/dev 의 데이터가 포함됩니다.

하이퍼바이저 지표

하이퍼바이저 지표에 샌드박스 컨테이너에 포함된 VM을 실행하는 하이퍼바이저에 대한 데이터가 표시됩니다. 이러한 메트릭은 주로 /proc/<pid>/[io, stat, status] 의 데이터를 포함합니다.

Kata 모니터링 메트릭

Kata monitor는 지표 데이터를 수집하고 Prometheus에서 사용할 수 있게 하는 프로세스입니다. kata 모니터 모니터 지표에는 kata-monitor 프로세스 자체의 리소스 사용량에 대한 자세한 정보가 표시됩니다. 이러한 지표에는 Prometheus 데이터 수집의 카운터도 포함됩니다.

Kata 컨테이너 shim v2 메트릭

Kata containerd shim v2 메트릭은 kata shim 프로세스에 대한 자세한 정보를 표시합니다. 이러한 메트릭에는 /proc/<pid>/[io, stat, status] 및 세부 리소스 사용량 지표의 데이터가 포함됩니다.

4.2. OPENSHIFT 샌드박스 컨테이너의 메트릭 보기

웹 콘솔의 **Metrics** 페이지에서 **OpenShift** 샌드박스 컨테이너의 메트릭에 액세스할 수 있습니다.

사전 요구 사항

- **Red Hat OpenShift 4.13**이 설치되어 있습니다.
- **OpenShift** 샌드박스 컨테이너가 설치되어 있습니다.
- **cluster-admin** 역할 또는 모든 프로젝트에 대한 보기 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.

절차

1. 웹 콘솔의 관리자 관점에서 **Observe** → **Metrics** 로 이동합니다.
2. 입력 필드에 관찰할 지표의 쿼리를 입력합니다.

   모든 **kata** 관련 지표는 **kata** 로 시작합니다. **kata** 를 입력하면 사용 가능한 모든 **kata** 지표가 포함된 목록이 표시됩니다.

쿼리의 메트릭은 페이지에 시각화됩니다.

추가 리소스

- 메트릭을 확인하는 **PromQL** 쿼리를 생성하는 방법에 대한 자세한 내용은 메트릭 쿼리를 참조하십시오.

4.3. OPENSHIFT 샌드박스 컨테이너 대시보드 보기

웹 콘솔의 대시보드 페이지에서 **OpenShift** 샌드박스 컨테이너 대시보드에 액세스할 수 있습니다.

사전 요구 사항

- Red Hat OpenShift 4.13이 설치되어 있습니다.
- OpenShift 샌드박스 컨테이너가 설치되어 있습니다.
- cluster-admin 역할 또는 모든 프로젝트에 대한 보기 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.

절차

1. 웹 콘솔의 관리자 관점에서 Observe → Dashboards 로 이동합니다.
2. 대시보드 드롭다운 목록에서 샌드박스 컨테이너 대시보드를 선택합니다.
3. 선택 사항: 시간 범위 목록에서 그래프의 시간 범위를 선택합니다.
   - 미리 정의된 기간을 선택합니다.
   - 시간 범위 목록에서 사용자 지정 시간 범위를 선택하여 사용자 지정 시간 범위를 설정합니다.
     a. 보려는 데이터의 날짜 및 시간 범위를 정의합니다.
     b. 저장을 클릭하여 사용자 지정 시간 범위를 저장합니다.
4. 선택 사항: 새로 고침 간격을 선택합니다.

대시보드는 Kata 게스트 OS 카테고리에서 다음 메트릭과 함께 페이지에 표시됩니다.

실행 중인 VM 수

클러스터에서 실행 중인 총 샌드박스 컨테이너 수를 표시합니다.

CPU 사용량(VM당)

개별 샌드박스 컨테이너의 CPU 사용량을 표시합니다.

메모리 사용량(VM당)

개별 샌드박스 컨테이너의 메모리 사용량을 표시합니다.

대시보드 내에서 각각의 그래프로 마우스를 이동하여 특정 항목에 대한 세부 정보를 표시합니다.

4.4. 추가 리소스

- 지원을 위한 데이터 수집에 대한 자세한 내용은 클러스터에 대한 데이터 수집을 참조하십시오.

5장. OPENSHIFT 샌드박스 컨테이너 설치 제거

Red Hat OpenShift 웹 콘솔 또는 OpenShift CLI(oc)를 사용하여 OpenShift 샌드박스 컨테이너를 설치 제거할 수 있습니다. 두 가지 절차는 아래에 설명되어 있습니다.

5.1. 웹 콘솔을 사용하여 OPENSHIFT 샌드박스 컨테이너 설치 제거

Red Hat OpenShift 웹 콘솔을 사용하여 관련 OpenShift 샌드박스 컨테이너 Pod, 리소스 및 네임스페이스를 삭제합니다.

5.1.1. 웹 콘솔을 사용하여 OpenShift 샌드박스 컨테이너 Pod 삭제

OpenShift 샌드박스 컨테이너를 설치 제거하려면 먼저 runtimeClass 로 kata 를 사용하는 실행 중인 모든 Pod를 삭제해야 합니다.

사전 요구 사항

- Red Hat OpenShift 4.13이 클러스터에 설치되어 있습니다.
- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- kata 를 runtimeClass 로 사용하는 Pod 목록이 있습니다.

절차

1. 관리자 화면에서 워크로드 → Pod 로 이동합니다.
2. 이름으로 검색 필드를 사용하여 삭제할 Pod를 검색합니다.
3. 포드 이름을 클릭하여 엽니다.
4. 세부 정보 페이지에서 런타임 클래스에 대해 kata 가 표시되는지 확인합니다.

5. 작업 메뉴를 클릭하고 **Pod** 삭제를 선택합니다.

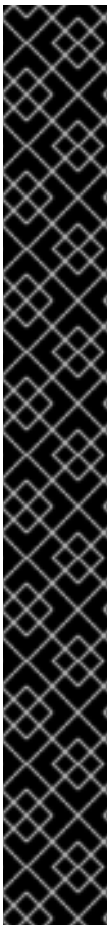
6. 확인 창에서 삭제를 클릭합니다.

추가 리소스

OpenShift CLI에서 **kata** 를 **runtimeClass** 로 사용하는 실행 중인 **Pod** 목록을 검색할 수 있습니다. 자세한 내용은 **OpenShift** 샌드박스 컨테이너 **Pod** 삭제 에서 참조하십시오.

5.1.2. 웹 콘솔을 사용하여 KataConfig 사용자 지정 리소스 삭제

KataConfig CR(사용자 정의 리소스)을 삭제하면 클러스터에서 **kata** 런타임 및 관련 리소스가 제거되고 제거됩니다.



중요

KataConfig CR을 삭제하면 작업자 노드가 자동으로 재부팅됩니다. 재부팅하는 데 **10** 분에서 **60**분 이상 걸릴 수 있습니다. 재부팅 시간을 방해하는 요소는 다음과 같습니다.

- 더 많은 수의 작업자 노드가 있는 대규모 **Red Hat OpenShift** 배포.
- **BIOS** 및 **iPXE** 유틸리티 활성화
- **SSD**가 아닌 하드 드라이브에 배포
- 가상 노드가 아닌 베어 메탈과 같은 물리적 노드에 배포.
- 느린 **CPU** 및 네트워크

사전 요구 사항

- **Red Hat OpenShift 4.13**이 클러스터에 설치되어 있습니다.

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- **kata** 를 **runtimeClass** 로 사용하는 실행 중인 **Pod**가 없습니다.

절차

1. 관리자 화면에서 **Operator** → 설치된 **Operator** 로 이동합니다.
2. 이름으로 검색 필드를 사용하여 **OpenShift** 샌드박스 컨테이너 **Operator**를 검색합니다.
3. **Operator**를 클릭하여 연 다음 **KataConfig** 탭을 선택합니다.
4. **KataConfig** 리소스의 옵션 메뉴

를 클릭한 다음 **KataConfig**삭제 를 선택합니다.

5. 확인 창에서 삭제를 클릭합니다.

kata 런타임 및 리소스가 제거될 때까지 기다린 후 다음 단계를 계속 진행하기 전에 작업자 노드가 재부팅될 때까지 기다립니다.

5.1.3. 웹 콘솔을 사용하여 OpenShift 샌드박스 컨테이너 Operator 삭제

OpenShift 샌드박스 컨테이너 **Operator**를 삭제하면 해당 **Operator**의 카탈로그 서브스크립션, **Operator group** 및 **CSV**(클러스터 서비스 버전)가 제거됩니다.

사전 요구 사항

- **Red Hat OpenShift 4.13**이 클러스터에 설치되어 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

절차

1. 관리자 화면에서 **Operator** → 설치된 **Operator** 로 이동합니다.

2. 이름으로 검색 필드를 사용하여 **OpenShift** 샌드박스 컨테이너 **Operator**를 검색합니다.

3. **Operator**의 옵션 메뉴  를 클릭하고 **Operator** 설치 제거를 선택합니다.

4. 확인 창에서 **Uninstall** 을 클릭합니다.

5.1.4. 웹 콘솔을 사용하여 OpenShift 샌드박스 컨테이너 네임스페이스 삭제

이전 명령을 실행하면 클러스터가 설치 프로세스 이전의 상태로 복원됩니다. **openshift-sandboxed-containers-operator** 네임스페이스를 삭제하여 **Operator**에 대한 네임스페이스 액세스를 취소할 수 있습니다.

사전 요구 사항

- **Red Hat OpenShift 4.13**이 클러스터에 설치되어 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

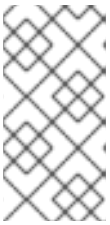
절차

1. 관리자 관점에서 관리 → 네임스페이스 로 이동합니다.

2. 이름으로 검색 필드를 사용하여 **openshift-sandboxed-containers-operator** 네임스페이스를 검색합니다.

3. 네임스페이스의 옵션 메뉴 

를 클릭하고 네임스페이스 삭제를 선택합니다.



참고

네임스페이스 삭제 옵션을 사용할 수 없으면 네임스페이스를 삭제할 수 있는 권한이 없음을 의미합니다.

4. 네임스페이스 삭제 창에서 **openshift-sandboxed-containers-operator** 를 입력하고 삭제를 클릭합니다.

5. 삭제를 클릭합니다.

5.1.5. 웹 콘솔을 사용하여 KataConfig 사용자 지정 리소스 정의 삭제

KataConfig CRD(사용자 정의 리소스 정의)를 사용하면 **KataConfig CR**을 정의할 수 있습니다. 설치 제거 프로세스를 완료하려면 클러스터에서 **KataConfig CRD**를 삭제합니다.

사전 요구 사항

- **Red Hat OpenShift 4.13**이 클러스터에 설치되어 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 클러스터에서 **KataConfig CR**을 제거했습니다.
- 클러스터에서 **OpenShift** 샌드박스 컨테이너 **Operator**를 제거했습니다.

절차

1. 관리자 관점에서 **Administration → CustomResourceDefinitions** 로 이동합니다.
2. 이름으로 검색 필드를 사용하여 **KataConfig** 를 검색합니다.
3.

KataConfig CRD의 옵션 메뉴

를 클릭한 다음 CustomResourceDefinition 삭제 를 선택합니다.

4. 확인 창에서 삭제를 클릭합니다.

5. KataConfig CRD가 목록에서 사라질 때까지 기다립니다. 이 작업은 몇 분 정도 걸릴 수 있습니다.

5.2. CLI를 사용하여 OPENSHIFT 샌드박스 컨테이너 설치 제거

Red Hat OpenShift CLI(명령줄 인터페이스) 를 사용하여 OpenShift 샌드박스 컨테이너를 설치 제거할 수 있습니다. 표시된 순서대로 다음 단계를 따릅니다.

5.2.1. CLI를 사용하여 OpenShift 샌드박스 컨테이너 Pod 삭제

OpenShift 샌드박스 컨테이너를 설치 제거하려면 먼저 runtimeClass 로 kata 를 사용하는 실행 중인 모든 Pod를 삭제해야 합니다.

사전 요구 사항

- OpenShift CLI(oc)가 설치되어 있습니다.
- jq(명령줄 JSON 프로세서)가 설치되어 있습니다.

절차

1. 다음 명령을 실행하여 kata 를 runtimeClass 로 사용하는 Pod를 검색합니다.

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

2. 각 Pod를 삭제하려면 다음 명령을 실행합니다.

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

5.2.2. CLI를 사용하여 KataConfig 사용자 지정 리소스 삭제

클러스터에서 kata 런타임 및 CRI-O 구성 및 RuntimeClass 와 같은 모든 관련 리소스를 제거하고 설치 제거합니다.

사전 요구 사항

- Red Hat OpenShift 4.13이 클러스터에 설치되어 있습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.
- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

중요

KataConfig CR을 삭제하면 작업자 노드가 자동으로 재부팅됩니다. 재부팅하는 데 10분에서 60분 이상 걸릴 수 있습니다. 재부팅 시간을 방해하는 요소는 다음과 같습니다.

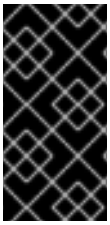
- 더 많은 수의 작업자 노드가 있는 대규모 Red Hat OpenShift 배포.
- BIOS 및 iPXE 유틸리티 활성화
- SSD가 아닌 하드 드라이브에 배포
- 가상 노드가 아닌 베어 메탈과 같은 물리적 노드에 배포.
- 느린 CPU 및 네트워크

절차

1. 다음 명령을 실행하여 KataConfig 사용자 지정 리소스를 삭제합니다.

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

OpenShift 샌드박스된 컨테이너 Operator는 클러스터에서 런타임을 활성화하기 위해 처음 생성된 모든 리소스를 제거합니다.



중요

삭제하는 동안 CLI는 모든 작업자 노드가 재부팅될 때까지 응답을 중지합니다. 확인을 수행하거나 다음 절차를 진행하기 전에 프로세스가 완료될 때까지 기다립니다.

검증

- KataConfig 사용자 지정 리소스가 삭제되었는지 확인하려면 다음 명령을 실행합니다.

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

출력 예

```
No KataConfig instances exist
```

5.2.3. CLI를 사용하여 OpenShift 샌드박스 컨테이너 Operator 삭제

Operator 서브스크립션, Operator group, CSV(클러스터 서비스 버전) 및 네임스페이스를 삭제하여 클러스터에서 OpenShift 샌드박스 컨테이너 Operator를 제거합니다.

사전 요구 사항

- Red Hat OpenShift 4.10이 클러스터에 설치되어 있어야 합니다.
- OpenShift CLI(oc)가 설치되어 있습니다.
- comand-line JSON 프로세서 (jq)를 설치했습니다.

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

절차

1. 다음 명령을 실행하여 서브스크립션에서 **OpenShift** 샌드박스 컨테이너의 **CSV(**클러스터 서비스 버전**)** 이름을 가져옵니다.

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

2. 다음 명령을 실행하여 **OLM(Operator Lifecyle Manager)**에서 **OpenShift** 샌드박스 컨테이너 **Operator** 서브스크립션을 삭제합니다.

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

3. 다음 명령을 실행하여 **OpenShift** 샌드박스 컨테이너의 **CSV** 이름을 삭제합니다.

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

4. 다음 명령을 실행하여 **OpenShift** 샌드박스 컨테이너 **Operator group** 이름을 가져옵니다.

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

5. 다음 명령을 실행하여 **OpenShift** 샌드박스 컨테이너 **Operator group** 이름을 삭제합니다.

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

6. 다음 명령을 실행하여 **OpenShift** 샌드박스 컨테이너 네임스페이스를 삭제합니다.

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

5.2.4. CLI를 사용하여 KataConfig 사용자 지정 리소스 정의 삭제

KataConfig CRD(사용자 정의 리소스 정의**)**를 사용하면 **KataConfig CR**을 정의할 수 있습니다. 클러스터에서 **KataConfig CRD**를 삭제합니다.

사전 요구 사항

- OpenShift CLI(oc)가 설치되어 있습니다.
- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 클러스터에서 KataConfig CR을 제거했습니다.
- 클러스터에서 OpenShift 샌드박스 컨테이너 Operator를 제거했습니다.

절차

1. 다음 명령을 실행하여 KataConfig CRD를 삭제합니다.

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

검증

- KataConfig CRD가 삭제되었는지 확인하려면 다음 명령을 실행합니다.

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

출력 예

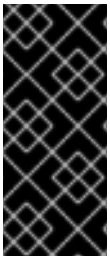
```
Unknown CR KataConfig
```

6장. OPENSHIFT 샌드박스 컨테이너 업그레이드

OpenShift 샌드박스 컨테이너 구성 요소의 업그레이드는 다음 세 단계로 구성됩니다.

- Red Hat OpenShift를 업그레이드하여 Kata 런타임 및 해당 종속 항목을 업데이트합니다.
- OpenShift 샌드박스 컨테이너 Operator를 업그레이드하여 Operator 서브스크립션을 업데이트합니다.
- KataConfig CR(사용자 정의 리소스)을 수동으로 패치하여 모니터 Pod를 업데이트합니다.

아래에 언급된 한 가지 예외를 사용하여 OpenShift 샌드박스 컨테이너 Operator 업그레이드 전이나 후에 Red Hat OpenShift를 업그레이드할 수 있습니다. 항상 OpenShift 샌드박스 컨테이너 Operator를 업그레이드한 후 KataConfig 패치를 즉시 적용합니다.



중요

OpenShift 샌드박스 컨테이너 1.3을 사용하여 Red Hat OpenShift 4.11로 업그레이드하는 경우 먼저 OpenShift 샌드박스 컨테이너를 1.2에서 1.3으로 업그레이드한 다음 Red Hat OpenShift를 4.10에서 4.11로 업그레이드하는 것이 좋습니다.

6.1. OPENSHIFT 샌드박스 컨테이너 리소스 업그레이드

OpenShift 샌드박스 컨테이너 리소스는 RHCOS(Red Hat Enterprise Linux CoreOS) 확장을 사용하여 클러스터에 배포됩니다.

RHCOS 확장 sandboxed containers에는 Kata 컨테이너 런타임, 하이퍼바이저 QEMU 및 기타 종속 항목과 같은 Kata 컨테이너를 실행하는 데 필요한 구성 요소가 포함되어 있습니다. 클러스터를 Red Hat OpenShift의 새 릴리스로 업그레이드하여 확장을 업그레이드합니다.

Red Hat OpenShift 업그레이드에 대한 자세한 내용은 클러스터 업데이트를 참조하십시오.

6.2. OPENSHIFT 샌드박스 컨테이너 OPERATOR 업그레이드

OLM(Operator Lifecycle Manager)을 사용하여 OpenShift 샌드박스 컨테이너 Operator를 수동 또는

자동으로 업그레이드합니다. 초기 배포 중에 수동 또는 자동 업그레이드를 선택하는 경우 향후 업그레이드 모드가 결정됩니다. 수동 업그레이드의 경우 웹 콘솔에는 클러스터 관리자가 설치할 수 있는 사용 가능한 업데이트가 표시됩니다.

OLM(Operator Lifecycle Manager)에서 OpenShift 샌드박스 컨테이너 Operator를 업그레이드하는 방법에 대한 자세한 내용은 설치된 Operator 업데이트를 참조하십시오.

6.3. OPENSHIFT 샌드박스 컨테이너 업그레이드에서 POD 모니터링

OpenShift 샌드박스 컨테이너를 업그레이드한 후 KataConfig CR에서 monitor 이미지를 업데이트하여 모니터 Pod를 업그레이드해야 합니다. 그렇지 않으면 모니터 Pod가 이전 버전의 이미지를 계속 실행합니다.

웹 콘솔 또는 CLI를 사용하여 업데이트를 수행할 수 있습니다.

6.3.1. 웹 콘솔을 사용하여 모니터 Pod 업그레이드

Red Hat OpenShift의 KataConfig YAML 파일에는 모니터 이미지의 버전 번호가 포함되어 있습니다. 올바른 버전으로 버전 번호를 업데이트합니다.

사전 요구 사항

- Red Hat OpenShift 4.13이 클러스터에 설치되어 있습니다.
- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

절차

1. Red Hat OpenShift의 관리자 화면에서 Operator → 설치된 Operator 로 이동합니다.
2. OpenShift 샌드박스 컨테이너 Operator 를 선택하고 KataConfig 탭으로 이동합니다.
3. 이름으로 검색 필드를 사용하여 KataConfig 리소스를 검색합니다. KataConfig 리소스의 기본 이름은 example-kataconfig 입니다.

4. **KataConfig** 리소스를 선택하고 **KataConfig** 탭으로 이동합니다.

5. **kataMonitorImage** 의 버전 번호를 수정합니다.

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

6. 저장을 클릭합니다.

6.3.2. CLI를 사용하여 모니터 Pod 업그레이드

KataConfig CR에서 모니터 이미지를 수동으로 패치하여 모니터 **Pod**를 업데이트할 수 있습니다.

사전 요구 사항

- **Red Hat OpenShift 4.13**이 클러스터에 설치되어 있습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

절차

- **Red Hat OpenShift CLI**에서 다음 명령을 실행합니다.

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

  where: <kataconfig_name > :: example-kataconfig 와 같은 **Kata** 구성 파일의 이름을 지정합니다.

7장. OPENSHIFT 샌드박스 컨테이너 데이터 수집

OpenShift 샌드박스 컨테이너의 문제를 해결할 때 지원 케이스를 열고 must-gather 툴을 사용하여 디버깅 정보를 제공할 수 있습니다.

클러스터 관리자인 경우 보다 자세한 로그를 사용하여 로그를 직접 검토할 수도 있습니다.

7.1. RED HAT 지원을 위한 OPENSHIFT 샌드박스 컨테이너 데이터 수집

지원 사례를 여는 경우 클러스터에 대한 디버깅 정보를 Red Hat 지원에 제공하면 도움이 됩니다.

must-gather 툴을 사용하면 가상 머신 및 OpenShift 샌드박스 컨테이너와 관련된 기타 데이터를 포함하여 Red Hat OpenShift 클러스터에 대한 진단 정보를 수집할 수 있습니다.

즉각 지원을 받을 수 있도록 Red Hat OpenShift 및 OpenShift 샌드박스 컨테이너 둘 다에 대한 진단 정보를 제공하십시오.

7.1.1. must-gather 툴 정보

oc adm must-gather CLI 명령은 다음과 같은 문제를 디버깅하는 데 필요한 정보를 클러스터에서 수집합니다.

- 리소스 정의
- 서비스 로그

기본적으로 oc adm must-gather 명령은 기본 플러그인 이미지를 사용하여 ./must-gather.local 에 씁니다.

또는 다음 섹션에 설명된 대로 적절한 인수를 사용하여 명령을 실행하여 특정 정보를 수집할 수 있습니다.

- 하나 이상의 특정 기능과 관련된 데이터를 수집하려면 다음 섹션에 나열된 대로 이미지에 --image 인수를 사용합니다.

예를 들면 다음과 같습니다.

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

- 감사 로그를 수집하려면 다음 섹션에 설명된 대로 **-- /usr/bin/gather_audit_logs** 인수를 사용하십시오.

  예를 들면 다음과 같습니다.

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

  참고

  감사 로그는 파일 크기를 줄이기 위해 기본 정보 세트의 일부로 수집되지 않습니다.

oc adm must-gather 를 실행하면 클러스터의 새 프로젝트에서 임의의 이름의 새 **Pod**가 생성됩니다. 해당 **Pod**에 대한 데이터가 수집되어 **must-gather.local**로 시작하는 새 디렉터리에 저장됩니다. 이 디렉터리는 현재 작업 중인 디렉터리에 생성되어 있습니다.

예를 들면 다음과 같습니다.

```
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
...
```

필요한 경우 **--run-namespace** 옵션을 사용하여 특정 네임스페이스에서 **oc adm must-gather** 명령을 실행할 수 있습니다.

예를 들면 다음과 같습니다.

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

must-gather 를 사용하여 OpenShift 샌드박스 컨테이너 데이터를 수집하려면 OpenShift 샌드박스 컨테이너 이미지를 지정해야 합니다.

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

7.2. OPENSHIFT 샌드박스 컨테이너 로그 데이터 정보

클러스터에 대한 로그 데이터를 수집할 때 다음 기능 및 오브젝트가 OpenShift 샌드박스 컨테이너와 연결됩니다.

- 모든 네임스페이스 및 OpenShift 샌드박스 컨테이너 리소스에 속하는 하위 오브젝트
- 모든 OpenShift 샌드박스 컨테이너 CRD(사용자 정의 리소스 정의)

kata 런타임을 사용하여 실행되는 각 Pod에 대해 다음 OpenShift 샌드박스 컨테이너 구성 요소 로그가 수집됩니다.

- Kata 에이전트 로그
- Kata 런타임 로그
- QEMU 로그
- 감사 로그
- CRI-O 로그

7.3. OPENSHIFT 샌드박스 컨테이너에 대한 디버그 로그 활성화

클러스터 관리자는 OpenShift 샌드박스 컨테이너에 대한 보다 자세한 로그 수준을 수집할 수 있습니다. KataConfig CR에서 logLevel 필드를 변경하여 로깅을 개선할 수도 있습니다. 이렇게 하면 OpenShift 샌드박스 컨테이너를 실행하는 작업자 노드의 CRI-O 런타임의 log_level 이 변경됩니다.

절차

1. 기존 KataConfig CR의 logLevel 필드를 debug 로 변경합니다.

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

참고

이 명령을 실행하는 경우 KataConfig CR의 이름을 참조합니다. OpenShift 샌드박스 컨테이너를 설정할 때 CR을 생성하는 데 사용한 이름입니다.

검증

1. UPDATED 필드가 True 로 표시될 때까지 kata-oc 머신 구성 풀을 모니터링합니다. 즉, 모든 작업자 노드가 업데이트됩니다.

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

출력 예

```
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. log_level 이 CRI-O에서 업데이트되었는지 확인합니다.

a. 머신 구성 풀의 노드에 oc debug 세션을 열고 chroot /host를 실행합니다.

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

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

b. **crio.conf** 파일의 변경 사항을 확인합니다.

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

출력 예

```
log_level = "debug"
```

7.3.1. OpenShift 샌드박스 컨테이너에 대한 디버그 로그 보기

클러스터 관리자는 **OpenShift** 샌드박스 컨테이너의 향상된 디버그 로그를 사용하여 문제를 해결할 수 있습니다. 각 노드의 로그가 노드 저널에 출력됩니다.

다음 **OpenShift** 샌드박스 컨테이너 구성 요소의 로그를 확인할 수 있습니다.

- **Kata** 에이전트
- **Kata** 런타임 **(containerd-shim-kata-v2)**
- **virtiofsd**

QEMU는 경고 및 오류 로그만 생성합니다. 이러한 경고 및 오류는 추가 **qemuPid** 필드를 사용하여 **Kata** 런타임 로그와 **CRI-O** 로그 모두에서 노드 저널에 출력됩니다.

QEMU 로그 예

```
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
```

Kata 런타임은 **QEMU**가 시작될 때 로깅 **QEMU** 시작 및 **QEMU**가 중지되면 로깅 **QEMU** 를 중지합니다. 이러한 두 로그 메시지 사이에 **qemuPid** 필드가 있는 오류가 표시됩니다. **QEMU**의 실제 오류 메시지가 빨간색으로 표시됩니다.

QEMU 게스트의 콘솔도 노드 저널에 출력됩니다. **Kata** 에이전트 로그와 함께 게스트 콘솔 로그를 볼 수 있습니다.

사전 요구 사항

- **OpenShift CLI(oc)**가 설치되어 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

절차

- **Kata** 에이전트 로그 및 게스트 콘솔 로그를 검토하려면 다음을 실행합니다.

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

- **kata** 런타임 로그를 검토하려면 다음을 실행합니다.

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

- **virtiofsd** 로그를 검토하려면 다음을 실행합니다.

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

- QEMU 로그를 검토하려면 다음을 실행합니다.

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

7.4. 추가 리소스

- 지원을 위한 데이터 수집에 대한 자세한 내용은 클러스터에 대한 데이터 수집을 참조하십시오.