OpenShift Container Platform 4.10

가상화

OpenShift Virtualization 설치, 사용법, 릴리스 정보

Last Updated: 2023-10-18

OpenShift Container Platform 4.10 가상화

OpenShift Virtualization 설치, 사용법, 릴리스 정보

법적 공지

초록

이 문서에서는 OpenShift Container Platform에서 OpenShift Virtualization을 사용하는 방법에 대한 정보를 제공합니다.

차례

1장. OPENSHIFT VIRTUALIZATION 정보

OpenShift Virtualization의 기능 및 지원 범위에 대해 알아보십시오.

1.1. OPENSHIFT VIRTUALIZATION으로 수행할 수 있는 작업

OpenShift Virtualization은 컨테이너 워크로드와 함께 가상 머신 워크로드를 실행하고 관리할 수 있는 OpenShift Container Platform의 애드온입니다.

OpenShift Virtualization은 Kubernetes 사용자 정의 리소스를 사용하여 가상화 작업을 활성화하여 OpenShift Container Platform 클러스터에 새 오브젝트를 추가합니다. 다음과 같은 가상화 작업이 지원됩니다.

- Linux 및 Windows 가상 머신 생성 및 관리
- 다양한 콘솔 및 CLI 툴을 통해 가상 머신에 연결
- 기존 가상 머신 가져오기 및 복제
- 가상 머신에 연결된 네트워크 인터페이스 컨트롤러 및 스토리지 디스크 관리
- 노드 간 실시간 가상 머신 마이그레이션

향상된 웹 콘솔에서 제공되는 그래픽 포털을 통해 OpenShift Container Platform 클러스터 컨테이너 및 인프라와 함께 가상화 리소스를 관리할 수 있습니다.

OpenShift Virtualization은 Red Hat OpenShift Data Foundation 기능과 함께 잘 작동하도록 설계 및 테스트되었습니다.

중요

OpenShift Data Foundation을 사용하여 OpenShift Virtualization을 배포할 때 Windows 가상 머신 디스크용 전용 스토리지 클래스를 생성해야 합니다. 자세한 내용은 Windows VM용 ODF PersistentVolume 최적화 를 참조하십시오.

OVN-Kubernetes,OpenShift SDN 또는 인증된 OpenShift CNI 플러그인에 나열된 다른 인증 기본 CNI(Container Network Interface) 네트워크 공급자 중 하나와 함께 OpenShift Virtualization을 사용할 수 있습니다.

1.1.1. OpenShift Virtualization 지원 클러스터 버전

OpenShift Virtualization 4.10은 OpenShift Container Platform 4.10 클러스터에서 사용할 수 있습니다. OpenShift Virtualization의 최신 z-stream 릴리스를 사용하려면 먼저 최신 버전의 OpenShift Container Platform으로 업그레이드해야 합니다.

2장. OPENSHIFT VIRTUALIZATION 시작하기

기본 OpenShift Virtualization 환경을 설치하고 구성하여 해당 기능 및 기능을 확인할 수 있습니다.

참고

클러스터 구성 절차에는 **cluster-admin** 권한이 필요합니다.

2.1. 사전 준비 사항

- 설치 요구 사항을 검토합니다.
- 복제, 스냅샷 및 실시간 마이그레이션에 필요한 스토리지 기능을 검토합니다. 자세한 내용은 CSI 지원 스토리지 공급자 사용을 참조하십시오.
- OpenShift Virtualization Operator 를 설치합니다.
- **virtctl** 툴 을 설치합니다.

2.2. 시작하기

가상 머신 생성

- 마법사를 사용하여 RHEL 가상 머신을 생성합니다.
- Windows 가상 머신을 생성합니다.
  - Windows 부팅 소스 를 생성하고 사용자 지정합니다.
  - 마법사를 사용하여 Windows 가상 머신을 생성합니다.
  - Windows 가상 머신에 VirtIO 드라이버 및 QEMU 게스트 에이전트를 설치합니다.

가상 머신에 연결

- 웹 콘솔을 사용하여 가상 머신의 직렬 콘솔 또는 VNC 콘솔에 연결합니다.
- SSH를 사용하여 가상 머신에 연결합니다.
- RDP를 사용하여 Windows 가상 머신에 연결합니다.

가상 머신 관리

- 웹 콘솔에서 가상 머신을 중지, 시작, 일시 중지 및 다시 시작합니다.
- **virtctl** 을 사용하여 가상 머신을 관리하고 포트를 노출하고 명령줄에서 가상 머신의 직렬 콘솔에 연결합니다.

2.3. 다음 단계

보조 네트워크에 VM 연결

- 가상 머신을 Linux 브리지 네트워크에 연결합니다 .

- 가상 머신을 SR-IOV 네트워크에 연결합니다.

OpenShift Virtualization 환경 모니터링

- **가상화 개요** 페이지에서 리소스, 세부 정보, 상태 및 상위 소비자를 모니터링합니다.
- 가상 머신 대시보드에서 가상 머신에 대한 상위 수준 정보를 봅니다.
- **가상 머신 로그를 확인합니다.**

배포 자동화

- **Ansible을 사용하여 가상 머신 배포를 자동화합니다.**
- **sysprep 로 Windows 가상 머신 배포를 자동화합니다.**

2.4. 추가 리소스

- **Kubernetes NMState Operator 정보**
- **가상 머신용 노드 지정**
- **실시간 마이그레이션**
- **가상 머신 템플릿**
- **로컬 스토리지 구성**
- **백업 및 복원**

3장. OPENSHIFT VIRTUALIZATION 릴리스 정보

3.1. RED HAT OPENSHIFT VIRTUALIZATION 정보

Red Hat OpenShift Virtualization을 사용하면 기존 VM(가상 머신)을 컨테이너와 함께 실행되는 OpenShift Container Platform으로 가져와 네이티브 Kubernetes 오브젝트로 관리할 수 있습니다.

OpenShift Virtualization은  아이콘으로 표시됩니다.

OVN-Kubernetes 또는 OpenShiftSDN 기본 CNI(컨테이너 네트워크 인터페이스) 네트워크 공급자와 함께 OpenShift Virtualization을 사용할 수 있습니다.

OpenShift Virtualization으로 수행할 수 있는 작업에 대해 자세히 알아보십시오.

3.1.1. OpenShift Virtualization 지원 클러스터 버전

OpenShift Virtualization 4.10은 OpenShift Container Platform 4.10 클러스터에서 사용할 수 있습니다. OpenShift Virtualization의 최신 z-stream 릴리스를 사용하려면 먼저 최신 버전의 OpenShift Container Platform으로 업그레이드해야 합니다.

3.1.2. 지원되는 게스트 운영 체제

OpenShift Virtualization에서 지원되는 게스트 운영 체제를 보려면 Red Hat Virtualization 및 OpenShift Virtualization에서 인증된 게스트 운영 체제, Red Hat Virtualization 및 OpenShift Virtualization 을 참조하십시오.

3.2. 보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 Red Hat CTO Chris Wright의 메시지에서 참조하십시오.

3.3. 새로운 기능 및 변경된 기능

- OpenShift Virtualization은 Microsoft의 Windows SVVP(서버 가상화 유효성 검사 프로그램)에서 Windows Server 워크로드를 실행하도록 인증되었습니다.
  SVVP 인증은 다음에 적용됩니다.
  - Red Hat Enterprise Linux CoreOS 작업자. SVVP 카탈로그에서는 *RHEL CoreOS 8의 Red Hat OpenShift Container Platform 4*라고 함)
  - Intel 및 AMD CPU
- OpenShift Virtualization이 OpenShift Service Mesh와 통합되었습니다. 가상 머신을 서비스 메시에 연결 하여 IPv4로 가상 머신 워크로드를 실행하는 pod 간 트래픽을 모니터링, 시각화 및 제어할 수 있습니다.
- OpenShift Virtualization에서는 이제 사전 정의된 부팅 소스의 자동 가져오기 및 업데이트를위한 통합 API를 제공합니다.

3.3.1. 퀵스타트

- 여러 OpenShift Virtualization 기능에 대한 퀵스타트 둘러보기를 사용할 수 있습니다. 둘러보기를 보려면 OpenShift Virtualization 콘솔 헤더에 있는 메뉴 표시줄에서 Help 아이콘?을 클릭한 다음 퀵스타트를 선택합니다. 필터 필드에 가상 머신 키워드를 입력하여 사용 가능한 둘러보기를필터링할 수 있습니다.

3.3.2. 설치

- **virt-launcher** Pod와 같은 OpenShift Virtualization 워크로드는 이제 실시간 마이그레이션을 지원하는 경우 자동으로 업데이트됩니다. **HyperConverged** 사용자 정의 리소스를 편집하여워크로드 업데이트 전략을 구성하거나 향후 자동 업데이트를 비활성화할 수 있습니다.

- 이제 SNO(Single Node OpenShift)라고도 하는 단일 노드 클러스터와 함께 OpenShift Virtualization을 사용할 수 있습니다.

참고

단일 노드 클러스터는 고가용성 작업을 위해 구성되지 않으므로 OpenShift Virtualization 동작이 크게 변경됩니다.

- 이제 모든 OpenShift Virtualization 컨트롤 플레인 구성 요소에 대해 리소스 요청 및 우선순위 클래스를 정의합니다.

3.3.3. 네트워킹

- 단일 **NodeNetworkConfigurationPolicy** 매니페스트를 사용하여 여러 nmstate 지원 노드를 동시에 구성할 수 있습니다.

- SR-IOV 네트워크 인터페이스에 연결된 가상 머신에 대해 실시간 마이그레이션이 기본적으로 지원됩니다.

3.3.4. 스토리지

- 핫플러그 가상 디스크가 있는 가상 머신에는 온라인 스냅샷이 지원됩니다. 그러나 가상 머신 사양에 없는 핫플러그 디스크는 스냅샷에 포함되지 않습니다.

- HPP(Hostpath provisioner)와 함께 Kubernetes CSI(Container Storage Interface) 드라이버를 사용하여 가상 머신의 로컬 스토리지를 구성할 수 있습니다. CSI 드라이버를 사용하면 로컬 스토리지를 구성할 때 기존 OpenShift Container Platform 노드 및 클러스터의 중단이 최소화됩니다.

3.3.5. 웹 콘솔

- OpenShift Virtualization 대시보드는 가상 머신 및 관련 Pod에 대한 리소스 소비 데이터를 제공합니다. OpenShift Virtualization 대시보드에 표시된 시각화 지표는 Prometheus Query Language(PromQL) 쿼리를 기반으로 합니다.

3.4. 사용되지 않거나 삭제된 기능

3.4.1. 더 이상 사용되지 않는 기능

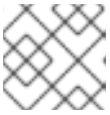
더 이상 사용되지 않는 기능은 현재 릴리스에 포함되어 있으며 지원됩니다. 그러나 향후 릴리스에서 제거될 예정이므로 새로운 배포에는 사용하지 않는 것이 좋습니다.

- 향후 릴리스에서는 레거시 HPP 사용자 지정 리소스 및 관련 스토리지 클래스에 대한 지원이 더 이상 사용되지 않습니다. OpenShift Virtualization 4.10부터 HPP Operator는 Kubernetes CSI(Container Storage Interface) 드라이버를 사용하여 로컬 스토리지를 구성합니다. Operator는 HPP 사용자 지정 리소스 및 관련 스토리지 클래스의 기존(기존) 형식을 계속 지원합니다. HPP Operator를 사용하는 경우 마이그레이션 전략의 일부로 CSI 드라이버의 스토리지 클래스를 생성할 계획입니다.

3.4.2. 삭제된 기능

현재 릴리스에서 제거된 기능은 지원되지 않습니다.

- 이 릴리스에서는 VM Import Operator가 OpenShift Virtualization에서 제거되었습니다. 이는 Migration Toolkit for Virtualization으로 교체됩니다.
- 이 릴리스에서는 2021년 12월 31일에 EOL(End of Life)에 도달한 CentOS Linux 8용 템플릿이 제거됩니다. 그러나 OpenShift Container Platform에는 CentOS 스트림 8 및 CentOS 스트림 9에 대한 템플릿이 포함되어 있습니다.



참고

모든 CentOS 배포판은 커뮤니티에서 지원됩니다.

3.5. 기술 프리뷰 기능

이 릴리스의 일부 기능은 현재 기술 프리뷰 단계에 있습니다. 이러한 실험적 기능은 프로덕션용이 아닙니다. Red Hat 고객 포털은 다음과 같은 기능에 대한 기술 프리뷰 기능 지원 범위를 제공합니다.

- 이제 Red Hat Enterprise Linux 9 Beta 템플릿을 사용하여 가상 머신을 생성할 수 있습니다.
- 이제 AWS 베어 메탈 노드에 OpenShift Virtualization을 배포할 수 있습니다.
- OpenShift Virtualization 위험 경고에는 즉각적인 주의가 필요한 문제, 각 경고가 발생하는 이유, 문제의 원인을 진단하는 문제 해결 프로세스 및 각 경고 해결 단계에 대한 해당 문제에 대한 설명이 있습니다.
- 클러스터 관리자는 OpenShift Virtualization 플러그인에서 데이터 보호용 OpenShift API를 사용하여 VM이 포함된 네임스페이스를 백업할 수 있습니다.
- 이제 관리자는 **HyperConverged** CR을 편집하여 가상 그래픽 처리장치(vGPU)와 같은 중재 장치를 선언하고 노출할 수 있습니다. 그러면 가상 시스템 소유자가 이러한 장치를 VM에 할당할 수 있습니다.
- 단일 **NodeNetworkConfigurationPolicy** 매니페스트를 클러스터에 적용하여 브릿지에 연결된 NIC의 고정 IP 구성을 전송할 수 있습니다.
- IBM Cloud Bare Metal Server에 OpenShift Virtualization을 설치할수 있습니다. 다른 클라우드 공급자가 제공하는 베어 메탈 서버는 지원되지 않습니다.

3.6. 버그 수정

- 복제 소스를 사용하기 전에 복제 작업을 시작하면 해결 방법을 사용하지 않고 복제 작업이 성공적으로 완료됩니다. (BZ#1855182)
- VM이 버전 4.8 이전 OpenShift Virtualization에서 제공한 삭제된 템플릿을 참조하는 경우 가상 머신 편집에 실패합니다. OpenShift Virtualization 4.8 이상에서 삭제된 OpenShift Virtualization

제공 템플릿은 OpenShift Virtualization Operator에서 자동으로 다시 생성합니다. (BZ#1929165)

- VNC 콘솔에서 가상 머신을 사용할 때 키 전송 및 연결 해제 버튼을 성공적으로 사용할 수 있습니다. (BZ#1964789)
- 가상 머신을 생성할 때 고유한 FQDN(정규화된 도메인 이름)에 클러스터 도메인 이름이 포함됩니다. (BZ#1998300)
- 가상 디스크를 핫플러그한 다음 **virt-launcher** Pod를 강제로 삭제하면 더 이상 데이터가 손실되지 않습니다. (BZ#2007397)
- 파일 시스템을 다른 중요한 구성 요소와 공유하는 경로에 HPP(Hostpath provisioner)를 설치하려고 하면 OpenShift Virtualization에서 HPPSharingPoolPathWithOS 경고를 발행합니다. HPP를 사용하여 가상 머신 디스크에 스토리지를 제공하려면 노드의 루트 파일 시스템과 별도의 전용 스토리지로 구성합니다. 그렇지 않으면 노드가 스토리지 부족하여 작동하지 않을 수 있습니다. (BZ#2038985)
- 가상 머신 디스크를 프로비저닝하는 경우 OpenShift Virtualization은 이제 각 VM 디스크 PVC에 대해 KubePersistentVolumeFillingUp 경고를 발행하는 대신 요청된 디스크 크기를 수용하기에 충분한 PVC(영구 볼륨 클레임)를 할당합니다. 가상 머신 자체에서 디스크 사용량을 모니터링할 수 있습니다. (BZ#2039489)
- 이제 핫플러그 디스크가 있는 VM의 가상 머신 스냅샷을 생성할 수 있습니다. (BZ#2042908)
- 클러스터 전체 프록시 구성을 사용할 때 VM 이미지를 성공적으로 가져올 수 있습니다. (BZ#2046271)

3.7. 확인된 문제

- 단일 스택 IPv6 클러스터에서는 OpenShift Virtualization을 실행할 수 없습니다. (BZ#2193267)
- 다른 SELinux 컨텍스트가 있는 두 개의 Pod를 사용하면 **ocs-storagecluster-cephfs** 스토리지 클래스가 있는 VM이 마이그레이션되지 않고 VM 상태가 일시 중지됨으로 변경됩니다. 두 Pod 모두 공유 **ReadWriteMany** CephFS 볼륨에 동시에 액세스하려고 하기 때문입니다. (BZ#2092271)
  - 이 문제를 해결하려면 **ocs-storagecluster-ceph-rbd** 스토리지 클래스를 사용하여 Red Hat Ceph Storage를 사용하는 클러스터에서 VM을 실시간 마이그레이션합니다.
- OpenShift Virtualization 4.10.5로 업데이트하면 일부 VM(가상 머신)이 실시간 마이그레이션 루프에 고정됩니다. 이는 VM 매니페스트의 **spec.volumes.containerDisk.path** 필드가 상대 경로로 설정된 경우 발생합니다.
  - 이 문제를 해결하려면 VM 매니페스트를 삭제하고 재생성하여 **spec.volumes.containerDisk.path** 필드의 값을 절대 경로로 설정합니다. 그런 다음 OpenShift Virtualization을 업데이트할 수 있습니다.
- 단일 노드에 50개 이상의 이미지가 포함된 경우 노드 간에 Pod 스케줄링이 분배될 수 있습니다. 이는 노드의 이미지 목록이 기본적으로 50으로 단축되기 때문입니다. (BZ#1984442)
  - 이 문제를 해결하려면 KubeletConfig 오브젝트를 편집하고 **nodeStatusMaxImages** 의 값을 **-1** 로 설정하여 이미지 제한을 비활성화할 수 있습니다.
- 노드에 42자를 초과하는 정규화된 도메인 이름(FQDN)이 있는 클러스터에 hostpath 프로비전 프로그램을 배포하는 경우 프로비전 프로그램이 PVC를 바인딩하지 못합니다. (BZ#2057157)

  오류 메시지의 예

```
E0222 17:52:54.088950       1 reflector.go:138] k8s.io/client-
go/informers/factory.go:134: Failed to watch *v1beta1.CSIStorageCapacity: failed to
list *v1beta1.CSIStorageCapacity: unable to parse requirement: values[0]
[csi.storage.k8s.io/managed-by]: Invalid value: "external-provisioner-<node_FQDN>":
must be no more than 63 characters (1)
```

(1) 오류 메시지는 최대 63자까지 참조하지만 여기에는 노드의 FQDN 접두사가 붙은 **external-provisioner-** 문자열이 포함됩니다.

- 해결방법은 다음 명령을 실행하여 hostpath 프로비저너 CSI 드라이버에서 **storageCapacity** 옵션을 비활성화합니다.

```
$ oc patch csidriver kubevirt.io.hostpath-provisioner --type merge --patch
'{"spec": {"storageCapacity": false}}'
```

- OpenShift Container Platform 클러스터에서 OVN-Kubernetes를 기본 CNI(Container Network Interface) 공급자로 사용하는 경우 OVN-Kubernetes의 호스트 네트워크 토폴로지 변경으로 인해 Linux 브리지 또는 본딩 장치를 호스트의 기본 인터페이스에 연결할 수 없습니다. (BZ#1885605)
  - 해결 방법으로 호스트에 연결된 보조 네트워크 인터페이스를 사용하거나 OpenShift SDN 기본 CNI 공급자로 전환할 수 있습니다.
- 실시간으로 마이그레이션할 수 없는 가상 머신을 실행하면 OpenShift Container Platform 클러스터 업그레이드가 차단될 수 있습니다. 여기에는 hostpath 프로비전 프로그램 스토리지 또는 SR-IOV 네트워크 인터페이스를 사용하는 가상 머신이 포함됩니다.
  - 해결 방법으로 클러스터를 업그레이드하는 동안 전원이 꺼지도록 가상 머신을 재구성할 수 있습니다. 가상 머신 구성 파일의 **spec** 섹션에서 다음을 수행합니다.
    1. **evictionStrategy** 및 **runStrategy** 필드를 수정합니다.
       a. **evictionStrategy: LiveMigrate** 필드를 제거합니다. 제거 전략을 구성하는 방법에 대한 자세한 내용은 가상 머신 제거 전략 구성을 참조하십시오.
       b. **runStrategy** 필드를 **Always**로 설정합니다.
    2. 다음 명령을 실행하여 기본 CPU 모델을 설정합니다.

참고

실시간 마이그레이션을 지원하는 가상 머신을 시작하기 전에 이러한 변경을 수행해야 합니다.

```
$ oc annotate --overwrite -n openshift-cnv hyperconverged kubevirt-
hyperconverged kubevirt.kubevirt.io/jsonpatch='[
 {
    "op": "add",
    "path": "/spec/configuration/cpuModel",
    "value": "<cpu_model>" (1)
 }
]'
```

1 <cpu_model>을 실제 CPU 모델 값으로 바꿉니다. 모든 노드에 대해oc describe node <node>를 실행한 후cpu-model-<name> 라벨에서 이 값을 확인할 수 있습니다. 모든 노드에 존재하는 CPU 모델을 선택합니다.

- Red Hat Ceph Storage 또는 Red Hat OpenShift Data Foundation Storage를 사용하는 경우 한 번에 100개 이상의 VM을 복제하지 못할 수 있습니다. (BZ#1989527)
  - 이 문제를 해결하려면 스토리지 프로필 매니페스트에 spec.cloneStrategy: copy 를 설정하여 호스트 지원 사본을 수행할 수 있습니다. 예를 들면 다음과 같습니다.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: StorageProfile
metadata:
  name: <provisioner_class>
#   ...
spec:
  claimPropertySets:
  - accessModes:
    - ReadWriteOnce
    volumeMode: Filesystem
  cloneStrategy: copy (1)
status:
  provisioner: <provisioner>
  storageClass: <provisioner_class>
```

1 기본 복제 방법은 copy.

- 경우에 따라 여러 가상 머신에서 동일한 PVC를 읽기-쓰기 모드로 마운트할 수 있으므로 데이터가 손상될 수 있습니다. (BZ#1992753)
  - 이 문제를 해결하려면 여러 VM에서 읽기-쓰기 모드에서 단일 PVC를 사용하지 마십시오.
- Pod PDB(Disruption Budget)를 사용하면 Pod가 조정 가능한 가상 머신 이미지의 중단을 방지할 수 있습니다. PDB에서 Pod 중단을 탐지하면 openshift-monitoring 은 LiveMigrate 제거 전략을 사용하는 가상 머신 이미지에 대해 60분마다 PodDisruptionBudgetAtLimit 경고를 보냅니다. (BZ#2026733)
  - 해결 방법으로 경고가 지연 됩니다.
- 대규모 클러스터에서 OpenShift Virtualization MAC 풀 관리자는 부팅하는 데 시간이 너무 오래 걸릴 수 있으며 OpenShift Virtualization이 준비되지 않을 수 있습니다. (BZ#2035344)
  - 이 문제를 해결하려면 MAC 풀링 기능이 필요하지 않은 경우 다음 명령을 실행하여 이 하위 구성 요소를 비활성화합니다.

```
$ oc annotate --overwrite -n openshift-cnv hco kubevirt-hyperconverged
'networkaddonsconfigs.kubevirt.io/jsonpatch=[
 {
   "op": "replace"
   "path": "/spec/kubeMacPool"
   "value": null
 }
]'
```

- OpenShift Virtualization은 Pod에서 사용하는 서비스 계정 토큰을 해당 특정 Pod에 연결합니다. OpenShift Virtualization은 토큰이 포함된 디스크 이미지를 생성하여 서비스 계정 볼륨을 구현합니다. VM을 마이그레이션하면 서비스 계정 볼륨이 유효하지 않습니다. (BZ#2037611)
    - 이 문제를 해결하려면 사용자 계정 토큰이 특정 Pod에 바인딩되지 않으므로 서비스 계정이 아닌 사용자 계정을 사용하십시오.
- 종료 중에 VM이 충돌하거나 중단되면 새 종료 요청이 VM을 중지하지 않습니다. (BZ#2040766)
- 드라이버를 설치하기 전에 중재된 장치를 활성화하도록 **HyperConverged** CR(사용자 정의 리소스)을 구성하는 경우 중재 장치 사용이 발생하지 않습니다. 이 문제는 업데이트를 통해 트리거할 수 있습니다. 예를 들어 NVIDIA 드라이버를 설치하는 **daemonset** 앞에 **virt-handler** 가 업데이트되면 노드는 가상 머신 GPU를 제공할 수 없습니다. (BZ#2046298)
    - 해결 방법으로 다음을 수행합니다.
        1. **HyperConverged** CR에서 **mediatedDevicesConfiguration** 및 **permittedHostDevices** 를 제거합니다.
        2. 사용하려는 구성으로 **mediatedDevicesConfiguration** 및 **permittedHostDevices** 스탠자를 모두 업데이트합니다.
- VM 마법사의 YAML 예는 하드 코딩되며 항상 최신 업스트림 변경 사항이 포함되지는 않습니다. (BZ#2055492)
- **csi-clone** 복제 전략을 사용하여 100개 이상의 VM을 복제하면 Ceph CSI가 복제본을 제거하지 못할 수 있습니다. 복제본 수동 삭제도 실패할 수 있습니다. (BZ#2055595)
    - 이 문제를 해결하려면 **ceph-mgr** 을 다시 시작하여 VM 복제를 삭제할 수 있습니다.
- 권한이 없는 사용자는 **VM** 네트워크 인터페이스 탭에서 네트워크 인터페이스 추가 버튼을 사용할 수 없습니다. (BZ#2056420)
    - 이 문제를 해결하려면 권한이 없는 사용자가 VM 마법사를 사용하여 VM을 생성하는 동안 추가 네트워크 인터페이스를 추가할 수 있습니다.
- 권한이 없는 사용자는 RBAC 규칙으로 인해 VM에 디스크를 추가할 수 없습니다. (BZ#2056421)
    - 이 문제를 해결하려면 특정 사용자가 디스크를 추가할 수 있도록 RBAC 규칙을 수동으로 추가합니다.
- 웹 콘솔에는 사용자 정의 네임스페이스에 배포된 가상 머신 템플릿이 표시되지 않습니다. 웹 콘솔의 기본 네임스페이스에 배포된 템플릿만 표시합니다. (BZ#2054650)
    - 이 문제를 해결하려면 사용자 정의 네임스페이스에 템플릿을 배포하지 마십시오.
- SNO(Single Node OpenShift) 클러스터에서 VMI에 **spec.evictionStrategy** 필드가 **LiveMigrate** 로 설정된 경우 클러스터 업데이트가 실패합니다. 실시간 마이그레이션이 성공하려면 클러스터에 작업자 노드가 두 개 이상 있어야 합니다. (BZ#2073880)
    - 두 가지 해결 방법이 있습니다.
        - VM 선언에서 **spec.evictionStrategy** 필드를 제거합니다.
        - OpenShift Container Platform을 업데이트하기 전에 VM을 수동으로 중지합니다.

4장. 설치

4.1. OPENSHIFT VIRTUALIZATION을 위한 클러스터 준비

OpenShift Virtualization을 설치하기 전에 이 섹션을 검토하여 클러스터가 요구 사항을 충족하는지 확인합니다.

중요

사용자 프로비저닝, 설치 관리자 프로비저닝 또는 지원 설치 프로그램을 포함하여 설치 방법을 사용하여 OpenShift Container Platform을 배포할 수 있습니다. 그러나 설치 방법과 클러스터 토폴로지는 스냅샷 또는 실시간 마이그레이션과 같은 OpenShift Virtualization 기능에 영향을 줄 수 있습니다.

단일 노드 OpenShift 차이점

단일 노드 클러스터에 OpenShift Virtualization을 설치할 수 있습니다. 자세한 내용은 단일 노드 OpenShift 정보를 참조하십시오. 단일 노드 OpenShift는 고가용성을 지원하지 않으므로 다음과 같은 차이점이 있습니다.

- Pod 중단 예산은 지원되지 않습니다.
- 실시간 마이그레이션 은 지원되지 않습니다.
- 데이터 볼륨 또는 스토리지 프로필을 사용하는 템플릿 또는 가상 머신에 **evictionStrategy** 가 설정되지 않아야 합니다.

FIPS 모드

FIPS 모드에서 클러스터를 설치하는 경우 OpenShift Virtualization에 추가 설정이 필요하지 않습니다.

IPv6

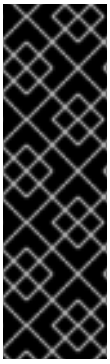
단일 스택 IPv6 클러스터에서는 OpenShift Virtualization을 실행할 수 없습니다. (BZ#2193267)

4.1.1. 하드웨어 및 운영 체제 요구 사항

OpenShift Virtualization에 대한 다음 하드웨어 및 운영 체제 요구 사항을 검토합니다.

지원되는 플랫폼

- 온프레미스 베어 메탈 서버
- Amazon Web Services 베어 메탈 인스턴스. 자세한 내용은 AWS 베어 메탈 노드에 OpenShift Virtualization 배포를 참조하십시오.
- IBM Cloud 베어 메탈 서버. 자세한 내용은 IBM Cloud Bare Metal 노드에 OpenShift Virtualization 배포를 참조하십시오.



중요

AWS 베어 메탈 인스턴스 또는 IBM Cloud Bare Metal Server에 OpenShift Virtualization을 설치하는 것은 기술 프리뷰 기능입니다. 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.

- 다른 클라우드 공급자가 제공하는 베어 메탈 인스턴스 또는 서버는 지원되지 않습니다.

CPU 요구사항

- RHEL (Red Hat Enterprise Linux) 8에서 지원
- Intel 64 또는 AMD64 CPU 확장 지원
- Intel VT 또는 AMD-V 하드웨어 가상화 확장 기능 활성화
- NX(실행 없음) 플래그를 사용할 수 있음

스토리지 요구사항

- OpenShift Container Platform에서 지원

주의

Red Hat OpenShift Data Foundation을 사용하여 OpenShift Virtualization을 배포하는 경우 Windows 가상 머신 디스크용 전용 스토리지 클래스를 생성해야 합니다. 자세한 내용은 Windows VM용 ODF PersistentVolume 최적화를 참조하십시오.

운영 체제 요구 사항

- 작업자 노드에 설치된 RHCOS(Red Hat Enterprise Linux CoreOS)

참고

RHEL 작업자 노드는 지원되지 않습니다.

- 클러스터가 다른 CPU가 있는 작업자 노드를 사용하는 경우 서로 다른 CPU의 기능이 다르기 때문에 실시간 마이그레이션 오류가 발생할 수 있습니다. 이러한 오류를 방지하려면 각 노드에 적절한 용량이 있는 CPU를 사용하고 가상 머신에 노드 유사성을 설정하여 마이그레이션이 성공하도록 합니다. 자세한 내용은 필수 노드 유사성 규칙 구성을 참조하십시오.

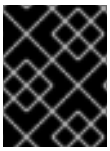
추가 리소스

- RHCOS 정보.

- 지원되는 CPU에 대한 Red Hat Ecosystem Catalog
- 지원되는 스토리지.

4.1.2. 물리적 리소스 오버헤드 요구사항

OpenShift Virtualization은 OpenShift Container Platform의 추가 기능이며 클러스터를 계획할 때 고려해야 하는 추가 오버헤드를 적용합니다. 각 클러스터 머신은 OpenShift Container Platform 요구 사항 이외에도 다음과 같은 오버헤드 요구 사항을 충족해야 합니다. 클러스터에서 물리적 리소스를 초과 구독하면 성능에 영향을 미칠 수 있습니다.



중요

이 문서에 명시된 수치는 Red Hat의 테스트 방법론 및 설정을 기반으로 한 것입니다. 고유한 개별 설정 및 환경에 따라 수치가 달라질 수 있습니다.

4.1.2.1. 메모리 오버헤드

아래 식을 사용하여 OpenShift Virtualization의 메모리 오버헤드 값을 계산합니다.

클러스터 메모리 오버헤드

```
Memory overhead per infrastructure node ≈ 150 MiB
```

```
Memory overhead per worker node ≈ 360 MiB
```

또한, OpenShift Virtualization 환경 리소스에는 모든 인프라 노드에 분산된 총 2179MiB의 RAM이 필요합니다.

가상 머신 메모리 오버헤드

```
Memory overhead per virtual machine ≈ (1.002 * requested memory) + 146 MiB  \
            + 8 MiB * (number of vCPUs) \ 1
          + 16 MiB * (number of graphics devices) 2
```

1 가상 머신에서 요청한 가상 CPU 수

2 가상 머신에서 요청한 가상 그래픽 카드 수

환경에 SR-IOV(Single Root I/O Virtualization) 네트워크 장치 또는 GPU(Graphics Processing Unit)가 포함된 경우 각 장치에 대해 1GiB의 추가 메모리 오버헤드를 할당합니다.

4.1.2.2. CPU 오버헤드

아래 식을 사용하여 OpenShift Virtualization에 대한 클러스터 프로세서 오버헤드 요구 사항을 계산합니다. 가상 머신당 CPU 오버헤드는 개별 설정에 따라 다릅니다.

클러스터 CPU 오버헤드

```
CPU overhead for infrastructure nodes ≈ 4 cores
```

OpenShift Virtualization은 로깅, 라우팅 및 모니터링과 같은 클러스터 수준 서비스의 전반적인 사용률을 높입니다. 이 워크로드를 처리하려면 인프라 구성 요소를 호스팅하는 노드에 4 개의 추가 코어 (4000밀리코어)가 해당 노드에 분산되어 있는지 확인합니다.

> CPU overhead for worker nodes ≈ 2 cores + CPU overhead per virtual machine

가상 머신을 호스팅하는 각 작업자 노드는 가상 머신 워크로드에 필요한 CPU 외에도 OpenShift Virtualization 관리 워크로드에 대한 2개의 추가 코어(2000밀리코어)용 용량이 있어야 합니다.

가상 머신 CPU 오버헤드

전용 CPU가 요청되면 클러스터 CPU 오버헤드 요구 사항에 대한 1:1 영향이 있습니다. 그러지 않으면 가상 머신에 필요한 CPU 수에 대한 구체적인 규칙이 없습니다.

4.1.2.3. 스토리지 오버헤드

아래 지침을 사용하여 OpenShift Virtualization 환경에 대한 스토리지 오버헤드 요구 사항을 추정할 수 있습니다.

클러스터 스토리지 오버헤드

> Aggregated storage overhead per node ≈ 10 GiB

10GiB는 OpenShift Virtualization을 설치할 때 클러스터의 각 노드에 대해 예상되는 온디스크 스토리지 영향입니다.

가상 머신 스토리지 오버헤드

가상 머신당 스토리지 오버헤드는 가상 머신 내의 리소스 할당 요청에 따라 다릅니다. 이 요청은 클러스터의 다른 위치에서 호스팅되는 노드 또는 스토리지 리소스의 임시 스토리지에 대한 요청일 수 있습니다. 현재 OpenShift Virtualization은 실행 중인 컨테이너 자체에 대한 추가 임시 스토리지를 할당하지 않습니다.

4.1.2.4. 예

클러스터 관리자가 클러스터에서 10개의 가상 머신을 호스팅하는 경우 1GiB RAM과 2개의 vCPU가 장착된 메모리의 클러스터 전체에 대한 영향은 11.68GiB입니다. 클러스터의 각 노드에 대한 디스크 스토리지 영향은 10GiB이며 호스트 가상 머신 워크로드가 최소 2개 코어인 작업자 노드에 대한 CPU 영향은 최소 2개입니다.

4.1.3. 오브젝트 최대값

클러스터를 계획할 때 다음과 같은 테스트된 오브젝트 최대값을 고려해야 합니다.

- OpenShift Container Platform 오브젝트 최대.
- OpenShift Virtualization 오브젝트 최대값.

4.1.4. 제한된 네트워크 환경

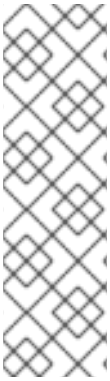
인터넷 연결이 없는 제한된 환경에서 OpenShift Virtualization을 설치하는 경우 제한된 네트워크에 대해 Operator Lifecycle Manager를 구성해야 합니다.

인터넷 연결이 제한된 경우 Red Hat 제공 OperatorHub에 액세스하도록 Operator Lifecycle Manager에서 프록시 지원을 구성 할 수 있습니다.

4.1.5. 실시간 마이그레이션

실시간 마이그레이션에는 다음과 같은 요구 사항이 있습니다.

- RWX(**ReadWriteMany**) 액세스 모드를 사용하여 스토리지 공유.
- 충분한 RAM 및 네트워크 대역폭.
- 가상 머신에서 호스트 모델 CPU를 사용하는 경우 노드는 가상 머신의 호스트 모델 CPU를 지원해야 합니다.



참고

노드 드레이닝을 지원하기 위해 클러스터에 메모리 요청 용량이 충분한지 확인하여 실시간 마이그레이션을 수행해야 합니다. 다음 계산을 사용하여 필요한 예비 메모리를 확인할 수 있습니다.

> Product of (Maximum number of nodes that can drain in parallel) and (Highest total VM memory request allocations across nodes)

클러스터에서 병렬로 실행할 수 있는 기본 마이그레이션 수는5입니다.

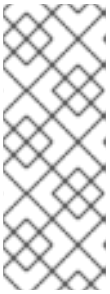
4.1.6. 스냅샷 및 복제

스냅샷 및 복제 요구 사항은 OpenShift Virtualization 스토리지 기능을 참조하십시오.

4.1.7. 클러스터 고가용성 옵션

클러스터에 대해 다음과 같은 HA(고가용성) 옵션 중 하나를 구성할 수 있습니다.

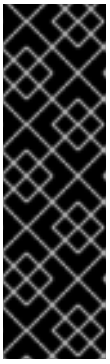
- 머신 상태 점검 을 배포하여 설치 관리자 프로비저닝 인프라 (IPI)의 자동 고가용성을 사용할 수 있습니다.



참고

설치 관리자 프로비저닝 인프라를 사용하여 설치하고 MachineHealthCheck가 올바르게 구성된 OpenShift Container Platform 클러스터에서는 노드가 MachineHealthCheck에 실패하여 클러스터에서 사용할 수 없게 되는 경우 재활용됩니다. 실패한 노드에서 실행된 VM에서 다음에 수행되는 작업은 일련의 조건에 따라 다릅니다. 잠재적 결과 및 RunStrategies가 이러한 결과에 미치는 영향에 대한 자세한 내용은 가상 머신에 대한 RunStrategies 정보를 참조하십시오.

- OpenShift Container Platform 클러스터에서 Node Health Check Operator를 사용하여 IPI 및 비IPI에 대한 자동 고가용성을 사용하여 **NodeHealthCheck** 컨트롤러를 배포할 수 있습니다. 컨트롤러는 비정상적인 노드를 식별하고 Self Node Remediation Operator를 사용하여 비정상 노드를 수정합니다.



중요

Node Health Check Operator는 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.

- 모니터링 시스템 또는 자격을 갖춘 사람이 노드 가용성을 모니터링하는 모든 플랫폼에 대한 고가용성을 사용할 수 있습니다. 노드가 손실되면 노드를 종료하고 **oc delete node <lost_node>**를 실행합니다.

참고

외부 모니터링 시스템 또는 인증된 사용자 모니터링 노드 상태가 없으면 가상 머신의 가용성이 저하됩니다.

4.2. OPENSHIFT VIRTUALIZATION 구성 요소를 위한 노드 지정

노드 배치 규칙을 구성하여 OpenShift Virtualization Operator, 워크로드 및 컨트롤러를 배포할 노드를 지정합니다.

참고

OpenShift Virtualization을 설치한 후에는 일부 구성 요소에 대한 노드 배치를 구성할 수 있지만, 워크로드에 대한 노드 배치를 구성하려면 가상 머신이 없어야 합니다.

4.2.1. 가상화 구성 요소를 위한 노드 배치 정보

다음이 수행되도록 OpenShift Virtualization이 구성 요소를 배포하는 위치를 사용자 지정하는 것이 좋습니다.

- 가상 머신은 가상화 워크로드를 위한 노드에만 배포됩니다.
- Operator는 인프라 노드에만 배포됩니다.
- 특정 노드는 OpenShift Virtualization의 영향을 받지 않습니다. 예를 들어, 클러스터에서 실행되는 가상화와 관련이 없는 워크로드가 있으며 해당 워크로드가 OpenShift Virtualization과 격리되기를 원하는 경우가 이에 해당합니다.

4.2.1.1. 가상화 구성 요소에 노드 배치 규칙을 적용하는 방법

해당 오브젝트를 직접 편집하거나 웹 콘솔을 사용하여 구성 요소의 노드 배치 규칙을 지정할 수 있습니다.

- OLM(Operator Lifecycle Manager)이 배포하는 OpenShift Virtualization Operator의 경우, OLM 서브스크립션 오브젝트를 직접 편집합니다. 현재는 웹 콘솔을 사용하여서브스크립션 오브젝트에 대한 노드 배치 규칙을 구성할 수 없습니다.
- OpenShift Virtualization Operator가 배포하는 구성 요소의 경우, OpenShift Virtualization 설치 중에 웹 콘솔을 사용하여 **HyperConverged** 오브젝트를 직접 편집하거나 구성합니다.

- hostpath 프로비전 프로그램의 경우, **HostPathProvisioner** 오브젝트를 직접 편집하거나 웹 콘솔을 사용하여 이를 구성합니다.

주의

hostpath 프로비전 프로그램과 가상화 구성 요소를 동일한 노드에 예약해야 합니다. 예약하지 않으면 hostpath 프로비전 프로그램을 사용하는 가상화 Pod를 실행할 수 없습니다.

오브젝트에 따라, 다음 규칙 유형 중 하나 이상을 사용할 수 있습니다.

nodeSelector

이 필드에서 지정하는 키-값 쌍으로 라벨이 지정된 노드에 Pod를 예약할 수 있습니다. 노드에는 나열된 모든 쌍과 정확히 일치하는 라벨이 있어야 합니다.

유사성

더 많은 표현 구문을 사용하여 노드와 Pod의 일치 규칙을 설정할 수 있습니다. 유사성을 사용하면 규칙 적용 방법을 보다 자세하게 설정할 수 있습니다. 예를 들어, 규칙을 엄격한 요구 사항이 아닌 기본 설정으로 지정할 수 있으므로 규칙이 충족되지 않은 경우에도 Pod를 예약할 수 있습니다.

허용 오차

일치하는 테인트가 있는 노드에 Pod를 예약할 수 있습니다. 테인트가 노드에 적용되는 경우, 해당 노드는 테인트를 허용하는 Pod만 허용합니다.

4.2.1.2. OLM 서브스크립션 오브젝트에서의 노드 배치

OLM이 OpenShift Virtualization Operator를 배포하는 노드를 지정하려면, OpenShift Virtualization 설치 중에 서브스크립션 오브젝트를 편집합니다. 다음 예와 같이 **spec.config** 필드에 노드 배치 규칙을 추가할 수 있습니다.

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: hco-operatorhub
  namespace: openshift-cnv
spec:
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  name: kubevirt-hyperconverged
  startingCSV: kubevirt-hyperconverged-operator.v4.10.10
  channel: "stable"
  config: 1
```

1 **config** 필드는 **nodeSelector** 및 허용 오차를 지원하지만 유사성은 지원되지 않습니다.

4.2.1.3. HyperConverged 오브젝트에서의 노드 배치

OpenShift Virtualization이 해당 구성 요소를 배포하는 노드를 지정하려면 OpenShift Virtualization을 설치하는 동안 생성한 HyperConverged Cluster 사용자 정의 리소스(CR) 파일에 **nodePlacement** 개체를 포

함할 수 있습니다. 다음 예와 같이 **spec.infra** 및 **spec.workloads** 필드에 **nodePlacement**를 추가할 수 있습니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  infra:
    nodePlacement: 1
    ...
  workloads:
    nodePlacement:
    ...
```

1 The **nodePlacement** 필드는 **nodeSelector**,**affinity** 및 **tolerations** 필드를 지원합니다.

4.2.1.4. HostPathProvisioner 오브젝트에서의 노드 배치

hostpath 프로비전 프로그램을 설치할 때 생성할 **HostPathProvisioner** 오브젝트의 **spec.workload** 필드에 노드 배치 규칙을 구성할 수 있습니다.

```
apiVersion: hostpathprovisioner.kubevirt.io/v1beta1
kind: HostPathProvisioner
metadata:
  name: hostpath-provisioner
spec:
  imagePullPolicy: IfNotPresent
  pathConfig:
    path: "</path/to/backing/directory>"
    useNamingPrefix: false
  workload: 1
```

1 워크로드 필드는 **nodeSelector**, 유사성 및 허용 오차 필드를 지원합니다.

4.2.1.5. 추가 리소스

- 가상 머신용 노드 지정
- 노드 선택기를 사용하여 특정 노드에 Pod 배치
- 노드 유사성 규칙을 사용하여 노드에 Pod 배치 제어
- 노드 테인트를 사용하여 Pod 배치 제어
- CLI를 사용한 OpenShift Virtualization 설치
- 웹 콘솔을 사용한 OpenShift Virtualization 설치
- 가상 머신 로컬 스토리지 구성

4.2.2. 예시 매니페스트

다음 예시 YAML 파일은 **nodePlacement**, **affinity** 및 **tolerations** 오브젝트를 사용하여 OpenShift Virtualization 구성 요소를 위한 노드 배치를 사용자 지정합니다.

4.2.2.1. Operator Lifecycle Manager 서브스크립션 오브젝트

4.2.2.1.1. 예: OLM 서브스크립션 오브젝트에서 nodeSelector를 사용한 노드 배치

이 예에서는 **example.io/example-infra-key = example-infra-value**로 라벨이 지정된 노드에 OLM이 OpenShift Virtualization Operator를 배치하도록 **nodeSelector**를 구성합니다.

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: hco-operatorhub
  namespace: openshift-cnv
spec:
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  name: kubevirt-hyperconverged
  startingCSV: kubevirt-hyperconverged-operator.v4.10.10
  channel: "stable"
  config:
    nodeSelector:
      example.io/example-infra-key: example-infra-value
```

4.2.2.1.2. 예: OLM 서브스크립션 오브젝트에서 허용 오차를 사용한 노드 배치

이 예에서는 OpenShift Virtualization Operator를 배포하기 위해 OLM에 예약된 노드가 **key=virtualization:NoSchedule** 테인트로 라벨이 지정됩니다. 허용 오차가 일치하는 Pod만 이러한 노드에 예약됩니다.

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: hco-operatorhub
  namespace: openshift-cnv
spec:
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  name: kubevirt-hyperconverged
  startingCSV: kubevirt-hyperconverged-operator.v4.10.10
  channel: "stable"
  config:
    tolerations:
    - key: "key"
      operator: "Equal"
      value: "virtualization"
      effect: "NoSchedule"
```

4.2.2.2. HyperConverged 오브젝트

4.2.2.2.1. 예: HyperConverged Cluster CR에서 nodeSelector를 사용한 노드 배치

이 예에서는 인프라 리소스가 **example.io/example-infra-key = example-infra-value**로 라벨이 지정된 노드에 배치되고 워크로드가 **example.io/example-workloads-key = example-workloads-value**로 라벨이 지정된 노드에 배치되도록 **nodeSelector**가 구성됩니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  infra:
    nodePlacement:
      nodeSelector:
        example.io/example-infra-key: example-infra-value
  workloads:
    nodePlacement:
      nodeSelector:
        example.io/example-workloads-key: example-workloads-value
```

4.2.2.2.2. 예: HyperConverged Cluster CR에서 유사성을 사용한 노드 배치

이 예에서는 인프라 리소스가 **example.io/example-infra-key = example-value**로 라벨이 지정된 노드에 배치되고 워크로드가 **example.io/example-workloads-key = example-workloads-value**로 라벨이 지정된 노드에 배치되도록 유사성이 구성됩니다. 워크로드에 9개 이상의 CPU를 사용하는 것이 좋지만, 사용할 수 없는 경우에도 Pod가 예약됩니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  infra:
    nodePlacement:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: example.io/example-infra-key
                operator: In
                values:
                - example-infra-value
  workloads:
    nodePlacement:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: example.io/example-workloads-key
                operator: In
                values:
```

```
        - example-workloads-value
    preferredDuringSchedulingIgnoredDuringExecution:
    - weight: 1
      preference:
        matchExpressions:
        - key: example.io/num-cpus
          operator: Gt
          values:
          - 8
```

4.2.2.2.3. 예: HyperConverged Cluster CR에서 허용 오차를 사용한 노드 배치

이 예에서는 OpenShift Virtualization 구성 요소를 위해 예약된 노드가 **key=virtualization:NoSchedule** 테인트로 라벨이 지정됩니다. 허용 오차가 일치하는 Pod만 이러한 노드에 예약됩니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  workloads:
    nodePlacement:
      tolerations:
      - key: "key"
        operator: "Equal"
        value: "virtualization"
        effect: "NoSchedule"
```

4.2.2.3. HostPathProvisioner 오브젝트

4.2.2.3.1. 예: HostPathProvisioner 오브젝트에서 nodeSelector를 사용한 노드 배치

이 예에서는 라벨이 **example.io/example-workloads-key = example-workloads-value**로 지정된 노드에 워크로드가 배치되도록 **nodeSelector**가 구성됩니다.

```
apiVersion: hostpathprovisioner.kubevirt.io/v1beta1
kind: HostPathProvisioner
metadata:
  name: hostpath-provisioner
spec:
  imagePullPolicy: IfNotPresent
  pathConfig:
    path: "</path/to/backing/directory>"
    useNamingPrefix: false
  workload:
    nodeSelector:
      example.io/example-workloads-key: example-workloads-value
```

4.3. 웹 콘솔을 사용한 OPENSHIFT VIRTUALIZATION 설치

OpenShift Virtualization을 설치하여 OpenShift Container Platform 클러스터에 가상화 기능을 추가합니다.

OpenShift Container Platform 4.10 웹 콘솔 을 사용하여 OpenShift Virtualization Operator를 구독하고 배포할 수 있습니다.

4.3.1. OpenShift Virtualization Operator 설치

OpenShift Virtualization Operator는 OpenShift Container Platform 웹 콘솔을 사용하여 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.10을 설치합니다.
- OpenShift Container Platform 웹 콘솔에 **cluster-admin** 권한이 있는 사용자로 로그인합니다.

프로세스

1. 관리자로 Operator → OperatorHub를 클릭합니다.
2. 키워드로 필터링 필드에 OpenShift Virtualization을 입력합니다.
3. OpenShift Virtualization 타일을 선택합니다.
4. Operator에 대한 정보를 확인하고 Install을 클릭합니다.
5. Operator 설치 페이지에서 다음을 수행합니다.
   a. 사용 가능한 업데이트 채널 옵션 목록에서 stable을 선택합니다. 이렇게 하면 OpenShift Container Platform 버전과 호환되는 OpenShift Virtualization 버전을 설치할 수 있습니다.
   b. 설치된 네임스페이스의 경우 Operator 권장 네임스페이스 옵션이 선택되어 있는지 확인합니다. 그러면 필수 **openshift-cnv** 네임스페이스에 Operator가 설치되고, 해당 네임스페이스가 존재하지 않는 경우 자동으로 생성됩니다.

주의

openshift-cnv 이외의 네임스페이스에 OpenShift Virtualization Operator를 설치하려고 하면 설치가 실패합니다.

   c. 승인 전략의 경우 기본값인 자동을 선택하여 OpenShift Virtualization이 안정적인 업데이트 채널에서 새 버전을 사용할 수 있을 때 자동으로 업데이트되도록 하는 것이 좋습니다.
   수동 승인 전략을 선택할 수 있지만 클러스터의 지원 가능성 및 기능에 미칠 위험이 높기 때문에 이 방법은 권장할 수 없습니다. 이러한 위험을 완전히 이해하고 자동을 사용할 수 없는 경우에만 수동을 선택합니다.

주의

해당 OpenShift Container Platform 버전과 함께 사용할 때만 OpenShift Virtualization을 지원하므로 누락된 OpenShift Virtualization 업데이트가 없으면 클러스터가 지원되지 않을 수 있습니다.

6. **openshift-cnv** 네임스페이스에서 Operator를 사용할 수 있도록설치를 클릭합니다.
7. Operator가 설치되면 HyperConverged 생성을 클릭합니다.
8. 선택 사항: OpenShift Virtualization 구성 요소에 대한 Infra 및 워크로드 노드 배치 옵션을 구성합니다.
9. 생성을 클릭하여 OpenShift Virtualization을 시작합니다.

검증

- 워크로드 → Pods 페이지로 이동하여 모두실행 중 상태가 될 때까지 OpenShift Virtualization Pod를 모니터링합니다. 모든 Pod에 실행 중 상태가 표시되면 OpenShift Virtualization을 사용할 수 있습니다.

4.3.2. 다음 단계

다음 구성 요소를 추가로 구성하는 것이 좋습니다.

- hostpath 프로비전 프로그램은 OpenShift Virtualization용으로 설계된 로컬 스토리지 프로비전 프로그램입니다. 가상 머신의 로컬 스토리지를 구성하려면 먼저 hostpath 프로비전 프로그램을 활성화해야 합니다.

4.4. CLI를 사용한 OPENSHIFT VIRTUALIZATION 설치

OpenShift Virtualization을 설치하여 OpenShift Container Platform 클러스터에 가상화 기능을 추가합니다. 명령줄을 사용하여 OpenShift Virtualization Operator를 구독하고 배포하여 클러스터에 매니페스트를 적용할 수 있습니다.

참고

OpenShift Virtualization에서 구성 요소를 설치할 노드를 지정하려면 노드 배치 규칙을 구성합니다.

4.4.1. 사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.10을 설치합니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 로그인합니다.

4.4.2. CLI를 사용하여 OpenShift Virtualization 카탈로그 구독

OpenShift Virtualization을 설치하기 전에 OpenShift Virtualization 카탈로그를 구독해야 합니다. 구독하면 **openshift-cnv** 네임스페이스에서 OpenShift Virtualization Operator에 액세스할 수 있습니다.

구독하려면 클러스터에 단일 매니페스트를 적용하여 **Namespace**, **OperatorGroup**, **Subscription** 오브젝트를 구성합니다.

절차

1. 다음 매니페스트를 포함하는 YAML 파일을 만듭니다.

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-cnv
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: kubevirt-hyperconverged-group
  namespace: openshift-cnv
spec:
  targetNamespaces:
    - openshift-cnv
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: hco-operatorhub
  namespace: openshift-cnv
spec:
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  name: kubevirt-hyperconverged
  startingCSV: kubevirt-hyperconverged-operator.v4.10.10
  channel: "stable" (1)
```

(1) **stable** 채널을 사용하면 OpenShift Container Platform 버전과 호환되는 OpenShift Virtualization 버전을 설치할 수 있습니다.

2. 다음 명령을 실행하여 OpenShift Virtualization에 필요한 **Namespace**, **OperatorGroup** 및 **Subscription** 오브젝트를 생성합니다.

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

참고

YAML 파일에서 인증서 교체 매개 변수를 구성할 수 있습니다.

4.4.3. CLI를 사용하여 OpenShift Virtualization Operator 배포

oc CLI를 사용하여 OpenShift Virtualization Operator를 배포할 수 있습니다.

사전 요구 사항

- **openshift-cnv** 네임스페이스의 OpenShift Virtualization 카탈로그에 대한 구독이 활성 상태여야 합니다.

절차

1. 다음 매니페스트를 포함하는 YAML 파일을 만듭니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
```

2. 다음 명령을 실행하여 OpenShift Virtualization Operator를 배포합니다.

```
$ oc apply -f <file_name>.yaml
```

검증

- **openshift-cnv** 네임스페이스에서 CSV(클러스터 서비스 버전)의 **PHASE**를 확인하여 OpenShift Virtualization이 성공적으로 배포되었는지 확인합니다. 다음 명령을 실행합니다.

```
$ watch oc get csv -n openshift-cnv
```

배포에 성공하면 다음 출력이 표시됩니다.

출력 예

```
NAME                                      DISPLAY                    VERSION   REPLACES   PHASE
kubevirt-hyperconverged-operator.v4.10.10   OpenShift Virtualization   4.10.10
Succeeded
```

4.4.4. 다음 단계

다음 구성 요소를 추가로 구성하는 것이 좋습니다.

- hostpath 프로비전 프로그램은 OpenShift Virtualization용으로 설계된 로컬 스토리지 프로비전 프로그램입니다. 가상 머신의 로컬 스토리지를 구성하려면 먼저 hostpath 프로비전 프로그램을 활성화해야 합니다.

4.5. VIRTCTL 클라이언트 활성화

virtctl 클라이언트는 OpenShift Virtualization 리소스를 관리하는 명령줄 유틸리티입니다. Linux, macOS, Windows 배포에 사용할 수 있습니다.

4.5.1. virtctl 클라이언트 다운로드 및 설치

4.5.1.1. virtctl 클라이언트 다운로드

ConsoleCLIDownload CR(사용자 정의 리소스)에 있는 링크를 사용하여 **virtctl** 클라이언트를 다운로드합니다.

절차

1. 다음 명령을 실행하여 **ConsoleCLIDownload** 오브젝트를 확인합니다.

```
$ oc get ConsoleCLIDownload virtctl-clidownloads-kubevirt-hyperconverged -o yaml
```

2. 배포에 나열된 링크를 사용하여 **virtctl** 클라이언트를 다운로드합니다.

4.5.1.2. virtctl 클라이언트 설치

운영 체제의 적절한 위치에서 다운로드한 후 **virtctl** 클라이언트를 추출하고 설치합니다.

사전 요구 사항

- **virtctl** 클라이언트를 다운로드해야 합니다.

절차

- Linux의 경우:
  1. tarball을 추출합니다. 다음 CLI 명령은 tarball과 동일한 디렉터리에 압축을 풉니다.

```
$ tar -xvf <virtctl-version-distribution.arch>.tar.gz
```

  2. 추출된 폴더 계층 구조로 이동하여 다음 명령을 실행하여 **virtctl** 바이너리를 실행할 수 있도록 합니다.

```
$ chmod +x <virtctl-file-name>
```

  3. **virtctl** 바이너리를 **PATH** 환경 변수의 디렉터리로 이동합니다.
  4. 경로를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

- Windows 사용자의 경우:
  1. 아카이브의 압축을 해제하고 압축을 풉니다.
  2. 추출된 폴더 계층 구조로 이동하고 **virtctl** 실행 파일을 두 번 클릭하여 클라이언트를 설치합니다.
  3. **virtctl** 바이너리를 **PATH** 환경 변수의 디렉터리로 이동합니다.
  4. 경로를 확인하려면 다음 명령을 실행합니다.

```
C:\> path
```

- macOS 사용자의 경우:
  1. 아카이브의 압축을 해제하고 압축을 풉니다.
  2. **virtctl** 바이너리를 **PATH** 환경 변수의 디렉터리로 이동합니다.
  3. 경로를 확인하려면 다음 명령을 실행합니다.

```
echo $PATH
```

4.5.2. 추가 설정 옵션

4.5.2.1. yum 유틸리티를 사용하여 virtctl 클라이언트 설치

kubevirt-virtctl 패키지에서 **virtctl** 클라이언트를 설치합니다.

절차

- **kubevirt-virtctl** 패키지를 설치합니다.

```
# yum install kubevirt-virtctl
```

4.5.2.2. OpenShift Virtualization 리포지토리 활성화

Red Hat은 Red Hat Enterprise Linux 8 및 Red Hat Enterprise Linux 7 모두에 OpenShift Virtualization 리포지토리를 제공합니다.

- Red Hat Enterprise Linux 8 리포지토리: **cnv-4.10-for-rhel-8-x86_64-rpms**
- Red Hat Enterprise Linux 7 리포지토리: **rhel-7-server-cnv-4.10-rpms**

subscription-manager에서 리포지토리를 활성화하는 프로세스는 두 플랫폼에서 동일합니다.

절차

- 다음 명령을 실행하여 시스템에 적합한 OpenShift Virtualization 리포지토리를 활성화하십시오.

```
# subscription-manager repos --enable <repository>
```

4.5.3. 추가 리소스

- OpenShift Virtualization에 CLI 툴 사용하기

4.6. 웹 콘솔을 사용하여 OPENSHIFT VIRTUALIZATION 설치 제거

OpenShift Container Platform 웹 콘솔을 사용하여 OpenShift Virtualization을 설치 제거할 수 있습니다.

4.6.1. 사전 요구 사항

- OpenShift Virtualization 4.10이 설치되어 있어야 합니다.
- 가상 머신, 가상 머신 인스턴스, 데이터 볼륨을 모두 삭제해야 합니다.

중요

이러한 오브젝트를 삭제하지 않고 OpenShift Virtualization을 제거하려고 하면 오류가 발생합니다.

4.6.2. OpenShift Virtualization Operator 배포 사용자 정의 리소스 삭제

OpenShift Virtualization을 설치 제거하려면 먼저 OpenShift Virtualization Operator 배포 사용자 정의 리소스를 삭제해야 합니다.

사전 요구 사항

- OpenShift Virtualization Operator 배포 사용자 정의 리소스를 만듭니다.

절차

1. OpenShift Container Platform 웹 콘솔의 프로젝트 목록에서 **openshift-cnv**를 선택합니다.
2. Operator → 설치된 Operator 페이지로 이동합니다.
3. OpenShift Virtualization을 클릭합니다.
4. OpenShift Virtualization Operator 배포 탭을 클릭합니다.
5. kubevirt-hyperconverged 사용자 정의 리소스가 포함된 행에서 옵션 메뉴  를 클릭합니다. 확장된 메뉴에서 HyperConverged 클러스터 삭제를 클릭합니다.
6. 확인 창에서 삭제를 클릭합니다.
7. 워크로드 → Pods 페이지로 이동하여 Operator Pod만 실행 중인지 확인합니다.
8. 터미널 창을 열고 다음 명령을 실행하여 나머지 리소스를 정리합니다.

```
$ oc delete apiservices v1alpha3.subresources.kubevirt.io -n openshift-cnv
```

4.6.3. OpenShift Virtualization 카탈로그 구독 삭제

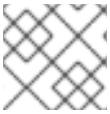
OpenShift Virtualization 제거를 완료하려면 OpenShift Virtualization 카탈로그 구독을 삭제하십시오.

사전 요구 사항

- 활성 상태의 OpenShift Virtualization 카탈로그 구독

절차

1. Operator → OperatorHub 페이지로 이동합니다.
2. OpenShift Virtualization을 검색한 후 선택합니다.
3. 제거를 클릭합니다.



참고

이제 **openshift-cnv** 네임스페이스를 삭제할 수 있습니다.

4.6.4. 웹 콘솔을 사용하여 네임스페이스 삭제

OpenShift Container Platform 웹 콘솔을 사용하여 네임스페이스를 삭제할 수 있습니다.

참고

네임스페이스 삭제 옵션을 사용하려면 네임스페이스를 삭제할 수 있는 권한이 있어야 합니다.

절차

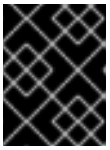
1. 관리 → 네임스페이스로 이동합니다.
2. 네임스페이스 목록에서 삭제하려는 네임스페이스를 찾습니다.
3. 네임스페이스 목록 맨 오른쪽에 있는 옵션 메뉴 에서 네임스페이스 삭제를 선택합니다.
4. 네임스페이스 삭제 창이 열리면 삭제할 네임스페이스 이름을 필드에 입력합니다.
5. 삭제를 클릭합니다.

4.7. CLI를 사용하여 OPENSHIFT VIRTUALIZATION 설치 제거

OpenShift Container Platform CLI를 사용하여 OpenShift Virtualization의 설치를 제거할 수 있습니다.

4.7.1. 사전 요구 사항

- OpenShift Virtualization 4.10이 설치되어 있어야 합니다.
- 가상 머신, 가상 머신 인스턴스, 데이터 볼륨을 모두 삭제해야 합니다.



중요

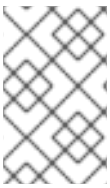
이러한 오브젝트를 삭제하지 않고 OpenShift Virtualization을 제거하려고 하면 오류가 발생합니다.

4.7.2. OpenShift Virtualization 삭제

CLI를 사용하여 OpenShift Virtualization을 삭제할 수 있습니다.

사전 요구 사항

- OpenShift CLI(oc)를 설치합니다.
- **cluster-admin** 권한이 있는 계정을 사용하여 OpenShift Virtualization 클러스터에 액세스할 수 있어야 합니다.



참고

CLI를 사용하여 OLM에서 OpenShift Virtualization Operator 구독을 삭제하면 CSV(**ClusterServiceVersion**) 오브젝트가 클러스터에서 삭제되지 않습니다. OpenShift Virtualization을 완전히 설치 제거하려면 CSV를 명시적으로 삭제해야 합니다.

절차

1. **HyperConverged** 사용자 정의 리소스를 삭제합니다.
   -

```
$ oc delete HyperConverged kubevirt-hyperconverged -n openshift-cnv
```

2. OLM(Operator Lifecycle Manager)에서 OpenShift Virtualization Operator 구독을 삭제합니다.

```
$ oc delete subscription kubevirt-hyperconverged -n openshift-cnv
```

3. OpenShift Virtualization의 CSV(클러스터 서비스 버전) 이름을 환경 변수로 설정합니다.

```
$ CSV_NAME=$(oc get csv -n openshift-cnv -o=jsonpath="{.items[0].metadata.name}")
```

4. 이전 단계에서 CSV 이름을 지정하여 OpenShift Virtualization 클러스터에서 CSV를 삭제합니다.

```
$ oc delete csv ${CSV_NAME} -n openshift-cnv
```

CSV가 성공적으로 삭제되었다는 확인 메시지가 표시되면 OpenShift Virtualization이 설치 제거됩니다.

출력 예

```
clusterserviceversion.operators.coreos.com "kubevirt-hyperconverged-
operator.v4.10.10" deleted
```

5장. OPENSHIFT VIRTUALIZATION 업데이트

OLM(Operator Lifecycle Manager)에서 OpenShift Virtualization에 z-stream 및 마이너 버전 업데이트를 제공하는 방법을 알아봅니다.

5.1. OPENSHIFT VIRTUALIZATION 업데이트 정보

- OLM(Operator Lifecycle Manager)은 OpenShift Virtualization Operator의 라이프사이클을 관리합니다. OpenShift Container Platform 설치 중에 배포되는 Marketplace Operator는 클러스터에서 외부 Operator를 사용할 수 있도록 합니다.
- OLM은 OpenShift Virtualization에 z-stream 및 마이너 버전 업데이트를 제공합니다. OpenShift Container Platform을 다음 마이너 버전으로 업데이트하면 마이너 버전 업데이트를 사용할 수 있습니다. OpenShift Container Platform을 먼저 업데이트하지 않고 OpenShift Virtualization을 다음 마이너 버전으로 업데이트할 수 없습니다.
- OpenShift Virtualization 서브스크립션은 stable 이라는 단일 업데이트 채널을 사용합니다. stable 채널을 사용하면 OpenShift Virtualization 및 OpenShift Container Platform 버전이 호환됩니다.
- 서브스크립션의 승인 전략이 자동으로 설정된 경우 stable 채널에서 새 버전의 Operator를 사용할 수 있는 즉시 업데이트 프로세스가 시작됩니다. 자동 승인 전략을 사용하여 지원 가능한 환경을 유지하는 것이 좋습니다. OpenShift Virtualization의 각 부 버전은 해당 OpenShift Container Platform 버전을 실행하는 경우에만 지원됩니다. 예를 들어 OpenShift Container Platform 4.10에서 OpenShift Virtualization 4.10을 실행해야 합니다.
    - 수동 승인 전략을 선택할 수 있지만 클러스터의 지원 가능성과 기능에 미칠 위험이 높기 때문에 이 방법은 권장되지 않는 것이 좋습니다. 수동 승인 전략을 사용하면 보류 중인 모든 업데이트를 수동으로 승인해야 합니다. OpenShift Container Platform 및 OpenShift Virtualization 업데이트가 동기화되지 않으면 클러스터가 지원되지 않습니다.
- 업데이트를 완료하는 데 걸리는 시간은 네트워크 연결에 따라 달라집니다. 대부분의 자동 업데이트는 15분 이내에 완료됩니다.
- OpenShift Virtualization 업데이트에서는 네트워크 연결이 중단되지 않습니다.
- 데이터 볼륨 및 관련 영구 볼륨 클레임은 업데이트 중에 유지됩니다.

중요

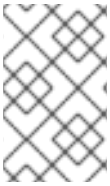
hostpath 프로비전 프로그램을 사용하는 가상 머신이 실행 중인 경우 실시간 마이그레이션할 수 없으며 OpenShift Container Platform 클러스터 업데이트를 차단할 수 있습니다.

해결 방법으로 클러스터 업데이트 중에 전원이 자동으로 꺼지도록 가상 머신을 재구성할 수 있습니다. **evictionStrategy: LiveMigrate** 필드를 제거하고 **runStrategy** 필드를 **Always**로 설정합니다.

5.2. 자동 워크로드 업데이트 구성

5.2.1. 워크로드 업데이트 정보

OpenShift Virtualization, **libvirt**,**virt-launcher**, **qemu** 를 포함한 가상 머신 워크로드를 업데이트할 때 실시간 마이그레이션을 지원하는 경우 자동으로 업데이트합니다.



참고

각 가상 머신에는 VMI(가상 머신 인스턴스)를 실행하는 **virt-launcher** Pod가 있습니다. **virt-launcher** Pod는 가상 머신(VM) 프로세스를 관리하는 데 사용되는**libvirt** 의 인스턴스를 실행합니다.

HyperConverged CR (사용자 정의 리소스)의 **spec.workloadUpdateStrategy** 스탠자를 편집하여 워크로드가 업데이트되는 방법을 구성할 수 있습니다. 사용 가능한 워크로드 업데이트 방법은 **LiveMigrate** 및 **Evict** 입니다.

Evict 메서드는 VMI Pod를 종료하므로 기본적으로**LiveMigrate** 업데이트 전략만 활성화됩니다.

LiveMigrate 가 유일한 업데이트 전략이 활성화된 경우:

- 실시간 마이그레이션을 지원하는 VMI가 업데이트 프로세스 중에 마이그레이션됩니다. VM 게스트는 업데이트된 구성 요소가 활성화된 새 Pod로 이동합니다.
- 실시간 마이그레이션을 지원하지 않는 VMI는 중단되거나 업데이트되지 않습니다.
    - VMI에 **LiveMigrate** 제거 전략이 있지만 실시간 마이그레이션을 지원하지 않는 경우 업데이트되지 않습니다.

LiveMigrate 및 **Evict** 를 모두 활성화하는 경우:

- 실시간 마이그레이션을 지원하는 VMI는 **LiveMigrate** 업데이트 전략을 사용합니다.
- 실시간 마이그레이션을 지원하지 않는 VMI는 **Evict** 업데이트 전략을 사용합니다. VMI가항상 **runStrategy** 값이 있는 **VirtualMachine** 오브젝트에 의해 제어되면 업데이트된 구성 요소가 있는 새 VMI가 새 Pod에 생성됩니다.

마이그레이션 시도 및 타임아웃
워크로드를 업데이트할 때 다음 기간 동안 Pod가 **Pending** 상태인 경우 실시간 마이그레이션이 실패합니다.

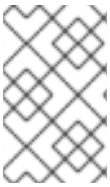
5분

Pod가 **Unschedulable** 이기 때문에 보류 중인 경우 .

15분

어떤 이유로든 Pod가 보류 상태에 있는 경우.

VMI를 마이그레이션하지 못하면 **virt-controller** 가 이를 다시 마이그레이션하려고 합니다. 새**virt-launcher** Pod에서 체결 가능한 모든 VMI가 실행될 때까지 이 프로세스를 반복합니다. VMI가 부적절하게 구성된 경우 이러한 시도에서는 무기한 반복할 수 있습니다.



참고

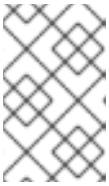
각 시도는 마이그레이션 오브젝트에 해당합니다. 가장 최근 5개의 시도만 버퍼에 저장됩니다. 이렇게 하면 디버깅 정보를 유지하면서 마이그레이션 오브젝트가 시스템에서 누적되지 않습니다.

5.2.2. 워크로드 업데이트 방법 구성

HyperConverged CR(사용자 정의 리소스)을 편집하여 워크로드 업데이트 방법을 구성할 수 있습니다.

사전 요구 사항

- 실시간 마이그레이션을 업데이트 방법으로 사용하려면 먼저 클러스터에서 실시간 마이그레이션을 활성화해야 합니다.



참고

VirtualMachineInstance CR에 **evictionStrategy: LiveMigrate** 가 포함되어 있고 VMI(가상 머신 인스턴스)가 실시간 마이그레이션을 지원하지 않으면 VMI가 업데이트되지 않습니다.

절차

1. 기본 편집기에서 **HyperConverged** CR을 열려면 다음 명령을 실행합니다.

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

2. **HyperConverged** CR의 **workloadUpdateStrategy** 스탠자를 편집합니다. 예를 들면 다음과 같습니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
spec:
  workloadUpdateStrategy:
    workloadUpdateMethods: 1
    - LiveMigrate 2
    - Evict 3
    batchEvictionSize: 10 4
    batchEvictionInterval: "1m0s" 5
...
```

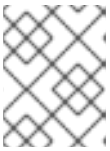
1. 자동화된 워크로드 업데이트를 수행하는 데 사용할 수 있는 방법입니다. 사용 가능한 값은 **LiveMigrate** 및 **Evict** 입니다. 이 예에 표시된 대로 두 옵션을 모두 활성화하면 업데이트에서 실시간 마이그레이션을 지원하지 않는 VMI에 실시간 마이그레이션 및 **Evict**를 지원하는 VMI에 **LiveMigrate**를 사용합니다. 자동 워크로드 업데이트를 비활성화하려면 **workloadUpdateStrategy** 스탠자를 제거하거나 **workloadUpdateMethods: []** 를 설정하여 배열을 비워 둘 수 있습니다.

2. 중단이 적은 업데이트 방법입니다. VMI(가상 머신) 게스트를 업데이트된 구성 요소가 활성화된 새 Pod로 마이그레이션하여 실시간 마이그레이션을 지원하는 VMI가 업데이트됩니다. **LiveMigrate**가 나열된 유일한 워크로드 업데이트 방법인 경우 실시간 마이그레이션을 지원하지 않는 VMI는 중단되거나 업데이트되지 않습니다.

3. 업그레이드 중 VMI Pod를 종료하는 중단 방법입니다. **Evict**는 클러스터에서 실시간 마이그레이션이 활성화되지 않은 경우 사용 가능한 유일한 업데이트 방법입니다. **runStrategy: always** 구성된 **VirtualMachine** 오브젝트에서 VMI를 제어하는 경우 업데이트된 구성 요소가 있는 새 VMI가 새 Pod에 생성됩니다.

4. **Evict** 방법을 사용하여 한 번에 업데이트해야 할 VMI의 수입니다. 이는**LiveMigrate** 방법에는 적용되지 않습니다.

5. 다음 워크로드 배치를 제거하기 전에 대기하는 간격입니다. 이는 **LiveMigrate** 방법에는 적용되지 않습니다.



참고

HyperConverged CR의 **spec.liveMigrationConfig** 스탠자를 편집하여 실시간 마이그레이션 제한 및 타임아웃을 구성할 수 있습니다.

3. 변경 사항을 적용하려면 편집기를 저장하고 종료합니다.

5.3. 보류 중인 OPERATOR 업데이트 승인

5.3.1. 보류 중인 Operator 업데이트 수동 승인

설치된 Operator의 서브스크립션에 있는 승인 전략이 수동으로 설정된 경우 새 업데이트가 현재 업데이트 채널에 릴리스될 때 업데이트를 수동으로 승인해야 설치가 시작됩니다.

사전 요구 사항

- OLM(Operator Lifecycle Manager)을 사용하여 이전에 설치한 Operator입니다.

절차

1. OpenShift Container Platform 웹 콘솔의 관리자 관점에서 Operator → 설치된 Operator로 이동합니다.
2. 보류 중인 업데이트가 있는 Operator에 업그레이드 사용 가능 상태가 표시됩니다. 업데이트할 Operator 이름을 클릭합니다.
3. 서브스크립션 탭을 클릭합니다. 승인이 필요한 업데이트는업그레이드 상태 옆에 표시됩니다. 예를 들어 1 승인 필요가 표시될 수 있습니다.
4. 1 승인 필요를 클릭한 다음 설치 계획 프리뷰를 클릭합니다.
5. 업데이트에 사용할 수 있는 것으로 나열된 리소스를 검토합니다. 문제가 없는 경우 승인을 클릭합니다.
6. Operator → 설치된 Operator 페이지로 다시 이동하여 업데이트 진행 상황을 모니터링합니다. 완료되면 상태가 성공 및 최신으로 변경됩니다.

5.4. 업데이트 상태 모니터링

5.4.1. OpenShift Virtualization 업그레이드 상태 모니터링

OpenShift Virtualization Operator 업그레이드 상태를 모니터링하려면 CSV(클러스터 서비스 버전) **PHASE**를 확인합니다. 웹 콘솔에서 또는 여기에 제공된 명령을 실행하여 CSV 조건을 모니터링할 수도 있습니다.

참고

PHASE 및 조건 값은 사용 가능한 정보를 기반으로 한 근사치입니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 로그인합니다.

- OpenShift CLI(oc)를 설치합니다.

절차

1. 다음 명령을 실행합니다.

```
$ oc get csv -n openshift-cnv
```

2. **PHASE** 필드를 확인하여 출력을 검토합니다. 예를 들면 다음과 같습니다.

출력 예

```
VERSION  REPLACES                                   PHASE
4.9.0    kubevirt-hyperconverged-operator.v4.8.2    Installing
4.9.0    kubevirt-hyperconverged-operator.v4.9.0    Replacing
```

3. 선택 사항: 다음 명령을 실행하여 모든 OpenShift Virtualization 구성 요소 조건을 집계한 상태를 모니터링합니다.

```
$ oc get hco -n openshift-cnv kubevirt-hyperconverged \
-o=jsonpath='{range .status.conditions[*]}{.type}{"\t"}{.status}{"\t"}{.message}{"\n"}
{end}'
```

업그레이드가 완료되면 다음과 같은 결과가 나타납니다.

출력 예

```
ReconcileComplete  True  Reconcile completed successfully
Available          True  Reconcile completed successfully
Progressing        False Reconcile completed successfully
Degraded           False Reconcile completed successfully
Upgradeable        True  Reconcile completed successfully
```

5.4.2. 오래된 OpenShift Virtualization 워크로드 보기

CLI를 사용하여 오래된 워크로드 목록을 볼 수 있습니다.

참고

클러스터에 오래된 가상화 Pod가 있는 경우 **OutdatedVirtualMachineInstanceWorkloads** 경고가 실행됩니다.

절차

- 오래된 VMI(가상 머신 인스턴스) 목록을 보려면 다음 명령을 실행합니다.

```
$ oc get vmi -l kubevirt.io/outdatedLauncherImage --all-namespaces
```

참고

VMI가 자동으로 업데이트되도록 워크로드 업데이트를 구성합니다.

5.5. 추가 리소스

- Operator란 무엇인가?
- Operator Lifecycle Manager 개념 및 리소스
- CSV(클러스터 서비스 버전)
- 가상 머신 실시간 마이그레이션
- 가상 머신 제거 전략 구성
- 실시간 마이그레이션 제한 및 타임아웃 구성

6장. KUBEVIRT-CONTROLLER 및 VIRT-LAUNCHER에 부여된 추가 보안 권한

kubevirt-controller 및 virt-launcher Pod에는 일반적인 Pod 소유자 외에 일부 SELinux 정책 및 보안 컨텍스트 제약 조건 권한이 부여됩니다. 가상 머신은 이러한 권한을 통해 OpenShift Virtualization 기능을 사용할 수 있습니다.

6.1. VIRT-LAUNCHER POD에 대해 확장된 SELINUX 정책

virt-launcher Pod에 대한 **container_t** SELinux 정책은 OpenShift Virtualization의 필수 기능을 사용하도록 확장됩니다.

- 네트워크 멀티 큐에는 다음 정책이 필요하므로 사용 가능한 vCPU 수가 증가함에 따라 네트워크 성능을 확장할 수 있습니다.
  - **allow process self (tun_socket (relabelfrom relabelto attach_queue))**
- 다음 정책을 통해 **virt-launcher** 는 **/proc/cpuinfo** 및 **/proc / uptime** 을 포함하여 /proc 디렉토리 아래에 있는 파일을 읽을 수 있습니다.
  - **allow process proc_type (file (getattr open read))**
- 다음 정책을 사용하면 **libvirtd** 가 네트워크 관련 디버그 메시지를 중계할 수 있습니다.
  - **allow process self (netlink_audit_socket (nlmsg_relay))**

참고

이 정책이 없으면 네트워크 디버그 메시지를 릴레이하는 시도가 차단됩니다. 이렇게 하면 노드의 감사 로그를 SELinux 거부로 채울 수 있습니다.

- 다음 정책을 사용하면 **libvirtd** 가 대규모 페이지를 지원하는 데 필요한**hugetblfs** 에 액세스할 수 있습니다.
  - **allow process hugetlbfs_t (dir (add_name create write remove_name rmdir setattr))**
  - **allow process hugetlbfs_t (file (create unlink))**
- 다음 정책을 통해 **virtiofs** 가 파일 시스템을 마운트하고 NFS에 액세스할 수 있습니다.
  - **allow process nfs_t(dir(mounton))**
  - **allow process proc_t (dir) (mounton)**
  - **allow process proc_t (filesystem (mount unmount))**

6.2. KUBEVIRT-CONTROLLER 서비스 계정에 대한 추가 OPENSHIFT CONTAINER PLATFORM 보안 컨텍스트 제약 조건 및 LINUX 기능

SCC(보안 컨텍스트 제약 조건)는 Pod에 대한 권한을 제어합니다. 이러한 권한에는 컨테이너 모음인 Pod에서 수행할 수 있는 작업과 액세스할 수 있는 리소스가 포함됩니다. Pod가 시스템에 수용되려면 일련의 조건을 함께 실행해야 하는데, SCC를 사용하여 이러한 조건을 정의할 수 있습니다.

kubevirt-controller는 클러스터의 가상 머신에 대해 virt-launcher Pod를 생성하는 클러스터 컨트롤러입니다. 이러한 virt-launcher Pod에는 **kubevirt-controller** 서비스 계정에서 권한을 부여합니다.

6.2.1. kubevirt-controller 서비스 계정에 부여된 추가 SCC

kubevirt-controller 서비스 계정에는 적절한 권한으로 virt-launcher Pod를 생성할 수 있도록 추가 SCC 및 Linux 기능이 부여됩니다. 가상 머신은 이러한 확장된 권한을 통해 일반적인 Pod의 범위를 벗어나는 OpenShift Virtualization 기능을 활용할 수 있습니다.

kubevirt-controller 서비스 계정에는 다음 SCC가 부여됩니다.

- **scc.AllowHostDirVolumePlugin = true**
  이를 통해 가상 머신에서 hostpath 볼륨 플러그인을 사용할 수 있습니다.
- **scc.AllowPrivilegedContainer = false**
  virt-launcher Pod가 권한 있는 컨테이너로 실행되지 않습니다.
- **scc.AllowedCapabilities = []corev1.Capability{"NET_ADMIN", "NET_RAW", "SYS_NICE"}**
  추가 Linux 기능인 **NET_ADMIN**, **NET_RAW**, **SYS_NICE**를 제공합니다.

6.2.2. kubevirt-controller에 대한 SCC 및 RBAC 정의 보기

oc 툴을 사용하여 **kubevirt-controller**에 대한 **SecurityContextConstraints** 정의를 볼 수 있습니다.

```
$ oc get scc kubevirt-controller -o yaml
```

oc 툴을 사용하여 **kubevirt-controller** clusterrole에 대한 RBAC 정의를 볼 수 있습니다.

```
$ oc get clusterrole kubevirt-controller -o yaml
```

6.3. 추가 리소스

- 보안 컨텍스트 제약 조건 관리
- RBAC를 사용하여 권한 정의 및 적용
- RHEL(Red Hat Enterprise Linux) 문서에서 가상 머신 네트워크 성능 최적화
- 가상 머신에서 대규모 페이지 사용
- RHEL 문서에서 대규모 페이지 구성

7장. CLI 툴 사용

다음은 클러스터에서 리소스를 관리하는 데 사용되는 두 가지 기본 CLI 툴입니다.

- OpenShift Virtualization **virtctl** 클라이언트
- OpenShift Container Platform **oc** 클라이언트

7.1. 사전 요구 사항

- **virtctl** 클라이언트를 활성화해야 합니다.

7.2. OPENSHIFT CONTAINER PLATFORM 클라이언트 명령

OpenShift Container Platform **oc** 클라이언트는 **VirtualMachine(vm)** 및 **VirtualMachineInstance(vmi)** 오브젝트 유형을 포함하여 OpenShift Container Platform 리소스를 관리하는 명령줄 유틸리티입니다.

참고

-n <namespace> 플래그를 사용하여 다른 프로젝트를 지정할 수 있습니다.

표 7.1. **oc** 명령

명령	설명
oc login -u <user_name>	OpenShift Container Platform 클러스터에 **<user_name>**으로 로그인합니다.
oc get <object_type>	현재 프로젝트에서 지정된 오브젝트 유형의 오브젝트 목록을 표시합니다.
oc describe <object_type> <resource_name>	현재 프로젝트의 특정 리소스에 대한 세부 정보를 표시합니다.
oc create -f <object_config>	파일 이름 또는 stdin에서 현재 프로젝트에 리소스를 만듭니다.
oc edit <object_type> <resource_name>	현재 프로젝트의 리소스를 편집합니다.
oc delete <object_type> <resource_name>	현재 프로젝트의 리소스를 삭제합니다.

oc 클라이언트 명령에 대한 자세한 내용은OpenShift Container Platform CLI 툴 설명서를 참조하십시오.

7.3. VIRTCTL 클라이언트 명령

virtctl 클라이언트는 OpenShift Virtualization 리소스를 관리하는 명령줄 유틸리티입니다.

virtctl 명령 목록을 보려면 다음 명령을 실행합니다.

```
$ virtctl help
```

특정 명령과 함께 사용할 수 있는 옵션 목록을 보려면 **-h** 또는 **--help** 플래그와 함께 실행하십시오. 예를 들면 다음과 같습니다.

```
$ virtctl image-upload -h
```

virtctl 명령과 함께 사용할 수 있는 글로벌 명령 옵션 목록을 보려면 다음 명령을 실행합니다.

```
$ virtctl options
```

다음 표에는 OpenShift Virtualization 설명서 전체에서 사용되는 **virtctl** 명령이 포함되어 있습니다.

표 7.2. **virtctl** 클라이언트 명령

명령	설명
virtctl start <vm_name>	가상 머신을 시작합니다.
virtctl start --paused <vm_name>	일시 중지된 상태에서 가상 머신을 시작합니다. 이 옵션을 사용하면 VNC 콘솔에서 부팅 프로세스를 중단할 수 있습니다.
virtctl stop <vm_name>	가상 머신을 중지합니다.
virtctl stop <vm_name> --grace-period 0 --force	가상 머신을 강제 중지합니다. 이 옵션을 사용하면 데이터 불일치 또는 데이터 손실이 발생할 수 있습니다.
virtctl pause vm|vmi <object_name>	가상 머신 또는 가상 머신 인스턴스를 일시 정지합니다. 머신 상태는 메모리에 유지됩니다.
virtctl unpause vm|vmi <object_name>	가상 머신 또는 가상 머신 인스턴스의 일시 정지를 해제합니다.
virtctl migrate <vm_name>	가상 머신을 마이그레이션합니다.
virtctl restart <vm_name>	가상 머신을 재시작합니다.
virtctl expose <vm_name>	가상 머신 또는 가상 머신 인스턴스의 지정된 포트를 전달하고 서비스를 노드의 지정된 포트에 노출하는 서비스를 생성합니다.
virtctl console <vmi_name>	가상 머신 인스턴스의 직렬 콘솔에 연결합니다.
virtctl vnc --kubeconfig=$KUBECONFIG <vmi_name>	가상 머신 인스턴스에 대한 VNC(Virtual Network Client) 연결을 엽니다. 로컬 시스템에 원격 뷰어가 필요한 VNC를 통해 가상 머신 인스턴스의 그래픽 콘솔에 액세스합니다.
virtctl vnc --kubeconfig=$KUBECONFIG --proxy-only=true <vmi-name>	포트 번호를 표시하고 VNC 연결을 통해 뷰어를 사용하여 가상 머신 인스턴스에 수동으로 연결합니다.

명령	설명
virtctl vnc --kubeconfig=$KUBECONFIG --port=<port-number> <vmi-name>	해당 포트를 사용할 수 있는 경우 지정된 포트에서 프록시를 실행할 포트 번호를 지정합니다. 포트 번호를 지정하지 않으면 프록시는 임의의 포트에서 실행됩니다.
virtctl image-upload dv <datavolume_name> --image-path=</path/to/image> --no-create	이미 존재하는 데이터 볼륨에 가상 머신 이미지를 업로드합니다.
virtctl image-upload dv <datavolume_name> --size=<datavolume_size> --image-path=</path/to/image>	가상 머신 이미지를 새 데이터 볼륨에 업로드합니다.
virtctl version	클라이언트 및 서버 버전 정보를 표시합니다.
virtctl fslist <vmi_name>	게스트 머신에서 사용 가능한 전체 파일 시스템 목록을 반환합니다.
virtctl guestosinfo <vmi_name>	운영 체제에 대한 게스트 에이전트 정보를 반환합니다.
virtctl userlist <vmi_name>	게스트 머신에 로그인한 전체 사용자 목록을 반환합니다.

7.4. VIRTCTL GUESTFS를 사용하여 컨테이너 생성

virtctl guestfs 명령을 사용하여 **libguestfs-tools** 및 연결된 PVC(영구 볼륨 클레임)를 사용하여 대화형 컨테이너를 배포할 수 있습니다.

절차

- **libguestfs-tools**를 사용하여 컨테이너를 배포하려면 PVC를 마운트하고 쉘을 연결하려면 다음 명령을 실행합니다.

```
$ virtctl guestfs -n <namespace> <pvc_name> 1
```

1 PVC 이름은 필수 인수입니다. 이를 포함하지 않으면 오류 메시지가 표시됩니다.

7.5. LIBGUESTFS 툴 및 VIRTCTL GUESTFS

Libguestfs 툴을 사용하면 VM(가상 머신) 디스크 이미지에 액세스하고 수정할 수 있습니다**libguestfs** 툴을 사용하여 게스트의 파일을 보고 편집하고, 가상 시스템을 복제 및 빌드하며, 디스크를 포맷하고 크기를 조정할 수 있습니다.

virtctl guestfs 명령과 해당 하위 명령을 사용하여 PVC에서 VM 디스크를 수정, 검사 및 디버깅할 수도 있습니다. 가능한 하위 명령의 전체 목록을 보려면 명령줄에 **virt-** 을 입력하고 Tab 키를 누릅니다. 예를 들면 다음과 같습니다.

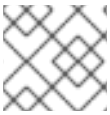
명령	설명
virt-edit -a /dev/vda /etc/motd	터미널에서 파일을 대화식으로 편집합니다.
virt-customize -a /dev/vda --ssh-inject root:string:<public key example>	ssh 키를 게스트에 삽입하고 로그인을 만듭니다.
virt-df -a /dev/vda -h	VM에서 사용하는 디스크 공간 크기를 확인하십시오.
virt-customize -a /dev/vda --run-command 'rpm -qa > /rpm-list'	전체 목록이 포함된 출력 파일을 생성하여 게스트에 설치된 모든 RPM의 전체 목록을 확인하십시오.
virt-cat -a /dev/vda /rpm-list	터미널에서 **virt-customize -a /dev/vda --run-command 'rpm -qa > /rpm-list'** 명령을 사용하여 생성된 모든 RPM의 출력 파일 목록을 표시합니다.
virt-sysprep -a /dev/vda	템플릿으로 사용할 가상 시스템 디스크 이미지를 봉인합니다.

기본적으로 virtctl guestfs 는 VM 디스크를 관리하는 데 필요한 모든 내용으로 세션을 생성합니다. 그러나 이 명령은 동작을 사용자 지정하려는 경우 여러 플래그 옵션도 지원합니다.

플래그 옵션	설명
--h 또는 **--help**	**guestfs**에 대한 도움말을 제공합니다.
<pvc_name> 인수가 있는 **-n <namespace>** 옵션	특정 네임스페이스에서 PVC를 사용하려면 다음을 수행합니다. **-n <namespace>** 옵션을 사용하지 않는 경우 현재 프로젝트가 사용됩니다. 프로젝트를 변경하려면 **oc project <namespace>**를 사용합니다. **<pvc_name>** 인수를 포함하지 않으면 오류 메시지가 표시됩니다.
--image string	**libguestfs-tools** 컨테이너 이미지를 나열합니다. **--image** 옵션을 사용하여 사용자 지정 이미지를 사용하도록 컨테이너를 구성할 수 있습니다.
--kvm	**libguestfs-tools** 컨테이너에서 **kvm**이 사용됨을 나타냅니다. 기본적으로 **virtctl guestfs**는 대화형 컨테이너에 대해 **kvm**을 설정하므로 QEMU를 사용하기 때문에 **libguest-tools** 실행 속도가 훨씬 빨라집니다. 클러스터에 **kvm** 지원 노드가 없는 경우 **--kvm=false** 옵션을 설정하여 **kvm**을 비활성화해야 합니다. 설정되지 않은 경우 **libguestfs-tools** Pod는 모든 노드에서 예약할 수 없으므로 보류 중으로 유지됩니다.

플래그 옵션	설명
--pull-policy string	**libguestfs** 이미지의 가져오기 정책을 표시합니다. **pull-policy** 옵션을 설정하여 이미지의 가져오기 정책을 덮어쓸 수도 있습니다.

또한 명령은 다른 pod에서 PVC를 사용 중인지 확인합니다. 이 경우 오류 메시지가 표시됩니다. 그러나 libguestfs-tools 프로세스가 시작되면 동일한 PVC를 사용하는 새 Pod를 방지할 수 없습니다. 동일한 PVC에 액세스하는 VM을 시작하기 전에 활성 virtctl guestfs Pod가 없는지 확인해야 합니다.



참고

virtctl guestfs 명령은 대화형 Pod에 연결된 단일 PVC만 허용합니다.

7.6. 추가 리소스

- libguestfs: 가상 머신 디스크 이미지에 액세스하고 수정하기 위한 툴입니다

8장. 가상 머신

8.1. 가상 머신 생성

가상 머신을 생성하려면 다음 절차 중 하나를 사용하십시오.

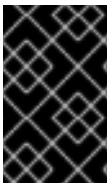
- 빠른 시작 기능 둘러보기
- 마법사 실행
- 가상 머신 마법사를 사용하여 사전 구성된 YAML 파일 붙여넣기
- CLI 사용

주의

openshift-* 네임스페이스에 가상 머신을 생성하지 마십시오. 대신 새 네임스페이스를 만들거나 **openshift** 접두사 없이 기존 네임스페이스를 사용하십시오.

웹 콘솔에서 가상 머신을 생성하는 경우 부팅 소스로 구성된 가상 머신 템플릿을 선택합니다. 부팅 소스가 있는 가상 머신 템플릿은 사용 가능한 부팅 소스로 라벨이 지정되거나 사용자 지정된 라벨 텍스트가 표시됩니다. 사용 가능한 부팅 소스와 함께 템플릿을 사용하면 가상 머신 생성 프로세스가 활성화됩니다.

부팅 소스가 없는 템플릿은 부팅 소스 필요로 라벨이 지정됩니다. 가상 머신에 부팅 소스를 추가하기 위한 단계를 완료하면 이러한 템플릿을 사용할 수 있습니다.



중요

스토리지 동작의 차이로 인해 일부 가상 머신 템플릿은 단일 노드 OpenShift와 호환되지 않습니다. 호환성을 보장하기 위해 데이터 볼륨 또는 스토리지 프로필을 사용하는 템플릿 또는 가상 머신의 **evictionStrategy** 필드를 설정하지 마십시오.

8.1.1. 빠른 시작을 사용한 가상 머신 생성

웹 콘솔은 가상 머신을 생성하기 위한 명령 기능 둘러보기가 포함된 빠른 시작을 제공합니다. 관리자로 도움말 메뉴를 선택하여 빠른 시작 카탈로그에 액세스할 수 있습니다. 빠른 시작 타일을 클릭하고 둘러보기를 시작할 때 시스템이 프로세스를 안내합니다.

빠른 시작의 작업은 Red Hat 템플릿을 선택하면 시작됩니다. 그런 다음 부팅 소스를 추가하고 운영 체제 이미지를 가져올 수 있습니다. 마지막으로 사용자 지정 템플릿을 저장하고 가상 머신을 생성할 수 있습니다.

사전 요구 사항

- 운영 체제 이미지의 URL 링크를 다운로드할 수 있는 웹 사이트에 액세스합니다.

절차

1. 웹 콘솔의 도움말 메뉴에서 빠른 시작을 선택합니다.

2. 빠른 시작 카탈로그에서 타일을 클릭합니다. 예: Red Hat Linux Enterprise Linux 가상 머신 생성

3. 기능 둘러보기의 지침에 따라 운영 체제 이미지를 가져오고 가상 머신을 생성하는 작업을 완료합니다. 가상화 → VirtualMachines 페이지에 가상 머신이 표시됩니다.

8.1.2. 가상 머신 마법사를 실행하여 가상 머신 생성

웹 콘솔에는 가상 머신 템플릿을 선택하고 가상 머신 생성 프로세스를 안내하는 마법사가 있습니다. Red Hat 가상 머신 템플릿은 운영 체제, 플레이버(CPU 및 메모리), 워크로드 유형(서버)가 기본값으로 설정된 운영 체제 이미지로 사전 구성됩니다. 부팅 소스로 템플릿을 구성하면 사용자 정의 라벨 텍스트 또는 기본 라벨 텍스트인 사용 가능 부팅 소스로 라벨이 지정됩니다. 그러면 이러한 템플릿을 가상 머신을 생성하는 데 사용할 수 있습니다.

사전 구성 템플릿 목록에서 템플릿을 선택하고, 설정을 검토한 후 템플릿에서 가상 머신 만들기 마법사에서 가상 머신을 생성할 수 있습니다. 가상 머신을 사용자 지정하는 경우, 마법사는 일반, 네트워킹, 스토리지, 고급 및 검토를 통해 안내합니다. 마법사에서 모든 필수 필드는 *로 표시됩니다.

나중에 NIC(네트워크 인터페이스 컨트롤러) 및 스토리지 디스크를 생성하여 가상 머신에 연결합니다.

절차

1. 사이드 메뉴에서 워크로드 → 가상화를 클릭합니다.

2. 가상 머신 탭 또는 템플릿 탭에서 생성을 클릭하고 마법사를 통한 가상 머신을 선택합니다.

3. 부팅 소스로 구성된 템플릿을 선택합니다.

4. 다음을 클릭하여 검토 및 생성 단계로 이동합니다.

5. 가상 머신을 지금 시작하지 않으려면 생성 후 가상 머신 시작확인란의 선택을 해제합니다.

6. 가상 머신 생성을 클릭하고 마법사를 종료하거나 마법사를 사용하여 가상 머신을 사용자 지정합니다.

7. 가상 머신 사용자 지정을 클릭하여 일반 단계로 이동합니다.

   a. 선택 사항: 이름 필드를 편집하여 가상 머신의 사용자 지정 이름을 지정합니다.

   b. 선택 사항: 설명 필드에 설명을 추가합니다.

8. 다음을 클릭하여 네트워킹 단계로 이동합니다. 기본적으로 nic0 NIC가 연결되어 있습니다.

   a. 선택 사항: 네트워크 인터페이스 추가를 클릭하여 추가 NIC를 생성합니다.

   b. 선택 사항: 옵션 메뉴  를 클릭하고 삭제를 선택하여 일부 또는 모든 NIC를 제거할 수 있습니다. 가상 머신을 생성하기 위해 NIC를 연결하지 않아도 됩니다. 가상 머신을 생성한 후에 NIC를 생성할 수 있습니다.

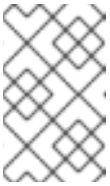
9. 다음을 클릭하여 스토리지 단계로 이동합니다.

   a. 선택 사항: 디스크 추가를 클릭하여 추가 디스크를 생성합니다. 이러한 디스크는 옵션 메뉴  를 클릭하고 삭제를 선택하여 제거할 수 있습니다.

b. **선택 사항: 옵션 메뉴** **를 클릭하여 디스크를 편집하고 변경 사항을 저장합니다.**

10. **다음을 클릭하여 고급 단계로 이동하여 다음 옵션 중 하나를 선택합니다.**

a. **VM을 생성하기 위해 Linux 템플릿을 선택한 경우 Cloud-init의 세부 정보를 검토하고 SSH 액세스를 구성합니다.**



참고

cloud-init 또는 마법사에서 사용자 지정 스크립트를 사용하여 SSH 키를 정적으로 삽입합니다. 이를 통해 가상 시스템을 안전하고 원격으로 관리하고 정보를 관리 및 전송할 수 있습니다. 이 단계에서는 VM을 보호하는 것이 좋습니다.

b. **Windows 템플릿을 선택하여 VM을 생성하는 경우 SysPrep 섹션을 사용하여 자동 Windows 설정을 위해 XML 형식으로 응답 파일을 업로드합니다.**

11. **다음을 클릭하여 검토 단계로 이동하고 가상 머신의 설정을 확인합니다.**

12. **가상 머신 생성을 클릭합니다.**

13. **가상 머신 세부 정보 보기를 클릭하여 이 가상 머신에 대한개요를 확인합니다.**
가상 머신은 가상 머신 탭에 나열됩니다.

웹 콘솔 마법사를 실행하는 경우 가상 머신 마법사 필드 섹션을 참조하십시오.

8.1.2.1. 가상 머신 마법사 필드

이름	매개변수	설명
이름		이름에는 소문자(**a-z**), 숫자(**0-9**), 하이픈(**-**)이 최대 253자까지 포함될 수 있습니다. 첫 문자와 마지막 문자는 영숫자여야 합니다. 이름에는 대문자, 공백, 마침표(**.**) 또는 특수 문자가 포함되어서는 안 됩니다.
설명		선택적 설명 필드입니다.
운영 체제		템플릿에서 가상 머신에 대해 선택된 운영 체제입니다. 템플릿에서 가상 머신을 생성할 때 이 필드를 편집할 수 없습니다.
부팅 소스	URL (PVC 생성)	HTTP 또는 HTTPS 끝점의 사용 가능한 이미지에서 콘텐츠를 가져옵니다. 예: 운영 체제 이미지가 있는 웹 페이지에서 URL 링크를 획득합니다.

이름	매개변수	설명
	복제(PVC 생성)	클러스터에서 사용 가능한 기존 영구 볼륨 클레임을 선택하고 복제합니다.
	레지스트리(PVC 생성)	클러스터에서 액세스할 수 있는 레지스트리의 부팅 가능한 운영 체제 컨테이너에서 가상 머신을 프로비저닝합니다. 예를 들면 ***kubevirt/cirros-registry-disk-demo***입니다.
	PXE (네트워크 부팅-네트워크 인터페이스 추가)	네트워크의 서버에서 운영 체제를 부팅합니다. PXE 부팅 가능 네트워크 연결 정의가 필요합니다.
영구 볼륨 클레임 프로젝트		PVC 복제에 사용할 프로젝트 이름입니다.
영구 볼륨 클레임 이름		기존 PVC를 복제하는 경우 이 가상 머신 템플릿에 적용할 PVC 이름입니다.
CD-ROM 부팅 소스로 마운트		운영 체제를 설치하기 위한 추가 디스크가 CD-ROM에 필요합니다. 확인란을 선택하여 디스크를 추가하고 나중에 사용자 지정합니다.
플레이버	매우 작음, 작음, 중간, 큼, 사용자 정의	해당 템플릿과 연결된 운영 체제에 따라 가상 시스템에 할당된 사전 정의된 값을 사용하여 가상 머신 템플릿의 CPU 및 메모리 양을 미리 설정합니다. 기본 템플릿을 선택하는 경우 사용자 지정 값을 사용하여 템플릿의 **cpus** 및 **memsize** 값을 재정의하여 사용자 지정 템플릿을 생성할 수 있습니다. 또는 템플릿 **세부 정보 페이지의 일반 탭에서 cpus 및 memsize 값을 수정하여 사용자 지정 템플릿을** 생성할 수 있습니다.

이름	매개변수	설명
워크로드 유형  **참고** 잘못된 **워크로드 유형을** 선택하는 경우 성능 또는 리소스 사용률 문제(예: 느린 UI)가 있을 수 있습니다.	데스크탑	데스크탑에서 사용할 가상 머신 구성입니다. 소규모에서 사용하기에 매우 적합합니다. 웹 콘솔과 함께 사용하는 것이 좋습니다. 이 템플릿 클래스 또는 서버 템플릿 클래스를 사용하여 **guaranteed** VM 성능보다 VM 밀도를 우선시합니다.
	서버	성능을 밸런싱하고 다양한 서버 워크로드와 호환됩니다. 이 템플릿 클래스 또는 데스크탑 템플릿 클래스를 사용하여 **guaranteed** VM 성능보다 VM 밀도를 우선시합니다.
	고성능 (CPU 관리자 필요)	고성능 워크로드에 최적화된 가상 머신 구성입니다. 이 템플릿 클래스를 사용하여 VM 밀도보다 **guaranteed** VM 성능 우선 순위를 지정합니다.
생성 후 가상 머신 시작.		이 확인란은 기본적으로 선택되며 가상 머신이 생성 후 실행됩니다. 가상 머신이 생성될 때 시작하지 않도록 하려면 확인란의 선택을 해제합니다.

CPU 관리자를 활성화하여 고성능 워크로드 프로필을 사용합니다.

8.1.2.1.1. 네트워킹 필드

이름	설명
이름	네트워크 인터페이스 컨트롤러의 이름입니다.
모델	네트워크 인터페이스 컨트롤러의 모델을 나타냅니다. 지원되는 값은 **e1000e** 및 **virtio**입니다.
네트워크	사용 가능한 네트워크 연결 정의 목록입니다.
유형	사용 가능한 바인딩 방법 목록입니다. 네트워크 인터페이스에 적합한 바인딩 방법을 선택합니다. • 기본 Pod 네트워크: **masquerade** • Linux 브리지 네트워크: **bridge** • SR-IOV 네트워크: **SR-IOV**

이름	설명
MAC 주소	네트워크 인터페이스 컨트롤러의 MAC 주소입니다. MAC 주소를 지정하지 않으면 주소가 자동으로 할당됩니다.

8.1.2.2. 스토리지 필드

이름	선택	설명
소스	비어있음 (PVC 생성)	빈 디스크를 만듭니다.
	URL을 통해 가져오기 (PVC 생성)	URL(HTTP 또는 HTTPS 끝점)을 통해 콘텐츠를 가져옵니다.
	기존 PVC 사용	클러스터에서 이미 사용 가능한 PVC를 사용합니다.
	기존 PVC 복제 (PVC 생성)	클러스터에서 사용 가능한 기존 PVC를 선택하고 복제합니다.
	레지스트리를 통해 가져오기(PVC 생성)	컨테이너 레지스트리를 통해 콘텐츠를 가져옵니다.
	컨테이너 (임시)	클러스터에서 액세스할 수 있는 레지스트리에 있는 컨테이너에서 콘텐츠를 업로드합니다. 컨테이너 디스크는 CD-ROM 또는 임시 가상 머신과 같은 읽기 전용 파일 시스템에만 사용해야 합니다.
이름		디스크 이름입니다. 이름에는 소문자(**a-z**), 숫자(**0-9**), 하이픈(**-**), 마침표(**.**)가 최대 253자까지 포함될 수 있습니다. 첫 문자와 마지막 문자는 영숫자여야 합니다. 이름에는 대문자, 공백 또는 특수 문자가 포함되어서는 안 됩니다.
크기		디스크 크기(GiB)입니다.

이름	선택	설명
유형		디스크의 유형입니다. 예: 디스크 또는 CD-ROM
인터페이스		디스크 장치의 유형입니다. 지원되는 인터페이스는 **virtIO**, **SATA**, **SCSI**입니다.
스토리지 클래스		디스크를 만드는 데 사용되는 스토리지 클래스입니다.

고급 스토리지 설정

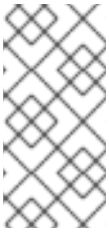
다음 고급 스토리지 설정은 선택 사항이며 비어 있는 Blank,URL을 통해 가져오기, 기존 PVC 복제 디스크에 사용할 수 있습니다. OpenShift Virtualization 4.11 이전에는 이러한 매개변수를 지정하지 않으면 kubevirt-storage-class-defaults 구성 맵의 기본값이 사용됩니다. OpenShift Virtualization 4.11 이상에서는 시스템에서 스토리지 프로필 의 기본값을 사용합니다.

참고

OpenShift Virtualization용 스토리지를 프로비저닝할 때 스토리지 프로필을 사용하여 일관된 고급 스토리지 설정을 보장합니다.

볼륨 모드 및 액세스 모드를 수동으로 지정하려면 기본적으로 최적화된 StorageProfile 설정 적용 확인란의 선택을 취소해야 합니다.

이름	모드 설명	매개변수	매개변수 설명
볼륨 모드	영구 볼륨에서 포맷된 파일 시스템을 사용하는지 또는 원시 블록 상태를 사용하는지를 정의합니다. 기본값은 **Filesystem**입니다.	파일 시스템	파일 시스템 기반 볼륨에 가상 디스크를 저장합니다.
		블록	가상 디스크를 블록 볼륨에 직접 저장합니다. 기본 스토리지에서 지원하는 경우에만 **Block**을 사용하십시오.
액세스 모드	영구 볼륨의 액세스 모드입니다.	ReadWriteOnce (RWO)	볼륨은 단일 노드에서 읽기-쓰기로 마운트할 수 있습니다.

이름	모드 설명	매개변수	매개변수 설명
		ReadWriteMany (RWX)	볼륨은 한 번에 여러 노드에서 읽기-쓰기로 마운트할 수 있습니다.  **참고** 이는 가상 머신의 노드 간 실시간 마이그레이션 등 일부 기능에 필요합니다.
		ReadOnlyMany (ROX)	볼륨은 여러 노드에서만 읽기로 마운트할 수 있습니다.

8.1.2.3. Cloud-init 필드

이름	설명
호스트 이름	가상 머신의 특정 호스트 이름을 설정합니다.
승인된 SSH 키	가상 머신의 *~/.ssh/authorized_keys*에 복사되는 사용자의 공개 키입니다.
사용자 정의 스크립트	기타 옵션을 사용자 정의 cloud-init 스크립트를 붙여넣는 필드로 교체합니다.

스토리지 클래스 기본값을 구성하려면 스토리지 프로필을 사용합니다. 자세한 내용은 스토리지 프로파일 사용자 지정을 참조하십시오.

8.1.2.4. 사전 구성된 YAML 파일에 붙여넣어 가상 머신 생성

YAML 구성 파일을 쓰거나 붙여넣어 가상 머신을 생성합니다. YAML 편집 화면을 열 때마다 기본적으로 유효한 example 가상 머신 구성이 제공됩니다.

생성을 클릭할 때 YAML 구성이 유효하지 않으면 오류 메시지에 오류가 발생하는 매개변수가 표시됩니다. 한 번에 하나의 오류만 표시됩니다.

참고

편집하는 동안 YAML 화면을 벗어나면 구성 변경 사항이 취소됩니다.

절차

1. **사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.**
2. **생성을 클릭하고 YAML 사용을 선택합니다.**

3. 편집 가능한 창에서 가상 머신 구성을 작성하거나 붙여넣습니다.
   a. 또는 YAML 화면에서 기본적으로 제공되는 **example** 가상 머신을 사용하십시오.
4. 선택 사항: YAML 구성 파일을 현재 상태로 다운로드하려면 다운로드를 클릭합니다.
5. 생성을 클릭하여 가상 머신을 생성합니다.

가상 머신은 VirtualMachines 페이지에 나열됩니다.

8.1.3. CLI를 사용하여 가상 머신 생성

virtualMachine 매니페스트에서 가상 머신을 생성할 수 있습니다.

절차

1. VM의 **VirtualMachine** 매니페스트를 편집합니다. 예를 들어 다음 매니페스트에서는 RHEL(Red Hat Enterprise Linux) VM을 구성합니다.

예 8.1. RHEL VM의 매니페스트 예

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  labels:
    app: <vm_name> 1
  name: <vm_name>
spec:
  dataVolumeTemplates:
  - apiVersion: cdi.kubevirt.io/v1beta1
    kind: DataVolume
    metadata:
      name: <vm_name>
    spec:
      sourceRef:
        kind: DataSource
        name: rhel9
        namespace: openshift-virtualization-os-images
      storage:
        resources:
          requests:
            storage: 30Gi
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/domain: <vm_name>
    spec:
      domain:
        cpu:
          cores: 1
          sockets: 2
          threads: 1
        devices:
          disks:
          - disk:
```

```
        bus: virtio
      name: rootdisk
    - disk:
        bus: virtio
      name: cloudinitdisk
    interfaces:
    - masquerade: {}
      name: default
    rng: {}
  features:
    smm:
      enabled: true
  firmware:
    bootloader:
      efi: {}
  resources:
    requests:
      memory: 8Gi
evictionStrategy: LiveMigrate
networks:
- name: default
  pod: {}
volumes:
- dataVolume:
    name: <vm_name>
  name: rootdisk
- cloudInitNoCloud:
    userData: |-
      #cloud-config
      user: cloud-user
      password: '<password>' 2
      chpasswd: { expire: False }
  name: cloudinitdisk
```

1 가상 머신의 이름을 지정합니다.

2 cloud-user의 암호를 지정합니다.

2. 매니페스트 파일을 사용하여 가상 머신을 생성합니다.

```
$ oc create -f <vm_manifest_file>.yaml
```

3. 선택 사항: 가상 머신을 시작합니다.

```
$ virtctl start <vm_name>
```

8.1.4. 가상 머신 스토리지 볼륨 유형

스토리지 볼륨 유형	설명

스토리지 볼륨 유형	설명
임시	네트워크 볼륨을 읽기 전용 백업 저장소로 사용하는 로컬 COW(기록 중 복사) 이미지입니다. 백업 볼륨은 **PersistentVolumeClaim**이어야 합니다. 임시 이미지는 가상 머신이 시작되고 모든 쓰기를 로컬로 저장할 때 생성됩니다. 임시 이미지는 가상 머신이 중지, 재시작 또는 삭제될 때 삭제됩니다. 백업 볼륨(PVC)은 어떤 식으로든 변경되지 않습니다.
persistentVolumeClaim	사용 가능한 PV를 가상 머신에 연결합니다. PV를 연결하면 세션이 바뀌어도 가상 머신 데이터가 지속됩니다. 기존 가상 머신을 OpenShift Container Platform으로 가져올 때는 CDI를 사용하여 기존 가상 머신 디스크를 PVC로 가져와서 PVC를 가상 머신 인스턴스에 연결하는 것이 좋습니다. PVC 내에서 디스크를 사용하려면 몇 가지 요구 사항이 있습니다.
dataVolume	데이터 볼륨은 가져오기, 복제 또는 업로드 작업을 통해 가상 머신 디스크를 준비하는 프로세스를 관리하여 **PersistentVolumeClaim** 디스크 유형에 빌드합니다. 이 볼륨 유형을 사용하는 VM은 볼륨이 준비된 후 시작할 수 있습니다. **type: dataVolume** 또는 **type: ""**로 지정합니다. **type**에 다른 값(예: **persistentVolumeClaim**)을 지정하면 경고가 표시되고 가상 머신이 시작되지 않습니다.
cloudInitNoCloud	참조된 cloud-init NoCloud 데이터 소스가 포함된 디스크를 연결하여 가상 머신에 사용자 데이터 및 메타데이터를 제공합니다. 가상 머신 디스크 내부에 cloud-init을 설치해야 합니다.

스토리지 볼륨 유형	설명
containerDisk	컨테이너 이미지 레지스트리에 저장된 가상 머신 디스크와 같은 이미지를 참조합니다. 이 이미지는 가상 머신이 시작될 때 레지스트리에서 가져와서 가상 머신에 디스크로 연결됩니다. **containerDisk** 볼륨은 단일 가상 머신에 국한되지 않으며, 영구 스토리지가 필요하지 않은 다수의 가상 머신 클론을 생성하는 데 유용합니다. 컨테이너 이미지 레지스트리에는 RAW 및 QCOW2 형식의 디스크 유형만 지원됩니다. 이미지 크기를 줄이기 위해 QCOW2를 사용하는 것이 좋습니다.  **참고** **containerDisk**는 임시 볼륨입니다. 이 볼륨은 가상 머신이 중지, 재시작 또는 삭제될 때 삭제됩니다. **containerDisk** 볼륨은 CD-ROM과 같은 읽기 전용 파일 시스템이나 일회용 가상 머신에 유용합니다.
emptyDisk	가상 머신 인터페이스의 라이프사이클에 연결된 추가 스파스(sparse) QCOW2 디스크를 생성합니다. 해당 데이터는 가상 머신에서 게스트가 시작한 재부팅 후에는 유지되지만 가상 머신이 중지되거나 웹 콘솔에서 재시작되면 삭제됩니다. 빈 디스크는 임시 디스크의 제한된 임시 파일 시스템 크기를 초과하지 않도록 애플리케이션 종속성 및 데이터를 저장하는 데 사용됩니다. 디스크 **용량** 크기도 제공해야 합니다.

8.1.5. 가상 머신 RunStrategies 정보

가상 머신에 대한 RunStrategy는 일련의 조건에 따라 VMI(가상 머신 인스턴스)의 동작을 결정합니다. spec.runStrategy 설정은 spec.running 설정의 대안으로, 가상 머신 구성 프로세스 내에 있습니다. spec.runStrategy 설정을 사용하면 true 또는 false 응답만 있는 spec.running 설정과 달리 VMI를 만들고 관리하는 방법에 대한 유연성을 높일 수 있습니다. 그러나 두 설정은 함께 사용할 수 없습니다. spec.running 또는 spec.runStrategy 중 하나만 사용할 수 있습니다. 둘 다 사용하면 오류가 발생합니다.

RunStrategies는 다음과 같이 네 가지로 정의되어 있습니다.

Always

가상 머신이 생성될 때 VMI가 항상 존재합니다. 어떠한 이유로 원본이 중지되면 새 VMI가 생성되는데, 이러한 동작은 spec.running: true와 동일합니다.

RerunOnFailure

오류로 인해 이전 인스턴스가 실패하면 VMI가 다시 생성됩니다. 가상 머신이 종료될 때와 같이 성공적으로 중지되면 인스턴스가 다시 생성되지 않습니다.

Manual

start, stop, restart virtctl 클라이언트 명령을 사용하여 VMI의 상태 및 존재를 제어할 수 있습니다.

Halted

가상 머신이 생성될 때 VMI가 존재하지 않으며 이 동작은 **spec.running: false**와 동일합니다.

start, stop, restart virtctl 명령의 다양한 조합은 사용되는**RunStrategy**에 영향을 미칩니다.

다음 표에는 다양한 상태에 따른 VM 전환이 표시되어 있습니다. 첫 번째 열에는 VM의 초기 **RunStrategy**가 표시되어 있습니다. 각 추가 열에는 virtctl 명령과 해당 명령이 실행된 후의 새 **RunStrategy**가 표시되어 있습니다.

초기 RunStrategy	시작	중지	재시작
Always	-	Halted	Always
RerunOnFailure	-	Halted	RerunOnFailure
Manual	Manual	Manual	Manual
Halted	Always	-	-

참고

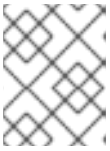
설치 관리자 프로비저닝 인프라를 사용하여 설치한 OpenShift Virtualization 클러스터에서 노드가 MachineHealthCheck에 실패하여 클러스터에서 노드를 사용할 수 없는 경우, 새 노드에 RunStrategy가 **Always** 또는 **RerunOnFailure**인 VM이 다시 예약됩니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  RunStrategy: Always (1)
  template:
...
```

(1) VMI의 현재 **RunStrategy** 설정입니다.

8.1.6. 추가 리소스

- KubeVirt v0.49.0 API 참조의 **VirtualMachineSpec** 정의에는 가상 머신 사양의 매개변수 및 계층 구조에 대한 광범위한 컨텍스트가 있습니다.



참고

KubeVirt API 참조는 업스트림 프로젝트 참조이며 OpenShift Virtualization에서 지원되지 않는 매개변수를 포함할 수 있습니다.

- 가상 머신에 **containerDisk** 볼륨으로 추가할 컨테이너 디스크를 준비를 참조하십시오.
- 머신 상태 점검 배포 및 활성화에 대한 자세한 내용은 머신 상태 점검 배포를 참조하십시오.
- 설치 관리자 프로비저닝 인프라에 대한 자세한 내용은 설치 관리자 프로비저닝 인프라 개요를 참조하십시오.

- 스토리지 프로파일 사용자 정의

8.2. 가상 머신 편집

웹 콘솔의 YAML 편집기를 사용하거나 명령줄에서 OpenShift CLI를 사용하여 가상 머신 구성을 업데이트할 수 있습니다. 가상 머신 세부 정보에서 매개변수 서브 세트를 업데이트할 수도 있습니다.

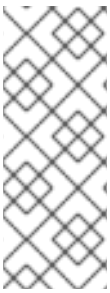
8.2.1. 웹 콘솔에서 가상 머신 편집

관련 필드 옆에 있는 연필 아이콘을 클릭하여 웹 콘솔에서 가상 머신에 대해 선택된 값을 편집합니다. CLI를 사용하여 다른 값을 편집할 수 있습니다.

사전 구성 Red Hat 템플릿과 사용자 지정 가상 머신 템플릿 모두에 대해 라벨 및 주석을 편집할 수 있습니다. 다른 모든 값은 Red Hat 템플릿 또는 가상 머신 템플릿 생성 마법사를 사용하여 생성한 사용자 지정 가상 머신 템플릿만 편집할 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 선택 사항: 필터 드롭다운 메뉴를 사용하여 상태, 템플릿, 노드 또는 운영 체제(OS)와 같은 속성으로 가상 머신 목록을 정렬합니다.
3. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
4. 연필 아이콘을 클릭하여 필드를 편집할 수 있도록 합니다.
5. 관련 사항을 변경하고 저장을 클릭합니다.



참고

가상 머신이 실행 중인 경우 가상 머신을 재시작해야 Boot Order 또는 Flavor에 대한 변경 사항이 적용됩니다.

관련 필드의 오른쪽에서 보류 중인 변경 사항 보기를 클릭하여 보류 중인 변경 사항을 볼 수 있습니다. 페이지 상단의 보류 중인 변경 사항 배너에는 가상 머신이 재시작될 때 적용되는 모든 변경 사항 목록이 표시됩니다.

8.2.1.1. 가상 머신 필드

다음 표에는 OpenShift Container Platform 웹 콘솔에서 편집할 수 있는 가상 머신 필드가 나열되어 있습니다.

표 8.1. 가상 머신 필드

탭	필드 또는 기능

탭	필드 또는 기능
세부 정보	• 라벨 • 주석 • 설명 • CPU/Memory • 부팅 모드 • 부팅 순서 • GPU 장치 • 호스트 장치 • SSH 액세스
YAML	• 사용자 정의 리소스를 보고 편집하거나 다운로드합니다.
스케줄링	• 노드 선택기 • 허용 오차 • 유사성 규칙 • 전용 리소스 • 제거 전략 • Descheduler 설정
네트워크 인터페이스	• 네트워크 인터페이스를 추가, 편집 또는 삭제합니다.
디스크	• 디스크를 추가, 편집 또는 삭제합니다.
스크립트	• cloud-init 설정
스냅샷	• 가상 머신 스냅샷을 추가, 복원 또는 삭제합니다.

8.2.2. 웹 콘솔을 사용하여 가상 머신 YAML 구성 편집

웹 콘솔에서 가상 머신의 YAML 구성을 편집할 수 있습니다. 일부 매개변수는 수정할 수 없습니다. 구성이 유효하지 않은 상태에서 저장을 클릭하면 해당 매개변수를 변경할 수 없다는 오류 메시지가 표시됩니다.

참고

편집하는 동안 YAML 화면을 벗어나면 구성 변경 사항이 취소됩니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택합니다.
3. YAML 탭을 클릭하여 편집 가능한 구성을 표시합니다.
4. 선택 사항: 다운로드를 클릭하여 YAML 파일을 현재 상태에서 로컬로 다운로드할 수 있습니다.
5. 파일을 편집하고 저장을 클릭합니다.

확인 메시지는 수정이 완료되었음을 나타내며 오브젝트의 업데이트된 버전 번호를 포함합니다.

8.2.3. CLI를 사용하여 가상 머신 YAML 구성 편집

CLI를 사용하여 가상 머신 YAML 구성을 편집하려면 다음 절차를 사용하십시오.

사전 요구 사항

- YAML 오브젝트 구성 파일을 사용하여 가상 머신을 구성했습니다.
- oc CLI를 설치했습니다.

절차

1. 다음 명령을 실행하여 가상 머신 구성을 업데이트합니다.

```
$ oc edit <object_type> <object_ID>
```

2. 오브젝트 구성을 엽니다.
3. YAML을 편집합니다.
4. 실행 중인 가상 머신을 편집하는 경우 다음 중 하나를 수행해야 합니다.
   - 가상 머신을 재시작합니다.
   - 새 구성을 적용하려면 다음 명령을 실행합니다.

```
$ oc apply <object_type> <object_ID>
```

8.2.4. 가상 머신에 가상 디스크 추가

가상 디스크를 가상 머신에 추가하려면 다음 절차를 사용하십시오.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 화면을 엽니다.
3. 디스크 탭을 클릭한 다음 디스크 추가를 클릭합니다.
4. 디스크 추가 창에서 소스 ,이름 , 크기,유형,인터페이스, 스토리지 클래스를 지정합니다.
   a. 선택 사항: 빈 디스크 소스를 사용하고 데이터 볼륨을 생성할 때 최대 쓰기 성능이 필요한 경우 사전 할당을 활성화할 수 있습니다. 이를 수행하려면 사전 할당 활성화 확인란을 선택합니다.
   b. 선택 사항: 최적화된 StorageProfile 설정 적용을 지우고 가상 디스크의 볼륨 모드 및 액세스 모드를 변경할 수 있습니다. 이러한 매개변수를 지정하지 않으면 **kubevirt-storage-class-defaults** 구성 맵의 기본값이 사용됩니다.
5. 추가를 클릭합니다.

참고

가상 머신이 실행 중인 경우 새 디스크는 재시작 보류 상태에 있으며, 가상 머신을 재시작할 때까지 연결되지 않습니다.

페이지 상단의 보류 중인 변경 사항 배너에는 가상 머신이 재시작될 때 적용되는 모든 변경 사항 목록이 표시됩니다.

스토리지 클래스 기본값을 구성하려면 스토리지 프로필을 사용합니다. 자세한 내용은 스토리지 프로파일 사용자 지정을 참조하십시오.

8.2.4.1. VirtualMachine의 CD-ROM 편집

가상 머신을 위한 CD-ROM을 편집하려면 다음 절차를 사용하십시오.

절차

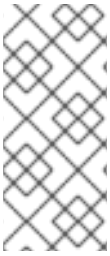
1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 화면을 엽니다.
3. 디스크 탭을 클릭합니다.
4. 편집하려는 CD-ROM의 옵션 메뉴  를 클릭한 후 편집을 선택합니다.
5. CD-ROM 편집 창에서 소스, 영구 볼륨 클레임, 이름, 유형 및 인터페이스 필드를 편집합니다.
6. 저장을 클릭합니다.

8.2.4.2. 스토리지 필드

이름	선택	설명
소스	비어있음 (PVC 생성)	빈 디스크를 만듭니다.

이름	선택	설명
	URL을 통해 가져오기 (PVC 생성)	URL(HTTP 또는 HTTPS 끝점)을 통해 콘텐츠를 가져옵니다.
	기존 PVC 사용	클러스터에서 이미 사용 가능한 PVC를 사용합니다.
	기존 PVC 복제 (PVC 생성)	클러스터에서 사용 가능한 기존 PVC를 선택하고 복제합니다.
	레지스트리를 통해 가져오기(PVC 생성)	컨테이너 레지스트리를 통해 콘텐츠를 가져옵니다.
	컨테이너 (임시)	클러스터에서 액세스할 수 있는 레지스트리에 있는 컨테이너에서 콘텐츠를 업로드합니다. 컨테이너 디스크는 CD-ROM 또는 임시 가상 머신과 같은 읽기 전용 파일 시스템에만 사용해야 합니다.
이름		디스크 이름입니다. 이름에는 소문자(**a-z**), 숫자(**0-9**), 하이픈(**-**), 마침표(**.**)가 최대 253자까지 포함될 수 있습니다. 첫 문자와 마지막 문자는 영숫자여야 합니다. 이름에는 대문자, 공백 또는 특수 문자가 포함되어서는 안 됩니다.
크기		디스크 크기(GiB)입니다.
유형		디스크의 유형입니다. 예: 디스크 또는 CD-ROM
인터페이스		디스크 장치의 유형입니다. 지원되는 인터페이스는 **virtIO**, **SATA**, **SCSI**입니다.
스토리지 클래스		디스크를 만드는 데 사용되는 스토리지 클래스입니다.

고급 스토리지 설정
다음 고급 스토리지 설정은 선택 사항이며 비어 있는 Blank,URL을 통해 가져오기, 기존 PVC 복제 디스크에 사용할 수 있습니다. OpenShift Virtualization 4.11 이전에는 이러한 매개변수를 지정하지 않으면 kubevirt-storage-class-defaults 구성 맵의 기본값이 사용됩니다. OpenShift Virtualization 4.11 이상에서는 시스템에서 스토리지 프로필 의 기본값을 사용합니다.



참고

OpenShift Virtualization용 스토리지를 프로비저닝할 때 스토리지 프로필을 사용하여 일관된 고급 스토리지 설정을 보장합니다.

볼륨 모드 및 액세스 모드를 수동으로 지정하려면 기본적으로 최적화된 StorageProfile 설정 적용 확인란의 선택을 취소해야 합니다.

이름	모드 설명	매개변수	매개변수 설명
볼륨 모드	영구 볼륨에서 포맷된 파일 시스템을 사용하는지 또는 원시 블록 상태를 사용하는지를 정의합니다. 기본값은 **Filesystem**입니다.	파일 시스템	파일 시스템 기반 볼륨에 가상 디스크를 저장합니다.
		블록	가상 디스크를 블록 볼륨에 직접 저장합니다. 기본 스토리지에서 지원하는 경우에만 **Block**을 사용하십시오.
액세스 모드	영구 볼륨의 액세스 모드입니다.	ReadWriteOnce (RWO)	볼륨은 단일 노드에서 읽기-쓰기로 마운트할 수 있습니다.
		ReadWriteMany (RWX)	볼륨은 한 번에 여러 노드에서 읽기-쓰기로 마운트할 수 있습니다.  **참고** 이는 가상 머신의 노드 간 실시간 마이그레이션 등 일부 기능에 필요합니다.
		ReadOnlyMany (ROX)	볼륨은 여러 노드에서만 읽기로 마운트할 수 있습니다.

8.2.5. 가상 머신에 네트워크 인터페이스 추가

가상 머신에 네트워크 인터페이스를 추가하려면 다음 절차를 사용하십시오.

절차

1. **사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.**
2. **가상 머신을 선택하여 VirtualMachine 세부 정보 화면을 엽니다.**
3. **네트워크 인터페이스 탭을 클릭합니다.**
4. **네트워크 인터페이스 추가를 클릭합니다.**

5. 네트워크 인터페이스 추가 창에서 네트워크 인터페이스의 이름, 모델, 네트워크, 유형, MAC 주소를 지정합니다.

6. 추가를 클릭합니다.

참고

가상 머신이 실행 중인 경우 새 네트워크 인터페이스는 재시작 보류 상태에 있으며, 가상 머신을 재시작할 때까지 변경 사항이 적용되지 않습니다.

페이지 상단의 보류 중인 변경 사항 배너에는 가상 머신이 재시작될 때 적용되는 모든 변경 사항 목록이 표시됩니다.

8.2.5.1. 네트워킹 필드

이름	설명
이름	네트워크 인터페이스 컨트롤러의 이름입니다.
모델	네트워크 인터페이스 컨트롤러의 모델을 나타냅니다. 지원되는 값은 **e1000e** 및 **virtio**입니다.
네트워크	사용 가능한 네트워크 연결 정의 목록입니다.
유형	사용 가능한 바인딩 방법 목록입니다. 네트워크 인터페이스에 적합한 바인딩 방법을 선택합니다. • 기본 Pod 네트워크: **masquerade** • Linux 브리지 네트워크: **bridge** • SR-IOV 네트워크: **SR-IOV**
MAC 주소	네트워크 인터페이스 컨트롤러의 MAC 주소입니다. MAC 주소를 지정하지 않으면 주소가 자동으로 할당됩니다.

8.2.6. 추가 리소스

- 스토리지 프로파일 사용자 정의

8.3. 부팅 순서 편집

웹 콘솔 또는 CLI를 사용하여 부팅 순서 목록 값을 업데이트할 수 있습니다.

가상 머신 개요 페이지의 부팅 순서를 사용하여 다음을 수행할 수 있습니다.

- 디스크 또는 NIC(네트워크 인터페이스 컨트롤러)를 선택하고 부팅 순서 목록에 추가합니다.
- 부팅 순서 목록에서 디스크 또는 NIC 순서를 편집합니다.
- 부팅 순서 목록에서 디스크 또는 NIC를 제거하고 부팅 가능 소스 인벤토리로 반환합니다.

8.3.1. 웹 콘솔에서 부팅 순서 목록에 항목 추가

웹 콘솔을 사용하여 부팅 순서 목록에 항목을 추가합니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 세부 정보 탭을 클릭합니다.
4. 부팅 순서 오른쪽에 있는 연필 아이콘을 클릭합니다. YAML 구성이 존재하지 않거나 처음으로 부팅 순서 목록을 생성하는 경우 다음 메시지가 표시됩니다. 선택된 리소스가 없습니다. VM은 YAML 파일에 나타나는 순서에 따라 디스크에서 부팅하려고 합니다.
5. 소스 추가를 클릭하고 가상 시스템의 부팅 가능한 디스크 또는 NIC(네트워크 인터페이스 컨트롤러)를 선택합니다.
6. 부팅 순서 목록에 추가 디스크 또는 NIC를 추가합니다.
7. 저장을 클릭합니다.

참고

가상 머신이 실행 중인 경우 가상 머신을 재시작해야 Boot Order 변경 사항이 적용됩니다.

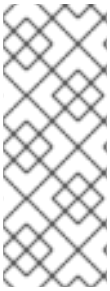
부팅 순서 필드 오른쪽에 있는보류 중인 변경 사항 보기를 클릭하면 보류 중인 변경 사항을 볼 수 있습니다. 페이지 상단의 보류 중인 변경 사항배너에는 가상 머신이 재시작될 때 적용되는 모든 변경 사항 목록이 표시됩니다.

8.3.2. 웹 콘솔에서 부팅 순서 목록 편집

웹 콘솔에서 부팅 순서 목록을 편집합니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 세부 정보 탭을 클릭합니다.
4. 부팅 순서 오른쪽에 있는 연필 아이콘을 클릭합니다.
5. 부팅 순서 목록에서 항목을 이동하는 적절한 방법을 선택합니다.
   - 화면 판독기를 사용하지 않는 경우 이동할 항목 옆에 있는 화살표 아이콘 위로 마우스를 가져가서 항목을 위 또는 아래로 끌어 원하는 위치에 놓습니다.
   - 화면 판독기를 사용하는 경우 위쪽 화살표 키 또는 아래쪽 화살표 키를 눌러 부팅 순서 목록에서 항목을 이동합니다. 그런 다음 Tab 키를 눌러 원하는 위치에 항목을 놓습니다.
6. 저장을 클릭합니다.



참고

가상 머신이 실행중인 경우 가상 머신을 재시작해야 부팅 순서 목록의 변경 사항이 적용됩니다.

부팅 순서 필드 오른쪽에 있는보류 중인 변경 사항 보기를 클릭하면 보류 중인 변경 사항을 볼 수 있습니다. 페이지 상단의 보류 중인 변경 사항배너에는 가상 머신이 재시작될 때 적용되는 모든 변경 사항 목록이 표시됩니다.

8.3.3. YAML 구성 파일의 부팅 순서 목록 편집

CLI를 사용하여 YAML 구성 파일의 부팅 순서 목록을 편집합니다.

절차

1. 다음 명령을 실행하여 가상 머신의 YAML 구성 파일을 엽니다.

```
$ oc edit vm example
```

2. YAML 파일을 편집하고 디스크 또는 NIC(네트워크 인터페이스 컨트롤러)와 연결된 부팅 순서 값을 수정합니다. 예를 들면 다음과 같습니다.

```
disks:
  - bootOrder: 1 1
    disk:
      bus: virtio
    name: containerdisk
  - disk:
      bus: virtio
    name: cloudinitdisk
  - cdrom:
      bus: virtio
    name: cd-drive-1
interfaces:
  - boot Order: 2 2
    macAddress: '02:96:c4:00:00'
    masquerade: {}
    name: default
```

1 디스크에 지정된 부팅 순서 값입니다.

2 네트워크 인터페이스 컨트롤러에 지정된 부팅 순서 값입니다.

3. YAML 파일을 저장합니다.

4. 컨텐츠 다시 로드를 클릭하여 YAML 파일의 업데이트된 부팅 순서 값을 웹 콘솔의 부팅 순서 목록에 적용합니다.

8.3.4. 웹 콘솔의 부팅 순서 목록에서 항목 제거

웹 콘솔을 사용하여 부팅 순서 목록에서 항목을 제거합니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 세부 정보 탭을 클릭합니다.
4. 부팅 순서 오른쪽에 있는 연필 아이콘을 클릭합니다.
5. 해당 항목 옆에 있는 제거 아이콘 ⛔ 을 클릭합니다. 항목이 부팅 순서 목록에서 제거되고 사용 가능한 부팅 소스 목록에 저장됩니다. 부팅 순서 목록의 모든 항목을 제거하면 다음 메시지가 표시됩니다. 선택된 리소스가 없습니다. VM은 YAML 파일에 나타나는 순서에 따라 디스크에서 부팅하려고 합니다.

참고

가상 머신이 실행 중인 경우 가상 머신을 재시작해야 Boot Order 변경 사항이 적용됩니다.

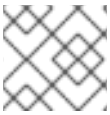
부팅 순서 필드 오른쪽에 있는보류 중인 변경 사항 보기를 클릭하면 보류 중인 변경 사항을 볼 수 있습니다. 페이지 상단의 보류 중인 변경 사항배너에는 가상 머신이 재시작될 때 적용되는 모든 변경 사항 목록이 표시됩니다.

8.4. 가상 머신 삭제

웹 콘솔에서 또는 oc 명령줄 인터페이스를 사용하여 가상 머신을 삭제할 수 있습니다.

8.4.1. 웹 콘솔을 사용하여 가상 머신 삭제

가상 머신을 삭제하면 클러스터에서 가상 머신이 영구적으로 제거됩니다.



참고

가상 머신을 삭제하면 사용하는 데이터 볼륨이 자동으로 삭제됩니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 삭제할 가상 머신의 옵션 메뉴 ⋮ 를 클릭하고 삭제 를 선택합니다.
   - 또는 가상 머신 이름을 클릭하여 VirtualMachine 세부 정보 페이지를 열고 작업 → 삭제 를 클릭합니다.
3. 확인 팝업 창에서 삭제를 클릭하여 가상 머신을 영구적으로 삭제합니다.

8.4.2. CLI를 사용하여 가상 머신 삭제

oc CLI(명령줄 인터페이스)를 사용하여 가상 머신을 삭제할 수 있습니다.oc 클라이언트를 사용하면 여러 가상 머신에서 작업을 수행할 수 있습니다.

참고

가상 머신을 삭제하면 사용하는 데이터 볼륨이 자동으로 삭제됩니다.

사전 요구 사항

- 삭제할 가상 머신의 이름을 확인합니다.

절차

- 다음 명령을 실행하여 가상 머신을 삭제합니다.

```
$ oc delete vm <vm_name>
```

참고

이 명령은 현재 프로젝트에 존재하는 오브젝트만 삭제합니다. 삭제하려는 오브젝트가 다른 프로젝트 또는 네임스페이스에 있는 경우 **-n <project_name>** 옵션을 지정하십시오.

8.5. 가상 머신 인스턴스 관리

OpenShift Virtualization 환경 외부에서 독립적으로 생성된 독립 실행형 VMI(가상 머신 인스턴스)가 있는 경우 웹 콘솔을 사용하거나 CLI(명령줄 인터페이스)에서 **oc** 또는 **virtctl** 명령을 사용하여 관리할 수 있습니다.

virtctl 명령은 **oc** 명령보다 많은 가상화 옵션을 제공합니다. 예를 들어**virtctl** 을 사용하여 VM을 일시 중지하거나 포트를 노출할 수 있습니다.

8.5.1. 가상 머신 인스턴스 정보

VMI(가상 머신 인스턴스)는 실행 중인 VM(가상 머신)을 나타냅니다. VMI가 VM 또는 다른 오브젝트에 속하는 경우, 웹 콘솔의 해당 소유자를 통해 또는 **oc** CLI(명령줄 인터페이스)를 사용하여 VMI를 관리합니다.

독립 실행형 VMI는 스크립트, 자동화 또는 CLI의 다른 방법을 통해 독립적으로 생성 및 시작됩니다. OpenShift Virtualization 환경 외부에서 개발 및 시작된 독립 실행형 VMI가 사용자 환경에 있을 수 있습니다. CLI를 사용하여 이러한 독립 실행형 VMI를 계속 관리할 수 있습니다. 다음과 같이 독립 실행형 VMI와 관련된 특정 작업에 웹 콘솔을 사용할 수도 있습니다.

- 독립 실행형 VMI 및 세부 정보를 나열합니다.
- 독립 실행형 VMI의 라벨 및 주석을 편집합니다.
- 독립 실행형 VMI를 삭제합니다.

VM을 삭제하면 관련 VMI가 자동으로 삭제됩니다. 독립 실행형 VMI는 VM 또는 다른 오브젝트에 속하지 않기 때문에 직접 삭제합니다.

참고

OpenShift Virtualization을 설치 제거하기 전에 CLI 또는 웹 콘솔을 사용하여 독립 실행형 VMI를 나열하고 확인하십시오. 그런 다음 처리 중인 VMI를 삭제합니다.

8.5.2. CLI를 사용하여 모든 가상 머신 인스턴스 나열

oc CLI(명령줄 인터페이스)를 사용하면 독립 실행형 VMI(가상 머신 인스턴스) 및 가상 머신에 속하는 VMI를 포함하여 클러스터의 모든 VMI를 나열할 수 있습니다.

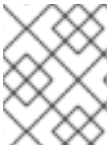
절차

- 다음 명령을 실행하여 VMI를 모두 나열합니다.

```
$ oc get vmis -A
```

8.5.3. 웹 콘솔을 사용하여 독립 실행형 가상 머신 인스턴스 나열

웹 콘솔을 사용하면 VM(가상 머신)에 속하지 않는 클러스터의 독립 실행형 VMI(가상 머신 인스턴스)를 나열하고 확인할 수 있습니다.



참고

VM 또는 다른 오브젝트에 속하는 VMI는 웹 콘솔에 표시되지 않습니다. 웹 콘솔에는 독립 실행형 VMI만 표시됩니다. 클러스터의 모든 VMI를 나열하려면 CLI를 사용해야 합니다.

절차

- 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
  독립 실행형 VMI는 이름 옆에 어두운 색상의 배지로 식별할 수 있습니다.

8.5.4. 웹 콘솔을 사용하여 독립 실행형 가상 머신 인스턴스 편집

웹 콘솔을 사용하여 독립 실행형 VMI(가상 머신 인스턴스)의 주석 및 레이블을 편집할 수 있습니다. 기타 필드는 편집할 수 없습니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 독립 실행형 VMI를 선택하여 VirtualMachineInstance details 페이지를 엽니다.
3. 세부 정보 탭에서 주석 또는 라벨 옆에 있는 연필 아이콘을 클릭합니다.
4. 관련 사항을 변경하고 저장을 클릭합니다.

8.5.5. CLI를 사용하여 독립 실행형 가상 머신 인스턴스 삭제

oc CLI(명령줄 인터페이스)를 사용하여 독립 실행형 VMI(가상 머신 인스턴스)를 삭제할 수 있습니다.

사전 요구 사항

- 삭제할 VMI의 이름을 확인합니다.

절차

- 다음 명령을 실행하여 VMI를 삭제합니다.

```
$ oc delete vmi <vmi_name>
```

8.5.6. 웹 콘솔을 사용하여 독립 실행형 가상 머신 인스턴스 삭제

웹 콘솔에서 독립 실행형 VMI(가상 머신 인스턴스)를 삭제합니다.

절차

1. OpenShift Container Platform 웹 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 작업 → VirtualMachineInstance 삭제를 클릭합니다.
3. 확인 팝업 창에서 삭제를 클릭하여 독립 실행형 VMI를 영구적으로 삭제합니다.

8.6. 가상 머신 상태 제어

웹 콘솔에서 가상 머신을 중지, 시작, 재시작, 일시 정지 해제할 수 있습니다.

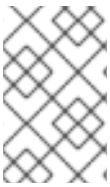
virtctl 을 사용하여 가상 머신 상태를 관리하고 CLI에서 다른 작업을 수행할 수 있습니다. 예를 들어 virtctl 을 사용하여 VM을 강제 중지하거나 포트를 노출할 수 있습니다.

8.6.1. 가상 머신 시작

웹 콘솔에서 가상 머신을 시작할 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 시작할 가상 머신이 포함된 행을 찾습니다.
3. 사용 사례에 적합한 메뉴로 이동합니다.
   - 여러 가상 머신에서 작업을 수행할 수 있는 이 페이지를 유지하려면 다음을 수행하십시오.
     a. 행의 맨 오른쪽 끝에 있는 옵션 메뉴 ⋮ 를 클릭합니다.
   - 시작하기 전에 선택한 가상 머신에 대한 포괄적인 정보를 보려면 다음을 수행하십시오.
     a. 가상 머신 이름을 클릭하여 VirtualMachine 세부 정보 페이지에 액세스합니다.
     b. 작업을 클릭합니다.
4. 재시작 을 선택합니다.
5. 확인 창에서 시작을 클릭하여 가상 머신을 시작합니다.



참고

URL 소스에서 프로비저닝된 가상 머신을 처음 시작하면 가상 머신의 상태가가져오는 중이 되고 OpenShift Virtualization은 URL 끝점에서 컨테이너를 가져옵니다. 이미지 크기에 따라 이 프로세스에 몇 분이 걸릴 수 있습니다.

8.6.2. 가상 머신 재시작

웹 콘솔에서 실행 중인 가상 머신을 재시작할 수 있습니다.

중요

오류를 방지하려면 가져오는 중 상태의 가상 머신을 재시작하지 마십시오.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 재시작할 가상 머신이 포함된 행을 찾습니다.
3. 사용 사례에 적합한 메뉴로 이동합니다.
   - 여러 가상 머신에서 작업을 수행할 수 있는 이 페이지를 유지하려면 다음을 수행하십시오.
     a. 행의 맨 오른쪽 끝에 있는 옵션 메뉴  를 클릭합니다.
   - 재시작하기 전에 선택한 가상 머신에 대한 포괄적인 정보를 보려면 다음을 수행하십시오.
     a. 가상 머신 이름을 클릭하여 VirtualMachine 세부 정보 페이지에 액세스합니다.
     b. 작업 → 재시작 을 클릭합니다.
4. 확인 창에서 재시작을 클릭하여 가상 머신을 재시작합니다.

8.6.3. 가상 머신 중지

웹 콘솔에서 가상 머신을 중지할 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 중지할 가상 머신이 포함된 행을 찾습니다.
3. 사용 사례에 적합한 메뉴로 이동합니다.
   - 여러 가상 머신에서 작업을 수행할 수 있는 이 페이지를 유지하려면 다음을 수행하십시오.
     a. 행의 맨 오른쪽 끝에 있는 옵션 메뉴  를 클릭합니다.
   - 중지하기 전에 선택한 가상 머신에 대한 포괄적인 정보를 보려면 다음을 수행하십시오.
     a. 가상 머신 이름을 클릭하여 VirtualMachine 세부 정보 페이지에 액세스합니다.
     b. 작업 → 중지 를 클릭합니다.
4. 확인 창에서 중지를 클릭하여 가상 머신을 중지합니다.

8.6.4. 가상 머신 정지 해제

웹 콘솔에서 일시 정지된 가상 머신의 일시 정지를 해제할 수 있습니다.

사전 요구 사항

- 가상 머신 중 하나 이상의 상태가 일시 정지됨이어야 합니다.

참고

virtctl 클라이언트를 사용하여 가상 머신을 일시 정지할 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 일시 정지를 해제할 가상 머신이 포함된 행을 찾습니다.
3. 사용 사례에 적합한 메뉴로 이동합니다.
   - 여러 가상 머신에서 작업을 수행할 수 있는 이 페이지를 유지하려면 다음을 수행하십시오.
     a. 상태 열에서 일시 정지됨을 클릭합니다.
   - 일시 정지하기 전에 선택한 가상 머신에 대한 포괄적인 정보를 보려면 다음을 수행하십시오.
     a. 가상 머신 이름을 클릭하여 VirtualMachine 세부 정보 페이지에 액세스합니다.
     b. 상태 오른쪽에 있는 연필 아이콘을 클릭합니다.
4. 확인 창에서 일시 정지 해제를 클릭하여 가상 머신의 일시 정지를 해제합니다.

8.7. 가상 머신 콘솔에 액세스

OpenShift Virtualization에서는 다양한 제품 작업을 수행할 수 있도록 여러 개의 가상 머신 콘솔을 제공합니다. OpenShift Container Platform 웹 콘솔과 CLI 명령을 사용하여 이러한 콘솔에 액세스할 수 있습니다.

8.7.1. OpenShift Container Platform 웹 콘솔에서 가상 머신 콘솔에 액세스

OpenShift Container Platform 웹 콘솔에서 직렬 콘솔 또는 VNC 콘솔을 사용하여 가상 머신에 연결할 수 있습니다.

OpenShift Container Platform 웹 콘솔에서 RDP(원격 데스크탑 프로토콜)를 사용하는 데스크탑 뷰어 콘솔을 사용하여 Windows 가상 머신에 연결할 수 있습니다.

8.7.1.1. 직렬 콘솔 연결

웹 콘솔의 VirtualMachine 세부 정보 페이지에 있는콘솔 탭에서 실행 중인 가상 머신의 직렬 콘솔에 연결합니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 콘솔 탭을 클릭합니다. 기본적으로 VNC 콘솔이 열립니다.

4. 연결 끊기 를 클릭하여 한 번에 하나의 콘솔 세션만 엽니다. 그렇지 않으면 VNC 콘솔 세션이 백그라운드에서 활성 상태로 유지됩니다.
5. VNC 콘솔 드롭다운 목록을 클릭하고 직렬 콘솔을 선택합니다.
6. 연결 끊기 를 클릭하여 콘솔 세션을 종료합니다.
7. 선택 사항: 새 창에서 콘솔 열기를 클릭하여 직렬 콘솔을 별도의 창에서 엽니다.

8.7.1.2. VNC 콘솔에 연결

웹 콘솔의 VirtualMachine 세부 정보 페이지에 있는 콘솔 탭에서 실행 중인 가상 머신의 VNC 콘솔에 연결합니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 콘솔 탭을 클릭합니다. 기본적으로 VNC 콘솔이 열립니다.
4. 선택 사항: 새 창에서 콘솔 열기를 클릭하여 VNC 콘솔을 별도의 창에서 엽니다.
5. 선택 사항: 키 보내기를 클릭하여 가상 머신에 키 조합을 보냅니다.
6. 콘솔 창 외부에서 클릭한 다음 연결 끊기 를 클릭하여 세션을 종료합니다.

8.7.1.3. RDP를 사용하여 Windows 가상 머신에 연결

RDP(원격 데스크탑 프로토콜)를 사용하는 데스크탑 뷰어 콘솔에서는 Windows 가상 머신 연결을 위해 개선된 콘솔 환경을 제공합니다.

RDP를 사용하여 Windows 가상 머신에 연결하려면 웹 콘솔의 VirtualMachine Details 페이지에 있는 콘솔 탭에서 가상 머신의 **console.rdp** 파일을 다운로드하여 선호하는 RDP 클라이언트에 제공하십시오.

사전 요구 사항

- Windows 가상 머신이 실행 중이고 QEMU 게스트 에이전트가 설치되어 있습니다. **qemu-guest-agent**는 VirtIO 드라이버에 포함되어 있습니다.
- 계층 2 NIC가 가상 머신에 연결되어 있습니다.
- RDP 클라이언트가 Windows 가상 머신과 동일한 네트워크의 머신에 설치되어 있습니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. Windows 가상 머신을 클릭하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 콘솔 탭을 클릭합니다.
4. 콘솔 목록에서 데스크탑 뷰어를 선택합니다.

5. 네트워크 인터페이스 목록에서 계층 2 NIC를 선택합니다.
6. 원격 데스크탑 시작을 클릭하여 **console.rdp** 파일을 다운로드합니다.
7. RDP 클라이언트를 열고 **console.rdp** 파일을 참조합니다. 예를 들면 다음과 같이remmina를 사용합니다.

```
$ remmina --connect /path/to/console.rdp
```

8. 관리자 이름 및 암호를 입력하여 Windows 가상 머신에 연결합니다.

8.7.2. CLI 명령을 사용하여 가상 머신 콘솔에 액세스

8.7.2.1. SSH를 통해 가상 머신 인스턴스에 액세스

VM(가상 머신)에 포트 22를 노출하면 SSH를 사용하여 VM에 액세스할 수 있습니다.

virtctl expose 명령은 VMI(가상 머신 인스턴스) 포트를 노드 포트에 전달하고, 활성화된 액세스 권한에 대해 서비스를 생성합니다. 다음 예에서는 클러스터 노드의 특정 포트에서 **<fedora-vm>** 가상 머신의 포트 22로 트래픽을 전달하는 **fedora-vm-ssh** 서비스를 생성합니다.

사전 요구 사항

- VMI와 동일한 프로젝트에 있어야 합니다.
- 액세스하려는 VMI가 가상 바인딩 방법을 사용하여 기본 Pod 네트워크에 연결되어 있어야 합니다.
- 액세스하려는 VMI가 실행 중이어야 합니다.
- OpenShift CLI(oc)를 설치합니다.

절차

1. 다음 명령을 실행하여 **fedora-vm-ssh** 서비스를 생성합니다.

```
$ virtctl expose vm <fedora-vm> --port=22 --name=fedora-vm-ssh --type=NodePort 1
```

1 **<fedora-vm>**은 **fedora-vm-ssh** 서비스를 실행하는 VM의 이름입니다.

2. 서비스를 점검하여 서비스에서 감지한 포트를 확인합니다.

```
$ oc get svc
```

출력 예

```
NAME            TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
fedora-vm-ssh   NodePort   127.0.0.1      <none>        22:32551/TCP     6s
```

이 예에서 서비스는 **32551** 포트를 획득했습니다.

3. SSH를 통해 VMI에 로그인합니다. 클러스터 노드의 **ipAddress** 및 이전 단계에서 찾은 포트를 사용하십시오.

```
$ ssh username@<node_IP_address> -p 32551
```

8.7.2.2. YAML 구성을 사용하여 SSH를 통해 가상 머신에 액세스

virtctl expose 명령을 실행할 필요 없이 VM(가상 머신)에 대한 SSH 연결을 활성화할 수 있습니다. VM의 YAML 파일 및 서비스에 대한 YAML 파일이 구성 및 적용되면 서비스는 SSH 트래픽을 VM으로 전달합니다.

다음 예제에서는 VM의 YAML 파일 및 서비스 YAML 파일의 구성을 보여줍니다.

사전 요구 사항

- OpenShift CLI(**oc**)를 설치합니다.
- **oc create namespace** 명령을 사용하고 네임스페이스 이름을 지정하여 VM의 YAML 파일의 네임스페이스를 만듭니다.

절차

1. VM의 YAML 파일에서 SSH 연결을 위해 서비스를 노출하는 레이블과 값을 추가합니다. 인터페이스의 **masquerade** 기능을 활성화합니다.

   VirtualMachine 정의 예

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  namespace: ssh-ns 1
  name: vm-ssh
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/vm: vm-ssh
        special: vm-ssh 2
    spec:
      domain:
        devices:
          disks:
          - disk:
              bus: virtio
            name: containerdisk
          - disk:
              bus: virtio
            name: cloudinitdisk
          interfaces:
          - masquerade: {} 3
            name: testmasquerade 4
          rng: {}
        machine:
          type: ""
        resources:
          requests:
```

```
        memory: 1024M
    networks:
    - name: testmasquerade
      pod: {}
    volumes:
    - name: containerdisk
      containerDisk:
        image: kubevirt/fedora-cloud-container-disk-demo
    - name: cloudinitdisk
      cloudInitNoCloud:
        userData: |
          #cloud-config
          user: fedora
          password: fedora
          chpasswd: {expire: False}
# ...
```

1. oc create namespace 명령으로 생성된 네임스페이스의 이름입니다.
2. 서비스에서 SSH 트래픽 연결에 활성화된 가상 머신 인스턴스를 식별하는 데 사용하는 레이블입니다. 레이블은 이 YAML 파일에 **label**로 추가되고 서비스 YAML 파일에 **selector**로 추가된 모든 **key:value** 쌍일 수 있습니다.
3. 인터페이스 유형은 **masquerade**입니다.
4. 이 인터페이스의 이름은 **testmasquerade**입니다.

2. VM을 생성합니다.

```
$ oc create -f <path_for_the_VM_YAML_file>
```

3. VM을 시작합니다.

```
$ virtctl start vm-ssh
```

4. 서비스의 YAML 파일에서 서비스 이름, 포트 번호 및 대상 포트를 지정합니다.

   예시 Service 오브젝트

```
apiVersion: v1
kind: Service
metadata:
  name: svc-ssh 1
  namespace: ssh-ns 2
spec:
  ports:
  - targetPort: 22 3
    protocol: TCP
    port: 27017
  selector:
    special: vm-ssh 4
  type: NodePort
# ...
```

1. SSH 서비스의 이름입니다.
2. **oc create namespace** 명령으로 생성된 네임스페이스의 이름입니다.
3. SSH 연결의 대상 포트 번호입니다.
4. 선택기 이름과 값은 VM의 YAML 파일에 지정된 레이블과 일치해야 합니다.

5. 서비스를 생성합니다.

```
$ oc create -f <path_for_the_service_YAML_file>
```

6. VM이 실행 중인지 확인합니다.

```
$ oc get vmi
```

출력 예

```
NAME    AGE    PHASE      IP            NODENAME
vm-ssh 6s      Running    10.244.196.152  node01
```

7. 서비스를 점검하여 서비스에서 감지한 포트를 확인합니다.

```
$ oc get svc
```

출력 예

```
NAME         TYPE      CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
svc-ssh     NodePort      10.106.236.208 <none>       27017:30093/TCP   22s
```

이 예에서 서비스는 포트 번호 30093을 획득했습니다.

8. 다음 명령을 실행하여 노드의 IP 주소를 가져옵니다.

```
$ oc get node <node_name> -o wide
```

출력 예

```
NAME    STATUS   ROLES   AGE    VERSION INTERNAL-IP     EXTERNAL-IP
node01  Ready    worker  6d22h  v1.23.0  192.168.55.101   <none>
```

9. VM이 실행 중인 노드의 IP 주소와 포트 번호를 지정하여 SSH를 통해 VM에 로그인합니다. **oc get svc** 명령에서 표시되는 포트 번호와**oc get node** 명령에서 표시되는 노드의 IP 주소를 사용합니다. 다음 예제에서는 사용자 이름, 노드의 IP 주소 및 포트 번호와 함께 **ssh** 명령을 보여줍니다.

```
$ ssh fedora@192.168.55.101 -p 30093
```

8.7.2.3. 가상 머신 인스턴스의 직렬 콘솔에 액세스

virtctl console 명령은 지정된 가상 머신 인스턴스에 대한 직렬 콘솔을 엽니다.

사전 요구 사항

- **virt-viewer** 패키지가 설치되어 있어야 합니다.
- 액세스하려는 가상 머신 인스턴스가 실행 중이어야 합니다.

절차

- **virtctl**을 사용하여 직렬 콘솔에 연결합니다.

```
$ virtctl console <VMI>
```

8.7.2.4. VNC를 사용하여 가상 머신 인스턴스의 그래픽 콘솔에 액세스

virtctl 클라이언트 유틸리티는 **remote-viewer** 기능을 사용하여 실행 중인 가상 머신 인스턴스에 대해 그래픽 콘솔을 열 수 있습니다. 이 기능은 **virt-viewer** 패키지에 포함되어 있습니다.

사전 요구 사항

- **virt-viewer** 패키지가 설치되어 있어야 합니다.
- 액세스하려는 가상 머신 인스턴스가 실행 중이어야 합니다.

참고

원격 머신에서 SSH를 통해 **virtctl**을 사용하는 경우 X 세션을 머신으로 전달해야 합니다.

절차

1. **virtctl** 유틸리티를 사용하여 그래픽 인터페이스에 연결합니다.

```
$ virtctl vnc <VMI>
```

2. 명령이 실패하는 경우 **-v** 플래그를 사용하여 문제 해결 정보를 수집합니다.

```
$ virtctl vnc <VMI> -v 4
```

8.7.2.5. RDP 콘솔을 사용하여 Windows 가상 머신에 연결

RDP(원격 데스크탑 프로토콜)에서는 Windows 가상 머신 연결을 위해 개선된 콘솔 환경을 제공합니다.

RDP를 사용하여 Windows 가상 머신에 연결하려면 연결된 L2 NIC의 IP 주소를 RDP 클라이언트로 지정하십시오.

사전 요구 사항

- Windows 가상 머신이 실행 중이고 QEMU 게스트 에이전트가 설치되어 있습니다. **qemu-guest-agent**는 VirtIO 드라이버에 포함되어 있습니다.
- 계층 2 NIC가 가상 머신에 연결되어 있습니다.
- RDP 클라이언트가 Windows 가상 머신과 동일한 네트워크의 머신에 설치되어 있습니다.

절차

1. **oc** CLI 툴을 통해 액세스 토큰이 있는 사용자로 OpenShift Virtualization 클러스터에 로그인합니다.

```
$ oc login -u <user> https://<cluster.example.com>:8443
```

2. **oc describe vmi**를 사용하여 실행 중인 Windows 가상 머신의 구성을 표시합니다.

```
$ oc describe vmi <windows-vmi-name>
```

출력 예

```
...
spec:
  networks:
  - name: default
    pod: {}
  - multus:
      networkName: cnv-bridge
    name: bridge-net
...
status:
  interfaces:
  - interfaceName: eth0
    ipAddress: 198.51.100.0/24
    ipAddresses:
      198.51.100.0/24
    mac: a0:36:9f:0f:b1:70
    name: default
  - interfaceName: eth1
    ipAddress: 192.0.2.0/24
    ipAddresses:
      192.0.2.0/24
      2001:db8::/32
    mac: 00:17:a4:77:77:25
    name: bridge-net
...
```

3. 계층 2 네트워크 인터페이스의 IP 주소를 확인하고 복사합니다. 위 예제에서는 **192.0.2.0**이며, IPv6를 선호하는 경우 **2001:db8::**입니다.

4. RDP 클라이언트를 열고 이전 단계에서 복사한 IP 주소를 사용하여 연결합니다.

5. 관리자 이름 및 암호를 입력하여 Windows 가상 머신에 연결합니다.

8.8. SYSPREP을 사용하여 WINDOWS 설치 자동화

Microsoft DVD 이미지와 **sysprep** 을 사용하여 Windows 가상 머신의 설치, 설정 및 소프트웨어 프로비저닝을 자동화할 수 있습니다.

8.8.1. Windows DVD를 사용하여 VM 디스크 이미지 생성

Microsoft는 다운로드용 디스크 이미지를 제공하지 않지만 Windows DVD를 사용하여 디스크 이미지를 만들 수 있습니다. 그러면 이 디스크 이미지를 사용하여 가상 머신을 생성할 수 있습니다.

절차

1. OpenShift Virtualization 웹 콘솔에서 스토리지 → PersistentVolumeClaims → Create PersistentVolumeClaim With Data upload form 을 클릭합니다.
2. 원하는 프로젝트를 선택합니다.
3. 영구 볼륨 클레임 이름을 설정합니다.
4. Windows DVD에서 VM 디스크 이미지를 업로드합니다. 이제 이미지를 새 Windows VM을 생성하기 위해 부팅 소스로 사용할 수 있습니다.

8.8.2. 디스크 이미지를 사용하여 Windows 설치

디스크 이미지를 사용하여 가상 머신에 Windows를 설치할 수 있습니다.

사전 요구 사항

- Windows DVD를 사용하여 디스크 이미지를 생성해야 합니다.
- **autounattend.xml** 응답 파일을 생성해야 합니다. 자세한 내용은Microsoft 설명서를 참조하십시오.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → 카탈로그 를 클릭합니다.
2. Windows 템플릿을 선택하고 VirtualMachine 사용자 지정을 클릭합니다.
3. 디스크 소스 목록에서 업로드(PVC로 새 파일 업로드) 를 선택하고 DVD 이미지로 이동합니다.
4. 검토를 클릭하고 VirtualMachine을 생성합니다.
5. 이 가상 머신에 대해 사용 가능한 운영 체제 소스를 지우십시오
6. 생성 후 이 VirtualMachine을 지웁니다.
7. 스크립트 탭의 Sysprep 섹션에서 편집 을 클릭합니다.
8. **autounattend.xml** 응답 파일로 이동하여저장을 클릭합니다.
9. VirtualMachine 생성을 클릭합니다.
10. YAML 탭에서 **running:false** 를 **runStrategy: RerunOnFailure** 로 교체하고저장을 클릭합니다.

VM은 **autounattend.xml** 응답 파일이 포함된 **sysprep** 디스크로 시작됩니다.

8.8.3. sysprep을 사용하여 Windows VM 일반화

이미지를 일반화하면 이미지가 VM(가상 머신)에 배포될 때 모든 시스템별 구성 데이터를 제거할 수 있습니다.

VM을 일반화하기 전에 **sysprep** 툴이 자동 Windows 설치 후 응답 파일을 감지할 수 없는지 확인해야 합니다.

절차

1. OpenShift Container Platform 콘솔에서 가상화 → VirtualMachines 를 클릭합니다.
2. Windows VM을 선택하여 VirtualMachine details 페이지를 엽니다.
3. 디스크 탭을 클릭합니다.
4. **sysprep** 디스크의 옵션 메뉴  를 클릭하고 분리를 선택합니다.
5. 분리 를 클릭합니다.
6. **sysprep** 도구의 탐지를 방지하기 위해 **C:\Windows\Panther\unattend.xml** 의 이름을 바꿉니다.
7. 다음 명령을 실행하여 **sysprep** 프로그램을 시작합니다.

```
%WINDIR%\System32\Sysprep\sysprep.exe /generalize /shutdown /oobe /mode:vm
```

8. **sysprep** 툴이 완료되면 Windows VM이 종료됩니다. 이제 VM의 디스크 이미지를 Windows VM의 설치 이미지로 사용할 수 있습니다.

이제 VM을 전문으로 설정할 수 있습니다.

8.8.4. Windows 가상 머신 특수

VM(가상 머신)을 전문으로 설정하면 일반적인 Windows 이미지에서 VM에 대한 컴퓨터 관련 정보가 구성됩니다.

사전 요구 사항

- 일반 Windows 디스크 이미지가 있어야 합니다.
- **unattend.xml** 응답 파일을 생성해야 합니다. 자세한 내용은Microsoft 설명서를 참조하십시오.

절차

1. OpenShift Container Platform 콘솔에서 가상화 → 카탈로그 를 클릭합니다.
2. Windows 템플릿을 선택하고 VirtualMachine 사용자 지정을 클릭합니다.
3. 디스크 소스 목록에서 PVC(복제 PVC) 를 선택합니다.
4. 영구 볼륨 클레임 프로젝트 및 일반 Windows 이미지의 영구 볼륨 클레임 이름을 지정합니다.
5. 검토를 클릭하고 VirtualMachine을 생성합니다.
6. 스크립트 탭을 클릭합니다.
7. Sysprep 섹션에서 편집 을 클릭하고 **unattend.xml** 응답 파일을 찾아 저장을 클릭합니다.
8. VirtualMachine 생성을 클릭합니다.

초기 부팅 과정에서 Windows는 **unattend.xml** 응답 파일을 사용하여 VM을 전문으로 설정합니다. 이제 VM을 사용할 준비가 되었습니다.

8.8.5. 추가 리소스

- 가상 머신 생성
- Microsoft, Sysprep(일반화) Windows 설치
- Microsoft, generalize
- Microsoft, 전문

8.9. 실패한 노드를 해결하여 가상 머신 장애 조치 트리거

노드가 실패하고 머신 상태 점검이 클러스터에 배포되지 않는 경우, **RunStrategy: Always**가 구성된 VM(가상 머신)이 정상 노드에 자동으로 재배치되지 않습니다. VM 장애 조치를 트리거하려면 **Node** 오브젝트를 수동으로 삭제해야 합니다.

참고

설치 관리자 프로비저닝 인프라를 사용하여 클러스터를 설치하고 머신 상태 점검을 올바르게 구성한 경우

- 실패한 노드는 자동으로 재활용됩니다.
- **RunStrategy**가 **Always** 또는 **RerunOnFailure**로 설정된 가상 머신은 정상 노드에 자동으로 예약됩니다.

8.9.1. 사전 요구 사항

- 가상 머신을 실행 중이던 노드에 **NotReady** 조건이 있습니다.
- 실패한 노드에서 실행 중이던 가상 머신의 **RunStrategy**가 **Always**로 설정되어 있습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

8.9.2. 베어 메탈 클러스터에서 노드 삭제

CLI를 사용하여 노드를 삭제하면 Kubernetes에서 노드 오브젝트가 삭제되지만 노드에 존재하는 Pod는 삭제되지 않습니다. 복제 컨트롤러에서 지원하지 않는 기본 Pod는 OpenShift Container Platform에 액세스할 수 없습니다. 복제 컨트롤러에서 지원하는 Pod는 사용 가능한 다른 노드로 다시 예약됩니다. 로컬 매니페스트 Pod를 삭제해야 합니다.

절차

다음 단계를 완료하여 베어 메탈에서 실행 중인 OpenShift Container Platform 클러스터에서 노드를 삭제합니다.

1. 노드를 예약 불가능으로 표시합니다.

```
$ oc adm cordon <node_name>
```

2. 노드의 모든 Pod를 드레이닝합니다.

```
$ oc adm drain <node_name> --force=true
```

노드가 오프라인 상태이거나 응답하지 않는 경우 이 단계가 실패할 수 있습니다. 노드가 응답하지 않더라도 공유 스토리지에 쓰는 워크로드를 계속 실행되고 있을 수 있습니다. 데이터 손상을 방지하려면 계속하기 전에 물리적 하드웨어의 전원을 끕니다.

3. 클러스터에서 노드를 삭제합니다.

```
$ oc delete node <node_name>
```

노드 오브젝트가 클러스터에서 삭제되어도 재부팅 후 또는 kubelet 서비스가 재시작되면 클러스터에 다시 참여할 수 있습니다. 노드와 노드의 모든 데이터를 영구적으로 삭제하려면 노드를 해제해야 합니다.

4. 물리 하드웨어의 전원을 끈 경우 노드가 클러스터에 다시 참여할 수 있도록 해당 하드웨어를 다시 켭니다.

8.9.3. 가상 머신 장애 조치 확인

비정상 노드에서 모든 리소스가 종료되면 VM이 재배치될 때마다 정상 노드에 새 VMI(가상 머신 인스턴스)가 자동으로 생성됩니다. VMI가 생성되었는지 확인하려면 **oc** CLI를 사용하여 모든 VMI를 확인합니다.

8.9.3.1. CLI를 사용하여 모든 가상 머신 인스턴스 나열

oc CLI(명령줄 인터페이스)를 사용하면 독립 실행형 VMI(가상 머신 인스턴스) 및 가상 머신에 속하는 VMI를 포함하여 클러스터의 모든 VMI를 나열할 수 있습니다.

절차

- 다음 명령을 실행하여 VMI를 모두 나열합니다.

```
$ oc get vmis -A
```

8.10. 가상 머신에 QEMU 게스트 에이전트 설치

QEMU 게스트 에이전트는 가상 머신에서 실행되고 가상 머신, 사용자, 파일 시스템, 보조 네트워크에 대한 정보를 호스트에 전달하는 데몬입니다.

8.10.1. Linux 가상 머신에 QEMU 게스트 에이전트 설치

qemu-guest-agent는 광범위하게 사용되며, Red Hat 가상 머신에 기본적으로 제공됩니다. 에이전트를 설치하고 서비스를 시작합니다.

VM(가상 머신)에 QEMU 게스트 에이전트가 설치되어 실행되고 있는지 확인하려면 **AgentConnected**가 VM 사양에 나열되어 있는지 확인합니다.

참고

가장 높은 무결성을 가진 온라인(실행 상태) VM의 스냅샷을 생성하려면 QEMU 게스트 에이전트를 설치합니다.

QEMU 게스트 에이전트는 시스템 워크로드에 따라 VM의 파일 시스템을 가능한 한 많이 정지하여 일관된 스냅샷을 사용합니다. 이렇게 하면 스냅샷을 생성하기 전에 진행 중인 I/O가 디스크에 기록됩니다. 게스트 에이전트가 없으면 정지를 수행할 수 없으며 최상의 스냅샷을 생성합니다. 스냅샷이 수행된 조건은 웹 콘솔 또는 CLI에 표시되는 스냅샷 표시에 반영됩니다.

절차

1. 콘솔 중 하나 또는 SSH를 통해 가상 머신 명령줄에 액세스합니다.
2. 가상 머신에 QEMU 게스트 에이전트를 설치합니다.

```
$ yum install -y qemu-guest-agent
```

3. 서비스가 지속되는지 확인하고 다음을 시작합니다.

```
$ systemctl enable --now qemu-guest-agent
```

8.10.2. Windows 가상 머신에 QEMU 게스트 에이전트 설치

Windows 가상 머신의 경우 QEMU 게스트 에이전트는 VirtIO 드라이버에 포함됩니다. 기존 또는 새 Windows 설치에 드라이버를 설치합니다.

VM(가상 머신)에 QEMU 게스트 에이전트가 설치되어 실행되고 있는지 확인하려면 **AgentConnected**가 VM 사양에 나열되어 있는지 확인합니다.

참고

가장 높은 무결성을 가진 온라인(실행 상태) VM의 스냅샷을 생성하려면 QEMU 게스트 에이전트를 설치합니다.

QEMU 게스트 에이전트는 시스템 워크로드에 따라 VM의 파일 시스템을 가능한 한 많이 정지하여 일관된 스냅샷을 사용합니다. 이렇게 하면 스냅샷을 생성하기 전에 진행 중인 I/O가 디스크에 기록됩니다. 게스트 에이전트가 없으면 정지를 수행할 수 없으며 최상의 스냅샷을 생성합니다. 스냅샷이 수행된 조건은 웹 콘솔 또는 CLI에 표시되는 스냅샷 표시에 반영됩니다.

8.10.2.1. 기존 Windows 가상 머신에 VirtIO 드라이버 설치

연결된 SATA CD 드라이브에서 기존 Windows 가상 머신에 VirtIO 드라이버를 설치합니다.

참고

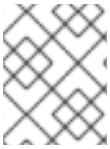
다음 절차에서는 일반적인 방법을 사용하여 Windows에 드라이버를 추가합니다. 프로세스는 Windows 버전마다 약간 다를 수 있습니다. 특정 설치 단계는 사용 중인 Windows 버전의 설치 설명서를 참조하십시오.

절차

1. 가상 머신을 시작하고 그래픽 콘솔에 연결합니다.
2. Windows 사용자 세션에 로그인합니다.
3. 장치 관리자를 열고 기타 장치를 확장하여 알 수 없는 장치를 나열합니다.
   a. **Device Properties**을 열어 알 수 없는 장치를 확인합니다. 장치를 마우스 오른쪽 버튼으로 클릭하고 속성을 선택합니다.
   b. 세부 정보 탭을 클릭하고 속성 목록에서 하드웨어 ID를 선택합니다.
   c. 하드웨어 ID의 값을 지원되는 VirtIO 드라이버와 비교합니다.
4. 장치를 마우스 오른쪽 단추로 클릭하고 드라이버 소프트웨어 업데이트를 선택합니다.
5. 컴퓨터에서 드라이버 소프트웨어 찾아보기를 클릭하고 VirtIO 드라이버가 있는 연결된 SATA CD 드라이브를 찾습니다. 드라이버는 드라이버 유형, 운영 체제, CPU 아키텍처에 따라 계층적으로 정렬됩니다.
6. 다음을 클릭하여 드라이버를 설치합니다.
7. 필요한 모든 VirtIO 드라이버에 대해 이 과정을 반복합니다.
8. 드라이버 설치 후 닫기를 클릭하여 창을 닫습니다.
9. 가상 머신을 재부팅하여 드라이버 설치를 완료합니다.

8.10.2.2. Windows 설치 중 VirtIO 드라이버 설치

Windows를 설치하는 동안 연결된 SATA CD 드라이버에서 VirtIO 드라이버를 설치합니다.



참고

이 절차에서는 일반적인 Windows 설치 방법을 사용하며, 설치 방법은 Windows 버전마다 다를 수 있습니다. 설치 중인 Windows 버전에 대한 설명서를 참조하십시오.

절차

1. 가상 머신을 시작하고 그래픽 콘솔에 연결합니다.
2. Windows 설치 프로세스를 시작합니다.
3. 고급 설치를 선택합니다.
4. 저장 대상은 드라이버가 로드되어야 인식됩니다. **Load driver**를 클릭합니다.
5. 드라이버는 SATA CD 드라이브로 연결되어 있습니다. 확인을 클릭하고 스토리지 드라이버를 로드할 CD 드라이브를 찾습니다. 드라이버는 드라이버 유형, 운영 체제, CPU 아키텍처에 따라 계층적으로 정렬됩니다.
6. 필요한 모든 드라이버에 대해 위의 두 단계를 반복합니다.
7. Windows 설치를 완료합니다.

8.11. 가상 머신에 대한 QEMU 게스트 에이전트 정보 보기

QEMU 게스트 에이전트가 가상 머신에서 실행되는 경우, 웹 콘솔을 사용하여 가상 머신, 사용자, 파일 시스템, 보조 네트워크에 대한 정보를 볼 수 있습니다.

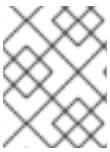
8.11.1. 사전 요구 사항

- 가상 머신에 QEMU 게스트 에이전트를 설치합니다.

8.11.2. 웹 콘솔의 QEMU 게스트 에이전트 정보

QEMU 게스트 에이전트가 설치되면 VirtualMachine 세부 정보 페이지의 개요 및 세부 정보 탭에 호스트 이름, 운영 체제, 시간대, 로그인한 사용자에 대한 정보가 표시됩니다.

VirtualMachine 세부 정보 페이지에는 가상 머신에 설치된 게스트 운영 체제에 대한 정보가 표시됩니다.세부 정보 탭에는 로그인한 사용자에 대한 정보가 포함된 테이블이 표시됩니다.디스크 탭에는 파일 시스템에 대한 정보가 포함된 테이블이 표시됩니다.



참고

QEMU 게스트 에이전트가 설치되지 않은 경우 개요 및 세부 정보 탭에 가상 머신이 생성될 때 지정된 운영 체제에 대한 정보가 표시됩니다.

8.11.3. 웹 콘솔에서 QEMU 게스트 에이전트 정보 보기

웹 콘솔을 사용하여 QEMU 게스트 에이전트에서 호스트로 전달하는 가상 머신에 대한 정보를 볼 수 있습니다.

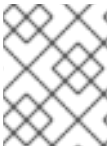
절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신 이름을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 활성 사용자를 보려면 세부 정보 탭을 클릭합니다.
4. 파일 시스템에 대한 정보를 보려면 디스크 탭을 클릭합니다.

8.12. 가상 머신에서 구성 맵, 시크릿, 서비스 계정 관리

시크릿, 구성 맵, 서비스 계정을 사용하여 구성 데이터를 가상 머신에 전달할 수 있습니다. 예를 들면 다음을 수행할 수 있습니다.

- 가상 머신에 시크릿을 추가하여 자격 증명이 필요한 서비스에 대한 액세스 권한을 부여합니다.
- Pod 또는 다른 오브젝트에서 데이터를 사용할 수 있도록 구성 맵에 기밀이 아닌 구성 데이터를 저장합니다.
- 서비스 계정을 특정 구성 요소와 연결하여 해당 구성 요소가 API 서버에 액세스하도록 허용합니다.



참고

OpenShift Virtualization은 시크릿, 구성 맵, 서비스 계정을 가상 머신 디스크로 노출하므로 추가 오버헤드 없이 여러 플랫폼에서 사용할 수 있습니다.

8.12.1. 가상 머신에 시크릿, 구성 맵 또는 서비스 계정 추가

OpenShift Container Platform 웹 콘솔을 사용하여 가상 머신에 시크릿, 구성 맵 또는 서비스 계정을 추가합니다.

이러한 리소스는 가상 머신에 디스크로 추가됩니다. 그런 다음 다른 디스크를 마운트할 때와 같이 시크릿, 구성 맵 또는 서비스 계정을 마운트합니다.

가상 머신이 실행 중인 경우 가상 머신을 재시작해야 변경 사항이 적용됩니다. 새로 추가된 리소스는 페이지 상단 보류 중인 변경 사항 배너의 환경 및 디스크 탭 모두에서 보류 중인 변경 사항으로 표시됩니다.

사전 요구 사항

- 추가할 시크릿, 구성 맵 또는 서비스 계정은 대상 가상 머신과 동일한 네임스페이스에 있어야 합니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 환경 탭에서 구성 맵, 시크릿 또는 서비스 계정 추가를클릭합니다.
4. 리소스 선택을 클릭하고 목록에서 리소스를 선택합니다. 선택한 리소스에 대해 6자리 일련 번호가 자동으로 생성됩니다.
5. 선택 사항: 다시 로드 를 클릭하여 환경을 마지막 저장된 상태로 되돌립니다.
6. 저장을 클릭합니다.

검증

1. VirtualMachine 세부 정보 페이지에서 디스크 탭을 클릭하고 시크릿, 구성 맵 또는 서비스 계정이 디스크 목록에 포함되어 있는지 확인합니다.
2. 작업 → 재시작을 클릭하여 가상 머신을재시작 합니다.

이제 다른 디스크를 마운트할 때와 같이 시크릿, 구성 맵 또는 서비스 계정을 마운트할 수 있습니다.

8.12.2. 가상 머신에서 시크릿, 구성 맵 또는 서비스 계정 제거

OpenShift Container Platform 웹 콘솔을 사용하여 가상 머신에서 시크릿, 구성 맵 또는 서비스 계정을 제거합니다.

사전 요구 사항

- 하나 이상의 시크릿, 구성 맵 또는 서비스 계정이 가상 머신에 연결되어 있어야 합니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 환경 탭을 클릭합니다.

4. 목록에서 삭제할 항목을 찾아 항목 오른쪽에 있는 ⛔ 제거를 클릭합니다.
5. 저장을 클릭합니다.

참고

다시 로드를 클릭하여 폼을 마지막 저장된 상태로 재설정할 수 있습니다.

검증

1. VirtualMachine 세부 정보 페이지에서 디스크 탭을 클릭합니다.
2. 제거한 시크릿, 구성 맵 또는 서비스 계정이 더 이상 디스크 목록에 포함되어 있지 않은지 확인합니다.

8.12.3. 추가 리소스

- Pod에 민감한 데이터 제공
- 서비스 계정 이해 및 생성
- 구성 맵 이해

8.13. 기존 WINDOWS 가상 머신에 VIRTIO 드라이버 설치

8.13.1. VirtIO 드라이버 정보

VirtIO 드라이버는 Microsoft Windows 가상 머신을 OpenShift Virtualization에서 실행하는 데 필요한 준가상화 장치 드라이버입니다. 지원되는 드라이버는 Red Hat Ecosystem Catalog의 **container-native-virtualization/virtio-win** 컨테이너 디스크에 있습니다.

드라이버 설치를 사용하려면 **container-native-virtualization/virtio-win** 컨테이너 디스크를 가상 머신에 SATA CD 드라이브로 연결해야 합니다. VirtIO 드라이버는 가상 머신에 Windows를 설치하는 동안 설치하거나 기존 Windows 설치에 추가할 수 있습니다.

드라이버를 설치하면 **container-native-virtualization/virtio-win** 컨테이너 디스크를 가상 머신에서 제거할 수 있습니다.

새 Windows 가상 머신에 Virtio 드라이버 설치도 참조하십시오.

8.13.2. Microsoft Windows 가상 머신에 지원되는 VirtIO 드라이버

표 8.2. 지원되는 드라이버

드라이버 이름	하드웨어 ID	설명
viostor	VEN_1AF4&DEV_1001 VEN_1AF4&DEV_1042	블록 드라이버입니다. **기타 장치** 그룹에 **SCSI 컨트롤러**로 표시되기도 합니다.
viorng	VEN_1AF4&DEV_1005 VEN_1AF4&DEV_1044	엔트로피 소스 드라이버입니다. **기타 장치** 그룹에 **PCI 장치**로 표시되기도 합니다.

드라이버 이름	하드웨어 ID	설명
NetKVM	VEN_1AF4&DEV_1000 VEN_1AF4&DEV_1041	네트워크 드라이버입니다. **기타 장치** 그룹에 **이더넷 컨트롤러**로 표시되기도 합니다. VirtIO NIC가 구성된 경우에만 사용할 수 있습니다.

8.13.3. 가상 머신에 VirtIO 드라이버 컨테이너 디스크 추가

OpenShift Virtualization에서는 Microsoft Windows용 VirtIO 드라이버를 컨테이너 디스크로 배포하며, Red Hat Ecosystem Catalog에서 사용할 수 있습니다. 이러한 드라이버를 Windows 가상 머신에 설치하려면 가상 머신 구성 파일에서 **container-native-virtualization/virtio-win** 컨테이너 디스크를 가상 머신에 SATA CD 드라이브로 연결하십시오.

사전 요구 사항

- Red Hat Ecosystem Catalog에서 **container-native-virtualization/virtio-win** 컨테이너 디스크를 다운로드합니다. 이 작업은 컨테이너 디스크가 클러스터에 없는 경우 Red Hat 레지스트리에서 다운로드되므로 필수 사항은 아니지만, 설치 시간을 줄일 수 있습니다.

절차

1. Windows 가상 머신 구성 파일에서 **container-native-virtualization/virtio-win** 컨테이너 디스크를 **cdrom** 디스크로 추가합니다. 컨테이너 디스크가 클러스터에 없는 경우 레지스트리에서 다운로드됩니다.

```
spec:
  domain:
    devices:
      disks:
        - name: virtiocontainerdisk
          bootOrder: 2 (1)
          cdrom:
            bus: sata
volumes:
  - containerDisk:
      image: container-native-virtualization/virtio-win
    name: virtiocontainerdisk
```

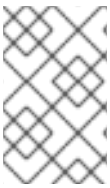
(1) OpenShift Virtualization은 **VirtualMachine** 구성 파일에 정의된 순서대로 가상 머신 디스크를 부팅합니다. **container-native-virtualization/virtio-win** 컨테이너 디스크 전에 기타 디스크를 가상 머신에 정의하거나, 선택적 **bootOrder** 매개변수를 사용하여 가상 머신을 올바른 디스크에서 부팅할 수 있습니다. 디스크에 **bootOrder**를 지정하는 경우 구성의 모든 디스크에 지정해야 합니다.

2. 가상 머신이 시작되면 디스크를 사용할 수 있습니다.
   - 실행 중인 가상 머신에 컨테이너 디스크를 추가할 때는 CLI에 **oc apply -f <vm.yaml>**을 사용하거나 가상 머신을 재부팅하여 변경 사항을 적용합니다.
   - 가상 머신이 실행 중이 아닌 경우에는 **virtctl start <vm>**을 사용합니다.

가상 머신이 시작되면 연결된 SATA CD 드라이브에서 VirtIO 드라이버를 설치할 수 있습니다.

8.13.4. 기존 Windows 가상 머신에 VirtIO 드라이버 설치

연결된 SATA CD 드라이브에서 기존 Windows 가상 머신에 VirtIO 드라이버를 설치합니다.



참고

다음 절차에서는 일반적인 방법을 사용하여 Windows에 드라이버를 추가합니다. 프로세스는 Windows 버전마다 약간 다를 수 있습니다. 특정 설치 단계는 사용 중인 Windows 버전의 설치 설명서를 참조하십시오.

절차

1. 가상 머신을 시작하고 그래픽 콘솔에 연결합니다.
2. Windows 사용자 세션에 로그인합니다.
3. 장치 관리자를 열고 기타 장치를 확장하여 알 수 없는 장치를 나열합니다.
   a. **Device Properties**을 열어 알 수 없는 장치를 확인합니다. 장치를 마우스 오른쪽 버튼으로 클릭하고 속성을 선택합니다.
   b. 세부 정보 탭을 클릭하고 속성 목록에서 하드웨어 ID를 선택합니다.
   c. 하드웨어 ID의 값을 지원되는 VirtIO 드라이버와 비교합니다.
4. 장치를 마우스 오른쪽 단추로 클릭하고 드라이버 소프트웨어 업데이트를 선택합니다.
5. 컴퓨터에서 드라이버 소프트웨어 찾아보기를 클릭하고 VirtIO 드라이버가 있는 연결된 SATA CD 드라이브를 찾습니다. 드라이버는 드라이버 유형, 운영 체제, CPU 아키텍처에 따라 계층적으로 정렬됩니다.
6. 다음을 클릭하여 드라이버를 설치합니다.
7. 필요한 모든 VirtIO 드라이버에 대해 이 과정을 반복합니다.
8. 드라이버 설치 후 닫기를 클릭하여 창을 닫습니다.
9. 가상 머신을 재부팅하여 드라이버 설치를 완료합니다.

8.13.5. 가상 머신에서 VirtIO 컨테이너 디스크 제거

필요한 모든 VirtIO 드라이버를 가상 머신에 설치한 후에는 **container-native-virtualization/virtio-win** 컨테이너 디스크를 더 이상 가상 머신에 연결할 필요가 없습니다. 가상 머신 구성 파일에서 **container-native-virtualization/virtio-win** 컨테이너 디스크를 제거하십시오.

절차

1. 구성 파일을 편집하여 **disk** 및 **volume**을 제거합니다.

```
$ oc edit vm <vm-name>
```

```
spec:
  domain:
```

```
    devices:
      disks:
        - name: virtiocontainerdisk
          bootOrder: 2
          cdrom:
            bus: sata
volumes:
  - containerDisk:
      image: container-native-virtualization/virtio-win
    name: virtiocontainerdisk
```

2. 가상 머신을 재부팅하여 변경 사항을 적용합니다.

8.14. 새로운 WINDOWS 가상 머신에 VIRTIO 드라이버 설치

8.14.1. 사전 요구 사항

- ISO를 데이터 볼륨으로 가져와서 가상 머신에 연결하는 등 가상 머신에서 Windows 설치 미디어에 액세스할 수 있습니다.

8.14.2. VirtIO 드라이버 정보

VirtIO 드라이버는 Microsoft Windows 가상 머신을 OpenShift Virtualization에서 실행하는 데 필요한 준가상화 장치 드라이버입니다. 지원되는 드라이버는 Red Hat Ecosystem Catalog의 **container-native-virtualization/virtio-win** 컨테이너 디스크에 있습니다.

드라이버 설치를 사용하려면 **container-native-virtualization/virtio-win** 컨테이너 디스크를 가상 머신에 SATA CD 드라이브로 연결해야 합니다. VirtIO 드라이버는 가상 머신에 Windows를 설치하는 동안 설치하거나 기존 Windows 설치에 추가할 수 있습니다.

드라이버를 설치하면 **container-native-virtualization/virtio-win** 컨테이너 디스크를 가상 머신에서 제거할 수 있습니다.

기존 Windows 가상 머신에 VirtIO 드라이버 설치도 참조하십시오.

8.14.3. Microsoft Windows 가상 머신에 지원되는 VirtIO 드라이버

표 8.3. 지원되는 드라이버

드라이버 이름	하드웨어 ID	설명
viostor	VEN_1AF4&DEV_1001 VEN_1AF4&DEV_1042	블록 드라이버입니다. **기타 장치** 그룹에 **SCSI 컨트롤러**로 표시되기도 합니다.
viorng	VEN_1AF4&DEV_1005 VEN_1AF4&DEV_1044	엔트로피 소스 드라이버입니다. **기타 장치** 그룹에 **PCI 장치**로 표시되기도 합니다.

드라이버 이름	하드웨어 ID	설명
NetKVM	VEN_1AF4&DEV_1000 VEN_1AF4&DEV_1041	네트워크 드라이버입니다. **기타 장치** 그룹에 **이더넷 컨트롤러**로 표시되기도 합니다. VirtIO NIC가 구성된 경우에만 사용할 수 있습니다.

8.14.4. 가상 머신에 VirtIO 드라이버 컨테이너 디스크 추가

OpenShift Virtualization에서는 Microsoft Windows용 VirtIO 드라이버를 컨테이너 디스크로 배포하며, Red Hat Ecosystem Catalog에서 사용할 수 있습니다. 이러한 드라이버를 Windows 가상 머신에 설치하려면 가상 머신 구성 파일에서 **container-native-virtualization/virtio-win** 컨테이너 디스크를 가상 머신에 SATA CD 드라이브로 연결하십시오.

사전 요구 사항

- Red Hat Ecosystem Catalog에서 **container-native-virtualization/virtio-win** 컨테이너 디스크를 다운로드합니다. 이 작업은 컨테이너 디스크가 클러스터에 없는 경우 Red Hat 레지스트리에서 다운로드되므로 필수 사항은 아니지만, 설치 시간을 줄일 수 있습니다.

절차

1. Windows 가상 머신 구성 파일에서 **container-native-virtualization/virtio-win** 컨테이너 디스크를 **cdrom** 디스크로 추가합니다. 컨테이너 디스크가 클러스터에 없는 경우 레지스트리에서 다운로드됩니다.

```
spec:
  domain:
    devices:
      disks:
        - name: virtiocontainerdisk
          bootOrder: 2 (1)
          cdrom:
            bus: sata
volumes:
  - containerDisk:
      image: container-native-virtualization/virtio-win
    name: virtiocontainerdisk
```

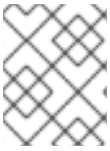
(1) OpenShift Virtualization은 **VirtualMachine** 구성 파일에 정의된 순서대로 가상 머신 디스크를 부팅합니다. **container-native-virtualization/virtio-win** 컨테이너 디스크 전에 기타 디스크를 가상 머신에 정의하거나, 선택적 **bootOrder** 매개변수를 사용하여 가상 머신을 올바른 디스크에서 부팅할 수 있습니다. 디스크에 **bootOrder**를 지정하는 경우 구성의 모든 디스크에 지정해야 합니다.

2. 가상 머신이 시작되면 디스크를 사용할 수 있습니다.
   - 실행 중인 가상 머신에 컨테이너 디스크를 추가할 때는 CLI에 **oc apply -f <vm.yaml>**을 사용하거나 가상 머신을 재부팅하여 변경 사항을 적용합니다.
   - 가상 머신이 실행 중이 아닌 경우에는 **virtctl start <vm>**을 사용합니다.

가상 머신이 시작되면 연결된 SATA CD 드라이브에서 VirtIO 드라이버를 설치할 수 있습니다.

8.14.5. Windows 설치 중 VirtIO 드라이버 설치

Windows를 설치하는 동안 연결된 SATA CD 드라이버에서 VirtIO 드라이버를 설치합니다.



참고

이 절차에서는 일반적인 Windows 설치 방법을 사용하며, 설치 방법은 Windows 버전마다 다를 수 있습니다. 설치 중인 Windows 버전에 대한 설명서를 참조하십시오.

절차

1. 가상 머신을 시작하고 그래픽 콘솔에 연결합니다.
2. Windows 설치 프로세스를 시작합니다.
3. 고급 설치를 선택합니다.
4. 저장 대상은 드라이버가 로드되어야 인식됩니다. **Load driver**를 클릭합니다.
5. 드라이버는 SATA CD 드라이브로 연결되어 있습니다. 확인을 클릭하고 스토리지 드라이버를 로드할 CD 드라이브를 찾습니다. 드라이버는 드라이버 유형, 운영 체제, CPU 아키텍처에 따라 계층적으로 정렬됩니다.
6. 필요한 모든 드라이버에 대해 위의 두 단계를 반복합니다.
7. Windows 설치를 완료합니다.

8.14.6. 가상 머신에서 VirtIO 컨테이너 디스크 제거

필요한 모든 VirtIO 드라이버를 가상 머신에 설치한 후에는 **container-native-virtualization/virtio-win** 컨테이너 디스크를 더 이상 가상 머신에 연결할 필요가 없습니다. 가상 머신 구성 파일에서 **container-native-virtualization/virtio-win** 컨테이너 디스크를 제거하십시오.

절차

1. 구성 파일을 편집하여 **disk** 및 **volume**을 제거합니다.

```
$ oc edit vm <vm-name>
```

```
spec:
  domain:
    devices:
      disks:
        - name: virtiocontainerdisk
          bootOrder: 2
          cdrom:
            bus: sata
volumes:
  - containerDisk:
      image: container-native-virtualization/virtio-win
    name: virtiocontainerdisk
```

2. 가상 머신을 재부팅하여 변경 사항을 적용합니다.

8.15. 고급 가상 머신 관리

8.15.1. 가상 시스템의 리소스 할당량 작업

가상 시스템의 리소스 할당량을 생성하고 관리합니다.

8.15.1.1. 가상 머신에 대한 리소스 할당량 제한 설정

요청만 사용하는 리소스 할당량은 VM(가상 머신)에서 자동으로 작동합니다. 리소스 할당량에서 제한을 사용하는 경우 VM에 리소스 제한을 수동으로 설정해야 합니다. 리소스 제한은 리소스 요청보다 100MiB 이상이어야 합니다.

절차

1. **VirtualMachine** 매니페스트를 편집하여 VM에 대한 제한을 설정합니다. 예를 들면 다음과 같습니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: with-limits
spec:
  running: false
  template:
    spec:
      domain:
# ...
        resources:
          requests:
            memory: 128Mi
          limits:
            memory: 256Mi  (1)
```

(1) **limits.memory** 값이 **requests.memory** 값보다 **100Mi** 이상 크기 때문에 이 구성이 지원됩니다.

2. **VirtualMachine** 매니페스트를 저장합니다.

8.15.1.2. 추가 리소스

- 프로젝트당 리소스 할당량
- 다중 프로젝트의 리소스 할당량

8.15.2. 가상 머신용 노드 지정

노드 배치 규칙을 사용하여 특정 노드에 VM(가상 머신)을 배치할 수 있습니다.

8.15.2.1. 가상 머신의 노드 배치 정보

VM(가상 머신)이 적절한 노드에서 실행되도록 노드 배치 규칙을 구성할 수 있습니다. 다음과 같은 경우 이 작업을 수행할 수 있습니다.

- 여러 개의 VM이 있습니다. 내결함성을 보장하기 위해 서로 다른 노드에서 실행하려고 합니다.
- 두 개의 가상 머신이 있습니다. 중복 노드 간 라우팅을 방지하기 위해 VM을 동일한 노드에서 실행하려고 합니다.
- VM에는 사용 가능한 모든 노드에 존재하지 않는 특정 하드웨어 기능이 필요합니다.
- 노드에 기능을 추가하는 Pod가 있으며 해당 노드에 VM을 배치하여 해당 기능을 사용할 수 있습니다.

참고

가상 머신 배치는 워크로드에 대한 기존 노드 배치 규칙에 의존합니다. 워크로드가 구성 요소 수준의 특정 노드에서 제외되면 해당 노드에 가상 머신을 배치할 수 없습니다.

VirtualMachine 매니페스트의 **spec** 필드에 다음 규칙 유형을 사용할 수 있습니다.

nodeSelector

이 필드에서 지정하는 키-값 쌍으로 레이블이 지정된 노드에서 가상 머신을 예약할 수 있습니다. 노드에는 나열된 모든 쌍과 정확히 일치하는 라벨이 있어야 합니다.

유사성

더 많은 표현 구문을 사용하여 노드와 가상 머신의 일치 규칙을 설정할 수 있습니다. 예를 들어, 규칙을 엄격한 요구 사항이 아닌 기본 설정으로 지정할 수 있으므로 규칙이 충족되지 않은 경우에도 가상 머신을 예약할 수 있습니다. 가상 머신 배치에는 Pod 유사성, Pod 비유사성 및 노드 유사성이 지원됩니다. **VirtualMachine** 워크로드 유형이 **Pod** 오브젝트를 기반으로 하므로 Pod 유사성은 가상 머신에서 작동합니다.

참고

유사성 규칙은 스케줄링 중에만 적용됩니다. 제약 조건이 더 이상 충족되지 않는 경우 OpenShift Container Platform은 실행 중인 워크로드를 다시 예약하지 않습니다.

허용 오차

일치하는 테인트가 있는 노드에 가상 머신을 예약할 수 있습니다. 테인트가 노드에 적용되는 경우, 해당 노드는 테인트를 허용하는 가상 머신만 허용합니다.

8.15.2.2. 노드 배치의 예

다음 예시 YAML 파일 조각에서는 **nodePlacement, affinity** 및 **tolerations** 필드를 사용하여 가상 머신의 노드 배치를 사용자 지정합니다.

8.15.2.2.1. 예: nodeSelector를 사용한 VM 노드 배치

이 예에서 가상 시스템에는 **example-key-1 = example-value-1** 및 **example-key-2 = example-value-2** 레이블을 모두 포함하는 메타데이터가 있는 노드가 필요합니다.

주의

이 설명에 맞는 노드가 없으면 가상 머신이 예약되지 않습니다.

VM 매니페스트 예

```
metadata:
  name: example-vm-node-selector
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  template:
    spec:
      nodeSelector:
        example-key-1: example-value-1
        example-key-2: example-value-2
...
```

8.15.2.2.2. 예: Pod 유사성 및 Pod 비유사성을 사용한 VM 노드 배치

이 예에서는 **example-key-1 = example-value-1** 레이블이 있는 실행 중인 pod가 있는 노드에 VM을 예약해야 합니다. 노드에 실행 중인 Pod가 없는 경우 VM은 예약되지 않습니다.

가능한 경우 **example-key-2 = example-value-2** 레이블이 있는 Pod가 있는 노드에 VM이 예약되지 않습니다. 그러나 모든 후보 노드에 이 레이블이 있는 Pod가 있는 경우 스케줄러는 이 제약 조건을 무시합니다.

VM 매니페스트 예

```
metadata:
  name: example-vm-pod-affinity
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution: 1
      - labelSelector:
          matchExpressions:
          - key: example-key-1
            operator: In
            values:
            - example-value-1
        topologyKey: kubernetes.io/hostname
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution: 2
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchExpressions:
            - key: example-key-2
```

```
            operator: In
            values:
            - example-value-2
        topologyKey: kubernetes.io/hostname
...
```

1. **requiredDuringSchedulingIgnoredDuringExecution** 규칙 유형을 사용하는 경우 제약 조건이 충족되지 않으면 VM이 예약되지 않습니다.
2. **preferredDuringSchedulingIgnoredDuringExecution** 규칙 유형을 사용하는 경우 필요한 모든 제약 조건이 충족되면 여전히 VM이 예약됩니다.

8.15.2.2.3. 예: 노드 선호도를 사용한 VM 노드 배치

이 예에서 VM은 **example.io/example-key = example-value-1** 레이블 또는 **example.io/example-key = example-value-2** 레이블이 있는 노드에 예약해야 합니다. 노드에 레이블 중 하나만 있는 경우 제약 조건이 충족됩니다. 레이블이 모두 없으면 VM이 예약되지 않습니다.

가능한 경우 스케줄러는 **example-node-label-key = example-node-label-value** 레이블이 있는 노드를 피합니다. 그러나 모든 후보 노드에 이 레이블이 있으면 스케줄러는 이 제약 조건을 무시합니다.

VM 매니페스트 예

```
metadata:
  name: example-vm-node-affinity
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution: 1
        nodeSelectorTerms:
        - matchExpressions:
          - key: example.io/example-key
            operator: In
            values:
            - example-value-1
            - example-value-2
      preferredDuringSchedulingIgnoredDuringExecution: 2
      - weight: 1
        preference:
          matchExpressions:
          - key: example-node-label-key
            operator: In
            values:
            - example-node-label-value
...
```

1. **requiredDuringSchedulingIgnoredDuringExecution** 규칙 유형을 사용하는 경우 제약 조건이 충족되지 않으면 VM이 예약되지 않습니다.
2. **preferredDuringSchedulingIgnoredDuringExecution** 규칙 유형을 사용하는 경우 필요한 모든 제약 조건이 충족되면 여전히 VM이 예약됩니다.

8.15.2.2.4. 예: 허용 오차를 사용한 VM 노드 배치

이 예에서는 가상 머신에 예약된 노드가 **key=virtualization:NoSchedule** 테인트로 레이블이 지정됩니다. 이 가상 머신에는 **tolerations**가 일치하므로 테인트된 노드에 예약할 수 있습니다.

참고

해당 테인트가 있는 노드에 스케줄링하는 데 테인트를 허용하는 가상 머신은 필요하지 않습니다.

VM 매니페스트 예

```
metadata:
  name: example-vm-tolerations
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  tolerations:
  - key: "key"
    operator: "Equal"
    value: "virtualization"
    effect: "NoSchedule"
...
```

8.15.2.3. 추가 리소스

- 가상화 구성 요소에 대한 노드 지정
- 노드 선택기를 사용하여 특정 노드에 Pod 배치
- 노드 유사성 규칙을 사용하여 노드에 Pod 배치 제어
- 노드 테인트를 사용하여 Pod 배치 제어

8.15.3. 인증서 교체 구성

기존 인증서를 교체하도록 인증서 교체 매개 변수를 구성합니다.

8.15.3.1. 인증서 교체 구성

웹 콘솔에서 또는 **HyperConverged** CR(사용자 정의 리소스)에 설치 후 OpenShift Virtualization을 설치하는 동안 이 작업을 수행할 수 있습니다.

절차

1. 다음 명령을 실행하여 **HyperConverged** CR을 엽니다.

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

2. 다음 예와 같이 **spec.certConfig** 필드를 편집합니다. 시스템 과부하를 방지하려면 모든 값이 10분 이상인지 확인합니다. golang **ParseDuration** 형식을 준수하는 문자열로 모든 값을 표현합니다.

```
apiVersion: hco.kubevirt.io/v1beta1
```

```
kind: HyperConverged
metadata:
 name: kubevirt-hyperconverged
 namespace: openshift-cnv
spec:
 certConfig:
   ca:
     duration: 48h0m0s
     renewBefore: 24h0m0s (1)
   server:
     duration: 24h0m0s (2)
     renewBefore: 12h0m0s (3)
```

(1) ca.renewBefore의 값은 ca.duration 값보다 작거나 같아야 합니다.

(2) server.duration의 값은 ca.duration 값보다 작거나 같아야 합니다.

(3) server.renewBefore의 값은 server.duration 값보다 작거나 같아야 합니다.

3. YAML 파일을 클러스터에 적용합니다.

8.15.3.2. 인증서 교체 매개변수 문제 해결

기본값이 다음 조건 중 하나와 충돌하지 않는 한 하나 이상의 certConfig 값을 삭제하면 기본값으로 되돌아갑니다.

- ca.renewBefore의 값은 ca.duration 값보다 작거나 같아야 합니다.
- server.duration의 값은 ca.duration 값보다 작거나 같아야 합니다.
- server.renewBefore의 값은 server.duration 값보다 작거나 같아야 합니다.

기본값이 이러한 조건과 충돌하면 오류가 발생합니다.

다음 예제에서 server.duration 값을 제거하는 경우 기본값24h0m0s는 ca.duration 값보다 크므로 지정된 조건과 충돌합니다.

예

```
certConfig:
  ca:
    duration: 4h0m0s
    renewBefore: 1h0m0s
  server:
    duration: 4h0m0s
    renewBefore: 4h0m0s
```

이 경우 다음과 같은 오류 메시지가 표시됩니다.

```
error: hyperconvergeds.hco.kubevirt.io "kubevirt-hyperconverged" could not be patched:
admission webhook "validate-hco.kubevirt.io" denied the request: spec.certConfig:
ca.duration is smaller than server.duration
```

오류 메시지는 첫 번째 충돌만 표시합니다. 진행하기 전에 모든 certConfig 값을 검토합니다.

8.15.4. 관리 작업 자동화

Red Hat Ansible Automation Platform을 사용하여 OpenShift Virtualization 관리 작업을 자동화할 수 있습니다. Ansible Playbook을 사용하여 새 가상 머신을 생성하며 기본 사항을 살펴보십시오.

8.15.4.1. Red Hat Ansible Automation 정보

Ansible은 시스템 구성, 소프트웨어 배포, 롤링 업데이트 수행에 사용되는 자동화 툴입니다. Ansible은 OpenShift Virtualization을 지원하며, Ansible 모듈을 사용하면 템플릿, 영구 볼륨 클레임, 가상 머신 작업과 같은 클러스터 관리 작업을 자동화할 수 있습니다.

Ansible을 통해 OpenShift Virtualization 관리를 자동화할 수 있으며, 이러한 자동화는 **oc** CLI 툴 또는 API를 통해서도 수행할 수 있습니다. Ansible은 KubeVirt 모듈을 다른 Ansible 모듈과 통합할 수 있다는 점에서 고유합니다.

8.15.4.2. 가상 머신 생성 자동화

kubevirt_vm Ansible Playbook을 사용하면 Red Hat Ansible Automation Platform을 통해 OpenShift Container Platform 클러스터에 가상 머신을 생성할 수 있습니다.

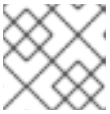
사전 요구 사항

- Red Hat Ansible Engine 버전 2.8 이상

절차

1. **kubevirt_vm** 작업이 포함되도록 Ansible Playbook YAML 파일을 편집합니다.

```
kubevirt_vm:
  namespace:
  name:
  cpu_cores:
  memory:
  disks:
    - name:
      volume:
        containerDisk:
          image:
      disk:
        bus:
```



참고

이 스니펫에는 플레이북의 **kubevirt_vm** 부분만 포함됩니다.

2. 생성할 가상 머신을 반영하도록 **namespace**, **cpu_cores** 수, **memory**, **disks** 등의 값을 편집합니다. 예를 들면 다음과 같습니다.

```
kubevirt_vm:
  namespace: default
  name: vm1
  cpu_cores: 1
  memory: 64Mi
  disks:
```

```
    - name: containerdisk
      volume:
        containerDisk:
          image: kubevirt/cirros-container-disk-demo:latest
      disk:
        bus: virtio
```

3. 생성 후 즉시 가상 머신을 부팅하려면 YAML 파일에 **state: running**을 추가합니다. 예를 들면 다음과 같습니다.

```
kubevirt_vm:
  namespace: default
  name: vm1
  state: running 1
  cpu_cores: 1
```

1 이 값을 **state: absent**로 변경하면 이미 존재하는 가상 머신이 삭제됩니다.

4. 플레이북의 파일 이름을 유일한 인수로 사용하여 **ansible-playbook** 명령을 실행합니다.

```
$ ansible-playbook create-vm.yaml
```

5. 출력을 검토하여 실행이 성공했는지 확인합니다.

출력 예

```
(...)
TASK [Create my first VM] ************************************************************************
changed: [localhost]

PLAY RECAP
*****************************************************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

6. 플레이북 파일에 **state:running**을 포함하지 않은 경우 지금 VM을 부팅하려면**state:running**을 포함하도록 파일을 편집한 후 플레이북을 다시 실행합니다.

```
$ ansible-playbook create-vm.yaml
```

가상 머신이 생성되었는지 확인하려면 VM 콘솔에 액세스하십시오.

8.15.4.3. 예: 가상 머신을 생성하는 Ansible Playbook

kubevirt_vm Ansible Playbook을 사용하여 가상 머신 생성을 자동화할 수 있습니다.

다음 YAML 파일은 **kubevirt_vm** 플레이북의 예입니다. 이 예에는 샘플 값이 포함되어 있으며, 플레이북을 실행하는 경우 해당 값을 사용자의 정보로 교체해야 합니다.

```
---
- name: Ansible Playbook 1
  hosts: localhost
  connection: local
```

```
tasks:
 - name: Create my first VM
   kubevirt_vm:
     namespace: default
     name: vm1
     cpu_cores: 1
     memory: 64Mi
     disks:
       - name: containerdisk
         volume:
           containerDisk:
             image: kubevirt/cirros-container-disk-demo:latest
         disk:
           bus: virtio
```

추가 정보

- 플레이북 소개
- 플레이북 검증 툴

8.15.5. 가상 머신에 UEFI 모드 사용

UEFI(Unified Extensible Firmware Interface) 모드에서 VM(가상 머신)을 부팅할 수 있습니다.

8.15.5.1. 가상 머신의 UEFI 모드 정보

기존 BIOS와 같은 UEFI(Unified Extensible Firmware Interface)는 컴퓨터가 시작될 때 하드웨어 구성 요소 및 운영 체제 이미지 파일을 초기화합니다. UEFI는 BIOS보다 최신 기능 및 사용자 지정 옵션을 지원하므로 부팅 시간이 단축됩니다.

ESP(EFI System Partition)라는 특수 파티션에 저장된 **.efi** 확장자로 파일에 초기화 및 시작에 대한 모든 정보를 저장합니다. ESP에는 컴퓨터에 설치된 운영 체제용 부트 로더 프로그램도 포함되어 있습니다.

8.15.5.2. UEFI 모드에서 가상 머신 부팅

VirtualMachine 매니페스트를 편집하여 UEFI 모드에서 부팅하도록 가상 머신을 구성할 수 있습니다.

사전 요구 사항

- OpenShift CLI(**oc**)를 설치합니다.

절차

1. **VirtualMachine** 매니페스트 파일을 편집하거나 생성합니다.**spec.firmware.bootloader** 스탠자를 사용하여 UEFI 모드를 구성합니다.

   보안 부팅이 활성화된 UEFI 모드에서 부팅

   ```
   apiversion: kubevirt.io/v1
   kind: VirtualMachine
   metadata:
     labels:
       special: vm-secureboot
   ```

```
  name: vm-secureboot
spec:
  template:
    metadata:
      labels:
        special: vm-secureboot
    spec:
      domain:
        devices:
          disks:
          - disk:
              bus: virtio
            name: containerdisk
        features:
          acpi: {}
          smm:
            enabled: true (1)
        firmware:
          bootloader:
            efi:
              secureBoot: true (2)
...
```

1. OpenShift Virtualization에서는 UEFI 모드에서 Secure Boot를 사용하려면**SMM**(시스템 관리 모드)을 활성화해야 합니다.
2. OpenShift Virtualization은 UEFI 모드를 사용할 때 Secure Boot를 포함하거나 사용하지 않고 VM을 지원합니다. Secure Boot가 활성화된 경우 UEFI 모드가 필요합니다. 그러나 Secure Boot를 사용하지 않고 UEFI 모드를 활성화할 수 있습니다.

2. 다음 명령을 실행하여 클러스터에 매니페스트를 적용합니다.

```
$ oc create -f <file_name>.yaml
```

8.15.6. 가상 머신에 대한 PXE 부팅 구성

OpenShift Virtualization에서는 PXE 부팅 또는 네트워크 부팅을 사용할 수 있습니다. 네트워크 부팅의 경우 로컬로 연결된 스토리지 장치 없이 컴퓨터에서 운영 체제 또는 기타 프로그램을 부팅 및 로드할 수 있습니다. 예를 들어, 새 호스트를 배포할 때 PXE 서버에서 원하는 OS 이미지를 선택할 수 있습니다.

8.15.6.1. 사전 요구 사항

- Linux 브리지가 연결되어 있어야 합니다.
- PXE 서버는 브리지와 동일한 VLAN에 연결되어 있어야 합니다.

8.15.6.2. 지정된 MAC 주소로 PXE 부팅

관리자는 PXE 네트워크에 대한 **NetworkAttachmentDefinition** 오브젝트를 생성한 후 네트워크를 통해 클라이언트를 부팅할 수 있습니다. 그런 다음 가상 머신 인스턴스 구성 파일에서 네트워크 연결 정의를 참조한 후 가상 머신 인스턴스를 시작할 수 있습니다. PXE 서버에 필요한 경우 가상 머신 인스턴스 구성 파일에 MAC 주소를 지정할 수도 있습니다.

사전 요구 사항

- Linux 브리지가 연결되어 있어야 합니다.
- PXE 서버는 브리지와 동일한 VLAN에 연결되어 있어야 합니다.

절차

1. 클러스터에서 PXE 네트워크를 구성합니다.
   a. PXE 네트워크 **pxe-net-conf**에 대한 네트워크 연결 정의 파일을 만듭니다.

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: pxe-net-conf
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "pxe-net-conf",
    "plugins": [
      {
        "type": "cnv-bridge",
        "bridge": "br1",
        "vlan": 1 (1)
      },
      {
        "type": "cnv-tuning" (2)
      }
    ]
  }'
```

(1) 선택 사항: VLAN 태그.

(2) **cnv-tuning** 플러그인은 사용자 정의 MAC 주소를 지원합니다.

참고

VLAN이 요청된 액세스 포트를 통해 가상 머신 인스턴스가 브리지 **br1**에 연결됩니다.

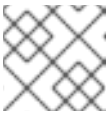
2. 이전 단계에서 만든 파일을 사용하여 네트워크 연결 정의를 생성합니다.

```
$ oc create -f pxe-net-conf.yaml
```

3. 인터페이스 및 네트워크에 대한 세부 정보를 포함하도록 가상 머신 인스턴스 구성 파일을 편집합니다.
   a. PXE 서버에 필요한 경우 네트워크 및 MAC 주소를 지정합니다. MAC 주소를 지정하지 않으면 값이 자동으로 할당됩니다.
   인터페이스가 먼저 부팅되도록 **bootOrder**가 **1**로 설정되어 있는지 확인하십시오. 이 예에서는 인터페이스가 **<pxe-net>**이라는 네트워크에 연결되어 있습니다.

```
interfaces:
```

```
- masquerade: {}
  name: default
- bridge: {}
  name: pxe-net
  macAddress: de:00:00:00:00:de
  bootOrder: 1
```



참고

부팅 순서는 인터페이스 및 디스크에 대해 전역적입니다.

b. 운영 체제가 프로비저닝되면 올바르게 부팅되도록 부팅 장치 번호를 디스크에 할당합니다. 디스크의 **bootOrder** 값을 **2**로 설정합니다.

```
devices:
  disks:
  - disk:
      bus: virtio
    name: containerdisk
    bootOrder: 2
```

c. 네트워크를 이전에 생성한 네트워크 연결 정의에 연결하도록 지정합니다. 이 시나리오에서 **<pxe-net>**는 **<pxe-net-conf>**라는 네트워크 연결 정의에 연결됩니다.

```
networks:
- name: default
  pod: {}
- name: pxe-net
  multus:
    networkName: pxe-net-conf
```

4. 가상 머신 인스턴스를 생성합니다.

```
$ oc create -f vmi-pxe-boot.yaml
```

출력 예

```
virtualmachineinstance.kubevirt.io "vmi-pxe-boot" created
```

1. 가상 머신 인스턴스가 실행될 때까지 기다립니다.

```
$ oc get vmi vmi-pxe-boot -o yaml | grep -i phase
  phase: Running
```

2. VNC를 사용하여 가상 머신 인스턴스를 확인합니다.

```
$ virtctl vnc vmi-pxe-boot
```

3. 부팅 화면에서 PXE 부팅에 성공했는지 확인합니다.

4. 가상 머신 인스턴스에 로그인합니다.

```
$ virtctl console vmi-pxe-boot
```

5. 가상 머신의 인터페이스 및 MAC 주소를 확인하고, 브릿지에 연결된 인터페이스에 MAC 주소가 지정되었는지 확인합니다. 이 예제에서는 IP 주소 없이 PXE 부팅에 **eth1**을 사용했습니다. 다른 인터페이스인 **eth0**은 OpenShift Container Platform에서 IP 주소를 가져왔습니다.

```
$ ip addr
```

출력 예

```
...
3. eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen
1000
  link/ether de:00:00:00:00:de brd ff:ff:ff:ff:ff:ff
```

8.15.6.3. OpenShift Virtualization 네트워킹 용어집

OpenShift Virtualization은 사용자 정의 리소스 및 플러그인을 사용하여 고급 네트워킹 기능을 제공합니다.

다음 용어는 OpenShift Virtualization 설명서 전체에서 사용됩니다.

CNI(컨테이너 네트워크 인터페이스(Container Network Interface))

컨테이너 네트워크 연결에 중점을 둔 Cloud Native Computing Foundation 프로젝트입니다. OpenShift Virtualization에서는 CNI 플러그인을 사용하여 기본 Kubernetes 네트워킹 기능을 기반으로 빌드합니다.

Multus

Pod 또는 가상 머신에서 필요한 인터페이스를 사용할 수 있도록 여러 CNI가 존재할 수 있는 "메타" CNI 플러그인입니다.

CRD(사용자 정의 리소스 정의(Custom Resource Definition))

사용자 정의 리소스를 정의할 수 있는 Kubernetes API 리소스 또는 CRD API 리소스를 사용하여 정의한 오브젝트입니다.

네트워크 연결 정의 (NAD)

Pod, 가상 머신, 가상 머신 인스턴스를 하나 이상의 네트워크에 연결할 수 있는 Multus 프로젝트에서 도입한 CRD입니다.

노드 네트워크 구성 정책(NNCP)

노드에서 요청된 네트워크 구성에 대한 설명입니다. **NodeNetworkConfigurationPolicy** 매니페스트를 클러스터에 적용하는 방식으로 인터페이스 추가 및 제거를 포함하여 노드 네트워크 구성을 업데이트합니다.

PXE(Preboot eXecution Environment)

관리자가 네트워크를 통해 서버에서 클라이언트 머신을 부팅할 수 있는 인터페이스입니다. 네트워크 부팅을 통해 운영 체제 및 기타 소프트웨어를 클라이언트에 원격으로 로드할 수 있습니다.

8.15.7. 가상 머신에서 대규모 페이지 사용

대규모 페이지를 클러스터의 가상 머신 백업 메모리로 사용할 수 있습니다.

8.15.7.1. 사전 요구 사항

- 노드에 사전 할당된 대규모 페이지가 구성되어 있어야 합니다.

8.15.7.2. 대규모 페이지의 기능

메모리는 페이지라는 블록으로 관리됩니다. 대부분의 시스템에서 한 페이지는 4Ki입니다. 1Mi 메모리는 256페이지와 같고 1Gi 메모리는 256,000페이지에 해당합니다. CPU에는 하드웨어에서 이러한 페이지 목록을 관리하는 내장 메모리 관리 장치가 있습니다. TLB(Translation Lookaside Buffer)는 가상-물리적 페이지 매핑에 대한 소규모 하드웨어 캐시입니다. TLB에 하드웨어 명령어로 전달된 가상 주소가 있으면 매핑을 신속하게 확인할 수 있습니다. 가상 주소가 없으면 TLB 누락이 발생하고 시스템에서 소프트웨어 기반 주소 변환 속도가 느려져 성능 문제가 발생합니다. TLB 크기는 고정되어 있으므로 TLB 누락 가능성을 줄이는 유일한 방법은 페이지 크기를 늘리는 것입니다.

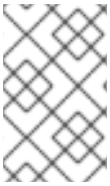
대규모 페이지는 4Ki보다 큰 메모리 페이지입니다. x86_64 아키텍처에서 일반적인 대규모 페이지 크기는 2Mi와 1Gi입니다. 다른 아키텍처에서는 크기가 달라집니다. 대규모 페이지를 사용하려면 애플리케이션이 인식할 수 있도록 코드를 작성해야 합니다. THP(투명한 대규모 페이지)에서는 애플리케이션 지식 없이 대규모 페이지 관리를 자동화하려고 하지만 한계가 있습니다. 특히 페이지 크기 2Mi로 제한됩니다. THP에서는 THP 조각 모음 작업으로 인해 메모리 사용률이 높아지거나 조각화가 발생하여 노드에서 성능이 저하될 수 있으며 이로 인해 메모리 페이지가 잠길 수 있습니다. 이러한 이유로 일부 애플리케이션은 THP 대신 사전 할당된 대규모 페이지를 사용하도록 설계(또는 권장)할 수 있습니다.

OpenShift Virtualization에서는 사전 할당된 대규모 페이지를 사용하도록 가상 머신을 구성할 수 있습니다.

8.15.7.3. 가상 머신용 대규모 페이지 구성

가상 머신 구성에 **memory.hugepages.pageSize** 및 **resources.requests.memory** 매개변수를 포함하여 사전 할당된 대규모 페이지를 사용하도록 가상 머신을 구성할 수 있습니다.

메모리 요청은 페이지 크기로 나눌 수 있어야합니다. 예를 들면 페이지 크기가 **1Gi**인 **500Mi**의 메모리는 요청할 수 없습니다.



참고

호스트와 게스트 OS의 메모리 레이아웃은 관련이 없습니다. 가상 머신 매니페스트에서 요청된 대규모 페이지가 QEMU에 적용됩니다. 게스트 내부의 대규모 페이지는 사용 가능한 가상 머신 인스턴스 메모리 양을 기준으로만 구성할 수 있습니다.

실행 중인 가상 머신을 편집하는 경우 변경 사항을 적용하려면 가상 머신을 재부팅해야 합니다.

사전 요구 사항

- 노드에 사전 할당된 대규모 페이지가 구성되어 있어야 합니다.

절차

1. 가상 머신 구성에서 **spec.domain**에 **resources.requests.memory** 및 **memory.hugepages.pageSize** 매개변수를 추가합니다. 다음 구성 스니펫에서는 가상 머신에서 각 페이지 크기가 **1Gi**인 총 **4Gi**의 메모리를 요청합니다.

```
kind: VirtualMachine
...
spec:
  domain:
    resources:
      requests:
        memory: "4Gi" 1
    memory:
```

```
      hugepages:
        pageSize: "1Gi" (2)
...
```

(1) 가상 머신에 대해 요청된 총 메모리 양입니다. 이 값은 페이지 크기로 나눌 수 있어야 합니다.

(2) 각 대규모 페이지의 크기입니다. x86_64 아키텍처에 유효한 값은 **1Gi** 및 **2Mi**입니다. 페이지 크기는 요청된 메모리보다 작아야 합니다.

2. 가상 머신 구성을 적용합니다.

```
$ oc apply -f <virtual_machine>.yaml
```

8.15.8. 가상 머신 전용 리소스 사용

성능 향상을 위해 CPU와 같은 노드 리소스를 가상 머신에 전용으로 지정할 수 있습니다.

8.15.8.1. 전용 리소스 정보

가상 머신에 전용 리소스를 사용하면 가상 머신의 워크로드가 다른 프로세스에서 사용하지 않는 CPU에 예약됩니다. 전용 리소스를 사용하면 가상 머신의 성능과 대기 시간 예측 정확도를 개선할 수 있습니다.

8.15.8.2. 사전 요구 사항

- 노드에 CPU 관리자를 구성해야 합니다. 가상 머신 워크로드를 예약하기 전에 노드에**cpumanager = true** 라벨이 있는지 확인하십시오.
- 가상 머신의 전원을 꺼야 합니다.

8.15.8.3. 가상 머신 전용 리소스 활성화

세부 정보 탭에서 가상 머신 전용 리소스를 활성화합니다. Red Hat 템플릿에서 생성된 가상 머신은 전용 리소스로 구성할 수 있습니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 스케줄링 탭에서 전용 리소스 옆에 연필 아이콘을 클릭합니다.
4. 전용 리소스(보장된 정책)를 사용하여 이 워크로드 예약을 선택합니다.
5. 저장을 클릭합니다.

8.15.9. 가상 머신 예약

호환성을 위해 VM의 CPU 모델과 정책 특성이 노드에서 지원하는 CPU 모델 및 정책 특성과 일치하도록 하면 노드에 VM(가상 머신)을 예약할 수 있습니다.

8.15.9.1. 정책 특성

VM(가상 머신)을 노드에 예약할 때 호환성을 위해 일치하는 정책 특성과 CPU 기능을 지정하면 VM을 예약할 수 있습니다. VM에 지정되는 정책 특성에 따라 VM이 노드에서 예약되는 방식이 결정됩니다.

정책 특성	설명
force	VM이 노드에 강제로 예약됩니다. 호스트 CPU에서 VM CPU를 지원하지 않는 경우에도 마찬가지입니다.
require	VM이 특정 CPU 모델 및 기능 사양으로 구성되지 않은 경우 VM에 적용되는 기본 정책입니다. 이 기본 정책 특성 또는 다른 정책 특성 중 하나를 사용하여 CPU 노드 검색을 지원하도록 노드를 구성하지 않으면 해당 노드에 VM이 예약되지 않습니다. 호스트 CPU가 VM의 CPU를 지원하거나 하이퍼바이저가 지원되는 CPU 모델을 에뮬레이션할 수 있어야 합니다.
optional	호스트의 물리적 머신 CPU에서 VM을 지원하는 경우 해당 VM이 노드에 추가됩니다.
disable	CPU 노드 검색을 통해 VM을 예약할 수 없습니다.
forbid	호스트 CPU에서 기능을 지원하고 CPU 노드 검색을 사용할 수 있는 경우에도 VM을 예약할 수 없습니다.

8.15.9.2. 정책 특성 및 CPU 기능 설정

각 VM(가상 머신)에 대한 정책 특성 및 CPU 기능을 설정하면 정책 및 기능에 따라 노드에 VM을 예약할 수 있습니다. 설정한 CPU 기능은 호스트 CPU의 지원 여부 또는 하이퍼바이저의 에뮬레이션 여부를 확인하기 위해 검증됩니다.

절차

- **VM 구성 파일의 domain 사양을 편집합니다. 다음 예제에서는 VM(가상 머신)에 대한 CPU 기능 및 require 정책을 설정합니다.**

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: myvm
spec:
  template:
    spec:
      domain:
        cpu:
          features:
            - name: apic (1)
              policy: require (2)
```

(1) VM의 CPU 기능 이름입니다.

(2) VM의 정책 특성입니다.

8.15.9.3. 지원되는 CPU 모델을 사용하여 가상 머신 예약

VM(가상 머신)의 CPU 모델을 구성하여 해당 CPU 모델이 지원되는 노드에 예약할 수 있습니다.

절차

- 가상 머신 구성 파일의 **domain** 사양을 편집합니다. 다음 예는 VM에 대해 정의된 특정 CPU 모델을 보여줍니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: myvm
spec:
  template:
    spec:
      domain:
        cpu:
          model: Conroe 1
```

1 VM의 CPU 모델입니다.

8.15.9.4. 호스트 모델을 사용하여 가상 머신 예약

VM(가상 머신)의 CPU 모델이 **host-model**로 설정되어 있으면 VM은 예약된 노드의 CPU 모델을 상속합니다.

절차

- VM 구성 파일의 **domain** 사양을 편집합니다. 다음 예제에서는 가상 머신에 지정된**host-model**을 보여줍니다.

```
apiVersion: kubevirt/v1alpha3
kind: VirtualMachine
metadata:
  name: myvm
spec:
  template:
    spec:
      domain:
        cpu:
          model: host-model 1
```

1 예약된 노드의 CPU 모델을 상속하는 VM입니다.

8.15.10. PCI 패스스루 구성

PCI(Peripheral Component Interconnect) 패스스루 기능을 사용하면 가상 머신에서 하드웨어 장치에 액세스하고 관리할 수 있습니다. PCI 패스스루가 구성되면 PCI 장치는 게스트 운영 체제에 물리적으로 연결된 것처럼 작동합니다.

클러스터 관리자는 **oc** CLI(명령줄 인터페이스)를 사용하여 클러스터에서 사용할 수 있는 호스트 장치를 노출하고 관리할 수 있습니다.

8.15.10.1. PCI 패스스루를 위한 호스트 장치 준비 정보

CLI를 사용하여 PCI 패스스루를 위한 호스트 장치를 준비하려면 **MachineConfig** 오브젝트를 생성하고 커널 인수를 추가하여 IOMMU(Input-Output Memory Management Unit)를 활성화합니다. PCI 장치를 VFIO(가상 기능 I/O) 드라이버에 연결한 다음 **HyperConverged** CR(사용자 정의 리소스)의 **allowedHostDevices** 필드를 편집하여 클러스터에 노출합니다. OpenShift Virtualization Operator를 처음 설치할 때 **permittedHostDevices** 목록이 비어 있습니다.

CLI를 사용하여 클러스터에서 PCI 호스트 장치를 제거하려면 **HyperConverged** CR에서 PCI 장치 정보를 삭제합니다.

8.15.10.1.1. IOMMU 드라이버를 활성화하려면 커널 인수 추가

커널에서 IOMMU(Input-Output Memory Management Unit) 드라이버를 활성화하려면 **MachineConfig** 개체를 생성하고 커널 인수를 추가합니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 관리자 권한을 보유하고 있어야 합니다.
- Intel 또는 AMD CPU 하드웨어.
- BIOS의 Directed I/O 확장용 Intel Virtualization Technology 또는 AMD IOMMU(Basic Input/Output System)가 활성화되어 있습니다.

절차

1. 커널 인수를 식별하는 **MachineConfig** 오브젝트를 만듭니다. 다음 예제에서는 Intel CPU에 대한 커널 인수를 보여줍니다.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker (1)
  name: 100-worker-iommu (2)
spec:
  config:
    ignition:
      version: 3.2.0
  kernelArguments:
      - intel_iommu=on (3)
...
```

(1) 새 커널 인수를 작업자 노드에만 적용합니다.

(2) **name**은 머신 구성 및 그 용도 중 이 커널 인수의 순위(100)를 나타냅니다. AMD CPU가 있는 경우 커널 인수를 **amd_iommu=on**으로 지정합니다.

(3) Intel CPU의 커널 인수를 **intel_iommu**로 식별합니다.

2. 새 **MachineConfig** 오브젝트를 만듭니다.

```
$ oc create -f 100-worker-kernel-arg-iommu.yaml
```

검증

- 새 **MachineConfig** 오브젝트가 추가되었는지 확인합니다.

```
$ oc get MachineConfig
```

8.15.10.1.2. VFIO 드라이버에 PCI 장치 바인딩

PCI 장치를 VFIO(Virtual Function I/O) 드라이버에 바인딩하려면 각 장치에서 **vendor-ID** 및 **device-ID** 값을 가져오고 값으로 목록을 생성합니다. **MachineConfig** 오브젝트에 이 목록을 추가합니다. **MachineConfig** Operator는 PCI 장치가 있는 노드에 **/etc/modprobe.d/vfio.conf**를 생성하고 PCI 장치를 VFIO 드라이버에 바인딩합니다.

사전 요구 사항

- CPU에 IOMMU를 사용하도록 커널 인수를 추가했습니다.

절차

1. **lspci** 명령을 실행하여 PCI 장치의 **vendor-ID** 및 **device-ID**를 가져옵니다.

```
$ lspci -nnv | grep -i nvidia
```

출력 예

```
02:01.0 3D controller [0302]: NVIDIA Corporation GV100GL [Tesla V100 PCIe 32GB]
[10de:1eb8] (rev a1)
```

2. Virtual config 파일 **100-worker-vfiopci.bu**를 생성하여 PCI 장치를 VFIO 드라이버에 바인딩합니다.

참고

Butane에 대한 자세한 내용은 "Butane을 사용하여 머신 구성 생성"을 참조하십시오.

예

```
variant: openshift
version: 4.10.0
metadata:
  name: 100-worker-vfiopci
  labels:
    machineconfiguration.openshift.io/role: worker 1
storage:
  files:
  - path: /etc/modprobe.d/vfio.conf
    mode: 0644
    overwrite: true
    contents:
      inline: |
        options vfio-pci ids=10de:1eb8 2
  - path: /etc/modules-load.d/vfio-pci.conf 3
```

```
    mode: 0644
    overwrite: true
    contents:
      inline: vfio-pci
```

1. 새 커널 인수를 작업자 노드에만 적용합니다.
2. 단일 장치를 VFIO 드라이버에 바인딩하려면 이전에 결정한 **vendor-ID** 값 (**10de**) 및 **device-ID** 값 (**1eb8**) 을 지정합니다. 공급업체 및 장치 정보를 사용하여 여러 장치 목록을 추가할 수 있습니다.
3. 작업자 노드에서 vfio-pci 커널 모듈을 로드하는 파일입니다.

3. Butane을 사용하여 작업자 노드로 전달할 구성이 포함된 **MachineConfig** 오브젝트 파일 **100-worker-vfiopci.yaml**을 생성합니다.

```
$ butane 100-worker-vfiopci.bu -o 100-worker-vfiopci.yaml
```

4. 작업자 노드에 **MachineConfig** 오브젝트를 적용합니다.

```
$ oc apply -f 100-worker-vfiopci.yaml
```

5. **MachineConfig** 오브젝트가 추가되었는지 확인합니다.

```
$ oc get MachineConfig
```

출력 예

```
NAME                             GENERATEDBYCONTROLLER                      IGNITIONVERSION
AGE
00-master                        d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0             25h
00-worker                        d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0             25h
01-master-container-runtime      d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0
25h
01-master-kubelet                d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0
25h
01-worker-container-runtime      d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0
25h
01-worker-kubelet                d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0
25h
100-worker-iommu                                                            3.2.0             30s
100-worker-vfiopci-configuration                                            3.2.0             30s
```

검증

- VFIO 드라이버가 로드되었는지 확인합니다.

```
$ lspci -nnk -d 10de:
```

출력은 VFIO 드라이버가 사용 중인지 확인합니다.

출력 예

```
04:00.0 3D controller [0302]: NVIDIA Corporation GP102GL [Tesla P40] [10de:1eb8] (rev a1)
        Subsystem: NVIDIA Corporation Device [10de:1eb8]
        Kernel driver in use: vfio-pci
        Kernel modules: nouveau
```

8.15.10.1.3. CLI를 사용하여 클러스터에 PCI 호스트 장치 노출

클러스터에 PCI 호스트 장치를 노출하려면 PCI 장치에 대한 세부 정보를 **HyperConverged** CR(사용자 정의 리소스)의 **spec.permittedHostDevices** 배열에 추가합니다.

절차

1. 다음 명령을 실행하여 기본 편집기에서 **HyperConverged** CR을 편집합니다.

```
$ oc edit hyperconverged kubevirt-hyperconverged -n openshift-cnv
```

2. **spec.permittedHostDevices.pciHostDevices** 어레이에 PCI 장치 정보를 추가합니다. 예를 들면 다음과 같습니다.

   설정 파일 예

```
apiVersion: hco.kubevirt.io/v1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  permittedHostDevices: 1
    pciHostDevices: 2
    - pciDeviceSelector: "10DE:1DB6" 3
      resourceName: "nvidia.com/GV100GL_Tesla_V100" 4
    - pciDeviceSelector: "10DE:1EB8"
      resourceName: "nvidia.com/TU104GL_Tesla_T4"
    - pciDeviceSelector: "8086:6F54"
      resourceName: "intel.com/qat"
      externalResourceProvider: true 5
...
```

1. 클러스터에서 사용할 수 있는 호스트 장치입니다.
2. 노드에서 사용할 수 있는 PCI 장치 목록입니다.
3. **vendor-ID** 및 **device-ID**는 PCI 장치를 식별해야 합니다.
4. PCI 호스트 장치의 이름입니다.
5. 선택 사항: 이 필드를 **true** 로 설정하면 리소스가 외부 장치 플러그인에서 제공되었음을 나타냅니다. OpenShift Virtualization에서는 클러스터에서 이 장치를 사용할 수 있지만 할당 및 모니터링은 외부 장치 플러그인에 그대로 둡니다.

참고

위의 예제 스니펫은 이름이 **nvidia.com/GV100GL_Tesla_V100**이고 **nvidia.com/TU104GL_Tesla_T4**가 **HyperConverged** CR에서 허용된 호스트 장치 목록에 추가된 두 개의 PCI 호스트 장치를 보여줍니다. 이러한 장치는 OpenShift Virtualization에서 작동하도록 테스트 및 검증되었습니다.

3. 변경 사항을 저장하고 편집기를 종료합니다.

검증

- 다음 명령을 실행하여 PCI 호스트 장치가 노드에 추가되었는지 확인합니다. 예제 출력에서는 각각**nvidia.com/GV100GL_Tesla_V100, nvidia.com/TU104GL_Tesla_T4, intel.com/qat** 리소스 이름과 연결된 하나의 장치가 있음을 보여줍니다.

```
$ oc describe node <node_name>
```

출력 예

```
Capacity:
  cpu:                      64
  devices.kubevirt.io/kvm:      110
  devices.kubevirt.io/tun:      110
  devices.kubevirt.io/vhost-net:  110
  ephemeral-storage:            915128Mi
  hugepages-1Gi:                0
  hugepages-2Mi:                0
  memory:                     131395264Ki
  nvidia.com/GV100GL_Tesla_V100   1
  nvidia.com/TU104GL_Tesla_T4     1
  intel.com/qat:                1
  pods:                       250
Allocatable:
  cpu:                      63500m
  devices.kubevirt.io/kvm:      110
  devices.kubevirt.io/tun:      110
  devices.kubevirt.io/vhost-net:  110
  ephemeral-storage:            863623130526
  hugepages-1Gi:                0
  hugepages-2Mi:                0
  memory:                     130244288Ki
  nvidia.com/GV100GL_Tesla_V100   1
  nvidia.com/TU104GL_Tesla_T4     1
  intel.com/qat:                1
  pods:                       250
```

8.15.10.1.4. CLI를 사용하여 클러스터에서 PCI 호스트 장치 제거

클러스터에서 PCI 호스트 장치를 제거하려면 **HyperConverged** CR(사용자 정의 리소스)에서 해당 장치의 정보를 삭제합니다.

절차

1. 다음 명령을 실행하여 기본 편집기에서 **HyperConverged** CR을 편집합니다.

```
$ oc edit hyperconverged kubevirt-hyperconverged -n openshift-cnv
```

2. 적절한 장치의 **pciDeviceSelector**,**resourceName** 및 **externalResourceProvider** (해당되는 경우) 필드를 삭제하여 **spec.permittedHostDevices.pciHostDevices** 어레이에서 PCI 장치 정보를 제거합니다. 이 예에서는 **intel.com/qat** 리소스가 삭제되었습니다.

   설정 파일 예

```
apiVersion: hco.kubevirt.io/v1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  permittedHostDevices:
    pciHostDevices:
    - pciDeviceSelector: "10DE:1DB6"
      resourceName: "nvidia.com/GV100GL_Tesla_V100"
    - pciDeviceSelector: "10DE:1EB8"
      resourceName: "nvidia.com/TU104GL_Tesla_T4"
...
```

3. 변경 사항을 저장하고 편집기를 종료합니다.

검증

- 다음 명령을 실행하여 PCI 호스트 장치가 노드에서 제거되었는지 확인합니다. 예제 출력에서는 **intel.com/qat** 리소스 이름과 연결된 장치가 0개 있음을 보여줍니다.

```
$ oc describe node <node_name>
```

출력 예

```
Capacity:
  cpu:                      64
  devices.kubevirt.io/kvm:      110
  devices.kubevirt.io/tun:      110
  devices.kubevirt.io/vhost-net:  110
  ephemeral-storage:            915128Mi
  hugepages-1Gi:                0
  hugepages-2Mi:                0
  memory:                     131395264Ki
  nvidia.com/GV100GL_Tesla_V100   1
  nvidia.com/TU104GL_Tesla_T4     1
  intel.com/qat:                0
  pods:                       250
Allocatable:
  cpu:                      63500m
  devices.kubevirt.io/kvm:      110
  devices.kubevirt.io/tun:      110
  devices.kubevirt.io/vhost-net:  110
  ephemeral-storage:            863623130526
  hugepages-1Gi:                0
  hugepages-2Mi:                0
```

```
memory:                    130244288Ki
nvidia.com/GV100GL_Tesla_V100   1
nvidia.com/TU104GL_Tesla_T4     1
intel.com/qat:              0
pods:                       250
```

8.15.10.2. PCI 패스스루의 가상 머신 구성

PCI 장치를 클러스터에 추가하고 나면 가상 머신에 할당할 수 있습니다. 이제 PCI 장치를 가상 머신에 물리적으로 연결된 것처럼 사용할 수 있습니다.

8.15.10.2.1. 가상 머신에 PCI 장치 할당

PCI 장치를 클러스터에서 사용할 수 있는 경우 가상 머신에 할당하고 PCI 패스스루를 활성화할 수 있습니다.

절차

- 가상 시스템에 PCI 장치를 호스트 장치로 할당합니다.

  예

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  domain:
    devices:
      hostDevices:
      - deviceName: nvidia.com/TU104GL_Tesla_T4 1
        name: hostdevices1
```

1 호스트 장치로 클러스터에서 허용되는 PCI 장치의 이름입니다. 가상 시스템은 이 호스트 장치에 액세스할 수 있습니다.

검증

- 다음 명령을 사용하여 가상 시스템에서 호스트 장치를 사용할 수 있는지 확인합니다.

```
$ lspci -nnk | grep NVIDIA
```

  출력 예

```
$ 02:01.0 3D controller [0302]: NVIDIA Corporation GV100GL [Tesla V100 PCIe 32GB]
[10de:1eb8] (rev a1)
```

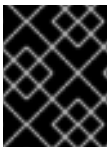
8.15.10.3. 추가 리소스

- BIOS에서 Intel VT-X 및 AMD-V 가상화 하드웨어 확장 활성화
- 파일 권한 관리
- 설치 후 시스템 구성 작업

8.15.11. vGPU 패스스루 구성

가상 머신에서 가상 GPU(vGPU) 하드웨어에 액세스할 수 있습니다. 가상 머신에 vGPU를 할당하면 다음을 수행할 수 있습니다.

- 기본 하드웨어 GPU의 일부에 액세스하여 가상 머신에서 높은 성능의 이점을 얻을 수 있습니다.
- 리소스 집약적인 I/O 작업을 간소화합니다.



중요

vGPU 패스스루는 베어 메탈 환경에서 실행되는 클러스터에 연결된 장치에만 할당할 수 있습니다.

8.15.11.1. 가상 머신에 vGPU 패스스루 장치 할당

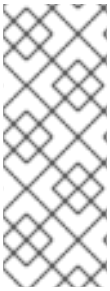
OpenShift Container Platform 웹 콘솔을 사용하여 vGPU 패스스루 장치를 가상 머신에 할당합니다.

사전 요구 사항

- 가상 머신을 중지해야 합니다.

절차

1. OpenShift Container Platform 웹 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines를 클릭합니다.
2. 장치를 할당할 가상 머신을 선택합니다.
3. 세부 정보 탭에서 GPU 장치를 클릭합니다.
vGPU 장치를 호스트 장치로 추가하는 경우 VNC 콘솔을 사용하여 장치에 액세스할 수 없습니다.
4. GPU 장치 추가를 클릭하고 Name 을 입력하고 장치 이름 목록에서 장치를 선택합니다.
5. 저장을 클릭합니다.
6. YAML 탭을 클릭하여 새 장치가 **hostDevices** 섹션의 클러스터 구성에 추가되었는지 확인합니다.



참고

사용자 지정 템플릿 또는 YAML 파일에서 생성한 가상 머신에 하드웨어 장치를 추가할 수 있습니다. Windows 10 또는 RHEL 7과 같은 특정 운영 체제에 대해 사전 제공되는 부팅 소스 템플릿에 장치를 추가할 수 없습니다.

클러스터에 연결된 리소스를 표시하려면 사이드 메뉴에서 컴퓨팅 → 하드웨어 장치를 클릭합니다.

8.15.11.2. 추가 리소스

- 가상 머신 생성
- 가상 머신 템플릿 생성

8.15.12. 중재 장치 구성

OpenShift Virtualization은 **HyperConverged** CR(사용자 정의 리소스)에 장치 목록을 제공하는 경우 가상 GPU(vGPU)와 같은 중재 장치를 자동으로 생성합니다.

중요

중재된 장치의 선언적 구성은 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.

8.15.12.1. NVIDIA GPU Operator 사용 정보

NVIDIA GPU Operator는 OpenShift Container Platform 클러스터에서 NVIDIA GPU 리소스를 관리하고 부트스트랩 GPU 노드와 관련된 작업을 자동화합니다. GPU는 클러스터의 특수 리소스이므로 애플리케이션 워크로드를 GPU에 배포하기 전에 일부 구성 요소를 설치해야 합니다. 이러한 구성 요소에는 컴퓨팅 통합 장치 아키텍처(ECDHEDA), Kubernetes 장치 플러그인, 컨테이너 런타임 등을 활성화하는 NVIDIA 드라이버(자동 노드 레이블링, 모니터링 등)가 포함됩니다.

참고

NVIDIA GPU Operator는 NVIDIA에서만 지원됩니다. NVIDIA에서 지원을 얻는 방법에 대한 자세한 내용은 NVIDIA에서 지원을 참조하십시오.

OpenShift Container Platform OpenShift Virtualization에서 GPU를 활성화하는 방법에는 여기에 설명된 OpenShift Container Platform 네이티브 방법과 NVIDIA GPU Operator를 사용하는 두 가지 방법이 있습니다.

NVIDIA GPU Operator는 OpenShift Container Platform OpenShift Virtualization이 OpenShift Container Platform에서 실행되는 가상화된 워크로드에 GPU를 노출할 수 있는 Kubernetes Operator입니다. 사용자는 GPU 지원 가상 머신을 쉽게 프로비저닝 및 관리할 수 있으므로 다른 워크로드와 동일한 플랫폼에서 복잡한 AI/머신 학습(AI/ML) 워크로드를 실행할 수 있습니다. 또한 인프라의 GPU 용량을 쉽게 확장할 수 있어 GPU 기반 워크로드를 빠르게 확장할 수 있습니다.

NVIDIA GPU Operator를 사용하여 GPU 가속 VM을 실행하기 위한 작업자 노드를 프로비저닝하는 방법에 대한 자세한 내용은 OpenShift Virtualization을 사용하는 NVIDIA GPU Operator를 참조하십시오.

8.15.12.2. OpenShift Virtualization에서 가상 GPU 사용 정보

일부 그래픽 처리 장치(GPU) 카드는 가상 GPU(vGPU) 생성을 지원합니다. 관리자가 **HyperConverged** CR(사용자 정의 리소스)에서 구성 세부 정보를 제공하는 경우 OpenShift Virtualization은 vGPU 및 기타 중재 장치를 자동으로 생성할 수 있습니다. 이 자동화는 대규모 클러스터에 특히 유용합니다.

참고

기능 및 지원 세부 사항은 하드웨어 공급 업체의 설명서를 참조하십시오.

중재 장치

하나 이상의 가상 장치로 구분된 물리적 장치입니다. vGPU는 중개 장치(mdev) 유형입니다. 물리 GPU의 성능은 가상 장치로 나뉩니다. 하나 이상의 가상 머신(VM)에 중재된 장치를 할당할 수 있지만 게스트 수는 GPU와 호환 가능해야 합니다. 일부 GPU는 여러 게스트를 지원하지 않습니다.

8.15.12.2.1. 사전 요구 사항

- 하드웨어 벤더가 드라이버를 제공하는 경우 중재 장치를 생성하려는 노드에 설치합니다.
  - NVIDIA 카드를 사용하는 경우 NVIDIA GRID 드라이버를 설치했습니다.

8.15.12.2.2. 구성 개요

중재된 장치를 구성할 때 관리자는 다음 작업을 완료해야 합니다.

- 중재 장치를 만듭니다.
- 중재된 장치를 클러스터에 노출합니다.

HyperConverged CR에는 두 작업을 모두 수행하는 API가 포함되어 있습니다.

중재 장치 생성

```
...
spec:
  mediatedDevicesConfiguration:
    mediatedDevicesTypes: 1
    - <device_type>
    nodeMediatedDeviceTypes: 2
    - mediatedDevicesTypes: 3
      - <device_type>
      nodeSelector: 4
        <node_selector_key>: <node_selector_value>
...
```

1. 필수: 클러스터의 글로벌 설정을 구성합니다.
2. 선택 사항: 특정 노드 또는 노드 그룹에 대한 글로벌 구성을 재정의합니다. 글로벌 **mediatedDevicesTypes** 구성과 함께 사용해야 합니다.
3. **nodeMediatedDeviceTypes** 를 사용하는 경우 필수 항목입니다. 지정된 노드의 글로벌 **mediatedDevicesTypes** 구성을 덮어씁니다.
4. **nodeMediatedDeviceTypes** 를 사용하는 경우 필수 항목입니다.키:값 쌍을 포함해야 합니다.

클러스터에 중재된 장치 노출

```
...
  permittedHostDevices:
    mediatedDevices:
    - mdevNameSelector: GRID T4-2Q 1
      resourceName: nvidia.com/GRID_T4-2Q 2
...
```

1 호스트의 이 값에 매핑되는 중재된 장치를 노출합니다.

참고

/sys/bus/pci/devices/<slot>:<domain>.<function>/mdev_types/<type>/name 의 내용을 확인하여 장치가 지원하는 중재 장치 유형을 확인할 수 있습니다.

예를 들어 **nvidia-231** 유형의 이름 파일에는 선택기 문자열**GRID T4-2Q** 가 포함되어 있습니다. **GRID T4-2Q** 를 **mdevNameSelector** 값으로 사용하면 노드가**nvidia-231** 유형을 사용할 수 있습니다.

2 **resourceName** 은 노드에 할당된 것과 일치해야 합니다. 다음 명령을 사용하여**resourceName** 을 찾습니다.

```
$ oc get $NODE -o json \
  | jq '.status.allocatable \
    | with_entries(select(.key | startswith("nvidia.com/"))) \
    | with_entries(select(.value != "0"))'
```

8.15.12.2.3. vGPU가 노드에 할당되는 방법

각 물리적 장치에 대해 OpenShift Virtualization은 다음 값을 구성합니다.

- 단일 mdev 유형.
- 선택한 **mdev** 유형의 최대 인스턴스 수입니다.

클러스터 아키텍처는 장치를 생성하고 노드에 할당하는 방법에 영향을 미칩니다.

노드당 여러 카드가 있는 대규모 클러스터

유사한 vGPU 유형을 지원할 수 있는 카드가 여러 개 있는 노드에서는 라운드 로빈 방식으로 관련 장치 유형이 생성됩니다. 예를 들면 다음과 같습니다.

```
...
mediatedDevicesConfiguration:
  mediatedDevicesTypes:
  - nvidia-222
  - nvidia-228
  - nvidia-105
  - nvidia-108
...
```

이 시나리오에서 각 노드에는 다음 vGPU 유형을 지원하는 두 개의 카드가 있습니다.

```
nvidia-105
...
nvidia-108
nvidia-217
nvidia-299
...
```

각 노드에서 OpenShift Virtualization은 다음과 같은 vGPU를 생성합니다.

- 첫 번째 카드에 nvidia-105 유형의 vGPU입니다.
- 2 두 번째 카드에 nvidia-108 유형의 vGPU.

하나의 노드에는 하나 이상의 요청된 vGPU 유형을 지원하는 단일 카드가 있습니다.

OpenShift Virtualization은 **mediatedDevicesTypes** 목록에서 처음 제공되는 지원되는 유형을 사용합니다.
예를 들어 노드 카드의 카드는 **nvidia-223** 및 **nvidia-224** 를 지원합니다. 다음 **mediatedDevicesTypes** 목록이 구성되어 있습니다.

```
...
mediatedDevicesConfiguration:
  mediatedDevicesTypes:
  - nvidia-22
  - nvidia-223
  - nvidia-224
...
```

이 예에서 OpenShift Virtualization에서는 **nvidia-223** 유형을 사용합니다.

8.15.12.2.4. 중재 장치의 변경 및 제거 정보

클러스터의 중재 장치 구성은 다음을 통해 OpenShift Virtualization으로 업데이트할 수 있습니다.

- **HyperConverged** CR을 편집하고 **mediatedDevicesTypes** 스탠자의 내용을 변경합니다.
- node **MediatedDeviceTypes** 노드 선택기와 일치하는 노드 레이블 변경
- **HyperConverged** CR의 **spec.mediatedDevicesConfiguration** 및 **spec.permittedHostDevices** 스탠자에서 장치 정보를 제거합니다.

참고

spec.permittedHostDevices 스탠자에서 **spec.mediatedDevicesConfiguration** 스탠자를 제거하지 않고 장치 정보를 제거하면 동일한 노드에 새로운 중재 장치 유형을 생성할 수 없습니다. 중재된 장치를 올바르게 제거하려면 두 스탠자 모두에서 장치 정보를 제거합니다.

특정 변경에 따라 이러한 작업으로 인해 OpenShift Virtualization이 중재된 장치를 재구성하거나 클러스터 노드에서 해당 장치를 제거합니다.

8.15.12.2.5. 중재된 장치에 대한 호스트 준비

중재 장치를 구성하려면 먼저 IOMMU(Input-Output Memory Management Unit) 드라이버를 활성화해야 합니다.

8.15.12.2.5.1. IOMMU 드라이버를 활성화하려면 커널 인수 추가

커널에서 IOMMU(Input-Output Memory Management Unit) 드라이버를 활성화하려면 **MachineConfig** 개체를 생성하고 커널 인수를 추가합니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 관리자 권한을 보유하고 있어야 합니다.
- Intel 또는 AMD CPU 하드웨어.
- BIOS의 Directed I/O 확장용 Intel Virtualization Technology 또는 AMD IOMMU(Basic Input/Output System)가 활성화되어 있습니다.

절차

1. 커널 인수를 식별하는 **MachineConfig** 오브젝트를 만듭니다. 다음 예제에서는 Intel CPU에 대한 커널 인수를 보여줍니다.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker 1
  name: 100-worker-iommu 2
spec:
  config:
    ignition:
      version: 3.2.0
  kernelArguments:
      - intel_iommu=on 3
...
```

1. 새 커널 인수를 작업자 노드에만 적용합니다.
2. **name**은 머신 구성 및 그 용도 중 이 커널 인수의 순위(100)를 나타냅니다. AMD CPU가 있는 경우 커널 인수를 **amd_iommu=on**으로 지정합니다.
3. Intel CPU의 커널 인수를 **intel_iommu**로 식별합니다.

2. 새 **MachineConfig** 오브젝트를 만듭니다.

```
$ oc create -f 100-worker-kernel-arg-iommu.yaml
```

검증

- 새 **MachineConfig** 오브젝트가 추가되었는지 확인합니다.

```
$ oc get MachineConfig
```

8.15.12.2.6. 중재 장치 추가 및 제거

중재된 장치를 추가하거나 제거할 수 있습니다.

8.15.12.2.6.1. 중재 장치 생성 및 노출

HyperConverged CR(사용자 정의 리소스)을 편집하여 가상 GPU(vGPU)와 같은 중재 장치를 노출하고 생성할 수 있습니다.

사전 요구 사항

- IOMMU(Input-Output Memory Management Unit) 드라이버를 활성화했습니다.

절차

1. 다음 명령을 실행하여 기본 편집기에서 **HyperConverged** CR을 편집합니다.

```
$ oc edit hyperconverged kubevirt-hyperconverged -n openshift-cnv
```

2. **HyperConverged** CR사양에 중재 장치 정보를 추가하여**mediatedDevicesConfiguration** 및 **permittedHostDevices** 스탠자를 포함하도록 합니다. 예를 들면 다음과 같습니다.

   설정 파일 예

```
apiVersion: hco.kubevirt.io/v1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  mediatedDevicesConfiguration: <.>
    mediatedDevicesTypes: <.>
    - nvidia-231
    nodeMediatedDeviceTypes: <.>
    - mediatedDevicesTypes: <.>
      - nvidia-233
      nodeSelector:
        kubernetes.io/hostname: node-11.redhat.com
  permittedHostDevices: <.>
    mediatedDevices:
    - mdevNameSelector: GRID T4-2Q
      resourceName: nvidia.com/GRID_T4-2Q
    - mdevNameSelector: GRID T4-8Q
      resourceName: nvidia.com/GRID_T4-8Q
...
```

   <.>는 미디어 장치를 생성합니다. <.> Required: Global **mediatedDevicesTypes** configuration. <.> 선택 사항: 특정 노드의 글로벌 구성을 재정의합니다. <.> **nodeMediatedDeviceTypes** 를 사용하는 경우 필요합니다. <.> 클러스터에 미디어 지정된 장치를 노출합니다.

3. 변경 사항을 저장하고 편집기를 종료합니다.

검증

- 다음 명령을 실행하여 장치가 특정 노드에 추가되었는지 확인할 수 있습니다.

```
$ oc describe node <node_name>
```

8.15.12.2.6.2. CLI를 사용하여 클러스터에서 중재된 장치 제거

클러스터에서 중재 장치를 제거하려면 **HyperConverged** CR(사용자 정의 리소스)에서 해당 장치의 정보를 삭제합니다.

절차

1. 다음 명령을 실행하여 기본 편집기에서 **HyperConverged** CR을 편집합니다.

```
$ oc edit hyperconverged kubevirt-hyperconverged -n openshift-cnv
```

2. **HyperConverged** CR의 **spec.mediatedDevicesConfiguration** 및 **spec.permittedHostDevices** 스탠자에서 장치 정보를 제거합니다. 두 항목을 모두 제거하면 나중에 동일한 노드에서 새 중재 장치 유형을 만들 수 있습니다. 예를 들면 다음과 같습니다.

   설정 파일 예

```
apiVersion: hco.kubevirt.io/v1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  mediatedDevicesConfiguration:
    mediatedDevicesTypes: 1
      - nvidia-231
  permittedHostDevices:
    mediatedDevices: 2
    - mdevNameSelector: GRID T4-2Q
      resourceName: nvidia.com/GRID_T4-2Q
```

   1 **nvidia-231** 장치 유형을 제거하려면 **mediatedDevicesTypes** 배열에서 삭제합니다.

   2 **GRID T4-2Q** 장치를 제거하려면 **mdevNameSelector** 필드와 해당 **resourceName** 필드를 삭제합니다.

3. 변경 사항을 저장하고 편집기를 종료합니다.

8.15.12.3. 중재 장치 사용

vGPU는 중재된 장치 유형입니다. 물리적 GPU의 성능은 가상 장치로 나뉩니다. 중재 장치를 하나 이상의 가상 머신에 할당할 수 있습니다.

8.15.12.3.1. 가상 머신에 중재 장치 할당

가상 GPU(vGPU)와 같은 중재 장치를 가상 머신에 할당합니다.

사전 요구 사항

- 중재 장치는 **HyperConverged** 사용자 정의 리소스에 구성되어 있습니다.

절차

- **VirtualMachine** 매니페스트의 **spec.domain.devices.gpus** 스탠자를 편집하여 VM(가상 머신)에 중재된 장치를 할당합니다.

  가상 머신 매니페스트 예

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
```

```
spec:
  domain:
    devices:
      gpus:
      - deviceName: nvidia.com/TU104GL_Tesla_T4 1
        name: gpu1 2
      - deviceName: nvidia.com/GRID_T4-1Q
        name: gpu2
```

1 중재된 장치와 관련된 리소스 이름입니다.

2 VM에서 장치를 식별하는 이름입니다.

검증

- 가상 머신에서 장치를 사용할 수 있는지 확인하려면 **VirtualMachine** 매니페스트의 **deviceName** 값을 사용하여 < **device_name** >을 대체하여 다음 명령을 실행합니다.

```
$ lspci -nnk | grep <device_name>
```

8.15.12.4. 추가 리소스

- BIOS에서 Intel VT-X 및 AMD-V 가상화 하드웨어 확장 활성화

8.15.13. 워치독 구성

워치독 장치에 대해 VM(가상 머신)을 구성하고, 워치독을 설치한 후 워치독 서비스를 시작하여 워치독을 노출합니다.

8.15.13.1. 사전 요구 사항

- 가상 머신에는 **i6300esb** 워치독 장치에 대한 커널 지원이 있어야 합니다. RHEL(Red Hat Enterprise Linux) 이미지는 **i6300esb**를 지원합니다.

8.15.13.2. 워치독 장치 정의

운영 체제(OS)가 더 이상 응답하지 않을 때 워치독이 진행되는 방식을 정의합니다.

표 8.4. 사용 가능한 작업

poweroff	VM(가상 시스템)의 전원이 즉시 꺼집니다. **spec.running**이 **true**로 설정되었거나 **spec.runStrategy**가 **manual**로 설정되지 않은 경우 VM이 재부팅됩니다.
reset	VM이 재부팅되고 게스트 OS가 반응할 수 없습니다. 게스트 OS가 재부팅하는 데 필요한 시간은 활성 프로브가 시간 초과될 수 있으므로 이 옵션을 사용하지 않습니다. 클러스터 수준 보호에서 활성 프로브가 실패하고 강제로 다시 예약하는 경우 이 시간 초과로 VM을 재부팅하는 시간을 연장할 수 있습니다.
shutdown	VM은 모든 서비스를 중지하여 정상적으로 전원을 끕니다.

절차

1. 다음 콘텐츠를 사용하여 YAML 파일을 생성합니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: vm2-rhel84-watchdog
  name: <vm-name>
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/vm: vm2-rhel84-watchdog
    spec:
      domain:
        devices:
          watchdog:
            name: <watchdog>
            i6300esb:
              action: "poweroff" (1)
...
```

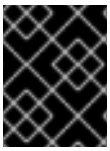
(1) watchdog 작업 (poweroff, reset, 또는 shutdown)을 지정합니다.

위의 예제에서는 poweroff 작업을 사용하여 RHEL8 VM에서 i6300esb 워치독 장치를 구성하고 장치를 /dev/watchdog로 노출합니다.

이제 워치독 바이너리에서 이 장치를 사용할 수 있습니다.

2. 다음 명령을 실행하여 클러스터에 YAML 파일을 적용합니다.

```
$ oc apply -f <file_name>.yaml
```



중요

이 절차는 워치독 기능을 테스트하는 데만 제공되며 프로덕션 시스템에서 실행해서는 안 됩니다.

1. 다음 명령을 실행하여 VM이 워치독 장치에 연결되어 있는지 확인합니다.

```
$ lspci | grep watchdog -i
```

2. 다음 명령 중 하나를 실행하여 워치독이 활성 상태인지 확인합니다.

- 커널 패닉을 트리거합니다.

```
# echo c > /proc/sysrq-trigger
```

- 워치독 서비스를 종료합니다.

```
# pkill -9 watchdog
```

8.15.13.3. 워치독 장치 설치

가상 머신에 **watchdog** 패키지를 설치하고 워치독 서비스를 시작합니다.

절차

1. root 사용자로 **watchdog** 패키지 및 종속성을 설치합니다.

```
# yum install watchdog
```

2. **/etc/watchdog.conf** 파일에서 다음 행의 주석을 제거한 후 변경 사항을 저장합니다.

```
#watchdog-device = /dev/watchdog
```

3. 워치독 서비스가 부팅 시 시작되도록 활성화합니다.

```
# systemctl enable --now watchdog.service
```

8.15.13.4. 추가 리소스

- 상태 점검을 사용하여 애플리케이션 상태 모니터링

8.15.14. 사전 정의된 부팅 소스 자동 가져오기 및 업데이트

시스템 정의 및 OpenShift Virtualization 또는 *사용자가 생성하는 사용자* 정의에 포함된 부팅 소스를 사용할 수 있습니다. 시스템 정의 부팅 소스 가져오기 및 업데이트는 제품 기능 게이트에서 제어합니다. 기능 게이트를 사용하여 업데이트를 활성화, 비활성화 또는 다시 활성화할 수 있습니다. 사용자 정의 부팅 소스는 제품 기능 게이트에 의해 제어되지 않으며 자동 가져오기 및 업데이트를 옵트인하거나 사용하지 않도록 개별적으로 관리해야 합니다.

중요

부팅 소스를 자동으로 가져오고 업데이트하려면 기본 스토리지 클래스를 설정해야 합니다.

8.15.14.1. 자동 부팅 소스 업데이트 활성화

OpenShift Virtualization 4.9의 사전 정의된 부팅 소스가 있는 경우 자동 부팅 소스 업데이트를 선택해야 합니다. OpenShift Virtualization 4.10 이후의 모든 사전 정의된 부팅 소스는 기본적으로 자동으로 업데이트됩니다.

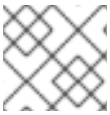
절차

- 다음 명령을 사용하여 **dataImportCron** 레이블을 데이터 소스에 적용합니다.

```
$ oc label --overwrite DataSource rhel8 -n openshift-virtualization-os-images
cdi.kubevirt.io/dataImportCron=true
```

8.15.14.2. 자동 부팅 소스 업데이트 비활성화

연결이 끊긴 환경에서 로그 수를 줄이거나 사전 정의된 부팅 소스의 자동 가져오기 및 업데이트를 비활성화하여 리소스 사용량을 줄일 수 있습니다. **HyperConverged** CR(사용자 정의 리소스)에서 **spec.featureGates.enableCommonBootImageImport** 필드를 **false** 로 설정합니다.



참고

사용자 정의 부팅 소스는 이 설정의 영향을 받지 않습니다.

절차

- 다음 명령을 사용하여 자동 업데이트를 비활성화합니다.

```
$ oc patch hco kubevirt-hyperconverged -n openshift-cnv --type json -p '[{"op":
"replace", "path": "/spec/featureGates/enableCommonBootImageImport", "value":
false}]'
```

8.15.14.3. 자동 부팅 소스 업데이트 다시 활성화

이전에 자동 부팅 소스 업데이트를 비활성화한 경우 해당 기능을 수동으로 다시 활성화해야 합니다. **HyperConverged** CR(사용자 정의 리소스)에서 **spec.featureGates.enableCommonBootImageImport** 필드를 **true** 로 설정합니다.

절차

- 다음 명령을 사용하여 자동 업데이트를 다시 활성화합니다.

```
$ oc patch hco kubevirt-hyperconverged -n openshift-cnv --type json -p '[{"op":
"replace", "path": "/spec/featureGates/enableCommonBootImageImport", "value":
true}]'
```

8.15.14.4. 사용자 정의 부팅 소스에서 자동 업데이트 활성화

OpenShift Virtualization은 기본적으로 사전 정의된 부팅 소스를 자동으로 업데이트하지만 사용자 정의 부팅 소스를 자동으로 업데이트하지는 않습니다. **HyperConverged** CR(사용자 정의 리소스)을 편집하여 사용자 정의 부팅 소스에서 자동 가져오기 및 업데이트를 활성화해야 합니다.

절차

1. 다음 명령을 사용하여 편집을 위해 **HyperConverged** CR을 엽니다.

```
$ oc edit -n openshift-cnv HyperConverged
```

2. **HyperConverged** CR을 편집하여 적절한 템플릿과**dataImportCronTemplates** 섹션에 부팅 소스를 지정합니다. 예를 들면 다음과 같습니다.

CentOS 7의 예

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
spec:
  dataImportCronTemplates:
```

```
- metadata:
    name: centos7-image-cron
    annotations:
      cdi.kubevirt.io/storage.bind.immediate.requested: "true" (1)
  spec:
    schedule: "0 */12 * * *" (2)
    template:
      spec:
        source:
          registry: (3)
            url: docker://quay.io/containerdisks/centos:7-2009
        storage:
          resources:
            requests:
              storage: 10Gi
    managedDataSource: centos7 (4)
    retentionPolicy: "None" (5)
```

(1) 이 주석은 **volumeBindingMode** 가 **WaitForFirstConsumer** 로 설정된 스토리지 클래스에 필요합니다.

(2) cron 형식으로 지정된 작업의 스케줄입니다.

(3) 레지스트리 소스에서 데이터 볼륨을 생성하려면 을 사용합니다. 노드 docker 캐시를 기반으로 하는 노드 **pullMethod** 가 아닌 기본 **Pod pullMethod** 를 사용합니다. 노드 Docker 캐시는 **Container.Image** 를 통해 레지스트리 이미지를 사용할 수 있지만 CDI 가져오기에 액세스할 수 있는 권한이 없는 경우 유용합니다.

(4) 사용자 지정 이미지가 사용 가능한 부팅 소스로 감지되려면 이미지의 **managedDataSource** 의 이름이 VM 템플릿 YAML 파일의 **spec.dataVolumeTemplates.spec.sourceRef.name** 에 있는 템플릿의 **DataSource** 이름과 일치해야 합니다.

(5) cron 작업이 삭제될 때 모두 데이터 볼륨 및 데이터 소스를 유지합니다. cron 작업이 삭제될 때 데이터 볼륨 및 데이터 소스를 삭제하려면 **None** 을 사용합니다.

8.15.15. 가상 머신에서 Descheduler 제거 활성화

Descheduler를 사용하여 Pod를 더 적절한 노드에 다시 예약할 수 있도록 Pod를 제거할 수 있습니다. Pod가 가상 머신인 경우 Pod를 제거하면 가상 머신이 다른 노드로 실시간 마이그레이션됩니다.

중요

가상 머신에 대한 Descheduler 제거는 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.

8.15.15.1. Descheduler 프로필

기술 프리뷰 **DevPreviewLongLifecycle** 프로필을 사용하여 가상 머신에서 Descheduler를 활성화합니다. 이는 현재 OpenShift Virtualization에서 사용할 수 있는 유일한 Descheduler 프로필입니다. 적절한 예약을 보장하기 위해 예상되는 로드에 대한 CPU 및 메모리 요청이 있는 VM을 생성합니다.

DevPreviewLongLifecycle

이 프로필은 노드 간 리소스 사용량의 균형을 조정하고 다음 전략을 활성화합니다.

- **RemovePodsHavingTooManyRestarts:** 컨테이너가 너무 여러 번 다시 시작된 Pod와 모든 컨테이너에서 다시 시작하는 합계(Init Container 포함)가 100 이상인 Pod를 제거합니다. VM 게스트 운영 체제를 다시 시작해도 이 수는 늘어나지 않습니다.
- **LowNodeUtilization:** 활용도가 낮은 노드가 있는 경우 활용도가 높은 노드에서 Pod를 제거합니다. 제거된 Pod의 대상 노드는 스케줄러에 따라 결정됩니다.
  - 모든 임계값(CPU, 메모리, Pod 수)에서 사용량이 20% 미만인 경우 노드는 활용도가 낮은 것으로 간주됩니다.
  - 모든 임계값(CPU, 메모리, Pod 수)에서 사용량이 50%를 초과하면 노드는 과도하게 사용되는 것으로 간주됩니다.

8.15.15.2. Descheduler 설치

Descheduler는 기본적으로 사용할 수 없습니다. Descheduler를 활성화하려면 OperatorHub에서 Kube Descheduler Operator를 설치하고 Descheduler 프로필을 한 개 이상 활성화해야 합니다.

사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스합니다.

프로세스

1. OpenShift Container Platform 웹 콘솔에 로그인합니다.
2. Kube Descheduler Operator에 필요한 네임스페이스를 생성합니다.
   a. 관리 → 네임스페이스로 이동하여 네임스페이스 생성을 클릭합니다.
   b. 이름 필드에 **openshift-kube-descheduler-operator** 를 입력하고 Labels 필드에 **openshift.io/cluster-monitoring=true** 를 입력하여 Descheduler 지표를 활성화한 후생성 을 클릭합니다.
3. Kube Descheduler Operator를 설치합니다.
   a. Operators → OperatorHub로 이동합니다.
   b. 필터 박스에 Kube Descheduler Operator를 입력합니다.
   c. Kube Descheduler Operator를 선택하고 설치를 클릭합니다.
   d. Operator 설치 페이지에서 클러스터의 특정 네임스페이스를 선택합니다. 드롭다운 메뉴에서 openshift-kube-descheduler-operator를 선택합니다.
   e. 업데이트 채널 및 승인 전략 값을 원하는 값으로 조정합니다.

f. 설치를 클릭합니다.

4. Descheduler 인스턴스를 생성합니다.

a. Operator → 설치된 Operator 페이지에서 Kube Descheduler Operator를 클릭합니다.

b. Kube Descheduler 탭을 선택하고 KubeDescheduler 생성을 클릭합니다.

c. 필요에 따라 설정을 편집합니다.

i. Profiles 섹션을 확장하고 **DevPreviewLongLifecycle** 를 선택합니다. **AffinityAndTaints** 프로파일은 기본적으로 활성화되어 있습니다.

중요

현재 OpenShift Virtualization에서 사용할 수 있는 유일한 프로필은 **DevPreviewLongLifecycle** 입니다.

나중에 OpenShift CLI(**oc**)를 사용하여 Descheduler의 프로필 및 설정을 구성할 수도 있습니다.

8.15.15.3. VM(가상 머신)에서 Descheduler 제거 활성화

Descheduler가 설치되면 **VirtualMachine** CR(사용자 정의 리소스)에 주석을 추가하여 VM에서 Descheduler 제거를 활성화할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 웹 콘솔 또는 OpenShift CLI(**oc**)에 Descheduler를 설치합니다.
- VM이 실행되고 있지 않은지 확인합니다.

절차

1. VM을 시작하기 전에 **Descheduler.alpha.kubernetes.io/evict** 주석을 **VirtualMachine** CR에 추가합니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  template:
    metadata:
      annotations:
        descheduler.alpha.kubernetes.io/evict: "true"
```

2. 설치 중에 웹 콘솔에서 **DevPreviewLongLifecycle** 프로필을 아직 설정하지 않은 경우 **KubeDescheduler** 오브젝트의 **spec.profile** 섹션에 **DevPreviewLongLifecycle** 를 지정합니다.

```
apiVersion: operator.openshift.io/v1
kind: KubeDescheduler
metadata:
  name: cluster
  namespace: openshift-kube-descheduler-operator
spec:
```

```
deschedulingIntervalSeconds: 3600
profiles:
- DevPreviewLongLifecycle
```

이제 VM에서 Descheduler가 활성화됩니다.

8.15.15.4. 추가 리소스

- Descheduler를 사용하여 Pod 제거

8.16. 가상 머신 가져오기

8.16.1. 데이터 볼륨 가져오기에 필요한 TLS 인증서

8.16.1.1. 데이터 볼륨 가져오기 인증을 위한 TLS 인증서 추가

레지스트리 또는 HTTPS에서 데이터를 가져오려면 이러한 소스 끝점에 대한 TLS 인증서를 구성 맵에 추가해야 합니다. 이 구성 맵은 대상 데이터 볼륨의 네임스페이스에 있어야 합니다.

TLS 인증서의 상대 파일 경로를 참조하여 구성 맵을 만듭니다.

절차

1. 올바른 네임스페이스에 있는지 확인합니다. 구성 맵은 동일한 네임스페이스에 있는 경우에만 데이터 볼륨에서 참조할 수 있습니다.

```
$ oc get ns
```

2. config map을 생성합니다.

```
$ oc create configmap <configmap-name> --from-file=</path/to/file/ca.pem>
```

8.16.1.2. 예: TLS 인증서에서 생성한 구성 맵

다음은 **ca.pem** TLS 인증서에서 생성한 구성 맵의 예입니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: tls-certs
data:
  ca.pem: |
    -----BEGIN CERTIFICATE-----
    ... <base64 encoded cert> ...
    -----END CERTIFICATE-----
```

8.16.2. 데이터 볼륨을 사용하여 가상 머신 이미지 가져오기

데이터 볼륨을 사용하여 가상 머신 이미지를 PVC(영구 볼륨 클레임)로 가져오려면 CDI(Containerized Data Importer)를 사용합니다. 영구 저장을 위해 데이터 볼륨을 가상 머신에 연결할 수 있습니다.

가상 머신 이미지는 HTTP 또는 HTTPS 끝점에서 호스팅하거나, 컨테이너 디스크에 빌드하고 컨테이너 레지스트리에 저장할 수 있습니다.

중요

디스크 이미지를 PVC로 가져오면 PVC에 요청한 전체 스토리지 용량을 사용하도록 디스크 이미지가 확장됩니다. 이 공간을 사용하기 위해 가상 머신의 디스크 파티션 및 파일 시스템을 확장해야 할 수 있습니다.

크기 조정 절차는 가상 머신에 설치된 운영 체제에 따라 다릅니다. 자세한 내용은 운영 체제 설명서를 참조하십시오.

8.16.2.1. 사전 요구 사항

- 끝점에 TLS 인증서가 필요한 경우 인증서가 데이터 볼륨과 동일한 네임스페이스의 구성 맵에 포함되어 있고 데이터 볼륨 구성에 참조되어 있어야 합니다.
- 컨테이너 디스크를 가져오려면 다음을 수행합니다.
  - 가져오기 전에 가상 머신 이미지에서 컨테이너 디스크를 준비하여 컨테이너 레지스트리에 저장해야 할 수 있습니다.
  - 컨테이너 레지스트리에 TLS가 없는 경우 컨테이너 디스크를 가져오기 전에 **HyperConverged** 사용자 지정 리소스의 **insecureRegistries** 필드에 레지스트리를 추가해야 합니다.
- 이 작업을 완료하려면 스토리지 클래스를 정의하거나 CDI 스크래치 공간을 준비해야 할 수 있습니다.

8.16.2.2. CDI 지원 작업 매트릭스

이 매트릭스에는 끝점에 대한 콘텐츠 유형에 따라 지원되는 CDI 작업과 이러한 작업 중 스크래치 공간이 필요한 작업이 표시되어 있습니다.

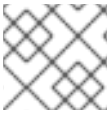
콘텐츠 유형	HTTP	HTTPS	HTTP 기본 인증	레지스트리	업로드
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* □ GZ □ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt(RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* □ GZ □ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 지원되는 작업

□ 지원되지 않는 작업

* 스크래치 공간 필요

** 사용자 정의 인증 기관이 필요한 경우 스크래치 공간 필요



참고

이제 CDI에서 OpenShift Container Platform 클러스터 전체 프록시 구성을 사용합니다.

8.16.2.3. 데이터 볼륨 정보

DataVolume 오브젝트는 CDI(Containerized Data Importer) 프로젝트에서 제공하는 사용자 정의 리소스입니다. 데이터 볼륨은 기본 PVC(영구 볼륨 클레임)와 관련된 가져오기, 복제, 업로드 작업을 오케스트레이션합니다. 데이터 볼륨은 OpenShift Virtualization과 통합되며 PVC가 준비되기 전에 가상 머신이 시작되지 않도록 합니다.

8.16.2.4. 데이터 볼륨을 사용하여 가상 머신 이미지 가져오기

데이터 볼륨을 사용하여 가상 머신 이미지를 스토리지로 가져올 수 있습니다.

가상 머신 이미지는 HTTP 또는 HTTPS 끝점에서 호스팅하거나 이미지를 컨테이너 디스크에 빌드하고 컨테이너 레지스트리에 저장할 수 있습니다.

VirtualMachine 구성 파일에 이미지의 데이터 소스를 지정합니다. 가상 머신이 생성되면 가상 머신 이미지가 있는 데이터 볼륨을 스토리지로 가져옵니다.

사전 요구 사항

- 가상 머신 이미지를 가져오려면 다음이 있어야 합니다.
  - RAW, ISO 또는 QCOW2 형식의 가상 머신 디스크 이미지(필요한 경우 **xz** 또는 **gz**를 사용하여 압축)
  - 데이터 소스에 액세스하는 데 필요한 인증 자격 증명과 함께 이미지가 호스팅되는 HTTP 또는 HTTPS 끝점.
- 컨테이너 디스크를 가져오려면 데이터 소스에 액세스하는 데 필요한 인증 자격 증명과 함께 컨테이너 디스크에 빌드하고 컨테이너 레지스트리에 저장된 가상 머신 이미지가 있어야 합니다.
- 가상 머신이 자체 서명된 인증서 또는 시스템 CA 번들에서 서명하지 않은 인증서를 사용하는 서버와 통신해야 하는 경우 데이터 볼륨과 동일한 네임스페이스에 구성 맵을 생성해야 합니다.

절차

1. 데이터 소스에 인증이 필요한 경우 데이터 소스 자격 증명을 지정하여 **Secret** 매니페스트를 생성하고 **endpoint-secret.yaml** 로 저장합니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: endpoint-secret (1)
  labels:
    app: containerized-data-importer
type: Opaque
data:
  accessKeyId: "" (2)
  secretKey:   "" (3)
```

(1) 보안의 이름을 지정합니다.

2 base64로 인코딩된 키 ID 또는 사용자 이름을 지정합니다.

3 base64로 인코딩된 시크릿 키 또는 암호를 지정합니다.

2. 보안 매니페스트 를 적용합니다.

```
$ oc apply -f endpoint-secret.yaml
```

3. 가져올 가상 머신 이미지의 데이터 소스를 지정하여 **VirtualMachine** 매니페스트를 편집하여 **vm-fedora-datavolume.yaml** 로 저장합니다.

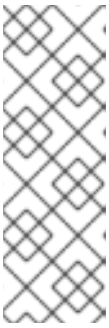
```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  creationTimestamp: null
  labels:
    kubevirt.io/vm: vm-fedora-datavolume
  name: vm-fedora-datavolume 1
spec:
  dataVolumeTemplates:
  - metadata:
      creationTimestamp: null
      name: fedora-dv 2
    spec:
      storage:
        resources:
          requests:
            storage: 10Gi
        storageClassName: local
      source:
        http: 3
          url: "https://mirror.arizona.edu/fedora/linux/releases/35/Cloud/x86_64/images/Fedora-
Cloud-Base-35-1.2.x86_64.qcow2" 4
          secretRef: endpoint-secret 5
          certConfigMap: "" 6
    status: {}
  running: true
  template:
    metadata:
      creationTimestamp: null
      labels:
        kubevirt.io/vm: vm-fedora-datavolume
    spec:
      domain:
        devices:
          disks:
          - disk:
              bus: virtio
            name: datavolumedisk1
        machine:
          type: ""
        resources:
          requests:
            memory: 1.5Gi
```

```
    terminationGracePeriodSeconds: 180
    volumes:
    - dataVolume:
        name: fedora-dv
      name: datavolumedisk1
status: {}
```

1. 가상 머신의 이름을 지정합니다.
2. 데이터 볼륨의 이름을 지정합니다.
3. HTTP 또는 HTTPS 끝점에 대해 **http** 를 지정합니다. 레지스트리에서 가져온 컨테이너 디스크 이미지의 레지스트리를 지정합니다.
4. 가져올 가상 머신 이미지의 URL 또는 레지스트리 끝점을 지정합니다. 이 예에서는 HTTPS 끝점에서 가상 머신 이미지를 참조합니다. 컨테이너 레지스트리 끝점은 예를 들면 **url: "docker://kubevirt/fedora-cloud-container-disk-demo:latest"**와 같습니다.
5. 데이터 소스에 대한 보안을 생성한 경우 시크릿 이름을 지정합니다.
6. 선택 사항: CA 인증서 구성 맵을 지정합니다.

4. 가상 머신을 생성합니다.

```
$ oc create -f vm-fedora-datavolume.yaml
```



참고

oc create 명령은 데이터 볼륨과 가상 머신을 생성합니다. CDI 컨트롤러는 올바른 주석을 사용하여 기본 PVC를 생성하고 가져오기 프로세스가 시작됩니다. 가져오기가 완료되면 데이터 볼륨 상태가 성공으로 변경됩니다. 가상 머신을 시작할 수 있습니다.

데이터 볼륨 프로비저닝은 백그라운드에서 이루어지므로 프로세스를 모니터링할 필요가 없습니다.

검증

1. 가져오기 Pod는 지정된 URL에서 가상 머신 이미지 또는 컨테이너 디스크를 다운로드하여 프로비저닝된 PV에 저장합니다. 다음 명령을 실행하여 가져오기 Pod의 상태를 확인합니다.

```
$ oc get pods
```

2. 다음 명령을 실행하여 상태가 **Succeeded** 될 때까지 데이터 볼륨을 모니터링합니다.

```
$ oc describe dv fedora-dv 1
```

1. **VirtualMachine** 매니페스트에 정의된 데이터 볼륨 이름을 지정합니다.

3. 프로비저닝이 완료되었고 가상 머신이 직렬 콘솔에 액세스하여 시작되었는지 확인합니다.

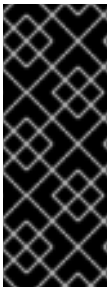
```
$ virtctl console vm-fedora-datavolume
```

8.16.2.5. 추가 리소스

- 데이터 볼륨 작업에 대한 쓰기 성능을 향상하기 위해 사전 할당 모드를 구성합니다.

8.16.3. 데이터 볼륨을 사용하여 블록 스토리지로 가상 머신 이미지 가져오기

기존 가상 머신 이미지를 OpenShift Container Platform 클러스터로 가져올 수 있습니다. OpenShift Virtualization은 데이터 볼륨을 사용하여 데이터 가져오기와 기본 PVC(영구 볼륨 클레임) 생성을 자동화합니다.



중요

디스크 이미지를 PVC로 가져오면 PVC에 요청한 전체 스토리지 용량을 사용하도록 디스크 이미지가 확장됩니다. 이 공간을 사용하기 위해 가상 머신의 디스크 파티션 및 파일 시스템을 확장해야 할 수 있습니다.

크기 조정 절차는 가상 머신에 설치된 운영 체제에 따라 다릅니다. 자세한 내용은 운영 체제 설명서를 참조하십시오.

8.16.3.1. 사전 요구 사항

- CDI 지원 작업 매트릭스에 따라 스크래치 공간이 필요한 경우, 이 작업을 완료하기 위해서는 먼저 스토리지 클래스를 정의하거나 CDI 스크래치 공간을 준비해야 합니다.

8.16.3.2. 데이터 볼륨 정보

Dataolume 오브젝트는 CDI(Containerized Data Importer) 프로젝트에서 제공하는 사용자 정의 리소스입니다. 데이터 볼륨은 기본 PVC(영구 볼륨 클레임)와 관련된 가져오기, 복제, 업로드 작업을 오케스트레이션합니다. 데이터 볼륨은 OpenShift Virtualization과 통합되며 PVC가 준비되기 전에 가상 머신이 시작되지 않도록 합니다.

8.16.3.3. 블록 영구 볼륨 정보

PV(블록 영구 볼륨)는 원시 블록 장치에서 지원하는 PV입니다. 이러한 볼륨은 파일 시스템이 없으며 오버헤드를 줄여 가상 머신의 성능을 향상시킬 수 있습니다.

원시 블록 볼륨은 PV 및 PVC(영구 볼륨 클레임) 사양에 **volumeMode:Block**을 지정하여 프로비저닝합니다.

8.16.3.4. 로컬 블록 영구 볼륨 생성

파일을 채우고 루프 장치로 마운트하여 노드에 로컬 블록 PV(영구 볼륨)를 생성합니다. 그런 다음 PV 매니페스트에서 이 루프 장치를 **Block** 볼륨으로 참조하고 가상 머신 이미지의 블록 장치로 사용할 수 있습니다.

절차

1. 로컬 PV를 생성할 노드에 **root**로 로그인합니다. 이 절차에서는 예제로 **node01**을 사용합니다.
2. 블록 장치로 사용할 수 있도록 파일을 생성하고 null 문자로 채웁니다. 다음 예제에서는 크기가 2Gb(20X100Mb 블록)인 파일 **loop10**을 생성합니다.

```
$ dd if=/dev/zero of=<loop10> bs=100M count=20
```

3. **loop10** 파일을 루프 장치로 마운트합니다.

```
$ losetup </dev/loop10>d3 <loop10> 1 2
```

1. 루프 장치가 마운트된 파일 경로입니다.
2. 이전 단계에서 생성된 파일은 루프 장치로 마운트됩니다.

4. 마운트된 루프 장치를 참조하는 **PersistentVolume** 매니페스트를 생성합니다.

```
kind: PersistentVolume
apiVersion: v1
metadata:
 name: <local-block-pv10>
 annotations:
spec:
 local:
  path: </dev/loop10> 1
 capacity:
  storage: <2Gi>
 volumeMode: Block 2
 storageClassName: local 3
 accessModes:
  - ReadWriteOnce
 persistentVolumeReclaimPolicy: Delete
 nodeAffinity:
  required:
   nodeSelectorTerms:
   - matchExpressions:
    - key: kubernetes.io/hostname
      operator: In
      values:
      - <node01> 4
```

1. 노드에 있는 루프 장치의 경로입니다.
2. 블록 PV임을 나타냅니다.
3. 선택 사항: PV의 스토리지 클래스를 설정합니다. 생략하면 클러스터 기본값이 사용됩니다.
4. 블록 장치가 마운트된 노드입니다.

5. 블록 PV를 생성합니다.

```
# oc create -f <local-block-pv10.yaml> 1
```

1. 이전 단계에서 생성한 영구 볼륨의 파일 이름입니다.

8.16.3.5. 데이터 볼륨을 사용하여 블록 스토리지로 가상 머신 이미지 가져오기

데이터 볼륨을 사용하여 가상 머신 이미지를 블록 스토리지로 가져올 수 있습니다. 가상 머신을 생성하기 전에 **VirtualMachine** 매니페스트에서 데이터 볼륨을 참조합니다.

사전 요구 사항

- RAW, ISO 또는 QCOW2 형식의 가상 머신 디스크 이미지(필요한 경우 **xz** 또는 **gz**를 사용하여 압축)
- 데이터 소스에 액세스하는 데 필요한 인증 자격 증명과 함께 이미지가 호스팅되는 HTTP 또는 HTTPS 끝점.

절차

1. 데이터 소스에 인증이 필요한 경우 데이터 소스 자격 증명을 지정하여 **Secret** 매니페스트를 생성하고 **endpoint-secret.yaml** 로 저장합니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: endpoint-secret 1
  labels:
    app: containerized-data-importer
type: Opaque
data:
  accessKeyId: "" 2
  secretKey:   "" 3
```

1. 보안의 이름을 지정합니다.
2. base64로 인코딩된 키 ID 또는 사용자 이름을 지정합니다.
3. base64로 인코딩된 시크릿 키 또는 암호를 지정합니다.

2. 보안 매니페스트 를 적용합니다.

```
$ oc apply -f endpoint-secret.yaml
```

3. 가상 머신 이미지의 데이터 소스 및 **storage.volumeMode** 용 **Block** 을 지정하여 **DataVolume** 매니페스트를 생성합니다.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: import-pv-datavolume 1
spec:
  storageClassName: local 2
    source:
      http:
        url:
"https://mirror.arizona.edu/fedora/linux/releases/35/Cloud/x86_64/images/Fedora-Cloud-Base-35-1.2.x86_64.qcow2" 3
        secretRef: endpoint-secret 4
  storage:
    volumeMode: Block 5
```

```
  resources:
    requests:
      storage: 10Gi
```

1. 데이터 볼륨의 이름을 지정합니다.
2. 선택 사항: 스토리지 클래스를 설정하거나 클러스터 기본값을 승인하도록 생략합니다.
3. 가져올 이미지의 HTTP 또는 HTTPS URL을 지정합니다.
4. 데이터 소스에 대한 보안을 생성한 경우 시크릿 이름을 지정합니다.
5. 알려진 스토리지 프로비저너에 대해 볼륨 모드 및 액세스 모드가 자동으로 감지됩니다. 그렇지 않으면 **Block** 을 지정합니다.

4. 가상 머신 이미지를 가져올 데이터 볼륨을 생성합니다.

```
$ oc create -f import-pv-datavolume.yaml
```

가상 머신을 생성하기 전에 **VirtualMachine** 매니페스트에서 이 데이터 볼륨을 참조할 수 있습니다.

8.16.3.6. CDI 지원 작업 매트릭스

이 매트릭스에는 끝점에 대한 콘텐츠 유형에 따라 지원되는 CDI 작업과 이러한 작업 중 스크래치 공간이 필요한 작업이 표시되어 있습니다.

콘텐츠 유형	HTTP	HTTPS	HTTP 기본 인증	레지스트리	업로드
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt(RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 지원되는 작업

☐ 지원되지 않는 작업

* 스크래치 공간 필요

** 사용자 정의 인증 기관이 필요한 경우 스크래치 공간 필요

참고

이제 CDI에서 OpenShift Container Platform 클러스터 전체 프록시 구성을 사용합니다.

8.16.3.7. 추가 리소스

- 데이터 볼륨 작업에 대한 쓰기 성능을 향상하기 위해 사전 할당 모드를 구성합니다.

8.17. 가상 머신 복제

8.17.1. 네임스페이스 간에 데이터 볼륨을 복제할 수 있는 사용자 권한 활성화

네임스페이스의 격리 특성으로 인해 기본적으로 사용자는 다른 네임스페이스에 리소스를 복제할 수 없습니다.

사용자가 가상 머신을 다른 네임스페이스에 복제할 수 있도록 하려면 **cluster-admin** 역할의 사용자가 새 클러스터 역할을 만들어야 합니다. 이 클러스터 역할을 사용자에게 바인딩하면 사용자가 가상 머신을 대상 네임스페이스에 복제할 수 있습니다.

8.17.1.1. 사전 요구 사항

- cluster-admin 역할의 사용자만 클러스터 역할을 생성할 수 있습니다.

8.17.1.2. 데이터 볼륨 정보

Dataolume 오브젝트는 CDI(Containerized Data Importer) 프로젝트에서 제공하는 사용자 정의 리소스입니다. 데이터 볼륨은 기본 PVC(영구 볼륨 클레임)와 관련된 가져오기, 복제, 업로드 작업을 오케스트레이션합니다. 데이터 볼륨은 OpenShift Virtualization과 통합되며 PVC가 준비되기 전에 가상 머신이 시작되지 않도록 합니다.

8.17.1.3. 데이터 볼륨 복제를 위한 RBAC 리소스 생성

datavolumes 리소스에 대한 모든 작업 권한을 활성화하는 새 클러스터 역할을 만듭니다.

절차

1. **ClusterRole** 매니페스트를 만듭니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <datavolume-cloner> 1
rules:
- apiGroups: ["cdi.kubevirt.io"]
  resources: ["datavolumes/source"]
  verbs: ["*"]
```

1 클러스터 역할의 고유 이름입니다.

2. 클러스터에 클러스터 역할을 만듭니다.

```
$ oc create -f <datavolume-cloner.yaml> 1
```

1 이전 단계에서 만든 **ClusterRole** 매니페스트 파일 이름입니다.

3. 소스 및 대상 네임스페이스 모두에 적용되고 이전 단계에서 만든 클러스터 역할을 참조하는 **RoleBinding** 매니페스트를 만듭니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
```

```
metadata:
  name: <allow-clone-to-user> 1
  namespace: <Source namespace> 2
subjects:
- kind: ServiceAccount
  name: default
  namespace: <Destination namespace> 3
roleRef:
  kind: ClusterRole
  name: datavolume-cloner 4
  apiGroup: rbac.authorization.k8s.io
```

1. 역할 바인딩의 고유 이름입니다.
2. 소스 데이터 볼륨의 네임스페이스입니다.
3. 데이터 볼륨이 복제되는 네임스페이스입니다.
4. 이전 단계에서 만든 클러스터 역할의 이름입니다.

4. 클러스터에 역할 바인딩을 만듭니다.

```
$ oc create -f <datavolume-cloner.yaml> 1
```

1. 이전 단계에서 만든 **RoleBinding** 매니페스트 파일 이름입니다.

8.17.2. 가상 머신 디스크를 새 데이터 볼륨으로 복제

데이터 볼륨 구성 파일에서 소스 PVC(영구 볼륨 클레임)를 참조하여 가상 머신 디스크의 PVC를 새 데이터 볼륨으로 복제할 수 있습니다.

> **주의**
>
> **volumeMode: Block**이 있는 PV(영구 볼륨)에서 **volumeMode: Filesystem**인 PV로 복제하는 등 다양한 볼륨 모드 간 작업 복제가 지원됩니다.
>
> 그러나 **contentType: kubevirt**인 경우에만 다양한 볼륨 모드 간에 복제할 수 있습니다.

작은 정보

사전 할당을 활성화하거나 단일 데이터 볼륨에 대해 복제 중에 디스크 공간을 사전 할당하는 경우 CDI(Containerized Data Importer)가 디스크 공간을 사전 할당합니다. 사전 할당을 통해 쓰기 성능이 향상됩니다. 자세한 내용은 데이터 볼륨에 대한 사전 할당 사용을 참조하십시오.

8.17.2.1. 사전 요구 사항

- 가상 머신 디스크의 PVC를 다른 네임스페이스로 복제하려면 추가 권한이 필요합니다.

8.17.2.2. 데이터 볼륨 정보

Dataolume 오브젝트는 CDI(Containerized Data Importer) 프로젝트에서 제공하는 사용자 정의 리소스입니다. 데이터 볼륨은 기본 PVC(영구 볼륨 클레임)와 관련된 가져오기, 복제, 업로드 작업을 오케스트레이션합니다. 데이터 볼륨은 OpenShift Virtualization과 통합되며 PVC가 준비되기 전에 가상 머신이 시작되지 않도록 합니다.

8.17.2.3. 가상 머신 디스크의 영구 볼륨 클레임을 새 데이터 볼륨으로 복제

기존 가상 머신 디스크의 PVC(영구 볼륨 클레임)를 새 데이터 볼륨으로 복제할 수 있습니다. 그러면 새 데이터 볼륨을 새 가상 머신에 사용할 수 있습니다.

참고

데이터 볼륨이 가상 머신과 독립적으로 생성되는 경우 데이터 볼륨의 라이프사이클은 가상 머신과 독립적입니다. 가상 머신이 삭제되어도 데이터 볼륨이나 연결된 PVC가 삭제되지 않습니다.

사전 요구 사항

- 사용할 기존 가상 머신 디스크의 PVC를 결정합니다. PVC와 연결된 가상 머신의 전원을 꺼야 복제할 수 있습니다.
- OpenShift CLI(oc)를 설치합니다.

절차

1. 복제하려는 가상 머신 디스크를 검사하여 연결된 PVC의 이름과 네임스페이스를 확인합니다.
2. 데이터 볼륨에 대해 새 데이터 볼륨의 이름, 소스 PVC의 이름과 네임스페이스, 새 데이터 볼륨의 크기를 지정하는 YAML 파일을 생성합니다.
예를 들면 다음과 같습니다.

```yaml
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <cloner-datavolume> 1
spec:
  source:
    pvc:
      namespace: "<source-namespace>" 2
      name: "<my-favorite-vm-disk>" 3
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: <2Gi> 4
```

1. 새 데이터 볼륨의 이름입니다.
2. 소스 PVC가 존재하는 네임스페이스입니다.
3. 소스 PVC의 이름입니다.

4 새 데이터 볼륨의 크기입니다. 충분한 공간을 할당해야 합니다. 그러지 않으면 복제 작업이 실패합니다. 크기는 소스 PVC와 같거나 커야 합니다.

3. 데이터 볼륨을 생성하여 PVC 복제를 시작합니다.

```
$ oc create -f <cloner-datavolume>.yaml
```

참고

데이터 볼륨이 있으면 PVC가 준비될 때까지 가상 머신이 시작되지 않으므로 PVC가 복제되는 동안 새 데이터 볼륨을 참조하는 가상 머신을 생성할 수 있습니다.

8.17.2.4. CDI 지원 작업 매트릭스

이 매트릭스에는 끝점에 대한 콘텐츠 유형에 따라 지원되는 CDI 작업과 이러한 작업 중 스크래치 공간이 필요한 작업이 표시되어 있습니다.

콘텐츠 유형	HTTP	HTTPS	HTTP 기본 인증	레지스트리	업로드
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt(RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 지원되는 작업

☐ 지원되지 않는 작업

* 스크래치 공간 필요

** 사용자 정의 인증 기관이 필요한 경우 스크래치 공간 필요

8.17.3. 데이터 볼륨 템플릿을 사용하여 가상 머신 복제

기존 VM의 PVC(영구 볼륨 클레임)를 복제하여 새 가상 머신을 생성할 수 있습니다. 가상 머신 구성 파일에 **dataVolumeTemplate**을 포함하여 원래 PVC에서 새 데이터 볼륨을 생성합니다.

주의

volumeMode: Block이 있는 PV(영구 볼륨)에서 **volumeMode: Filesystem**인 PV로 복제하는 등 다양한 볼륨 모드 간 작업 복제가 지원됩니다.

그러나 **contentType: kubevirt**인 경우에만 다양한 볼륨 모드 간에 복제할 수 있습니다.

작은 정보

사전 할당을 활성화하거나 단일 데이터 볼륨에 대해 복제 중에 디스크 공간을 사전 할당하는 경우 CDI(Containerized Data Importer)가 디스크 공간을 사전 할당합니다. 사전 할당을 통해 쓰기 성능이 향상됩니다. 자세한 내용은 데이터 볼륨에 대한 사전 할당 사용을 참조하십시오.

8.17.3.1. 사전 요구 사항

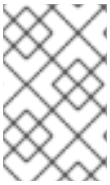
- 가상 머신 디스크의 PVC를 다른 네임스페이스로 복제하려면 추가 권한이 필요합니다.

8.17.3.2. 데이터 볼륨 정보

Dataolume 오브젝트는 CDI(Containerized Data Importer) 프로젝트에서 제공하는 사용자 정의 리소스입니다. 데이터 볼륨은 기본 PVC(영구 볼륨 클레임)와 관련된 가져오기, 복제, 업로드 작업을 오케스트레이션합니다. 데이터 볼륨은 OpenShift Virtualization과 통합되며 PVC가 준비되기 전에 가상 머신이 시작되지 않도록 합니다.

8.17.3.3. 데이터 볼륨 템플릿을 사용하여 복제된 영구 볼륨 클레임에서 새 가상 머신 생성

기존 가상 머신의 PVC(영구 볼륨 클레임)를 데이터 볼륨에 복제하는 가상 머신을 생성할 수 있습니다. 가상 머신 매니페스트에서 **dataVolumeTemplate**을 참조하면 **source** PVC가 데이터 볼륨에 복제되어 가상 머신 생성에 자동으로 사용됩니다.



참고

데이터 볼륨이 가상 머신의 데이터 볼륨 템플릿의 일부로 생성되면 데이터 볼륨의 라이프사이클이 가상 머신에 따라 달라집니다. 가상 머신이 삭제되면 데이터 볼륨 및 연결된 PVC도 삭제됩니다.

사전 요구 사항

- 사용할 기존 가상 머신 디스크의 PVC를 결정합니다. PVC와 연결된 가상 머신의 전원을 꺼야 복제할 수 있습니다.
- OpenShift CLI(**oc**)를 설치합니다.

절차

1. 복제하려는 가상 머신을 검사하여 연결된 PVC의 이름과 네임스페이스를 확인합니다.

2. **VirtualMachine** 오브젝트에 대한 YAML 파일을 만듭니다. 다음 가상 머신 예제에서는**source-namespace** 네임스페이스에 있는**my-favorite-vm-disk**를 복제합니다. **favorite-clone**이라는 **2Gi** 데이터 볼륨이 **my-favorite-vm-disk**에서 생성됩니다.
예를 들면 다음과 같습니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: vm-dv-clone
  name: vm-dv-clone (1)
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/vm: vm-dv-clone
    spec:
      domain:
        devices:
          disks:
          - disk:
              bus: virtio
            name: root-disk
        resources:
          requests:
            memory: 64M
      volumes:
      - dataVolume:
          name: favorite-clone
        name: root-disk
  dataVolumeTemplates:
  - metadata:
      name: favorite-clone
    spec:
      storage:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 2Gi
      source:
        pvc:
          namespace: "source-namespace"
          name: "my-favorite-vm-disk"
```

(1) 생성할 가상 머신입니다.

3. PVC 복제 데이터 볼륨으로 가상 머신을 생성합니다.

```
$ oc create -f <vm-clone-datavolumetemplate>.yaml
```

8.17.3.4. CDI 지원 작업 매트릭스

이 매트릭스에는 끝점에 대한 콘텐츠 유형에 따라 지원되는 CDI 작업과 이러한 작업 중 스크래치 공간이 필요한 작업이 표시되어 있습니다.

콘텐츠 유형	HTTP	HTTPS	HTTP 기본 인증	레지스트리	업로드
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* □ GZ □ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt(RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* □ GZ □ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 지원되는 작업

□ 지원되지 않는 작업

* 스크래치 공간 필요

** 사용자 정의 인증 기관이 필요한 경우 스크래치 공간 필요

8.17.4. 가상 머신 디스크를 새 블록 스토리지 데이터 볼륨에 복제

데이터 볼륨 구성 파일의 소스 PVC(영구 볼륨 클레임)를 참조하여 가상 머신 디스크의 PVC를 새 블록 데이터 볼륨에 복제할 수 있습니다.

> 주의
>
> **volumeMode: Block**이 있는 PV(영구 볼륨)에서 **volumeMode: Filesystem**인 PV로 복제하는 등 다양한 볼륨 모드 간 작업 복제가 지원됩니다.
>
> 그러나 **contentType: kubevirt**인 경우에만 다양한 볼륨 모드 간에 복제할 수 있습니다.

작은 정보

사전 할당을 활성화하거나 단일 데이터 볼륨에 대해 복제 중에 디스크 공간을 사전 할당하는 경우 CDI(Containerized Data Importer)가 디스크 공간을 사전 할당합니다. 사전 할당을 통해 쓰기 성능이 향상됩니다. 자세한 내용은 데이터 볼륨에 대한 사전 할당 사용을 참조하십시오.

8.17.4.1. 사전 요구 사항

- 가상 머신 디스크의 PVC를 다른 네임스페이스로 복제하려면 추가 권한이 필요합니다.

8.17.4.2. 데이터 볼륨 정보

Dataolume 오브젝트는 CDI(Containerized Data Importer) 프로젝트에서 제공하는 사용자 정의 리소스입

니다. 데이터 볼륨은 기본 PVC(영구 볼륨 클레임)와 관련된 가져오기, 복제, 업로드 작업을 오케스트레이션합니다. 데이터 볼륨은 OpenShift Virtualization과 통합되며 PVC가 준비되기 전에 가상 머신이 시작되지 않도록 합니다.

8.17.4.3. 블록 영구 볼륨 정보

PV(블록 영구 볼륨)는 원시 블록 장치에서 지원하는 PV입니다. 이러한 볼륨은 파일 시스템이 없으며 오버헤드를 줄여 가상 머신의 성능을 향상시킬 수 있습니다.

원시 블록 볼륨은 PV 및 PVC(영구 볼륨 클레임) 사양에 **volumeMode:Block**을 지정하여 프로비저닝합니다.

8.17.4.4. 로컬 블록 영구 볼륨 생성

파일을 채우고 루프 장치로 마운트하여 노드에 로컬 블록 PV(영구 볼륨)를 생성합니다. 그런 다음 PV 매니페스트에서 이 루프 장치를 **Block** 볼륨으로 참조하고 가상 머신 이미지의 블록 장치로 사용할 수 있습니다.

절차

1. 로컬 PV를 생성할 노드에 **root**로 로그인합니다. 이 절차에서는 예제로**node01**을 사용합니다.
2. 블록 장치로 사용할 수 있도록 파일을 생성하고 null 문자로 채웁니다. 다음 예제에서는 크기가 2Gb(20X100Mb 블록)인 파일 **loop10**을 생성합니다.

```
$ dd if=/dev/zero of=<loop10> bs=100M count=20
```

3. **loop10** 파일을 루프 장치로 마운트합니다.

```
$ losetup </dev/loop10>d3 <loop10> 1 2
```

1. 루프 장치가 마운트된 파일 경로입니다.
2. 이전 단계에서 생성된 파일은 루프 장치로 마운트됩니다.

4. 마운트된 루프 장치를 참조하는 **PersistentVolume** 매니페스트를 생성합니다.

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: <local-block-pv10>
  annotations:
spec:
  local:
    path: </dev/loop10> 1
  capacity:
    storage: <2Gi>
  volumeMode: Block 2
  storageClassName: local 3
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  nodeAffinity:
    required:
```

```
    nodeSelectorTerms:
    - matchExpressions:
      - key: kubernetes.io/hostname
        operator: In
        values:
        - <node01> 4
```

1. 노드에 있는 루프 장치의 경로입니다.
2. 블록 PV임을 나타냅니다.
3. 선택 사항: PV의 스토리지 클래스를 설정합니다. 생략하면 클러스터 기본값이 사용됩니다.
4. 블록 장치가 마운트된 노드입니다.

5. 블록 PV를 생성합니다.

```
# oc create -f <local-block-pv10.yaml> 1
```

1. 이전 단계에서 생성한 영구 볼륨의 파일 이름입니다.

8.17.4.5. 가상 머신 디스크의 영구 볼륨 클레임을 새 데이터 볼륨으로 복제

기존 가상 머신 디스크의 PVC(영구 볼륨 클레임)를 새 데이터 볼륨으로 복제할 수 있습니다. 그러면 새 데이터 볼륨을 새 가상 머신에 사용할 수 있습니다.

참고

데이터 볼륨이 가상 머신과 독립적으로 생성되는 경우 데이터 볼륨의 라이프사이클은 가상 머신과 독립적입니다. 가상 머신이 삭제되어도 데이터 볼륨이나 연결된 PVC가 삭제되지 않습니다.

사전 요구 사항

- 사용할 기존 가상 머신 디스크의 PVC를 결정합니다. PVC와 연결된 가상 머신의 전원을 꺼야 복제할 수 있습니다.
- OpenShift CLI(oc)를 설치합니다.
- 소스 PVC와 크기가 같거나 더 큰 블록 PV(영구 볼륨)가 한 개 이상 사용 가능합니다.

절차

1. 복제하려는 가상 머신 디스크를 검사하여 연결된 PVC의 이름과 네임스페이스를 확인합니다.
2. 데이터 볼륨에 대해 새 데이터 볼륨의 이름, 소스 PVC의 이름과 네임스페이스, 사용 가능한 블록 PV를 사용하도록 하는 **volumeMode: Block**, 새 데이터 볼륨의 크기를 지정하는 YAML 파일을 생성합니다.
   예를 들면 다음과 같습니다.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
```

```
  name: <cloner-datavolume> 1
spec:
  source:
    pvc:
      namespace: "<source-namespace>" 2
      name: "<my-favorite-vm-disk>" 3
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: <2Gi> 4
    volumeMode: Block 5
```

1. 새 데이터 볼륨의 이름입니다.
2. 소스 PVC가 존재하는 네임스페이스입니다.
3. 소스 PVC의 이름입니다.
4. 새 데이터 볼륨의 크기입니다. 충분한 공간을 할당해야 합니다. 그러지 않으면 복제 작업이 실패합니다. 크기는 소스 PVC와 같거나 커야 합니다.
5. 대상이 블록 PV임을 나타냅니다.

3. 데이터 볼륨을 생성하여 PVC 복제를 시작합니다.

```
$ oc create -f <cloner-datavolume>.yaml
```

참고

데이터 볼륨이 있으면 PVC가 준비될 때까지 가상 머신이 시작되지 않으므로 PVC가 복제되는 동안 새 데이터 볼륨을 참조하는 가상 머신을 생성할 수 있습니다.

8.17.4.6. CDI 지원 작업 매트릭스

이 매트릭스에는 끝점에 대한 콘텐츠 유형에 따라 지원되는 CDI 작업과 이러한 작업 중 스크래치 공간이 필요한 작업이 표시되어 있습니다.

콘텐츠 유형	HTTP	HTTPS	HTTP 기본 인증	레지스트리	업로드
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt(RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 지원되는 작업

□ 지원되지 않는 작업

* 스크래치 공간 필요

** 사용자 정의 인증 기관이 필요한 경우 스크래치 공간 필요

8.18. 가상 머신 네트워킹

8.18.1. 기본 Pod 네트워크에 대한 가상 머신 구성

masquerade 바인딩 모드를 사용하도록 네트워크 인터페이스를 구성하여 가상 머신을 기본 내부 Pod 네트워크에 연결할 수 있습니다.

참고

기본 Pod 네트워크에 연결된 가상 네트워크 인터페이스 카드(vNIC)의 트래픽은 실시간 마이그레이션 중에 중단됩니다.

8.18.1.1. 명령줄에서 가상 모드 구성

가상 모드를 사용하여 Pod IP 주소를 통해 나가는 가상 머신의 트래픽을 숨길 수 있습니다. 가상 모드에서는 NAT(Network Address Translation)를 사용하여 가상 머신을 Linux 브리지를 통해 Pod 네트워크 백엔드에 연결합니다.

가상 머신 구성 파일을 편집하여 가상 모드를 사용하도록 설정하고 트래픽이 가상 머신에 유입되도록 허용하십시오.

사전 요구 사항

- 가상 머신은 DHCP를 사용하여 IPv4 주소를 가져오도록 구성해야 합니다. 아래 예제는 DHCP를 사용하도록 구성되어 있습니다.

절차

1. 가상 머신 구성 파일의 **interfaces** 스펙을 편집합니다.

```
kind: VirtualMachine
spec:
  domain:
    devices:
      interfaces:
        - name: default
          masquerade: {} 1
          ports: 2
            - port: 80
  networks:
  - name: default
    pod: {}
```

1. 가상 모드를 사용하여 연결합니다.
2. 선택 사항: 각각 **port** 필드에 지정된 가상 머신에서 노출하려는 포트를 나열합니다.**port** 값은 0에서 65536 사이의 숫자여야 합니다. **ports** 어레이를 사용하지 않으면 유효한 범위의 모든 포트가 들어오는 트래픽에 열려 있습니다. 이 예에서 들어오는 트래픽은 포트 **80** 에서 허용됩

니다.

참고

포트 49152 및 49153은 libvirt 플랫폼에서 사용하도록 예약되며 이러한 포트로 들어오는 다른 모든 트래픽은 삭제됩니다.

2. 가상 머신을 생성합니다.

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

8.18.1.2. 듀얼 스택(IPv4 및 IPv6)을 사용하여 가상 모드 구성

cloud-init를 사용하여 기본 Pod 네트워크에서 IPv6 및 IPv4를 모두 사용하도록 새 VM(가상 머신)을 구성할 수 있습니다.

가상 머신 인스턴스 구성의 **Network.pod.vmIPv6NetworkCIDR** 필드는 VM의 정적 IPv6 주소와 게이트웨이 IP 주소를 결정합니다. 이는 virt-launcher Pod에서 IPv6 트래픽을 가상 머신으로 라우팅하는 데 사용되며 외부적으로 사용되지 않습니다. **Network.pod.vmIPv6NetworkCIDR** 필드는 CIDR(Classless Inter-Domain Routing) 표기법으로 IPv6 주소 블록을 지정합니다. 기본값은 **fd10:0:2::2/120** 입니다. 네트워크 요구 사항에 따라 이 값을 편집할 수 있습니다.

가상 시스템이 실행 중이면 가상 시스템의 들어오고 나가는 트래픽이 virt-launcher Pod의 IPv4 주소와 고유한 IPv6 주소로 라우팅됩니다. 그런 다음 virt-launcher Pod는 IPv4 트래픽을 가상 시스템의 DHCP 주소로 라우팅하고 IPv6 트래픽을 가상 시스템의 IPv6 주소로 정적으로 설정합니다.

사전 요구 사항

- OpenShift Container Platform 클러스터는 듀얼 스택용으로 구성된 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자를 사용해야 합니다.

절차

1. 새 가상 시스템 구성에서 **masquerade**가 있는 인터페이스를 포함하고 cloud-init를 사용하여 IPv6 주소 및 기본 게이트웨이를 구성합니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: example-vm-ipv6
...
        interfaces:
          - name: default
            masquerade: {} 1
            ports:
              - port: 80 2
      networks:
      - name: default
        pod: {}
      volumes:
      - cloudInitNoCloud:
          networkData: |
            version: 2
```

```
      ethernets:
        eth0:
          dhcp4: true
          addresses: [ fd10:0:2::2/120 ] 3
          gateway6: fd10:0:2::1 4
```

1. 가상 모드를 사용하여 연결합니다.
2. 포트 80에서 가상 머신으로 들어오는 트래픽을 허용합니다.
3. 가상 머신 인스턴스 구성의 **Network.pod.vmIPv6NetworkCIDR** 필드에 따라 결정되는 정적 IPv6 주소입니다. 기본값은 **fd10:0:2::2/120** 입니다.
4. 가상 머신 인스턴스 구성의 **Network.pod.vmIPv6NetworkCIDR** 필드에 정의된 게이트웨이 IP 주소입니다. 기본값은 **fd10:0:2::1** 입니다.

2. 네임스페이스에서 가상 머신을 생성합니다.

```
$ oc create -f example-vm-ipv6.yaml
```

검증

- IPv6가 구성되었는지 확인하려면 가상 시스템을 시작하고 가상 시스템 인스턴스의 인터페이스 상태를 확인하여 IPv6 주소가 있는지 확인합니다.

```
$ oc get vmi <vmi-name> -o jsonpath="{.status.interfaces[*].ipAddresses}"
```

8.18.2. 가상 머신 노출 서비스 생성

Service 오브젝트를 사용하여 클러스터 내에서 또는 클러스터 외부에 가상 머신을 노출할 수 있습니다.

8.18.2.1. 서비스 정보

Kubernetes *서비스*는 포드 집합에서 실행되는 애플리케이션을 네트워크 서비스로 노출하는 추상 방법입니다. 서비스를 사용하면 애플리케이션이 트래픽을 수신할 수 있습니다. **Service** 오브젝트에 **spec.type** 을 지정하여 다양한 방식으로 서비스를 노출할 수 있습니다.

ClusterIP
: 클러스터 내의 내부 IP 주소에 서비스를 노출합니다. **ClusterIP** 는 기본 서비스유형 입니다.

NodePort
: 클러스터에서 선택한 각 노드의 동일한 포트에 서비스를 노출합니다. **NodePort** 를 사용하면 클러스터 외부에서 서비스에 액세스할 수 있습니다.

LoadBalancer
: 현재 클라우드에서 외부 로드 밸런서를 생성하고(지원되는 경우) 고정 외부 IP 주소를 서비스에 할당합니다.

8.18.2.1.1. 듀얼 스택 지원

클러스터에 대해 IPv4 및 IPv6 이중 스택 네트워킹을 사용하도록 설정한 경우 **Service** 개체에 **spec.ipFamilyPolicy** 및 **spec.ipFamilies** 필드를 정의하여 IPv4, IPv6 또는 둘 다 사용하는 서비스를 생성할 수 있습니다.

spec.ipFamilyPolicy 필드는 다음 값 중 하나로 설정할 수 있습니다.

SingleStack

컨트롤 플레인은 첫 번째 구성된 서비스 클러스터 IP 범위를 기반으로 서비스에 대한 클러스터 IP 주소를 할당합니다.

PreferDualStack

컨트롤 플레인은 듀얼 스택이 구성된 클러스터에서 서비스에 대해 IPv4 및 IPv6 클러스터 IP 주소를 모두 할당합니다.

RequireDualStack

이 옵션은 듀얼 스택 네트워킹이 활성화되지 않은 클러스터에 실패합니다. 듀얼 스택이 구성된 클러스터의 경우 해당 동작은 값이 **PreferDualStack**으로 설정된 경우와 동일합니다. 컨트롤 플레인은 IPv4 및 IPv6 주소 범위의 클러스터 IP 주소를 할당합니다.

spec.ipFamilies 필드를 다음 배열 값 중 하나로 설정하여 단일 스택에 사용할 IP 제품군을 정의하거나 이중 스택의 IP 제품군 순서를 정의할 수 있습니다.

- **[IPv4]**
- **[IPv6]**
- **[IPv4, IPv6]**
- **[IPv6, IPv4]**

8.18.2.2. 가상 머신을 서비스로 노출

클러스터 내부 또는 외부에서 실행 중인 VM(가상 머신)에 연결할 **ClusterIP**,**NodePort** **LoadBalancer** 서비스를 생성합니다.

절차

1. **VirtualMachine** 매니페스트를 편집하여 서비스 생성을 위한 라벨을 추가합니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: vm-ephemeral
  namespace: example-namespace
spec:
  running: false
  template:
    metadata:
      labels:
        special: key (1)
# ...
```

(1) **spec.template.metadata.labels** 섹션에 **special: key** 라벨을 추가합니다.

참고

가상 머신의 라벨은 Pod로 전달됩니다. **special: key** 레이블은서비스 매니페스트의 **spec.selector** 특성의 레이블과 일치해야 합니다.

2. VirtualMachine 매니페스트 파일을 저장하여 변경 사항을 적용합니다.

3. VM을 노출할 서비스 매니페스트를 생성합니다.

```
apiVersion: v1
kind: Service
metadata:
  name: vmservice 1
  namespace: example-namespace 2
spec:
  externalTrafficPolicy: Cluster 3
  ports:
  - nodePort: 30000 4
    port: 27017
    protocol: TCP
    targetPort: 22 5
  selector:
    special: key 6
  type: NodePort 7
```

1. Service 오브젝트의 이름입니다.
2. Service 오브젝트가 있는 네임스페이스입니다. 이는 VirtualMachine 매니페스트의 metadata.namespace 필드와 일치해야 합니다.
3. 선택 사항: 노드에서 외부 IP 주소에서 수신되는 서비스 트래픽을 배포하는 방법을 지정합니다. 이는 NodePort 및 LoadBalancer 서비스 유형에만 적용됩니다. 기본값은 Cluster 로 트래픽을 모든 클러스터 끝점으로 균등하게 라우팅합니다.
4. 선택 사항: 설정하면 nodePort 값이 모든 서비스에서 고유해야 합니다. 지정하지 않으면 30000 이상의 범위의 값이 동적으로 할당됩니다.
5. 선택 사항: 서비스에서 노출할 VM 포트입니다. 포트 목록이 VM 매니페스트에 정의된 경우 열려 있는 포트를 참조해야 합니다. targetPort 를 지정하지 않으면포트 와 동일한 값을 사용합니다.
6. VirtualMachine 매니페스트의 spec.template.metadata.labels 스탠자에 추가한 라벨 참조입니다.
7. 서비스 유형입니다. 가능한 값은 ClusterIP,NodePort 및 LoadBalancer 입니다.

4. 서비스 매니페스트 파일을 저장합니다.

5. 다음 명령을 실행하여 서비스를 생성합니다.

```
$ oc create -f <service_name>.yaml
```

6. VM을 시작합니다. VM이 이미 실행 중인 경우 다시 시작합니다.

검증

1. Service 오브젝트를 쿼리하여 사용할 수 있는지 확인합니다.

```
$ oc get service -n example-namespace
```

-

ClusterIP 서비스의 출력 예

```
NAME        TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)     AGE
vmservice   ClusterIP   172.30.3.149   <none>        27017/TCP   2m
```

NodePort 서비스의 출력 예

```
NAME        TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)           AGE
vmservice   NodePort    172.30.232.73   <none>        27017:30000/TCP   5m
```

LoadBalancer 서비스의 출력 예

```
NAME        TYPE           CLUSTER-IP     EXTERNAL-IP                  PORT(S)           AGE
vmservice   LoadBalancer   172.30.27.5    172.29.10.235,172.29.10.235
27017:31829/TCP   5s
```

2. 가상 머신에 연결하는 적절한 방법을 선택합니다.

- ClusterIP 서비스의 경우 서비스 IP 주소와 서비스 포트를 사용하여 클러스터 내에서 VM에 연결합니다. 예를 들면 다음과 같습니다.

```
$ ssh fedora@172.30.3.149 -p 27017
```

- NodePort 서비스의 경우 노드 IP 주소와 클러스터 네트워크 외부의 노드 포트를 지정하여 VM에 연결합니다. 예를 들면 다음과 같습니다.

```
$ ssh fedora@$NODE_IP -p 30000
```

- LoadBalancer 서비스의 경우 vinagre 클라이언트를 사용하여 공용 IP 주소 및 포트를 사용하여 가상 머신에 연결합니다. 외부 포트는 동적으로 할당됩니다.

8.18.2.3. 추가 리소스

- NodePort를 사용하여 수신 클러스터 트래픽 구성
- 로드 밸런서를 사용하여 수신 클러스터 트래픽 구성

8.18.3. Linux 브리지 네트워크에 가상 머신 연결

기본적으로 OpenShift Virtualization은 하나의 내부 Pod 네트워크를 사용하여 설치됩니다.

추가 네트워크에 연결하려면 Linux 브리지 네트워크 연결 정의(NAD)를 생성해야 합니다.

가상 머신을 추가 네트워크에 연결하려면 다음을 수행합니다.

1. Linux 브리지 노드 네트워크 구성 정책을 만듭니다.
2. Linux 브리지 네트워크 연결 정의를 만듭니다.
3. 가상 머신이 네트워크 연결 정의를 인식할 수 있도록 가상 머신을 구성합니다.

스케줄링, 인터페이스 유형 및 기타 노드 네트워킹 활동에 대한 자세한 내용은 노드 네트워킹 섹션을 참조하십시오.

8.18.3.1. 네트워크 연결 정의를 통해 네트워크에 연결

8.18.3.1.1. Linux 브리지 노드 네트워크 구성 정책 생성

NodeNetworkConfigurationPolicy 매니페스트 YAML 파일을 사용하여 Linux 브리지를 만듭니다.

절차

- **NodeNetworkConfigurationPolicy** 매니페스트를 생성합니다. 이 예제에는 해당 정보로 교체해야 하는 샘플 값이 포함되어 있습니다.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: br1-eth1-policy (1)
spec:
  desiredState:
    interfaces:
      - name: br1 (2)
        description: Linux bridge with eth1 as a port (3)
        type: linux-bridge (4)
        state: up (5)
        ipv4:
          enabled: false (6)
        bridge:
          options:
            stp:
              enabled: false (7)
          port:
            - name: eth1 (8)
```

(1) 정책 이름입니다.

(2) 인터페이스 이름입니다.

(3) 선택 사항: 사람이 읽을 수 있는 인터페이스 설명입니다.

(4) 인터페이스 유형입니다. 이 예제에서는 브리지를 만듭니다.

(5) 생성 후 인터페이스에 요청되는 상태입니다.

(6) 이 예에서 IPv4를 비활성화합니다.

(7) 이 예에서 STP를 비활성화합니다.

(8) 브리지가 연결된 노드 NIC입니다.

8.18.3.2. Linux 브리지 네트워크 연결 정의 생성

주의

가상 머신의 네트워크 연결 정의에서 IPAM(IP 주소 관리) 구성은 지원되지 않습니다.

8.18.3.2.1. 웹 콘솔에서 Linux 브리지 네트워크 연결 정의 생성

네트워크 관리자는 네트워크 연결 정의를 생성하여 Pod 및 가상 머신에 계층 2 네트워킹을 제공할 수 있습니다.

절차

1. 웹 콘솔에서 네트워킹 → 네트워크 연결 정의를 클릭합니다.
2. 네트워크 연결 정의 생성을 클릭합니다.

   참고

   네트워크 연결 정의는 Pod 또는 가상 머신과 동일한 네임스페이스에 있어야 합니다.
3. 고유한 이름과 선택적 설명을 입력합니다.
4. 네트워크 유형 목록을 클릭하고 CNV Linux 브리지를 선택합니다.
5. 브리지 이름 필드에 브리지 이름을 입력합니다.
6. 선택 사항: 리소스에 VLAN ID가 구성된 경우 VLAN 태그 번호 필드에 ID 번호를 입력합니다.
7. 선택 사항: MAC 스푸핑 검사를 선택하여 MAC 스푸핑 필터링을 활성화합니다. 이 기능은 단일 MAC 주소만 Pod를 종료할 수 있도록 하여 MAC 스푸핑 공격에 대한 보안을 제공합니다.
8. 생성을 클릭합니다.

   참고

   Linux 브리지 네트워크 연결 정의는 가상 머신을 VLAN에 연결하는 가장 효율적인 방법입니다.

8.18.3.2.2. CLI에서 Linux 브리지 네트워크 연결 정의 생성

네트워크 관리자는 **cnv-bridge** 유형의 네트워크 연결 정의를 구성하여 Pod 및 가상 머신에 계층 2 네트워킹을 제공할 수 있습니다.

사전 요구 사항

- 노드는 nftables를 지원해야 하며 MAC 스푸핑 검사를 사용하려면 **nft** 바이너리를 배포해야 합니다.

절차

1. 가상 머신과 동일한 네임스페이스에 네트워크 연결 정의를 생성합니다.

2. 다음 예와 같이 네트워크 연결 정의에 가상 머신을 추가합니다.

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: <bridge-network> (1)
  annotations:
    k8s.v1.cni.cncf.io/resourceName: bridge.network.kubevirt.io/<bridge-interface> (2)
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "<bridge-network>", (3)
    "type": "cnv-bridge", (4)
    "bridge": "<bridge-interface>", (5)
    "macspoofchk": true, (6)
    "vlan": 1 (7)
  }'
```

(1) NetworkAttachmentDefinition 개체의 이름입니다.

(2) 선택 사항: 노드 선택용 주석 키-값 쌍입니다. 여기서 **bridge-interface** 는 일부 노드에 구성된 브리지 이름과 일치해야 합니다. 네트워크 연결 정의에 이 주석을 추가하면 **bridge-interface** 브리지가 연결된 노드에서만 가상 머신 인스턴스가 실행됩니다.

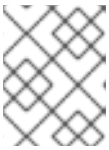
(3) 구성의 이름입니다. 구성 이름이 네트워크 연결 정의의 **name** 값과 일치하는 것이 좋습니다.

(4) 이 네트워크 연결 정의에 대한 네트워크를 제공하는 CNI(컨테이너 네트워크 인터페이스) 플러그인의 실제 이름입니다. 다른 CNI를 사용하려는 경우를 제외하고 이 필드를 변경하지 마십시오.

(5) 노드에 구성된 Linux 브리지의 이름입니다.

(6) 선택 사항: MAC 스푸핑 검사를 활성화하는 플래그입니다. **true**로 설정하면 Pod 또는 게스트 인터페이스의 MAC 주소를 변경할 수 없습니다. 이 속성은 단일 MAC 주소만 Pod를 종료할 수 있도록 허용하여 MAC 스푸핑 공격에 대한 보안을 제공합니다.

(7) 선택 사항: VLAN 태그. 노드 네트워크 구성 정책에는 추가 VLAN 구성이 필요하지 않습니다.



참고

Linux 브리지 네트워크 연결 정의는 가상 머신을 VLAN에 연결하는 가장 효율적인 방법입니다.

3. 네트워크 연결 정의를 만듭니다.

```
$ oc create -f <network-attachment-definition.yaml> (1)
```

(1) 여기서 **<network-attachment-definition.yaml>**은 네트워크 연결 정의 매니페스트의 파일 이름입니다.

검증

- 다음 명령을 실행하여 네트워크 연결 정의가 생성되었는지 확인합니다.

```
$ oc get network-attachment-definition <bridge-network>
```

8.18.3.3. Linux 브리지 네트워크에 대한 가상 머신 구성

8.18.3.3.1. 웹 콘솔에서 가상 머신의 NIC를 생성

웹 콘솔에서 추가 NIC를 생성하고 가상 머신에 연결합니다.

사전 요구 사항

- 네트워크 연결 정의를 사용할 수 있어야 합니다.

절차

1. OpenShift Container Platform 콘솔의 올바른 프로젝트에서 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 네트워크 인터페이스 탭을 클릭하여 가상 머신에 이미 연결된 NIC를 확인합니다.
4. 네트워크 인터페이스 추가를 클릭하여 목록에 새 슬롯을 만듭니다.
5. 추가 네트워크의 네트워크 목록에서 네트워크 연결 정의를 선택합니다.
6. 새 NIC의 이름, 모델, 유형, MAC 주소를 입력합니다.
7. 저장을 클릭하여 NIC를 저장하고 가상 머신에 연결합니다.

8.18.3.3.2. 네트워킹 필드

이름	설명
이름	네트워크 인터페이스 컨트롤러의 이름입니다.
모델	네트워크 인터페이스 컨트롤러의 모델을 나타냅니다. 지원되는 값은 **e1000e** 및 **virtio**입니다.
네트워크	사용 가능한 네트워크 연결 정의 목록입니다.
유형	사용 가능한 바인딩 방법 목록입니다. 네트워크 인터페이스에 적합한 바인딩 방법을 선택합니다. • 기본 Pod 네트워크: **masquerade** • Linux 브리지 네트워크: **bridge** • SR-IOV 네트워크: **SR-IOV**

이름	설명
MAC 주소	네트워크 인터페이스 컨트롤러의 MAC 주소입니다. MAC 주소를 지정하지 않으면 주소가 자동으로 할당됩니다.

8.18.3.3.3. CLI의 추가 네트워크에 가상 머신 연결

브리지 인터페이스를 추가하고 가상 머신 구성에서 네트워크 연결 정의를 지정하여 가상 머신을 추가 네트워크에 연결합니다.

이 절차에서는 YAML 파일을 사용하여 구성을 편집하고 업데이트된 파일을 클러스터에 적용하는 방법을 시연합니다. 또는 **oc edit <object> <name>** 명령을 사용하여 기존 가상 머신을 편집할 수도 있습니다.

사전 요구 사항

- 구성을 편집하기 전에 가상 머신을 종료합니다. 실행 중인 가상 머신을 편집하는 경우 변경 사항을 적용하려면 가상 머신을 다시 시작해야 합니다.

절차

1. 브리지 네트워크에 연결하려는 가상 머신 구성을 생성하거나 편집합니다.
2. **spec.template.spec.domain.devices.interfaces** 목록에 브리지 인터페이스를 추가하고 **spec.template.spec.networks** 목록에 네트워크 연결 정의를 추가합니다. 이 예제에서는**a-bridge-network** 네트워크 연결 정의에 연결하는**bridge-net** 브리지 인터페이스를 추가합니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
   name: <example-vm>
spec:
 template:
   spec:
     domain:
       devices:
         interfaces:
           - masquerade: {}
             name: <default>
           - bridge: {}
             name: <bridge-net> 1
...
     networks:
       - name: <default>
         pod: {}
       - name: <bridge-net> 2
         multus:
           networkName: <network-namespace>/<a-bridge-network> 3
...
```

1. 브리지 인터페이스의 이름입니다.
2. 네트워크의 이름입니다. 이 값은 해당 **spec.template.spec.domain.devices.interfaces** 항목의 **name** 값과 일치해야 합니다.
3. 네트워크 연결 정의의 이름, 존재하는 네임스페이스가 접두사로 지정됩니다. 네임스페이스는 **default** 네임스페이스 또는 VM을 생성할 동일한 네임스페이스여야 합니다. 이 경우**multus** 가 사용됩니다. Multus는 Pod 또는 가상 머신에서 필요한 인터페이스를 사용할 수 있도록 여러 CNI가 존재할 수 있는 클라우드 네트워크 인터페이스(CNI) 플러그인입니다.

3. 설정을 적용합니다.

```
$ oc apply -f <example-vm.yaml>
```

4. 선택 사항: 실행 중인 가상 머신을 편집한 경우 변경 사항을 적용하려면 가상 머신을 다시 시작해야 합니다.

8.18.4. SR-IOV 네트워크에 가상 머신 연결

다음 단계를 수행하여 VM(가상 머신)을 SR-IOV(Single Root I/O Virtualization) 네트워크에 연결할 수 있습니다.

1. SR-IOV 네트워크 장치를 구성합니다.
2. SR-IOV 네트워크를 구성합니다.
3. VM을 SR-IOV 네트워크에 연결합니다.

8.18.4.1. 전제 조건

- 호스트의 펌웨어에 글로벌 SR-IOV 및 VT-d 설정이 활성화되어있어야 합니다.
- SR-IOV Network Operator가 설치되어 있어야 합니다.

8.18.4.2. SR-IOV 네트워크 장치 구성

SR-IOV Network Operator는 **SriovNetworkNodePolicy.sriovnetwork.openshift.io** CustomResourceDefinition을 OpenShift Container Platform에 추가합니다. SriovNetworkNodePolicy CR(사용자 정의 리소스)을 만들어 SR-IOV 네트워크 장치를 구성할 수 있습니다.

참고

SriovNetworkNodePolicy 오브젝트에 지정된 구성을 적용하면 SR-IOV Operator가 노드를 비우고 경우에 따라 노드를 재부팅할 수 있습니다.

구성 변경 사항을 적용하는 데 몇 분이 걸릴 수 있습니다.

사전 요구 사항

- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

- SR-IOV Network Operator가 설치되어 있습니다.
- 비운 노드에서 제거된 워크로드를 처리하기 위해 클러스터에 사용 가능한 노드가 충분합니다.
- SR-IOV 네트워크 장치 구성에 대한 컨트롤 플레인 노드를 선택하지 않았습니다.

절차

1. SriovNetworkNodePolicy 오브젝트를 생성한 후 YAML을<name>-sriov-node-network.yaml 파일에 저장합니다. <name>을 이 구성의 이름으로 바꿉니다.

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: <name> 1
  namespace: openshift-sriov-network-operator 2
spec:
  resourceName: <sriov_resource_name> 3
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true" 4
  priority: <priority> 5
  mtu: <mtu> 6
  numVfs: <num> 7
  nicSelector: 8
    vendor: "<vendor_code>" 9
    deviceID: "<device_id>" 10
    pfNames: ["<pf_name>", ...] 11
    rootDevices: ["<pci_bus_id>", "..."] 12
  deviceType: vfio-pci 13
  isRdma: false 14
```

1. CR 오브젝트의 이름을 지정합니다.
2. SR-IOV Operator가 설치된 네임스페이스를 지정합니다.
3. SR-IOV 장치 플러그인의 리소스 이름을 지정합니다. 리소스 이름에 대해 여러 SriovNetworkNodePolicy 오브젝트를 생성할 수 있습니다.
4. 구성할 노드를 선택하려면 노드 선택기를 지정합니다. 선택한 노드의 SR-IOV 네트워크 장치만 구성됩니다. SR-IOV CNI(Container Network Interface) 플러그인 및 장치 플러그인은 선택한 노드에만 배포됩니다.
5. 선택 사항: 0에서 99 사이의 정수 값을 지정합니다. 숫자가 작을수록 우선 순위가 높아지므로 우선 순위 10은 우선 순위 99보다 높습니다. 기본값은 99입니다.
6. 선택 사항: 가상 기능의 최대 전송 단위(MTU) 값을 지정합니다. 최대 MTU 값은 NIC 모델마다 다를 수 있습니다.
7. SR-IOV 물리적 네트워크 장치에 생성할 가상 기능(VF) 수를 지정합니다. Intel NIC(Network Interface Controller)의 경우 VF 수는 장치에서 지원하는 총 VF보다 클 수 없습니다. Mellanox NIC의 경우 VF 수는 128보다 클 수 없습니다.
8. nicSelector 매핑은 Operator가 구성할 이더넷 장치를 선택합니다. 모든 매개변수에 값을 지정할 필요는 없습니다. 의도하지 않게 이더넷 장치를 선택할 가능성을 최소화하기 위해 이더넷 어댑터를 충분히 정밀하게 식별하는 것이 좋습니다. rootDevices를 지정하면 vendor,

deviceID 또는 **pfNames**의 값도 지정해야 합니다. **pfNames**와 **rootDevices**를 동시에 지정하는 경우 동일한 장치를 가리키는지 확인하십시오.

9 선택 사항: SR-IOV 네트워크 장치의 공급업체 16진 코드를 지정합니다. 허용되는 유일한 값은 **8086** 또는 **15b3**입니다.

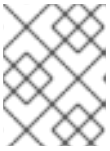
10 선택 사항: SR-IOV 네트워크 장치의 장치 16진수 코드를 지정합니다. 허용되는 값은 **158b**, **1015**, **1017**입니다.

11 선택 사항:이 매개변수는 이더넷 장치에 대한 하나 이상의 PF(물리적 기능) 이름 배열을 허용합니다.

12 이 매개변수는 이더넷 장치의 물리적 기능을 위해 하나 이상의 PCI 버스 주소 배열을 허용합니다. 주소를 **0000:02: 00.1** 형식으로 입력합니다.

13 **vfio-pci** 드라이버 유형은 OpenShift Virtualization의 가상 기능에 필요합니다.

14 선택 사항: 원격 직접 메모리 액세스(RDMA) 모드 사용 여부를 지정합니다. Mellanox 카드의 경우 **isRdma**를 **false**로 설정합니다. 기본값은 **false**입니다.



참고

isRDMA 플래그가 **true**로 설정된 경우 RDMA 가능 VF를 일반 네트워크 장치로 계속 사용할 수 있습니다. 어느 모드에서나 장치를 사용할 수 있습니다.

2. 선택 사항: SR-IOV 가능 클러스터 노드에 **SriovNetworkNodePolicy.Spec.NodeSelector** 레이블이 지정되지 않은 경우 레이블을 지정합니다. 노드 레이블링에 대한 자세한 내용은 "노드에서 라벨을 업데이트하는 방법"을 참조하십시오.

3. **SriovNetworkNodePolicy** 오브젝트를 생성합니다.

```
$ oc create -f <name>-sriov-node-network.yaml
```

<name>은 이 구성의 이름을 지정합니다.

구성 업데이트를 적용하면 **sriov-network-operator** 네임스페이스의 모든 Pod가 **Running** 상태로 전환됩니다.

4. SR-IOV 네트워크 장치가 구성되어 있는지 확인하려면 다음 명령을 입력합니다. **<node_name>**을 방금 구성한 SR-IOV 네트워크 장치가 있는 노드 이름으로 바꿉니다.

```
$ oc get sriovnetworknodestates -n openshift-sriov-network-operator <node_name> -o jsonpath='{.status.syncStatus}'
```

8.18.4.3. SR-IOV 추가 네트워크 구성

SriovNetwork 오브젝트를 생성하여 SR-IOV 하드웨어를 사용하는 추가 네트워크를 구성할 수 있습니다.

SriovNetwork 오브젝트를 생성하면 SR-IOV Network Operator가 **NetworkAttachmentDefinition** 오브젝트를 자동으로 생성합니다.

참고

SriovNetwork 오브젝트가 Pod 또는 가상 머신에**running** 상태의 경우 수정하거나 삭제하지 마십시오.

사전 요구 사항

- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 로그인합니다.

절차

1. 다음 **SriovNetwork** 오브젝트를 생성한 후 YAML을**<name>-sriov-network.yaml** 파일에 저장합니다. **<name>**을 이 추가 네트워크의 이름으로 변경합니다.

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetwork
metadata:
  name: <name> 1
  namespace: openshift-sriov-network-operator 2
spec:
  resourceName: <sriov_resource_name> 3
  networkNamespace: <target_namespace> 4
  vlan: <vlan> 5
  spoofChk: "<spoof_check>" 6
  linkState: <link_state> 7
  maxTxRate: <max_tx_rate> 8
  minTxRate: <min_rx_rate> 9
  vlanQoS: <vlan_qos> 10
  trust: "<trust_vf>" 11
  capabilities: <capabilities> 12
```

1. **<name>**을 오브젝트의 이름으로 바꿉니다. SR-IOV Network Operator는 동일한 이름으로 **NetworkAttachmentDefinition** 오브젝트를 생성합니다.
2. SR-IOV Network Operator가 설치된 네임스페이스를 지정합니다.
3. **<sriov_resource_name>**을 이 추가 네트워크에 대한 SR-IOV 하드웨어를 정의하는 **SriovNetworkNodePolicy** 오브젝트의 **spec.resourceName** 매개변수 값으로 바꿉니다.
4. **<target_namespace>**를 SriovNetwork의 대상 네임스페이스로 바꿉니다. 대상 네임스페이스의 pod 또는 가상 머신만 SriovNetwork에 연결할 수 있습니다.
5. 선택 사항: **<vlan>**을 추가 네트워크의 VLAN(Virtual LAN) ID로 바꿉니다. 정수 값은**0**에서 **4095** 사이여야 합니다. 기본값은 **0**입니다.
6. 선택 사항: **<spoof_check>**를 VF의 위조 확인 모드로 바꿉니다. 허용되는 값은 문자열**"on"** 및 **"off"**입니다.

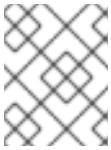
중요

SR-IOV Network Operator가 지정한 값을 따옴표로 묶거나 CR을 거부해야 합니다.

7 선택 사항: **<link_state>**를 가상 기능(VF)의 링크 상태로 바꿉니다. 허용되는 값은**enable, disable** 및 **auto**입니다.

8 선택 사항: VF의 경우 **<max_tx_rate>**를 최대 전송 속도(Mbps)로 바꿉니다.

9 선택 사항: VF의 경우 **<min_tx_rate>**를 최소 전송 속도(Mbps)로 바꿉니다. 이 값은 항상 최대 전송 속도보다 작거나 같아야 합니다.



참고

인텔 NIC는 **minTxRate** 매개변수를 지원하지 않습니다. 자세한 내용은 BZ#1772847에서 참조하십시오.

10 선택 사항: **<vlan_qos>**를 VF의 IEEE 802.1p 우선 순위 레벨로 바꿉니다. 기본값은**0**입니다.

11 선택 사항: **<trust_vf>**를 VF의 신뢰 모드로 바꿉니다. 허용되는 값은 문자열**"on"** 및 **"off"**입니다.

중요

SR-IOV Network Operator가 지정한 값을 따옴표로 묶거나 CR을 거부해야 합니다.

12 선택 사항: **<capabilities>**를 이 네트워크에 구성할 수 있는 기능으로 바꿉니다.

2. 오브젝트를 생성하려면 다음 명령을 입력합니다. **<name>**을 이 추가 네트워크의 이름으로 변경합니다.

```
$ oc create -f <name>-sriov-network.yaml
```

3. 선택 사항: 이전 단계에서 생성한 **SriovNetwork** 오브젝트에 연결된 **NetworkAttachmentDefinition** 오브젝트가 존재하는지 확인하려면 다음 명령을 입력합니다. **<namespace>**를 **SriovNetwork** 오브젝트에 지정한 네임스페이스로 바꿉니다.

```
$ oc get net-attach-def -n <namespace>
```

8.18.4.4. SR-IOV 네트워크에 가상 머신 연결

VM 구성에 네트워크 세부 정보를 포함하여 VM(가상 머신)을 SR-IOV 네트워크에 연결할 수 있습니다.

절차

1. VM 구성의 **spec.domain.devices.interfaces** 및 **spec.networks** 에 SR-IOV 네트워크 세부 정보를 포함합니다.

```
kind: VirtualMachine
...
spec:
  domain:
    devices:
      interfaces:
      - name: <default> 1
        masquerade: {} 2
      - name: <nic1> 3
```

```
      sriov: {}
  networks:
  - name: <default> 4
    pod: {}
  - name: <nic1> 5
    multus:
      networkName: <sriov-network> 6
...
```

1. Pod 네트워크에 연결된 인터페이스의 고유 이름입니다.
2. 기본 Pod 네트워크에 대한 **masquerade** 바인딩입니다.
3. SR-IOV 인터페이스의 고유 이름입니다.
4. Pod 네트워크 인터페이스의 이름입니다. 이전에 정의한 **interfaces.name**과 동일해야 합니다.
5. SR-IOV 인터페이스의 이름입니다. 이전에 정의한 **interfaces.name**과 동일해야 합니다.
6. SR-IOV 네트워크 연결 정의의 이름입니다.

2. 가상 머신 구성을 적용합니다.

```
$ oc apply -f <vm-sriov.yaml> 1
```

1. 가상 머신 YAML 파일의 이름입니다.

8.18.5. 서비스 메시에 가상 머신 연결

OpenShift Virtualization이 OpenShift Service Mesh와 통합되었습니다. IPv4를 사용하여 기본 Pod 네트워크에서 가상 머신 워크로드를 실행하는 pod 간 트래픽을 모니터링, 시각화 및 제어할 수 있습니다.

8.18.5.1. 사전 요구 사항

- Service Mesh Operator를 설치하고 서비스 메시 컨트롤 플레인을 배포해야 합니다.
- 가상 머신이 서비스 메시 멤버 롤에 생성되는 네임스페이스를 추가해야 합니다.
- 기본 Pod 네트워크에 **masquerade** 바인딩 방법을 사용해야 합니다.

8.18.5.2. 서비스 메시의 가상 머신 구성

서비스 메시에 VM(가상 머신) 워크로드를 추가하려면 **sidecar.istio.io/inject** 주석을 **true** 로 설정하여 VM 구성 파일에서 자동 사이드카 삽입을 활성화합니다. 그런 다음 VM을 서비스로 노출하여 메시에서 애플리케이션을 확인합니다.

사전 요구 사항

- 포트 충돌을 방지하려면 Istio 사이드카 프록시에서 사용하는 포트를 사용하지 마십시오. 여기에는 포트 15000, 15001, 15006, 15008, 15020, 15021, 15090이 포함됩니다.

절차

1. VM 구성 파일을 편집하여 **sidecar.istio.io/inject: "true"** 주석을 추가합니다.

   설정 파일 예

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: vm-istio
  name: vm-istio
spec:
  runStrategy: Always
  template:
    metadata:
      labels:
        kubevirt.io/vm: vm-istio
        app: vm-istio (1)
      annotations:
        sidecar.istio.io/inject: "true" (2)
    spec:
      domain:
        devices:
          interfaces:
          - name: default
            masquerade: {} (3)
          disks:
          - disk:
              bus: virtio
            name: containerdisk
          - disk:
              bus: virtio
            name: cloudinitdisk
        resources:
          requests:
            memory: 1024M
      networks:
      - name: default
        pod: {}
      terminationGracePeriodSeconds: 180
      volumes:
      - containerDisk:
          image: registry:5000/kubevirt/fedora-cloud-container-disk-demo:devel
        name: containerdisk
```

   (1) 서비스 선택기 특성과 일치해야 하는 키/값 쌍(라벨)입니다.

   (2) 자동 사이드카 삽입을 활성화하는 주석입니다.

   (3) 기본 Pod 네트워크에 사용할 바인딩 방법(masquerade 모드)입니다.

2. VM 구성을 적용합니다.

```
$ oc apply -f <vm_name>.yaml 1
```

1 가상 머신 YAML 파일의 이름입니다.

3. VM을 서비스 메시에 노출하는 **Service** 오브젝트를 생성합니다.

```
apiVersion: v1
kind: Service
metadata:
  name: vm-istio
spec:
  selector:
    app: vm-istio 1
  ports:
    - port: 8080
      name: http
      protocol: TCP
```

1 서비스에서 대상으로 하는 Pod 세트를 결정하는 서비스 선택기입니다. 이 속성은 VM 구성 파일의 **spec.metadata.labels** 필드에 해당합니다. 위의 예에서 **vm-istio** 라는 **Service** 오브젝트는 **app=vm-istio** 라벨이 있는 모든 Pod에서 TCP 포트 8080을 대상으로 합니다.

4. 서비스를 생성합니다.

```
$ oc create -f <service_name>.yaml 1
```

1 서비스 YAML 파일의 이름입니다.

8.18.6. 가상 머신용 IP 주소 구성

가상 머신에 대해 동적으로 또는 정적으로 프로비저닝된 IP 주소를 구성할 수 있습니다.

사전 요구 사항

- 가상 머신은 외부 네트워크에 연결되어 있어야 합니다.
- 가상 시스템의 동적 IP를 구성하려면 추가 네트워크에서 DHCP 서버를 사용할 수 있어야 합니다.

8.18.6.1. cloud-init를 사용하여 새 가상 머신의 IP 주소 구성

cloud-init를 사용하여 가상 머신을 생성할 때 IP 주소를 구성할 수 있습니다. IP 주소는 동적으로 또는 정적으로 프로비저닝될 수 있습니다.

절차

- 가상 머신을 구성하고 가상 머신 구성의 **spec.volumes.cloudInitNoCloud.networkData** 필드에 cloud-init 네트워크 세부 정보를 포함합니다.
  a. 동적 IP를 구성하려면 인터페이스 이름과 **dhcp4** 부울을 지정합니다.

```
kind: VirtualMachine
```

```
spec:
...
  volumes:
  - cloudInitNoCloud:
      networkData: |
        version: 2
        ethernets:
          eth1: (1)
            dhcp4: true (2)
```

(1) 인터페이스 이름입니다.

(2) DHCP를 사용하여 IPv4 주소를 프로비저닝합니다.

b. 고정 IP를 구성하려면 인터페이스 이름과 IP 주소를 지정합니다.

```
kind: VirtualMachine
spec:
...
  volumes:
  - cloudInitNoCloud:
      networkData: |
        version: 2
        ethernets:
          eth1: (1)
            addresses:
            - 10.10.10.14/24 (2)
```

(1) 인터페이스 이름입니다.

(2) 가상 머신의 고정 IP 주소입니다.

8.18.7. 가상 머신에서 NIC의 IP 주소 보기

웹 콘솔 또는 **oc** 클라이언트를 사용하여 NIC(네트워크 인터페이스 컨트롤러)의 IP 주소를 볼 수 있습니다. QEMU 게스트 에이전트는 가상 머신의 보조 네트워크에 대한 추가 정보를 표시합니다.

8.18.7.1. 사전 요구 사항

- 가상 머신에 QEMU 게스트 에이전트를 설치합니다.

8.18.7.2. CLI에서 가상 머신 인터페이스의 IP 주소 보기

네트워크 인터페이스 구성은 **oc describe vmi <vmi_name>** 명령에 포함되어 있습니다.

가상 머신에서 **ip addr**을 실행하거나 **oc get vmi <vmi_name> -o yaml**을 실행하여 IP 주소 정보를 볼 수도 있습니다.

절차

- **oc describe** 명령을 사용하여 가상 머신 인터페이스 구성을 표시합니다.

```
$ oc describe vmi <vmi_name>
```

출력 예

```
...
Interfaces:
  Interface Name:  eth0
  Ip Address:      10.244.0.37/24
  Ip Addresses:
    10.244.0.37/24
    fe80::858:aff:fef4:25/64
  Mac:             0a:58:0a:f4:00:25
  Name:            default
  Interface Name:  v2
  Ip Address:      1.1.1.7/24
  Ip Addresses:
    1.1.1.7/24
    fe80::f4d9:70ff:fe13:9089/64
  Mac:             f6:d9:70:13:90:89
  Interface Name:  v1
  Ip Address:      1.1.1.1/24
  Ip Addresses:
    1.1.1.1/24
    1.1.1.2/24
    1.1.1.4/24
    2001:de7:0:f101::1/64
    2001:db8:0:f101::1/64
    fe80::1420:84ff:fe10:17aa/64
  Mac:             16:20:84:10:17:aa
```

8.18.7.3. 웹 콘솔에서 가상 머신 인터페이스의 IP 주소 보기

IP 정보는 가상 머신의 VirtualMachine 세부 정보 페이지에 표시됩니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신 이름을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.

연결된 각 NIC에 대한 정보는 세부 정보 탭의 IP 주소 아래에 표시됩니다.

8.18.8. 가상 머신의 MAC 주소 풀 사용

KubeMacPool 구성 요소는 네임스페이스의 가상 머신 NIC에 대한 MAC 주소 풀 서비스를 제공합니다.

8.18.8.1. About KubeMacPool

KubeMacPool은 네임스페이스당 MAC 주소 풀을 제공하고 풀의 가상 머신 NIC에 MAC 주소를 할당합니다. 이렇게 하면 다른 가상 머신의 MAC 주소와 충돌하지 않는 고유한 MAC 주소가 NIC에 할당됩니다.

해당 가상 머신에서 생성된 가상 머신 인스턴스에서는 재부팅 시 할당되는 MAC 주소가 유지됩니다.

참고

KubeMacPool은 가상 머신과 독립적으로 생성된 가상 머신 인스턴스는 처리하지 않습니다.

OpenShift Virtualization을 설치할 때 KubeMacPool은 기본적으로 활성화됩니다. 네임스페이스에 **mutatevirtualmachines.kubemacpool.io=ignore** 레이블을 추가하여 네임스페이스의 MAC 주소 풀을 비활성화합니다. 레이블을 제거하여 네임스페이스에 대해 KubeMacPool을 다시 활성화합니다.

8.18.8.2. CLI에서 네임스페이스의 MAC 주소 풀 비활성화

mutatevirtualmachines.kubemacpool.io=allocate 레이블을 네임스페이스에 추가하여 네임스페이스에서 가상 머신의 MAC 주소 풀을 비활성화합니다.

절차

- **mutatevirtualmachines.kubemacpool.io=ignore** 레이블을 네임스페이스에 추가합니다. 다음 예제에서는 **<namespace1>** 및 **<namespace2>** 네임스페이스에 KubeMacPool 라벨을 추가합니다.

```
$ oc label namespace <namespace1> <namespace2>
mutatevirtualmachines.kubemacpool.io=ignore
```

8.18.8.3. CLI에서 네임스페이스의 MAC 주소 풀을 다시 활성화

네임스페이스에 대해 KubeMacPool을 비활성화하고 다시 활성화하려면 네임스페이스에서 **mutatevirtualmachines.kubemacpool.io=ignore** 레이블을 제거합니다.

참고

이전 버전의 OpenShift Virtualization에서는 **mutatevirtualmachines.kubemacpool.io=allocate** 레이블을 사용하여 네임스페이스에 KubeMacPool을 활성화했습니다. 이는 여전히 지원되지만 KubeMacPool의 중복은 기본적으로 활성화되어 있습니다.

절차

- 네임스페이스에서 KubeMacPool 라벨을 제거합니다. 다음 예제에서는 **<namespace1>** 및 **<namespace2>** 네임스페이스에 KubeMacPool을 다시 사용하도록 설정합니다.

```
$ oc label namespace <namespace1> <namespace2>
mutatevirtualmachines.kubemacpool.io-
```

8.19. 가상 머신 디스크

8.19.1. 스토리지 기능

다음 표를 사용하여 OpenShift Virtualization의 로컬 및 공유 영구 스토리지에 대한 기능 가용성을 확인합니다.

8.19.1.1. OpenShift Virtualization 스토리지 기능 매트릭스

표 8.5. OpenShift Virtualization 스토리지 기능 매트릭스

	가상 머신 실시간 마이그레이션	호스트 지원 가상 머신 디스크 복제	스토리지 지원 가상 머신 디스크 복제	가상 머신 스냅샷
OpenShift Data Foundation: RBD 블록 모드 볼륨	있음	있음	있음	있음
OpenShift Virtualization hostpath 프로비전 프로그램	아니요	있음	아니요	아니요
기타 다중 노드 쓰기 가능 스토리지	예 [1]	있음	예 [2]	예 [2]
기타 단일 노드 쓰기 가능 스토리지	아니요	있음	예 [2]	예 [2]

1. PVC에서 ReadWriteMany 액세스 모드를 요청해야 합니다.
2. 스토리지 공급자는 Kubernetes 및 CSI Snapshot API를 모두 지원해야 합니다.

참고

다음을 사용하는 가상 머신은 실시간 마이그레이션할 수 없습니다.

- RWO(ReadWriteOnce) 액세스 모드를 사용하는 스토리지 클래스
- GPU와 같은 패스스루 기능

이러한 가상 머신의 경우 **evictionStrategy** 필드를 **LiveMigrate**로 설정하지 않도록 합니다.

8.19.2. 가상 머신 로컬 스토리지 구성

HPP(hostpath provisioner)를 사용하여 가상 머신의 로컬 스토리지를 구성할 수 있습니다.

8.19.2.1. hostpath 프로비전 프로그램 정보

OpenShift Virtualization Operator를 설치하면 HPP(Hostpath Provisioner) Operator가 자동으로 설치됩니다. HPP는 Hostpath Provisioner Operator가 생성한 OpenShift Virtualization용으로 설계된 로컬 스토리지 프로비전 프로그램입니다. HPP를 사용하려면 HPP 사용자 정의 리소스(CR)를 생성해야 합니다.

중요

OpenShift Virtualization 4.10에서 HPP Operator는 Kubernetes CSI 드라이버를 구성합니다. Operator는 HPP CR의 기존(레거시) 형식도 인식합니다.

기존 HPP 및 CSI(Container Storage Interface) 드라이버는 여러 릴리스에서 병렬로 지원됩니다. 그러나 경우에 따라 레거시 HPP는 더 이상 지원되지 않습니다. HPP를 사용하는 경우 마이그레이션 전략의 일부로 CSI 드라이버의 스토리지 클래스를 생성할 계획입니다.

기존 클러스터에서 OpenShift Virtualization 버전 4.10으로 업그레이드하는 경우 HPP Operator가 업그레이드되고 시스템에서 다음 작업을 수행합니다.

- CSI 드라이버가 설치되어 있습니다.

- CSI 드라이버는 기존 HPP CR의 콘텐츠로 구성됩니다.

새 클러스터에 OpenShift Virtualization 버전 4.10을 설치하는 경우 다음 작업을 수행해야 합니다.

- 기본 스토리지 풀을 사용하여 HPP CR을 생성합니다.
- CSI 드라이버의 스토리지 클래스를 생성합니다.

선택 사항: 여러 HPP 볼륨에 대해 PVC 템플릿으로 스토리지 풀을 생성할 수 있습니다.

8.19.2.2. 기본 스토리지 풀을 사용하여 hostpath 프로비전 프로그램 생성

storagePools 스탠자를 사용하여 HPP CR(사용자 정의 리소스)을 생성하여 기본 스토리지 풀로 hostpath 프로비전 프로그램(HPP)을 구성합니다. 스토리지 풀은 CSI 드라이버에서 사용하는 이름과 경로를 지정합니다.

사전 요구 사항

- **spec.storagePools.path** 에 지정된 디렉터리에는 읽기/쓰기 액세스 권한이 있어야 합니다.
- 스토리지 풀은 운영 체제와 동일한 파티션에 있을 수 없습니다. 그렇지 않으면 운영 체제 파티션이 용량으로 채워질 수 있으며 이로 인해 성능에 영향을 주거나 노드가 불안정하거나 사용할 수 없게 됩니다.

절차

1. 다음 예와 같이 **storagePools** 스탠자를 사용하여 **hpp_cr.yaml** 파일을 생성합니다.

```
apiVersion: hostpathprovisioner.kubevirt.io/v1beta1
kind: HostPathProvisioner
metadata:
  name: hostpath-provisioner
spec:
  imagePullPolicy: IfNotPresent
  storagePools: 1
  - name: any_name
    path: "/var/myvolumes" 2
workload:
  nodeSelector:
    kubernetes.io/os: linux
```

1. **storagePools** 스탠자는 여러 항목을 추가할 수 있는 배열입니다.
2. 이 노드 경로에 스토리지 풀 디렉터리를 지정합니다.

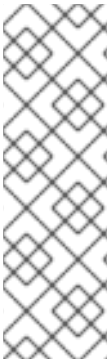
2. 파일을 저장하고 종료합니다.
3. 다음 명령을 실행하여 HPP를 생성합니다.

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

8.19.2.3. 스토리지 클래스 생성 정보

스토리지 클래스를 생성할 때 해당 스토리지 클래스에 속하는 PV(영구 볼륨)의 동적 프로비저닝에 영향을 주는 매개변수를 설정합니다. **StorageClass** 오브젝트를 생성한 후에는 이 오브젝트의 매개변수를 업데이트할 수 없습니다.

HPP(hostpath provisioner)를 사용하려면 **storagePools** 스탠자를 사용하여 CSI 드라이버에 대한 관련 스토리지 클래스를 생성해야 합니다.



참고

가상 머신은 로컬 PV를 기반으로 하는 데이터 볼륨을 사용합니다. 로컬 PV는 특정 노드에 바인딩됩니다. 디스크 이미지는 가상 머신에서 사용할 수 있는 반면 가상 머신은 이전에 로컬 스토리지 PV가 고정된 노드에 예약할 수 없습니다.

이 문제를 해결하려면 Kubernetes Pod 스케줄러를 사용하여 올바른 노드의 PV에 PVC(영구 볼륨 클레임)를 바인딩합니다. **volumeBindingMode** 매개변수가 **WaitForFirstConsumer** 로 설정된 **StorageClass** 값을 사용하면 PVC를 사용하여 Pod를 생성할 때까지 PV의 바인딩 및 프로비저닝이 지연됩니다.

8.19.2.3.1. storagePools 스탠자를 사용하여 CSI 드라이버의 스토리지 클래스 생성

hostpath 프로비전 프로그램(HPP) CSI 드라이버에 대한 스토리지 클래스 CR(사용자 정의 리소스)을 생성합니다.

사전 요구 사항

- OpenShift Virtualization 4.10 이상이 있어야 합니다.

절차

1. **storageclass_csi.yaml** 파일을 생성하여 스토리지 클래스를 정의합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: hostpath-csi (1)
provisioner: kubevirt.io.hostpath-provisioner
reclaimPolicy: Delete (2)
volumeBindingMode: WaitForFirstConsumer (3)
parameters:
  storagePool: my-storage-pool (4)
```

(1) 스토리지 클래스에 의미 있는 이름을 할당합니다. 이 예에서 **csi** 는 클래스가 레거시 프로비저너 대신 CSI 프로비저너를 사용하도록 지정하는 데 사용됩니다. 기존 또는 CSI 드라이버 프로비저닝을 기반으로 스토리지 클래스에 대한 설명 이름을 선택하면 마이그레이션 전략을 쉽게 구현할 수 있습니다.

(2) **reclaimPolicy**에 사용할 수 있는 값은**Delete** 및 **Retain** 두 가지입니다. 값을 지정하지 않으면 기본값은 **Delete** 입니다.

(3) **volumeBindingMode** 매개변수는 동적 프로비저닝 및 볼륨 바인딩이 발생하는 시기를 결정합니다. PVC(영구 볼륨 클레임)를 사용하는 Pod가 생성된 후 PV(영구 볼륨)의 바인딩 및 프로비저닝을 지연하려면 **WaitForFirstConsumer** 를 지정합니다. 이렇게 하면 PV에서 Pod의 스케줄링 요구 사항을 충족할 수 있습니다.

(4) HPP CR에 정의된 스토리지 풀의 이름을 지정합니다.

2. 파일을 저장하고 종료합니다.

3. 다음 명령을 실행하여 **StorageClass** 오브젝트를 만듭니다.

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

8.19.2.3.2. 레거시 hostpath 프로비전 프로그램의 스토리지 클래스 생성

storagePool 매개변수 없이 **StorageClass** 오브젝트를 생성하여 레거시 hostpath 프로비전 프로그램(HPP)에 대한 스토리지 클래스를 생성합니다.

절차

1. **storageclass.yaml** 파일을 생성하여 스토리지 클래스를 정의합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: hostpath-provisioner
provisioner: kubevirt.io/hostpath-provisioner
reclaimPolicy: Delete (1)
volumeBindingMode: WaitForFirstConsumer (2)
```

(1) **reclaimPolicy**에 사용할 수 있는 값은**Delete** 및 **Retain** 두 가지입니다. 값을 지정하지 않는 경우 스토리지 클래스는 기본값인 **Delete**로 설정됩니다.

(2) **volumeBindingMode** 값은 동적 프로비저닝 및 볼륨 바인딩이 발생하는 시기를 결정합니다. PVC(영구 볼륨 클레임)를 사용하는 Pod가 생성된 후 영구 볼륨의 바인딩 및 프로비저닝을 지연하려면 **WaitForFirstConsumer** 값을 지정합니다. 이렇게 하면 PV에서 Pod의 스케줄링 요구 사항을 충족할 수 있습니다.

2. 파일을 저장하고 종료합니다.

3. 다음 명령을 실행하여 **StorageClass** 오브젝트를 만듭니다.

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

추가 리소스

- 스토리지 클래스

8.19.2.4. PVC 템플릿으로 생성된 스토리지 풀 정보

단일 대규모 PV(영구 볼륨)가 있는 경우HPP(사용자 정의 리소스) CR(사용자 정의 리소스)에서 PVC 템플릿을 정의하여 스토리지 풀을 생성할 수 있습니다.

PVC 템플릿으로 생성된 스토리지 풀에는 여러 HPP 볼륨이 포함될 수 있습니다. PV를 더 작은 볼륨으로 분할하면 데이터 할당에 대한 유연성이 향상됩니다.

PVC 템플릿은 **PersistentVolumeClaim** 오브젝트의 **spec** 스탠자를 기반으로 합니다.

PersistentVolumeClaim 오브젝트의 예

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: iso-pvc
spec:
  volumeMode: Block (1)
  storageClassName: my-storage-class
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

(1) 이 값은 블록 볼륨 모드 PV에만 필요합니다.

HPP CR에서 **pvcTemplate** 사양을 사용하여 스토리지 풀을 정의합니다. Operator는 HPP CSI 드라이버가 포함된 각 노드의 **pvcTemplate** 사양에서 PVC를 생성합니다. PVC 템플릿에서 생성된 PVC는 하나의 대규모 PV를 사용하므로 HPP가 더 작은 동적 볼륨을 생성할 수 있습니다.

기본 스토리지 풀을 PVC 템플릿에서 생성된 스토리지 풀과 결합할 수 있습니다.

8.19.2.4.1. PVC 템플릿을 사용하여 스토리지 풀 생성

HPP CR(사용자 정의 리소스)에 PVC 템플릿을 지정하여 여러 hostspath 프로비전 프로그램(HPP) 볼륨에 대한 스토리지 풀을 생성할 수 있습니다.

사전 요구 사항

- **spec.storagePools.path** 에 지정된 디렉터리에는 읽기/쓰기 액세스 권한이 있어야 합니다.
- 스토리지 풀은 운영 체제와 동일한 파티션에 있을 수 없습니다. 그렇지 않으면 운영 체제 파티션이 용량으로 채워질 수 있으며 이로 인해 성능에 영향을 주거나 노드가 불안정하거나 사용할 수 없게 됩니다.

절차

1. 다음 예에 따라 **storagePools** 스탠자에서 PVC(영구 볼륨) 템플릿을 지정하는 HPP CR에 **hpp_pvc_template_pool.yaml** 파일을 생성합니다.

```
apiVersion: hostpathprovisioner.kubevirt.io/v1beta1
kind: HostPathProvisioner
metadata:
  name: hostpath-provisioner
spec:
  imagePullPolicy: IfNotPresent
  storagePools: (1)
  - name: my-storage-pool
    path: "/var/myvolumes" (2)
    pvcTemplate:
      volumeMode: Block (3)
      storageClassName: my-storage-class (4)
      accessModes:
      - ReadWriteOnce
```

```
    resources:
      requests:
        storage: 5Gi 5
  workload:
    nodeSelector:
      kubernetes.io/os: linux
```

1. **storagePools** 스탠자는 기본 및 PVC 템플릿 스토리지 풀을 모두 포함할 수 있는 배열입니다.
2. 이 노드 경로에 스토리지 풀 디렉터리를 지정합니다.
3. 선택 사항: **volumeMode** 매개변수는 프로비저닝된 볼륨 형식과 일치하는 경우**Block** 또는 **Filesystem** 일 수 있습니다. 값을 지정하지 않으면 기본값은**Filesystem** 입니다. **volumeMode** 가 **Block** 인 경우 마운트 Pod는 마운트하기 전에 블록 볼륨에 XFS 파일 시스템을 생성합니다.
4. storageClassName 매개변수 가 생략되면 기본 스토리지 클래스가 PVC를 생성하는 데 사용됩니다. storageClassName을 생략하면 HPP 스토리지 클래스가 기본 스토리지 클래스가 아닌지 확인합니다.
5. 정적으로 또는 동적으로 프로비저닝된 스토리지를 지정할 수 있습니다. 두 경우 모두 요청된 스토리지 크기가 사실상 분할하려는 볼륨에 적합한지 확인하십시오. 그렇지 않으면 PVC를 대규모 PV에 바인딩할 수 없습니다. 사용 중인 스토리지 클래스가 동적으로 프로비저닝된 스토리지를 사용하는 경우 일반적인 요청의 크기와 일치하는 할당 크기를 선택합니다.

2. 파일을 저장하고 종료합니다.
3. 다음 명령을 실행하여 스토리지 풀로 HPP를 생성합니다.

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

추가 리소스

- 스토리지 프로파일 사용자 정의

8.19.3. 데이터 볼륨 생성

데이터 볼륨을 생성할 때 CDI(Containerized Data Importer)는 PVC(영구 볼륨 클레임)를 생성하고 PVC를 데이터로 채웁니다. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 가상 머신 사양에 **dataVolumeTemplate** 리소스를 사용하여 생성할 수 있습니다. PVC API 또는 스토리지 API를 사용하여 데이터 볼륨을 생성합니다.

중요

OpenShift Container Platform 컨테이너 스토리지에서 OpenShift Virtualization을 사용하는 경우 가상 머신 디스크를 생성할 때 RBD 블록 모드 PVC(영구 볼륨 클레임)를 지정합니다. 가상 머신 디스크에서는 RBD 블록 모드 볼륨이 더 효율적이며 Ceph FS 또는 RBD 파일 시스템 모드 PVC보다 더 나은 성능을 제공합니다.

RBD 블록 모드 PVC를 지정하려면 'ocs-storagecluster-ceph-rbd' 스토리지 클래스와 **VolumeMode: Block** 을 사용합니다.

작은 정보

가능한 경우 스토리지 API를 사용하여 공간 할당을 최적화하고 성능을 극대화합니다.

스토리지 프로필은 CDI가 관리하는 사용자 지정 리소스입니다. 관련 스토리지 클래스를 기반으로 권장 스토리지 설정을 제공합니다. 각 스토리지 클래스에 대해 스토리지 프로필이 할당됩니다.

스토리지 프로필을 사용하면 데이터 볼륨을 빠르게 생성하고 코딩을 줄이고 잠재적인 오류를 최소화할 수 있습니다.

인식된 스토리지 유형의 경우 CDI는 PVC 생성을 최적화하는 값을 제공합니다. 그러나 스토리지 프로필을 사용자 지정하는 경우 스토리지 클래스에 대한 자동 설정을 구성할 수 있습니다.

8.19.3.1. 스토리지 API를 사용하여 데이터 볼륨 생성

스토리지 API를 사용하여 데이터 볼륨을 생성할 때 CDI(Containerized Data Interface)는 선택한 스토리지 클래스에서 지원하는 스토리지 유형에 따라 PVC(영구 볼륨 클레임) 할당을 최적화합니다. 데이터 볼륨 이름, 네임스페이스 및 할당할 스토리지 크기만 지정해야 합니다.

예를 들면 다음과 같습니다.

- Ceph RBD를 사용하는 경우 **accessModes**가 자동으로 **ReadWriteMany**로 설정되어 실시간 마이그레이션이 가능합니다. **volumeMode**가 **Block**으로 설정되어 성능을 극대화합니다.
- **volumeMode: Filesystem**을 사용하는 경우 파일 시스템 오버헤드를 수용하기 위해 필요한 경우 CDI에서 자동으로 더 많은 공간을 요청합니다.

다음 YAML에서 스토리지 API를 사용하면 사용 가능한 공간이 2GB인 데이터 볼륨을 요청합니다. 사용자가 필요한 PVC(영구 볼륨 클레임) 크기를 올바르게 추정하기 위해 **volumeMode**를 알 필요가 없습니다. CDI는 **accessModes** 및 **volumeMode** 속성의 최적 조합을 자동으로 선택합니다. 이러한 최적 값은 스토리지 유형 또는 스토리지 프로필에 정의된 기본값을 기반으로 합니다. 사용자 지정 값을 제공하려면 시스템 단위로 계산된 값을 재정의합니다.

데이터 볼륨 정의 예

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <datavolume> 1
spec:
  source:
    pvc: 2
      namespace: "<source_namespace>" 3
      name: "<my_vm_disk>" 4
  storage: 5
    resources:
      requests:
        storage: 2Gi 6
    storageClassName: <storage_class> 7
```

1 새 데이터 볼륨의 이름입니다.

2 가져오기 소스가 기존 PVC(영구 볼륨 클레임)임을 나타냅니다.

3. 소스 PVC가 존재하는 네임스페이스입니다.
4. 소스 PVC의 이름입니다.
5. 스토리지 API를 사용하여 할당을 나타냅니다.
6. PVC에 요청한 사용 가능한 공간의 양을 지정합니다.
7. 선택 사항: 스토리지 클래스의 이름입니다. 스토리지 클래스를 지정하지 않으면 시스템 기본 스토리지 클래스가 사용됩니다.

8.19.3.2. PVC API를 사용하여 데이터 볼륨 생성

PVC API를 사용하여 데이터 볼륨을 생성할 때 CDI(Containerized Data Interface)는 다음 필드에 지정된 값을 기반으로 데이터 볼륨을 생성합니다.

- **accessModes** (**ReadWriteOnce**,**ReadWriteMany** 또는 **ReadOnlyMany**)
- **volumeMode** (**Filesystem** 또는 **Block**)
- **storage**의 **capacity** (예:**5Gi**)

다음 YAML에서 PVC API를 사용하면 2GB의 스토리지 용량으로 데이터 볼륨을 할당합니다. 실시간 마이그레이션을 활성화하려면 **ReadWriteMany**의 액세스 모드를 지정합니다. 시스템에서 지원할 수 있는 값을 알고 있으므로 기본값인 **Filesystem** 대신 **Block** 스토리지를 지정합니다.

데이터 볼륨 정의 예

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <datavolume> 1
spec:
  source:
    pvc: 2
      namespace: "<source_namespace>" 3
      name: "<my_vm_disk>" 4
  pvc: 5
    accessModes: 6
      - ReadWriteMany
    resources:
      requests:
        storage: 2Gi 7
    volumeMode: Block 8
    storageClassName: <storage_class> 9
```

1. 새 데이터 볼륨의 이름입니다.
2. **source** 섹션에서 **pvc**는 가져오기 소스가 기존 PVC(영구 볼륨 클레임)임을 나타냅니다.
3. 소스 PVC가 존재하는 네임스페이스입니다.
4. 소스 PVC의 이름입니다.

5 PVC API를 사용하여 할당을 나타냅니다.

6 PVC API를 사용하는 경우 **accessModes**가 필요합니다.

7 데이터 볼륨에 요청 중인 공간의 양을 지정합니다.

8 대상이 블록 PVC임을 지정합니다.

9 선택 사항으로 스토리지 클래스를 지정합니다. 스토리지 클래스를 지정하지 않으면 시스템 기본 스토리지 클래스가 사용됩니다.

중요

PVC API를 사용하여 명시적으로 데이터 볼륨을 할당하고 **volumeMode: Block**을 사용하지 않는 경우 파일 시스템 오버헤드를 고려합니다.

파일 시스템 오버헤드는 메타데이터를 유지 관리하기 위해 파일 시스템에 필요한 공간입니다. 파일 시스템 메타데이터에 필요한 공간 크기는 파일 시스템에 따라 다릅니다. 스토리지 용량 요청의 파일 시스템 오버헤드를 고려하지 않으면 가상 머신 디스크를 수용하기에 충분하지 않은 기본 PVC(영구 볼륨 클레임)가 발생할 수 있습니다.

스토리지 API를 사용하는 경우 CDI는 파일 시스템 오버헤드를 인수하고 더 큰 PVC(영구 볼륨 클레임)를 요청하여 할당 요청이 성공했는지 확인합니다.

8.19.3.3. 스토리지 프로파일 사용자 정의

프로비저너의 스토리지 클래스에 대해 **StorageProfile** 오브젝트를 편집하여 기본 매개변수를 지정할 수 있습니다. 이러한 기본 매개변수는 **DataVolume** 오브젝트에 구성되지 않은 경우에만 PVC(영구 볼륨 클레임)에 적용됩니다.

사전 요구 사항

- 계획된 구성이 스토리지 클래스 및 해당 공급자에 의해 지원되는지 확인하십시오. 스토리지 프로필에 호환되지 않는 구성을 지정하면 볼륨 프로비저닝이 실패합니다.

참고

스토리지 프로필의 빈 **status** 섹션은 스토리지 프로비저너가 CDI(Containerized Data Interface)에서 인식되지 않았음을 나타냅니다. CDI에서 인식하지 않는 스토리지 프로비저너가 있는 경우 스토리지 프로필을 사용자 정의해야 합니다. 이 경우 관리자는 스토리지 프로필에 적절한 값을 설정하여 성공적으로 할당되도록 합니다.

주의

데이터 볼륨을 생성하고 YAML 속성을 생략하고 이러한 특성이 스토리지 프로필에 정의되지 않으면 요청된 스토리지가 할당되지 않고 기본 PVC(영구 볼륨 클레임)가 생성되지 않습니다.

절차

1. 스토리지 프로파일을 편집합니다. 이 예에서 CDI는 제공자를 인식하지 못합니다.

```
$ oc edit -n openshift-cnv storageprofile <storage_class>
```

스토리지 프로필 예

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: StorageProfile
metadata:
  name: <unknown_provisioner_class>
#   ...
spec: {}
status:
  provisioner: <unknown_provisioner>
  storageClass: <unknown_provisioner_class>
```

2. 스토리지 프로파일에 필요한 속성 값을 제공합니다.

스토리지 프로필 예

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: StorageProfile
metadata:
  name: <unknown_provisioner_class>
#   ...
spec:
  claimPropertySets:
  - accessModes:
    - ReadWriteOnce (1)
    volumeMode:
      Filesystem (2)
status:
  provisioner: <unknown_provisioner>
  storageClass: <unknown_provisioner_class>
```

(1) 선택한 **accessModes**입니다.

(2) 선택한 **volumeMode**입니다.

변경 사항을 저장하면 선택한 값이 스토리지 프로필 **status** 요소에 표시됩니다.

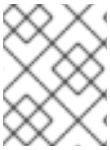
8.19.3.3.1. 스토리지 프로필을 사용하여 기본 복제 전략 설정

스토리지 프로필을 사용하여 스토리지 클래스의 기본 복제 방법을 설정하여 *복제 전략*을 생성할 수 있습니다. 예를 들어, 스토리지 벤더가 특정 복제 방법만 지원하는 경우 복제 전략을 설정하면 유용할 수 있습니다. 또한 리소스 사용을 제한하거나 성능을 극대화하는 방법을 선택할 수 있습니다.

스토리지 프로필의 **cloneStrategy** 특성을 다음 값 중 하나로 설정하여 전략을 복제할 수 있습니다.

- **snapshot** - 이 방법은 스냅샷이 구성될 때 기본적으로 사용됩니다. 이 복제 전략에서는 임시 볼륨 스냅샷을 사용하여 볼륨을 복제합니다. 스토리지 프로비저너는 CSI 스냅샷을 지원해야 합니다.
- **copy** - 이 방법은 소스 Pod와 대상 Pod를 사용하여 소스 볼륨의 데이터를 대상 볼륨으로 복사합니다. 호스트 지원 복제는 가장 효율적인 복제 방법입니다.

- **CSI-clone** - 이 방법은 CSI 복제 API를 사용하여 임시 볼륨 스냅샷을 사용하지 않고 기존 볼륨을 효율적으로 복제합니다. 스토리지 프로파일이 정의되지 않은 경우 기본적으로 사용되는 **snapshot** 또는 **copy**와 달리 CSI 볼륨 복제는 프로비저너의 스토리지 클래스에 대해**StorageProfile** 오브젝트에 지정된 경우에만 사용됩니다.



참고

YAML **spec** 섹션의 기본**claimPropertySets** 를 수정하지 않고 CLI를 사용하여 복제 전략을 설정할 수도 있습니다.

스토리지 프로필 예

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: StorageProfile
metadata:
  name: <provisioner_class>
#   ...
spec:
  claimPropertySets:
  - accessModes:
    - ReadWriteOnce (1)
    volumeMode:
      Filesystem (2)
  cloneStrategy:
  csi-clone (3)
status:
  provisioner: <provisioner>
  storageClass: <provisioner_class>
```

(1) 선택한 **accessModes**입니다.

(2) 선택한 **volumeMode**입니다.

(3) 선택한 기본 복제 방법입니다. 이 예에서는 CSI 볼륨 복제가 지정되어 있습니다.

8.19.3.4. 추가 리소스

- 스토리지 클래스 생성 정보
- 스마트 복제를 사용하여 데이터 볼륨 복제

8.19.4. 컴퓨팅 리소스 할당량이 있는 네임스페이스를 사용하도록 CDI 구성

CDI(Containerized Data Importer)를 사용하면 CPU 및 메모리 리소스가 제한된 네임스페이스로 가상 머신 디스크를 가져오고, 업로드하고, 복제할 수 있습니다.

8.19.4.1. 네임스페이스의 CPU 및 메모리 할당량 정보

ResourceQuota 오브젝트로 정의하는 *리소스 할당량*은 네임스페이스에 제한을 적용하여 해당 네임스페이스 내의 리소스에서 사용할 수 있는 총 컴퓨팅 리소스 양을 제한합니다.

HyperConverged 사용자 지정 리소스 (CR)는 CDI(Containerized Data Importer)에 대한 사용자 구성을 정의합니다. CPU 및 메모리 요청 및 한계 값은 기본값인 **0**으로 설정되어 있습니다. 이렇게 하면 컴퓨팅 리소

스 요구 사항 없이 CDI에서 생성한 Pod에 기본값을 제공하고 할당량으로 제한되는 네임스페이스에서 해당 Pod를 실행할 수 있습니다.

8.19.4.2. CPU 및 메모리 기본값 덮어쓰기

spec.resourceRequirements.storageWorkloads 스탠자를 **HyperConverged** CR(사용자 정의 리소스)에 추가하여 CPU 및 메모리 요청의 기본 설정과 사용 사례에 대한 제한을 수정합니다.

사전 요구 사항

- OpenShift CLI(**oc**)를 설치합니다.

절차

1. 다음 명령을 실행하여 **HyperConverged** CR을 편집합니다.

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

2. **spec.resourceRequirements.storageWorkloads** 스탠자를 CR에 추가하여 사용 사례에 따라 값을 설정합니다. 예를 들면 다음과 같습니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
spec:
  resourceRequirements:
    storageWorkloads:
      limits:
        cpu: "500m"
        memory: "2Gi"
      requests:
        cpu: "250m"
        memory: "1Gi"
```

3. 편집기를 저장하고 종료하여 **HyperConverged** CR을 업데이트합니다.

8.19.4.3. 추가 리소스

- 프로젝트당 리소스 할당량

8.19.5. 데이터 볼륨 주석 관리

DV(데이터 볼륨) 주석을 사용하면 Pod 동작을 관리할 수 있습니다. 하나 이상의 주석을 데이터 볼륨에 추가하면 생성된 가져오기 Pod로 전파할 수 있습니다.

8.19.5.1. 예: 데이터 볼륨 주석

이 예에서는 가져오기 Pod에서 사용하는 네트워크를 제어하도록 DV(데이터 볼륨) 주석을 구성할 수 있는 방법을 보여줍니다. **v1.multus-cni.io/default-network: bridge-network** 주석을 사용하면 Pod에서 **bridge-network**라는 multus 네트워크를 기본 네트워크로 사용합니다. 가져오기 Pod에서 클러스터 및 보조 multus 네트워크의 기본 네트워크를 모두 사용하도록 하려면 **k8s.v1.cni.cncf.io/networks: <network_name>** 주석을 사용합니다.

Multus 네트워크 주석 예

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: dv-ann
  annotations:
    v1.multus-cni.io/default-network: bridge-network (1)
spec:
  source:
    http:
      url: "example.exampleurl.com"
  pvc:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 1Gi
```

(1) Multus 네트워크 주석

8.19.6. 데이터 볼륨에 사전 할당 사용

CDI(Containerized Data Importer)는 데이터 볼륨을 생성할 때 쓰기 성능을 개선하기 위해 디스크 공간을 사전 할당할 수 있습니다.

특정 데이터 볼륨에 대해 사전 할당을 실행할 수 있습니다.

8.19.6.1. 사전 할당 정보

CDI(Containerized Data Importer)는 데이터 볼륨에 QEMU 사전 할당 모드를 사용하여 쓰기 성능을 향상시킬 수 있습니다. 사전 할당 모드를 사용하여 작업 가져오기 및 업로드 및 빈 데이터 볼륨을 생성할 때 사용할 수 있습니다.

사전 할당이 활성화된 경우 CDI는 기본 파일 시스템 및 장치 유형에 따라 더 나은 사전 할당 방법을 사용합니다.

fallocate

파일 시스템이 이를 지원하는 경우, CDI는 **posix_fallocate** 함수를 사용하여 운영 체제의 **fallocate** 호출을 통해 공간을 미리 할당하며, 이를 통해 블록을 할당하고 초기화되지 않음으로 표시합니다.

full

fallocate 모드를 사용할 수 없는 경우 **full** 모드는 기본 스토리지에 데이터를 작성하여 이미지의 공간을 할당합니다. 스토리지 위치에 따라, 비어 있는 할당된 모든 공간을 0으로 만들 수 있습니다.

8.19.6.2. 데이터 볼륨 사전 할당 활성화

데이터 볼륨 매니페스트에 **spec.preallocation** 필드를 포함하여 특정 데이터 볼륨에 대한 사전 할당을 활성화할 수 있습니다. 웹 콘솔에서 또는 OpenShift 클라이언트(**oc**)를 사용하여 사전 할당 모드를 활성화할 수 있습니다.

모든 CDI 소스 유형에서 사전 할당 모드가 지원됩니다.

절차

- 데이터 볼륨 매니페스트에 **spec.preallocation** 필드를 지정합니다.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: preallocated-datavolume
spec:
  source: 1
    ...
  pvc:
    ...
  preallocation: true 2
```

1. 모든 CDI 소스 유형은 사전 할당을 지원하지만 복제 작업에서는 사전 할당이 무시됩니다.
2. **preallocation** 필드는 기본값이 false인 부울입니다.

8.19.7. 웹 콘솔을 사용하여 로컬 디스크 이미지 업로드

웹 콘솔을 사용하여 로컬에 저장된 디스크 이미지 파일을 업로드할 수 있습니다.

8.19.7.1. 사전 요구 사항

- 가상 머신 이미지 파일이 IMG, ISO 또는 QCOW2 형식이어야 합니다.
- CDI 지원 작업 매트릭스에 따라 스크래치 공간이 필요한 경우, 이 작업을 완료하기 위해서는 먼저 스토리지 클래스를 정의하거나 CDI 스크래치 공간을 준비해야 합니다.

8.19.7.2. CDI 지원 작업 매트릭스

이 매트릭스에는 끝점에 대한 콘텐츠 유형에 따라 지원되는 CDI 작업과 이러한 작업 중 스크래치 공간이 필요한 작업이 표시되어 있습니다.

콘텐츠 유형	HTTP	HTTPS	HTTP 기본 인증	레지스트리	업로드
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt(RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 지원되는 작업

☐ 지원되지 않는 작업

* 스크래치 공간 필요

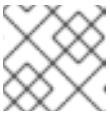
** 사용자 정의 인증 기관이 필요한 경우 스크래치 공간 필요

8.19.7.3. 웹 콘솔을 사용하여 이미지 파일 업로드

웹 콘솔을 사용하여 이미지 파일을 새 PVC(영구 볼륨 클레임)에 업로드합니다. 나중에 이 PVC를 사용하여 이미지를 새 가상 머신에 연결할 수 있습니다.

사전 요구 사항

- 다음 중 하나가 있어야 합니다.
  - ISO 또는 IMG 형식의 원시 가상 머신 이미지 파일
  - QCOW2 형식의 가상 머신 이미지 파일
- 최상의 결과를 얻으려면 업로드하기 전에 다음 지침에 따라 이미지 파일을 압축하십시오.
  - **xz** 또는 **gzip**을 사용하여 원시 이미지 파일을 압축합니다.

    참고

    압축된 원시 이미지 파일을 사용할 때 가장 효율적으로 업로드할 수 있습니다.

  - 클라이언트에 권장되는 방법을 사용하여 QCOW2 이미지 파일을 압축합니다.
    - Linux 클라이언트를 사용하는 경우 virt-sparsify 툴을 사용하여 QCOW2 파일을 *스파스(sparsify) 형식으로 변환*합니다.
    - Windows 클라이언트를 사용하는 경우 **xz** 또는 **gzip**을 사용하여 QCOW2 파일을 압축합니다.

절차

1. 웹 콘솔의 사이드 메뉴에서 스토리지 → 영구 볼륨 클레임을 클릭합니다.
2. 영구 볼륨 클레임 생성 드롭다운 목록을 클릭하여 확장합니다.
3. 사용할 데이터 업로드 폼을 클릭하여 영구 볼륨 클레임에 데이터 업로드 페이지를 엽니다.
4. 찾아보기를 클릭하여 파일 관리자를 열고 업로드할 이미지를 선택하거나,파일을 여기로 드래그하거나 업로드할 항목 찾아보기 필드로 파일을 드래그합니다.
5. 선택 사항: 이 이미지를 특정 운영 체제의 기본 이미지로 설정합니다.
   a. 이 데이터를 가상 머신 운영 체제에 연결확인란을 선택합니다.
   b. 목록에서 운영 체제를 선택합니다.
6. 영구 볼륨 클레임 이름 필드는 고유한 이름으로 자동으로 채워지며 편집할 수 없습니다. 필요한 경우 나중에 확인할 수 있도록 PVC에 지정된 이름을 기록해 두십시오.
7. 스토리지 클래스 목록에서 스토리지 클래스를 선택합니다.
8. 크기 필드에 PVC 크기 값을 입력합니다. 드롭다운 목록에서 해당 측정 단위를 선택합니다.

주의

PVC 크기는 압축되지 않은 가상 디스크의 크기보다 커야 합니다.

9. 선택한 스토리지 클래스와 일치하는 액세스 모드를 선택합니다.
10. 업로드를 클릭합니다.

8.19.7.4. 추가 리소스

- 데이터 볼륨 작업에 대한 쓰기 성능을 향상하기 위해 사전 할당 모드를 구성합니다.

8.19.8. virtctl 툴을 사용하여 로컬 디스크 이미지 업로드

virtctl 명령줄 유틸리티를 사용하여 로컬에 저장된 디스크 이미지를 신규 또는 기존 데이터 볼륨에 업로드할 수 있습니다.

8.19.8.1. 사전 요구 사항

- **kubevirt-virtctl** 패키지를 활성화합니다.
- CDI 지원 작업 매트릭스에 따라 스크래치 공간이 필요한 경우, 이 작업을 완료하기 위해서는 먼저 스토리지 클래스를 정의하거나 CDI 스크래치 공간을 준비해야 합니다.

8.19.8.2. 데이터 볼륨 정보

Dataolume 오브젝트는 CDI(Containerized Data Importer) 프로젝트에서 제공하는 사용자 정의 리소스입니다. 데이터 볼륨은 기본 PVC(영구 볼륨 클레임)와 관련된 가져오기, 복제, 업로드 작업을 오케스트레이션합니다. 데이터 볼륨은 OpenShift Virtualization과 통합되며 PVC가 준비되기 전에 가상 머신이 시작되지 않도록 합니다.

8.19.8.3. 업로드 데이터 볼륨 생성

로컬 디스크 이미지를 업로드하는 데 사용할 **upload** 데이터 소스가 있는 데이터 볼륨을 수동으로 생성할 수 있습니다.

절차

1. **spec: source: upload{}**를 지정하는 데이터 볼륨 구성을 만듭니다.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <upload-datavolume> 1
spec:
  source:
    upload: {}
  pvc:
    accessModes:
```

```
    - ReadWriteOnce
  resources:
    requests:
      storage: <2Gi> 2
```

1 데이터 볼륨의 이름입니다.

2 데이터 볼륨의 크기입니다. 이 값은 업로드하는 디스크의 크기와 같거나 커야 합니다.

2. 다음 명령을 실행하여 데이터 볼륨을 생성합니다.

```
$ oc create -f <upload-datavolume>.yaml
```

8.19.8.4. 데이터 볼륨에 로컬 디스크 이미지 업로드

virtctl CLI 유틸리티를 사용하여 클라이언트 머신의 로컬 디스크 이미지를 클러스터의 DV(데이터 볼륨)에 업로드할 수 있습니다. 클러스터에 이미 존재하는 DV를 사용하거나 이 절차 중에 새 DV를 만들 수 있습니다.

참고

로컬 디스크 이미지를 업로드한 후 가상 머신에 추가할 수 있습니다.

사전 요구 사항

- 다음 중 하나가 있어야 합니다.
  - ISO 또는 IMG 형식의 원시 가상 머신 이미지 파일
  - QCOW2 형식의 가상 머신 이미지 파일
- 최상의 결과를 얻으려면 업로드하기 전에 다음 지침에 따라 이미지 파일을 압축하십시오.
  - **xz** 또는 **gzip**을 사용하여 원시 이미지 파일을 압축합니다.

    참고

    압축된 원시 이미지 파일을 사용할 때 가장 효율적으로 업로드할 수 있습니다.

  - 클라이언트에 권장되는 방법을 사용하여 QCOW2 이미지 파일을 압축합니다.
    - Linux 클라이언트를 사용하는 경우 virt-sparsify 툴을 사용하여 QCOW2 파일을 *스파스(sparsify) 형식으로 변환*합니다.
    - Windows 클라이언트를 사용하는 경우 **xz** 또는 **gzip**을 사용하여 QCOW2 파일을 압축합니다.
- **kubevirt-virtctl** 패키지가 클라이언트 머신에 설치되어 있어야 합니다.
- 클라이언트 머신이 OpenShift Container Platform 라우터의 인증서를 신뢰하도록 구성되어 있어야 합니다.

절차

1. 다음 항목을 확인합니다.

- 사용할 업로드 데이터 볼륨의 이름. 이 데이터 볼륨이 없으면 자동으로 생성됩니다.
- 업로드 절차 중 데이터 볼륨을 생성하려는 경우 데이터 볼륨의 크기. 크기는 디스크 이미지의 크기보다 크거나 같아야 합니다.
- 업로드하려는 가상 머신 디스크 이미지의 파일 위치.

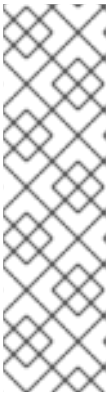
2. **virtctl image-upload** 명령을 실행하여 디스크 이미지를 업로드합니다. 이전 단계에서 확인한 매개변수를 지정합니다. 예를 들면 다음과 같습니다.

```
$ virtctl image-upload dv <datavolume_name> \ 1
--size=<datavolume_size> \ 2
--image-path=</path/to/image> \ 3
```

1 데이터 볼륨의 이름입니다.

2 데이터 볼륨의 크기입니다. 예를 들면**--size=500Mi, --size=1G**와 같습니다.

3 가상 머신 디스크 이미지의 파일 경로입니다.



참고

- 새 데이터 볼륨을 생성하지 않으려면 **--size** 매개변수를 생략하고 **--no-create** 플래그를 포함합니다.
- 디스크 이미지를 PVC에 업로드할 때 PVC 크기는 압축되지 않은 가상 디스크의 크기보다 커야 합니다.
- HTTPS를 사용할 때 비보안 서버 연결을 허용하려면 **--insecure** 매개변수를 사용하십시오. **--insecure** 플래그를 사용하면 업로드 끝점의 신뢰성이 확인되지 않습니다.

3. 선택사항입니다. 데이터 볼륨이 생성되었는지 확인하려면 다음 명령을 실행하여 모든 데이터 볼륨을 확인합니다.

```
$ oc get dvs
```

8.19.8.5. CDI 지원 작업 매트릭스

이 매트릭스에는 끝점에 대한 콘텐츠 유형에 따라 지원되는 CDI 작업과 이러한 작업 중 스크래치 공간이 필요한 작업이 표시되어 있습니다.

콘텐츠 유형	HTTP	HTTPS	HTTP 기본 인증	레지스트리	업로드
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*

콘텐츠 유형	HTTP	HTTPS	HTTP 기본 인증	레지스트리	업로드
KubeVirt(RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* □ GZ □ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 지원되는 작업

□ 지원되지 않는 작업

* 스크래치 공간 필요

** 사용자 정의 인증 기관이 필요한 경우 스크래치 공간 필요

8.19.8.6. 추가 리소스

- 데이터 볼륨 작업에 대한 쓰기 성능을 향상하기 위해 사전 할당 모드를 구성합니다.

8.19.9. 블록 스토리지 데이터 볼륨에 로컬 디스크 이미지 업로드

virtctl 명령줄 유틸리티를 사용하여 로컬 디스크 이미지를 블록 데이터 볼륨에 업로드할 수 있습니다.

이 워크플로우에서는 영구 볼륨으로 사용할 로컬 블록 장치를 생성한 후 이 블록 볼륨을 **upload** 데이터 볼륨과 연결하고, **virtctl**을 사용하여 로컬 디스크 이미지를 데이터 볼륨에 업로드합니다.

8.19.9.1. 사전 요구 사항

- **kubevirt-virtctl** 패키지를 활성화합니다.
- CDI 지원 작업 매트릭스에 따라 스크래치 공간이 필요한 경우, 이 작업을 완료하기 위해서는 먼저 스토리지 클래스를 정의하거나 CDI 스크래치 공간을 준비해야 합니다.

8.19.9.2. 데이터 볼륨 정보

Dataolume 오브젝트는 CDI(Containerized Data Importer) 프로젝트에서 제공하는 사용자 정의 리소스입니다. 데이터 볼륨은 기본 PVC(영구 볼륨 클레임)와 관련된 가져오기, 복제, 업로드 작업을 오케스트레이션합니다. 데이터 볼륨은 OpenShift Virtualization과 통합되며 PVC가 준비되기 전에 가상 머신이 시작되지 않도록 합니다.

8.19.9.3. 블록 영구 볼륨 정보

PV(블록 영구 볼륨)는 원시 블록 장치에서 지원하는 PV입니다. 이러한 볼륨은 파일 시스템이 없으며 오버헤드를 줄여 가상 머신의 성능을 향상시킬 수 있습니다.

원시 블록 볼륨은 PV 및 PVC(영구 볼륨 클레임) 사양에 **volumeMode:Block**을 지정하여 프로비저닝합니다.

8.19.9.4. 로컬 블록 영구 볼륨 생성

파일을 채우고 루프 장치로 마운트하여 노드에 로컬 블록 PV(영구 볼륨)를 생성합니다. 그런 다음 PV 매니페스트에서 이 루프 장치를 **Block** 볼륨으로 참조하고 가상 머신 이미지의 블록 장치로 사용할 수 있습니다.

절차

1. 로컬 PV를 생성할 노드에 **root**로 로그인합니다. 이 절차에서는 예제로**node01**을 사용합니다.
2. 블록 장치로 사용할 수 있도록 파일을 생성하고 null 문자로 채웁니다. 다음 예제에서는 크기가 2Gb(20X100Mb 블록)인 파일 **loop10**을 생성합니다.

```
$ dd if=/dev/zero of=<loop10> bs=100M count=20
```

3. **loop10** 파일을 루프 장치로 마운트합니다.

```
$ losetup </dev/loop10>d3 <loop10> 1 2
```

1 루프 장치가 마운트된 파일 경로입니다.

2 이전 단계에서 생성된 파일은 루프 장치로 마운트됩니다.

4. 마운트된 루프 장치를 참조하는 **PersistentVolume** 매니페스트를 생성합니다.

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: <local-block-pv10>
  annotations:
spec:
  local:
    path: </dev/loop10> 1
  capacity:
    storage: <2Gi>
  volumeMode: Block 2
  storageClassName: local 3
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - <node01> 4
```

1 노드에 있는 루프 장치의 경로입니다.

2 블록 PV임을 나타냅니다.

3 선택 사항: PV의 스토리지 클래스를 설정합니다. 생략하면 클러스터 기본값이 사용됩니다.

4 블록 장치가 마운트된 노드입니다.

5. 블록 PV를 생성합니다.

```
# oc create -f <local-block-pv10.yaml> 1
```

1 이전 단계에서 생성한 영구 볼륨의 파일 이름입니다.

8.19.9.5. 업로드 데이터 볼륨 생성

로컬 디스크 이미지를 업로드하는 데 사용할 **upload** 데이터 소스가 있는 데이터 볼륨을 수동으로 생성할 수 있습니다.

절차

1. **spec: source: upload{}**를 지정하는 데이터 볼륨 구성을 만듭니다.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <upload-datavolume> 1
spec:
  source:
    upload: {}
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: <2Gi> 2
```

1 데이터 볼륨의 이름입니다.

2 데이터 볼륨의 크기입니다. 이 값은 업로드하는 디스크의 크기와 같거나 커야 합니다.

2. 다음 명령을 실행하여 데이터 볼륨을 생성합니다.

```
$ oc create -f <upload-datavolume>.yaml
```

8.19.9.6. 데이터 볼륨에 로컬 디스크 이미지 업로드

virtctl CLI 유틸리티를 사용하여 클라이언트 머신의 로컬 디스크 이미지를 클러스터의 DV(데이터 볼륨)에 업로드할 수 있습니다. 클러스터에 이미 존재하는 DV를 사용하거나 이 절차 중에 새 DV를 만들 수 있습니다.

참고

로컬 디스크 이미지를 업로드한 후 가상 머신에 추가할 수 있습니다.

사전 요구 사항

- 다음 중 하나가 있어야 합니다.
  - ISO 또는 IMG 형식의 원시 가상 머신 이미지 파일
  - QCOW2 형식의 가상 머신 이미지 파일

- 최상의 결과를 얻으려면 업로드하기 전에 다음 지침에 따라 이미지 파일을 압축하십시오.
  - **xz** 또는 **gzip**을 사용하여 원시 이미지 파일을 압축합니다.

    참고

    압축된 원시 이미지 파일을 사용할 때 가장 효율적으로 업로드할 수 있습니다.

  - 클라이언트에 권장되는 방법을 사용하여 QCOW2 이미지 파일을 압축합니다.
    - Linux 클라이언트를 사용하는 경우 virt-sparsify 툴을 사용하여 QCOW2 파일을 *스파스(sparsify) 형식으로 변환*합니다.
    - Windows 클라이언트를 사용하는 경우 **xz** 또는 **gzip**을 사용하여 QCOW2 파일을 압축합니다.
- **kubevirt-virtctl** 패키지가 클라이언트 머신에 설치되어 있어야 합니다.
- 클라이언트 머신이 OpenShift Container Platform 라우터의 인증서를 신뢰하도록 구성되어 있어야 합니다.

절차

1. 다음 항목을 확인합니다.
   - 사용할 업로드 데이터 볼륨의 이름. 이 데이터 볼륨이 없으면 자동으로 생성됩니다.
   - 업로드 절차 중 데이터 볼륨을 생성하려는 경우 데이터 볼륨의 크기. 크기는 디스크 이미지의 크기보다 크거나 같아야 합니다.
   - 업로드하려는 가상 머신 디스크 이미지의 파일 위치.
2. **virtctl image-upload** 명령을 실행하여 디스크 이미지를 업로드합니다. 이전 단계에서 확인한 매개변수를 지정합니다. 예를 들면 다음과 같습니다.

```
$ virtctl image-upload dv <datavolume_name> \ 1
--size=<datavolume_size> \ 2
--image-path=</path/to/image> \ 3
```

1. 데이터 볼륨의 이름입니다.
2. 데이터 볼륨의 크기입니다. 예를 들면 **--size=500Mi, --size=1G**와 같습니다.
3. 가상 머신 디스크 이미지의 파일 경로입니다.

참고

- 새 데이터 볼륨을 생성하지 않으려면 **--size** 매개변수를 생략하고 **--no-create** 플래그를 포함합니다.
- 디스크 이미지를 PVC에 업로드할 때 PVC 크기는 압축되지 않은 가상 디스크의 크기보다 커야 합니다.
- HTTPS를 사용할 때 비보안 서버 연결을 허용하려면 **--insecure** 매개변수를 사용하십시오. **--insecure** 플래그를 사용하면 업로드 끝점의 신뢰성이 확인되지 않습니다.

3. 선택사항입니다. 데이터 볼륨이 생성되었는지 확인하려면 다음 명령을 실행하여 모든 데이터 볼륨을 확인합니다.

```
$ oc get dvs
```

8.19.9.7. CDI 지원 작업 매트릭스

이 매트릭스에는 끝점에 대한 콘텐츠 유형에 따라 지원되는 CDI 작업과 이러한 작업 중 스크래치 공간이 필요한 작업이 표시되어 있습니다.

콘텐츠 유형	HTTP	HTTPS	HTTP 기본 인증	레지스트리	업로드
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt(RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 지원되는 작업

☐ 지원되지 않는 작업

* 스크래치 공간 필요

** 사용자 정의 인증 기관이 필요한 경우 스크래치 공간 필요

8.19.9.8. 추가 리소스

- 데이터 볼륨 작업에 대한 쓰기 성능을 향상하기 위해 사전 할당 모드를 구성합니다.

8.19.10. 가상 머신 스냅샷 관리

VM 전원이 꺼져 있는지(오프라인) 또는 온라인(온라인)에 관계없이 VM(가상 머신) 스냅샷을 생성하고 삭제할 수 있습니다. 전원이 꺼진(오프라인) VM으로만 복원할 수 있습니다. OpenShift Virtualization에서는 VM 스냅샷을 지원합니다.

- Red Hat OpenShift Data Foundation

- Kubernetes Volume Snapshot API를 지원하는 CSI(Container Storage Interface) 드라이버가 있는 기타 클라우드 스토리지 공급자

온라인 스냅샷에는5m(필요한 경우)을 변경할 수 있는 기본 시간이 5분(분)입니다.

중요

핫플러그 가상 디스크가 있는 가상 머신에는 온라인 스냅샷이 지원됩니다. 그러나 가상 머신 사양에 없는 핫플러그 디스크는 스냅샷에 포함되지 않습니다.

참고

가장 높은 무결성을 가진 온라인(실행 상태) VM의 스냅샷을 생성하려면 QEMU 게스트 에이전트를 설치합니다.

QEMU 게스트 에이전트는 시스템 워크로드에 따라 VM 파일 시스템을 가능한 한 많이 정지하여 일관된 스냅샷을 사용합니다. 이렇게 하면 스냅샷을 생성하기 전에 진행 중인 I/O가 디스크에 기록됩니다. 게스트 에이전트가 없으면 정지를 수행할 수 없으며 최상의 스냅샷을 생성합니다. 스냅샷이 수행된 조건은 웹 콘솔 또는 CLI에 표시되는 스냅샷 표시에 반영됩니다.

8.19.10.1. 가상 머신 스냅샷 정보

*스냅샷*은 특정 시점의 VM(가상 머신) 상태 및 데이터를 나타냅니다. 스냅샷을 사용하면 백업 및 재해 복구를 위해 기존 VM을 (스냅샷에 표시된) 이전 상태로 복원하거나 이전 개발 버전으로 신속하게 롤백할 수 있습니다.

VM 스냅샷은 전원이 꺼지거나(중지됨 상태) 전원이 켜진(실행 중)인 VM에서 생성됩니다.

실행 중인 VM의 스냅샷을 생성하는 경우 컨트롤러는 QEMU 게스트 에이전트가 설치되어 실행 중인지 확인합니다. 이 경우 스냅샷을 생성하기 전에 VM 파일 시스템을 중지하고 스냅샷을 만든 후 파일 시스템을 취소합니다.

스냅샷에는 VM에 연결된 각 CSI(Container Storage Interface) 볼륨 복사본과 VM 사양 및 메타데이터 복사본이 저장됩니다. 스냅샷을 생성한 후에는 변경할 수 없습니다.

VM 스냅샷 기능을 사용하면 클러스터 관리자와 애플리케이션 개발자가 다음을 수행할 수 있습니다.

- 새 프로젝트 생성
- 특정 VM에 연결된 모든 스냅샷 나열
- 스냅샷에서 VM 복원
- 기존 VM 스냅샷 삭제

8.19.10.1.1. 가상 머신 스냅샷 컨트롤러 및 CRD(사용자 정의 리소스 정의)

스냅샷 관리를 위해 VM 스냅샷 기능에 다음과 같이 CRD로 정의되는 새 API 오브젝트 세 가지가 도입되었습니다.

- **VirtualMachineSnapshot**: 스냅샷을 생성하라는 사용자 요청을 나타냅니다. 여기에는 VM의 현재 상태 정보가 포함됩니다.
- **VirtualMachineSnapshotContent**: 클러스터의 프로비저닝 리소스를 나타냅니다(스냅샷). VM 스냅샷 컨트롤러에서 생성하며 VM을 복원하는 데 필요한 모든 리소스에 대한 참조를 포함합니다.

- **VirtualMachineRestore**: 스냅샷에서 VM을 복원하라는 사용자 요청을 나타냅니다.

VM 스냅샷 컨트롤러는 **VirtualMachineSnapshot** 오브젝트와 이 오브젝트에 대해 생성된 **VirtualMachineSnapshotContent** 오브젝트를 일대일 매핑으로 바인딩합니다.

8.19.10.2. Linux 가상 머신에 QEMU 게스트 에이전트 설치

qemu-guest-agent는 광범위하게 사용되며, Red Hat 가상 머신에 기본적으로 제공됩니다. 에이전트를 설치하고 서비스를 시작합니다.

VM(가상 머신)에 QEMU 게스트 에이전트가 설치되어 실행되고 있는지 확인하려면 **AgentConnected**가 VM 사양에 나열되어 있는지 확인합니다.

참고

가장 높은 무결성을 가진 온라인(실행 상태) VM의 스냅샷을 생성하려면 QEMU 게스트 에이전트를 설치합니다.

QEMU 게스트 에이전트는 시스템 워크로드에 따라 VM의 파일 시스템을 가능한 한 많이 정지하여 일관된 스냅샷을 사용합니다. 이렇게 하면 스냅샷을 생성하기 전에 진행 중인 I/O가 디스크에 기록됩니다. 게스트 에이전트가 없으면 정지를 수행할 수 없으며 최상의 스냅샷을 생성합니다. 스냅샷이 수행된 조건은 웹 콘솔 또는 CLI에 표시되는 스냅샷 표시에 반영됩니다.

절차

1. 콘솔 중 하나 또는 SSH를 통해 가상 머신 명령줄에 액세스합니다.
2. 가상 머신에 QEMU 게스트 에이전트를 설치합니다.

```
$ yum install -y qemu-guest-agent
```

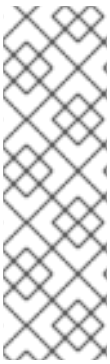
3. 서비스가 지속되는지 확인하고 다음을 시작합니다.

```
$ systemctl enable --now qemu-guest-agent
```

8.19.10.3. Windows 가상 머신에 QEMU 게스트 에이전트 설치

Windows 가상 머신의 경우 QEMU 게스트 에이전트는 VirtIO 드라이버에 포함됩니다. 기존 또는 새 Windows 설치에 드라이버를 설치합니다.

VM(가상 머신)에 QEMU 게스트 에이전트가 설치되어 실행되고 있는지 확인하려면 **AgentConnected**가 VM 사양에 나열되어 있는지 확인합니다.



참고

가장 높은 무결성을 가진 온라인(실행 상태) VM의 스냅샷을 생성하려면 QEMU 게스트 에이전트를 설치합니다.

QEMU 게스트 에이전트는 시스템 워크로드에 따라 VM의 파일 시스템을 가능한 한 많이 정지하여 일관된 스냅샷을 사용합니다. 이렇게 하면 스냅샷을 생성하기 전에 진행 중인 I/O가 디스크에 기록됩니다. 게스트 에이전트가 없으면 정지를 수행할 수 없으며 최상의 스냅샷을 생성합니다. 스냅샷이 수행된 조건은 웹 콘솔 또는 CLI에 표시되는 스냅샷 표시에 반영됩니다.

8.19.10.3.1. 기존 Windows 가상 머신에 VirtIO 드라이버 설치

연결된 SATA CD 드라이브에서 기존 Windows 가상 머신에 VirtIO 드라이버를 설치합니다.

참고

다음 절차에서는 일반적인 방법을 사용하여 Windows에 드라이버를 추가합니다. 프로세스는 Windows 버전마다 약간 다를 수 있습니다. 특정 설치 단계는 사용 중인 Windows 버전의 설치 설명서를 참조하십시오.

절차

1. 가상 머신을 시작하고 그래픽 콘솔에 연결합니다.
2. Windows 사용자 세션에 로그인합니다.
3. 장치 관리자를 열고 기타 장치를 확장하여 알 수 없는 장치를 나열합니다.
    a. **Device Properties**을 열어 알 수 없는 장치를 확인합니다. 장치를 마우스 오른쪽 버튼으로 클릭하고 속성을 선택합니다.
    b. 세부 정보 탭을 클릭하고 속성 목록에서 하드웨어 ID를 선택합니다.
    c. 하드웨어 ID의 값을 지원되는 VirtIO 드라이버와 비교합니다.
4. 장치를 마우스 오른쪽 단추로 클릭하고 드라이버 소프트웨어 업데이트를 선택합니다.
5. 컴퓨터에서 드라이버 소프트웨어 찾아보기를 클릭하고 VirtIO 드라이버가 있는 연결된 SATA CD 드라이브를 찾습니다. 드라이버는 드라이버 유형, 운영 체제, CPU 아키텍처에 따라 계층적으로 정렬됩니다.
6. 다음을 클릭하여 드라이버를 설치합니다.
7. 필요한 모든 VirtIO 드라이버에 대해 이 과정을 반복합니다.
8. 드라이버 설치 후 닫기를 클릭하여 창을 닫습니다.
9. 가상 머신을 재부팅하여 드라이버 설치를 완료합니다.

8.19.10.3.2. Windows 설치 중 VirtIO 드라이버 설치

Windows를 설치하는 동안 연결된 SATA CD 드라이버에서 VirtIO 드라이버를 설치합니다.

참고

이 절차에서는 일반적인 Windows 설치 방법을 사용하며, 설치 방법은 Windows 버전마다 다를 수 있습니다. 설치 중인 Windows 버전에 대한 설명서를 참조하십시오.

절차

1. 가상 머신을 시작하고 그래픽 콘솔에 연결합니다.
2. Windows 설치 프로세스를 시작합니다.
3. 고급 설치를 선택합니다.

4. 저장 대상은 드라이버가 로드되어야 인식됩니다. **Load driver**를 클릭합니다.

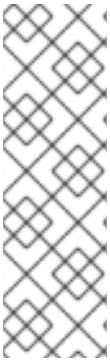
5. 드라이버는 SATA CD 드라이브로 연결되어 있습니다. 확인을 클릭하고 스토리지 드라이버를 로드할 CD 드라이브를 찾습니다. 드라이버는 드라이버 유형, 운영 체제, CPU 아키텍처에 따라 계층적으로 정렬됩니다.

6. 필요한 모든 드라이버에 대해 위의 두 단계를 반복합니다.

7. Windows 설치를 완료합니다.

8.19.10.4. 웹 콘솔에서 가상 머신 스냅샷 생성

웹 콘솔을 사용하여 VM(가상 머신) 스냅샷을 생성할 수 있습니다.



참고

가장 높은 무결성을 가진 온라인(실행 상태) VM의 스냅샷을 생성하려면 QEMU 게스트 에이전트를 설치합니다.

QEMU 게스트 에이전트는 시스템 워크로드에 따라 VM의 파일 시스템을 가능한 한 많이 정지하여 일관된 스냅샷을 사용합니다. 이렇게 하면 스냅샷을 생성하기 전에 진행 중인 I/O가 디스크에 기록됩니다. 게스트 에이전트가 없으면 정지를 수행할 수 없으며 최상의 스냅샷을 생성합니다. 스냅샷이 수행된 조건은 웹 콘솔 또는 CLI에 표시되는 스냅샷 표시에 반영됩니다.

VM 스냅샷에는 다음 요구 사항을 충족하는 디스크만 포함됩니다.

- 데이터 볼륨 또는 영구 볼륨 클레임 중 하나여야 합니다.
- CSI(Container Storage Interface) 볼륨 스냅샷을 지원하는 스토리지 클래스에 속해야 합니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.
3. 가상 머신이 실행 중인 경우 작업 → 중지 를 클릭하여 전원을 끕니다.
4. 스냅샷 탭을 클릭한 후스냅샷 찍기를 클릭합니다.
5. 스냅샷 이름 및 선택 사항으로설명 필드를 작성합니다.
6. 이 스냅샷에 포함된 디스크를 확장하여 스냅샷에 스토리지 볼륨이 포함되어 있는지 확인합니다.
7. VM에 스냅샷에 포함할 수 없는 디스크가 있고 계속 진행하려면 이 경고에 대해 인식하고 있으며 계속 진행하겠습니다. 확인란을 선택합니다.
8. 저장을 클릭합니다.

8.19.10.5. CLI에서 가상 머신 스냅샷 생성

VirtualMachineSnapshot 오브젝트를 생성하여 오프라인 또는 온라인 VM에 대한 VM(가상 머신) 스냅샷을 생성할 수 있습니다. kubevirt는 QEMU 게스트 에이전트와 조정하여 온라인 VM의 스냅샷을 생성합니다.

참고

가장 높은 무결성을 가진 온라인(실행 상태) VM의 스냅샷을 생성하려면 QEMU 게스트 에이전트를 설치합니다.

QEMU 게스트 에이전트는 시스템 워크로드에 따라 VM의 파일 시스템을 가능한 한 많이 정지하여 일관된 스냅샷을 사용합니다. 이렇게 하면 스냅샷을 생성하기 전에 진행 중인 I/O가 디스크에 기록됩니다. 게스트 에이전트가 없으면 정지를 수행할 수 없으며 최상의 스냅샷을 생성합니다. 스냅샷이 수행된 조건은 웹 콘솔 또는 CLI에 표시되는 스냅샷 표시에 반영됩니다.

사전 요구 사항

- PVC(영구 볼륨 클레임)이 CSI(Container Storage Interface) 볼륨 스냅샷을 지원하는 스토리지 클래스에 있는지 확인합니다.
- OpenShift CLI(oc)를 설치합니다.
- 선택 사항: 스냅샷을 생성할 VM의 전원을 끕니다.

절차

1. YAML 파일을 생성하여 새 VirtualMachineSnapshot의 이름과 소스 VM의 이름을 지정하는 VirtualMachineSnapshot 오브젝트를 정의합니다.
예를 들면 다음과 같습니다.

```
apiVersion: snapshot.kubevirt.io/v1alpha1
kind: VirtualMachineSnapshot
metadata:
  name: my-vmsnapshot 1
spec:
  source:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: my-vm 2
```

1 새 VirtualMachineSnapshot 오브젝트의 이름입니다.

2 소스 VM의 이름입니다.

2. VirtualMachineSnapshot 리소스를 생성합니다. 스냅샷 컨트롤러에서 VirtualMachineSnapshotContent 오브젝트를 생성하여 VirtualMachineSnapshot에 바인딩하고 VirtualMachineSnapshot 오브젝트의 status 및 readyToUse 필드를 업데이트합니다.

```
$ oc create -f <my-vmsnapshot>.yaml
```

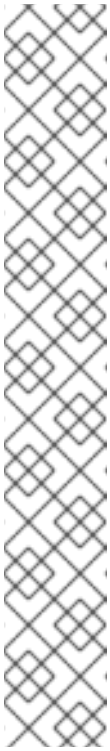
3. 선택 사항: 온라인 스냅샷을 생성하는 경우 wait 명령을 사용하여 스냅샷 상태를 모니터링할 수 있습니다.

   a. 다음 명령을 실행합니다.

   ```
   $ oc wait my-vm my-vmsnapshot --for condition=Ready
   ```

   b. 스냅샷 상태를 확인합니다.

- **InProgress** - 온라인 스냅샷 작업이 아직 진행 중입니다.
- **Succeeded** - 온라인 스냅샷 작업이 성공적으로 완료되었습니다.
- **Failed** - 온라인 스냅샷 작업이 실패했습니다.



참고

온라인 스냅샷의 기본 시간은 5분(5m)입니다. 5분 내에 스냅샷이 성공적으로 완료되지 않으면 상태가 **failed**로 설정됩니다. 나중에 파일 시스템이 손상되고 VM이 수정되지 않지만 **failed** 스냅샷 이미지를 삭제할 때까지 상태는 실패로 유지됩니다.

기본 시간 기한을 변경하려면 스냅샷 작업이 시간 초과되기 전에 지정할 시간(분)(**m**) 또는 초(초)(초)(초)를 사용하여 VM 스냅샷 사양에 **FailureDeadline** 특성을 추가합니다.

시간 기한을 설정하지 않으려면 **0**을 지정할 수 있지만 **0**은 응답하지 않는 VM이 발생할 수 있으므로 일반적으로 권장되지 않습니다.

m 또는 **s** 와 같은 시간 단위를 지정하지 않으면 기본값은 초(**s**)입니다.

검증

1. **VirtualMachineSnapshot** 오브젝트가 생성되고 **VirtualMachineSnapshotContent**에 바인딩되었는지 확인합니다. **readyToUse** 플래그를 **true**로 설정해야 합니다.

```
$ oc describe vmsnapshot <my-vmsnapshot>
```

출력 예

```
apiVersion: snapshot.kubevirt.io/v1alpha1
kind: VirtualMachineSnapshot
metadata:
  creationTimestamp: "2020-09-30T14:41:51Z"
  finalizers:
  - snapshot.kubevirt.io/vmsnapshot-protection
  generation: 5
  name: mysnap
  namespace: default
  resourceVersion: "3897"
  selfLink:
/apis/snapshot.kubevirt.io/v1alpha1/namespaces/default/virtualmachinesnapshots/my-vmsnapshot
  uid: 28eedf08-5d6a-42c1-969c-2eda58e2a78d
```

```
spec:
  source:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: my-vm
status:
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2020-09-30T14:42:03Z"
    reason: Operation complete
    status: "False" 1
    type: Progressing
  - lastProbeTime: null
    lastTransitionTime: "2020-09-30T14:42:03Z"
    reason: Operation complete
    status: "True" 2
    type: Ready
  creationTime: "2020-09-30T14:42:03Z"
  readyToUse: true 3
  sourceUID: 355897f3-73a0-4ec4-83d3-3c2df9486f4f
  virtualMachineSnapshotContentName: vmsnapshot-content-28eedf08-5d6a-42c1-
969c-2eda58e2a78d 4
```

1 **status** 필드의 **Progressing** 조건은 스냅샷이 아직 생성 중인지를 나타냅니다.

2 **status** 필드의 **Ready** 조건은 스냅샷 생성 프로세스가 완료되었는지를 나타냅니다.

3 스냅샷을 사용할 준비가 되었는지를 나타냅니다.

4 스냅샷이 스냅샷 컨트롤러에서 생성한 **VirtualMachineSnapshotContent** 오브젝트에 바인딩되도록 지정합니다.

2. **VirtualMachineSnapshotContent** 리소스의 **spec:volumeBackups** 속성을 확인하여 예상 PVC가 스냅샷에 포함되어 있는지 확인합니다.

8.19.10.6. 스냅샷 표시를 사용하여 온라인 스냅샷 생성 확인

스냅샷 표시는 온라인 VM(가상 시스템) 스냅샷 작업에 대한 컨텍스트 정보입니다. 오프라인 VM(가상 시스템) 스냅샷 작업에는 표시를 사용할 수 없습니다. 표시는 온라인 스냅샷 생성에 대한 세부 정보를 설명하는 데 유용합니다.

사전 요구 사항

- 표시를 보려면 CLI 또는 웹 콘솔을 사용하여 온라인 VM 스냅샷을 생성을 시도해야 합니다.

절차

1. 다음 중 하나를 수행하여 스냅샷 표시의 출력을 표시합니다.
   - CLI를 사용하여 생성된 스냅샷의 경우 status 필드의 VirtualMachineSnapshot 오브젝트 YAML의 표시기 출력을 확인합니다.
   - 웹 콘솔을 사용하여 생성한 스냅샷의 경우 스냅샷 세부 정보 화면에서 가상 머신 스냅샷 > 상태를 클릭합니다.
2. 온라인 VM 스냅샷의 상태를 확인합니다.
   - Online에서는 온라인 스냅샷 생성 중에 VM이 실행 중임을 나타냅니다.
   - NoGuestAgent는 온라인 스냅샷을 생성하는 동안 QEMU 게스트 에이전트가 실행되지 않았음을 나타냅니다. QEMU 게스트 에이전트가 설치되지 않았거나 실행 중이거나 다른 오류로 인해 QEMU 게스트 에이전트를 사용하여 파일 시스템을 중지하고 해석할 수 없습니다.

8.19.10.7. 웹 콘솔을 사용하여 스냅샷에서 가상 머신 복원

웹 콘솔에서 스냅샷으로 표시한 이전 구성으로 VM(가상 머신)을 복원할 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2.

2. 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.

3. 가상 머신이 실행 중인 경우 작업 → 중지 를 클릭하여 전원을 끕니다.

4. 스냅샷 탭을 클릭합니다. 페이지에는 가상 머신과 연결된 스냅샷 목록이 표시됩니다.

5. VM 스냅샷을 복원하는 다음 방법 중 하나를 선택합니다.

   a. VM을 복원하기 위해 소스로 사용할 스냅샷에서 복원을 클릭합니다.

   b. 스냅샷을 선택하여 스냅샷 세부 정보 화면을 열고 작업 → VirtualMachineSnapshot 복원을 클릭합니다.

6. 확인 팝업 창에서 복원을 클릭하여 스냅샷에 표시된 VM을 이전 구성으로 복원합니다.

8.19.10.8. CLI를 사용하여 스냅샷에서 가상 머신 복원

VM 스냅샷을 사용하여 기존 VM(가상 머신)을 이전 구성으로 복원할 수 있습니다. 오프라인 VM 스냅샷에서만 복원할 수 있습니다.

사전 요구 사항

- OpenShift CLI(oc)를 설치합니다.
- 이전 상태로 복원하려는 VM의 전원을 끕니다.

절차

1. YAML 파일을 생성하여 복원할 VM의 이름과 소스로 사용할 스냅샷 이름을 지정하는 VirtualMachineRestore 오브젝트를 정의합니다.

   예를 들면 다음과 같습니다.

```
apiVersion: snapshot.kubevirt.io/v1alpha1
kind: VirtualMachineRestore
metadata:
  name: my-vmrestore 1
spec:
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: my-vm 2
  virtualMachineSnapshotName: my-vmsnapshot 3
```

1 새 VirtualMachineRestore 오브젝트의 이름입니다.

2 복원할 대상 VM의 이름입니다.

3 소스로 사용할 VirtualMachineSnapshot 오브젝트의 이름입니다.

2. VirtualMachineRestore 리소스를 만듭니다. 스냅샷 컨트롤러는 VirtualMachineRestore 오브젝트의 상태 필드를 업데이트하고 기존 VM 구성을 스냅샷의 콘텐츠로 교체합니다.

```
$ oc create -f <my-vmrestore>.yaml
```

검증

- VM이 스냅샷에 표시된 이전 상태로 복원되었는지 확인합니다. complete 플래그가 true로 설정되어야 합니다.

```
$ oc get vmrestore <my-vmrestore>
```

출력 예

```
apiVersion: snapshot.kubevirt.io/v1alpha1
kind: VirtualMachineRestore
metadata:
creationTimestamp: "2020-09-30T14:46:27Z"
generation: 5
name: my-vmrestore
```

```
namespace: default
ownerReferences:
- apiVersion: kubevirt.io/v1
  blockOwnerDeletion: true
  controller: true
  kind: VirtualMachine
  name: my-vm
  uid: 355897f3-73a0-4ec4-83d3-3c2df9486f4f
  resourceVersion: "5512"
  selfLink:
/apis/snapshot.kubevirt.io/v1alpha1/namespaces/default/virtualmachinerestores/my-
vmrestore
  uid: 71c679a8-136e-46b0-b9b5-f57175a6a041
  spec:
    target:
      apiGroup: kubevirt.io
      kind: VirtualMachine
      name: my-vm
  virtualMachineSnapshotName: my-vmsnapshot
  status:
  complete: true (1)
  conditions:
  - lastProbeTime: null
  lastTransitionTime: "2020-09-30T14:46:28Z"
  reason: Operation complete
  status: "False" (2)
  type: Progressing
  - lastProbeTime: null
  lastTransitionTime: "2020-09-30T14:46:28Z"
  reason: Operation complete
  status: "True" (3)
  type: Ready
  deletedDataVolumes:
  - test-dv1
  restoreTime: "2020-09-30T14:46:28Z"
  restores:
  - dataVolumeName: restore-71c679a8-136e-46b0-b9b5-f57175a6a041-
datavolumedisk1
  persistentVolumeClaim: restore-71c679a8-136e-46b0-b9b5-f57175a6a041-
datavolumedisk1
  volumeName: datavolumedisk1
  volumeSnapshotName: vmsnapshot-28eedf08-5d6a-42c1-969c-2eda58e2a78d-
volume-datavolumedisk1
```

(1)

VM을 스냅샷에 표시된 상태로 복원하는 프로세스가 완료되었는지를 나타냅니다.

(2)

status 필드의 **Progressing** 조건은 **VM**이 아직 복원 중인지를 나타냅니다.

3

status 필드의 **Ready** 조건은 **VM** 복원 프로세스가 완료되었는지를 나타냅니다.

8.19.10.9. 웹 콘솔에서 가상 머신 스냅샷 삭제

웹 콘솔을 사용하여 기존 가상 머신 스냅샷을 삭제할 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachines** 를 클릭합니다.
2. 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.
3. 스냅샷 탭을 클릭합니다. 페이지에는 가상 머신과 연결된 스냅샷 목록이 표시됩니다.
4. 삭제할 가상 머신의 옵션 메뉴
   ⋮
   를 클릭하고 **VirtualMachineSnapshot** 삭제 를 선택합니다.
5. 확인 팝업 창에서 삭제를 클릭하여 스냅샷을 삭제합니다.

8.19.10.10. CLI에서 가상 머신 스냅샷 삭제

적절한 **VirtualMachineSnapshot** 오브젝트를 삭제하여 기존 **VM**(가상 머신) 스냅샷을 삭제할 수 있습니다.

사전 요구 사항

- **OpenShift CLI(oc)**를 설치합니다.

절차

- VirtualMachineSnapshot 오브젝트를 삭제합니다. 스냅샷 컨트롤러는 VirtualMachineSnapshot을 연결된 VirtualMachineSnapshotContent 오브젝트와 함께 삭제합니다.

```
$ oc delete vmsnapshot <my-vmsnapshot>
```

검증

- 스냅샷이 삭제되고 더 이상 이 VM에 연결되어 있지 않은지 확인합니다.

```
$ oc get vmsnapshot
```

8.19.10.11. 추가 리소스

- CSI 볼륨 스냅샷

8.19.11. 로컬 가상 머신 디스크를 다른 노드로 이동

로컬 볼륨 스토리지를 사용하는 가상 머신을 특정 노드에서 실행하도록 이동할 수 있습니다.

다음과 같은 이유로 가상 머신을 특정 노드로 이동할 수 있습니다.

- 현재 노드에서는 로컬 스토리지 구성을 제한합니다.
- 새 노드가 해당 가상 머신의 워크로드에 더 최적화되어 있습니다.

로컬 스토리지를 사용하는 가상 머신을 이동하려면 데이터 볼륨을 사용하여 기본 볼륨을 복제해야 합니다. 복제 작업이 완료되면 새 데이터 볼륨을 사용하도록 가상 머신 구성을 편집하거나 새 데이터 볼륨을 다른 가상 머신에 추가할 수 있습니다.

작은 정보

사전 할당을 활성화하거나 단일 데이터 볼륨에 대해 복제 중에 디스크 공간을 사전 할당하는 경우 CDI(Containerized Data Importer)가 디스크 공간을 사전 할당합니다. 사전 할당을 통해 쓰기 성능이 향상됩니다. 자세한 내용은 데이터 볼륨에 대한 사전 할당 사용을 참조하십시오.

참고

cluster-admin 역할이 없는 사용자는 다른 네임스페이스에 볼륨을 복제하려면 추가 사용자 권한이 있어야 합니다.

8.19.11.1. 다른 노드에 로컬 볼륨 복제

가상 머신 디스크를 특정 노드에서 실행하기 위해 기본 PVC(영구 볼륨 클레임)를 복제하여 가상 머신 디스크를 이동할 수 있습니다.

가상 머신 디스크가 올바른 노드에 복제되었는지 확인하려면 새 PV(영구 볼륨)를 생성하거나 올바른 노드에서 가상 머신 디스크를 확인합니다. 데이터 볼륨에서 참조할 수 있도록 PV에 고유한 라벨을 적용하십시오.

참고

대상 PV는 소스 PVC와 크기가 같거나 커야 합니다. 대상 PV가 소스 PVC보다 작으면 복제 작업이 실패합니다.

사전 요구 사항

- 가상 머신이 실행되고 있지 않아야 합니다. 가상 머신 디스크를 복제하기 전에 가상 머신의 전원을 끄십시오.

절차

1. 노드에 새 로컬 PV를 생성하거나 노드의 기존 로컬 PV를 확인합니다.

- nodeAffinity.nodeSelectorTerms 매개변수를 포함하는 로컬 PV를 생성합니다. 다음 매니페스트에서는 node01에 10Gi 로컬 PV를 생성합니다.

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: <destination-pv> 1
  annotations:
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi 2
```

```
  local:
    path: /mnt/local-storage/local/disk1 (3)
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - node01 (4)
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local
  volumeMode: Filesystem
```

(1) PV의 이름입니다.

(2) PV의 크기입니다. 충분한 공간을 할당해야 합니다. 그러지 않으면 복제 작업이 실패합니다. 크기는 소스 PVC와 같거나 커야 합니다.

(3) 노드의 마운트 경로입니다.

(4) PV를 생성하려는 노드의 이름입니다.

- 대상 노드에 이미 존재하는 PV를 확인합니다. 구성에서 **nodeAffinity** 필드를 확인하여 PV가 프로비저닝되는 노드를 확인할 수 있습니다.

```
$ oc get pv <destination-pv> -o yaml
```

다음 스니펫은 PV가 node01에 있음을 보여줍니다.

출력 예

```
...
spec:
  nodeAffinity:
    required:
```

```
    nodeSelectorTerms:
    - matchExpressions:
      - key: kubernetes.io/hostname (1)
        operator: In
        values:
        - node01 (2)
...
```

(1) kubernetes.io/hostname 키는 노드 호스트 이름을 사용하여 노드를 선택합니다.

(2) 노드의 호스트 이름입니다.

2. PV에 고유한 라벨을 추가합니다.

```
$ oc label pv <destination-pv> node=node01
```

3. 다음을 참조하는 데이터 볼륨 매니페스트를 생성합니다.

- 가상 머신의 PVC 이름 및 네임스페이스
- 이전 단계에서 PV에 적용한 라벨
- 대상 PV의 크기

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <clone-datavolume> (1)
spec:
  source:
    pvc:
      name: "<source-vm-disk>" (2)
      namespace: "<source-namespace>" (3)
```

```
  pvc:
    accessModes:
      - ReadWriteOnce
    selector:
      matchLabels:
        node: node01 4
    resources:
      requests:
        storage: <10Gi> 5
```

1

새 데이터 볼륨의 이름입니다.

2

소스 PVC의 이름입니다. PVC 이름을 모르는 경우 가상 머신 구성의 spec.volumes.persistentVolumeClaim.claimName에서 확인할 수 있습니다.

3

소스 PVC가 존재하는 네임스페이스입니다.

4

이전 단계에서 PV에 적용한 라벨입니다.

5

대상 PV의 크기

4. 클러스터에 데이터 볼륨 매니페스트를 적용하여 복제 작업을 시작합니다.

```
$ oc apply -f <clone-datavolume.yaml>
```

데이터 볼륨은 가상 머신의 PVC를 특정 노드의 PV에 복제합니다.

8.19.12. 빈 디스크 이미지를 추가하여 가상 스토리지 확장

OpenShift Virtualization에 빈 디스크 이미지를 추가하여 스토리지 용량을 늘리거나 새 데이터 파티션을 만들 수 있습니다.

8.19.12.1. 데이터 볼륨 정보

Dataolume 오브젝트는 CDI(Containerized Data Importer) 프로젝트에서 제공하는 사용자 정의 리소스입니다. 데이터 볼륨은 기본 PVC(영구 볼륨 클레임)와 관련된 가져오기, 복제, 업로드 작업을 오케스트레이션합니다. 데이터 볼륨은 OpenShift Virtualization과 통합되며 PVC가 준비되기 전에 가상 머신이 시작되지 않도록 합니다.

8.19.12.2. 데이터 볼륨에 빈 디스크 이미지 만들기

데이터 볼륨 구성 파일을 사용자 정의하고 배포하여 영구 볼륨 클레임에 빈 디스크 이미지를 새로 만들 수 있습니다.

사전 요구 사항

- 사용 가능한 영구 볼륨이 1개 이상 있습니다.
- OpenShift CLI(oc)를 설치합니다.

절차

1. DataVolume 매니페스트를 편집합니다.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: blank-image-datavolume
spec:
  source:
    blank: {}
  pvc:
    # Optional: Set the storage class or omit to accept the default
    # storageClassName: "hostpath"
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 500Mi
```

2. 다음 명령을 실행하여 빈 디스크 이미지를 만듭니다.

```
$ oc create -f <blank-image-datavolume>.yaml
```

8.19.12.3. 추가 리소스

- 데이터 볼륨 작업에 대한 쓰기 성능을 향상하기 위해 사전 할당 모드를 구성합니다.

8.19.13. 스마트 복제를 사용하여 데이터 볼륨 복제

스마트 복제는 **Red Hat OpenShift Data Foundation**의 기본 제공 기능입니다. 스마트 복제는 호스트 지원 복제보다 빠르고 효율적입니다.

스마트 복제를 활성화하기 위해 특별히 수행해야 할 작업은 없지만 이 기능을 사용하려면 스토리지 환경이 스마트 복제와 호환되는지 확인해야 합니다.

PVC(영구 볼륨 클레임) 소스를 사용하여 데이터 볼륨을 생성하면 복제 프로세스가 자동으로 시작됩니다. 해당 환경에서 스마트 복제를 지원하는지와 관계없이 데이터 볼륨 복제본은 항상 제공됩니다. 그러나 스토리지 공급자가 스마트 복제를 지원하는 경우에만 스마트 복제의 성능적인 이점을 활용할 수 있습니다.

8.19.13.1. 스마트 복제 정보

데이터 볼륨이 스마트 복제될 때는 다음 작업이 수행됩니다.

1. 소스 **PVC**(영구 볼륨 클레임)의 스냅샷이 생성됩니다.
2. 스냅샷에서 **PVC**가 생성됩니다.
3. 스냅샷이 삭제됩니다.

8.19.13.2. 데이터 볼륨 복제

사전 요구 사항

스마트 복제를 수행하려면 다음 조건이 필요합니다.

- 스토리지 공급자에서 스냅샷을 지원해야 합니다.
- 소스 및 대상 **PVC**가 동일한 스토리지 클래스에 정의되어 있어야 합니다.

- 소스 및 대상 PVC는 동일한 volumeMode를 공유합니다.

- VolumeSnapshotClass 오브젝트에서 소스 및 대상 PVC 모두에 정의된 스토리지 클래스를 참조해야 합니다.

절차

데이터 볼륨 복제를 시작하려면 다음을 수행합니다.

1. DataVolume 오브젝트에 대해 새 데이터 볼륨의 이름, 소스 PVC의 이름과 네임스페이스를 지정하는 YAML 파일을 생성합니다. 이 예제에서는 스토리지 API를 지정하므로 accessModes 또는 volumeMode를 지정할 필요가 없습니다. 최적의 값을 자동으로 계산합니다.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <cloner-datavolume> 1
spec:
  source:
    pvc:
      namespace: "<source-namespace>" 2
      name: "<my-favorite-vm-disk>" 3
  storage: 4
    resources:
      requests:
        storage: <2Gi> 5
```

1. 새 데이터 볼륨의 이름입니다.

2. 소스 PVC가 존재하는 네임스페이스입니다.

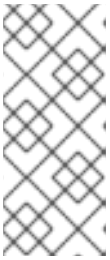
3. 소스 PVC의 이름입니다.

4. 스토리지 API로 할당을 지정합니다.

5.

2. 데이터 볼륨을 생성하여 PVC 복제를 시작합니다.

```
$ oc create -f <cloner-datavolume>.yaml
```



참고

데이터 볼륨이 있으면 PVC가 준비될 때까지 가상 머신이 시작되지 않으므로 PVC가 복제되는 동안 새 데이터 볼륨을 참조하는 가상 머신을 생성할 수 있습니다.

8.19.13.3. 추가 리소스

- 가상 머신 디스크의 영구 볼륨 클레임을 새 데이터 볼륨으로 복제
- 데이터 볼륨 작업에 대한 쓰기 성능을 향상하기 위해 사전 할당 모드를 구성합니다.
- 스토리지 프로파일 사용자 정의

8.19.14. 부팅 소스 생성 및 사용

부팅 소스에는 부팅 가능한 운영 체제(OS) 및 OS의 모든 구성 설정(예: 드라이버)이 포함되어 있습니다.

부팅 소스를 사용하여 특정 구성으로 가상 머신 템플릿을 생성합니다. 이러한 템플릿은 사용 가능한 여러 가상 머신을 생성하는 데 사용할 수 있습니다.

OpenShift Container Platform 웹 콘솔에서 빠른 시작 둘러보기를 사용하여 사용자 정의 부팅 소스 생성, 부팅 소스 및 기타 작업을 지원합니다. 도움말 메뉴에서 빠른 시작을 선택하여 빠른 시작 둘러보기를 확인합니다.

8.19.14.1. 가상 머신 및 부팅 소스 정보

가상 시스템은 가상 시스템 정의와 데이터 볼륨에서 지원하는 하나 이상의 디스크로 구성됩니다. 가상 머신 템플릿을 사용하면 사전 정의된 가상 머신 사양을 사용하여 가상 머신을 생성할 수 있습니다.

모든 가상 머신 템플릿에는 구성된 드라이버를 포함하여 완전히 구성된 가상 머신 디스크 이미지인 부팅 소스가 필요합니다. 각 가상 머신 템플릿에는 부팅 소스에 대한 포인터가 있는 가상 시스템 정의가 포함되어 있습니다. 각 부팅 소스에는 사전 정의된 이름과 네임스페이스가 있습니다. 일부 운영 체제의 경우 부팅 소스가 자동으로 제공됩니다. 제공되지 않는 경우 관리자는 사용자 지정 부팅 소스를 준비해야 합니다.

제공된 부팅 소스가 최신 버전의 운영 체제로 자동 업데이트됩니다. 자동 업데이트됨 부팅 소스의 경우 PVC(영구 볼륨 클레임)가 클러스터의 기본 스토리지 클래스를 사용하여 생성됩니다. 구성 후 다른 기본 스토리지 클래스를 선택하는 경우 이전 기본 스토리지 클래스로 구성된 클러스터 네임스페이스의 기존 데이터 볼륨을 삭제해야 합니다.

부팅 소스 기능을 사용하려면 OpenShift Virtualization의 최신 릴리스를 설치합니다. 네임스페이스 openshift-virtualization-os-images는 기능을 활성화하고 OpenShift Virtualization Operator와 함께 설치됩니다. 부팅 소스 기능이 설치되면 부팅 소스를 생성하고 템플릿에 연결한 다음 템플릿에서 가상 머신을 생성할 수 있습니다.

로컬 파일 업로드, 기존 PVC 복제, 레지스트리에서 가져오기 또는 URL을 통해 채워지는 PVC(영구 볼륨 클레임)를 사용하여 부팅 소스를 정의합니다. 웹 콘솔을 사용하여 가상 머신 템플릿에 부팅 소스를 연결합니다. 부팅 소스를 가상 머신 템플릿에 연결한 후 템플릿에서 완전히 구성된 즉시 사용할 수 있는 가상 시스템을 원하는 만큼 생성합니다.

8.19.14.2. 부팅 소스로 RHEL 이미지 가져오기

이미지의 URL을 지정하여 RHEL(Red Hat Enterprise Linux) 이미지를 부팅 소스로 가져올 수 있습니다.

사전 요구 사항

- 운영 체제 이미지를 사용하여 웹 페이지에 액세스할 수 있어야 합니다. 예를 들면 이미지가 포함된 Red Hat Enterprise Linux 웹 페이지를 다운로드합니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.
2. 부팅 소스를 구성할 RHEL 템플릿을 확인하고 소스 추가를 클릭합니다.

3. 템플릿에 부팅 소스 추가 창의 **Boot** 소스 유형 목록에서 **URL(PVC 생성)** 을 선택합니다.

4. **RHEL** 다운로드 페이지를 클릭하여 **Red Hat** 고객 포털에 액세스합니다. 사용 가능한 설치 프로그램 및 이미지 목록이 **Red Hat Enterprise Linux** 다운로드 페이지에 표시됩니다.

5. 다운로드하려는 **Red Hat Enterprise Linux KVM** 게스트 이미지를 식별합니다. 지금 다운로드를 마우스 오른쪽 버튼으로 클릭하고 이미지의 **URL**을 복사합니다.

6. 템플릿에 부팅 소스 추가 창에서 **URL** 가져오기 필드에 붙여넣고 저장 및 가져오기 를 클릭합니다.

검증

1. 템플릿 페이지의 부팅 소스 열에 템플릿에 녹색 확인 표시가 표시되는지 확인합니다.

이제 이 템플릿을 사용하여 **RHEL** 가상 머신을 생성할 수 있습니다.

8.19.14.3. 가상 머신 템플릿용 부팅 소스 추가

가상 머신 또는 사용자 지정 템플릿을 생성하기 위해 사용할 가상 머신 템플릿을 위한 부팅 소스를 구성할 수 있습니다. 가상 머신 템플릿이 부팅 소스로 구성된 경우 템플릿 페이지에서 사용할 수 있는 **Source** 로 레이블이 지정됩니다. 템플릿에 부팅 소스를 추가한 후 템플릿에서 새 가상 머신을 생성할 수 있습니다.

다음과 같은 **4**가지 방법으로 웹 콘솔에서 부팅 소스를 선택하고 추가할 수 있습니다.

- 로컬 파일 업로드 **(PVC** 생성**)**
- **URL (PVC** 생성**)**
- 복제**(PVC** 생성**)**

- 레지스트리(PVC 생성)

사전 요구 사항

- 부팅 소스를 추가하려면, **os-images.kubevirt.io:edit RBAC** 역할의 사용자 또는 관리자로 로그인해야 합니다. 부팅 소스가 추가된 템플릿에서 가상 머신을 생성하려면 특정 권한이 필요하지 않습니다.

- 로컬 파일을 업로드하려면 운영 체제 이미지 파일이 로컬 머신에 있어야 합니다.

- **URL**을 통해 가져오려면 운영 체제 이미지를 사용하여 웹 서버에 액세스해야 합니다. 예를 들면 이미지가 포함된 **Red Hat Enterprise Linux** 웹 페이지입니다.

- 기존 **PVC**를 복제하려면 **PVC**를 사용하여 프로젝트에 대한 액세스가 필요합니다.

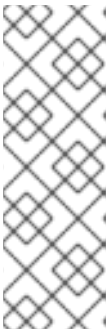
- 레지스트리를 통해 가져오려면 컨테이너 레지스트리에 대한 액세스가 필요합니다.

절차

1. **OpenShift Container Platform** 콘솔의 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.
2. 템플릿 옆에 있는 옵션 메뉴를 클릭하고 부팅 소스 편집 을 선택합니다.
3. 디스크 추가를 클릭합니다.
4. 디스크 추가 창에서 이 디스크 사용을 부팅 소스로 선택합니다.
5. 디스크 이름을 입력하고 소스(예 : **Blank(PVC** 생성)) 를 선택하거나 기존 **PVC** 사용.
6. 영구 볼륨 클레임(**PVC**) 크기에 값을 입력하여 압축이 해제되지 않은 이미지에 적합한 **PVC** 크기를 지정하고 필요한 추가 공간을 지정합니다.

7. 유형 (예: 디스크 또는 CD-ROM)을 선택합니다.

8. 선택 사항: 스토리지 클래스 를 클릭하고 디스크를 생성하는 데 사용되는 스토리지 클래스를 선택합니다. 일반적으로 이 스토리지 클래스는 모든 PVC에서 사용하도록 생성되는 기본 스토리지 클래스입니다.



참고

제공된 부팅 소스가 최신 버전의 운영 체제로 자동 업데이트됩니다. 자동 업데이트된 부팅 소스의 경우 PVC(영구 볼륨 클레임)가 클러스터의 기본 스토리지 클래스를 사용하여 생성됩니다. 구성 후 다른 기본 스토리지 클래스를 선택하는 경우 이전 기본 스토리지 클래스로 구성된 클러스터 네임스페이스의 기존 데이터 볼륨을 삭제해야 합니다.

9. 선택 사항: 최적화된 StorageProfile 설정을 적용하여 액세스 모드 또는 볼륨 모드를 편집합니다.

10. 다음과 같이 부팅 소스를 저장할 적절한 방법을 선택합니다.

    a. 로컬 파일을 업로드한 경우 저장 및 업로드를 클릭합니다.

    b. URL 또는 레지스트리에서 콘텐츠를 가져온 경우 저장 및 가져오기를 클릭합니다.

    c. 기존 PVC를 복제한 경우 저장 및 복제를 클릭합니다.

카탈로그 페이지에 부팅 소스가 포함된 사용자 정의 가상 머신 템플릿이 나열됩니다. 이 템플릿을 사용하여 가상 머신을 생성할 수 있습니다.

8.19.14.4. 연결된 부팅 소스를 사용하여 템플릿에서 가상 머신 생성

템플릿에 부팅 소스를 추가한 후 템플릿에서 가상 머신을 생성할 수 있습니다.

절차

1.

OpenShift Container Platform 웹 콘솔의 사이드 메뉴에서 가상화 → 카탈로그 를 클릭합니다.

2. 업데이트된 템플릿을 선택하고 Quick create VirtualMachine 을 클릭합니다.

VirtualMachine 세부 정보는 시작 상태와 함께 표시됩니다.

8.19.14.5. 사용자 정의 부팅 소스 생성

부팅 소스로 사용하기 위해 기존 디스크 이미지를 기반으로 사용자 지정 디스크 이미지를 준비할 수 있습니다.

다음 작업을 완료하려면 다음 절차를 사용하십시오.

- 사용자 지정 디스크 이미지 준비
- 사용자 지정 디스크 이미지에서 부팅 소스 생성
- 사용자 지정 템플릿에 부팅 소스 연결

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.
2. 사용자 지정할 템플릿의 부팅 소스 열에서 링크를 클릭합니다. 템플릿에 현재 정의된 소스가 있음을 나타내는 창이 표시됩니다.
3. 창에서 소스 사용자 지정 링크를 클릭합니다.
4. 부팅 소스 사용자 지정 프로세스에 대한 정보를 읽은 후에 사용자 지정 정보로 진행하려면 부팅 소스 사용자 지정 창에서 계속을 클릭합니다.

5. 부팅 소스 사용자 지정 준비 페이지의 새 템플릿 정의 섹션에서 다음을 수행합니다.

   a. 새 템플릿 네임스페이스 필드를 선택한 다음 프로젝트를 선택합니다.

   b. 새 템플릿 이름 필드에 사용자 지정 템플릿의 이름을 입력합니다.

   c. 새 템플릿 공급자 필드에 템플릿 프로바이더의 이름을 입력합니다.

   d. 새 템플릿 지원 필드를 선택한 다음, 생성한 사용자 지정 템플릿에 대한 지원 연락처를 나타내는 적절한 값을 선택합니다.

   e. 새 템플릿 플레이버 필드를 선택한 다음 생성한 사용자 지정 이미지의 적절한 **CPU** 및 메모리 값을 선택합니다.

6. 사용자 지정에 대한 부팅 소스 준비 섹션에서 필요한 경우 **cloud-init YAML** 스크립트를 사용자 지정하여 로그인 인증 정보를 정의합니다. 그렇지 않으면 스크립트가 기본 인증 정보를 생성합니다.

7. 사용자 지정 시작을 클릭합니다. 사용자 지정 프로세스가 시작되고 부팅 소스 사용자 지정 준비 페이지가 표시된 다음 부팅 소스 사용자 지정 페이지가 표시됩니다. 부팅 소스 사용자 지정 페이지에 실행 중인 스크립트의 출력이 표시됩니다. 스크립트가 완료되면 사용자 지정 이미지를 사용할 수 있습니다.

8. **VNC** 콘솔의 게스트 로그인 인증 정보 섹션에서 암호 표시를 클릭합니다. 로그인 인증 정보가 표시됩니다.

9. 이미지가 로그인할 준비가 되면 게스트 로그인 인증 정보 섹션에 표시된 사용자 이름과 암호를 제공하여 **VNC** 콘솔로 로그인합니다.

10. 사용자 지정 이미지가 예상대로 작동하는지 확인합니다. 부팅 가능한 경우 부팅 소스 사용 가능을 클릭합니다.

11. 사용자 지정 완료 및 템플릿 사용 가능 창에서 **I have sealed the boot source so it can be used as a template**를 선택한 다음 적용을 클릭합니다.

12. 부팅 소스 사용자 지정 완료 페이지에서 템플릿 생성 프로세스가 완료될 때까지 기다립니다. 템플릿 세부 정보 탐색또는 템플릿 목록으로 이동을 클릭하여 사용자 지정 부팅 소스에서 생성한 사용자 지정 템플릿을 확인합니다.

8.19.14.6. 추가 리소스

- 가상 머신 템플릿 생성
- 클라우드 이미지에서 Microsoft Windows 부팅 소스 생성
- OpenShift Container Platform에서 기존 Microsoft Windows 부팅 소스 사용자 정의
- CLI를 사용하여 PVC를 Microsoft Windows 템플릿의 부팅 소스로 설정
- 자동 스크립팅을 사용하여 부팅 소스 생성
- Pod 내에서 자동으로 부팅 소스 생성

8.19.15. 가상 디스크 핫플러그

가상 시스템 또는 가상 시스템 인스턴스를 중지하지 않고 가상 디스크를 추가하거나 제거하려면 해당 디스크를 핫 플러그 및 핫 플러그 해제합니다. 이 기능은 가동 중지 시간을 사용하지 않고 실행 중인 가상 시스템에 스토리지를 추가해야 하는 경우에 유용합니다.

가상 디스크를 *핫플러그*하는 경우 가상 머신이 실행되는 동안 가상 디스크를 가상 머신 인스턴스에 연결합니다.

가상 디스크를 *핫 플러그 해제*하는 경우 가상 머신이 실행되는 동안 가상 머신 인스턴스에서 가상 디스크를 분리합니다.

데이터 볼륨 및 PVC(영구 볼륨 클레임)만 핫 플러그로 연결할 수 있습니다. 컨테이너 디스크를 핫플러그 또는 핫 플러그 해제할 수 없습니다.

8.19.15.1. CLI를 사용하여 가상 디스크 핫플러그

가상 머신이 실행되는 동안 VMI(가상 머신 인스턴스)에 연결하려는 가상 디스크를 핫플러그합니다.

사전 요구 사항

- 가상 디스크를 핫 플러그하려면 실행 중인 가상 머신이 있어야 합니다.
- 핫 플러그에 사용할 수 있는 데이터 볼륨 또는 PVC(영구 볼륨 클레임)가 하나 이상 있어야 합니다.

절차

- 다음 명령을 실행하여 가상 디스크를 핫 플러그합니다.

```
$ virtctl addvolume <virtual-machine|virtual-machine-instance> --volume-name=
<datavolume|PVC> \
[--persist] [--serial=<label-name>]
```

  - 선택적 **--persist** 플래그를 사용하여 핫플러그 디스크를 가상 머신 사양에 영구적으로 마운트된 가상 디스크로 추가합니다. 가상 시스템을 중지, 다시 시작 또는 재부팅하여 가상 디스크를 영구적으로 마운트합니다. **--persist** 플래그를 지정한 후에는 더 이상 가상 디스크를 핫플러그하거나 연결할 수 없습니다. **--persist** 플래그는 가상 머신 인스턴스가 아닌 가상 머신에 적용됩니다.
  - 선택적 **--serial** 플래그를 사용하면 선택한 영숫자 문자열 레이블을 추가할 수 있습니다. 이렇게 하면 게스트 가상 시스템에서 핫플러그된 디스크를 식별하는 데 도움이 됩니다. 이 옵션을 지정하지 않으면 라벨의 기본값이 핫플러그된 데이터 볼륨 또는 PVC의 이름으로 설정됩니다.

8.19.15.2. CLI를 사용하여 가상 디스크 핫 플러그 연결

가상 머신이 실행되는 동안 VMI(가상 머신 인스턴스)에서 분리할 가상 디스크의 핫-언플러그입니다.

사전 요구 사항

- 가상 머신이 실행 중이어야 합니다.
-

하나 이상의 데이터 볼륨 또는 **PVC**(영구 볼륨 클레임)를 사용할 수 있고 핫플러그해야 합니다.

절차

- 다음 명령을 실행하여 가상 디스크의 핫 플러그를 해제합니다

```
$ virtctl removevolume <virtual-machine|virtual-machine-instance> --volume-name=
<datavolume|PVC>
```

8.19.15.3. 웹 콘솔을 사용하여 가상 디스크 핫플러그

가상 머신이 실행되는 동안 **VMI**(가상 머신 인스턴스)에 연결하려는 가상 디스크를 핫플러그합니다.

사전 요구 사항

- 가상 디스크를 핫 플러그하려면 실행 중인 가상 머신이 있어야 합니다.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachines** 를 클릭합니다.
2. 실행 중인 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.
3. 디스크 탭에서 디스크 추가를 클릭합니다.
4. 디스크 추가 창에서 핫플러그할 가상 디스크에 대한 정보를 입력합니다.
5. 추가를 클릭합니다.

8.19.15.4. 웹 콘솔을 사용하여 가상 디스크 핫플러그

가상 시스템이 실행되는 동안 **VMI**(가상 머신 인스턴스)에 연결하려는 가상 디스크를 핫-언플러그합니다.

사전 요구 사항

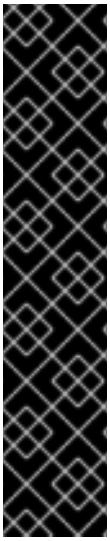
- 핫 플러그된 디스크를 연결하여 가상 시스템을 실행해야 합니다.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachines** 를 클릭합니다.
2. 핫 플러그 해제할 디스크로 실행 중인 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.
3. 디스크 탭에서 핫 언플러그하려는 가상 디스크의 옵션 메뉴  를 클릭합니다.
4. 삭제를 클릭합니다.

8.19.16. 가상 머신에서 컨테이너 디스크 사용

가상 머신 이미지를 컨테이너 디스크에 빌드하고 컨테이너 레지스트리에 저장할 수 있습니다. 그러면 컨테이너 디스크를 가상 머신의 영구 스토리지로 가져오거나 임시 저장을 위해 가상 머신에 직접 연결할 수 있습니다.



중요

대규모 컨테이너 디스크를 사용하는 경우 I/O 트래픽이 증가하여 작업자 노드에 영향을 미칠 수 있습니다. 이로 인해 노드를 사용할 수 없는 노드가 발생할 수 있습니다. 다음을 통해 해결할 수 있습니다.

- DeploymentConfig 오브젝트 정리
- 가비지 컬렉션 구성

8.19.16.1. 컨테이너 디스크 정보

컨테이너 디스크는 컨테이너 이미지 레지스트리에 컨테이너 이미지로 저장되어 있는 가상 머신 이미지입니다. 컨테이너 디스크를 사용하면 동일한 디스크 이미지를 여러 가상 머신에 제공하고 다수의 가상 머신 복제본을 생성할 수 있습니다.

컨테이너 디스크는 가상 머신에 연결된 데이터 볼륨을 사용하여 **PVC(**영구 볼륨 클레임**)**로 가져오거나 가상 머신에 임시 **containerDisk** 볼륨으로 직접 연결할 수 있습니다.

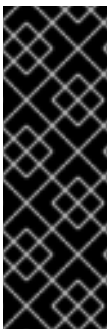
8.19.16.1.1. 데이터 볼륨을 사용하여 컨테이너 디스크를 PVC로 가져오기

CDI(Containerized Data Importer)를 사용하여 데이터 볼륨을 통해 컨테이너 디스크를 **PVC**로 가져옵니다. 그러면 영구 저장을 위해 데이터 볼륨을 가상 머신에 연결할 수 있습니다.

8.19.16.1.2. 컨테이너 디스크를 가상 머신에 containerDisk 볼륨으로 연결

containerDisk는 임시 볼륨입니다. 이 볼륨은 가상 머신이 중지, 재시작 또는 삭제될 때 삭제됩니다. **containerDisk** 볼륨이 포함된 가상 머신이 시작되면 레지스트리에서 컨테이너 이미지가 풀링되어 가상 머신이 호스팅되는 노드에서 호스팅됩니다.

CD-ROM과 같은 읽기 전용 파일 시스템이나 일회용 가상 머신에는 **containerDisk** 볼륨을 사용하십시오.



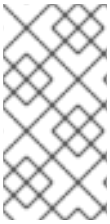
중요

데이터가 호스팅 노드의 로컬 스토리지에 임시로 기록되므로 읽기-쓰기 파일 시스템에는 **containerDisk** 볼륨을 사용하지 않는 것이 좋습니다. 이 볼륨을 사용하면 데이터를 대상 노드로 마이그레이션해야 하므로 노드 유지보수의 경우와 같이 가상 머신의 실시간 마이그레이션 속도가 느려집니다. 또한 노드의 전원이 끊기거나 노드가 예기치 않게 종료되면 모든 데이터가 손실됩니다.

8.19.16.2. 가상 머신용 컨테이너 디스크 준비

컨테이너 디스크를 가상 머신에서 사용하려면 가상 머신 이미지를 사용하여 빌드하고 컨테이너 레지스트리에 푸시해야 합니다. 그러면 데이터 볼륨을 사용하여 컨테이너 디스크를 **PVC**로 가져와서 가상 머신에 연결하거나 컨테이너 디스크를 임시 **containerDisk** 볼륨으로 가상 머신에 직접 연결할 수 있습니다.

컨테이너 디스크 내부의 디스크 이미지의 크기는 컨테이너 디스크가 호스팅되는 레지스트리의 최대 계층 크기로 제한됩니다.



참고

Red Hat Quay 의 경우 **Red Hat Quay**를 처음 배포할 때 생성되는 **YAML** 구성 파일을 편집하여 최대 계층 크기를 변경할 수 있습니다.

사전 요구 사항

- **podman**을 아직 설치하지 않은 경우 설치합니다.
- 가상 머신 이미지는 **QCOW2** 또는 **RAW** 형식이어야 합니다.

절차

1. **Dockerfile**을 생성하여 가상 머신 이미지를 컨테이너 이미지로 빌드합니다. 가상 머신 이미지는 **UID**가 **107**인 **QEMU**에 속하고 컨테이너 내부의 **/disk/** 디렉터리에 있어야 합니다. 그런 다음 **/disk/** 디렉터리에 대한 권한을 **0440**으로 설정해야 합니다.

   다음 예제에서는 **Red Hat UBI(Universal Base Image)**를 사용하여 첫 번째 단계에서 이러한 구성 변경을 처리하고, 두 번째 단계에서 최소 **scratch** 이미지를 사용하여 결과를 저장합니다.

```
$ cat > Dockerfile << EOF
FROM registry.access.redhat.com/ubi8/ubi:latest AS builder
ADD --chown=107:107 <vm_image>.qcow2 /disk/ 1
RUN chmod 0440 /disk/*

FROM scratch
COPY --from=builder /disk/* /disk/
EOF
```

   1. 여기서 **<vm_image>**는 **QCOW2** 또는 **RAW** 형식의 가상 머신 이미지입니다. 원격 가상 머신 이미지를 사용하려면 **<vm_image>.qcow2**를 원격 이미지의 전체 **URL**로 교체하십시오.

2. 컨테이너를 빌드하고 태그를 지정합니다.

```
$ podman build -t <registry>/<container_disk_name>:latest .
```

3. 컨테이너 이미지를 레지스트리에 푸시합니다.

```
$ podman push <registry>/<container_disk_name>:latest
```

컨테이너 레지스트리에 TLS가 없는 경우 컨테이너 디스크를 영구 스토리지로 가져오기 전에 이를 비보안 레지스트리로 추가해야 합니다.

8.19.16.3. 컨테이너 레지스트리에서 비보안 레지스트리를 사용하도록 TLS 비활성화

HyperConverged 사용자 정의 리소스의 insecureRegistries 필드를 편집하여 하나 이상의 컨테이너 레지스트리에 대해 TLS(전송 계층 보안)를 비활성화할 수 있습니다.

사전 요구 사항

- cluster-admin 역할의 사용자로 클러스터에 로그인합니다.

절차

- HyperConverged 사용자 지정 리소스를 편집하고 비보안 레지스트리 목록을 spec.storageImport.insecureRegistries 필드에 추가합니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  storageImport:
    insecureRegistries: 1
      - "private-registry-example-1:5000"
      - "private-registry-example-2:5000"
```

1

이 목록의 예제를 유효한 레지스트리 호스트 이름으로 바꿉니다.

8.19.16.4. 다음 단계

- 컨테이너 디스크를 가상 머신의 영구 스토리지로 가져옵니다.
- 임시 저장을 위해 containerDisk 볼륨을 사용하는 가상 머신을 생성합니다.

8.19.17. CDI 스크래치 공간 준비

8.19.17.1. 데이터 볼륨 정보

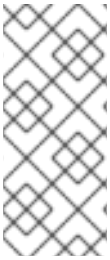
Dataolume 오브젝트는 **CDI(Containerized Data Importer)** 프로젝트에서 제공하는 사용자 정의 리소스입니다. 데이터 볼륨은 기본 **PVC**(영구 볼륨 클레임)와 관련된 가져오기, 복제, 업로드 작업을 오케스트레이션합니다. 데이터 볼륨은 **OpenShift Virtualization**과 통합되며 **PVC**가 준비되기 전에 가상 머신이 시작되지 않도록 합니다.

8.19.17.2. 스크래치 공간 정보

CDI(Containerized Data Importer)에는 가상 머신 이미지 가져오기 및 업로드와 같은 일부 작업을 완료하기 위해 스크래치 공간(임시 스토리지)이 필요합니다. 이 프로세스 동안 **CDI**는 대상 **DV**(데이터 볼륨)를 지원하는 **PVC** 크기와 같은 스크래치 공간 **PVC**를 프로비저닝합니다. 스크래치 공간 **PVC**는 작업이 완료되거나 중단된 후 삭제됩니다.

HyperConverged사용자 지정 리소스의 **spec.scratchSpaceStorageClass** 필드에서 스크래치 공간 **PVC**를 바인딩하는 데 사용되는 스토리지 클래스를 정의할 수 있습니다.

정의된 스토리지 클래스가 클러스터의 스토리지 클래스와 일치하지 않으면 클러스터에 정의된 기본 스토리지 클래스가 사용됩니다. 클러스터에 기본 스토리지 클래스가 정의되어 있지 않은 경우에는 원래 **DV** 또는 **PVC**를 프로비저닝하는 데 사용된 스토리지 클래스가 사용됩니다.



참고

CDI에서는 원본 데이터 볼륨을 지원하는 **PVC**에 관계없이 **file** 볼륨 모드로 스크래치 공간을 요청해야 합니다. 원본 **PVC**를 **block** 볼륨 모드로 지원하는 경우 **file** 볼륨 모드 **PVC**를 프로비저닝할 수 있는 스토리지 클래스를 정의해야 합니다.

수동 프로비저닝

스토리지 클래스가 없는 경우 **CDI**는 프로젝트에서 이미지의 크기 요구 사항과 일치하는 **PVC**를 사용합니다. 이러한 요구 사항과 일치하는 **PVC**가 없는 경우에는 **CDI** 가져오기 **Pod**가 적절한 **PVC**를 사용할 수 있거나 타임아웃 기능에서 **Pod**를 종료할 때까지 **Pending** 상태로 유지됩니다.

8.19.17.3. 스크래치 공간이 필요한 CDI 작업

유형	이유

유형	이유
레지스트리 가져오기	CDI에서는 이미지를 스크래치 공간으로 다운로드하고 레이어를 추출하여 이미지 파일을 찾아야 합니다. 그런 다음 해당 이미지 파일을 원시 디스크로 변환하기 위해 QEMU-IMG로 전달합니다.
이미지 업로드	QEMU-IMG에서는 STDIN의 입력을 허용하지 않습니다. 대신 변환을 위해 QEMU-IMG로 전달할 수 있을 때까지 업로드할 이미지를 스크래치 공간에 저장합니다.
보관된 이미지의 HTTP 가져오기	QEMU-IMG에서는 CDI에서 지원하는 아카이브 형식 처리 방법을 확인할 수 없습니다. 대신, QEMU-IMG에 전달할 때까지 해당 이미지를 보관하지 않고 스크래치 공간에 저장합니다.
인증된 이미지의 HTTP 가져오기	QEMU-IMG에서 인증을 부적절하게 처리합니다. 대신, QEMU-IMG로 전달할 때까지 이미지를 스크래치 공간에 저장하고 인증합니다.
사용자 정의 인증서의 HTTP 가져오기	QEMU-IMG에서는 HTTPS 끝점의 사용자 정의 인증서를 부적절하게 처리합니다. 대신, CDI에서는 파일을 QEMU-IMG에 전달할 때까지 이미지를 스크래치 공간에 다운로드합니다.

8.19.17.4. 스토리지 클래스 정의

spec.scratchSpaceStorageClass 필드를 HyperConverged CR(사용자 정의 리소스)에 추가하여 CR(Containerized Data Importer)에서 스크래치 공간을 할당할 때 사용하는 스토리지 클래스를 정의할 수 있습니다.

사전 요구 사항

- **OpenShift CLI(oc)를 설치합니다.**

절차

1. **다음 명령을 실행하여 HyperConverged CR을 편집합니다.**

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

2. **spec.scratchSpaceStorageClass 필드를 CR에 추가하여 해당 값을 클러스터에 존재하는 스토리지 클래스의 이름으로 설정합니다.**

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
spec:
  scratchSpaceStorageClass: "<storage_class>" 1
```

1

스토리지 클래스를 지정하지 않으면 **CDI**는 채워지는 영구 볼륨 클레임의 스토리지 클래스를 사용합니다.

3. 기본 편집기를 저장하고 종료하여 **HyperConverged CR**을 업데이트합니다.

8.19.17.5. CDI 지원 작업 매트릭스

이 매트릭스에는 끝점에 대한 콘텐츠 유형에 따라 지원되는 **CDI** 작업과 이러한 작업 중 스크래치 공간이 필요한 작업이 표시되어 있습니다.

콘텐츠 유형	HTTP	HTTPS	HTTP 기본 인증	레지스트리	업로드
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* □ GZ □ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt(RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* □ GZ □ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 지원되는 작업

□ 지원되지 않는 작업

* 스크래치 공간 필요

** 사용자 정의 인증 기관이 필요한 경우 스크래치 공간 필요

8.19.17.6. 추가 리소스

- 동적 프로비저닝

8.19.18. 영구 볼륨 다시 사용

정적으로 프로비저닝된 PV(영구 볼륨)를 다시 사용하려면 먼저 볼륨을 회수해야 합니다. 이를 위해서는 스토리지 구성을 다시 사용할 수 있도록 PV를 삭제해야 합니다.

8.19.18.1. 정적으로 프로비저닝된 영구 볼륨 회수 정보

PV(영구 볼륨)를 회수할 때는 PVC(영구 볼륨 클레임)에서 PV를 바인딩 해제하고 PV를 삭제합니다. 기본 스토리지에 따라 공유 스토리지를 수동으로 삭제해야 할 수도 있습니다.

그런 다음 PV 구성을 다시 사용하여 다른 이름으로 PV를 생성할 수 있습니다.

정적으로 프로비저닝된 PV를 회수하려면 PV에 Retain이라는 회수 정책이 있어야 합니다. 회수 정책이 없으면 PVC를 PV에서 바인딩 해제할 때 PV의 상태가 실패가 됩니다.

중요

OpenShift Container Platform 4에서는 Recycle 회수 정책이 사용되지 않습니다.

8.19.18.2. 정적으로 프로비저닝된 영구 볼륨 회수

PVC(영구 볼륨 클레임)를 바인딩 해제하고 PV를 삭제하여 정적으로 프로비저닝된 PV(영구 볼륨)를 회수합니다. 공유 스토리지를 수동으로 삭제해야 할 수도 있습니다.

정적으로 프로비저닝된 PV를 회수하는 방법은 기본 스토리지에 따라 다릅니다. 이 절차에서는 일반적인 접근법을 제공하며 사용 중인 스토리지에 따라 사용자 정의가 필요할 수 있습니다.

절차

1. PV의 회수 정책이 Retain으로 설정되어 있는지 확인합니다.

   a. PV의 회수 정책을 확인합니다.

```
$ oc get pv <pv_name> -o yaml | grep 'persistentVolumeReclaimPolicy'
```

b. **persistentVolumeReclaimPolicy**가 **Retain**으로 설정되지 않은 경우, 다음 명령을 사용하여 회수 정책을 편집합니다.

```
$ oc patch pv <pv_name> -p '{"spec":
{"persistentVolumeReclaimPolicy":"Retain"}}'
```

2. **PV**를 사용하는 리소스가 없는지 확인합니다.

```
$ oc describe pvc <pvc_name> | grep 'Mounted By:'
```

PVC를 사용하는 모든 리소스를 제거한 후 계속합니다.

3. **PVC**를 삭제하여 **PV**를 해제합니다.

```
$ oc delete pvc <pvc_name>
```

4. 선택 사항: **PV** 구성을 **YAML** 파일로 내보냅니다. 이 절차의 뒷부분에서 공유 스토리지를 수동으로 제거하는 경우 이 구성을 참조할 수 있습니다. **PV**를 회수한 후 새 **PV**를 동일한 스토리지 구성으로 생성하기 위해 이 파일의 **spec** 매개변수를 기반으로 사용할 수도 있습니다.

```
$ oc get pv <pv_name> -o yaml > <file_name>.yaml
```

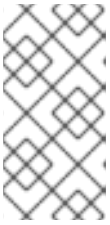
5. **PV**를 삭제합니다.

```
$ oc delete pv <pv_name>
```

6. 선택 사항: 스토리지 유형에 따라 공유 스토리지 폴더의 내용을 제거해야 할 수 있습니다.

```
$ rm -rf <path_to_share_storage>
```

7. 선택 사항: 삭제된 **PV**와 동일한 스토리지 구성을 사용하는 **PV**를 생성합니다. 회수된 **PV** 구성을 이전에 내보낸 경우 해당 파일의 **spec** 매개변수를 새 **PV** 매니페스트의 기반으로 사용할 수 있습니다.



참고

충돌을 피하려면 새 PV 오브젝트에 삭제한 오브젝트와 다른 이름을 지정하는 것이 좋습니다.

```
$ oc create -f <new_pv_name>.yaml
```

추가 리소스

- 가상 머신 로컬 스토리지 구성
- OpenShift Container Platform Storage 설명서에는 영구 스토리지에 대한 자세한 내용이 있습니다.

8.19.19. 가상 머신 디스크 확장

가상 머신(VM) 디스크 크기를 확장하여 디스크의 PVC(영구 볼륨 클레임)를 조정하여 더 큰 스토리지 용량을 제공할 수 있습니다.

그러나 VM 디스크 크기를 줄일 수는 없습니다.

8.19.19.1. 가상 머신 디스크 활성화

VM 디스크 확대를 사용하면 가상 머신에서 추가 공간을 사용할 수 있습니다. 그러나 VM 소유자가 스토리지를 사용하는 방법을 결정할 책임이 있습니다.

디스크가 Filesystem PVC인 경우 일치하는 파일은 파일 시스템 오버헤드를 위해 일부 공간을 예약하는 동안 나머지 크기로 확장됩니다.

절차

1. 확장할 VM 디스크의 PersistentVolumeClaim 매니페스트를 편집합니다.

```
$ oc edit pvc <pvc_name>
```

2. spec.resource.requests.storage 속성 값을 더 큰 크기로 변경합니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: vm-disk-expand
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 3Gi (1)
...
```

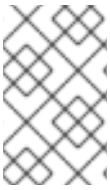
(1) VM 디스크 크기를 늘릴 수 있습니다.

8.19.19.2. 추가 리소스

- Windows에서 기본 볼륨 확장.
- Red Hat Enterprise Linux에서 데이터를 삭제하지 않고 기존 파일 시스템 파티션 확장.
- Red Hat Enterprise Linux에서 온라인으로 논리 볼륨 및 파일 시스템 확장.

8.19.20. 데이터 볼륨 삭제

oc 명령줄 인터페이스를 사용하여 데이터 볼륨을 수동으로 삭제할 수 있습니다.



참고

가상 머신을 삭제하면 사용하는 데이터 볼륨이 자동으로 삭제됩니다.

8.19.20.1. 데이터 볼륨 정보

Dataolume 오브젝트는 CDI(Containerized Data Importer) 프로젝트에서 제공하는 사용자 정의 리소스입니다. 데이터 볼륨은 기본 PVC(영구 볼륨 클레임)와 관련된 가져오기, 복제, 업로드 작업을 오케스트레이션합니다. 데이터 볼륨은 OpenShift Virtualization과 통합되며 PVC가 준비되기 전에 가상 머신이 시작되지 않도록 합니다.

8.19.20.2. 모든 데이터 볼륨 나열

oc 명령줄 인터페이스를 사용하여 클러스터의 데이터 볼륨을 나열할 수 있습니다.

절차

- 다음 명령을 실행하여 모든 데이터 볼륨을 나열합니다.

```
$ oc get dvs
```

8.19.20.3. 데이터 볼륨 삭제

oc CLI(명령줄 인터페이스)를 사용하여 데이터 볼륨을 삭제할 수 있습니다.

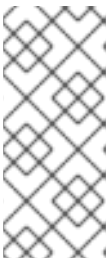
사전 요구 사항

- 삭제할 데이터 볼륨의 이름을 확인합니다.

절차

- 다음 명령을 실행하여 데이터 볼륨을 삭제합니다.

```
$ oc delete dv <datavolume_name>
```



참고

이 명령은 현재 프로젝트에 존재하는 오브젝트만 삭제합니다. 삭제하려는 오브젝트가 다른 프로젝트 또는 네임스페이스에 있는 경우 **-n <project_name>** 옵션을 지정하십시오.

9장. 가상 머신 템플릿

9.1. 가상 머신 템플릿 생성

9.1.1. 가상 머신 템플릿 정보

사전 구성 **Red Hat** 가상 머신 템플릿은 가상화 → 템플릿 페이지에 나열됩니다. 이러한 템플릿은 **Red Hat Enterprise Linux, Fedora, Microsoft Windows 10, Microsoft Windows Servers**의 다양한 버전에서 사용할 수 있습니다. 각 **Red Hat** 가상 머신 템플릿은 운영 체제 이미지, 운영 체제, 플레이버(**CPU** 및 메모리), 워크로드 유형(**server**)의 기본 설정으로 사전 구성됩니다.

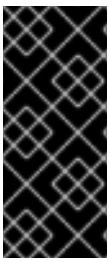
템플릿 페이지에는 다음 네 가지 유형의 가상 머신 템플릿이 표시됩니다.

- **Red Hat** 지원 템플릿은 **Red Hat**에서 완전하게 지원됩니다.
- 사용자 지원 템플릿은 사용자가 복제 및 생성한 **Red Hat** 지원 템플릿입니다.
- **Red Hat** 제공 템플릿은 **Red Hat**이 제한적으로 지원합니다.
- 사용자 제공 템플릿은 사용자가 복제 및 생성한 **Red Hat** 제공 템플릿입니다.

템플릿 카탈로그 의 필터를 사용하여 부팅 소스 가용성, 운영 체제 및 워크로드와 같은 속성에 따라 템플릿을 정렬할 수 있습니다.

Red Hat 지원 또는 **Red Hat** 제공 템플릿은 편집하거나 삭제할 수 없습니다. 템플릿을 복제하고 사용자 지정 가상 머신 템플릿으로 저장한 다음 편집할 수 있습니다.

YAML 파일 예제를 편집하여 사용자 지정 가상 머신 템플릿을 생성할 수도 있습니다.



중요

스토리지 동작의 차이로 인해 일부 가상 머신 템플릿은 단일 노드 **OpenShift**와 호환되지 않습니다. 호환성을 보장하기 위해 데이터 볼륨 또는 스토리지 프로필을 사용하는 템플릿 또는 가상 머신의 **evictionStrategy** 필드를 설정하지 마십시오.

9.1.2. 가상 머신 및 부팅 소스 정보

가상 시스템은 가상 시스템 정의와 데이터 볼륨에서 지원하는 하나 이상의 디스크로 구성됩니다. 가상 머신 템플릿을 사용하면 사전 정의된 가상 머신 사양을 사용하여 가상 머신을 생성할 수 있습니다.

모든 가상 머신 템플릿에는 구성된 드라이버를 포함하여 완전히 구성된 가상 머신 디스크 이미지인 부팅 소스가 필요합니다. 각 가상 머신 템플릿에는 부팅 소스에 대한 포인터가 있는 가상 시스템 정의가 포함되어 있습니다. 각 부팅 소스에는 사전 정의된 이름과 네임스페이스가 있습니다. 일부 운영 체제의 경우 부팅 소스가 자동으로 제공됩니다. 제공되지 않는 경우 관리자는 사용자 지정 부팅 소스를 준비해야 합니다.

제공된 부팅 소스가 최신 버전의 운영 체제로 자동 업데이트됩니다. 자동 업데이트됨 부팅 소스의 경우 **PVC**(영구 볼륨 클레임)가 클러스터의 기본 스토리지 클래스를 사용하여 생성됩니다. 구성 후 다른 기본 스토리지 클래스를 선택하는 경우 이전 기본 스토리지 클래스로 구성된 클러스터 네임스페이스의 기존 데이터 볼륨을 삭제해야 합니다.

부팅 소스 기능을 사용하려면 **OpenShift Virtualization**의 최신 릴리스를 설치합니다. 네임스페이스 **openshift-virtualization-os-images**는 기능을 활성화하고 **OpenShift Virtualization Operator**와 함께 설치됩니다. 부팅 소스 기능이 설치되면 부팅 소스를 생성하고 템플릿에 연결한 다음 템플릿에서 가상 머신을 생성할 수 있습니다.

로컬 파일 업로드, 기존 **PVC** 복제, 레지스트리에서 가져오기 또는 **URL**을 통해 채워지는 **PVC**(영구 볼륨 클레임)를 사용하여 부팅 소스를 정의합니다. 웹 콘솔을 사용하여 가상 머신 템플릿에 부팅 소스를 연결합니다. 부팅 소스를 가상 머신 템플릿에 연결한 후 템플릿에서 완전히 구성된 즉시 사용할 수 있는 가상 시스템을 원하는 만큼 생성합니다.

9.1.3. 웹 콘솔에서 가상 머신 템플릿 생성

OpenShift Container Platform 웹 콘솔에서 **YAML** 파일 예제를 편집하여 가상 머신 템플릿을 생성합니다.

절차

1. 웹 콘솔의 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.
2. 템플릿 생성을 클릭합니다.
3. **YAML** 파일을 편집하여 템플릿 매개변수를 지정합니다.

4. 생성을 클릭합니다.

템플릿이 템플릿 페이지에 표시됩니다.

5. 선택 사항: **YAML** 파일을 다운로드하여 저장하려면 다운로드를 클릭합니다.

9.1.4. 가상 머신 템플릿용 부팅 소스 추가

가상 머신 또는 사용자 지정 템플릿을 생성하기 위해 사용할 가상 머신 템플릿을 위한 부팅 소스를 구성할 수 있습니다. 가상 머신 템플릿이 부팅 소스로 구성된 경우 템플릿 페이지에서 사용할 수 있는 **Source** 로 레이블이 지정됩니다. 템플릿에 부팅 소스를 추가한 후 템플릿에서 새 가상 머신을 생성할 수 있습니다.

다음과 같은 4가지 방법으로 웹 콘솔에서 부팅 소스를 선택하고 추가할 수 있습니다.

- 로컬 파일 업로드 (**PVC** 생성)
- **URL** (**PVC** 생성)
- 복제(**PVC** 생성)
- 레지스트리(**PVC** 생성)

사전 요구 사항

- 부팅 소스를 추가하려면, **os-images.kubevirt.io:edit RBAC** 역할의 사용자 또는 관리자로 로그인해야 합니다. 부팅 소스가 추가된 템플릿에서 가상 머신을 생성하려면 특정 권한이 필요하지 않습니다.
- 로컬 파일을 업로드하려면 운영 체제 이미지 파일이 로컬 머신에 있어야 합니다.
- **URL**을 통해 가져오려면 운영 체제 이미지를 사용하여 웹 서버에 액세스해야 합니다. 예를 들면 이미지가 포함된 **Red Hat Enterprise Linux** 웹 페이지입니다.

- 기존 PVC를 복제하려면 PVC를 사용하여 프로젝트에 대한 액세스가 필요합니다.
- 레지스트리를 통해 가져오려면 컨테이너 레지스트리에 대한 액세스가 필요합니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.
2. 템플릿 옆에 있는 옵션 메뉴를 클릭하고 부팅 소스 편집 을 선택합니다.
3. 디스크 추가를 클릭합니다.
4. 디스크 추가 창에서 이 디스크 사용을 부팅 소스로 선택합니다.
5. 디스크 이름을 입력하고 소스(예 : Blank(PVC 생성)) 를 선택하거나 기존 PVC 사용.
6. 영구 볼륨 클레임(PVC) 크기에 값을 입력하여 압축이 해제되지 않은 이미지에 적합한 PVC 크기를 지정하고 필요한 추가 공간을 지정합니다.
7. 유형 (예: 디스크 또는 CD-ROM)을 선택합니다.
8. 선택 사항: 스토리지 클래스 를 클릭하고 디스크를 생성하는 데 사용되는 스토리지 클래스를 선택합니다. 일반적으로 이 스토리지 클래스는 모든 PVC에서 사용하도록 생성되는 기본 스토리지 클래스입니다.

참고

제공된 부팅 소스가 최신 버전의 운영 체제로 자동 업데이트됩니다. 자동 업데이트됨 부팅 소스의 경우 PVC(영구 볼륨 클레임)가 클러스터의 기본 스토리지 클래스를 사용하여 생성됩니다. 구성 후 다른 기본 스토리지 클래스를 선택하는 경우 이전 기본 스토리지 클래스로 구성된 클러스터 네임스페이스의 기존 데이터 볼륨을 삭제해야 합니다.

9.

선택 사항: 최적화된 **StorageProfile** 설정을 적용하여 액세스 모드 또는 볼륨 모드를 편집합니다.

10. 다음과 같이 부팅 소스를 저장할 적절한 방법을 선택합니다.

    a. 로컬 파일을 업로드한 경우 저장 및 업로드를 클릭합니다.

    b. **URL** 또는 레지스트리에서 콘텐츠를 가져온 경우 저장 및 가져오기를 클릭합니다.

    c. 기존 **PVC**를 복제한 경우 저장 및 복제를 클릭합니다.

카탈로그 페이지에 부팅 소스가 포함된 사용자 정의 가상 머신 템플릿이 나열됩니다. 이 템플릿을 사용하여 가상 머신을 생성할 수 있습니다.

9.1.4.1. 부팅 소스를 추가하기 위한 가상 머신 템플릿 필드

다음 표에서는 템플릿에 부팅 소스 추가 창의 필드에 대해 설명합니다. 이 창은 가상화 → 템플릿 페이지에서 가상 머신 템플릿에 대한 소스 추가를 클릭하면 표시됩니다.

이름	매개변수	설명
부팅 소스 유형	로컬 파일 업로드(PVC 생성)	로컬 장치에서 파일을 업로드합니다. gz, xz, tar, qcow2 등의 파일 형식이 지원됩니다.
	URL (PVC 생성)	HTTP 또는 HTTPS 끝점의 사용 가능한 이미지에서 콘텐츠를 가져옵니다. 이미지 다운로드를 사용할 수 있는 웹 페이지에서 다운로드 링크 URL을 가져와서 URL **가져오기 필드에 해당 URL** 링크를 입력합니다. 예: Red Hat Enterprise Linux 이미지의 경우 Red Hat Customer Portal에 로그온하고, 이미지 다운로드 페이지에 액세스한 후 KVM 게스트 이미지의 다운로드 링크 URL을 복사합니다.
	PVC(PVC 생성)	클러스터에서 이미 사용 가능한 PVC를 사용하여 복제합니다.
	레지스트리(PVC 생성)	클러스터에서 액세스할 수 있고 레지스트리에 위치한 부팅 가능한 운영 체제 컨테이너를 지정합니다. 예를 들면, kubevirt/cirros-registry-dis-demo입니다.

이름	매개변수	설명
소스 제공자		선택적 필드입니다. 템플릿을 만든 사용자의 소스 또는 템플릿을 만든 사용자 이름에 대한 설명 텍스트를 추가합니다. 예: Red Hat.
고급 스토리지 설정	StorageClass	디스크를 만드는 데 사용되는 스토리지 클래스입니다.
	액세스 모드	영구 볼륨의 액세스 모드입니다. 지원되는 액세스 모드는 **단일 사용자(RWO)**, **공유 액세스(RWX)**, **읽기 전용(ROX)**입니다. **단일 사용자(RWO)**를 선택하면 단일 노드에서 읽기/쓰기로 디스크를 마운트할 수 있습니다. **공유 액세스(RWX)**를 선택하면 여러 노드에서 읽기-쓰기로 디스크를 마운트할 수 있습니다. **kubevirt-storage-class-defaults** 구성 맵에서는 데이터 볼륨에 대한 액세스 모드 기본값을 제공합니다. 기본값은 클러스터의 각 스토리지 클래스에 대한 최상의 옵션에 따라 설정됩니다. **참고** 공유 액세스(RWX)는 가상 머신의 노드 간 실시간 마이그레이션 등 일부 기능에 필요합니다.
	볼륨 모드	영구 볼륨에서 포맷된 파일 시스템을 사용하는지 또는 원시 블록 상태를 사용하는지를 정의합니다. 지원되는 모드는 **블록** 및 **파일 시스템**입니다. **kubevirt-storage-class-defaults** 구성 맵에서는 데이터 볼륨에 대한 볼륨 모드 기본값을 제공합니다. 기본값은 클러스터의 각 스토리지 클래스에 대한 최상의 옵션에 따라 설정됩니다.

9.1.5. 가상 머신 템플릿을 즐겨 찾기로 표시

자주 사용하는 템플릿을 즐겨 찾기로 표시할 수 있습니다.

절차

1. **OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.**
2. **템플릿 옆에 있는 별 아이콘을 클릭하여 즐겨 찾기로 표시합니다.**

즐겨 찾는 템플릿은 템플릿 목록의 맨 위에 표시됩니다.

9.1.6. 공급자별 가상 머신 템플릿 목록 필터링

템플릿 탭에서 템플릿의 이름을 지정하거나 템플릿을 식별하는 라벨을 지정하여 이름으로 검색 필드로 가상 머신 템플릿을 검색할 수 있습니다. 공급자가 템플릿을 필터링하고 필터링 기준을 충족하는 템플릿만 표시할 수도 있습니다.

절차

1. **OpenShift Virtualization** 콘솔의 사이드 메뉴에서 워크로드 → 가상화를 클릭합니다.
2. 템플릿 탭을 클릭합니다.
3. 템플릿을 필터링하려면 필터를 클릭합니다.
4. 목록에서 **Red Hat** 지원, 사용자 지원, **Red Hat** 제공 및 사용자 제공 중에서 해당하는 확인란을 선택하여 템플릿을 필터링합니다.

9.1.7. 추가 리소스

- 부팅 소스 생성 및 사용
- 스토리지 프로파일 사용자 정의

9.2. 가상 머신 템플릿 편집

웹 콘솔에서 가상 머신 템플릿을 편집할 수 있습니다.

참고

Red Hat Virtualization Operator에서 제공하는 템플릿을 편집할 수 없습니다. 템플릿을 복제하면 편집할 수 있습니다.

9.2.1. 웹 콘솔에서 가상 머신 템플릿 편집

관련 필드 옆에 있는 연필 아이콘을 클릭하여 웹 콘솔의 웹 콘솔에서 가상 머신 템플릿에 대해 선택된 값을 편집합니다. CLI를 사용하여 다른 값을 편집할 수 있습니다.

사전 구성 Red Hat 템플릿과 사용자 지정 가상 머신 템플릿 모두에 대해 라벨 및 주석을 편집할 수 있습니다. 다른 모든 값은 Red Hat 템플릿 또는 가상 머신 템플릿 생성 마법사를 사용하여 생성한 사용자 지정 가상 머신 템플릿만 편집할 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.

2. 선택 사항: 필터 드롭다운 메뉴를 사용하여 상태, 템플릿, 노드 또는 운영 체제(OS)와 같은 속성으로 가상 머신 템플릿 목록을 정렬합니다.

3. 가상 머신 템플릿을 선택하여 템플릿 세부 정보 페이지를 엽니다.

4. 연필 아이콘을 클릭하여 필드를 편집할 수 있도록 합니다.

5. 관련 사항을 변경하고 저장을 클릭합니다.

가상 머신 템플릿을 편집해도 해당 템플릿에서 이미 생성된 가상 머신에는 영향을 미치지 않습니다.

9.2.1.1. 가상 머신 템플릿 필드

다음 표에는 OpenShift Container Platform 웹 콘솔에서 편집할 수 있는 가상 머신 템플릿 필드가 나열되어 있습니다.

표 9.1. 가상 머신 템플릿 필드

탭	필드 또는 기능

탭	필드 또는 기능
세부 정보	• 라벨 • 주석 • 표시 이름 • 설명 • 워크로드 프로필 • CPU/Memory • 부팅 모드 • GPU 장치 • 호스트 장치
YAML	• 사용자 정의 리소스를 보고 편집하거나 다운로드합니다.
스케줄링	• 노드 선택기 • 허용 오차 • 유사성 규칙 • 전용 리소스 • 제거 전략 • Descheduler 설정
네트워크 인터페이스	• 네트워크 인터페이스를 추가, 편집 또는 삭제합니다.
디스크	• 디스크를 추가, 편집 또는 삭제합니다.
스크립트	• cloud-init 설정
매개변수(선택 사항)	• 가상 머신 이름 • cloud-user 암호

9.2.1.2. 가상 머신 템플릿에 네트워크 인터페이스 추가

네트워크 인터페이스를 가상 머신 템플릿에 추가하려면 이 절차를 사용하십시오.

절차

1. 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.
2. 가상 머신 템플릿을 선택하여 템플릿 세부 정보 화면을 엽니다.
3. 네트워크 인터페이스 탭을 클릭합니다.
4. 네트워크 인터페이스 추가를 클릭합니다.
5. 네트워크 인터페이스 추가 창에서 네트워크 인터페이스의 이름, 모델, 네트워크, 유형, MAC 주소를 지정합니다.
6. 추가를 클릭합니다.

9.2.1.3. 가상 머신 템플릿에 가상 디스크 추가

가상 디스크를 가상 머신 템플릿에 추가하려면 이 절차를 사용하십시오.

절차

1. 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.
2. 가상 머신 템플릿을 선택하여 템플릿 세부 정보 화면을 엽니다.
3. 디스크 탭을 클릭한 다음 디스크 추가를 클릭합니다.
4. 디스크 추가 창에서 소스 ,이름 , 크기,유형,인터페이스, 스토리지 클래스를 지정합니다.
    a.

선택 사항: 빈 디스크 소스를 사용하고 데이터 볼륨을 생성할 때 최대 쓰기 성능이 필요한 경우 사전 할당을 활성화할 수 있습니다. 이를 수행하려면 사전 할당 활성화 확인란을 선택합니다.

b. 선택 사항: 최적화된 **StorageProfile** 설정 적용을 지우고 가상 디스크의 볼륨 모드 및 액세스 모드를 변경할 수 있습니다. 이러한 매개변수를 지정하지 않으면 **kubevirt-storage-class-defaults** 구성 맵의 기본값이 사용됩니다.

5. 추가를 클릭합니다.

9.2.1.4. 템플릿용 CD-ROM 편집

가상 머신 템플릿에 대해 **CD-ROM**을 편집하려면 다음 절차를 사용하십시오.

절차

1. 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.
2. 가상 머신 템플릿을 선택하여 템플릿 세부 정보 화면을 엽니다.
3. 디스크 탭을 클릭합니다.
4. 편집하려는 **CD-ROM**의 옵션 메뉴  를 클릭한 후 편집을 선택합니다.
5. **CD-ROM** 편집 창에서 소스, 영구 볼륨 클레임, 이름, 유형 및 인터페이스 필드를 편집합니다.
6. 저장을 클릭합니다.

9.3. 가상 머신 템플릿 전용 리소스 활성화

가상 머신은 성능 향상을 위해 CPU와 같은 노드 리소스를 전용으로 사용할 수 있습니다.

9.3.1. 전용 리소스 정보

가상 머신에 전용 리소스를 사용하면 가상 머신의 워크로드가 다른 프로세스에서 사용하지 않는 CPU에 예약됩니다. 전용 리소스를 사용하면 가상 머신의 성능과 대기 시간 예측 정확도를 개선할 수 있습니다.

9.3.2. 사전 요구 사항

- 노드에 CPU 관리자를 구성해야 합니다. 가상 머신 워크로드를 예약하기 전에 노드에 cpumanager = true 라벨이 있는지 확인하십시오.

9.3.3. 가상 머신 템플릿 전용 리소스 활성화

세부 정보 탭에서 가상 머신 템플릿 전용 리소스를 활성화합니다. Red Hat 템플릿에서 생성된 가상 머신은 전용 리소스로 구성할 수 있습니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.
2. 가상 머신 템플릿을 선택하여 템플릿 세부 정보 페이지를 엽니다.
3. 스케줄링 탭에서 전용 리소스 옆에 연필 아이콘을 클릭합니다.
4. 전용 리소스(보장된 정책)를 사용하여 이 워크로드 예약을 선택합니다.
5. 저장을 클릭합니다.

9.4. 사용자 정의 네임스페이스에 가상 머신 템플릿 배포

Red Hat은 openshift 네임스페이스에 설치된 사전 구성된 가상 머신 템플릿을 제공합니다. ssp-operator 는 기본적으로 가상 머신 템플릿을 openshift 네임스페이스에 배포합니다. openshift 네임스페

이스의 템플릿은 모든 사용자가 공개적으로 사용할 수 있습니다. 이러한 템플릿은 다른 운영 체제의 가상화 → 템플릿 페이지에 나열됩니다.

9.4.1. 템플릿에 대한 사용자 정의 네임스페이스 생성

해당 템플릿에 액세스할 수 있는 권한이 있는 사용자가 사용할 가상 머신 템플릿을 배포하는 데 사용되는 사용자 지정 네임스페이스를 생성할 수 있습니다. 사용자 정의 네임스페이스에 템플릿을 추가하려면 **HyperConverged CR**(사용자 정의 리소스)을 편집하고 **commonTemplatesNamespace** 를 사양에 추가하고 가상 머신 템플릿의 사용자 정의 네임스페이스를 지정합니다. **HyperConverged CR**을 수정한 후 **ssp-operator** 는 사용자 정의 네임스페이스에 템플릿을 채웁니다.

사전 요구 사항

- **OpenShift Container Platform CLI oc**를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 로그인합니다.

절차

- 다음 명령을 사용하여 사용자 정의 네임스페이스를 생성합니다.

```
$ oc create namespace <mycustomnamespace>
```

9.4.2. 사용자 정의 네임스페이스에 템플릿 추가

ssp-operator 는 기본적으로 가상 머신 템플릿을 **openshift** 네임스페이스에 배포합니다. **openshift** 네임스페이스의 템플릿은 모든 사용자가 공개적으로 사용할 수 있습니다. 사용자 정의 네임스페이스가 생성되고 해당 네임스페이스에 템플릿이 추가되면 **openshift** 네임스페이스에서 가상 머신 템플릿을 수정하거나 삭제할 수 있습니다. 사용자 정의 네임스페이스에 템플릿을 추가하려면 **ssp-operator** 가 포함된 **HyperConverged CR**(사용자 정의 리소스)을 편집합니다.

절차

1. **openshift** 네임스페이스에서 사용 가능한 가상 머신 템플릿 목록을 확인합니다.

```
$ oc get templates -n openshift
```

2. 다음 명령을 실행하여 기본 편집기에서 **HyperConverged CR**을 편집합니다.

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

3. 사용자 정의 네임스페이스에서 사용할 수 있는 가상 머신 템플릿 목록을 확인합니다.

```
$ oc get templates -n customnamespace
```

4. **commonTemplatesNamespace** 속성을 추가하고 사용자 정의 네임스페이스를 지정합니다. 예제:

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
spec:
  commonTemplatesNamespace: customnamespace (1)
```

(1) 템플릿을 배포하기 위한 사용자 정의 네임스페이스입니다.

5. 변경 사항을 저장하고 편집기를 종료합니다. **ssp-operator** 는 기본 **openshift** 네임스페이스에 존재하는 가상 머신 템플릿을 사용자 정의 네임스페이스에 추가합니다.

9.4.2.1. 사용자 정의 네임스페이스에서 템플릿 삭제

사용자 정의 네임스페이스에서 가상 머신 템플릿을 삭제하려면 **HyperConverged CR**(사용자 정의 리소스)에서 **commonTemplateNamespace** 특성을 제거하고 해당 사용자 정의 네임스페이스에서 각 템플릿을 삭제합니다.

절차

1. 다음 명령을 실행하여 기본 편집기에서 **HyperConverged CR**을 편집합니다.

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

2. **commonTemplateNamespace** 특성을 제거합니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
```

```
  name: kubevirt-hyperconverged
spec:
  commonTemplatesNamespace: customnamespace (1)
```

(1) 삭제할 commonTemplatesNamespace 속성입니다.

3. 삭제된 사용자 정의 네임스페이스에서 특정 템플릿을 삭제합니다.

```
$ oc delete templates -n customnamespace <template_name>
```

검증

- 템플릿이 사용자 지정 네임스페이스에서 삭제되었는지 확인합니다.

```
$ oc get templates -n customnamespace
```

9.4.2.2. 추가 리소스

- 가상 머신 템플릿 생성

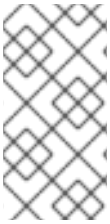
9.5. 가상 머신 템플릿 삭제

웹 콘솔을 사용하여 **Red Hat** 템플릿을 기반으로 사용자 지정된 가상 머신 템플릿을 삭제할 수 있습니다.

Red Hat 템플릿을 삭제할 수 없습니다.

9.5.1. 웹 콘솔에서 가상 머신 템플릿 삭제

가상 머신 템플릿을 삭제하면 클러스터에서 해당 템플릿이 영구적으로 제거됩니다.



참고

사용자 지정된 가상 머신 템플릿을 삭제할 수 있습니다. **Red Hat** 제공 템플릿은 삭제할 수 없습니다.

절차

1. **OpenShift Container Platform** 콘솔의 사이드 메뉴에서 가상화 → 템플릿을 클릭합니다.

2. 템플릿의 옵션 메뉴  를 클릭하고 템플릿 삭제 를 선택합니다.

3. 삭제를 클릭합니다.

10장. 실시간 마이그레이션

10.1. 가상 머신 실시간 마이그레이션

10.1.1. 실시간 마이그레이션 정보

실시간 마이그레이션은 가상 워크로드 또는 액세스를 중단하지 않고 실행 중인 VMI(가상 머신 인스턴스)를 클러스터의 다른 노드로 이동하는 프로세스입니다. VMI에서 LiveMigrate 제거 전략을 사용하는 경우 VMI가 유지보수 모드로 실행되는 노드가 유지보수 모드에 배치될 때 자동으로 마이그레이션됩니다. 마이그레이션할 VMI를 선택하여 실시간 마이그레이션을 수동으로 시작할 수도 있습니다.

다음 조건이 충족되면 실시간 마이그레이션을 사용할 수 있습니다.

- RWX(ReadWriteMany) 액세스 모드를 사용하여 스토리지 공유.
- 충분한 RAM 및 네트워크 대역폭.
- 가상 머신에서 호스트 모델 CPU를 사용하는 경우 노드는 가상 머신의 호스트 모델 CPU를 지원해야 합니다.

기본적으로 실시간 마이그레이션 트래픽은 TLS(Transport Layer Security)를 사용하여 암호화됩니다.

10.1.2. 추가 리소스

- 가상 머신 인스턴스를 다른 노드로 마이그레이션
- 실시간 마이그레이션 제한
- 스토리지 프로파일 사용자 정의

10.2. 실시간 마이그레이션 제한 및 타임아웃

마이그레이션 프로세스에서 클러스터를 전부 사용하지 않도록 실시간 마이그레이션 제한 및 타임아웃

을 적용합니다. HyperConverged CR(사용자 정의 리소스)을 편집하여 이러한 설정을 구성합니다.

10.2.1. 실시간 마이그레이션 제한 및 타임아웃 구성

openshift-cnv 네임스페이스에 있는 HyperConverged CR(사용자 정의 리소스)을 업데이트하여 클러스터의 실시간 마이그레이션 제한 및 타임아웃을 구성합니다.

절차

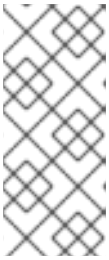
- HyperConverged CR을 편집하고 필요한 실시간 마이그레이션 매개변수를 추가합니다.

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

설정 파일 예

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  liveMigrationConfig: 1
    bandwidthPerMigration: 64Mi
    completionTimeoutPerGiB: 800
    parallelMigrationsPerCluster: 5
    parallelOutboundMigrationsPerNode: 2
    progressTimeout: 150
```

1 이 예에서 spec.liveMigrationConfig 배열에는 각 필드의 기본값이 포함되어 있습니다.



참고

해당 키/값 쌍을 삭제하고 파일을 저장하여 spec.liveMigrationConfig 필드의 기본값을 복원할 수 있습니다. 예를 들어 progressTimeout: <value>를 삭제하여 기본 progressTimeout: 150을 복원합니다.

10.2.2. 클러스터 수준의 실시간 마이그레이션 제한 및 타임아웃

표 10.1. 마이그레이션 매개변수

매개변수	설명	기본
parallelMigrationsPerCluster	클러스터에서 병렬로 실행되고 있는 마이그레이션의 수입니다.	5
parallelOutboundMigrationsPerNode	노드당 최대 아웃바운드 마이그레이션의 수입니다.	2
bandwidthPerMigration	각 마이그레이션의 대역폭 제한(MiB/s)입니다.	0 [1]
completionTimeoutPerGiB	이 시점에 메모리 GiB당 초 단위로 마이그레이션이 완료되지 않으면 마이그레이션이 취소됩니다. 예를 들어, 메모리가 6GiB인 가상 머신 인스턴스는 4800초 내에 마이그레이션이 완료되지 않으면 타임아웃됩니다. **Migration Method**가 **BlockMigration**인 경우 마이그레이션 디스크의 크기가 계산에 포함됩니다.	800
progressTimeout	이 시간(초) 내에 메모리 복사를 진행하지 못하면 마이그레이션이 취소됩니다.	150

1. **기본값은 무제한 입니다.**

10.3. 가상 머신 인스턴스를 다른 노드로 마이그레이션

웹 콘솔 또는 CLI를 사용하여 다른 노드로의 가상 머신 인스턴스 실시간 마이그레이션을 수동으로 시작합니다.

참고

가상 머신에서 호스트 모델 CPU를 사용하는 경우 호스트 CPU 모델을 지원하는 노드 간에만 해당 가상 머신의 실시간 마이그레이션을 수행할 수 있습니다.

10.3.1. 웹 콘솔에서 가상 머신 인스턴스 실시간 마이그레이션 시작

실행 중인 가상 머신 인스턴스를 클러스터의 다른 노드로 마이그레이션합니다.

참고

마이그레이션 작업은 모든 사용자에게 표시되지만 관리자만 가상 머신 마이그레이션을 시작할 수 있습니다.

절차

1. **OpenShift Container Platform** 콘솔의 사이드 메뉴에서 가상화 → **VirtualMachines** 를 클릭합니다.

2. 이 페이지에서 마이그레이션을 시작하면 동일한 페이지에서 여러 가상 머신에 대한 작업을 더 쉽게 수행할 수 있고 **VirtualMachine** 세부 정보 페이지에서 마이그레이션을 시작하면 선택한 가상 머신에 대한 포괄적인 세부 정보를 볼 수 있습니다.

   - 가상 머신 옆에 있는 옵션 메뉴

     를 클릭하고 마이그레이션 을 선택합니다.

   - 가상 머신 이름을 클릭하여 **VirtualMachine** 세부 정보 페이지를 열고 작업 → 마이그레이션 을 클릭합니다.

3. 마이그레이션을 클릭하여 가상 머신을 다른 노드로 마이그레이션합니다.

10.3.2. CLI에서 가상 머신 인스턴스 실시간 마이그레이션 시작

클러스터에서 **VirtualMachineInstanceMigration** 오브젝트를 생성하고 가상 머신 인스턴스의 이름을 참조하여 실행 중인 가상 머신 인스턴스의 실시간 마이그레이션을 시작합니다.

절차

1. 마이그레이션할 가상 머신 인스턴스에 대한 **VirtualMachineInstanceMigration** 구성 파일을 생성합니다. 예를 들면 **vmi-migrate.yaml**은 다음과 같습니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstanceMigration
metadata:
```

```
  name: migration-job
spec:
  vmiName: vmi-fedora
```

2. 다음 명령을 실행하여 클러스터에 오브젝트를 생성합니다.

```
$ oc create -f vmi-migrate.yaml
```

VirtualMachineInstanceMigration 오브젝트는 가상 머신 인스턴스의 실시간 마이그레이션을 트리거합니다. 이 오브젝트는 수동으로 삭제하지 않는 한 가상 머신 인스턴스가 실행되는 동안 클러스터에 존재합니다.

추가 리소스:

- 가상 머신 인스턴스의 실시간 마이그레이션 모니터링
- 가상 머신 인스턴스의 실시간 마이그레이션 취소

10.4. 전용 추가 네트워크를 통한 가상 머신 마이그레이션

실시간 마이그레이션을 위해 전용 **Multus** 네트워크를 구성할 수 있습니다. 전용 네트워크는 실시간 마이그레이션 중에 테넌트 워크로드에 대한 네트워크 포화도의 영향을 최소화합니다.

10.4.1. 가상 머신 실시간 마이그레이션을 위한 전용 보조 네트워크 구성

실시간 마이그레이션을 위한 전용 보조 네트워크를 구성하려면 먼저 **CLI**를 사용하여 네임스페이스에 대한 브리지 네트워크 연결 정의를 생성해야 합니다. 그런 다음 **NetworkAttachmentDefinition** 오브젝트의 이름을 **HyperConverged CR**(사용자 정의 리소스)에 추가합니다.

사전 요구 사항

- **OpenShift CLI(oc)**를 설치합니다.
- **cluster-admin** 역할의 사용자로 클러스터에 로그인했습니다.
- **Multus CNI(Container Network Interface)** 플러그인이 클러스터에 설치되어 있습니다.

- 클러스터의 모든 노드에는 NIC(네트워크 인터페이스 카드)가 두 개 이상 있으며 실시간 마이그레이션에 사용할 NIC가 동일한 VLAN에 연결됩니다.

- VM(가상 머신)이 LiveMigrate 제거 전략을 사용하여 실행됩니다.

절차

1. NetworkAttachmentDefinition 매니페스트를 생성합니다.

설정 파일 예

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
 name: my-secondary-network 1
 namespace: openshift-cnv
spec:
 config: '{
  "cniVersion": "0.3.1",
  "name": "migration-bridge",
  "type": "macvlan",
  "master": "eth1", 2
  "mode": "bridge",
  "ipam": {
   "type": "whereabouts", 3
   "range": "10.200.5.0/24" 4
  }
 }'
```

1 NetworkAttachmentDefinition 오브젝트의 이름입니다.

2 실시간 마이그레이션에 사용할 NIC의 이름입니다.

3 이 네트워크 연결 정의에 대한 네트워크를 제공하는 CNI 플러그인의 이름입니다.

4

보조 네트워크의 IP 주소 범위입니다. 이 범위는 기본 네트워크의 IP 주소와 겹치지 않아야 합니다.

2. 다음 명령을 실행하여 기본 편집기에서 HyperConverged CR을 엽니다.

```
oc edit hyperconverged kubevirt-hyperconverged -n openshift-cnv
```

3. NetworkAttachmentDefinition 오브젝트의 이름을 HyperConverged CR의 spec.liveMigrationConfig 스탠자에 추가합니다. 예를 들면 다음과 같습니다.

설정 파일 예

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
spec:
  liveMigrationConfig:
    completionTimeoutPerGiB: 800
    network: my-secondary-network 1
    parallelMigrationsPerCluster: 5
    parallelOutboundMigrationsPerNode: 2
    progressTimeout: 150
...
```

1

실시간 마이그레이션에 사용할 Multus NetworkAttachmentDefinition 오브젝트의 이름입니다.

4. 변경 사항을 저장하고 편집기를 종료합니다. virt-handler 포드가 다시 시작되고 보조 네트워크에 연결합니다.

검증

- 가상 머신이 실행되는 노드가 유지보수 모드에 배치되면 VM이 클러스터의 다른 노드로 자동

으로 마이그레이션됩니다. VMI(가상 머신 인스턴스) 메타데이터에서 대상 IP 주소를 확인하여 보조 네트워크가 아니라 기본 Pod 네트워크를 통해 마이그레이션이 발생했는지 확인할 수 있습니다.

```
oc get vmi <vmi_name> -o jsonpath='{.status.migrationState.targetNodeAddress}'
```

10.4.2. 추가 리소스

- 실시간 마이그레이션 제한 및 타임아웃

10.5. 가상 머신 인스턴스의 실시간 마이그레이션 모니터링

웹 콘솔 또는 CLI에서 가상 머신 인스턴스의 실시간 마이그레이션 진행 상태를 모니터링할 수 있습니다.

10.5.1. 웹 콘솔에서 가상 머신 인스턴스 실시간 마이그레이션 모니터링

마이그레이션 기간 동안 가상 머신의 상태는 마이그레이션 중입니다. 이 상태는 VirtualMachines 페이지 또는 마이그레이션 가상 머신의 VirtualMachine 세부 정보 페이지에 표시됩니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.

10.5.2. CLI에서 가상 머신 인스턴스 실시간 마이그레이션 모니터링

가상 머신 마이그레이션 상태는 VirtualMachineInstance 구성의 Status 구성 요소에 저장됩니다.

절차

- 마이그레이션 중인 가상 머신 인스턴스에 oc describe 명령을 사용합니다.

```
$ oc describe vmi vmi-fedora
```

출력 예

```
...
Status:
  Conditions:
    Last Probe Time:       <nil>
    Last Transition Time:  <nil>
    Status:                True
    Type:                  LiveMigratable
  Migration Method:  LiveMigration
  Migration State:
    Completed:                true
    End Timestamp:            2018-12-24T06:19:42Z
    Migration UID:            d78c8962-0743-11e9-a540-fa163e0c69f1
    Source Node:              node2.example.com
    Start Timestamp:          2018-12-24T06:19:35Z
    Target Node:              node1.example.com
    Target Node Address:      10.9.0.18:43891
    Target Node Domain Detected:  true
```

10.6. 가상 머신 인스턴스의 실시간 마이그레이션 취소

가상 머신 인스턴스가 원래 노드에 남아 있도록 실시간 마이그레이션을 취소합니다.

웹 콘솔 또는 CLI에서 실시간 마이그레이션을 취소할 수 있습니다.

10.6.1. 웹 콘솔에서 가상 머신 인스턴스의 실시간 마이그레이션 취소

웹 콘솔에서 가상 머신 인스턴스의 실시간 마이그레이션을 취소할 수 있습니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.
2. 가상 머신 옆에 있는 옵션 메뉴  를 클릭하고 마이그레이션 취소 를 선택합니다.

10.6.2. CLI에서 가상 머신 인스턴스 실시간 마이그레이션 취소

마이그레이션과 연결된 **VirtualMachineInstanceMigration** 오브젝트를 삭제하여 가상 머신 인스턴스의 실시간 마이그레이션을 취소합니다.

절차

- 이 예제에서 실시간 마이그레이션 작업인 **migration-job**을 트리거한 **VirtualMachineInstanceMigration** 오브젝트를 삭제합니다.

```
$ oc delete vmim migration-job
```

10.7. 가상 머신 제거 전략 구성

LiveMigrate 제거 전략을 사용하면 노드가 유지보수 또는 드레인 모드에 배치되는 경우 가상 머신 인스턴스가 중단되지 않습니다. 이 제거 전략이 포함된 가상 머신 인스턴스는 다른 노드로 실시간 마이그레이션됩니다.

10.7.1. LiveMigration 제거 전략을 사용하여 사용자 정의 가상 머신 구성

사용자 정의 가상 머신에는 **LiveMigration** 제거 전략만 구성하면 됩니다. 공통 템플릿에는 이 제거 전략이 기본적으로 구성되어 있습니다.

절차

1. 가상 머신 구성 파일의 **spec.template.spec** 섹션에 **evictionStrategy: LiveMigrate** 옵션을 추가합니다. 이 예제에서는 **oc edit**를 사용하여 **VirtualMachine** 구성 파일의 관련 스니펫을 업데이트합니다.

```
$ oc edit vm <custom-vm> -n <my-namespace>
```

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: custom-vm
spec:
  template:
    spec:
      evictionStrategy: LiveMigrate
...
```

2. 가상 머신을 재시작하여 업데이트를 적용합니다.

```
$ virtctl restart <custom-vm> -n <my-namespace>
```

11장. 노드 유지보수

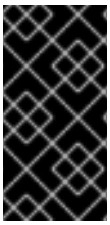
11.1. 노드 유지보수 정보

11.1.1. 노드 유지보수 모드 정보

노드는 **oc adm** 유틸리티 또는 **NodeMaintenance** 사용자 정의 리소스 (CR)를 사용하여 유지보수 모드로 전환할 수 있습니다.

노드를 유지보수 모드에 배치하면 노드가 스케줄링할 수 없는 것으로 표시되고 모든 가상 머신과 **Pod**가 드레인됩니다. **LiveMigrate** 제거 전략이 있는 가상 머신 인스턴스는 서비스 손실 없이 다른 노드로 실시간 마이그레이션됩니다. 이 제거 전략은 공통 템플릿으로 생성한 가상 머신에는 기본적으로 구성되지만 사용자 정의 가상 머신은 수동으로 구성해야 합니다.

제거 전략이 없는 가상 머신 인스턴스가 종료됩니다. **Running** 또는 **RerunOnFailure**의 **RunStrategy**가 있는 가상 머신은 다른 노드에서 다시 생성됩니다. **Manual**의 **RunStrategy**가 있는 가상 머신은 자동으로 다시 시작되지 않습니다.



중요

가상 머신에 실시간 마이그레이션할 공유 **ReadWriteMany (RWX)** 액세스 모드의 **PVC**(영구 볼륨 클레임)가 있어야 합니다.

OpenShift Virtualization의 일부로 설치하면 **Node Maintenance Operator**는 새로운 또는 삭제된 **NodeMaintenance CR**을 확인합니다. 새 **NodeMaintenance CR**이 감지되면 새 워크로드가 예약되지 않고 나머지 클러스터에서 노드가 차단됩니다. 제거할 수 있는 모든 **Pod**는 노드에서 제거됩니다. **NodeMaintenance CR**이 삭제되면 **CR**에서 참조되는 노드를 새 워크로드에 사용할 수 있습니다.

참고

노드 유지관리 작업에 **NodeMaintenance CR**을 사용하면 표준 **OpenShift Container Platform** 사용자 정의 리소스 처리를 사용하여 **oc adm cordon** 및 **oc adm drain** 명령과 동일한 결과를 얻을 수 있습니다.

11.1.2. 베어 메탈 노드 유지관리

베어 메탈 인프라에 **OpenShift Container Platform**을 배포할 때 클라우드 인프라에 배포하는 것과 비교하여 고려해야 할 추가 고려 사항이 있습니다. 클러스터 노드가 사용 후 삭제로 간주되는 클라우드 환경

에서와 달리 베어 메탈 노드를 다시 프로비저닝하려면 유지관리 작업에 더 많은 시간과 노력이 필요합니다.

예를 들어 치명적인 커널 오류가 발생하거나 **NIC** 카드 하드웨어 장애가 발생하는 것과 같이 베어메탈 노드에 장애가 발생한 경우 문제가 발생한 노드가 복구 되거나 교체되는 동안 장애가 발생한 노드의 워크로드를 클러스터의 다른 곳에서 다시 시작해야 합니다. 클러스터 관리자는 노드 유지관리 모드를 통해 노드의 전원을 정상적으로 끄고 워크로드를 클러스터의 다른 부분으로 이동하여 워크로드가 중단되지 않도록 할 수 있습니다. 유지보수 관리 중에 자세한 진행 상황 및 노드 상태 세부 정보가 제공됩니다.

추가 리소스:

- 가상 머신 **RunStrategies** 정보
- 가상 머신 실시간 마이그레이션
- 가상 머신 제거 전략 구성

11.2. 노드를 유지보수 모드로 설정

웹 콘솔, **CLI** 또는 **NodeMaintenance** 사용자 정의 리소스에서 노드를 유지관리 모드에 배치합니다.

11.2.1. 웹 콘솔에서 노드를 유지보수 모드로 설정

컴퓨팅 → 노드 목록의 각 노드에 있는 옵션 메뉴

를 사용하거나 노드 세부 정보 화면의 작업 컨트롤을 사용하여 노드를 유지보수 모드로 설정합니다.

절차

1. **OpenShift Container Platform** 콘솔에서 컴퓨팅 → 노드를 클릭합니다.
2. 이 화면에서 노드를 유지보수 모드로 설정하면 한 화면에서 여러 노드에 대한 작업을 더 쉽게 수행할 수 있습니다. 노드 세부 정보 화면에서 노드를 유지보수 모드로 설정하면 선택한 노드에 대한 포괄적인 세부 정보를 볼 수 있습니다.
   -

노드 끝에 있는 옵션 메뉴

를 클릭하고 유지보수 시작을 선택합니다.

- 노드 이름을 클릭하여 노드 세부 정보 화면을 열고 작업 → 유지보수 시작을 클릭합니다.

3. 확인 창에서 유지보수 시작을 클릭합니다.

노드는 **LiveMigration** 제거 전략이 있는 가상 머신 인스턴스를 실시간 마이그레이션하고 더 이상 노드를 스케줄링할 수 없습니다. 노드의 기타 모든 **Pod** 및 가상 머신이 삭제되고 다른 노드에서 다시 생성됩니다.

11.2.2. CLI에서 노드를 유지보수 모드로 설정

oc adm drain 명령을 사용하여 노드에서 **Pod**를 제거하거나 삭제하여 노드를 유지보수 모드로 설정합니다.

절차

1. 노드를 예약 불가능으로 표시합니다. 노드 상태가 **NotReady, SchedulingDisabled**로 변경됩니다.

```
$ oc adm cordon <node1>
```

2. 유지보수를 위해 노드를 드레이닝합니다. 노드는 **LiveMigratable** 조건이 **True**로 설정되고 **spec:evictionStrategy** 필드가 **LiveMigrate**로 설정된 가상 머신 인스턴스를 실시간으로 마이그레이션합니다. 노드의 기타 모든 **Pod** 및 가상 머신이 삭제되고 다른 노드에서 다시 생성됩니다.

```
$ oc adm drain <node1> --delete-emptydir-data --ignore-daemonsets=true --force
```

- **--delete-emptydir-data** 플래그는 **emptyDir** 볼륨을 사용하는 노드의 모든 가상 머신 인스턴스를 제거합니다. 이러한 볼륨의 데이터는 임시이며 종료 후 삭제될 수 있습니다.

- **--ignore-daemonsets=true** 플래그를 사용하면 데몬 세트가 무시되고 **Pod** 제거를 성공적으로 수행할 수 있습니다.

- --force 플래그는 복제본 세트 또는 데몬 세트 컨트롤러에서 관리하지 않는 Pod를 삭제합니다.

11.2.3. NodeMaintenance 사용자 정의 리소스를 사용하여 노드를 유지관리 모드로 설정

NodeMaintenance CR(사용자 정의 리소스)을 사용하여 노드를 유지관리 모드에 배치할 수 있습니다. NodeMaintenance CR을 적용하면 허용되는 모든 Pod가 제거되고 노드가 종료됩니다. 제거된 Pod는 클러스터의 다른 노드로 이동하기 위해 대기열에 있습니다.

사전 요구 사항

- OpenShift Container Platform CLI oc를 설치합니다.
- cluster-admin 권한이 있는 사용자로 클러스터에 로그인합니다.

절차

1. 다음 노드 유지관리 CR을 생성하고 파일을 nodemaintenance-cr.yaml로 저장합니다.

```
apiVersion: nodemaintenance.kubevirt.io/v1beta1
kind: NodeMaintenance
metadata:
  name: maintenance-example 1
spec:
  nodeName: node-1.example.com 2
  reason: "Node maintenance" 3
```

1. 노드 유지관리 CR 이름
2. 유지관리 모드에 배치할 노드의 이름
3. 유지 관리 이유에 대한 일반 텍스트 설명

2. 다음 명령을 실행하여 노드 유지관리 일정을 적용합니다.

```
$ oc apply -f nodemaintenance-cr.yaml
```

3. 다음 명령을 실행하여 유지관리 작업의 진행 상황을 확인하고 <node-name>을 노드 이름으로 교체합니다.

```
$ oc describe node <node-name>
```

출력 예

```
Events:
  Type    Reason                  Age                From     Message
  ----    ------                  ----               ----     -------
  Normal  NodeNotSchedulable      61m                kubelet  Node node-
1.example.com status is now: NodeNotSchedulable
```

11.2.3.1. 현재 NodeMaintenance CR 작업의 상태 확인

현재 NodeMaintenance CR 작업의 상태를 확인할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform CLI oc를 설치합니다.
- cluster-admin 권한이 있는 사용자로 로그인합니다.

절차

- 다음 명령을 실행하여 현재 노드 유지관리 작업의 상태를 확인합니다.

```
$ oc get NodeMaintenance -o yaml
```

출력 예

```
apiVersion: v1
items:
- apiVersion: nodemaintenance.kubevirt.io/v1beta1
  kind: NodeMaintenance
  metadata:
...
  spec:
    nodeName: node-1.example.com
    reason: Node maintenance
  status:
    evictionPods: 3 (1)
    pendingPods:
    - pod-example-workload-0
    - httpd
    - httpd-manual
    phase: Running
    lastError: "Last failure message" (2)
    totalpods: 5
...
```

(1) **evictionPods**는 제거로 예약된 **Pod** 수입니다.

(2) **lastError**는 최신 제거 오류(있는 경우)를 기록합니다.

추가 리소스:

- 유지보수 모드에서 노드 재시작
- 가상 머신 실시간 마이그레이션
- 가상 머신 제거 전략 구성

11.3. 유지관리 모드에서 노드 재시작

노드를 재시작하면 노드가 유지관리 모드에서 해제되어 노드를 다시 스케줄링할 수 있습니다.

웹 콘솔, CLI 또는 NodeMaintenance 사용자 정의 리소스를 삭제하여 유지보수 모드에서 노드를 재시작합니다.

11.3.1. 웹 콘솔에서 유지보수 모드로 노드 재시작

컴퓨팅 → 노드 목록의 각 노드에 있는 옵션 메뉴

를 사용하거나 노드 세부 정보 화면의 작업 컨트롤을 사용하여 노드를 유지보수 모드로 재시작합니다.

절차

1. OpenShift Container Platform 콘솔에서 컴퓨팅 → 노드를 클릭합니다.

2. 이 화면에서 노드를 재시작하면 한 화면에서 여러 노드에 대한 작업을 더 쉽게 수행할 수 있습니다. 노드 세부 정보 화면에서 노드를 재시작하면 선택한 노드에 대한 포괄적인 세부 정보를 볼 수 있습니다.

   - 노드 끝에 있는 옵션 메뉴

     를 클릭하고 유지보수 중지를 선택합니다.

   - 노드 이름을 클릭하여 노드 세부 정보 화면을 열고 작업 → 유지보수 중지를 클릭합니다.

3. 확인 창에서 유지보수 중지를 클릭합니다.

노드는 스케줄링할 수 있지만 유지보수 전에 노드에서 실행 중이던 가상 머신 인스턴스는 이 노드에 자동으로 마이그레이션되지 않습니다.

11.3.2. CLI에서 유지보수 모드로 노드 재시작

노드를 유지보수 모드에서 다시 스케줄링할 수 있도록 설정하여 노드를 다시 시작합니다.

절차

- 노드를 예약 가능으로 표시합니다. 그런 다음 노드에서 새 워크로드 예약을 다시 시작할 수 있습니다.

```
$ oc adm uncordon <node1>
```

11.3.3. NodeMaintenance CR을 사용하여 시작된 유지관리 모드에서 노드 재시작

NodeMaintenance CR을 삭제하여 노드를 재시작할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform CLI oc를 설치합니다.
- cluster-admin 권한이 있는 사용자로 클러스터에 로그인합니다.

절차

- 노드 유지관리 작업이 완료되면 활성 NodeMaintenance CR을 삭제합니다.

```
$ oc delete -f nodemaintenance-cr.yaml
```

출력 예

```
nodemaintenance.nodemaintenance.kubevirt.io "maintenance-example" deleted
```

11.4. TLS 인증서 자동 갱신

OpenShift Virtualization 구성 요소에 대한 모든 TLS 인증서는 자동으로 갱신되고 순환됩니다. 수동으로 새로 고치지 않아도 됩니다.

11.4.1. TLS 인증서 자동 갱신 예약

TLS 인증서는 다음 일정에 따라 자동으로 삭제되고 교체됩니다.

- KubeVirt 인증서는 매일 갱신됩니다.
- CDI(Containerized Data Importer) 컨트롤러 인증서는 15일마다 갱신됩니다.
- MAC 풀 인증서는 매년 갱신됩니다.

자동 TLS 인증서 순환이 수행되어도 작업이 중단되지 않습니다. 예를 들면 다음 작업이 중단되지 않고 계속 수행됩니다.

- 마이그레이션
- 이미지 업로드
- VNC 및 콘솔 연결

11.5. 더 이상 사용되지 않는 CPU 모델에 대한 노드 라벨링 관리

노드에서 VM CPU 모델 및 정책을 지원하는 경우 노드에서 VM(가상 머신)을 예약할 수 있습니다.

11.5.1. 더 이상 사용되지 않는 CPU 모델에 대한 노드 레이블 설정 정보

OpenShift Virtualization Operator는 사용되지 않는 CPU 모델의 미리 정의된 목록을 사용하여 노드가 스케줄링된 VM에 유효한 CPU 모델만 지원하도록 합니다.

기본적으로 다음 CPU 모델은 노드에 대해 생성된 레이블 목록에서 제거됩니다.

예 11.1. 더 이상 사용되지 않는 CPU 모델

```
"486"
```

```
Conroe
athlon
core2duo
coreduo
kvm32
kvm64
n270
pentium
pentium2
pentium3
pentiumpro
phenom
qemu32
qemu64
```

이 사전 정의된 목록은 **HyperConverged CR**에 표시되지 않습니다. 이 목록에서 **CPU** 모델을 *제거할* 수는 없지만 **HyperConverged CR**의 **spec.obsolicallyCPUs.cpuModels** 필드를 편집하여 목록에 추가할 수 있습니다.

11.5.2. CPU 기능의 노드 레이블링 정보

반복 프로세스를 거치는 동안 최소 **CPU** 모델의 기본 **CPU** 기능이 노드에 대해 생성되는 라벨 목록에서 제거됩니다.

예를 들면 다음과 같습니다.

- 환경에 두 가지 **CPU** 모델, **Penryn** 및 **Haswell**이 지원될 수 있습니다.
- **Penryn**이 **minCPU**의 **CPU** 모델로 지정되면 **Penryn**의 각 기본 **CPU** 기능은 **Haswell**에서 지원하는 각 **CPU** 기능 목록과 비교됩니다.

예 11.2. Penryn에서 지원하는 **CPU** 기능

```
apic
clflush
cmov
cx16
cx8
de
fpu
fxsr
lahf_lm
lm
mca
mce
```

```
mmx
msr
mtrr
nx
pae
pat
pge
pni
pse
pse36
sep
sse
sse2
sse4.1
ssse3
syscall
tsc
```

예 11.3. Haswell에서 지원하는 CPU 기능

```
aes
apic
avx
avx2
bmi1
bmi2
clflush
cmov
cx16
cx8
de
erms
fma
fpu
fsgsbase
fxsr
hle
invpcid
lahf_lm
lm
mca
mce
mmx
movbe
msr
mtrr
nx
pae
pat
pcid
pclmuldq
pge
pni
popcnt
```

```
pse
pse36
rdtscp
rtm
sep
smep
sse
sse2
sse4.1
sse4.2
ssse3
syscall
tsc
tsc-deadline
x2apic
xsave
```

- **Penryn** 및 **Haswell**이 특정 **CPU** 기능을 모두 지원하면 해당 기능에 대한 레이블이 생성되지 않습니다. 라벨은 **Haswell**에서만 지원하고 **Penryn**에서는 지원하지 않는 **CPU** 기능에 대해 생성됩니다.

예 11.4. CPU 기능 반복 후 생성된 노드 레이블

```
aes
avx
avx2
bmi1
bmi2
erms
fma
fsgsbase
hle
invpcid
movbe
pcid
pclmuldq
popcnt
rdtscp
rtm
sse4.2
tsc-deadline
x2apic
xsave
```

11.5.3. 더 이상 사용되지 않는 CPU 모델 구성

HyperConverged CR(사용자 정의 리소스)을 편집하여 더 이상 사용되지 않는 **CPU** 모델 목록을 구성할 수 있습니다.

절차

- HyperConverged 사용자 지정 리소스를 편집하여 obsoleteCPUs 배열에 더 이상 사용되지 않는 CPU 모델을 지정합니다. 예를 들면 다음과 같습니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  obsoleteCPUs:
    cpuModels: 1
      - "<obsolete_cpu_1>"
      - "<obsolete_cpu_2>"
    minCPUModel: "<minimum_cpu_model>" 2
```

1 cpuModels 배열의 예제 값을 더 이상 사용되지 않는 CPU 모델로 교체합니다. 지정한 모든 값은 더 이상 사용되지 않는 CPU 모델에 사전 정의된 목록에 추가됩니다. 사전 정의된 목록은 CR에 표시되지 않습니다.

2 이 값을 기본 CPU 기능에 사용할 최소 CPU 모델로 바꿉니다. 값을 지정하지 않으면 기본적으로 Penryn이 사용됩니다.

11.6. 노드 조정 방지

node-labeller가 노드를 조정하지 못하도록 하려면 skip-node 주석을 사용합니다.

11.6.1. skip-node 주석 사용

node-labeller에서 노드를 건너뛰려면 oc CLI를 사용하여 해당 노드에 주석을 답니다.

사전 요구 사항

- OpenShift CLI(oc)가 설치되어 있습니다.

절차

- 다음 명령을 실행하여 건너뛰려는 노드에 주석을 답니다.

```
$ oc annotate node <node_name> node-labeller.kubevirt.io/skip-node=true 1
```

1 <node_name>을 건너뛸 관련 노드의 이름으로 바꿉니다.

노드 주석을 제거하거나 **false**로 설정한 후 다음 주기에서 조정이 재개됩니다.

11.6.2. 추가 리소스

- 더 이상 사용되지 않는 CPU 모델에 대한 노드 라벨링 관리

12장. 노드 네트워킹

12.1. 노드 네트워크 상태 관찰

노드 네트워크 상태는 클러스터의 모든 노드에 대한 네트워크 구성입니다.

12.1.1. nmstate 정보

OpenShift Virtualization에서는 nmstate를 사용하여 노드 네트워크의 상태를 보고하고 구성합니다. 이를 통해 단일 구성 매니페스트를 클러스터에 적용하여(예: 모든 노드에서 Linux 브리지 생성) 네트워크 정책 구성을 수정할 수 있습니다.

노드 네트워킹은 다음 오브젝트에서 모니터링하고 업데이트합니다.

NodeNetworkState

해당 노드의 네트워크 상태를 보고합니다.

NodeNetworkConfigurationPolicy

노드에서 요청된 네트워크 구성을 설명합니다. NodeNetworkConfigurationPolicy 매니페스트를 클러스터에 적용하는 방식으로 인터페이스 추가 및 제거를 포함하여 노드 네트워크 구성을 업데이트합니다.

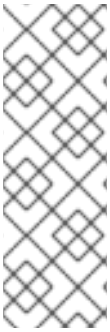
NodeNetworkConfigurationEnactment

각 노드에 적용된 네트워크 정책을 보고합니다.

OpenShift Virtualization에서는 다음과 같은 nmstate 인터페이스 유형을 사용할 수 있습니다.

- Linux 브리지
- VLAN
- 본딩

- 이더넷



참고

OpenShift Container Platform 클러스터에서 **OVN-Kubernetes**를 기본 **CNI(Container Network Interface)** 공급자로 사용하는 경우, **OVN-Kubernetes**의 호스트 네트워크 토폴로지 변경으로 인해 호스트의 기본 인터페이스에 **Linux** 브리지 또는 본딩을 연결할 수 없습니다. 해결 방법으로 호스트에 연결된 보조 네트워크 인터페이스를 사용하거나 **OpenShift SDN** 기본 **CNI** 공급자로 전환할 수 있습니다.

12.1.2. 노드의 네트워크 상태 보기

NodeNetworkState 오브젝트는 클러스터의 모든 노드에 존재합니다. 이 오브젝트는 주기적으로 업데이트되며 해당 노드의 네트워크 상태를 캡처합니다.

절차

1. 클러스터의 모든 **NodeNetworkState** 오브젝트를 나열합니다.

```
$ oc get nns
```

2. **NodeNetworkState** 오브젝트를 검사하여 해당 노드의 네트워크를 확인합니다. 이 예제의 출력은 명확성을 위해 수정되었습니다.

```
$ oc get nns node01 -o yaml
```

출력 예

```
apiVersion: nmstate.io/v1
kind: NodeNetworkState
metadata:
  name: node01 (1)
status:
  currentState: (2)
    dns-resolver:
...
    interfaces:
...
    route-rules:
...
```

```
    routes:
...
  lastSuccessfulUpdateTime: "2020-01-31T12:14:00Z" 3
```

1

NodeNetworkState 오브젝트의 이름은 노드에서 가져옵니다.

2

currentState에는 DNS, 인터페이스, 경로를 포함하여 노드에 대한 전체 네트워크 구성이 포함됩니다.

3

마지막으로 성공한 업데이트의 타임 스탬프 노드에 연결할 수 있는 동안 주기적으로 업데이트되고 보고서의 최신 상태를 평가하는 데 사용됩니다.

12.2. 노드 네트워크 구성 업데이트

NodeNetworkConfigurationPolicy 매니페스트를 클러스터에 적용하여 노드 네트워크 구성을 업데이트(예: 노드에서 인터페이스 추가 또는 제거)할 수 있습니다.

12.2.1. nmstate 정보

OpenShift Virtualization에서는 nmstate를 사용하여 노드 네트워크의 상태를 보고하고 구성합니다. 이를 통해 단일 구성 매니페스트를 클러스터에 적용하여(예: 모든 노드에서 Linux 브리지 생성) 네트워크 정책 구성을 수정할 수 있습니다.

노드 네트워킹은 다음 오브젝트에서 모니터링하고 업데이트합니다.

NodeNetworkState

해당 노드의 네트워크 상태를 보고합니다.

NodeNetworkConfigurationPolicy

노드에서 요청된 네트워크 구성을 설명합니다. NodeNetworkConfigurationPolicy 매니페스트를 클러스터에 적용하는 방식으로 인터페이스 추가 및 제거를 포함하여 노드 네트워크 구성을 업데이

트합니다.

NodeNetworkConfigurationEnactment

각 노드에 적용된 네트워크 정책을 보고합니다.

OpenShift Virtualization에서는 다음과 같은 **nmstate** 인터페이스 유형을 사용할 수 있습니다.

- **Linux** 브리지
- **VLAN**
- 본딩
- 이더넷

참고

OpenShift Container Platform 클러스터에서 **OVN-Kubernetes**를 기본 **CNI(Container Network Interface)** 공급자로 사용하는 경우, **OVN-Kubernetes**의 호스트 네트워크 토폴로지 변경으로 인해 호스트의 기본 인터페이스에 **Linux** 브리지 또는 본딩을 연결할 수 없습니다. 해결 방법으로 호스트에 연결된 보조 네트워크 인터페이스를 사용하거나 **OpenShift SDN** 기본 **CNI** 공급자로 전환할 수 있습니다.

12.2.2. 노드에서 인터페이스 만들기

NodeNetworkConfigurationPolicy 매니페스트를 클러스터에 적용하여 클러스터의 노드에서 인터페이스를 만듭니다. 매니페스트는 요청된 인터페이스 구성을 자세히 설명합니다.

기본적으로 매니페스트는 클러스터의 모든 노드에 적용됩니다. 특정 노드에 인터페이스를 추가하려면 **spec: nodeSelector** 매개변수와 노드 선택기에 적합한 **<key>:<value>**를 추가합니다.

여러 **nmstate** 지원 노드를 동시에 구성할 수 있습니다. 이 구성은 병렬로 노드의 **50%**에 적용됩니다. 이 전략을 사용하면 네트워크 연결이 실패하는 경우 전체 클러스터를 사용할 수 없게 됩니다. 정책 구성을 클러스터의 특정 부분에 병렬로 적용하려면 **maxUnavailable** 필드를 사용합니다.

절차

1. NodeNetworkConfigurationPolicy 매니페스트를 생성합니다. 다음 예제는 모든 작업자 노드에서 Linux 브리지를 구성하고 DNS 확인자를 구성합니다.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
 name: br1-eth1-policy 1
spec:
 nodeSelector: 2
  node-role.kubernetes.io/worker: "" 3
 maxUnavailable: 3 4
 desiredState:
  interfaces:
   - name: br1
     description: Linux bridge with eth1 as a port 5
     type: linux-bridge
     state: up
     ipv4:
       dhcp: true
       enabled: true
       auto-dns: false
     bridge:
       options:
         stp:
           enabled: false
       port:
         - name: eth1
  dns-resolver: 6
   config:
     search:
     - example.com
     - example.org
     server:
     - 8.8.8.8
```

1 정책 이름입니다.

2 선택 사항: nodeSelector 매개변수를 포함하지 않으면 정책이 클러스터의 모든 노드에 적용됩니다.

3 이 예제에서는 node-role.kubernetes.io/worker: "" 노드 선택기를 사용하여 클러스터의 모든 작업자 노드를 선택합니다.

4

선택 사항: 정책 구성을 동시에 적용할 수 있는 최대 **nmstate** 지원 노드 수를 지정합니다. 이 매개 변수는 백분율 값(문자열), 예를 들어 **"10%"** 또는 **3** 과 같은 절대 값(number)으로 설정할 수 있습니다.

5

선택 사항: 사람이 읽을 수 있는 인터페이스 설명입니다.

6

선택 사항: **DNS** 서버의 검색 및 서버 설정을 지정합니다.

2. 노드 네트워크 정책을 생성합니다.

```
$ oc apply -f br1-eth1-policy.yaml 1
```

1

노드 네트워크 구성 정책 매니페스트의 파일 이름입니다.

추가 리소스

- 다양한 인터페이스에 대한 예제 정책 구성
- 동일한 정책에서 여러 인터페이스를 만드는 예제
- 정책의 다양한 IP 관리 방법 예제

12.2.3. 노드에 노드 네트워크 정책 업데이트 확인

NodeNetworkConfigurationPolicy 매니페스트는 클러스터의 노드에 대해 요청된 네트워크 구성을 설명합니다. 노드 네트워크 정책에는 요청된 네트워크 구성과 클러스터 전체에 대한 정책 실행 상태가 포함됩니다.

노드 네트워크 정책을 적용하면 클러스터의 모든 노드에 대해 **NodeNetworkConfigurationEnactment** 오브젝트가 생성됩니다. 노드 네트워크 구성 시행은 해당 노드

에서 정책의 실행 상태를 나타내는 읽기 전용 오브젝트입니다. 정책이 노드에 적용되지 않으면 문제 해결을 위해 해당 노드에 대한 시행에 역추적이 포함됩니다.

절차

1. 정책이 클러스터에 적용되었는지 확인하려면 정책과 해당 상태를 나열합니다.

```
$ oc get nncp
```

2. 선택 사항: 정책을 구성하는 데 예상보다 오래 걸리는 경우 특정 정책의 요청된 상태 및 상태 조건을 검사할 수 있습니다.

```
$ oc get nncp <policy> -o yaml
```

3. 선택 사항: 모든 노드에서 정책을 구성하는 데 예상보다 오래 걸리는 경우 클러스터의 시행 상태를 나열할 수 있습니다.

```
$ oc get nnce
```

4. 선택 사항: 구성 실패에 대한 오류 보고를 포함하여 특정 시행의 구성을 확인하려면 다음 명령을 실행하십시오.

```
$ oc get nnce <node>.<policy> -o yaml
```

12.2.4. 노드에서 인터페이스 제거

NodeNetworkConfigurationPolicy 오브젝트를 편집하고 인터페이스의 **state**를 없음으로 설정하여 클러스터의 **1**개 이상의 노드에서 인터페이스를 제거할 수 있습니다.

노드에서 인터페이스를 제거해도 노드 네트워크 구성이 이전 상태로 자동 복원되지 않습니다. 이전 상태를 복원하려면 정책에서 노드 네트워크 구성을 정의해야 합니다.

브리지 또는 본딩 인터페이스를 제거하면 이전에 해당 브릿지 또는 본딩 인터페이스에 연결되었거나 종속되었던 클러스터의 모든 노드 **NIC**가 **down** 상태가 되어 연결할 수 없습니다. 연결 손실을 방지하기 위해, 노드 **NIC**를 동일한 정책으로 구성하여 **DHCP** 또는 고정 **IP** 주소의 상태를 **up**으로 구성합니다.

참고

인터페이스를 추가한 노드 네트워크 정책을 삭제해도 노드의 정책 구성은 변경되지 않습니다. **NodeNetworkConfigurationPolicy**는 클러스터의 오브젝트이지만 요청된 구성만 나타냅니다.
마찬가지로 인터페이스를 제거해도 정책은 삭제되지 않습니다.

절차

1.

인터페이스를 생성하는 데 사용되는 **NodeNetworkConfigurationPolicy** 매니페스트를 업데이트합니다. 다음 예에서는 **Linux** 브릿지를 제거한 후 연결이 손실되지 않도록 **DHCP**로 **eth1 NIC**를 구성합니다.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: <br1-eth1-policy> 1
spec:
  nodeSelector: 2
    node-role.kubernetes.io/worker: "" 3
  desiredState:
    interfaces:
    - name: br1
      type: linux-bridge
      state: absent 4
    - name: eth1 5
      type: ethernet 6
      state: up 7
      ipv4:
        dhcp: true 8
        enabled: true 9
```

1

정책 이름입니다.

2

선택 사항: **nodeSelector** 매개변수를 포함하지 않으면 정책이 클러스터의 모든 노드에 적용됩니다.

3

이 예제에서는 **node-role.kubernetes.io/worker: ""** 노드 선택기를 사용하여 클러스터의 모든 작업자 노드를 선택합니다.

4

absent 상태로 변경하면 인터페이스가 제거됩니다.

5

브리지 인터페이스에서 연결을 해제할 인터페이스의 이름입니다.

6

인터페이스 유형입니다. 이 예제에서는 이더넷 네트워킹 인터페이스를 생성합니다.

7

인터페이스에 요청되는 상태입니다.

8

선택 사항: dhcp를 사용하지 않는 경우 고정 IP를 설정하거나 IP 주소 없이 인터페이스를 종료할 수 있습니다.

9

이 예제에서 ipv4를 활성화합니다.

2. 노드에서 정책을 업데이트하고 인터페이스를 제거합니다.

```
$ oc apply -f <br1-eth1-policy.yaml> 1
```

1

정책 매니페스트의 파일 이름입니다.

12.2.5. 다양한 인터페이스에 대한 예제 정책 구성

12.2.5.1. 예: Linux 브리지 인터페이스 노드 네트워크 구성 정책

NodeNetworkConfigurationPolicy 매니페스트를 클러스터에 적용하여 클러스터의 노드에서 Linux 브리지 인터페이스를 만듭니다.

다음 YAML 파일은 Linux 브리지 인터페이스의 매니페스트 예제입니다. 여기에는 해당 정보로 교체해야 하는 샘플 값이 포함되어 있습니다.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: br1-eth1-policy (1)
spec:
  nodeSelector: (2)
    kubernetes.io/hostname: <node01> (3)
  desiredState:
    interfaces:
      - name: br1 (4)
        description: Linux bridge with eth1 as a port (5)
        type: linux-bridge (6)
        state: up (7)
        ipv4:
          dhcp: true (8)
          enabled: true (9)
        bridge:
          options:
            stp:
              enabled: false (10)
          port:
            - name: eth1 (11)
```

(1) 정책 이름입니다.

(2) 선택 사항: **nodeSelector** 매개변수를 포함하지 않으면 정책이 클러스터의 모든 노드에 적용됩니다.

(3) 이 예제에서는 **hostname** 노드 선택기를 사용합니다.

(4) 인터페이스 이름입니다.

(5) 선택 사항: 사람이 읽을 수 있는 인터페이스 설명입니다.

(6)

7

생성 후 인터페이스에 요청되는 상태입니다.

8

선택 사항: **dhcp**를 사용하지 않는 경우 고정 **IP**를 설정하거나 **IP** 주소 없이 인터페이스를 종료할 수 있습니다.

9

이 예제에서 **ipv4**를 활성화합니다.

10

이 예제에서 **stp**를 비활성화합니다.

11

브리지가 연결되는 노드 **NIC**입니다.

12.2.5.2. 예제: VLAN 인터페이스 노드 네트워크 구성 정책

NodeNetworkConfigurationPolicy 매니페스트를 클러스터에 적용하여 클러스터의 노드에서 **VLAN** 인터페이스를 만듭니다.

다음 **YAML** 파일은 **VLAN** 인터페이스의 매니페스트 예제입니다. 여기에는 해당 정보로 교체해야 하는 샘플 값이 포함되어 있습니다.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: vlan-eth1-policy 1
spec:
  nodeSelector: 2
    kubernetes.io/hostname: <node01> 3
  desiredState:
    interfaces:
    - name: eth1.102 4
      description: VLAN using eth1 5
      type: vlan 6
      state: up 7
```

```
    vlan:
      base-iface: eth1 8
      id: 102 9
```

1

정책 이름입니다.

2

선택 사항: **nodeSelector** 매개변수를 포함하지 않으면 정책이 클러스터의 모든 노드에 적용됩니다.

3

이 예제에서는 **hostname** 노드 선택기를 사용합니다.

4

인터페이스 이름입니다.

5

선택 사항: 사람이 읽을 수 있는 인터페이스 설명입니다.

6

인터페이스 유형입니다. 이 예제에서는 **VLAN**을 만듭니다.

7

생성 후 인터페이스에 요청되는 상태입니다.

8

VLAN이 연결되는 노드 **NIC**입니다.

9

VLAN 태그입니다.

12.2.5.3. 예제: 본딩 인터페이스 노드 네트워크 구성 정책

NodeNetworkConfigurationPolicy 매니페스트를 클러스터에 적용하여 클러스터의 노드에서 본딩 인터페이스를 만듭니다.

참고

OpenShift Virtualization에서는 다음 본딩 모드만 지원합니다.

- mode=1 active-backup
- mode=2 balance-xor
- mode=4 802.3ad
- mode=5 balance-tlb
- mode=6 balance-alb

다음 YAML 파일은 본딩 인터페이스의 매니페스트 예제입니다. 여기에는 해당 정보로 교체해야 하는 샘플 값이 포함되어 있습니다.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: bond0-eth1-eth2-policy (1)
spec:
  nodeSelector: (2)
    kubernetes.io/hostname: <node01> (3)
  desiredState:
    interfaces:
    - name: bond0 (4)
      description: Bond with ports eth1 and eth2 (5)
      type: bond (6)
      state: up (7)
      ipv4:
        dhcp: true (8)
        enabled: true (9)
      link-aggregation:
```

```
    mode: active-backup (10)
    options:
      miimon: '140' (11)
    port: (12)
    - eth1
    - eth2
  mtu: 1450 (13)
```

(1)

정책 이름입니다.

(2)

선택 사항: **nodeSelector** 매개변수를 포함하지 않으면 정책이 클러스터의 모든 노드에 적용됩니다.

(3)

이 예제에서는 **hostname** 노드 선택기를 사용합니다.

(4)

인터페이스 이름입니다.

(5)

선택 사항: 사람이 읽을 수 있는 인터페이스 설명입니다.

(6)

인터페이스 유형입니다. 이 예제에서는 본딩을 생성합니다.

(7)

생성 후 인터페이스에 요청되는 상태입니다.

(8)

선택 사항: **dhcp**를 사용하지 않는 경우 고정 **IP**를 설정하거나 **IP** 주소 없이 인터페이스를 종료할 수 있습니다.

(9)

이 예제에서 **ipv4**를 활성화합니다.

(10)

본딩의 드라이버 모드입니다. 이 예제에서는 활성 백업 모드를 사용합니다.

11

선택 사항: 이 예제에서는 miimon을 사용하여 140ms마다 본딩 링크를 검사합니다.

12

본딩의 하위 노드 NIC입니다.

13

선택 사항: 본딩의 MTU(최대 전송 단위)입니다. 지정하지 않는 경우 이 값은 기본적으로 1500으로 설정됩니다.

12.2.5.4. 예제: 이더넷 인터페이스 노드 네트워크 구성 정책

NodeNetworkConfigurationPolicy 매니페스트를 클러스터에 적용하여 클러스터의 노드에서 이더넷 인터페이스를 구성합니다.

다음 YAML 파일은 이더넷 인터페이스의 매니페스트 예제입니다. 여기에는 해당 정보로 교체해야 하는 샘플 값이 포함되어 있습니다.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: eth1-policy 1
spec:
  nodeSelector: 2
    kubernetes.io/hostname: <node01> 3
  desiredState:
    interfaces:
    - name: eth1 4
      description: Configuring eth1 on node01 5
      type: ethernet 6
      state: up 7
      ipv4:
        dhcp: true 8
        enabled: true 9
```

1

정책 이름입니다.

2

선택 사항: **nodeSelector** 매개변수를 포함하지 않으면 정책이 클러스터의 모든 노드에 적용됩니다.

3

이 예제에서는 **hostname** 노드 선택기를 사용합니다.

4

인터페이스 이름입니다.

5

선택 사항: 사람이 읽을 수 있는 인터페이스 설명입니다.

6

인터페이스 유형입니다. 이 예제에서는 이더넷 네트워킹 인터페이스를 생성합니다.

7

생성 후 인터페이스에 요청되는 상태입니다.

8

선택 사항: **dhcp**를 사용하지 않는 경우 고정 **IP**를 설정하거나 **IP** 주소 없이 인터페이스를 종료할 수 있습니다.

9

이 예제에서 **ipv4**를 활성화합니다.

12.2.5.5. 예제: 노드 네트워크 구성 정책이 동일한 여러 인터페이스

동일한 노드 네트워크 구성 정책으로 여러 개의 인터페이스를 생성할 수 있습니다. 이러한 인터페이스는 서로를 참조할 수 있으므로 단일 정책 매니페스트를 사용하여 네트워크 구성을 빌드하고 배포할 수 있습니다.

다음 예제 스니펫에서는 두 **NIC**에 걸친 **bond10**이라는 본딩과 이 본딩에 연결되는 **br1**이라는 **Linux** 브리지를 생성합니다.

```
#...
    interfaces:
    - name: bond10
      description: Bonding eth2 and eth3 for Linux bridge
      type: bond
      state: up
      link-aggregation:
        port:
        - eth2
        - eth3
    - name: br1
      description: Linux bridge on bond
      type: linux-bridge
      state: up
      bridge:
        port:
        - name: bond10
#...
```

12.2.6. 브리지에 연결된 NIC의 고정 IP 캡처

중요

NIC의 고정 IP를 캡처하는 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.

12.2.6.1. 예: 브릿지에 연결된 NIC에서 고정 IP 주소를 상속하는 Linux 브리지 인터페이스 노드 네트워크 구성 정책

클러스터의 노드에서 Linux 브리지 인터페이스를 만들고 단일 NodeNetworkConfigurationPolicy 매니페스트를 클러스터에 적용하여 NIC의 고정 IP 구성을 브리지에 전송합니다.

다음 YAML 파일은 Linux 브리지 인터페이스의 매니페스트 예제입니다. 여기에는 해당 정보로 교체해야 하는 샘플 값이 포함되어 있습니다.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: br1-eth1-copy-ipv4-policy (1)
spec:
```

```
  nodeSelector: (2)
    node-role.kubernetes.io/worker: ""
  capture:
    eth1-nic: interfaces.name=="eth1" (3)
    eth1-routes: routes.running.next-hop-interface=="eth1"
    br1-routes: capture.eth1-routes | routes.running.next-hop-interface := "br1"
  desiredState:
    interfaces:
      - name: br1
        description: Linux bridge with eth1 as a port
        type: linux-bridge (4)
        state: up
        ipv4: "{{ capture.eth1-nic.interfaces.0.ipv4 }}" (5)
        bridge:
          options:
            stp:
              enabled: false
          port:
            - name: eth1 (6)
    routes:
      config: "{{ capture.br1-routes.routes.running }}"
```

(1) 정책의 이름입니다.

(2) 선택 사항: **nodeSelector** 매개변수를 포함하지 않으면 정책이 클러스터의 모든 노드에 적용됩니다. 이 예제에서는 **node-role.kubernetes.io/worker: ""** 노드 선택기를 사용하여 클러스터의 모든 작업자 노드를 선택합니다.

(3) 브리지가 연결되는 노드 **NIC**에 대한 참조입니다.

(4) 인터페이스 유형입니다. 이 예제에서는 브리지를 만듭니다.

(5) 브리지 인터페이스의 **IP** 주소입니다. 이 값은 **spec.capture.eth1-nic** 항목에서 참조하는 **NIC**의 **IP** 주소와 일치합니다.

(6) 브리지가 연결되는 노드 **NIC**입니다.

추가 리소스

- NMPolicy 프로젝트 - 정책 구문

12.2.7. 예제: IP 관리

다음 예제 구성 스니펫에서는 다양한 IP 관리 방법을 보여줍니다.

이 예제에서는 ethernet 인터페이스 유형을 사용하여 예제를 단순화하면서 정책 구성에 관련 컨텍스트를 표시합니다. 이러한 IP 관리 예제는 다른 인터페이스 유형과 함께 사용할 수 있습니다.

12.2.7.1. 고정

다음 스니펫은 이더넷 인터페이스에서 IP 주소를 정적으로 구성합니다.

```
...
  interfaces:
  - name: eth1
    description: static IP on eth1
    type: ethernet
    state: up
    ipv4:
      dhcp: false
      address:
      - ip: 192.168.122.250 1
        prefix-length: 24
      enabled: true
...
```

1

이 값을 인터페이스의 고정 IP 주소로 교체합니다.

12.2.7.2. IP 주소 없음

다음 스니펫에서는 인터페이스에 IP 주소가 없습니다.

```
...
  interfaces:
  - name: eth1
    description: No IP on eth1
    type: ethernet
```

```
      state: up
      ipv4:
        enabled: false
...
```

12.2.7.3. 동적 호스트 구성

다음 스니펫에서는 동적 IP 주소, 게이트웨이 주소, DNS를 사용하는 이더넷 인터페이스를 구성합니다.

```
...
    interfaces:
    - name: eth1
      description: DHCP on eth1
      type: ethernet
      state: up
      ipv4:
        dhcp: true
        enabled: true
...
```

다음 스니펫에서는 동적 IP 주소를 사용하지만 동적 게이트웨이 주소 또는 DNS를 사용하지 않는 이더넷 인터페이스를 구성합니다.

```
...
    interfaces:
    - name: eth1
      description: DHCP without gateway or DNS on eth1
      type: ethernet
      state: up
      ipv4:
        dhcp: true
        auto-gateway: false
        auto-dns: false
        enabled: true
...
```

12.2.7.4. DNS

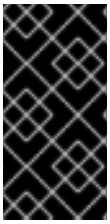
DNS 구성을 설정하는 것은 /etc/resolv.conf 파일을 수정하는 데 유용합니다. 다음 스니펫에서는 호스트에 DNS 구성을 설정합니다.

```
...
    interfaces: 1
      ...
      ipv4:
        ...
        auto-dns: false
```

```
      ...
  dns-resolver:
    config:
      search:
      - example.com
      - example.org
      server:
      - 8.8.8.8
...
```

1

auto-dns: false 를 사용하여 인터페이스를 구성하거나 Kubernetes NMState가 사용자 지정 DNS 설정을 저장하려면 인터페이스에서 고정 IP 구성을 사용해야 합니다.



중요

OVNKubernetes에서 관리하는 Open vSwitch 브리지인 br-ex 를 DNS 확인자를 구성할 때 인터페이스로 사용할 수 없습니다.

12.2.7.5. 고정 라우팅

다음 스니펫에서는 eth1 인터페이스에 고정 경로와 고정 IP를 구성합니다.

```
...
  interfaces:
  - name: eth1
    description: Static routing on eth1
    type: ethernet
    state: up
    ipv4:
      dhcp: false
      address:
      - ip: 192.0.2.251 1
        prefix-length: 24
      enabled: true
  routes:
    config:
    - destination: 198.51.100.0/24
      metric: 150
      next-hop-address: 192.0.2.1 2
      next-hop-interface: eth1
      table-id: 254
...
```

1

이더넷 인터페이스의 고정 IP 주소입니다.

2

노드 트래픽의 다음 홉 주소입니다. 이더넷 인터페이스에 설정된 IP 주소와 동일한 서브넷에 있어야 합니다.

12.3. 노드 네트워크 구성 문제 해결

노드 네트워크 구성에 문제가 발생하면 정책이 자동으로 롤백되고 시행이 실패로 보고됩니다. 여기에는 다음과 같은 문제가 포함됩니다.

- 호스트에 구성을 적용하지 못했습니다.
- 호스트와 기본 게이트웨이의 연결이 끊어졌습니다.
- 호스트와 API 서버의 연결이 끊어졌습니다.

12.3.1. 잘못된 노드 네트워크 구성 정책의 구성 문제 해결

노드 네트워크 구성 정책을 적용하여 전체 클러스터에 노드 네트워크 구성 변경 사항을 적용할 수 있습니다. 잘못된 구성을 적용하는 경우 다음 예제를 사용하여 실패한 노드 네트워크 정책의 문제를 해결하고 수정할 수 있습니다.

이 예에서는 컨트롤 플레인 노드(마스터)와 컴퓨팅(작업자) 노드가 각각 3개씩 있는 예시 클러스터에 Linux 브리지 정책을 적용합니다. 이 정책은 잘못된 인터페이스를 참조하므로 적용되지 않습니다. 오류를 찾기 위해 사용 가능한 NMState 리소스를 조사합니다. 그런 다음 올바른 구성으로 정책을 업데이트할 수 있습니다.

절차

1. 정책을 생성하여 클러스터에 적용합니다. 다음 예제에서는 ens01 인터페이스에서 간단한 브리지를 생성합니다.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: ens01-bridge-testfail
spec:
  desiredState:
    interfaces:
```

```
    - name: br1
      description: Linux bridge with the wrong port
      type: linux-bridge
      state: up
      ipv4:
        dhcp: true
        enabled: true
      bridge:
        options:
          stp:
            enabled: false
        port:
          - name: ens01
```

```
$ oc apply -f ens01-bridge-testfail.yaml
```

출력 예

```
nodenetworkconfigurationpolicy.nmstate.io/ens01-bridge-testfail created
```

2. 다음 명령을 실행하여 정책의 상태를 확인합니다.

```
$ oc get nncp
```

출력에 정책이 실패했다는 내용이 표시됩니다.

출력 예

```
NAME                STATUS
ens01-bridge-testfail   FailedToConfigure
```

그러나 정책 상태만으로는 모든 노드에서 실패했는지 노드 서브 세트에서 실패했는지 알 수 없습니다.

3.

노드 네트워크 구성 시행을 나열하여 정책이 모든 노드에서 성공적인지 확인합니다. 정책이 노드 서브 세트에서만 실패한 경우 특정 노드 구성에 문제가 있음을 나타냅니다. 정책이 모든 노드에서 실패하면 정책에 문제가 있음을 나타냅니다.

```
$ oc get nnce
```

출력에 정책이 모든 노드에서 실패했다는 내용이 표시됩니다.

출력 예

```
NAME                                STATUS
control-plane-1.ens01-bridge-testfail      FailedToConfigure
control-plane-2.ens01-bridge-testfail      FailedToConfigure
control-plane-3.ens01-bridge-testfail      FailedToConfigure
compute-1.ens01-bridge-testfail            FailedToConfigure
compute-2.ens01-bridge-testfail            FailedToConfigure
compute-3.ens01-bridge-testfail            FailedToConfigure
```

4. 실패한 시행 중 하나에서 역추적을 살펴봅니다. 다음 명령은 출력 툴 **jsonpath**를 사용하여 출력을 필터링합니다.

```
$ oc get nnce compute-1.ens01-bridge-testfail -o jsonpath='{.status.conditions[?
(@.type=="Failing")].message}'
```

이 명령은 간결하게 편집된 대규모 역추적 정보를 반환합니다.

출력 예

```
error reconciling NodeNetworkConfigurationPolicy at desired state apply: , failed to
execute nmstatectl set --no-commit --timeout 480: 'exit status 1' ''
...
libnmstate.error.NmstateVerificationError:
desired
=======
---
name: br1
type: linux-bridge
state: up
bridge:
```

```
  options:
    group-forward-mask: 0
    mac-ageing-time: 300
    multicast-snooping: true
    stp:
      enabled: false
      forward-delay: 15
      hello-time: 2
      max-age: 20
      priority: 32768
  port:
  - name: ens01
description: Linux bridge with the wrong port
ipv4:
  address: []
  auto-dns: true
  auto-gateway: true
  auto-routes: true
  dhcp: true
  enabled: true
ipv6:
  enabled: false
mac-address: 01-23-45-67-89-AB
mtu: 1500

current
=======
---
name: br1
type: linux-bridge
state: up
bridge:
  options:
    group-forward-mask: 0
    mac-ageing-time: 300
    multicast-snooping: true
    stp:
      enabled: false
      forward-delay: 15
      hello-time: 2
      max-age: 20
      priority: 32768
  port: []
description: Linux bridge with the wrong port
ipv4:
  address: []
  auto-dns: true
  auto-gateway: true
  auto-routes: true
  dhcp: true
  enabled: true
ipv6:
  enabled: false
mac-address: 01-23-45-67-89-AB
mtu: 1500
```

```
difference
==========
--- desired
+++ current
@@ -13,8 +13,7 @@
      hello-time: 2
      max-age: 20
      priority: 32768
-  port:
-  - name: ens01
+  port: []
   description: Linux bridge with the wrong port
   ipv4:
     address: []
   line 651, in _assert_interfaces_equal\n
current_state.interfaces[ifname],\nlibnmstate.error.NmstateVerificationError:
```

NmstateVerificationError는 **desired** 정책 구성, 노드에 있는 정책의 **current** 구성, 일치하지 않는 매개변수를 강조하는 **difference**를 나열합니다. 이 예에서 **port**는 **difference**에 포함되어 있으며, 이는 정책의 포트 구성이 문제임을 나타냅니다.

5. 정책이 제대로 구성되었는지 확인하기 위해 **NodeNetworkState** 오브젝트를 요청하여 하나 또는 모든 노드의 네트워크 구성을 확인합니다. 다음 명령에서는 **control-plane-1** 노드의 네트워크 구성을 반환합니다.

```
$ oc get nns control-plane-1 -o yaml
```

출력에 노드의 인터페이스 이름이 **ens1**인데 실패한 정책에서 **ens01**로 잘못 사용하고 있다는 내용이 표시됩니다.

출력 예

```
  - ipv4:
...
    name: ens1
    state: up
    type: ethernet
```

6. 기존 정책을 편집하여 오류를 수정합니다.

```
$ oc edit nncp ens01-bridge-testfail
```

```
...
        port:
         - name: ens1
```

정책을 저장하여 수정 사항을 적용합니다.

7. 정책 상태를 확인하여 업데이트가 완료되었는지 확인합니다.

```
$ oc get nncp
```

출력 예

```
NAME                    STATUS
ens01-bridge-testfail   SuccessfullyConfigured
```

업데이트된 정책이 클러스터의 모든 노드에 성공적으로 구성되었습니다.

13장. 로깅, 이벤트, 모니터링

13.1. 가상화 개요 검토

가상화 개요 페이지는 가상화 리소스, 세부 정보, 상태 및 상위 소비자에 대한 포괄적인 보기를 제공합니다. OpenShift Virtualization의 전반적인 상태에 대한 통찰력을 확보하여 데이터를 검사하여 식별한 특정 문제를 해결하기 위해 개입이 필요한지 확인할 수 있습니다.

시작하기 리소스를 사용하여 퀵 스타트에 액세스하고, 가상화의 최신 블로그를 읽고, 운영자를 사용하는 방법을 알아보십시오. 가상 시스템의 경고, 이벤트, 인벤토리 및 상태에 대한 완전한 정보를 얻습니다. Top Consumer 카드를 사용자 지정하여 프로젝트, 가상 시스템 또는 노드에 의해 특정 리소스의 높은 사용률에 대한 데이터를 가져옵니다. 대시보드 페이지에 대한 빠른 액세스를 위해 가상화 대시보드 보기를 클릭합니다.

13.1.1. 사전 요구 사항

Top Consumers 카드에서 vCPU wait 메트릭을 사용하려면 MachineConfig 오브젝트에 schedstats=enable 커널 인수를 적용해야 합니다. 이 커널 인수를 사용하면 디버깅 및 성능 튜닝에 사용되는 스케줄러 통계가 활성화되고 스케줄러에 약간의 부하가 추가됩니다. 커널 인수 적용에 대한 자세한 내용은 OpenShift Container Platform 머신 구성 작업 설명서를 참조하십시오.

13.1.2. 가상화 개요 페이지에서 적극적으로 모니터링되는 리소스

다음 표에서는 가상화 개요 페이지의 활성 모니터링 리소스, 메트릭 및 필드를 보여줍니다. 이 정보는 관련 데이터를 얻고 문제를 해결하기 위해 개입해야 할 때 유용합니다.

모니터링된 리소스, 필드 및 메트릭	설명
세부 정보	OpenShift Virtualization 에 대한 서비스 및 버전 정보에 대한 간략한 개요입니다.
상태	가상화 및 네트워킹에 대한 경고입니다.
activity	가상 머신에 대한 지속적인 이벤트입니다. 메시지는 Pod 생성 또는 다른 호스트로의 가상 머신 마이그레이션과 같은 클러스터의 최근 활동과 관련이 있습니다.
템플릿별 VM 실행	도넛형 차트에는 각 가상 머신 템플릿에 고유한 색상이 표시되고 각 템플릿을 사용하는 실행 중인 가상 머신의 수가 표시됩니다.
inventory	총 활성 가상 머신, 템플릿, 노드 및 네트워크 수입니다.

VM 상태	가상 머신의 현재 상태: **실행**,**프로비저닝**,**시작**,**마이그레이션**,**일시 중지**,**중지**,**종료** 및 **알 수 없음**.
권한	권한을 통해 기능을 사용하도록 설정하는 작업: **공용 템플릿에 대한 액세스**,**공용 부팅 소스에 대한 액세스**,**VM 복제**,**여러 네트워크에 VM 연결**,**로컬 디스크에서 기본 이미지 업로드** 및 **공유 템플릿 공유**.

13.1.3. 상위 소비에 대해 모니터링되는 리소스

가상화 개요 페이지의 Top Consumers 카드에는 리소스 사용량이 최대인 프로젝트, 가상 머신 또는 노드가 표시됩니다. 프로젝트, 가상 머신 또는 노드를 선택하고 특정 리소스의 상위 5개 또는 상위 10개 이상의 소비자를 볼 수 있습니다.

참고

최대 리소스 사용량 보기는 각 Top Consumers 카드 내에서 상위 5개 또는 상위 10개의 소비자에게 제한됩니다.

다음 표에서는 상위 소비자에 대해 모니터링되는 리소스를 보여줍니다.

상위 소비에 대해 모니터링되는 리소스	설명
CPU	CPU를 가장 많이 사용하는 프로젝트, 가상 시스템 또는 노드
메모리	가장 많은 메모리를 사용하는 프로젝트, 가상 시스템 또는 노드(바이트 단위)입니다. 디스플레이 단위(예: MiB 또는 GiB)는 리소스 소비 크기에 따라 결정됩니다.
사용된 파일 시스템	파일 시스템 사용량이 가장 많은 프로젝트, 가상 시스템 또는 노드(바이트 단위)입니다. 디스플레이 단위(예: MiB 또는 GiB)는 리소스 소비 크기에 따라 결정됩니다.
메모리 스왑	메모리를 스왑할 때 가장 많은 메모리 부족을 사용하는 프로젝트, 가상 시스템 또는 노드.
vCPU 대기 시간	vCPU의 최대 대기 시간(초)이 발생하는 프로젝트, 가상 시스템 또는 노드.
스토리지 처리량	스토리지 미디어(mbps)에서 데이터 전송 속도가 가장 높은 프로젝트, 가상 머신 또는 노드

스토리지 IOPS	일정 기간 동안 대규모 스토리지 IOPS(초당 입력/출력 작업)가 있는 프로젝트, 가상 머신 또는 노드

13.1.4. 프로젝트, 가상 시스템 및 노드의 상위 소비자 검토

가상화 개요 페이지에서 선택한 프로젝트, 가상 머신 또는 노드에 대한 리소스 상위 소비자를 볼 수 있습니다.

사전 요구 사항

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

절차

1. OpenShift Virtualization 웹 콘솔의 관리자 관점에서 가상화 → 개요 로 이동합니다.
2. Top Consumers 카드로 이동합니다.
3. 드롭다운 메뉴에서 최상위 5 표시 또는 상위 10 을 선택합니다.
4. Top Consumer 카드의 경우 드롭다운 메뉴에서 리소스 유형 (CPU,메모리,사용 파일 시스템, 메모리스왑,vCPU Wait) 또는 스토리지 처리량 을 선택합니다.
5. 프로젝트,VM 또는 노드별 을 선택합니다. 선택한 리소스의 상위 5개 또는 상위 10개 소비자 목록이 표시됩니다.

13.1.5. 추가 리소스

- 모니터링 개요
- 모니터링 대시보드 검토
- 대시보드

13.2. 가상 머신 로그 보기

13.2.1. 가상 머신 로그 정보

로그는 OpenShift Container Platform 빌드, 배포, Pod와 관련하여 수집됩니다. OpenShift Virtualization에서는 웹 콘솔 또는 CLI의 가상 머신 시작 관리자 Pod에서 가상 머신 로그를 검색할 수 있습니다.

-f 옵션은 로그 출력을 실시간으로 추적하므로 진행률을 모니터링하고 오류를 점검하는 데 유용합니다.

시작 관리자 Pod가 시작되지 않으면 **--previous** 옵션을 사용하여 마지막 시도의 로그를 확인하십시오.

주의

ErrImagePull 및 **ImagePullBackOff** 오류는 잘못된 배포 구성이나 참조하는 이미지 문제로 인해 발생할 수 있습니다.

13.2.2. CLI에서 가상 머신 로그 보기

가상 머신 시작 관리자 Pod에서 가상 머신 로그를 가져옵니다.

절차

- 다음 명령을 사용합니다.

```
$ oc logs <virt-launcher-name>
```

13.2.3. 웹 콘솔에서 가상 머신 로그 보기

연결된 가상 머신 시작 관리자 Pod에서 가상 머신 로그를 가져옵니다.

절차

1. OpenShift Container Platform 콘솔의 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.

2. 가상 머신을 선택하여 VirtualMachine 세부 정보 페이지를 엽니다.

3. 세부 정보 탭을 클릭합니다.

4. Pod 섹션에서 virt-launcher-<name > Pod를 클릭하여 Pod 세부 정보 페이지를 엽니다.

5. 로그 탭을 클릭하여 Pod 로그를 확인합니다.

13.3. 이벤트 보기

13.3.1. 가상 머신 이벤트 정보

OpenShift Container Platform 이벤트는 네임스페이스에 있는 중요 라이프사이클 정보로 이루어진 레코드이며, 리소스 스케줄링, 생성, 삭제와 관련된 문제를 모니터링하고 해결하는 데 유용합니다.

OpenShift Virtualization에서는 가상 머신 및 가상 머신 인스턴스에 대한 이벤트를 추가합니다. 해당 이벤트는 웹 콘솔이나 CLI에서 볼 수 있습니다.

OpenShift Container Platform 클러스터에서 시스템 이벤트 정보 보기도 참조하십시오.

13.3.2. 웹 콘솔에서 가상 머신 이벤트 보기

웹 콘솔의 VirtualMachine 세부 정보 페이지에서 실행 중인 가상 머신에 대한 스트리밍 이벤트를 볼 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → VirtualMachines 를 클릭합니다.

2. 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.

3. 가상 머신의 스트리밍 이벤트를 보려면 이벤트 탭을 클릭합니다.

- **ECDHE button**은 이벤트 스트림을 일시 중지합니다.
- ▶ 버튼은 일시 중지된 이벤트 스트림을 다시 시작합니다.

13.3.3. CLI에서 네임스페이스 이벤트 보기

OpenShift Container Platform 클라이언트를 사용하여 네임스페이스에 대한 이벤트를 가져옵니다.

절차

- 네임스페이스에서 **oc get** 명령을 사용합니다.

```
$ oc get events
```

13.3.4. CLI에서 리소스 이벤트 보기

이벤트는 리소스 설명에 포함되어 있으며 **OpenShift Container Platform** 클라이언트를 사용하여 가져올 수 있습니다.

절차

- 네임스페이스에서 **oc describe** 명령을 사용합니다. 다음 예제에서는 가상 머신, 가상 머신 인스턴스, 가상 머신에 대한 **virt-launcher Pod**에 대한 이벤트를 가져오는 방법을 보여줍니다.

```
$ oc describe vm <vm>
```

```
$ oc describe vmi <vmi>
```

```
$ oc describe pod virt-launcher-<name>
```

13.4. 이벤트 및 조건을 사용하여 데이터 볼륨 진단

oc describe 명령을 사용하여 데이터 볼륨 문제를 분석하고 해결합니다.

13.4.1. 조건 및 이벤트 정보

다음 명령으로 생성된 Conditions 및 Events 섹션의 출력을 검사하여 데이터 볼륨 문제를 진단합니다.

```
$ oc describe dv <DataVolume>
```

표시되는 Conditions 섹션에는 세 가지 Types이 있습니다.

- Bound
- Running
- Ready

Events 섹션에서는 다음과 같은 추가 정보를 제공합니다.

- 이벤트 Type
- 로깅 Reason
- 이벤트 Source
- 추가 진단 정보가 포함된 Message

oc describe의 출력에 항상 Events가 포함되는 것은 아닙니다.

Status, Reason 또는 Message가 변경되면 이벤트가 생성됩니다. 조건과 이벤트는 모두 데이터 볼륨의 상태 변화에 반응합니다.

예를 들어 가져오기 작업 중에 URL을 잘못 입력하면 가져오기 작업에서 404 메시지를 생성합니다. 이러한 메시지 변경으로 인해 원인이 포함된 이벤트가 생성됩니다. Conditions 섹션의 출력도 업데이트됩니다.

13.4.2. 조건 및 이벤트를 사용하여 데이터 볼륨 분석

describe 명령으로 생성된 Conditions 및 Events 섹션을 검사하여 PVC(영구 볼륨 클레임)와 관련된 데이터 볼륨 상태 및 작업이 활발하게 실행되고 있거나 완료되었는지의 여부를 확인합니다. 데이터 볼륨의 상태와 어떻게 해서 현재 상태가 되었는지에 대한 구체적인 정보를 제공하는 메시지가 표시될 수도 있습니다.

조건은 다양한 형태로 조합될 수 있습니다. 각각 고유의 컨텍스트에서 평가해야 합니다.

다음은 다양한 조합의 예입니다.

- Bound – 이 예제에는 성공적으로 바인딩된 PVC가 표시됩니다.

  Type은 Bound이므로 Status가 True입니다. PVC가 바인딩되지 않은 경우 Status는 False입니다.

  PVC가 바인딩되면 PVC가 바인딩되었음을 알리는 이벤트가 생성됩니다. 이 경우 Reason은 Bound이고 Status는 True입니다. Message는 데이터 볼륨이 속한 PVC를 나타냅니다.

  Events 섹션의 Message에서는 PVC가 바인딩된 기간(Age) 및 리소스(From)(이 경우 datavolume-controller)를 포함한 추가 세부 정보를 제공합니다.

  출력 예

```
Status:
 Conditions:
  Last Heart Beat Time:  2020-07-15T03:58:24Z
  Last Transition Time:  2020-07-15T03:58:24Z
  Message:               PVC win10-rootdisk Bound
  Reason:                Bound
  Status:                True
  Type:                  Bound
```

```
Events:
 Type    Reason    Age    From                Message
 ----    ------    ----   ----                -------
 Normal  Bound     24s    datavolume-controller  PVC example-dv Bound
```

- **Running** – 이 경우 **Type**은 **Running**이고 **Status**는 **False**입니다. 이는 시도한 작업을 실패하게 만드는 이벤트가 발생하여 상태가 **True**에서 **False**로 변경되었음을 나타냅니다.

  그러나 **Reason**이 **Completed**이고 **Message** 필드에 **Import Complete**라고 표시됩니다.

  Events 섹션의 **Reason** 및 **Message**에는 실패한 작업에 대한 추가 문제 해결 정보가 포함되어 있습니다. 이 예제에서는 **Message**에 **Events** 섹션의 첫 번째 **Warning**에 나열된 **404**로 인해 연결할 수 없다는 내용이 표시됩니다.

  이러한 정보를 통해 가져오기 작업이 실행 중이며 데이터 볼륨에 액세스하려는 다른 작업에 대한 경합이 발생한다는 것을 알 수 있습니다.

  출력 예

```
Status:
  Conditions:
    Last Heart Beat Time:  2020-07-15T04:31:39Z
    Last Transition Time:  2020-07-15T04:31:39Z
    Message:               Import Complete
    Reason:                Completed
    Status:                False
    Type:                  Running

Events:
  Type    Reason          Age              From                Message
  ----    ------          ----             ----                -------
  Warning  Error          12s (x2 over 14s)  datavolume-controller  Unable to connect
  to http data source: expected status code 200, got 404. Status: 404 Not Found
```

- **Ready** – **Type**이 **Ready**이고 **Status**가 **True**이면 다음 예제와 같이 데이터 볼륨을 사용할 준비가 된 것입니다. 데이터 볼륨을 사용할 준비가 되지 않은 경우에는 **Status**가 **False**입니다.

출력 예

```
Status:
  Conditions:
    Last Heart Beat Time: 2020-07-15T04:31:39Z
    Last Transition Time:  2020-07-15T04:31:39Z
    Status:               True
    Type:                 Ready
```

13.5. 가상 머신 워크로드에 대한 정보 보기

OpenShift Container Platform 웹 콘솔의 가상 머신 대시보드를 사용하여 가상 머신에 대한 개괄적인 정보를 볼 수 있습니다.

13.5.1. 가상 머신 대시보드 정보

가상화 → VirtualMachines 페이지로 이동하고 가상 머신(VM)을 클릭하여 VirtualMachine 세부 정보 페이지를 확인하여 OpenShift Container Platform 웹 콘솔에서 VM(가상 머신)에 액세스합니다.

개요 탭에는 다음 카드가 표시됩니다.

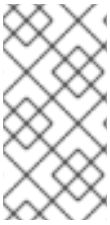
- 세부 정보에는 다음을 포함하여 가상 머신에 대한 식별 정보가 있습니다.
  - 이름
  - 네임스페이스
  - 작성일
  - 노드 이름

- IP 주소

- 인벤토리에는 다음을 포함하여 가상 머신의 리소스가 나열됩니다.
  - NIC(네트워크 인터페이스 컨트롤러)
  - 디스크

- 상태에는 다음이 포함됩니다.
  - 가상 머신의 현재 상태
  - QEMU 게스트 에이전트가 가상 머신에 설치되어 있는지를 표시하는 참고 사항

- 사용률에는 다음에 대한 사용 데이터가 표시되는 차트가 포함됩니다.
  - CPU
  - 메모리
  - 파일 시스템
  - 네트워크 전송



참고

드롭다운 목록을 사용하여 사용률 데이터의 기간을 선택합니다. 사용 가능한 옵션은 1시간, 6시간, 24시간입니다.

- 이벤트에는 지난 1시간 동안의 가상 머신 활동에 대한 메시지가 나열됩니다. 추가 이벤트를

보려면 모두 보기를 클릭합니다.

13.6. 가상 머신 상태 모니터링

VMI(가상 머신 인스턴스)는 연결 해제, 교착 상태 또는 외부 종속성 문제와 같은 일시적인 문제로 인해 VMI가 비정상 상태가 될 수 있습니다. 상태 점검은 준비 및 활성 프로브의 조합을 사용하여 VMI에서 정기적으로 진단을 수행합니다.

13.6.1. 준비 및 활성 프로브 정보

준비 및 활성 프로브를 사용하여 비정상적인 VMI(가상 머신 인스턴스)를 탐지하고 처리합니다. VMI 사양에 프로브를 하나 이상 추가하여 트래픽이 준비되지 않은 VMI에 도달하지 않고 VMI가 응답하지 않을 때 새 인스턴스가 생성되도록 할 수 있습니다.

*준비 상태 프로브*는 VMI가 서비스 요청을 수락할 준비가 되었는지 확인합니다. 프로브가 실패하면 VMI가 준비될 때까지 VMI가 사용 가능한 엔드포인트 목록에서 VMI가 제거됩니다.

*활성 프로브*는 VMI의 응답 여부를 결정합니다. 프로브가 실패하면 VMI가 삭제되고 응답을 복원하기 위해 새 인스턴스가 생성됩니다.

VirtualMachineInstance 오브젝트의 spec.readinessProbe 및 spec.livenessProbe 필드를 설정하여 준비 및 활성 프로브를 구성할 수 있습니다. 이러한 필드는 다음 테스트를 지원합니다.

HTTP GET

프로브는 웹 후크를 사용하여 VMI의 상태를 결정합니다. HTTP 응답 코드가 200에서 399 사이인 경우 테스트에 성공합니다. HTTP GET 테스트는 HTTP 상태 코드를 완전히 초기화할 때 반환하는 애플리케이션에 사용할 수 있습니다.

TCP 소켓

프로브는 VMI에 대한 소켓을 열려고 시도합니다. VMI는 프로브에서 연결을 설정할 수 있는 경우에만 정상으로 간주됩니다. TCP 소켓 테스트는 초기화가 완료된 후 수신 대기를 시작하는 애플리케이션에 사용할 수 있습니다.

13.6.2. HTTP 준비 상태 프로브 정의

VMI(가상 머신 인스턴스) 구성의 spec.readinessProbe.httpGet 필드를 설정하여 HTTP 준비 프로브를 정의합니다.

절차

1. VMI 구성 파일에 준비 프로브의 세부 정보를 포함합니다.

HTTP GET 테스트가 있는 샘플 준비 상태 프로브

```
# ...
spec:
  readinessProbe:
    httpGet: 1
      port: 1500 2
      path: /healthz 3
      httpHeaders:
      - name: Custom-Header
        value: Awesome
    initialDelaySeconds: 120 4
    periodSeconds: 20 5
    timeoutSeconds: 10 6
    failureThreshold: 3 7
    successThreshold: 3 8
# ...
```

1 VMI 연결에 수행할 HTTP GET 요청입니다.

2 프로브에서 쿼리하는 VMI의 포트입니다. 위의 예에서 프로브는 포트 1500을 쿼리합니다.

3 HTTP 서버에서 액세스할 경로입니다. 위의 예에서 서버의 /healthz 경로에 대한 핸들러가 성공 코드를 반환하면 VMI가 정상으로 간주됩니다. 핸들러에서 실패 코드를 반환하면 VMI가 사용 가능한 엔드포인트 목록에서 제거됩니다.

4 준비 프로브가 시작되기 전에 VMI가 시작된 후 시간(초)입니다.

5

6

프로브가 시간 초과되고 VMI가 실패한 것으로 간주되는 비활성 시간(초)입니다. 기본값은 1입니다. 이 값은 periodSeconds 보다 작아야 합니다.

7

프로브가 실패할 수 있는 횟수입니다. 기본값은 3입니다. 지정된 횟수의 시도 후 Pod가 Unready로 표시됩니다.

8

프로브에서 실패 후 성공한 것으로 간주하기 위해 성공으로 보고해야 하는 횟수입니다. 기본값은 1입니다.

2. 다음 명령을 실행하여 VMI를 생성합니다.

```
$ oc create -f <file_name>.yaml
```

13.6.3. TCP 준비 프로브 정의

VMI(가상 머신 인스턴스) 구성의 spec.readinessProbe.tcpSocket 필드를 설정하여 TCP 준비 프로브를 정의합니다.

절차

1. VMI 구성 파일에 TCP 준비 프로브 세부 정보를 포함합니다.

TCP 소켓 테스트를 사용하는 샘플 준비 상태 프로브

```
...
spec:
  readinessProbe:
    initialDelaySeconds: 120 1
    periodSeconds: 20 2
    tcpSocket: 3
      port: 1500 4
    timeoutSeconds: 10 5
...
```

1

준비 프로브가 시작되기 전에 VMI가 시작된 후 시간(초)입니다.

2

프로브 수행 사이의 지연 시간(초)입니다. 기본 지연 시간은 10초입니다. 이 값은 timeoutSeconds 보다 커야 합니다.

3

수행할 TCP 작업입니다.

4

프로브에서 쿼리하는 VMI의 포트입니다.

5

프로브가 시간 초과되고 VMI가 실패한 것으로 간주되는 비활성 시간(초)입니다. 기본 값은 1입니다. 이 값은 periodSeconds 보다 작아야 합니다.

2. 다음 명령을 실행하여 VMI를 생성합니다.

```
$ oc create -f <file_name>.yaml
```

13.6.4. HTTP 활성 프로브 정의

VMI(가상 머신 인스턴스) 구성의 spec.livenessProbe.httpGet 필드를 설정하여 HTTP 활성 프로브를 정의합니다. 준비 프로브와 동일한 방식으로 활성 프로브에 대한 HTTP 및 TCP 테스트를 모두 정의할 수 있습니다. 이 절차에서는 HTTP GET 테스트를 사용하여 샘플 활성 프로브를 구성합니다.

절차

1. VMI 구성 파일에 HTTP 활성 프로브의 세부 정보를 포함합니다.

HTTP GET 테스트가 포함된 샘플 활성 프로브

```
# ...
spec:
  livenessProbe:
    initialDelaySeconds: 120 1
    periodSeconds: 20 2
    httpGet: 3
      port: 1500 4
      path: /healthz 5
      httpHeaders:
      - name: Custom-Header
        value: Awesome
    timeoutSeconds: 10 6
# ...
```

1

활성 프로브가 시작되기 전에 VMI가 시작된 후 시간(초)입니다.

2

프로브 수행 사이의 지연 시간(초)입니다. 기본 지연 시간은 10초입니다. 이 값은 timeoutSeconds 보다 커야 합니다.

3

VMI 연결에 수행할 HTTP GET 요청입니다.

4

프로브에서 쿼리하는 VMI의 포트입니다. 위의 예에서 프로브는 포트 1500을 쿼리합니다. VMI는 cloud-init를 통해 포트 1500에 최소 HTTP 서버를 설치하고 실행합니다.

5

HTTP 서버에서 액세스할 경로입니다. 위의 예에서 서버의 /healthz 경로에 대한 핸들러가 성공 코드를 반환하면 VMI가 정상으로 간주됩니다. 핸들러에서 실패 코드를 반환하면 VMI가 삭제되고 새 인스턴스가 생성됩니다.

6

프로브가 시간 초과되고 VMI가 실패한 것으로 간주되는 비활성 시간(초)입니다. 기본값은 1입니다. 이 값은 periodSeconds 보다 작아야 합니다.

2. 다음 명령을 실행하여 VMI를 생성합니다.

```
$ oc create -f <file_name>.yaml
```

13.6.5. 템플릿: 상태 점검을 정의하기 위한 가상 머신 구성 파일

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  labels:
    special: vm-fedora
  name: vm-fedora
spec:
  template:
    metadata:
      labels:
        special: vm-fedora
    spec:
      domain:
        devices:
          disks:
          - disk:
              bus: virtio
            name: containerdisk
          - disk:
              bus: virtio
            name: cloudinitdisk
        resources:
          requests:
            memory: 1024M
      readinessProbe:
        httpGet:
          port: 1500
        initialDelaySeconds: 120
        periodSeconds: 20
        timeoutSeconds: 10
        failureThreshold: 3
        successThreshold: 3
      terminationGracePeriodSeconds: 180
      volumes:
      - name: containerdisk
        containerDisk:
          image: kubevirt/fedora-cloud-registry-disk-demo
      - cloudInitNoCloud:
          userData: |-
            #cloud-config
            password: fedora
            chpasswd: { expire: False }
            bootcmd:
              - setenforce 0
              - dnf install -y nmap-ncat
```

```
        - systemd-run --unit=httpserver nc -klp 1500 -e '/usr/bin/echo -e HTTP/1.1 200
OK\\n\\nHello World!'
      name: cloudinitdisk
```

13.6.6. 추가 리소스

- 상태 점검을 사용하여 애플리케이션 상태 모니터링

13.7. OPENSHIFT CONTAINER PLATFORM 대시 보드를 사용하여 클러스터 정보 검색

OpenShift Container Platform 대시보드에는 클러스터에 대한 개괄적인 정보가 포함되어 있으며 OpenShift Container Platform 웹 콘솔에서 홈 > 대시보드 > 개요를 클릭하여 액세스합니다.

OpenShift Container Platform 대시보드는 별도의 대시보드 *카드*에 표시되는 다양한 클러스터 정보를 제공합니다.

13.7.1. OpenShift Container Platform 대시 보드 페이지 정보

OpenShift Container Platform 대시 보드는 다음 카드로 구성됩니다.

- Details는 클러스터 정보에 대한 간략한 개요를 표시합니다.

  상태에는 ok, error, warning, progress 및 unknown이 포함되어 있습니다. 리소스는 사용자 정의 상태 이름을 추가 할 수 있습니다.

  - 클러스터 ID
  - 공급자
  - 버전

- Cluster Inventory는 리소스 수 및 관련 상태를 정보를 표시합니다. 이러한 정보는 문제 해결에 개입이 필요한 경우 매우 유용하며 다음과 같은 관련 정보가 포함되어 있습니다.

- 노드 수
- Pod 수
- 영구 스토리지 볼륨 요청
- 가상 머신(OpenShift Virtualization이 설치된 경우 사용 가능)
- 상태에 따라 나열되는 클러스터의 베어 메탈 호스트 (metal3 환경에서만 사용 가능)

- 클러스터 상태에는 관련 경보 및 설명을 포함하여 클러스터의 현재 상태가 전체적으로 요약되어 있습니다. OpenShift Virtualization이 설치된 경우 OpenShift Virtualization의 전반적인 상태도 진단됩니다. 하위 시스템이 한 개 이상 있는 경우 모두 보기를 클릭하여 각 하위 시스템의 상태를 확인하십시오.

- Cluster Capacity 차트는 관리자가 클러스터에 추가 리소스가 필요한 시기를 파악하는 데 도움이 됩니다. 이 차트에는 내부 링과 외부링이 포함되어 있으며 내부 링은 현재 소비를 표시하는 외부 링은 다음 정보를 포함하여 리소스에 설정된 임계 값을 표시합니다.
  - CPU 시간
  - 메모리 할당
  - 소비된 스토리지
  - 소비된 네트워크 리소스

- Cluster Utilization은 관리자가 리소스의 높은 소비 규모 및 빈도를 이해하는데 도움이 되도록 지정된 기간 동안 다양한 리소스의 용량을 표시합니다.

- Events는 Pod 생성 또는 다른 호스트로의 가상 머신 마이그레이션과 같은 클러스터의 최근 활동과 관련된 메시지를 표시합니다.

- **Top Consumers** 관리자가 클러스터 리소스가 소비되는 방식을 이해하는 데 도움이 됩니다. 리소스를 클릭하면 지정된 클러스터 리소스(CPU, 메모리 또는 스토리지)를 가장 많이 사용하는 Pod와 노드가 나열된 세부 정보 페이지로 이동합니다.

13.8. 가상 머신의 리소스 사용량 검토

OpenShift Container Platform 웹 콘솔의 대시보드는 클러스터 상태를 빠르게 파악할 수 있도록 클러스터 메트릭에 대한 시각적 표현을 제공합니다. 대시보드는 핵심 플랫폼 구성 요소에 대한 모니터링을 제공하는 모니터링 개요 에 속합니다.

OpenShift Virtualization 대시보드는 가상 시스템 및 관련 pod의 리소스 사용에 대한 데이터를 제공합니다. **OpenShift Virtualization** 대시보드에 표시된 시각화 지표는 **Prometheus Query Language(PromQL)** 쿼리를 기반으로 합니다.

모니터링 역할은 **OpenShift Virtualization** 대시보드에서 사용자 정의 네임스페이스를 모니터링하는 데 필요합니다.

13.8.1. 상위 소비자 검토 정보

OpenShift Virtualization 대시보드에서는 특정 기간을 선택하고 해당 기간 내에 리소스의 상위 소비자를 볼 수 있습니다. 상위 소비자는 가장 많은 리소스를 소비하는 가상 시스템 또는 **virt-launcher Pod**입니다.

다음 표는 대시보드에서 모니터링된 리소스를 보여주고 상위 소비자에 대해 각 리소스와 연결된 메트릭을 설명합니다.

모니터링된 리소스	설명
메모리 스왑 트래픽	메모리를 스왑할 때 가장 많은 메모리를 소비하는 가상 머신입니다.
vCPU 대기 시간	vCPU에 대한 최대 대기 시간(초)이 발생하는 가상 머신입니다.
Pod별 CPU 사용량	대부분의 CPU를 사용하는 **virt-launcher** Pod입니다.
네트워크 트래픽	가장 많은 양의 네트워크 트래픽(바이트)을 수신하여 네트워크를 포화 상태로 만드는 가상 머신입니다.

스토리지 트래픽	스토리지 관련 트래픽(바이트)의 양이 가장 많은 가상 머신입니다.
스토리지 IOPS	일정 기간 동안 초당 I/O 작업이 가장 많은 가상 머신입니다.
메모리 사용량	가장 많은 메모리를 사용하는 **virt-launcher** Pod(바이트)입니다.

참고

최대 리소스 사용량은 상위 5개의 소비자로 제한됩니다.

13.8.2. 상위 소비자 검토

관리자 화면에서 리소스의 상위 소비자가 표시되는 OpenShift Virtualization 대시보드를 볼 수 있습니다.

사전 요구 사항

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

절차

1. **OpenShift Virtualization 웹 콘솔의 관리자 화면에서 모니터링 → 대시보드로 이동합니다.**
2. **대시보드 목록에서 KubeVirt/Infrastructure Resources/Top Consumers 대시보드를 선택합니다.**
3. **기간 드롭다운 메뉴에서 사전 정의된 기간을 선택합니다. 표의 상위 소비자에 대한 데이터를 검토할 수 있습니다.**
4. **선택 사항: 표의 상위 소비자와 연관된 PromQL(Prometheus Query Language) 쿼리를 보거나 편집하려면 검토를 클릭합니다.**

13.8.3. 추가 리소스

- 모니터링 개요
- 모니터링 대시보드 검토

13.9. OPENSHIFT CONTAINER PLATFORM 클러스터 모니터링, 로깅, TELEMETRY

OpenShift Container Platform은 클러스터 수준에서 모니터링에 필요한 다양한 리소스를 제공합니다.

13.9.1. OpenShift Container Platform 모니터링 정보

OpenShift Container Platform에는 주요 플랫폼 구성 요소를 모니터링할 수 있는 사전 구성, 사전 설치, 자체 업데이트 모니터링 스택이 포함되어 있습니다. OpenShift Container Platform은 즉시 사용 가능한 모니터링 모범 사례를 제공합니다. 클러스터 관리자에게 클러스터 문제에 대해 즉시 알리는 일련의 경보가 기본적으로 포함되어 있습니다. OpenShift Container Platform 웹 콘솔의 기본 대시보드에는 클러스터 상태를 빠르게 파악할 수 있도록 클러스터 메트릭에 대한 그래픽 표현이 포함되어 있습니다.

OpenShift Container Platform 4.10을 설치한 후 클러스터 관리자는 선택 옵션으로 사용자 정의 프로젝트에 대한 모니터링을 활성화할 수 있습니다. 이 기능을 사용하면 클러스터 관리자, 개발자, 기타 사용자가 자신의 프로젝트에서 서비스와 Pod를 모니터링하는 방법을 지정할 수 있습니다. 그런 다음 OpenShift Container Platform 웹 콘솔에서 자체 프로젝트에 대한 메트릭을 쿼리하고, 대시보드를 검토하고, 경보 규칙과 침묵을 관리할 수 있습니다.

참고

클러스터 관리자는 개발자 및 다른 사용자에게 자신의 프로젝트를 모니터링할 수 있는 권한을 부여할 수 있습니다. 권한은 사전 정의된 모니터링 역할 중 하나를 할당하는 방식으로 부여합니다.

13.9.2. 로깅 하위 시스템 구성 요소 정보

로깅 하위 시스템 구성 요소에는 OpenShift Container Platform 클러스터의 각 노드에 배포된 수집기가 포함되어 있습니다. 이 수집기는 모든 노드와 컨테이너 로그를 수집하여 로그 저장소에 씁니다. 중앙 집중식 웹 UI에서 이렇게 집계된 데이터를 사용하여 고급 시각화 및 대시보드를 생성할 수 있습니다.

로깅 하위 시스템의 주요 구성 요소는 다음과 같습니다.

-

수집 - 클러스터에서 로그를 수집하고 형식을 지정한 후 로그 저장소로 전달하는 구성 요소입니다. 최신 구현은 Fluentd입니다.

- 로그 저장소 - 로그가 저장되는 위치입니다. 기본 구현은 Elasticsearch입니다. 기본 Elasticsearch 로그 저장소를 사용하거나 외부 로그 저장소로 로그를 전달할 수 있습니다. 기본 로그 저장소는 테스트를 거쳐 단기 스토리지용으로 최적화되었습니다.

- 시각화 - 로그, 그래프, 차트 등을 보는 데 사용할 수 있는 UI 구성 요소입니다. 최신 구현은 Kibana입니다.

OpenShift 로깅에 대한 자세한 내용은 OpenShift 로깅 설명서를 참조하십시오.

13.9.3. Telemetry 정보

Telemetry는 엄선된 클러스터 모니터링 지표의 일부를 Red Hat으로 보냅니다. Telemeter Client는 4분 30초마다 메트릭 값을 가져와 Red Hat에 데이터를 업로드합니다. 이러한 메트릭에 대한 설명은 이 설명서에서 제공됩니다.

Red Hat은 이러한 데이터 스트림을 사용하여 클러스터를 실시간으로 모니터링하고 필요에 따라 고객에게 영향을 미치는 문제에 대응합니다. 또한 Red Hat은 OpenShift Container Platform 업그레이드를 고객에게 제공하여 서비스 영향을 최소화하고 지속적으로 업그레이드 환경을 개선할 수 있습니다.

이러한 디버깅 정보는 Red Hat 지원 및 엔지니어링 팀에 제공되며, 지원 사례를 통해 보고된 데이터에 액세스하는 것과 동일한 제한 사항이 적용됩니다. Red Hat은 연결된 모든 클러스터 정보를 사용하여 OpenShift Container Platform을 개선하고 사용 편의성을 높입니다.

13.9.3.1. Telemetry에서 수집하는 정보

Telemetry에서 수집되는 정보는 다음과 같습니다.

13.9.3.1.1. 시스템 정보

- OpenShift Container Platform 클러스터 버전 및 업데이트 버전 가용성 확인에 사용되는 업데이트 세부 정보와 같은 버전 정보

- 클러스터당 사용 가능한 업데이트 수, 업데이트 진행 정보, 업데이트 진행 정보에 사용되는 채널 및 이미지 리포지터리, 업데이트에 발생하는 오류 수를 포함한 업데이트 정보

- 설치 중 생성된 임의의 고유 식별자
- Red Hat 지원이 클라우드 인프라 수준, 호스트 이름, IP 주소, Kubernetes Pod 이름, 네임스페이스 및 서비스의 노드 구성을 포함하여 고객에게 유용한 지원을 제공하는 데 도움이 되는 구성 세부 정보
- 클러스터 및 해당 조건 및 상태에 설치된 OpenShift Container Platform 프레임워크 구성 요소
- 성능이 저하된 Operator에 대해 "관련 개체"로 나열된 모든 네임스페이스에 대한 이벤트
- 성능 저하 소프트웨어에 대한 정보
- 인증서의 유효성에 대한 정보
- OpenShift Container Platform이 배포된 공급자 플랫폼의 이름 및 데이터 센터 위치

13.9.3.1.2. 크기 조정 정보

- CPU 코어 수 및 각각에 사용된 RAM 용량을 포함한 클러스터, 시스템 유형 및 머신 크기에 대한 정보
- 클러스터에서 실행 중인 가상 머신 인스턴스의 수
- etcd 멤버 수 및 etcd 클러스터에 저장된 오브젝트 수
- 빌드 전략 유형별 애플리케이션 빌드 수

13.9.3.1.3. 사용 정보

- 구성 요소, 기능 및 확장에 대한 사용 정보
- 기술 프리뷰 및 지원되지 않는 구성에 대한 사용량 세부 정보

Telemetry는 사용자 이름 또는 암호와 같은 식별 정보를 수집하지 않습니다. Red Hat은 개인 정보를 수집하지 않습니다. 개인 정보가 의도하지 않게 Red Hat에 수신된 경우 Red Hat은 이러한 정보를 삭제합니다. Telemetry 데이터가 개인 정보를 구성하는 범위까지, Red Hat의 개인정보 보호정책에 대한 자세한 내용은 Red Hat 개인정보처리방침을 참조하십시오.

13.9.4. CLI 문제 해결 및 디버깅 명령

oc 클라이언트 문제 해결 및 디버깅 명령 목록은 OpenShift Container Platform CLI 툴 설명서를 참조하십시오.

13.10. 가상 리소스에 대한 PROMETHEUS 쿼리

OpenShift Virtualization은 클러스터에서 인프라 리소스를 사용하는 방식을 모니터링하기 위한 메트릭을 제공합니다. 메트릭은 다음 리소스를 다룹니다.

- vCPU
- 네트워크
- 스토리지
- 게스트 메모리 스왑

OpenShift Container Platform 모니터링 대시보드를 사용하여 가상화 메트릭을 쿼리합니다.

13.10.1. 사전 요구 사항

- vCPU 지표를 사용하려면 MachineConfig 오브젝트에 schedstats=enable 커널 인수를 적용해야 합니다. 이 커널 인수를 사용하면 디버깅 및 성능 튜닝에 사용되는 스케줄러 통계가 활성

화되고 스케줄러에 약간의 부하가 추가됩니다. 커널 인수 적용에 대한 자세한 내용은 OpenShift Container Platform 머신 구성 작업 설명서를 참조하십시오.

- 게스트 메모리 스와핑 쿼리가 데이터를 반환하려면 가상 게스트에서 메모리 스와핑을 활성화해야 합니다.

13.10.2. 메트릭 쿼리

OpenShift Container Platform 모니터링 대시보드를 사용하면 Pacemaker에서 표시되는 메트릭을 검사하기 위해 Prometheus Query Language(PromQL) 쿼리를 실행할 수 있습니다. 이 기능을 사용하면 클러스터 상태 및 모니터링 중인 모든 사용자 정의 워크로드에 대한 정보가 제공됩니다.

클러스터 관리자는 모든 핵심 OpenShift Container Platform 및 사용자 정의 프로젝트에 대한 메트릭을 쿼리할 수 있습니다.

개발자는 메트릭을 쿼리할 때 프로젝트 이름을 지정해야 합니다. 선택한 프로젝트의 메트릭을 확인하는 데 필요한 권한이 있어야 합니다.

13.10.2.1. 클러스터 관리자로서 모든 프로젝트의 메트릭 쿼리

클러스터 관리자 또는 모든 프로젝트에 대한 보기 권한이 있는 사용자로 Metrics UI에서 모든 기본 OpenShift Container Platform 및 사용자 정의 프로젝트에 대한 메트릭에 액세스할 수 있습니다.

참고

클러스터 관리자만 OpenShift Container Platform 모니터링을 통해 제공되는 타사 UI에 액세스할 수 있습니다.

사전 요구 사항

- cluster-admin 클러스터 역할 또는 모든 프로젝트에 대한 보기 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.

절차

1.

OpenShift Container Platform 웹 콘솔의 관리자 관점에서 **Observe → Metrics**를 선택합니다.

2. 커서에 표시기 삽입을 선택하여 사전 정의된 쿼리 목록을 확인합니다.

3. 사용자 정의 쿼리를 생성하려면 표현식 필드에 **PromQL(Prometheus Query Language)** 쿼리를 추가합니다.

4. 여러 쿼리를 추가하려면 쿼리 추가를 선택합니다.

5. 쿼리를 삭제하려면 쿼리 옆에 있는

를 선택한 다음 쿼리 삭제를 선택합니다.

6. 쿼리 실행을 비활성화하려면 쿼리 옆에 있는

를 선택하고 쿼리 비활성화를 선택합니다.

7. 생성된 쿼리를 실행하려면 쿼리 실행을 선택합니다. 쿼리의 메트릭은 플롯에 시각화됩니다. 쿼리가 유효하지 않으면 **UI**에 오류 메시지가 표시됩니다.

참고

대량의 데이터에서 작동하는 쿼리 시간이 초과되거나 시계열 그래프에 있을 때 브라우저가 과부하될 수 있습니다. 이를 방지하려면 그래프 숨기기를 선택하고 메트릭 테이블만 사용하여 쿼리를 조정합니다. 그런 다음 실행 가능한 쿼리를 검색한 후 플롯을 활성화하여 그래프를 그립니다.

8. 선택 사항: 이제 페이지 **URL**에 실행한 쿼리가 포함되어 있습니다. 나중에 이 쿼리 세트를 다시 사용하려면 이 **URL**을 저장합니다.

추가 리소스

- PromQL 쿼리 생성에 대한 자세한 내용은 Prometheus 쿼리 문서를 참조하십시오.

13.10.2.2. 개발자로 사용자 정의 프로젝트의 메트릭 쿼리

사용자 정의 프로젝트의 메트릭에 대해 개발자 또는 프로젝트에 대한 보기 권한이 있는 사용자로 액세스할 수 있습니다.

개발자 관점에서 Metrics UI에는 선택한 프로젝트에 대한 사전 정의된 CPU, 메모리, 대역폭 및 네트워크 패킷 쿼리가 포함되어 있습니다. 프로젝트에 대한 CPU, 메모리, 대역폭, 네트워크 패킷 및 애플리케이션 메트릭에 대해 사용자 정의 Prometheus Query Language(PromQL) 쿼리를 실행할 수도 있습니다.

참고

개발자는 관리자 관점이 아닌 개발자 관점만 사용할 수 있습니다. 개발자는 한 번에 하나의 프로젝트의 메트릭만 쿼리할 수 있습니다. 개발자는 코어 플랫폼 구성 요소에 대해 OpenShift Container Platform 모니터링을 통해 제공되는 타사 UI에 액세스할 수 없습니다. 대신 사용자 정의 프로젝트에 Metrics UI를 사용합니다.

사전 요구 사항

- 개발자로 또는 메트릭을 확인하는 프로젝트에 대한 보기 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.
- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- 사용자 정의 프로젝트에 서비스를 배포했습니다.
- 서비스에서 모니터링 방법을 정의하는 데 사용할 ServiceMonitor CRD(사용자 정의 리소스 정의(Custom Resource Definition))가 생성되었습니다.

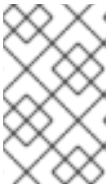
절차

1. OpenShift Container Platform 웹 콘솔의 개발자 관점에서 Observe → Metrics 를 선택합니다.

2.

2. **Project:** 목록에서 메트릭을 보려는 프로젝트를 선택합니다.

3. 쿼리 선택 목록에서 쿼리를 선택하거나 **PromQL** 표시를 선택하여 사용자 정의 **PromQL** 쿼리를 실행합니다.



참고

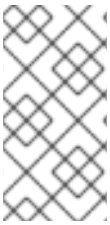
개발자 관점에서는 한 번에 하나의 쿼리만 실행할 수 있습니다.

추가 리소스

- **PromQL** 쿼리 생성에 대한 자세한 내용은 **Prometheus** 쿼리 문서를 참조하십시오.

13.10.3. 가상화 메트릭

다음 메트릭 설명에는 예제 **Prometheus Query Language(PromQL)** 쿼리가 포함됩니다. 이러한 메트릭은 **API**가 아니며 버전 간에 변경될 수 있습니다.



참고

다음 예제에서는 기간을 지정하는 **topk** 쿼리를 사용합니다. 해당 기간 동안 가상 머신을 삭제해도 쿼리 출력에 계속 표시될 수 있습니다.

13.10.3.1. vCPU 메트릭

다음 쿼리는 **I/O(**입력/출력**)** 대기 중인 가상 머신을 식별할 수 있습니다.

kubevirt_vmi_vcpu_wait_seconds

가상 시스템의 **vCPU**에 대해 대기 시간(초)을 반환합니다.

'0' 이상의 값은 **vCPU**가 실행하려고 하지만 호스트 스케줄러는 아직 실행할 수 없음을 의미합니다. 이러한 실행 불가능은 **I/O**에 문제가 있음을 나타냅니다.

참고

vCPU 지표를 쿼리하려면 먼저 MachineConfig 오브젝트에 schedstats=enable 커널 인수를 적용해야 합니다. 이 커널 인수를 사용하면 디버깅 및 성능 튜닝에 사용되는 스케줄러 통계가 활성화되고 스케줄러에 약간의 부하가 추가됩니다.

vCPU 대기 시간 쿼리의 예

```
topk(3, sum by (name, namespace) (rate(kubevirt_vmi_vcpu_wait_seconds[6m]))) > 0 1
```

1

이 쿼리는 6분 동안 매 순간에 I/O를 기다리는 상위 3개의 VM을 반환합니다.

13.10.3.2. 네트워크 메트릭

다음 쿼리는 네트워크를 포화 상태로 만드는 가상 머신을 식별할 수 있습니다.

kubevirt_vmi_network_receive_bytes_total

가상 시스템의 네트워크에서 수신된 총 트래픽(바이트 단위)을 반환합니다.

kubevirt_vmi_network_transmit_bytes_total

가상 시스템의 네트워크에서 전송된 총 트래픽(바이트 단위)을 반환합니다.

네트워크 트래픽 쿼리의 예

```
topk(3, sum by (name, namespace) (rate(kubevirt_vmi_network_receive_bytes_total[6m])) +
sum by (name, namespace) (rate(kubevirt_vmi_network_transmit_bytes_total[6m]))) > 0 1
```

1

이 쿼리는 6분 동안 매번 가장 많은 네트워크 트래픽을 전송하는 상위 3 개의 VM을 반환합니다.

13.10.3.3. 스토리지 메트릭

13.10.3.3.1. 스토리지 관련 트래픽

다음 쿼리는 대량의 데이터를 작성하는 VM을 식별할 수 있습니다.

kubevirt_vmi_storage_read_traffic_bytes_total

가상 머신의 스토리지 관련 트래픽의 총 양(바이트)을 반환합니다.

kubevirt_vmi_storage_write_traffic_bytes_total

가상 시스템의 스토리지 관련 트래픽의 총 스토리지 쓰기(바이트)를 반환합니다.

스토리지 관련 트래픽 쿼리의 예

```
topk(3, sum by (name, namespace) (rate(kubevirt_vmi_storage_read_traffic_bytes_total[6m]))
+ sum by (name, namespace) (rate(kubevirt_vmi_storage_write_traffic_bytes_total[6m]))) > 0
1
```

1

이 쿼리는 6분 동안 매번 가장 많은 스토리지 트래픽을 수행하는 상위 3 개의 VM을 반환합니다.

13.10.3.3.2. I/O 성능

다음 쿼리는 스토리지 장치의 I/O 성능을 확인할 수 있습니다.

kubevirt_vmi_storage_iops_read_total

가상 시스템이 초당 수행하는 쓰기 I/O 작업 양을 반환합니다.

kubevirt_vmi_storage_iops_write_total

가상 시스템이 초당 수행하는 읽기 I/O 작업의 양을 반환합니다.

I/O 성능 쿼리의 예

```
topk(3, sum by (name, namespace) (rate(kubevirt_vmi_storage_iops_read_total[6m])) + sum
by (name, namespace) (rate(kubevirt_vmi_storage_iops_write_total[6m]))) > 0 1
```

1

이 쿼리는 6분 동안 매 순간 초당 최대 I/O 작업을 수행하는 상위 3 개의 VM을 반환합니다.

13.10.3.4. 게스트 메모리 스왑 메트릭

다음 쿼리는 메모리 스와핑을 가장 많이 수행하는 스왑 사용 게스트를 식별할 수 있습니다.

kubevirt_vmi_memory_swap_in_traffic_bytes_total

가상 게스트가 스와핑하는 메모리의 총 양(바이트 단위)을 반환합니다.

kubevirt_vmi_memory_swap_out_traffic_bytes_total

가상 게스트가 스왑 아웃하는 메모리의 총 양(바이트 단위)을 반환합니다.

메모리 스와핑 쿼리의 예

```
topk(3, sum by (name, namespace)
(rate(kubevirt_vmi_memory_swap_in_traffic_bytes_total[6m])) + sum by (name, namespace)
(rate(kubevirt_vmi_memory_swap_out_traffic_bytes_total[6m]))) > 0 1
```

1

이 쿼리는 6분 동안 매 순간 게스트가 가장 많은 메모리 스왑을 수행하는 상위 3개의 VM을 반환합니다.

참고

메모리 스와핑은 가상 시스템이 메모리 부족 상태에 있음을 나타냅니다. 가상 시스템의 메모리 할당을 늘리면 이 문제가 완화될 수 있습니다.

13.10.4. 추가 리소스

- 모니터링 개요

13.11. 가상 머신에 대한 사용자 정의 메트릭 노출

OpenShift Container Platform에는 주요 플랫폼 구성 요소를 모니터링할 수 있는 사전 구성, 사전 설치 및 자체 업데이트 모니터링 스택이 포함되어 있습니다. 이 모니터링 스택은 **Prometheus** 모니터링 시스템을 기반으로 합니다. **Prometheus**는 시계열 데이터베이스이며 메트릭에 대한 규칙 평가 엔진입니다.

OpenShift Container Platform 모니터링 스택을 사용하는 것 외에도 **CLI**를 사용하여 사용자 정의 프로젝트에 대한 모니터링을 활성화하고 **node-exporter** 서비스를 통해 가상 머신에 대해 노출되는 사용자 정의 지표를 쿼리할 수 있습니다.

13.11.1. 노드 내보내기 서비스 구성

node-exporter 에이전트는 메트릭을 수집하려는 클러스터의 모든 가상 머신에 배포됩니다. **node-exporter** 에이전트를 서비스로 구성하여 가상 머신과 연결된 내부 지표 및 프로세스를 노출합니다.

사전 요구 사항

- **OpenShift Container Platform CLI oc**를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 클러스터에 로그인합니다.
- **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 생성합니다.
- **enableUserWorkload** 를 **true** 로 설정하여 **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config ConfigMap** 오브젝트를 구성합니다.

절차

1. Service YAML 파일을 생성합니다. 다음 예에서 파일은 node-exporter-service.yaml 라고 합니다.

```
kind: Service
apiVersion: v1
metadata:
 name: node-exporter-service 1
 namespace: dynamation 2
 labels:
  servicetype: metrics 3
spec:
 ports:
  - name: exmet 4
    protocol: TCP
    port: 9100 5
    targetPort: 9100 6
 type: ClusterIP
 selector:
  monitor: metrics 7
```

1. 가상 머신에서 지표를 노출하는 node-exporter 서비스입니다.

2. 서비스가 생성된 네임스페이스입니다.

3. 서비스의 레이블입니다. ServiceMonitor 는 이 라벨을 사용하여 이 서비스와 일치시킵니다.

4. ClusterIP 서비스의 포트 9100에서 지표를 노출하는 포트에 제공되는 이름입니다.

5. node-exporter-service 에서 요청을 수신하는 데 사용하는 대상 포트입니다.

6. monitor 레이블로 구성된 가상 머신의 TCP 포트 번호입니다.

7

가상 머신의 **Pod**와 일치시키는 데 사용되는 라벨입니다. 이 예에서는 라벨 **monitor** 및 지표 값이 있는 모든 가상 머신의 **Pod**가 일치합니다.

2. **node-exporter** 서비스를 생성합니다.

```
$ oc create -f node-exporter-service.yaml
```

13.11.2. 노드 내보내기 서비스를 사용하여 가상 머신 구성

의 **node-exporter** 파일을 가상 머신에 다운로드합니다. 그런 다음 가상 시스템이 부팅될 때 **node-exporter** 서비스를 실행하는 **systemd** 서비스를 생성합니다.

사전 요구 사항

- 구성 요소의 **Pod**가 **openshift-user-workload-monitoring** 프로젝트에서 실행됩니다.
- 이 사용자 정의 프로젝트를 모니터링해야 하는 사용자에게 **monitoring-edit** 역할을 부여합니다.

절차

1. 가상 머신에 로그인합니다.
2. **node-exporter** 파일에 적용되는 디렉터리 경로를 사용하여 의 **node-exporter** 파일을 가상 머신에 다운로드합니다.

```
$ wget
https://github.com/prometheus/node_exporter/releases/download/v1.3.1/node_exporte
r-1.3.1.linux-amd64.tar.gz
```

3. 실행 파일을 추출하여 **/usr/bin** 디렉터리에 배치합니다.

```
$ sudo tar xvf node_exporter-1.3.1.linux-amd64.tar.gz \
  --directory /usr/bin --strip 1 "*/node_exporter"
```

4.

이 디렉터리 경로에 **node_exporter.service** 파일을 만듭니다. **/etc/systemd/system.** 이 **systemd** 서비스 파일은 가상 머신이 재부팅될 때 **node-exporter** 서비스를 실행합니다.

```
[Unit]
Description=Prometheus Metrics Exporter
After=network.target
StartLimitIntervalSec=0

[Service]
Type=simple
Restart=always
RestartSec=1
User=root
ExecStart=/usr/bin/node_exporter

[Install]
WantedBy=multi-user.target
```

5. **systemd** 서비스를 활성화하고 시작합니다.

```
$ sudo systemctl enable node_exporter.service
$ sudo systemctl start node_exporter.service
```

검증

- **node-exporter** 에이전트에서 가상 머신의 지표를 보고하는지 확인합니다.

```
$ curl http://localhost:9100/metrics
```

출력 예

```
go_gc_duration_seconds{quantile="0"} 1.5244e-05
go_gc_duration_seconds{quantile="0.25"} 3.0449e-05
go_gc_duration_seconds{quantile="0.5"} 3.7913e-05
```

13.11.3. 가상 머신의 사용자 정의 모니터링 라벨 생성

단일 서비스에서 여러 가상 머신에 대한 쿼리를 활성화하려면 가상 머신의 **YAML** 파일에 사용자 지정 라벨을 추가합니다.

사전 요구 사항

- OpenShift Container Platform CLI oc를 설치합니다.
- cluster-admin 권한이 있는 사용자로 로그인합니다.
- 웹 콘솔에 액세스하여 가상 머신을 중지하고 다시 시작합니다.

절차

1. 가상 머신 구성 파일의 템플릿 사양을 편집합니다. 이 예에서 라벨 모니터에 는 값 메트릭 이 있습니다.

```
spec:
  template:
    metadata:
      labels:
        monitor: metrics
```

2. 가상 머신을 중지하고 재시작하여 monitor 레이블에 지정된 라벨 이름으로 새 Pod를 생성합니다.

13.11.3.1. 메트릭의 node-exporter 서비스 쿼리

/metrics 표준 이름 아래에 HTTP 서비스 끝점을 통해 가상 머신에 메트릭이 노출됩니다. 지표를 쿼리할 때 Prometheus는 가상 머신에서 노출하는 지표 끝점에서 메트릭을 직접 스크랩하고 볼 수 있는 이러한 지표를 제공합니다.

사전 요구 사항

- cluster-admin 권한 또는 monitoring-edit 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- node-exporter 서비스를 구성하여 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.

절차

1. 서비스의 네임스페이스를 지정하여 **HTTP** 서비스 끝점을 가져옵니다.

```
$ oc get service -n <namespace> <node-exporter-service>
```

2. **node-exporter** 서비스에 사용 가능한 모든 메트릭을 나열하려면 지표 리소스를 쿼리합니다.

```
$ curl http://<172.30.226.162:9100>/metrics | grep -vE "^#|^$"
```

출력 예

```
node_arp_entries{device="eth0"} 1
node_boot_time_seconds 1.643153218e+09
node_context_switches_total 4.4938158e+07
node_cooling_device_cur_state{name="0",type="Processor"} 0
node_cooling_device_max_state{name="0",type="Processor"} 0
node_cpu_guest_seconds_total{cpu="0",mode="nice"} 0
node_cpu_guest_seconds_total{cpu="0",mode="user"} 0
node_cpu_seconds_total{cpu="0",mode="idle"} 1.10586485e+06
node_cpu_seconds_total{cpu="0",mode="iowait"} 37.61
node_cpu_seconds_total{cpu="0",mode="irq"} 233.91
node_cpu_seconds_total{cpu="0",mode="nice"} 551.47
node_cpu_seconds_total{cpu="0",mode="softirq"} 87.3
node_cpu_seconds_total{cpu="0",mode="steal"} 86.12
node_cpu_seconds_total{cpu="0",mode="system"} 464.15
node_cpu_seconds_total{cpu="0",mode="user"} 1075.2
node_disk_discard_time_seconds_total{device="vda"} 0
node_disk_discard_time_seconds_total{device="vdb"} 0
node_disk_discarded_sectors_total{device="vda"} 0
node_disk_discarded_sectors_total{device="vdb"} 0
node_disk_discards_completed_total{device="vda"} 0
node_disk_discards_completed_total{device="vdb"} 0
node_disk_discards_merged_total{device="vda"} 0
node_disk_discards_merged_total{device="vdb"} 0
node_disk_info{device="vda",major="252",minor="0"} 1
node_disk_info{device="vdb",major="252",minor="16"} 1
node_disk_io_now{device="vda"} 0
node_disk_io_now{device="vdb"} 0
node_disk_io_time_seconds_total{device="vda"} 174
node_disk_io_time_seconds_total{device="vdb"} 0.054
node_disk_io_time_weighted_seconds_total{device="vda"} 259.79200000000003
node_disk_io_time_weighted_seconds_total{device="vdb"} 0.039
node_disk_read_bytes_total{device="vda"} 3.71867136e+08
node_disk_read_bytes_total{device="vdb"} 366592
node_disk_read_time_seconds_total{device="vda"} 19.128
node_disk_read_time_seconds_total{device="vdb"} 0.039
node_disk_reads_completed_total{device="vda"} 5619
node_disk_reads_completed_total{device="vdb"} 96
node_disk_reads_merged_total{device="vda"} 5
```

```
node_disk_reads_merged_total{device="vdb"} 0
node_disk_write_time_seconds_total{device="vda"} 240.66400000000002
node_disk_write_time_seconds_total{device="vdb"} 0
node_disk_writes_completed_total{device="vda"} 71584
node_disk_writes_completed_total{device="vdb"} 0
node_disk_writes_merged_total{device="vda"} 19761
node_disk_writes_merged_total{device="vdb"} 0
node_disk_written_bytes_total{device="vda"} 2.007924224e+09
node_disk_written_bytes_total{device="vdb"} 0
```

13.11.4. 노드 내보내기 서비스에 대한 ServiceMonitor 리소스 생성

Prometheus 클라이언트 라이브러리 및 **/metrics** 끝점에서 메트릭을 스크랩하여 **node-exporter** 서비스에서 노출하는 메트릭에 액세스하고 확인할 수 있습니다. **ServiceMonitor CRD**(사용자 정의 리소스 정의)를 사용하여 노드 내보내기 서비스를 모니터링합니다.

사전 요구 사항

- **cluster-admin** 권한 또는 **monitoring-edit** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- **node-exporter** 서비스를 구성하여 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.

절차

1. **ServiceMonitor** 리소스 구성에 대한 **YAML** 파일을 생성합니다. 이 예에서 서비스 모니터는 레이블 지표 와 모든 서비스와 일치하며 **30**초마다 **exmet** 포트를 쿼리합니다.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    k8s-app: node-exporter-metrics-monitor
  name: node-exporter-metrics-monitor (1)
  namespace: dynamation (2)
spec:
  endpoints:
  - interval: 30s (3)
    port: exmet (4)
    scheme: http
```

```
  selector:
    matchLabels:
      servicetype: metrics
```

1

2

ServiceMonitor 가 생성되는 네임스페이스입니다.

3

포트를 쿼리할 간격입니다.

4

30초마다 쿼리되는 포트의 이름입니다.

2. node-exporter 서비스에 대한 ServiceMonitor 구성을 생성합니다.

```
$ oc create -f node-exporter-metrics-monitor.yaml
```

13.11.4.1. 클러스터 외부의 노드 내보내기 서비스에 액세스

클러스터 외부에서 node-exporter 서비스에 액세스하고 노출된 지표를 볼 수 있습니다.

사전 요구 사항

- cluster-admin 권한 또는 monitoring-edit 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- node-exporter 서비스를 구성하여 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.

절차

1. node-exporter 서비스를 노출합니다.

```
$ oc expose service -n <namespace> <node_exporter_service_name>
```

2. 경로의 **FQDN**(완전화된 도메인 이름)을 가져옵니다.

```
$ oc get route -o=custom-columns=NAME:.metadata.name,DNS:.spec.host
```

출력 예

```
NAME                 DNS
node-exporter-service   node-exporter-service-dynamation.apps.cluster.example.org
```

3. **curl** 명령을 사용하여 **node-exporter** 서비스에 대한 지표를 표시합니다.

```
$ curl -s http://node-exporter-service-dynamation.apps.cluster.example.org/metrics
```

출력 예

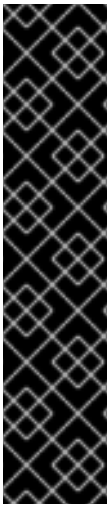
```
go_gc_duration_seconds{quantile="0"} 1.5382e-05
go_gc_duration_seconds{quantile="0.25"} 3.1163e-05
go_gc_duration_seconds{quantile="0.5"} 3.8546e-05
go_gc_duration_seconds{quantile="0.75"} 4.9139e-05
go_gc_duration_seconds{quantile="1"} 0.000189423
```

13.11.5. 추가 리소스

- 모니터링 스택 구성
- 사용자 정의 프로젝트 모니터링 활성화
- 메트릭 관리

- 모니터링 대시보드 검토
- 상태 점검을 사용하여 애플리케이션 상태 모니터링
- 구성 맵 생성 및 사용
- 가상 머신 상태 제어

13.12. OPENSHIFT VIRTUALIZATION 중요 경고



중요

OpenShift Virtualization 중요 경고는 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.

OpenShift Virtualization에는 문제가 발생하는 경우를 알려주는 경고가 있습니다. 심각한 경고에는 즉각적인 주의가 필요합니다.

각 경고에는 문제에 대한 해당 설명, 경고 발생 이유, 문제 해결 프로세스 및 경고 해결 단계가 있습니다.

13.12.1. 네트워크 경고

네트워크 경고는 **OpenShift Virtualization Network Operator**의 문제에 대한 정보를 제공합니다.

13.12.1.1. KubeMacPoolDown alert

설명

KubeMacPool 구성 요소는 MAC 주소를 할당하고 MAC 주소 충돌을 방지합니다.

이유

KubeMacPool-manager Pod가 다운되면 VirtualMachine 오브젝트 생성이 실패합니다.

문제 해결

1. Kubemacpool-manager Pod 네임스페이스 및 이름을 확인합니다.

```
$ export KMP_NAMESPACE="$(oc get pod -A --no-headers -l control-plane=mac-controller-manager | awk '{print $1}')"
```

```
$ export KMP_NAME="$(oc get pod -A --no-headers -l control-plane=mac-controller-manager | awk '{print $2}')"
```

2. Kubemacpool-manager Pod 설명 및 로그를 확인하여 문제의 원인을 확인합니다.

```
$ oc describe pod -n $KMP_NAMESPACE $KMP_NAME
```

```
$ oc logs -n $KMP_NAMESPACE $KMP_NAME
```

해결

지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.2. SSP 경고

SSP 경고는 OpenShift Virtualization SSP Operator의 문제에 대한 정보를 제공합니다.

13.12.2.1. SSPFailingToReconcile alert

설명

SSP Operator의 Pod는 시작되었지만 Pod의 조정 주기가 일관되게 실패합니다. 이 오류에는 템플릿 유효성 검사기를 배포하지 못하거나 공통 템플릿을 배포하지 못하는 리소스 업데이트 실패가 포함됩니다.

이유

SSP Operator를 조정하지 못하면 종속 구성 요소 배포 실패, 구성 요소 변경 사항 조정에 실패하거나 둘 다 실패합니다. 또한 공통 템플릿 및 템플릿 유효성 검증기를 재설정하여 업데이트합니다.

문제 해결

1. ssp-operator Pod의 로그에 오류가 있는지 확인합니다.

```
$ export NAMESPACE="$(oc get deployment -A | grep ssp-operator | awk '{print $1}')"
```

```
$ oc -n $NAMESPACE describe pods -l control-plane=ssp-operator
```

```
$ oc -n $NAMESPACE logs --tail=-1 -l control-plane=ssp-operator
```

2. 템플릿 유효성 검증기가 실행 중인지 확인합니다. 템플릿 유효성 검증기가 작동하지 않으면 Pod의 로그에 오류가 있는지 확인합니다.

```
$ export NAMESPACE="$($ oc get deployment -A | grep ssp-operator | awk '{print $1}')"
```

```
$ oc -n $NAMESPACE get pods -l name=virt-template-validator
```

```
$ oc -n $NAMESPACE describe pods -l name=virt-template-validator
```

```
$ oc -n $NAMESPACE logs --tail=-1 -l name=virt-template-validator
```

해결

지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.2.2. SSPOperatorDown 경고

설명

SSP Operator는 공통 템플릿과 템플릿 유효성 검증기를 배포하고 조정합니다.

이유

SSP Operator가 다운되면 종속 구성 요소 배포 실패, 구성 요소 변경 사항 조정에 실패하거나 둘 다 실패합니다. 또한 공통 템플릿 및 템플릿 유효성 검증기를 재설정하여 업데이트합니다.

문제 해결

1. ssp-operator의 Pod 네임스페이스를 확인합니다.

```
$ export NAMESPACE="$(oc get deployment -A | grep ssp-operator | awk '{print $1}')"
```

2. ssp-operator의 Pod가 현재 다운되었는지 확인합니다.

```
$ oc -n $NAMESPACE get pods -l control-plane=ssp-operator
```

3. ssp-operator의 Pod 설명 및 로그를 확인합니다.

```
$ oc -n $NAMESPACE describe pods -l control-plane=ssp-operator
```

```
$ oc -n $NAMESPACE logs --tail=-1 -l control-plane=ssp-operator
```

해결

지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.2.3. SSPTemplateValidatorDown alert

설명

템플릿 유효성 검증기를 통해 VM(가상 머신)이 할당된 템플릿을 위반하지 않는지 확인합니다.

이유

모든 템플릿 유효성 검사기 Pod가 다운되면 템플릿 유효성 검사기가 할당된 템플릿에 대해 VM의 유효성을 검사하지 못합니다.

문제 해결

1. ssp-operator Pod 및 virt-template-validator Pod의 네임스페이스를 확인합니다.

```
$ export NAMESPACE_SSP="$(oc get deployment -A | grep ssp-operator | awk '{print $1}')"
```

```
$ export NAMESPACE="$(oc get deployment -A | grep virt-template-validator | awk '{print $1}')"
```

2. virt-template-validator의 Pod가 현재 종료되었는지 확인합니다.

```
$ oc -n $NAMESPACE get pods -l name=virt-template-validator
```

3. Pod 설명 및 ssp-operator 및 virt-template-validator의 로그를 확인합니다.

```
$ oc -n $NAMESPACE_SSP describe pods -l name=ssp-operator
```

```
$ oc -n $NAMESPACE_SSP logs --tail=-1 -l name=ssp-operator
```

```
$ oc -n $NAMESPACE describe pods -l name=virt-template-validator
```

```
$ oc -n $NAMESPACE logs --tail=-1 -l name=virt-template-validator
```

해결

지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.3. 가상화 경고

가상화 경고는 OpenShift Virtualization Virt Operator의 문제에 대한 정보를 제공합니다.

13.12.3.1. NoLeadingVirtOperator alert

설명

지난 10분 동안 하나 이상의 virt-operator Pod가 Ready 상태인 경우에도 virt-operator Pod에 리더 리스가 포함되지 않습니다. 경고는 작동 중인 virt-operator Pod가 없음을 나타냅니다.

이유

virt-operator는 OpenShift Container Platform 클러스터에서 활성화된 첫 번째 Kubernetes Operator입니다. 주요 기능은 다음과 같습니다.

- 설치
- Live-update

- 클러스터의 실시간 업그레이드
- virt-controller, virt-handler, virt-launcher와 같은 최상위 컨트롤러의 라이프사이클 모니터링
- 최상위 컨트롤러 조정 관리

또한 virt-operator는 인증서 교체 및 일부 인프라 관리와 같은 클러스터 전체 작업을 담당합니다.

virt-operator 배포에는 운영 virt-operator Pod를 나타내는 리더 리스가 한 명의 리더 Pod가 있는 두 개의 Pod의 기본 복제본이 있습니다.

이 경고는 클러스터 수준에서 오류가 있음을 나타냅니다. 컨트롤러의 인증 교체, 업그레이드 및 조정과 같은 중요한 클러스터 전체의 관리 기능을 일시적으로 사용할 수 없을 수 있습니다.

문제 해결

Pod 로그에서 virt-operator Pod의 리더 상태를 확인합니다. Started leading 및 acquire leader 가 포함된 로그 메시지는 지정된 virt-operator Pod의 리더 상태를 나타냅니다.

또한 다음 명령을 사용하여 실행 중인 virt-operator Pod 및 Pod의 상태가 있는지 항상 확인합니다.

```
$ export NAMESPACE="$(oc get kubevirt -A -o custom-columns="":.metadata.namespace)"
$ oc -n $NAMESPACE get pods -l kubevirt.io=virt-operator
$ oc -n $NAMESPACE logs <pod-name>
$ oc -n $NAMESPACE describe pod <pod-name>
```

리더 Pod 예:

```
$ oc -n $NAMESPACE logs <pod-name> |grep lead
```

출력 예

```
{"component":"virt-operator","level":"info","msg":"Attempting to acquire leader
status","pos":"application.go:400","timestamp":"2021-11-30T12:15:18.635387Z"}
I1130 12:15:18.635452       1 leaderelection.go:243] attempting to acquire leader lease
<namespace>/virt-operator...
I1130 12:15:19.216582       1 leaderelection.go:253] successfully acquired lease
<namespace>/virt-operator
```

```
{"component":"virt-operator","level":"info","msg":"Started
leading","pos":"application.go:385","timestamp":"2021-11-30T12:15:19.216836Z"}
```

비leader Pod의 예:

```
$ oc -n $NAMESPACE logs <pod-name> |grep lead
```

출력 예

```
{"component":"virt-operator","level":"info","msg":"Attempting to acquire leader
status","pos":"application.go:400","timestamp":"2021-11-30T12:15:20.533696Z"}
I1130 12:15:20.533792       1 leaderelection.go:243] attempting to acquire leader lease
<namespace>/virt-operator...
```

해결

하나 이상의 **virt-operator Pod**가 **Ready** 상태에 있어도 **virt-operator Pod**가 리더 리스를 유지하지 않는 데는 몇 가지 이유가 있습니다. 근본 원인을 식별하고 적절한 조치를 취해야 합니다.

또는 지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.3.2. NoReadyVirtController alert

설명

virt-controller는 **VMI**(가상 머신 인스턴스)를 모니터링합니다. **virt-controller**는 **VMI** 오브젝트와 연결된 **Pod**의 라이프사이클을 생성하고 관리하여 관련 **Pod**도 관리합니다.

VMI 오브젝트는 항상 라이프사이클 동안 Pod와 연결합니다. 그러나 VMI 마이그레이션으로 인해 Pod 인스턴스가 시간이 지남에 따라 변경될 수 있습니다.

이 경고는 5분 동안 준비되지 않은 virt-controllers를 탐지할 때 발생합니다.

이유

virt-controller가 실패하면 VM 라이프사이클 관리가 완전히 실패합니다. 라이프사이클 관리 작업에는 새 VMI를 시작하거나 기존 VMI 종료가 포함됩니다.

문제 해결

1. 사용 가능한 복제본 및 조건에 대해 virt-controller의 vdeployment 상태를 확인합니다.

```
$ oc -n $NAMESPACE get deployment virt-controller -o yaml
```

2. virt-controller Pod가 있는지 확인하고 상태를 확인합니다.

```
get pods -n $NAMESPACE |grep virt-controller
```

3. virt-controller Pod의 이벤트를 확인합니다.

```
$ oc -n $NAMESPACE describe pods <virt-controller pod>
```

4. virt-controller Pod의 로그를 확인합니다.

```
$ oc -n $NAMESPACE logs <virt-controller pod>
```

5. 노드가 NotReady 상태에 있는 경우와 같이 노드에 문제가 있는지 확인합니다.

```
$ oc get nodes
```

해결

virt-controller Pod가 Ready 상태가 아닌 이유는 여러 가지가 있습니다. 근본 원인을 식별하고 적절한 조치를 취해야 합니다.

또는 지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.3.3. NoReadyVirtOperator alert

설명

지난 10분 동안 Ready 상태에서 virt-operator Pod를 탐지할 수 없습니다. virt-operator 배포에는 두 개의 Pod의 기본 복제본이 있습니다.

이유

virt-operator는 OpenShift Container Platform 클러스터에서 활성화된 첫 번째 Kubernetes Operator입니다. 주요 기능은 다음과 같습니다.

- 설치
- Live-update
- 클러스터의 실시간 업그레이드
- virt-controller, virt-handler, virt-launcher와 같은 최상위 컨트롤러의 라이프사이클 모니터링
- 최상위 컨트롤러 조정 관리

또한 virt-operator는 인증서 교체 및 일부 인프라 관리와 같은 클러스터 전체 작업을 담당합니다.

참고

virt-operator는 클러스터의 가상 머신을 직접 처리하지 않습니다. virt-operator의 고가용성은 사용자 정의 워크로드에 영향을 미치지 않습니다.

이 경고는 클러스터 수준에서 오류가 있음을 나타냅니다. 컨트롤러의 인증 교체, 업그레이드 및 조정

과 같은 중요한 클러스터 전체의 관리 기능을 일시적으로 사용할 수 없습니다.

문제 해결

1. 사용 가능한 복제본 및 조건이 있는지 **virt-operator**의 배포 상태를 확인합니다.

```
$ oc -n $NAMESPACE get deployment virt-operator -o yaml
```

2. **virt-controller Pod**의 이벤트를 확인합니다.

```
$ oc -n $NAMESPACE describe pods <virt-operator pod>
```

3. **virt-operator Pod**의 로그를 확인합니다.

```
$ oc -n $NAMESPACE logs <virt-operator pod>
```

4. **NotReady** 상태에 있는 경우와 같이 컨트롤 플레인 및 마스터의 노드에 문제가 있는지 확인합니다.

```
$ oc get nodes
```

해결

virt-operator Pod가 **Ready** 상태가 아닌 이유는 여러 가지가 있습니다. 근본 원인을 식별하고 적절한 조치를 취해야 합니다.

또는 지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.3.4. VirtAPIDown 경고

설명

모든 **OpenShift Container Platform API** 서버가 다운되었습니다.

이유

모든 **OpenShift Container Platform API** 서버가 다운되면 **OpenShift Container Platform** 엔티티에 대한 **API** 호출이 발생하지 않습니다.

문제 해결

1. 환경 변수 **NAMESPACE** 를 수정합니다.

```
$ export NAMESPACE="$(oc get kubevirt -A -o custom-
columns="":.metadata.namespace)"
```

2. 실행 중인 **virt-api Pod**가 있는지 확인합니다.

```
$ oc -n $NAMESPACE get pods -l kubevirt.io=virt-api
```

3. **oc logs** 및 **oc describe** 를 사용하여 **Pod**의 상태를 확인합니다.

4. **virt-api** 배포 상태를 확인합니다. 이러한 명령을 사용하여 관련 이벤트에 대해 알아보고 이미지 가져오기, 충돌 **Pod** 또는 기타 유사한 문제에 문제가 있는지 확인합니다.

```
$ oc -n $NAMESPACE get deployment virt-api -o yaml
```

```
$ oc -n $NAMESPACE describe deployment virt-api
```

5. 노드가 **NotReady** 상태에 있는 경우와 같이 노드에 문제가 있는지 확인합니다.

```
$ oc get nodes
```

해결

다음과 같은 몇 가지 이유로 **virt-api Pod**가 다운될 수 있습니다. 근본 원인을 식별하고 적절한 조치를 취해야 합니다.

또는 지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.3.5. VirtApiRESTErrorsBurst alert

설명

REST 호출 중 **80%** 이상이 지난 **5**분 동안 **virt-api**에서 실패합니다.

이유

virt-api에 대한 **REST** 호출의 매우 높은 속도가 높으면 응답 속도가 느리거나, **API** 호출 속도가 느려지거나 **API** 호출을 완전히 해제할 수 있습니다.

문제 해결

1. 환경 변수 **NAMESPACE** 를 수정합니다.

```
$ export NAMESPACE="$(oc get kubevirt -A -o custom-
columns="":.metadata.namespace)"
```

2. 실행 중인 **virt-api Pod** 수를 확인합니다.

```
$ oc -n $NAMESPACE get pods -l kubevirt.io=virt-api
```

3. **oc logs** 및 **oc describe** 를 사용하여 **Pod**의 상태를 확인합니다.

4. **virt-api** 배포 상태를 확인하여 자세한 내용을 확인하십시오. 이러한 명령은 관련 이벤트를 제공하고 이미지 가져오기 또는 충돌 **Pod**에 문제가 있는지 확인합니다.

```
$ oc -n $NAMESPACE get deployment virt-api -o yaml
```

```
$ oc -n $NAMESPACE describe deployment virt-api
```

5. 노드가 오버로드되었거나 **NotReady** 상태에 없는 경우와 같이 노드에 문제가 있는지 확인합니다.

```
$ oc get nodes
```

해결

실패한 **REST** 호출 속도가 높은 이유는 여러 가지가 있습니다. 근본 원인을 식별하고 적절한 조치를 취해야 합니다.

- 노드 리소스 소진
- 클러스터의 메모리가 충분하지 않음

- 노드 다운
- 스케줄러를 100% 사용할 수 없는 경우와 같이 API 서버 과부하)
- 네트워킹 문제

또는 지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.3.6. VirtControllerDown alert

설명

지난 5분 동안 virt-controllers 탐지가 발생하지 않으면 virt-controller 배포에는 기본 두 개의 Pod가 있습니다.

이유

virt-controller가 실패하면 새 VMI 시작 또는 기존 VMI 종료와 같은 VM 라이프사이클 관리 작업이 완전히 실패합니다.

문제 해결

1. 환경 변수 NAMESPACE 를 수정합니다.

```
$ export NAMESPACE="$(oc get kubevirt -A -o custom-
columns="":.metadata.namespace)"
```

2. virt-controller 배포의 상태를 확인합니다.

```
$ oc get deployment -n $NAMESPACE virt-controller -o yaml
```

3. virt-controller Pod의 이벤트를 확인합니다.

```
$ oc -n $NAMESPACE describe pods <virt-controller pod>
```

4. virt-controller Pod의 로그를 확인합니다.

```
$ oc -n $NAMESPACE logs <virt-controller pod>
```

5. 관리자 Pod의 로그를 확인하여 virt-controller Pod 생성에 실패하는 이유를 확인합니다.

```
$ oc get logs <virt-controller-pod>
```

로그의 virt-controller Pod 이름의 예는 virt-controller-7888c64d66-dzc9p 입니다. 그러나 virt-controller를 실행하는 Pod가 여러 개 있을 수 있습니다.

해결

virt-controller를 실행하지 않는 몇 가지 알려진 이유가 있습니다. 가능한 이유 목록에서 근본 원인을 확인하고 적절한 조치를 취합니다.

- 노드 리소스 소진
- 클러스터의 메모리가 충분하지 않음
- 노드 다운
- 스케줄러를 100% 사용할 수 없는 경우와 같이 API 서버 과부하)
- 네트워킹 문제

또는 지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.3.7. VirtControllerRESTErrorsBurst alert

설명

REST 호출 중 80% 이상이 지난 5분 동안 virt-controller에서 실패했습니다.

이유

virt-controller가 API 서버에 완전히 연결될 수 있습니다. 이러한 손실은 실행 중인 워크로드에는 영향을 미치지 않지만 상태 업데이트 및 마이그레이션과 같은 작업이 발생할 수 없습니다.

문제 해결

virt-controller REST 호출 실패와 관련된 일반적인 오류 유형은 다음 두 가지입니다.

- API 서버 과부하로 인해 시간 초과가 발생합니다. API 서버 지표와 응답 시간 및 전체 호출과 같은 세부 정보를 확인합니다.
- virt-controller Pod가 API 서버에 연결할 수 없습니다. 일반적인 원인은 다음과 같습니다.
  - 노드의 DNS 문제
  - 네트워킹 연결 문제

해결

virt-controller 로그를 확인하여 virt-controller pod가 API 서버에 연결할 수 없는지 확인합니다. 이 경우 Pod를 삭제하여 강제로 재시작합니다.

또한 노드 리소스가 소진되었는지 또는 클러스터에 메모리가 부족하여 연결에 실패하는지 확인합니다.

이 문제는 일반적으로 이 경고 범위를 벗어나는 DNS 또는 CNI 문제와 관련이 있습니다.

또는 지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.3.8. VirtHandlerRESTErrorsBurst alert

설명

REST 호출 중 80% 이상이 지난 5분 동안 virt-handler에서 실패했습니다.

이유

virt-handler에서 **API** 서버에 대한 연결을 손실했습니다. 영향을 받는 노드에서 워크로드를 실행하지만 상태 업데이트는 마이그레이션과 같은 전파 및 작업을 수행할 수 없습니다.

문제 해결

virt-operator REST 호출 실패와 관련된 두 가지 일반적인 오류 유형이 있습니다.

- **API** 서버 과부하로 인해 시간 초과가 발생합니다. **API** 서버 지표와 응답 시간 및 전체 호출과 같은 세부 정보를 확인합니다.
- **virt-operator Pod**가 **API** 서버에 연결할 수 없습니다. 일반적인 원인은 다음과 같습니다.
  - 노드의 **DNS** 문제
  - 네트워킹 연결 문제

해결

virt-handler가 **API** 서버에 연결할 수 없는 경우 **Pod**를 삭제하여 강제로 재시작합니다. 이 문제는 일반적으로 이 경고 범위를 벗어나는 **DNS** 또는 **CNI** 문제와 관련이 있습니다. 근본 원인을 식별하고 적절한 조치를 취해야 합니다.

또는 지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.3.9. VirtOperatorDown alert

설명

이 경고는 지난 **10**분 동안 **virt-operator Pod**가 **Running** 상태가 아닌 경우 발생합니다. **virt-operator** 배포에는 두 개의 **Pod**의 기본 복제본이 있습니다.

이유

virt-operator는 **OpenShift Container Platform** 클러스터에서 활성화된 첫 번째 **Kubernetes Operator**입니다. 주요 기능은 다음과 같습니다.

- 설치
- **Live-update**
- 클러스터의 실시간 업그레이드
- **virt-controller, virt-handler, virt-launcher**와 같은 최상위 컨트롤러의 라이프사이클 모니터링
- 최상위 컨트롤러 조정 관리

또한 **virt-operator**는 인증서 교체 및 일부 인프라 관리와 같은 클러스터 전체 작업을 담당합니다.

참고

virt-operator는 클러스터의 가상 머신을 직접 처리하지 않습니다. **virt-operator**의 고가용성은 사용자 정의 워크로드에 영향을 미치지 않습니다.

이 경고는 클러스터 수준에서 오류가 있음을 나타냅니다. 컨트롤러의 인증 교체, 업그레이드 및 조정과 같은 중요한 클러스터 전체의 관리 기능을 일시적으로 사용할 수 없습니다.

문제 해결

1. 환경 변수 **NAMESPACE** 를 수정합니다.

```
$ export NAMESPACE="$(oc get kubevirt -A -o custom-
columns="":.metadata.namespace)"
```

2. **virt-operator** 배포의 상태를 확인합니다.

```
$ oc get deployment -n $NAMESPACE virt-operator -o yaml
```

3. **virt-operator Pod**의 이벤트를 확인합니다.

```
$ oc -n $NAMESPACE describe pods <virt-operator pod>
```

4. virt-operator Pod의 로그를 확인합니다.

```
$ oc -n $NAMESPACE logs <virt-operator pod>
```

5. 관리자 Pod의 로그를 확인하여 virt-operator Pod 생성에 실패하는 이유를 확인합니다.

```
$ oc get logs <virt-operator-pod>
```

로그의 virt-operator Pod 이름의 예는 virt-operator-7888c64d66-dzc9p 입니다. 그러나 virt-operator를 실행하는 Pod가 여러 개 있을 수 있습니다.

해결

virt-operator 실행 중이 발생하지 않는 몇 가지 알려진 이유가 있습니다. 가능한 이유 목록에서 근본 원인을 확인하고 적절한 조치를 취합니다.

- 노드 리소스 소진
- 클러스터의 메모리가 충분하지 않음
- 노드 다운
- 스케줄러를 100% 사용할 수 없는 경우와 같이 API 서버 과부하)
- 네트워킹 문제

또는 지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.3.10. VirtOperatorRESTErrorsBurst alert

설명

REST 호출 중 80% 이상이 지난 5분 동안 virt-operator에서 실패했습니다.

이유

virt-operator에서 API 서버에 대한 연결이 끊어졌습니다. 업그레이드 및 컨트롤러 조정과 같은 클러스터 수준 작업이 작동하지 않습니다. VM 및 VMI와 같은 고객 워크로드에는 영향을 미치지 않습니다.

문제 해결

virt-operator REST 호출 실패와 관련된 두 가지 일반적인 오류 유형이 있습니다.

- API 서버 과부하로 인해 시간 초과가 발생합니다. API 서버 지표 및 응답 시간 및 전체 호출과 같은 세부 정보를 확인합니다.
- virt-operator Pod가 API 서버에 연결할 수 없습니다. 일반적인 원인은 네트워크 연결 문제 및 노드의 DNS 문제입니다. virt-operator 로그를 확인하여 Pod가 API 서버에 연결할 수 있는지 확인합니다.

```
$ export NAMESPACE="$(oc get kubevirt -A -o custom-
columns="":.metadata.namespace)"
```

```
$ oc -n $NAMESPACE get pods -l kubevirt.io=virt-operator
```

```
$ oc -n $NAMESPACE logs <pod-name>
```

```
$ oc -n $NAMESPACE describe pod <pod-name>
```

해결

virt-operator가 API 서버에 연결할 수 없는 경우 Pod를 삭제하여 강제로 재시작합니다. 이 문제는 일반적으로 이 경고 범위를 벗어나는 DNS 또는 CNI 문제와 관련이 있습니다. 근본 원인을 식별하고 적절한 조치를 취해야 합니다.

또는 지원 문제를 열고 문제 해결 프로세스에 수집된 정보를 제공합니다.

13.12.4. 추가 리소스

- 지원 요청

13.13. RED HAT 지원을 위한 데이터 수집

Red Hat 지원에 지원 케이스를 제출하면 다음 툴을 사용하여 OpenShift Container Platform 및 OpenShift Virtualization에 대한 디버깅 정보를 제공하는 것이 도움이 됩니다.

must-gather 툴

must-gather 툴은 리소스 정의 및 서비스 로그를 포함한 진단 정보를 수집합니다.

Prometheus

Prometheus는 시계열 데이터베이스이며 메트릭에 대한 규칙 평가 엔진입니다. Prometheus는 처리를 위해 Alertmanager에 경고를 보냅니다.

Alertmanager

Alertmanager 서비스는 Prometheus에서 수신한 경고를 처리합니다. 또한 Alertmanager는 경고를 외부 알림 시스템으로 전송합니다.

13.13.1. 환경에 대한 데이터 수집

환경에 대한 데이터를 수집하면 근본 원인을 분석하고 결정하는 데 필요한 시간이 최소화됩니다.

사전 요구 사항

- Prometheus 지표 데이터의 보존 시간을 최소 7일로 설정합니다.
- 관련 경고를 캡처하고 클러스터 외부에서 보거나 유지할 수 있도록 전용 DestinationRule으로 보내도록 Alertmanager를 구성합니다.
- 영향을 받는 노드와 가상 머신의 정확한 수를 기록합니다.

절차

1. 기본 must-gather 이미지를 사용하여 클러스터의 must-gather 데이터를 수집합니다.
2. 필요한 경우 Red Hat OpenShift Data Foundation에 대한 must-gather 데이터를 수집합니다.

3. OpenShift Virtualization must-gather 이미지를 사용하여 OpenShift Virtualization에 대한 must-gather 데이터를 수집합니다.

4. 클러스터의 Prometheus 지표를 수집합니다.

13.13.1.1. 추가 리소스

- Prometheus 지표 데이터의 보존 시간 구성
- 외부 시스템에 경고 알림을 전송하도록 Alertmanager 구성
- OpenShift Container Platform의 must-gather 데이터 수집
- Red Hat OpenShift Data Foundation의 must-gather 데이터 수집
- OpenShift Virtualization의 must-gather 데이터 수집
- 클러스터 관리자로 모든 프로젝트의 Prometheus 지표 수집

13.13.2. 가상 머신에 대한 데이터 수집

VM(가상 머신)에 대한 데이터를 수집하면 근본 원인을 분석하고 결정하는 데 필요한 시간이 최소화됩니다.

사전 요구 사항

- Windows VMs:
  - Red Hat 지원에 대한 Windows 패치 업데이트 세부 정보를 기록합니다.
  - VirtIO 드라이버의 최신 버전을 설치합니다. VirtIO 드라이버에는 QEMU 게스트 에이전트가 포함됩니다.

- RDP(Remote Desktop Protocol)가 활성화된 경우 RDP를 사용하여 VM에 연결하여 연결 소프트웨어에 문제가 있는지 확인합니다.

절차

1. 가상 머신에 대한 자세한 must-gather 데이터를 수집합니다.
2. 충돌한 VM의 스크린샷을 다시 시작하기 전에 수집합니다.
3. 가상 머신의 공통 요인을 기록합니다. 예를 들어 VM은 동일한 호스트 또는 네트워크를 갖습니다.

13.13.2.1. 추가 리소스

- Windows VM에 VirtIO 드라이버 설치
- 호스트에 액세스하지 않고 Windows VM에 VirtIO 드라이버 다운로드 및 설치
- 웹 콘솔 또는 명령줄을 사용하여 RDP로 Windows VM에 연결
- 가상 머신에대한 must-gather 데이터 수집

13.13.3. OpenShift Virtualization에 must-gather 툴 사용

OpenShift Virtualization 이미지로 must-gather 명령을 실행하여 OpenShift Virtualization 리소스에 대한 데이터를 수집할 수 있습니다.

기본 데이터 컬렉션에는 다음 리소스에 대한 정보가 포함됩니다.

- 하위 오브젝트를 포함한 OpenShift Virtualization Operator 네임스페이스
- OpenShift Virtualization 사용자 정의 리소스 정의

- 가상 머신이 포함된 네임스페이스
- 기본 가상 머신 정의

절차

- 다음 명령을 실행하여 OpenShift Virtualization에 대한 데이터를 수집합니다.

```
$ oc adm must-gather --image-stream=openshift/must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.10.10
```

13.13.3.1. must-gather 툴 옵션

다음 옵션에 스크립트 및 환경 변수 조합을 지정할 수 있습니다.

- 네임스페이스에서 자세한 VM(가상 머신) 정보 수집
- 지정된 VM에 대한 자세한 정보 수집
- 이미지 및 이미지 스트림 정보 수집
- must-gather 툴에서 사용하는 최대 병렬 프로세스 수 제한

13.13.3.1.1. 매개 변수

환경 변수

호환되는 스크립트의 환경 변수를 지정할 수 있습니다.

NS=<namespace_name>

지정한 네임스페이스에서 virt-launcher Pod 세부 정보를 포함한 가상 머신 정보를 수집합니다. 모든 네임스페이스에 대해 VirtualMachine 및 VirtualMachineInstance CR 데이터가 수집됩니다.

VM=<vm_name>

특정 가상 머신에 대한 세부 정보를 수집합니다. 이 옵션을 사용하려면 **NS** 환경 변수를 사용하여 네임스페이스를 지정해야 합니다.

PROS=<number_of_processes>

must-gather 툴이 사용하는 최대 병렬 프로세스 수를 수정합니다. 기본값은 **5** 입니다.

중요

너무 많은 병렬 프로세스를 사용하면 성능 문제가 발생할 수 있습니다. 최대 병렬 프로세스 수를 늘리는 것은 권장되지 않습니다.

스크립트

각 스크립트는 특정 환경 변수 조합에서만 호환됩니다.

gather_vms_details

OpenShift Virtualization 리소스에 속하는 **VM** 로그 파일, **VM** 정의 및 네임스페이스(및 하위 오브젝트)를 수집합니다. 네임스페이스 또는 **VM**을 지정하지 않고 이 매개변수를 사용하는 경우 **must-gather** 툴은 클러스터의 모든 **VM**에 대해 이 데이터를 수집합니다. 이 스크립트는 모든 환경 변수와 호환되지만 **VM** 변수를 사용하는 경우 네임스페이스를 지정해야 합니다.

gather

모든 네임스페이스에서 클러스터 데이터를 수집하고 기본 **VM** 정보만 포함하는 기본 **must-gather** 스크립트를 사용합니다. 이 스크립트는 **PROS** 변수와만 호환됩니다.

gather_images

이미지 및 이미지 스트림 사용자 정의 리소스 정보를 수집합니다. 이 스크립트는 **PROS** 변수와만 호환됩니다.

13.13.3.1.2. 사용 및 예

환경 변수는 선택 사항입니다. 호환 가능한 하나 이상의 환경 변수를 사용하여 단독으로 또는 하나 이상의 스크립트를 실행할 수 있습니다.

표 **13.1.** 호환되는 매개변수

스크립트	호환 가능한 환경 변수
gather_vms_details	• 네임스페이스의 경우 **NS=<namespace_name>** • VM: **VM=<vm_name> NS=<namespace_name>** • **PROS=<number_of_processes>**
gather	• **PROS=<number_of_processes>**
gather_images	• **PROS=<number_of_processes>**

must-gather 가 수집하는 데이터를 사용자 정의하려면 명령에 이중 대시(**--**)를 추가한 다음 공백과 하나 이상의 호환 가능한 매개변수를 추가합니다.

구문

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.10.10 \
  -- <environment_variable_1> <environment_variable_2> <script_name>
```

자세한 VM 정보

다음 명령은 **my namespace** 네임스페이스 에서 **my-vm VM**에 대한 자세한 **VM** 정보를 수집합니다.

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.10.10 \
  -- NS=mynamespace VM=my-vm gather_vms_details (1)
```

(1) VM 환경 변수를 사용하는 경우 NS 환경 변수가 필요합니다.

세 개의 병렬 프로세스로 제한된 기본 데이터 수집

다음 명령은 최대 3개의 병렬 프로세스를 사용하여 기본 must-gather 정보를 수집합니다.

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.10.10 \
  -- PROS=3 gather
```

이미지 및 이미지 스트림 정보

다음 명령은 클러스터에서 이미지 및 이미지 스트림 정보를 수집합니다.

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.10.10 \
  -- gather_images
```

13.13.3.2. 추가 리소스

- must-gather 툴 정보

14장. 백업 및 복원

14.1. 가상 머신 백업 및 복원

OpenShift API for Data Protection(OADP) 을 사용하여 가상 머신을 백업 및 복원합니다.

중요

OpenShift Virtualization의 OADP는 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.

사전 요구 사항

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

절차

1. 스토리지 공급자의 지침에 따라 OADP Operator 를 설치합니다.
2. kubevirt 및 openshift 플러그인을 사용하여 데이터 보호 애플리케이션을 설치합니다.
3. Backup CR(사용자 정의 리소스)을 생성하여 가상 머신을 백업합니다.
4. Restore CR을 생성하여 Backup CR을 복원합니다.

14.1.1. 추가 리소스

- OADP 기능 및 플러그인

- 문제 해결