OpenShift Container Platform 4.12

OpenShift 샌드박스 컨테이너 지원

OpenShift 샌드박스 컨테이너 가이드

Last Updated: 2023-03-18

OpenShift Container Platform 4.12 OpenShift 샌드박스 컨테이너 지원

OpenShift 샌드박스 컨테이너 가이드

법적 공지

초록

OpenShift Container Platform의 OpenShift 샌드박스형 컨테이너 지원은 추가 옵션 런타임으로 Kata Containers를 실행하는 데 대한 기본 지원을 제공합니다.

차례

1장. {SANDBOXED-CONTAINERS-FIRST} {SANDBOXED-CONTAINERS-VERSION} 릴리스 노트

1.1. 릴리스 정보

이 릴리스 노트에서는 Red Hat OpenShift Container Platform 4.12와 함께 OpenShift 샌드박스 컨테이너 1.3의 개발을 추적합니다.

이 제품은 OpenShift Container Platform 4.10부터 완전히 지원 및 활성화됩니다.

1.2. 새로운 기능 및 개선 사항

1.2.1. 메트릭 목록의 컨테이너 ID

관련 샌드박스 컨테이너의 ID가 있는 **sandbox_id** 가 이제 웹 콘솔의 **Metrics** 페이지의 메트릭 목록에 표시됩니다.

또한 **kata-monitor** 프로세스는 kata 관련 지표에 **cri_uid**,**cri_name**, **cri_namespace** 등의 세 가지 새 레이블을 추가합니다. 이러한 레이블을 사용하면 kata 관련 메트릭을 해당 kubernetes 워크로드와 연결할 수 있습니다.

kata 특정 메트릭에 대한 자세한 내용은 OpenShift 샌드박스 컨테이너 지표 정보를 참조하십시오.

1.2.2. AWS 베어 메탈에서 OpenShift 샌드박스 컨테이너 가용성

이전 버전에서는 AWS 베어 메탈의 OpenShift 샌드박스 컨테이너 가용성은 기술 프리뷰였습니다. 이번 릴리스에서는 AWS 베어 메탈 클러스터에 OpenShift 샌드박스 컨테이너를 설치할 수 있습니다.

1.2.3. 단일 노드 OpenShift에서 OpenShift 샌드박스 컨테이너 지원

이제 RHACM(Red Hat Advanced Cluster Management)에서 OpenShift 샌드박스 컨테이너 Operator를 설치할 때 OpenShift 샌드박스 컨테이너 Operator가 단일 노드 OpenShift 클러스터에서 작동합니다.

1.3. 업데이트

KataConfig 사용자 정의 리소스를 생성할 때 **kataMonitorImage** 가 더 이상 필요하지 않습니다. 이 업데이트는 OpenShift 샌드박스 컨테이너 1.3.2를 사용하여 구현되었으며 모든 OpenShift 샌드박스 컨테이너 Operator 버전에서 이전 버전과 호환됩니다.

1.4. 버그 수정

- 이전에는 OpenShift Container Platform 4.10에 OpenShift 샌드박스 컨테이너를 설치할 때 컨트롤러 관리자 POD가 다음 오류로 인해 시작되지 않았습니다.

  ```
  container has runAsNonRoot and image has non-numeric user (nonroot),
        cannot verify user is non-root
  ```

  이 문제로 인해 **KataConfig** CR을 생성할 수 없으며 OpenShift 샌드박스 컨테이너를 실행할 수 없었습니다.

이번 릴리스에서는 숫자 사용자 ID(499)를 사용하도록 관리자 컨테이너 이미지가 업데이트되었습니다. (KATA-1824)

- 이전 버전에서는 **KataConfig** CR을 생성하고 **openshift-sandboxed-containers-operator** 네임스페이스에서 Pod 상태를 관찰할 때 모니터링 Pod에 대한 많은 재시작이 표시되었습니다. 모니터 Pod는 **샌드박스-containers** 확장 설치의 일부로 설치된 특정 SELinux 정책을 사용합니다. 모니터 Pod가 즉시 생성되었습니다. 그러나 SELinux 정책을 아직 사용할 수 없어 Pod 생성 오류가 발생하고 Pod가 다시 시작되었습니다.
  이번 릴리스에서는 모니터 Pod가 생성될 때 SELinux 정책을 사용할 수 있으며 모니터 Pod가 **Running** 상태로 전환됩니다. (KATA-1338)

- 이전 버전에서는 OpenShift 샌드박스 컨테이너는 시작 시 SCC(보안 컨텍스트 제약 조건)를 배포하여 MCO(Machine Config Operator) Pod에서 사용할 수 없는 사용자 지정 SELinux 정책을 적용했습니다. 이로 인해 MCO Pod가 **CrashLoopBackOff** 상태로 변경되었으며 클러스터 업그레이드가 실패했습니다. 이번 릴리스에서는 OpenShift 샌드박스 컨테이너는 **KataConfig** CR을 생성할 때 SCC를 배포하고 사용자 지정 SELinux 정책을 사용하여 더 이상 적용하지 않습니다. (KATA-1373)

- 이전 버전에서는 OpenShift 샌드박스 컨테이너 Operator를 설치 제거할 때 **sandbox-containers-operator-scc** 사용자 정의 리소스가 삭제되지 않았습니다. 이번 릴리스에서는 OpenShift **샌드박스 컨테이너 Operator를 설치 제거할 때 sandbox-containers-operator-scc** 사용자 정의 리소스가 삭제됩니다. (KATA-1569)

1.5. 확인된 문제

- OpenShift 샌드박스 컨테이너를 사용하는 경우 컨테이너 수준에서 SELinux MCS(Multi-Category Security) 라벨을 설정하면 Pod가 다음 오류로 인해 시작되지 않습니다.

  ```
  Error: CreateContainer failed: EACCES: Permission denied: unknown
  ```

  컨테이너 수준에서 설정된 MCS 라벨은 virtiofsd에는 적용되지 않습니다. **kata** 런타임은 VM을 생성할 때 컨테이너의 보안 컨텍스트에 액세스할 수 없습니다.

  즉 virtiofsd는 적절한 SELinux 레이블로 실행되지 않으며 VM의 컨테이너를 대신하여 호스트 파일에 액세스할 수 없습니다. 모든 컨테이너는 VM의 모든 파일에 액세스할 수 있습니다.

  (KATA-1875)

- OpenShift 샌드박스 컨테이너를 사용하는 경우 OpenShift Container Platform 클러스터의 **hostPath** 볼륨에서 마운트된 파일 또는 디렉터리에 액세스할 때 SELinux 거부가 발생할 수 있습니다. 권한 있는 샌드박스 컨테이너를 실행할 때 권한 있는 샌드박스 컨테이너가 SELinux 검사를 비활성화하지 않기 때문에 이러한 거부가 발생할 수 있습니다.
  호스트에서 SELinux 정책을 수행하면 기본적으로 샌드박스 워크로드에서 호스트 파일 시스템을 완전히 격리할 수 있습니다. 또한 **virtiofsd** 데몬 또는 QEMU의 잠재적인 보안 결함에 대한 강력한 보호 기능도 제공합니다.

  마운트된 파일 또는 디렉터리에 호스트에 특정 SELinux 요구 사항이 없는 경우 대신 로컬 영구 볼륨을 사용할 수 있습니다. 컨테이너 런타임에 대한 SELinux 정책에 따라 파일이 **container_file_t**로 자동 레이블이 다시 지정됩니다. 자세한 내용은 로컬 볼륨을 사용한 영구저장장치를 참조하십시오.

  자동 레이블 지정은 마운트된 파일 또는 디렉터리가 호스트에 특정 SELinux 레이블이 있을 것으로 예상되는 경우 옵션이 아닙니다. 대신 **virtiofsd** 데몬이 이러한 특정 라벨에 액세스할 수 있도록 호스트에서 사용자 지정 SELinux 규칙을 설정할 수 있습니다. (BZ#1904609)

- 일부 OpenShift 샌드박스 컨테이너 Operator Pod는 컨테이너 CPU 리소스 제한을 사용하여 Pod

에 사용 가능한 CPU 수를 늘립니다. 이러한 Pod는 요청된 CPU보다 적은 수의 CPU를 수신할 수 있습니다. 컨테이너 내에서 기능을 사용할 수 있는 경우 **oc rsh <pod >**를 사용하여 Pod에 액세스하고 **lscpu** 명령을 실행하여 CPU 리소스 문제를 진단할 수 있습니다.

```
$ lscpu
```

출력 예

```
CPU(s):                  16
On-line CPU(s) list:     0-12,14,15
Off-line CPU(s) list:    13
```

오프라인 CPU 목록은 실행에서 실행으로 예기치 않게 변경될 수 있습니다.

이 문제를 해결하려면 Pod 주석을 사용하여 CPU 제한을 설정하지 않고 추가 CPU를 요청할 수 있습니다. 프로세서 할당 방법이 다르기 때문에 Pod 주석을 사용하는 CPU 요청은 이 문제의 영향을 받지 않습니다. CPU 제한을 설정하는 대신 Pod의 메타데이터에 다음 **주석을** 추가해야 합니다.

```
metadata:
  annotations:
    io.katacontainers.config.hypervisor.default_vcpus: "16"
```

(KATA-1376)

- 런타임 설치 진행률은 **kataConfig** CR(사용자 정의 리소스)의 **상태** 섹션에 표시됩니다. 그러나 다음 조건이 모두 해당되는 경우 진행률이 표시되지 않습니다.
  - 작업자 노드는 정의되어 있지 않습니다. **oc get machineconfigpool** 을 실행하여 머신 구성 풀의 작업자 노드 수를 확인할 수 있습니다.
  - 설치에 필요한 노드를 선택하려면 **kataConfigPoolSelector** 가 지정되지 않습니다.

  이 경우 Operator에서 컨트롤 플레인 및 작업자 역할이 모두 있는 통합 클러스터라고 가정하므로 컨트롤 플레인 노드에서 설치가 시작됩니다. **kataConfig** CR의 **status** 섹션은 설치 중에 업데이트되지 않습니다. (KATA-1017)

- OpenShift 샌드박스 컨테이너에서 이전 버전의 Buildah 툴을 사용하는 경우 빌드가 실패하고 다음 오류가 발생합니다.

  ```
  process exited with error: fork/exec /bin/sh: no such file or directory

  subprocess exited with status 1
  ```

  quay.io 에서 사용할 수 있는 최신 버전의 Buildah를 사용해야 합니다.

  (KATA-1278)

- 웹 콘솔의 KataConfig 탭에서 **YAML 보기에서 KataConfig 생성** 을 클릭하면 **KataConfig** YAML에 **spec** 필드가 없습니다. **양식 보기로 전환한 다음YAML 보기로** 돌아가 이 문제가 해결되어 전체 YAML이 표시됩니다. (KATA-1372)

- 웹 콘솔의 KataConfig 탭에서 **KataConfig** CR이 이미 존재하는지 여부에 관계없이 **404: Not found** 오류 메시지가 표시됩니다. 기존 **KataConfig** CR에 액세스하려면 **홈 > 검색** 으로 이동합니다. **리소스** 목록에서 KataConfig 를 선택합니다. (KATA-1605)

- OpenShift 샌드박스 컨테이너를 업그레이드해도 기존 **KataConfig** CR이 자동으로 업데이트되지 않습니다. 결과적으로 이전 배포의 모니터링 Pod가 다시 시작되지 않으며 오래된 **kataMonitor** 이미지로 계속 실행됩니다.
  다음 명령을 사용하여 **kataMonitor** 이미지를 업그레이드합니다.

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

  웹 콘솔에서 **KataConfig** YAML을 편집하여 **kataMonitor** 이미지를 업그레이드할 수도 있습니다.

  (KATA-1650)

1.6. 비동기 에라타 업데이트

OpenShift 샌드박스 컨테이너 4.12의 보안, 버그 수정 및 개선 업데이트는 Red Hat Network를 통해 비동기 에라타로 릴리스됩니다. 모든 OpenShift Container Platform 4.12 에라타는 Red Hat 고객 포털에서 사용할 수 있습니다. 비동기 에라타에 대한 자세한 내용은 OpenShift Container Platform 라이프 사이클을 참조하십시오.

Red Hat Customer Portal 사용자는 Red Hat 서브스크립션 관리(RHSM) 계정 설정에서 에라타 통지를 활성화할 수 있습니다. 에라타 알림이 활성화되면 사용자는 등록된 시스템과 관련된 새로운 에라타가 릴리스될 때마다 이메일로 알림을 받습니다.

참고

Red Hat Customer Portal 사용자 계정에는 OpenShift Container Platform에서 에라타 통지 이메일을 생성하기 위해 OpenShift Container Platform을 사용할 수 있는 등록된 시스템 및 권한이 필요합니다.

이 섹션은 향후 OpenShift 샌드박스 컨테이너 1.3과 관련된 비동기 에라타 릴리스의 개선 사항 및 버그 수정에 대한 정보 제공을 위해 지속적으로 업데이트됩니다.

1.6.1. RHSA-2022:6072 - OpenShift 샌드박스 컨테이너 1.3.0 이미지 릴리스, 버그 수정 및 개선 사항 권고

발행일: 2022년 8월 17일

OpenShift 샌드박스 컨테이너 릴리스 1.3.0이 공개되었습니다. 이 권고에는 개선 사항 및 버그 수정이 포함된 OpenShift 샌드박스 컨테이너에 대한 업데이트가 포함되어 있습니다.

업데이트에 포함된 버그 수정 목록은 RHSA-2022:6072 권고에 설명되어 있습니다.

1.6.2. RHSA-2022:7058 - OpenShift 샌드박스 컨테이너 1.3.1 보안 수정 및 버그 수정 권고

발행일: 2022년 10월 19일

OpenShift 샌드박스 컨테이너 릴리스 1.3.1이 공개되었습니다. 이 권고에는 보안 수정 및 버그 수정이 포함된 OpenShift 샌드박스 컨테이너에 대한 업데이트가 포함되어 있습니다.

업데이트에 포함된 버그 수정 목록은 RHSA-2022:7058 권고에 설명되어 있습니다.

1.6.3. RHBA-2023:0390 - OpenShift 샌드박스 컨테이너 1.3.2 버그 수정 권고

발행일: 2023년 1월 24일

OpenShift 샌드박스 컨테이너 릴리스 1.3.2가 공개되었습니다. 이 권고에는 버그 수정이 포함된 OpenShift 샌드박스 컨테이너에 대한 업데이트가 포함되어 있습니다.

업데이트에 포함된 버그 수정 목록은 RHBA-2023:0390 권고에 설명되어 있습니다.

1.6.4. RHBA-2023:0485 - OpenShift 샌드박스 컨테이너 1.3.3 버그 수정 권고

발행일: 2023년 1월 30일

OpenShift 샌드박스 컨테이너 릴리스 1.3.3이 공개되었습니다. 이 권고에는 버그 수정 및 컨테이너 업데이트가 포함된 OpenShift 샌드박스 컨테이너 업데이트가 포함되어 있습니다.

업데이트에 포함된 버그 수정 목록은 RHBA-2023:0485 권고에 설명되어 있습니다.

2장. OPENSHIFT 샌드박스 컨테이너 이해

OpenShift Container Platform의 OpenShift 샌드박스 컨테이너 지원은 추가 선택적 런타임으로 Kata Containers를 실행하는 데 대한 기본 지원을 제공합니다. 새로운 런타임은 전용 가상 머신(VM)에서 컨테이너를 지원하므로 워크로드 격리가 향상됩니다. 이 기능은 다음 작업을 수행하는 데 특히 유용합니다.

권한이 있거나 신뢰할 수 없는 워크로드 실행

OSC(OpenShift 샌드박스 컨테이너)를 사용하면 권한 있는 컨테이너를 실행하여 클러스터 노드를 손상시킬 수 있지만 특정 권한이 필요한 워크로드를 안전하게 실행할 수 있습니다. 특수 권한이 필요한 워크로드는 다음과 같습니다.

- CRI-O와 같은 표준 컨테이너 런타임에서 부여하는 기본 기능 이상으로 커널의 특수 기능이 필요한 워크로드(예: 낮은 수준의 네트워킹 기능에 액세스).
- 승격된 루트 권한이 필요한 워크로드(예: 특정 물리적 장치에 액세스). OpenShift 샌드박스 컨테이너를 사용하면 특정 장치만 VM에 전달하여 워크로드가 나머지 시스템에 액세스하거나 잘못 구성할 수 없도록 할 수 있습니다.
- **set-uid** 루트 바이너리를 설치하거나 사용하는 워크로드입니다. 이러한 바이너리는 특수 권한을 부여하므로 보안 위험이 발생할 수 있습니다. OpenShift 샌드박스 컨테이너를 사용하면 추가 권한이 가상 머신으로 제한되며 클러스터 노드에 대한 특별한 액세스 권한은 부여되지 않습니다.

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

각 워크로드에 대한 커널 격리 확인

OpenShift 샌드박스 컨테이너는 사용자 정의 커널 튜닝(예: **sysctl**, 스케줄러 변경 또는 캐시 튜닝) 및 사용자 정의 커널 모듈 생성(예: **트리** 또는 특수 인수)이 필요한 워크로드를 지원합니다.

테넌트 간에 동일한 워크로드 공유

OpenShift 샌드박스 컨테이너를 사용하면 동일한 OpenShift 클러스터를 공유하는 여러 조직에서 여러 사용자(테넌트)를 지원할 수 있습니다. 또한 이 시스템을 사용하면 CNF(컨테이너 네트워크 기능) 및 엔터프라이즈 애플리케이션과 같은 여러 공급업체에서 타사 워크로드를 실행할 수 있습니다. 예를 들어 타사 CNF는 사용자 지정 설정이 패킷 튜닝 또는 **sysctl** 변수를 방해하는 것을 원하지 않을 수 있습니다. 완전히 분리된 커널 내에서 실행하면 "아니요" 설정 문제가 발생하지 않도록 하는 데 도움이 됩니다.

소프트웨어 테스트를 위한 적절한 분리 및 샌드박스 확인

OpenShift 샌드박스 컨테이너를 사용하여 알려진 취약점으로 컨테이너화된 워크로드를 실행하거나 기존 애플리케이션의 문제를 처리할 수 있습니다. 또한 관리자는 이러한 격리를 통해 개발자에게 Pod에 대한 관리 제어 권한을 제공할 수 있습니다. 이는 개발자가 일반적으로 관리자가 부여한 구성을 테스트하거나 검증하려고 할 때 유용합니다. 예를 들어 관리자는 eBPF(커널 패킷 필터링)를 개발자에게 안전하고 안전하게 위임할 수 있습니다. 커널 패킷 필터링에는 **CAP_ADMIN** 또는 **CAP_BPF** 권한이 필요하므로 컨테이너 호스트 작업자 노드의 모든 프로세스에 대한 액세스 권한을 부여하므로 표준 CRI-O 구성에서 허용되지 않습니다. 마찬가지로 관리자는 SystemTap과 같은 대화형 툴에 대한 액세스 권한을 부여하거나 개발 중에 사용자 정의 커널 모듈 로드를 지원할 수 있습니다.

VM 경계를 통한 기본 리소스 제약 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 세밀한 액세스 제어를 제공합니다. 예를 들어 잘못된 컨테이너는 VM에 사용 가능한 메모리보다 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.

2.1. 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 1.3은 RHCOS(Red Hat Enterprise Linux CoreOS) 8.6에서 실행됩니다.

OpenShift 샌드박스 컨테이너 1.3은 OpenShift Container Platform 4.11과 호환됩니다.

2.2. 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 OpenShift Container Platform에서 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 컨테이너, 소프트웨어 업데이트 및 상태 모니터링과 같은 작업을 수행할 수 있습니다.

Kata 컨테이너

Kata 컨테이너는 OpenShift 샌드박스 컨테이너를 빌드하는 데 사용되는 핵심 업스트림 프로젝트입니다. OpenShift 샌드 박스 컨테이너는 Kata Container와 OpenShift Container Platform을 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.

2.3. OPENSHIFT 샌드박스 컨테이너 워크로드 관리

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

2.3.1. 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.

2.3.2. 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 옵션으로 OpenShift Container Platform 소프트웨어를 설치하기 위한 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

2.3.3. 가상화 및 OpenShift 샌드박스 컨테이너

OpenShift Virtualization의 클러스터에서 OpenShift 샌드박스 컨테이너를 사용할 수 있습니다.

OpenShift Virtualization 및 OpenShift 샌드박스 컨테이너를 동시에 실행하려면 VM을 마이그레이션하여 노드 재부팅을 차단하지 않도록 해야 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

2.4. 컴플라이언스 및 위험 관리 이해

OpenShift 샌드박스 컨테이너는 FIPS가 활성화된 클러스터에서 사용할 수 있습니다.

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

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

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

OpenShift Container Platform 컴플라이언스 프레임워크에 대한 Red Hat의 관점을 이해하려면 OpenShift 보안 가이드의 위험 관리 및 규제 준비 장을 참조하십시오.

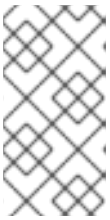
3장. OPENSHIFT 샌드박스 컨테이너 워크로드 배포

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

3.1. 사전 요구 사항

OpenShift 샌드박스 컨테이너를 설치하기 전에 OpenShift Container Platform 클러스터가 다음 요구 사항을 충족하는지 확인하십시오.

- Red Hat Enterprise Linux CoreOS (RHCOS) 작업자를 사용하여 온프레미스 베어 메탈 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) 베어 메탈 인스턴스에 OpenShift 샌드박스 컨테이너를 설치할 수 있습니다. 다른 클라우드 공급자가 제공하는 베어 메탈 인스턴스는 지원되지 않습니다.

3.1.1. OpenShift 샌드박스 컨테이너에 대한 리소스 요구 사항

OpenShift 샌드박스 컨테이너를 사용하면 사용자가 샌드박스 런타임(Kata) 내의 OpenShift Container Platform 클러스터에서 워크로드를 실행할 수 있습니다. 각 Pod는 VM(가상 머신)으로 표시됩니다. 각 VM은 QEMU 프로세스에서 실행되며 컨테이너 워크로드를 관리하기 위한 관리자 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 실행 중인 커널에 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 대해 실행되는 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.

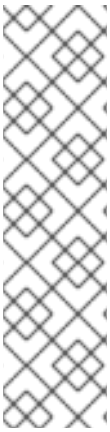
예를 들어 게스트에서 300Mi 파일 버퍼 캐시를 사용하는 경우 QEMU와 **virtiofsd**는 모두 300Mi 추가 메모리를 사용하는 것처럼 나타납니다. 그러나 세 가지 경우 모두 동일한 메모리가 사용됩니다. 즉, 총 메모리 사용량은 300Mi에 불과하며 이 값은 서로 다른 세 위치에 매핑됩니다. 이는 메모리 사용량 메트릭을 보고할 때 올바르게 계산됩니다.

추가 리소스

- 베어 메탈에 사용자 프로비저닝 클러스터 설치

3.1.2. 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 요구 사항을 준수하지 않을 수 있습니다. 노드가 불안정한 가장 일반적인 이유는 노드에서 가상화 지원이 부족하기 때문입니다. 자격이 없는 노드에서 샌드박스 워크로드를 실행하려고 하면 오류가 발생합니다. NFD(Node Feature Discovery) Operator 및 **NodeFeatureDiscovery** 리소스를 사용하여 노드 자격을 자동으로 확인할 수 있습니다.



참고

확인된 작업자 노드에만 Kata 런타임을 설치하려면 **feature.node.kubernetes.io/runtime.kata=true** 라벨을 선택한 노드에 적용하고 **KataConfig** 리소스에서 **checkNodeEligibility: true** 를 설정합니다.

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

이러한 두 시나리오에서는 **NodeFeatureDiscovery** 리소스를 생성할 필요가 없습니다. 노드에서 Kata 컨테이너를 실행할 수 있는지 확인하는 경우 **feature.node.kubernetes.io/runtime.kata=true** 라벨만 수동으로 적용해야 합니다.

다음 절차에서는 **feature.node.kubernetes.io/runtime.kata=true** 라벨을 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. **NodeFeatureDiscovery** 리소스를 생성하여 Kata 컨테이너 실행에 적합한 노드 기능을 감지합니다.

   a. **nfd.yaml** 파일에 다음 YAML을 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 CR** (사용자 정의 리소스)을 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. 다음과 같이 **KataConfig** 리소스에서 **checkNodeEligibility** 필드를 **true** 로 설정합니다.

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

3.2. 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 **kata** 를 워크로드 YAML 파일에 **runtimeClassName** 으로 수동으로 추가해야 합니다.

3.2.1. 웹 콘솔을 사용하여 OpenShift 샌드박스 컨테이너 Operator 설치

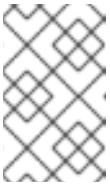
OOpenShift Container Platform 웹 콘솔에서 OpenShift 샌드박스 컨테이너 Operator를 설치할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 4.12가 설치되어 있어야 합니다.
- **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-1.3** 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 없으면 자동으로 생성됩니다.



참고

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

c. **승인 전략에 자동 이 선택되어 있는지 확인합니다. Automatic** 은 기본값이며 새 z-stream 릴리스를 사용할 수 있는 경우 OpenShift 샌드박스 컨테이너에 대한 자동 업데이트를 활성화합니다.

6. **설치**를 클릭합니다.

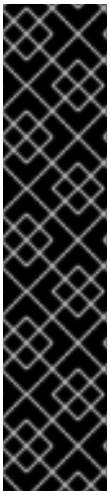
OpenShift 샌드박스 컨테이너 Operator가 클러스터에 설치되었습니다.

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 콘솔에서 KataConfig 사용자 지정 리소스 생성

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



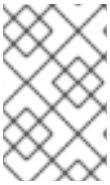
중요

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

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

사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.12가 설치되어 있습니다.
- **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 라벨을 추가한 다음 이를 생성할 때 **KataConfig** CR에 라벨을 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 만들기를** 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 속성을 입력합니다. 각 레이블은 키-값 쌍을 나타냅니다.
   - **checkNodeEligibility** (선택 사항): NFD(Node Feature Discovery Operator)를 사용하여 **kata** 를 **RuntimeClass** 로 실행하도록 노드 자격을 감지하려면 이 확인란을 선택합니다. 자세한 내용은 "클러스터 노드가 OpenShift 샌드박스 컨테이너를 실행할 수 있는지 확인"을 참조하십시오.
   - **kataConfigPoolSelector**: 기본적으로 **kata** 는 모든 노드에 **RuntimeClass** 로 설치됩니다. 선택한 노드에만 **kata** 를 **RuntimeClass** 로 설치하려면 **matchExpression** 을 추가해야 합니다.
     a. **kataConfigPoolSelector** 영역을 확장합니다.
     b. **kataConfigPoolSelector** 에서 **matchExpressions** 를 확장합니다. 레이블 선택기 요구 사항 목록입니다.
     c. **matchExpressions 추가를** 클릭합니다.
     d. **key** 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 추가를** 클릭합니다.
     g. **Value** 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 수정"을 참조하십시오.**
5. **생성을 클릭합니다.**

새 KataConfig CR이 생성되고 작업자 노드에 kata 를 RuntimeClass 로 설치합니다. kata 설치가 완료되고 작업자 노드가 재부팅될 때까지 기다린 후 다음 단계를 계속합니다.

중요

OpenShift 샌드박스 컨테이너는 Kata를 기본 런타임이 아닌 클러스터의 선택적 런타임으로만 설치합니다.

검증

1. KataConfig 탭에서 새 KataConfig CR을 선택합니다.

2. KataConfig 페이지에서 YAML 탭을 선택합니다.

3. 상태의 installationStatus 필드를 모니터링합니다.

업데이트가 있을 때마다 메시지가 표시됩니다. 다시 로드 를 클릭하여 업데이트된 KataConfig CR을 확인합니다.

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

3.2.3. 웹 콘솔을 사용하여 샌드박스 컨테이너에서 워크로드 배포

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

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

사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.12가 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

1.

1. 웹 콘솔의 관리자 화면에서 워크로드를 확장하고 생성할 워크로드 유형을 선택합니다.

2. 워크로드 페이지에서 워크로드를 생성하려면 클릭합니다.

3. 워크로드의 **YAML** 파일에서 컨테이너가 나열된 **spec** 필드에서 **runtimeClassName: kata** 를 추가합니다.

Pod의 예

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

배포 예

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

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

4. 저장을 클릭합니다.

OpenShift Container Platform은 워크로드를 생성하고 예약하기 시작합니다.

3.3. 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 정의 리소스를 생성해야 합니다. 샌드박스 컨테이너에 워크로드를 배포할 준비가 되면 kata 를 워크로드 YAML 파일에 runtimeClassName 으로 추가해야 합니다.

3.3.1. CLI를 사용하여 OpenShift 샌드박스 컨테이너 Operator 설치

OpenShift 샌드박스 컨테이너 Operator는 OpenShift Container Platform CLI를 사용하여 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.12가 설치되어 있어야 합니다.
- 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 OpenShift 샌드박스 컨테이너 Operator에 네임스페이스 를 등록합니다.

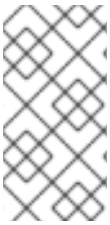
a. 다음 매니페스트를 포함하는 **Subscription** 오브젝트 **YAML** 파일을 생성합니다.

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-sandboxed-containers-operator
  namespace: openshift-sandboxed-containers-operator
spec:
  channel: "stable-1.3"
  installPlanApproval: Automatic
  name: sandboxed-containers-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: sandboxed-containers-operator.v1.3.2
```

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

- **Operator**가 올바르게 설치되었는지 확인합니다.

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

출력 예

```
NAME                                DISPLAY                                  VERSION  REPLACES  PHASE
openshift-sandboxed-containers.v1.3.2   openshift-sandboxed-containers-operator  1.3.2
1.3.1     Succeeded
```

추가 리소스

- CLI를 사용하여 OperatorHub에서 설치

3.3.2. 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 트리거되어 다음을 수행합니다.

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

참고

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

사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.12가 설치되어 있습니다.
- 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. 재부팅에는 10분에서 60분 이상 걸릴 수 있습니다. 재부팅 시간을 방해하는 요소는 다음과 같습니다.

- 더 많은 수의 작업자 노드가 있는 대규모 OpenShift Container Platform 배포
- BIOS 및ECDHE 유틸리티 활성화
- SSD가 아닌 하드 드라이브에 배포합니다.
- 가상 노드가 아닌 베어 메탈과 같은 물리적 노드에 배포합니다.
- 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) 노드 자격을 감지하려면 'checkNodeEligibility'를 true 로 설정하여 kata 를 RuntimeClass 로 실행합니다. 자세한 내용은 "클러스터 노드가 OpenShift 샌드박스 컨테이너를 실행할 수 있는지 확인"을 참조하십시오.

2. (선택 사항) 선택한 노드에만 kata 를 RuntimeClass 로 설치하려면 매니페스트에 라벨이 포

함된 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 구문은 지원되지 않습니다.

3. KataConfig 리소스를 생성합니다.

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

새 KataConfig CR이 생성되고 작업자 노드에 kata 를 RuntimeClass 로 설치합니다. kata 설치가 완료되고 작업자 노드가 재부팅될 때까지 기다린 후 다음 단계를 계속합니다.

중요

OpenShift 샌드박스 컨테이너는 Kata를 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

3.3.3. CLI를 사용하여 샌드박스 컨테이너에서 워크로드 배포

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

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

사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.12가 설치되어 있습니다.
- 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 예

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

Deployment 오브젝트의 예

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

OpenShift Container Platform은 워크로드를 생성하고 예약하기 시작합니다.

검증

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

3.4. 추가 리소스

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

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

4장. OPENSHIFT 샌드박스 컨테이너 모니터링

OpenShift Container Platform 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

관리자는 OpenShift 샌드박스 컨테이너 지표를 사용하여 샌드박스 컨테이너 실행 방법을 모니터링할 수 있습니다. 웹 콘솔의 지표 UI에서 이러한 메트릭을 쿼리할 수 있습니다.

다음 카테고리에 대해 OpenShift 샌드박스 컨테이너 지표가 수집됩니다.

Kata 에이전트 메트릭

Kata 에이전트 메트릭은 샌드박스 컨테이너에 포함된 VM에서 실행되는 kata 에이전트 프로세스에 대한 정보를 표시합니다. 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 메트릭

하이퍼바이저 지표는 샌드박스 컨테이너에 내장된 VM을 실행하는 하이퍼바이저에 관한 데이터를 표시합니다. 이러한 메트릭에는 주로 /proc/<pid>/[io, stat, status] 의 데이터가 포함됩니다.

Kata 모니터 메트릭

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

Kata containerd shim v2 지표

Kata containerd shim v2 메트릭에는 kata shim 프로세스에 대한 자세한 정보가 표시됩니다. 이러한 메트릭에는 /proc/<pid>/[io, stat, status] 및 자세한 리소스 사용 메트릭의 데이터가 포함됩니다.

4.2. OPENSHIFT 샌드박스 컨테이너의 지표 보기

웹 콘솔의 **Metrics** 페이지에서 **OpenShift** 샌드박스 컨테이너 지표에 액세스할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform 4.12**가 설치되어 있어야 합니다.
- **OpenShift** 샌드박스 컨테이너가 설치되어 있어야 합니다.
- **cluster-admin** 역할 또는 모든 프로젝트에 대한 보기 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.

절차

1. 웹 콘솔의 **Administrator** 관점에서 **Observe** → **Metrics** 로 이동합니다.
2. 입력 필드에 관찰할 지표에 대한 쿼리를 입력합니다.

   모든 **kata** 관련 메트릭은 **kata** 로 시작합니다. **kata** 를 입력하면 사용 가능한 모든 **kata** 메트릭이 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

- OpenShift Container Platform 4.12가 설치되어 있어야 합니다.
- OpenShift 샌드박스 컨테이너가 설치되어 있어야 합니다.
- cluster-admin 역할 또는 모든 프로젝트에 대한 보기 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.

절차

1. 웹 콘솔의 관리자 화면에서 Observe → Dashboards 로 이동합니다.
2. 대시보드 드롭다운 목록에서 샌드박스된 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 카테고리의 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

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

5.1. 웹 콘솔을 사용하여 OPENSHIFT 샌드박스 컨테이너 설치 제거

OpenShift Container Platform 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 kata 를 runtimeClass 로 사용하는 실행 중인 모든 Pod를 삭제해야 합니다.

사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.12가 설치되어 있어야 합니다.
- 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. 재부팅에는 **10**분에서 **60**분 이상 걸릴 수 있습니다. 재부팅 시간을 방해하는 요소는 다음과 같습니다.

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

사전 요구 사항

- 클러스터에 **OpenShift Container Platform 4.12**가 설치되어 있어야 합니다.

- **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 열고 **KataConfig** 탭을 선택합니다.
4. **KataConfig** 리소스의 옵션 메뉴

를 클릭한 다음 **KataConfig**삭제를 선택합니다.

5. 확인 창에서 삭제를 클릭합니다.

kata 런타임 및 리소스가 제거될 때까지 기다린 후 다음 단계를 계속하기 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

- 클러스터에 **OpenShift Container Platform 4.12**가 설치되어 있어야 합니다.
- **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 설치 제거를 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

- 클러스터에 **OpenShift Container Platform 4.12**가 설치되어 있어야 합니다.
- **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

- 클러스터에 **OpenShift Container Platform 4.12**가 설치되어 있어야 합니다.
- **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 삭제를 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

OpenShift Container Platform CLI(명령줄 인터페이스) 를 사용하여 OpenShift 샌드박스 컨테이너를 설치 제거할 수 있습니다. 표시되는 순서대로 아래 단계를 수행합니다.

5.2.1. CLI를 사용하여 OpenShift 샌드박스 컨테이너 Pod 삭제

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

사전 요구 사항

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

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

- 클러스터에 OpenShift Container Platform 4.12가 설치되어 있어야 합니다.
- OpenShift CLI(oc)가 설치되어 있습니다.
- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

중요

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

- 더 많은 수의 작업자 노드가 있는 대규모 OpenShift Container Platform 배포
- BIOS 및ECDHE 유틸리티 활성화
- SSD가 아닌 하드 드라이브에 배포합니다.
- 가상 노드가 아닌 베어 메탈과 같은 물리적 노드에 배포합니다.
- 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 **CLI**는 모든 작업자 노드가 재부팅될 때까지 응답하지 않습니다. 확인 작업을 수행하거나 다음 절차를 계속 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 **Operator**를 제거합니다.

사전 요구 사항

- 클러스터에 **OpenShift Container Platform 4.10**이 설치되어 있어야 합니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.
- **comand-line JSON processor (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 요소 업그레이드는 다음 세 단계로 구성됩니다.

- OpenShift Container Platform을 업그레이드하여 Kata 런타임 및 해당 종속 항목을 업데이트합니다.
- OpenShift 샌드박스 컨테이너 Operator를 업그레이드하여 Operator 서브스크립션을 업데이트합니다.
- KataConfig CR(사용자 정의 리소스)의 수동으로 패치를 적용하여 모니터 Pod를 업데이트합니다.

아래에 언급된 예외를 제외하고 OpenShift 샌드박스 컨테이너 Operator 업그레이드 전이나 후에 OpenShift Container Platform을 업그레이드할 수 있습니다. OpenShift 샌드박스 컨테이너 Operator를 업그레이드한 후 항상 KataConfig 패치를 즉시 적용합니다.

중요

OpenShift 샌드박스 컨테이너 1.3을 사용하여 OpenShift Container Platform 4.11로 업그레이드하는 경우 먼저 OpenShift 샌드박스 컨테이너를 1.2에서 1.3으로 업그레이드한 다음 OpenShift Container Platform을 4.10에서 4.11로 업그레이드하는 것입니다.

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 새 OpenShift Container Platform 릴리스로 업그레이드하여 확장 기능을 업그레이드합니다.

OpenShift Container Platform 업그레이드에 대한 자세한 내용은 클러스터 업데이트를 참조하십시오.

6.2. OPENSHIFT 샌드박스 컨테이너 OPERATOR 업그레이드

OLM(Operator Lifecycle Manager)을 사용하여 OpenShift 샌드박스 컨테이너 Operator를 수동으로 또는 자동으로 업그레이드합니다. 초기 배포 중 수동 또는 자동 업그레이드 중 하나를 선택하면 향후 업그레이드 모드가 결정됩니다. 수동 업그레이드를 위해 웹 콘솔에는 클러스터 관리자가 설치할 수 있는 사용 가능한 업데이트가 표시됩니다.

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

6.3. OPENSHIFT 샌드박스 컨테이너 업그레이드 POD 모니터링

OpenShift 샌드박스 컨테이너를 업그레이드한 후 모니터 Pod를 업그레이드하려면 KataConfig CR의 모니터 이미지를 업데이트해야 합니다. 그렇지 않으면 모니터 Pod는 이전 버전의 이미지를 계속 실행합니다.

웹 콘솔 또는 CLI를 사용하여 업데이트를 수행할 수 있습니다.

6.3.1. 웹 콘솔을 사용하여 모니터 Pod 업그레이드

OpenShift Container Platform의 KataConfig YAML 파일에는 모니터 이미지의 버전 번호가 포함되어 있습니다. 올바른 버전으로 버전 번호를 업데이트합니다.

사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.12가 설치되어 있어야 합니다.
- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

절차

1. OpenShift Container Platform의 관리자 관점에서 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 수정 :

```
  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 CR의 모니터 이미지를 수동으로 패치하여 모니터 Pod를 업데이트할 수 있습니다.

사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.12가 설치되어 있어야 합니다.
- OpenShift CLI(oc)가 설치되어 있습니다.
- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

절차

- OpenShift Container Platform CLI에서 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"}}'
```

여기서 < kataconfig_name > ::는 Kata 구성 파일의 이름 (예: example-kataconfig)을 지정합니다.

7장. OPENSHIFT 샌드박스 컨테이너 데이터 수집

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

클러스터 관리자인 경우 자체 로그를 검토하여 보다 자세한 수준의 로그를 활성화할 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 컨테이너 관련 기타 데이터를 포함하여 OpenShift Container Platform 클러스터에 대한 진단 정보를 수집할 수 있습니다.

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

7.1.1. must-gather 툴 정보

oc adm must-gather CLI 명령은 다음을 포함하여 문제를 디버깅하는 데 필요할 가능성이 높은 클러스터에서 정보를 수집합니다.

- 리소스 정의
- 서비스 로그

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

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

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

예를 들면 다음과 같습니다.

```
$ oc adm must-gather  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.12.0
```

- 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 프로젝트에 임의의 이름이 있는 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
...
```

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-rhel8:1.3.0
```

7.2. OPENSHIFT 샌드박스 컨테이너 로그 데이터 정보

클러스터에 대한 로그 데이터를 수집하면 다음 기능 및 오브젝트가 **OpenShift** 샌드박스 컨테이너와 연결됩니다.

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

kata 런타임으로 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 컨테이너의 디버그 로그 활성화

클러스터 관리자는 OpenShift 샌드박스 컨테이너에 대한 보다 자세한 수준의 로그를 수집할 수 있습니다. OpenShift 샌드박스 컨테이너를 실행하는 작업자 노드의 CRI-O 런타임에서 log_level 을 변경하여 로깅을 향상시킵니다.

절차

1. 다음 매니페스트를 사용하여 ContainerRuntimeConfig CR의 YAML 파일을 생성합니다.

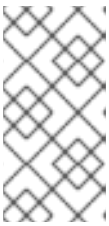
```
apiVersion: machineconfiguration.openshift.io/v1
kind: ContainerRuntimeConfig
metadata:
 name: crio-debug
spec:
 machineConfigPoolSelector:
  matchLabels:
```

```
    pools.operator.machineconfiguration.openshift.io/worker: '' (1)
containerRuntimeConfig:
  logLevel: debug
```

(1) 수정할 머신 구성 풀의 레이블을 지정합니다.

2. **ContainerRuntimeConfig CR**을 생성합니다.

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



참고

위에 나열된 파일 이름은 제안 사항입니다. 다른 이름을 사용하여 이 파일을 생성할 수 있습니다.

3. **CR**이 생성되었는지 확인합니다.

```
$ oc get ctrcfg
```

출력 예

```
NAME          AGE
crio-debug    3m19s
```

검증

1. 모든 작업자 노드의 **UPDATED** 필드가 **True** 로 표시될 때까지 머신 구성 풀을 모니터링합니다.

```
$ oc get mcp worker
```

출력 예

```
NAME    CONFIG             UPDATED  UPDATING  DEGRADED  MACHINECOUNT
READYMACHINECOUNT  UPDATEDMACHINECOUNT  DEGRADEDMACHINECOUNT
AGE
worker  rendered-worker-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 컨테이너의 디버그 로그 보기

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

다음 OpenShift 샌드박스 컨테이너 구성 요소의 로그를 확인할 수 있습니다.

- Kata 에이전트

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

QEMU의 로그는 노드 저널에 출력되지 않습니다. 그러나 QEMU 오류는 런타임에 보고되고 QEMU 게스트의 콘솔이 노드 저널에 인쇄됩니다. 이러한 로그를 Kata 에이전트 로그와 함께 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
```

7.4. 추가 리소스

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