OpenShift Container Platform 4.12

가상화

OpenShift Virtualization 설치, 사용법, 릴리스 정보

Last Updated: 2024-09-07

OpenShift Container Platform 4.12 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 원활하게 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 네트워크 플러그인 중 하나와 함께 OpenShift Virtualization을 사용할 수 있습니다.

1.1.1. OpenShift Virtualization 지원 클러스터 버전

OpenShift Container Platform 4.12 클러스터에서 사용할 수 있도록 OpenShift Virtualization 4.12가 지원됩니다. OpenShift Virtualization의 최신 z-stream 릴리스를 사용하려면 먼저 최신 버전의 OpenShift Container Platform으로 업그레이드해야 합니다.

1.2. 단일 노드 OPENSHIFT 차이점

단일 노드 클러스터에 OpenShift Virtualization을 설치할 수 있습니다.

지원 설치 프로그램을 사용하여 단일 노드 OpenShift 클러스터를 프로비저닝하면 사전 구성된 영구 스토리지가 자동으로 배포됩니다.

- OpenShift Virtualization 4.10 및 4.11에서는 HPP(HostPath Provisioner)가 자동으로 설치됩니다.

- OpenShift Virtualization 4.12에서 OpenShift Data Foundation Logical Volume Manager Operator는 즉시 제공되는 스토리지 솔루션입니다. HPP를 사용하여 수동으로 배포할 수도 있습니다.

참고

단일 노드 OpenShift는 고가용성을 지원하지 않습니다. 다중 노드 클러스터의 기능에 다음과 같은 차이점이 있습니다.

- Pod 중단 예산은 지원되지 않습니다.
- 실시간 마이그레이션은 지원되지 않습니다.
- 스토리지 동작의 차이로 인해 일부 가상 머신 템플릿은 단일 노드 OpenShift와 호환되지 않습니다. 호환성을 보장하기 위해 데이터 볼륨 또는 스토리지 프로필을 사용하는 가상 머신에 제거 전략이 설정되어 있지 않아야 합니다.

1.3. 추가 리소스

- 단일 노드 OpenShift 정보
- 지원되는 설치 관리자
- HostPath Provisioner (HPP)
- OpenShift Container Platform Data Foundation Logical Volume Manager Operator
- Pod 중단 예산
- 실시간 마이그레이션
- 제거 전략
- OpenShift Virtualization 4.x에 지원되는 제한 사항

2장. OPENSHIFT VIRTUALIZATION 아키텍처

OpenShift Virtualization 아키텍처에 대해 알아보기.

2.1. OPENSHIFT VIRTUALIZATION 아키텍처의 작동 방식

OpenShift Virtualization을 설치한 후 OLM(Operator Lifecycle Manager)은 OpenShift Virtualization의 각 구성 요소에 대해 Operator Pod를 배포합니다.

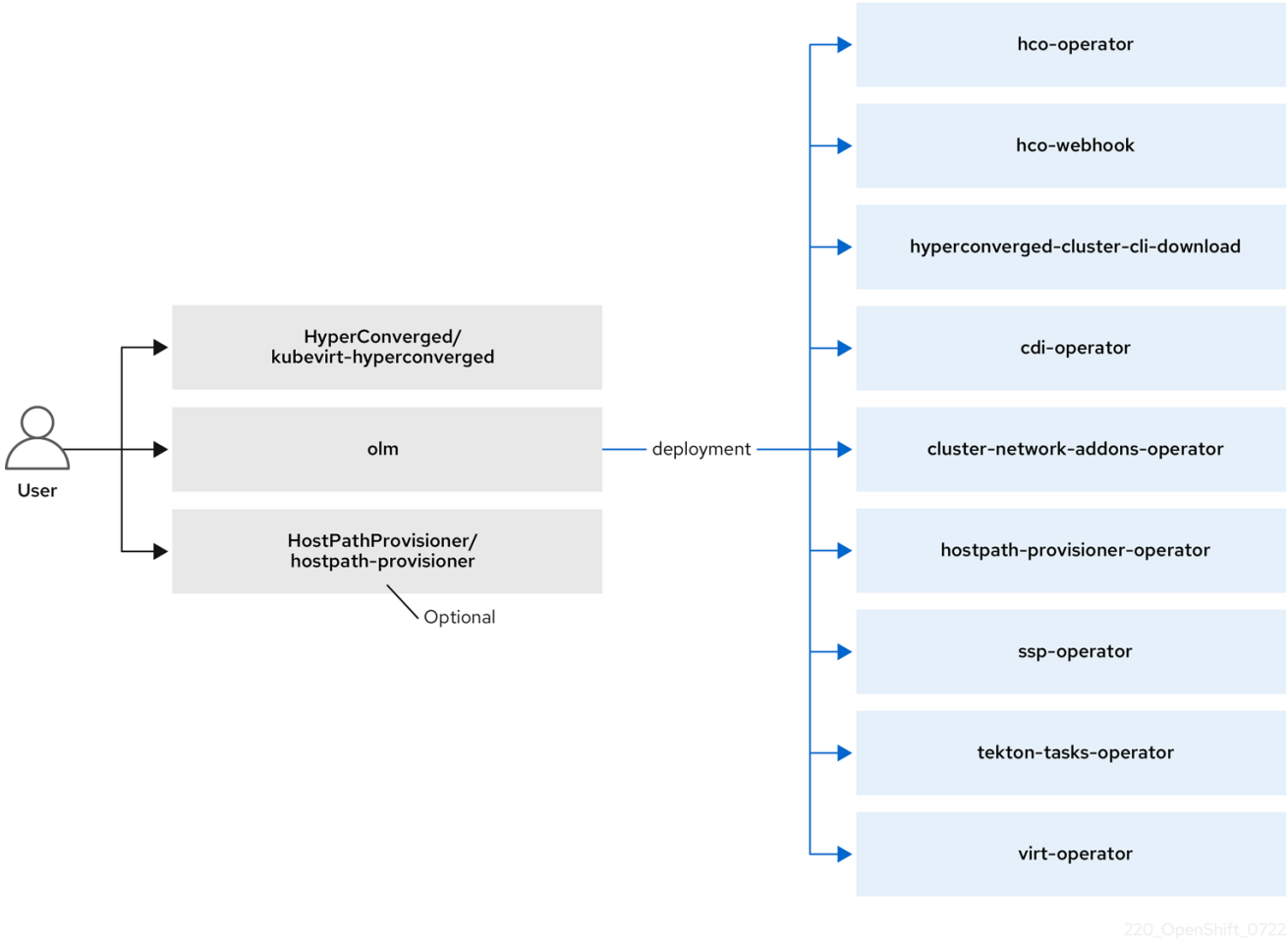
- 컴퓨팅: **virt-operator**
- 스토리지: **cdi-operator**
- 네트워크: **cluster-network-addons-operator**
- 스케일링: **ssp-operator**
- templating: **tekton-tasks-operator**

또한 OLM은 다른 구성 요소의 배포, 구성 및 라이프사이클을 담당하는 **hyperconverged-cluster-operator** Pod 및 **hco-webhook**, **hyperconverged-cluster-cli-download** 의 여러 도우미 Pod를 배포합니다.

모든 Operator Pod가 배포된 후 **HyperConverged** CR(사용자 정의 리소스)을 생성해야 합니다. **HyperConverged** CR에 설정된 구성은 단일 정보 소스 및 OpenShift Virtualization의 진입점 역할을 하며 CR의 동작을 안내합니다.

HyperConverged CR은 조정 루프 내에서 다른 모든 구성 요소의 Operator에 대한 해당 CR을 생성합니다. 그런 다음 각 Operator는 데몬 세트, 구성 맵, OpenShift Virtualization 컨트롤 플레인에 대한 추가 구성 요소와 같은 리소스를 생성합니다. 예를 들어 **hco-operator** 에서 **KubeVirt** CR을 생성할 때 **virt-operator** 는 이를 조정하고 **virt-controller**,**virt-handler**, **virt-api** 와 같은 추가 리소스를 생성합니다.

OLM은 **hostpath-provisioner-operator** 를 배포하지만 HBA(Host **path** 프로비전 프로그램) CR을 생성할 때까지 작동하지 않습니다.



추가 리소스

- 하이퍼컨버지드 CR 구성
- Virtctl 클라이언트 명령

2.2. HCO-OPERATOR 정보

hco-operator (HCO)는 OpenShift Virtualization 배포 및 관리를 위한 단일 진입점과 의견된 기본값을 사용하여 여러 도우미 운영자를 제공합니다. 또한 해당 Operator에 대한 사용자 정의 리소스(CR)를 생성합니다.

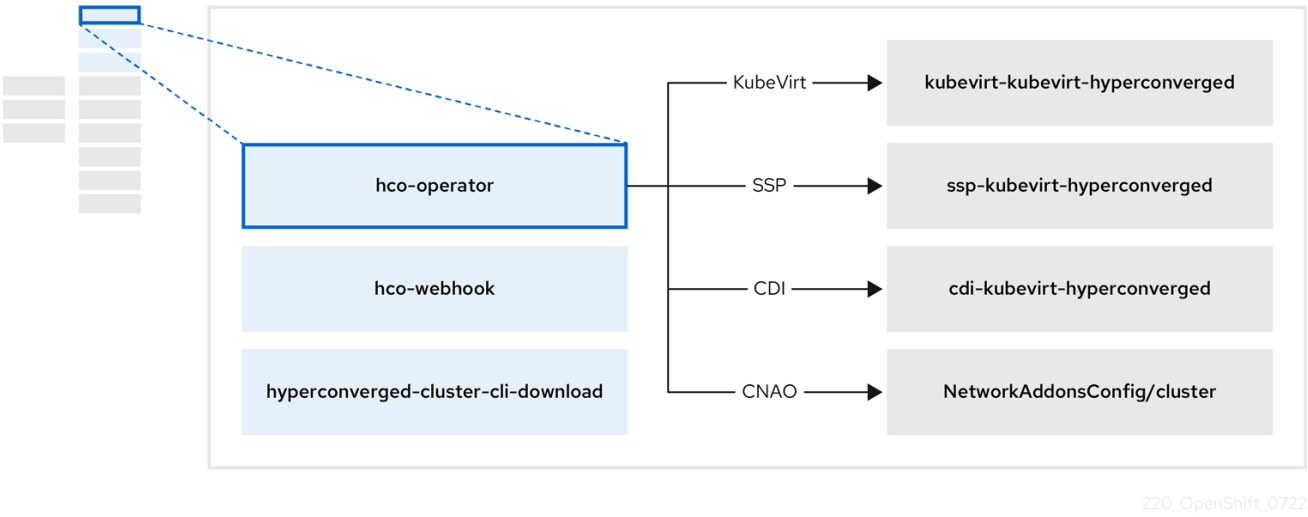


표 2.1. HCO-operator 구성 요소

구성 요소	설명
deployment/hco-webhook	**HyperConverged** 사용자 정의 리소스 콘텐츠를 검증합니다.
deployment/hyperconverged-cluster-cli-download	클러스터에서 직접 다운로드할 수 있도록 **virtctl** 툴 바이너리를 클러스터에 제공합니다.
KubeVirt/kubevirt-kubevirt-hyperconverged	OpenShift Virtualization에 필요한 모든 Operator, CR 및 오브젝트를 포함합니다.
SSP/ssp-kubevirt-hyperconverged	SSP CR입니다. 이는 HCO에 의해 자동으로 생성됩니다.
CDI/cdi-kubevirt-hyperconverged	A CDI CR. 이는 HCO에 의해 자동으로 생성됩니다.
NetworkAddonsConfig/cluster	**cluster-network-addons-operator** 에 의해 관리되고 있는 CR.

2.3. CDI-OPERATOR 정보

cdi-operator 는 데이터 볼륨을 사용하여 가상 머신(VM) 이미지를 PVC(영구 볼륨 클레임)로 가져오는 CDI(Containerized Data Importer) 및 관련 리소스를 관리합니다.

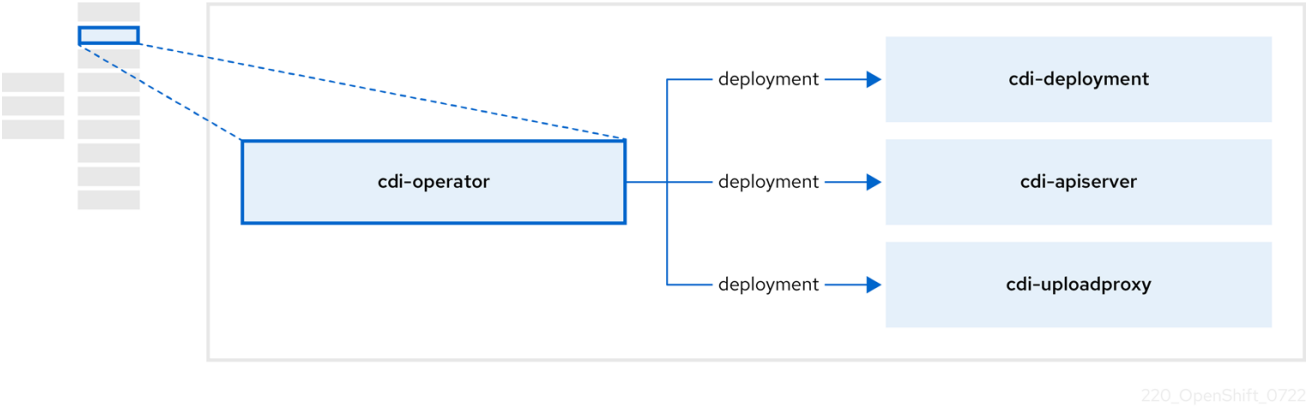


표 2.2. CDI-operator 구성 요소

구성 요소	설명
deployment/cdi-apiserver	보안 업로드 토큰을 발행하여 VM 디스크를 PVC에 업로드하는 권한 부여를 관리합니다.
deployment/cdi-uploadproxy	올바른 PVC에 쓸 수 있도록 외부 디스크 업로드 트래픽을 적절한 업로드 서버 Pod로 보냅니다. 유효한 업로드 토큰이 필요합니다.
pod/cdi-importer	데이터 볼륨을 생성할 때 가상 머신 이미지를 PVC로 가져오는 도우미 Pod

2.4. CLUSTER-NETWORK-ADDONS-OPERATOR 정보

cluster-network-addons-operator 는 클러스터에 네트워킹 구성 요소를 배포하고 확장된 네트워크 기능을 위한 관련 리소스를 관리합니다.

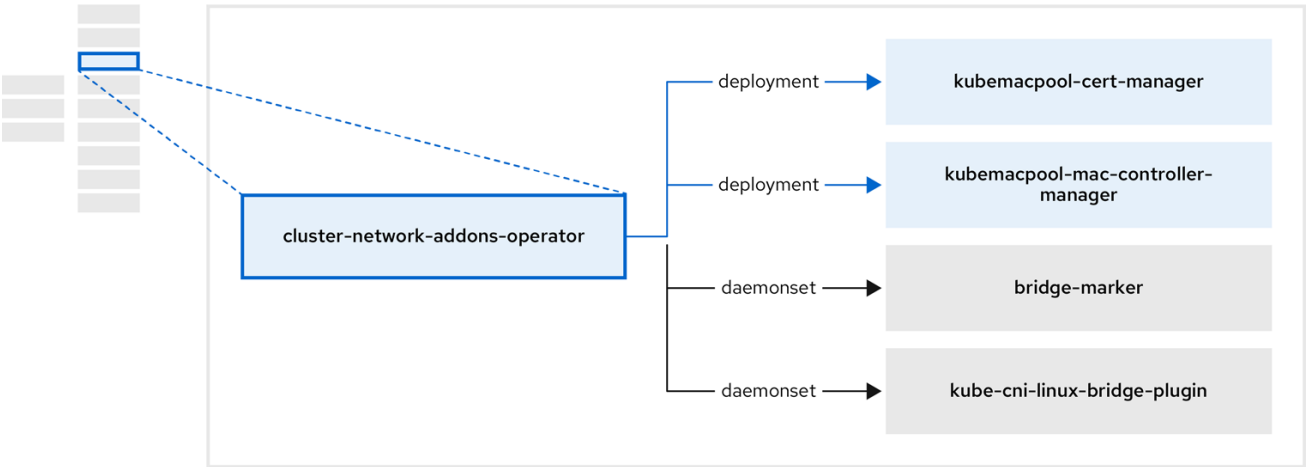


표 2.3. cluster-network-addons-operator components

구성 요소	설명
deployment/kubemacpool-cert-manager	Kubemacpool의 웹 후크의 TLS 인증서를 관리합니다.
deployment/kubemacpool-mac-controller-manager	VM(가상 머신) NIC(네트워크 인터페이스 카드)에 대한 MAC 주소 풀링 서비스를 제공합니다.
daemonset/bridge-marker	노드에서 사용 가능한 네트워크 브릿지를 노드 리소스로 표시합니다.
daemonset/kube-cni-linux-bridge-plugin	클러스터 노드에 CNI 플러그인을 설치하여 네트워크 연결 정의를 통해 VM을 Linux 브리지에 연결할 수 있습니다.

2.5. HOSTPATH-PROVISIONER-OPERATOR 정보

hostpath-provisioner-operator 는 다중 노드 호스트 경로 프로비전 프로그램(HPP) 및 관련 리소스를 배포하고 관리합니다.

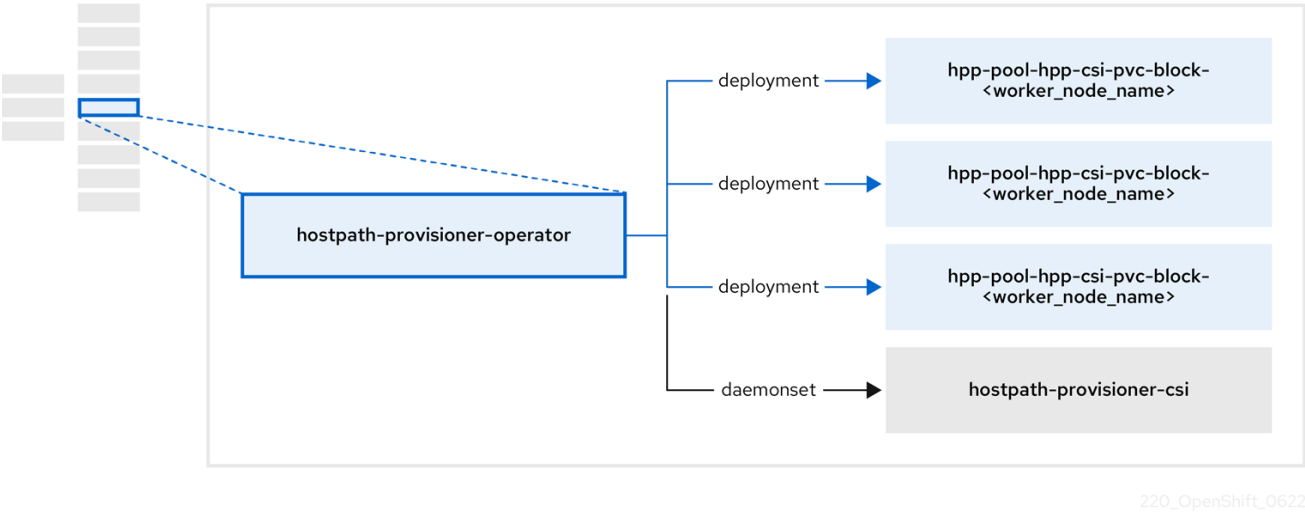


표 2.4. hostPath-provisioner-operator 구성 요소

구성 요소	설명
deployment/hpp-pool-hpp-csi-pvc-block-<worker_node_name>	HPP(Hostpath 프로비전 프로그램)가 실행되도록 지정된 각 노드에 대한 작업자를 제공합니다. Pod는 지정된 백업 스토리지를 노드에 마운트합니다.
daemonset/hostpath-provisioner-csi	HPP의 CSI(Container Storage Interface) 드라이버 인터페이스를 구현합니다.
daemonset/hostpath-provisioner	HPP의 레거시 드라이버 인터페이스를 구현합니다.

2.6. SSP-OPERATOR 정보

ssp-operator 는 공통 템플릿, 관련 기본 부팅 소스 및 템플릿 유효성 검증기를 배포합니다.

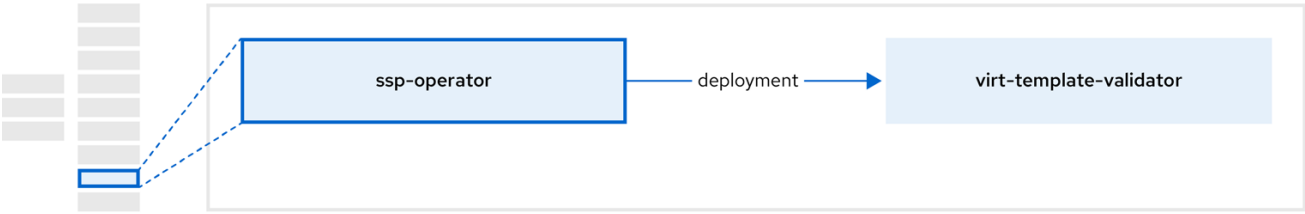


표 2.5. SSP-operator 구성 요소

구성 요소	설명
deployment/virt-template-validator	템플릿에서 생성된 가상 머신에서 **vm.kubevirt.io/validations** 주석을 확인하고 유효하지 않은 경우 해당 주석을 거부합니다.

2.7. TEKTON-TASKS-OPERATOR 정보

tekton-tasks-operator 는 VM의 OpenShift Pipelines 사용을 보여주는 예제 파이프라인을 배포합니다. 또한 사용자가 템플릿에서 VM을 생성하고 템플릿을 복사 및 수정하고 데이터 볼륨을 생성할 수 있는 추가 OpenShift Pipeline 작업을 배포합니다.

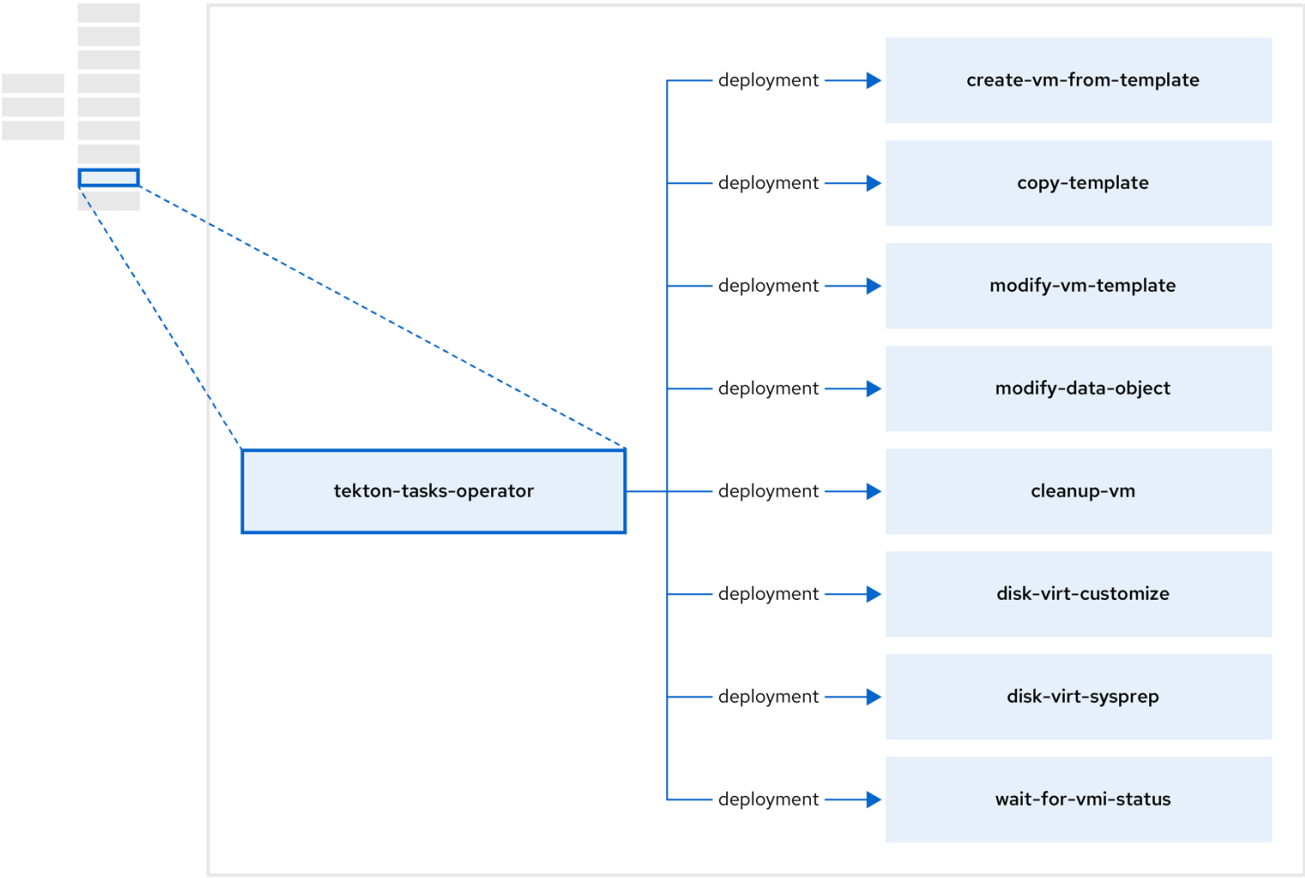
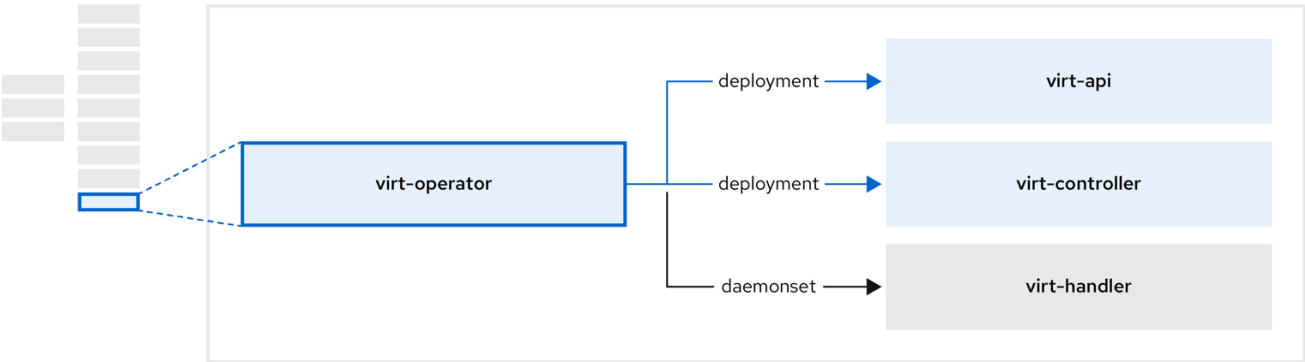


표 2.6. Tekton-tasks-operator 구성 요소

구성 요소	설명
deployment/create-vm-from-template	템플릿에서 VM을 생성합니다.
deployment/copy-template	VM 템플릿을 복사합니다.
deployment/modify-vm-template	VM 템플릿을 생성하거나 제거합니다.
deployment/modify-data-object	데이터 볼륨 또는 데이터 소스를 생성하거나 제거합니다.
deployment/cleanup-vm	VM에서 스크립트 또는 명령을 실행한 다음 VM을 중지하거나 삭제합니다.
deployment/disk-virt-customize	**virt- customize** 를 사용하여 대상 PVC에서 사용자 지정 스크립트를 실행합니다.
deployment/disk-virt-sysprep	**virt- sysprep** 을 사용하여 대상 PVC에서 sysprep 스크립트를 실행합니다.
deployment/wait-for-vmi-status	특정 VMI 상태가 될 때까지 기다린 다음 해당 상태에 따라 실패하거나 성공합니다.

2.8. VIRT-OPERATOR 정보

virt-operator 는 현재 VM(가상 머신) 워크로드를 중단하지 않고 OpenShift Virtualization을 배포, 업그레이드, 관리합니다.



220_OpenShift_0622

표 2.7. virt-operator 구성 요소

구성 요소	설명
deployment/virt-api	모든 가상화 관련 흐름에 대한 진입점 역할을 하는 HTTP API 서버입니다.

구성 요소	설명
deployment/virt-controller	새 VM 인스턴스 오브젝트 생성을 관찰하고 해당 Pod를 생성합니다. Pod가 노드에 예약되면 **virt-controller** 가 VM을 노드 이름으로 업데이트합니다.
daemonset/virt-handler	VM의 변경 사항을 모니터링하고 **virt-launcher** 에 필요한 작업을 수행하도록 지시합니다. 이 구성 요소는 노드에 따라 다릅니다.
pod/virt-launcher	**libvirt** 및 **qemu** 에서 구현된 대로 사용자가 생성한 VM을 포함합니다.

3장. OPENSHIFT VIRTUALIZATION 시작하기

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

참고

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

3.1. OPENSHIFT VIRTUALIZATION 계획 및 설치

OpenShift Container Platform 클러스터에 OpenShift Virtualization을 계획 및 설치합니다.

- OpenShift Virtualization용 베어 메탈 클러스터를 계획합니다.
- OpenShift Virtualization을 위한 클러스터를 준비합니다.
- OpenShift Virtualization Operator를 설치합니다.
- **virtctl** CLI(명령줄 인터페이스) 툴을 설치합니다.

계획 및 설치 리소스

- CSI 지원 스토리지 공급자 사용
- 가상 머신의 로컬 스토리지 구성.
- Kubernetes NMState Operator 설치
- 가상 머신의 노드 지정.
- **virtctl** 명령.

3.2. 가상 머신 생성 및 관리

웹 콘솔을 사용하여 VM(가상 머신)을 생성합니다.

- 빠른 VM 생성
- VM을 생성할 템플릿을 사용자 지정합니다.

VM에 연결합니다.

- 웹 콘솔을 사용하여 VM의 직렬 콘솔 또는 VNC 콘솔에 연결합니다.
- SSH를 사용하여 VM에 연결합니다.
- RDP를 사용하여 Windows VM에 연결합니다.

VM을 관리합니다.

- 웹 콘솔을 사용하여 VM을 중지, 시작, 일시 중지 및 다시 시작합니다 .
- **virtctl** CLI 툴을 사용하여 VM을 관리하거나, 포트를 노출하거나, 직렬 콘솔에 연결합니다 .

3.3. 다음 단계

- VM을 보조 네트워크에 연결합니다.
  - VM을 Linux 브리지 네트워크에 연결합니다.
  - VM을 SR-IOV 네트워크에 연결합니다.

> 참고
>
> VM은 기본적으로 Pod 네트워크에 연결됩니다. Linux 브리지 또는 SR-IOV와 같은 보조 네트워크를 구성한 다음 VM 구성에 네트워크를 추가해야 합니다.

- 웹 콘솔을 사용하여 리소스, 세부 정보, 상태 및 상위 소비자를 모니터링합니다.
- 웹 콘솔을 사용하여 VM 워크로드에 대한 고급 정보를 봅니다.
- CLI를 사용하여 OpenShift Virtualization 로그를 확인합니다.
- **sysprep**을 사용하여 Windows VM 배포를 자동화 합니다.
- 실시간 마이그레이션 VM.
- VM 백업 및 복원.

4장. 웹 콘솔 개요

OpenShift Container Platform 웹 콘솔의 가상화 섹션에는 OpenShift **Virtualization** 환경을 관리하고 모니터링하기 위한 다음 페이지가 포함되어 있습니다.

표 **4.1.** 가상화 페이지

페이지	설명
개요 페이지	OpenShift Virtualization 환경을 관리하고 모니터링합니다.
카탈로그 페이지	템플릿 카탈로그에서 VirtualMachine을 생성합니다.
VirtualMachines 페이지	VirtualMachine을 구성하고 모니터링합니다.
템플릿 페이지	템플릿을 생성하고 관리합니다.
데이터 소스 페이지	VirtualMachine 부팅 소스의 DataSources를 생성하고 관리합니다.
MigrationPolicies 페이지	워크로드의 MigrationPolicies를 생성하고 관리합니다.

표 **4.2.** 키

icon	설명
	아이콘 편집
	링크 아이콘

4.1. 개요 페이지

개요 페이지에는 리소스, 지표, 마이그레이션 진행 상황 및 클러스터 수준 설정이 표시됩니다.

예 **4.1.** 개요 페이지

요소	설명
virtctl 다운로드	**virtctl** 명령줄 툴을 다운로드하여 리소스를 관리합니다.
개요 탭	리소스, 사용량, 경고 및 상태.
상위 소비자 탭	CPU, 메모리 및 스토리지 리소스의 상위 소비자입니다.
마이그레이션 탭	실시간 마이그레이션 상태.

요소	설명
설정 탭	실시간 마이그레이션 제한 및 사용자 권한을 포함한 클러스터 전체 설정

4.1.1. 개요 탭

개요 탭에는 리소스, 사용량, 경고 및 상태가 표시됩니다.

예 4.2. 개요 탭

요소	설명
"시작된 리소스" 카드	• "빠른 시작" 타일: step-by-step instructions and tasks를 사용하여 VirtualMachine을 생성, 가져오기 및 실행하는 방법. • "기능 강조 표시" 타일: 주요 가상화 기능에 대한 최신 정보를 읽어 보십시오. • "관련 연산자" 타일: Kubernetes NMState Operator 또는 OpenShift Data Foundation Operator와 같은 Operator를 설치합니다.
"VirtualMachines" tile	지난 7일의 추세를 보여주는 차트가 있는 VirtualMachine 수입니다.
"vCPU 사용량" 타일	지난 7일의 추세를 보여주는 차트가 있는 vCPU 사용량.
"메모리" 타일	지난 7일의 추세를 보여주는 차트가 있는 메모리 사용량입니다.
"스토리지" 타일	지난 7일의 추세를 보여주는 차트가 있는 스토리지 사용량입니다.
"알erts" 타일	심각도별로 그룹화된 OpenShift Virtualization 경고.
"VirtualMachine statuses" 타일	상태로 그룹화된 VirtualMachine 수입니다.
"템플릿당 가상 머신" 차트	템플릿 이름으로 그룹화된 템플릿에서 생성된 VirtualMachine 수입니다.

4.1.2. 상위 소비자 탭

상위 소비자 탭에는 CPU, 메모리 및 스토리지의 상위 소비자가 표시됩니다.

예 4.3. 상위 소비자 탭

요소	설명
가상화 대시보드 보기	**Observe → Dashboards** 에 연결하여 OpenShift Virtualization의 상위 소비자가 표시됩니다.
기간 목록	결과를 필터링할 기간을 선택합니다.
상위 소비자 목록	결과를 필터링할 상위 소비자 수를 선택합니다.
"CPU" 차트	CPU 사용량이 가장 많은 VirtualMachine
"메모리" 차트	메모리 사용량이 가장 많은 VirtualMachine
"메모리 스왑 트래픽" 차트	메모리 스왑 트래픽이 가장 높은 VirtualMachine
"vCPU 대기" 차트	vCPU 대기 기간이 가장 높은 VirtualMachine
"스토리지 처리량" 차트	스토리지 처리량이 가장 높은 VirtualMachine
"스토리지 IOPS" 차트	초당 가장 높은 스토리지 입력/출력 작업이 있는 VirtualMachine입니다.

4.1.3. 마이그레이션 탭

Migrations 탭에 VirtualMachineInstance 마이그레이션 상태가 표시됩니다.

예 **4.4.** 마이그레이션 탭

요소	설명
기간 목록	VirtualMachineInstanceMigrations를 필터링할 기간을 선택합니다.
VirtualMachineInstanceMigrations 테이블	VirtualMachineInstance 마이그레이션 목록입니다.

4.1.4. 설정 탭

Settings 탭에는 다음 탭의 클러스터 전체 설정이 표시됩니다.

표 **4.3.** 설정 탭의 탭

탭	설명
일반 탭	OpenShift Virtualization 버전 및 업데이트 상태
실시간 마이그레이션 탭	실시간 마이그레이션 제한 및 네트워크 설정
템플릿 프로젝트 탭	Red Hat 템플릿용 프로젝트.
사용자 권한 탭	클러스터 전체 사용자 권한.

4.1.4.1. 일반 탭

General (일반) 탭에 OpenShift Virtualization 버전 및 업데이트 상태가 표시됩니다.

예 **4.5.** 일반 탭

레이블	설명
서비스 이름	OpenShift Virtualization
공급자	Red Hat
설치된 버전	4.12.13
업데이트 상태	예 **:** 최신버전
채널	업데이트를 위해 선택된 채널입니다.

4.1.4.2. 실시간 마이그레이션 탭

실시간 마이그레이션 탭에서 실시간 마이그레이션 을 구성할 수 있습니다.

예 **4.6.** 실시간 마이그레이션 탭

요소	설명
max. 클러스터 필드당 마이그레이션	클러스터당 최대 실시간 마이그레이션 수를 선택합니다.
max. 노드 필드당 마이그레이션	노드당 최대 실시간 마이그레이션 수를 선택합니다.
실시간 마이그레이션 네트워크 목록	실시간 마이그레이션을 위해 전용 보조 네트워크를 선택합니다.

4.1.4.3. 템플릿 프로젝트 탭

템플릿 프로젝트 탭에서 템플릿용 프로젝트를 선택할 수 있습니다.

예 4.7. 템플릿 프로젝트 탭

요소	설명
프로젝트 목록	Red Hat 템플릿을 저장할 프로젝트를 선택합니다. 기본 템플릿 프로젝트는 **openshift** 입니다. 여러 템플릿 프로젝트를 정의하려면 각 프로젝트의 템플릿 페이지에 템플릿을 복제해야 합니다.

4.1.4.4. 사용자 권한 탭

User 권한 탭에는 작업에 대한 클러스터 전체 사용자 권한이 표시됩니다.

예 4.8. 사용자 권한 탭

요소	설명
사용자 권한 테이블	DestinationRule **템플릿 및 권한과 같은** 작업 목록입니다.

4.2. 카탈로그 페이지

카탈로그 페이지에서 템플릿을 선택하여 VirtualMachine을 생성할 수 있습니다.

예 4.9. 카탈로그 페이지

요소	설명
templates 프로젝트 목록	템플릿이 있는 프로젝트를 선택합니다. 기본적으로 Red Hat 템플릿은 **openshift** 프로젝트에 저장됩니다. **개요 → 설정 → 템플릿 프로젝트 탭에서 템플릿 프로젝트를** 편집할 수 있습니다.
모든 항목|기본 템플릿	**Default 템플릿을** 클릭하여 기본 템플릿만 표시합니다.
부팅 소스 사용 가능 확인란	확인란을 선택하여 사용 가능한 부팅 소스가 있는 템플릿을 표시합니다.
운영 체제 확인란	선택한 운영 체제가 있는 템플릿을 표시하려면 확인란을 선택합니다.
워크로드 확인란	선택한 워크로드가 있는 템플릿을 표시하려면 확인란을 선택합니다.

요소	설명
검색 필드	템플릿을 키워드로 검색합니다.
템플릿	템플릿 타일을 클릭하여 템플릿 세부 정보를 보고 VirtualMachine을 생성합니다.

4.3. VIRTUALMACHINES 페이지

VirtualMachines 페이지에서 **VirtualMachines**를 생성하고 관리할 수 있습니다.

예 **4.10. VirtualMachines** 페이지

요소	설명
생성 → 카탈로그에서	카탈로그 페이지에 VirtualMachine을 생성합니다.
YAML을 사용하여 → 생성	YAML 구성 파일을 편집하여 VirtualMachine을 생성합니다.
필터 필드	상태, 템플릿, 운영 체제 또는 노드별로 VirtualMachine을 필터링합니다.
검색 필드	이름 또는 라벨별로 VirtualMachines를 검색합니다.
VirtualMachines 테이블	VirtualMachines 목록입니다. VirtualMachine 옆에 있는 옵션 메뉴 ⋮ 를 클릭하여 **중지**,**재시작**,**일시 중지**,**복제**,**마이그레이션**,**SSH 명령 복사**,**라벨 편집**,**주석 편집** 또는 **삭제를** 선택합니다. VirtualMachine을 클릭하여 VirtualMachine 세부 정보 페이지로 이동합니다.

4.3.1. VirtualMachine 세부 정보 페이지

VirtualMachine 세부 정보 페이지에서 **VirtualMachine**을 구성할 수 있습니다.

예 **4.11. VirtualMachine** 세부 정보 페이지

요소	설명
작업 메뉴	**작업** 메뉴를 클릭하여 **중지**,**다시 시작**,**일시 중지**,**복제**,**마이그레이션**,**SSH 복사**,**레이블 편집**,**주석 편집** 또는 **삭제를** 선택합니다.
개요 탭	리소스 사용, 경고, 디스크 및 장치.

요소	설명
세부 정보 탭	VirtualMachine 구성입니다.
메트릭 탭	메모리, CPU, 스토리지, 네트워크 및 마이그레이션 지표.
YAML 탭	VirtualMachine YAML 구성 파일.
스케줄링 탭	구성 예약.
환경 탭	구성 맵, 시크릿 및 서비스 계정 관리.
이벤트 탭	VirtualMachine 이벤트 스트림
콘솔 탭	콘솔 세션 관리.
네트워크 인터페이스 탭	네트워크 인터페이스 관리.
디스크 탭	디스크 관리.
스크립트 탭	cloud-init 및 SSH 키 관리.
스냅샷 탭	스냅샷 관리.

4.3.1.1. 개요 탭

개요 탭에는 리소스 사용량, 경고 및 구성 정보가 표시됩니다.

예 4.12. 개요 탭

요소	설명
"자세한 내용" 타일	일반 VirtualMachine 정보.
"사용" 타일	**CPU**,**메모리**,**스토리지** 및 **네트워크 전송** 차트.
"하드웨어 장치" 타일	GPU 및 호스트 장치
"알erts" 타일	심각도별로 그룹화된 OpenShift Virtualization 경고.
"snapshots" 타일	**스냅샷** 및 **스냅샷** 테이블 가져오기.
"네트워크 인터페이스" 타일	**네트워크 인터페이스** 테이블.

요소	설명
"디스크" 타일	**디스크** 테이블.

4.3.1.2. 세부 정보 탭

세부 정보 탭에서 **VirtualMachine**을 구성할 수 있습니다.

예 **4.13.** 세부 정보 탭

요소	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
이름	VirtualMachine 이름입니다.
네임스페이스	VirtualMachine 네임스페이스입니다.
라벨	편집 아이콘을 클릭하여 레이블을 편집합니다.
주석	편집 아이콘을 클릭하여 주석을 편집합니다.
설명	편집 아이콘을 클릭하여 설명을 입력합니다.
운영 체제	운영 체제 이름입니다.
CPU|Memory	편집 아이콘을 클릭하여 CPU|Memory 요청을 편집합니다. CPU 수는 소켓 * 스레드 * 코어 수를 사용하여 계산합니다.
머신 유형	VirtualMachine 머신 유형.
부팅 모드	편집 아이콘을 클릭하여 부팅 모드를 편집합니다.
일시 중지 모드로 시작	편집 아이콘을 클릭하여 이 설정을 활성화합니다.
템플릿	VirtualMachine을 생성하는 데 사용되는 템플릿의 이름입니다.
작성 위치	VirtualMachine 생성 날짜.
소유자	VirtualMachine 소유자.
상태	VirtualMachine 상태.
Pod	**virt-launcher** Pod 이름입니다.

요소	설명
VirtualMachineInstance	VirtualMachineInstance 이름입니다.
부팅 순서	편집 아이콘을 클릭하여 부팅 소스를 선택합니다.
IP 주소	VirtualMachine의 IP 주소입니다.
호스트 이름	VirtualMachine의 호스트 이름입니다.
시간대	VirtualMachine의 시간대입니다.
노드	VirtualMachine이 실행 중인 노드입니다.
워크로드 프로필	편집 아이콘을 클릭하여 워크로드 프로필을 편집합니다.
virtctl을 사용한 SSH	복사 아이콘을 클릭하여 **virtctl ssh** 명령을 클립보드에 복사합니다.
NodePort를 통한 SSH 연결	**SSH 액세스를 위해 VirtualMachine을 노출할 서비스** 생성을 선택하면 **ssh -p <port>** 명령이 생성됩니다. 복사 아이콘을 클릭하여 명령을 클립보드에 복사합니다.
GPU 장치	편집 아이콘을 클릭하여 GPU 장치를 추가합니다.
호스트 장치	편집 아이콘을 클릭하여 호스트 장치를 추가합니다.
서비스 섹션	서비스를 확인합니다.
활성 사용자 섹션	활성 사용자 보기.

4.3.1.3. 지표 탭

Metrics 탭에는 메모리, **CPU**, 스토리지, 네트워크 및 마이그레이션 사용량 차트가 표시됩니다.

예 **4.14.** 지표 탭

요소	설명
시간 범위 목록	결과를 필터링할 시간 범위를 선택합니다.
가상화 대시보드	현재 프로젝트의 **워크로드** 탭에 연결합니다.
사용률 섹션	**메모리**,**CPU** 및 **네트워크 인터페이스** 차트.
스토리지 섹션	**스토리지 총 읽기/쓰기** 및 **스토리지 iops 총 읽기/쓰기** 차트입니다.

요소	설명
네트워크 섹션	**네트워크(Network Out)** 및 **네트워크 대역폭** 차트입니다.
마이그레이션 섹션	**마이그레이션** 및 **KV 데이터 전송 속도** 차트.

4.3.1.4. YAML 탭

YAML 탭에서 YAML 파일을 편집하여 VirtualMachine을 구성할 수 있습니다.

예 4.15. YAML 탭

요소	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
저장 버튼	YAML 파일에 대한 변경 사항을 저장합니다.
Reload 버튼	변경 사항을 삭제하고 YAML 파일을 다시 로드합니다.
취소 버튼	**YAML** 탭을 종료합니다.
다운로드 버튼	YAML 파일을 로컬 머신에 다운로드합니다.

4.3.1.5. 스케줄링 탭

스케줄링 탭에서 예약을 구성할 수 있습니다.

예 4.16. 스케줄링 탭

설정	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
노드 선택기	편집 아이콘을 클릭하여 대상 노드를 지정하는 라벨을 추가합니다.
허용 오차	편집 아이콘을 클릭하여 대상 노드를 지정하는 허용 오차를 추가합니다.
유사성 규칙	편집 아이콘을 클릭하여 선호도 규칙을 추가합니다.
Descheduler 스위치	Descheduler를 활성화하거나 비활성화합니다. Descheduler는 실행 중인 Pod를 제거하여 더 적합한 노드에 Pod를 다시 예약할 수 있습니다.

설정	설명
전용 리소스	편집 아이콘을 클릭하여 **전용 리소스를 사용하여 이 워크로드 예약을 선택합니다 (보장된 정책)**.
제거 전략	편집 아이콘을 클릭하여 VirtualMachineInstance 제거 전략으로 **LiveMigrate** 를 선택합니다.

4.3.1.6. 환경 탭

환경 탭에서 구성 맵, 시크릿 및 서비스 계정을 관리할 수 있습니다.

예 **4.17.** 환경 탭

요소	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
구성 맵, 시크릿 또는 서비스 계정 추가	링크를 클릭하고 리소스 목록에서 구성 맵, 시크릿 또는 서비스 계정을 선택합니다.

4.3.1.7. 이벤트 탭

이벤트 탭에는 **VirtualMachine** 이벤트 목록이 표시됩니다.

4.3.1.8. 콘솔 탭

콘솔 탭에서 **VirtualMachine**에 대한 콘솔 세션을 열 수 있습니다.

예 **4.18.** 콘솔 탭

요소	설명
게스트 로그인 인증 정보 섹션	**게스트 로그인 자격 증명을** 확장하여 **cloud-init** 로 생성된 인증 정보를 확인합니다. 복사 아이콘을 클릭하여 자격 증명을 클립보드에 복사합니다.
콘솔 목록	**VNC 콘솔** 또는 **직렬 콘솔을** 선택합니다. RDP(Remote **Desktop Protocol)를 사용하여 Windows VirtualMachines에 연결하도록 Desktop 뷰어를** 선택할 수 있습니다. 동일한 네트워크의 머신에 RDP 클라이언트를 설치해야 합니다.
키 목록 보내기	콘솔에 보낼 키 입력 조합을 선택합니다.

요소	설명
연결 해제 버튼	콘솔 연결을 끊습니다. 새 콘솔 세션을 여는 경우 콘솔 연결을 수동으로 연결 해제해야 합니다. 그렇지 않으면 첫 번째 콘솔 세션이 백그라운드에서 계속 실행됩니다.

4.3.1.9. 네트워크 인터페이스 탭

네트워크 인터페이스 탭에서 네트워크 인터페이스를 관리할 수 있습니다.

예 **4.19.** 네트워크 인터페이스 탭

설정	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
네트워크 인터페이스 추가 버튼	네트워크 인터페이스를 VirtualMachine에 추가합니다.
필터 필드	인터페이스 유형별로 필터링합니다.
검색 필드	이름 또는 라벨별로 네트워크 인터페이스를 검색합니다.
네트워크 인터페이스 테이블	네트워크 인터페이스 목록입니다. 네트워크 인터페이스 옆에 있는 옵션 메뉴 ⋮ 를 클릭하여 **편집** 또는 **삭제** 를 선택합니다.

4.3.1.10. 디스크 탭

디스크 탭에서 디스크를 관리할 수 있습니다.

예 **4.20.** 디스크 탭

설정	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
디스크 추가 버튼	VirtualMachine에 디스크를 추가합니다.
필터 필드	디스크 유형별로 필터링합니다.

설정	설명
검색 필드	이름으로 디스크를 검색합니다.
디스크 테이블	VirtualMachine 디스크 목록입니다. 디스크 옆에 있는 옵션 메뉴 ⋮ 를 클릭하여 **편집** 또는 분리를 **선택합니다**.
파일 시스템 테이블	VirtualMachine 파일 시스템 목록입니다.

4.3.1.11. 스크립트 탭

스크립트 탭에서 **VirtualMachine**의 **cloud-init** 및 **SSH** 키를 관리할 수 있습니다.

예 **4.21.** 스크립트 탭

요소	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
cloud-init	편집 아이콘을 클릭하여 cloud-init 설정을 편집합니다.
인증된 SSH 키	편집 아이콘을 클릭하여 새 보안을 생성하거나 기존 보안을 연결합니다.

4.3.1.12. 스냅샷 탭

스냅샷을 생성하고 **Snapshots** 탭의 스냅샷에서 **VirtualMachines**를 복원할 수 있습니다.

예 **4.22.** 스냅샷 탭

요소	설명
스냅샷 버튼 찍기	스냅샷을 생성합니다.
필터 필드	상태별로 스냅샷을 필터링합니다.
검색 필드	이름 또는 라벨별로 스냅샷을 검색합니다.

요소	설명
스냅샷 테이블	스냅샷 목록입니다. 스냅샷 옆에 있는 옵션 메뉴 ⋮ 를 클릭하여 **라벨 편집**,**주석 편집**, VirtualMachineSnapshot**편집**, **VirtualMachineSnapshot 삭제를** 선택합니다.

4.4. 템플릿 페이지

템플릿 페이지에서 **VirtualMachine** 템플릿을 생성, 편집, 복제할 수 있습니다.

참고

Red Hat 템플릿을 편집할 수 없습니다. **Red Hat** 템플릿을 복제하고 편집하여 사용자 지정 템플릿을 생성할 수 있습니다.

예 **4.23.** 템플릿 페이지

요소	설명
템플릿 생성 버튼	YAML 구성 파일을 편집하여 템플릿을 생성합니다.
필터 필드	유형, 부팅 소스, 템플릿 공급자 또는 운영 체제별로 템플릿을 필터링합니다.
검색 필드	이름 또는 레이블별로 템플릿을 검색합니다.
템플릿 테이블	템플릿 목록. 템플릿 옆에 있는 옵션 메뉴 ⋮ 를 클릭하여 **편집**,**복제**, 부팅 소스**편집**,**부팅 소스 참조 편집**,**라벨 편집**,**주석 편집** 또는 **삭제를** 선택합니다.

4.4.1. 템플릿 세부 정보 페이지

템플릿 설정을 보고 템플릿 세부 정보 페이지에서 사용자 지정 템플릿을 편집할 수 있습니다.

예 **4.24.** 템플릿 세부 정보 페이지

요소	설명
작업 메뉴	**작업** 메뉴를 클릭하여 **편집**,**복제**, 부팅 소스**편집**,**부팅 소스 참조 편집**,**레이블 편집**,**주석 편집** 또는 **삭제를** 선택합니다.

요소	설명
세부 정보 탭	템플릿 설정 및 구성입니다.
YAML 탭	YAML 구성 파일.
스케줄링 탭	구성 예약.
네트워크 인터페이스 탭	네트워크 인터페이스 관리.
디스크 탭	디스크 관리.
스크립트 탭	cloud-init, SSH 키, Sysprep 관리.
매개변수 탭	매개 변수.

4.4.1.1. 세부 정보 탭

세부 정보 탭에서 사용자 지정 템플릿을 구성할 수 있습니다.

예 **4.25.** 세부 정보 탭

요소	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
이름	템플릿 이름입니다.
네임스페이스	템플릿 네임스페이스입니다.
라벨	편집 아이콘을 클릭하여 레이블을 편집합니다.
주석	편집 아이콘을 클릭하여 주석을 편집합니다.
표시 이름	편집 아이콘을 클릭하여 표시 이름을 편집합니다.
설명	편집 아이콘을 클릭하여 설명을 입력합니다.
운영 체제	운영 체제 이름입니다.
CPU|Memory	편집 아이콘을 클릭하여 CPU|Memory 요청을 편집합니다. CPU 수는 소켓 * 스레드 * 코어 수를 사용하여 계산합니다.
머신 유형	템플릿 시스템 유형.

요소	설명
부팅 모드	편집 아이콘을 클릭하여 부팅 모드를 편집합니다.
기본 템플릿	이 템플릿을 생성하는 데 사용되는 기본 템플릿의 이름입니다.
작성 위치	템플릿 생성 날짜.
소유자	템플릿 소유자.
부팅 순서	템플릿 부팅 순서입니다.
부팅 소스	부팅 소스 가용성.
공급자	템플릿 공급자.
지원	템플릿 지원 수준입니다.
GPU 장치	편집 아이콘을 클릭하여 GPU 장치를 추가합니다.
호스트 장치	편집 아이콘을 클릭하여 호스트 장치를 추가합니다.

4.4.1.2. YAML 탭

YAML 탭에서 **YAML** 파일을 편집하여 사용자 정의 템플릿을 구성할 수 있습니다.

예 **4.26. YAML** 탭

요소	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
저장 버튼	YAML 파일에 대한 변경 사항을 저장합니다.
Reload 버튼	변경 사항을 삭제하고 YAML 파일을 다시 로드합니다.
취소 버튼	**YAML** 탭을 종료합니다.
다운로드 버튼	YAML 파일을 로컬 머신에 다운로드합니다.

4.4.1.3. 스케줄링 탭

스케줄링 탭에서 예약을 구성할 수 있습니다.

예 **4.27.** 스케줄링 탭

설정	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
노드 선택기	편집 아이콘을 클릭하여 대상 노드를 지정하는 라벨을 추가합니다.
허용 오차	편집 아이콘을 클릭하여 대상 노드를 지정하는 허용 오차를 추가합니다.
유사성 규칙	편집 아이콘을 클릭하여 선호도 규칙을 추가합니다.
Descheduler 스위치	Descheduler를 활성화하거나 비활성화합니다. Descheduler는 실행 중인 Pod를 제거하여 더 적합한 노드에 Pod를 다시 예약할 수 있습니다.
전용 리소스	편집 아이콘을 클릭하여 **전용 리소스를 사용하여 이 워크로드 예약을 선택합니다 (보장된 정책)**.
제거 전략	편집 아이콘을 클릭하여 VirtualMachineInstance 제거 전략으로 **LiveMigrate** 를 선택합니다.

4.4.1.4. 네트워크 인터페이스 탭

네트워크 인터페이스 탭에서 네트워크 인터페이스를 관리할 수 있습니다.

예 **4.28.** 네트워크 인터페이스 탭

설정	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
네트워크 인터페이스 추가 버튼	네트워크 인터페이스를 템플릿에 추가합니다.
필터 필드	인터페이스 유형별로 필터링합니다.
검색 필드	이름 또는 라벨별로 네트워크 인터페이스를 검색합니다.
네트워크 인터페이스 테이블	네트워크 인터페이스 목록입니다. 네트워크 인터페이스 옆에 있는 옵션 메뉴 ⋮ 를 클릭하여 **편집** 또는 **삭제** 를 선택합니다.

4.4.1.5. 디스크 탭

디스크 탭에서 디스크를 관리할 수 있습니다.

예 4.29. 디스크 탭

설정	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
디스크 추가 버튼	템플릿에 디스크를 추가합니다.
필터 필드	디스크 유형별로 필터링합니다.
검색 필드	이름으로 디스크를 검색합니다.
디스크 테이블	템플릿 디스크 목록. 디스크 옆에 있는 옵션 메뉴 ⋮ 를 클릭하여 **편집** 또는 분리를 **선택합니다**.

4.4.1.6. 스크립트 탭

Scripts 탭에서 **cloud-init** 설정, **SSH** 키 및 **Sysprep** 응답 파일을 관리할 수 있습니다.

예 4.30. 스크립트 탭

요소	설명
YAML 스위치	YAML 구성 파일의 실시간 변경 사항을 보려면 **ON** 으로 설정합니다.
cloud-init	편집 아이콘을 클릭하여 cloud-init 설정을 편집합니다.
인증된 SSH 키	편집 아이콘을 클릭하여 새 보안을 생성하거나 기존 보안을 연결합니다.
sys	편집 아이콘을 클릭하여 **Autounattend.xml** 또는 **Unattend.xml** 응답 파일을 업로드하여 Windows VirtualMachine 설정을 자동화합니다.

4.4.1.7. 매개변수 탭

Parameters 탭에서 선택한 템플릿 설정을 편집할 수 있습니다.

예 4.31. 매개변수 탭

요소	설명
VM 이름	생성된 값으로 **생성됨(expression)** 을 선택하고 **Value** 를 선택하여 기본값을 설정하거나 **Default 값 유형** 목록에서 **None** 을 선택합니다.
데이터 소스 네임스페이스	생성된 값으로 **생성됨(expression)** 을 선택하고 **Value** 를 선택하여 기본값을 설정하거나 **Default 값 유형** 목록에서 **None** 을 선택합니다.
클라우드 사용자 암호	생성된 값으로 **생성됨(expression)** 을 선택하고 **Value** 를 선택하여 기본값을 설정하거나 **Default 값 유형** 목록에서 **None** 을 선택합니다.

4.5. 데이터 소스 페이지

DataSources 페이지에서 **VirtualMachine** 부팅 소스에 대한 **DataSources**를 생성하고 구성할 수 있습니다.

DataSource를 생성할 때 자동 부팅 소스 업데이트를 비활성화하지 않는 한 **DataImportCron** 리소스는 폴링하고 디스크 이미지를 가져올 **cron** 작업을 정의합니다.

예 **4.32.** 데이터 소스 페이지

요소	설명
DataSource → 양식 생성	레지스트리 URL, 디스크 크기, 버전 수, cron 표현식을 양식으로 입력하여 DataSource를 생성합니다.
YAML을 사용하여 DataSources → YAML 생성	YAML 구성 파일을 편집하여 DataSource를 생성합니다.
필터 필드	사용 가능한 DataImportCron과 같은 속성에 따라 DataSources를 필터링합니다.
검색 필드	이름 또는 라벨별로 DataSource를 검색합니다.
데이터 소스 테이블	데이터 소스 목록. DataSource 옆에 있는 옵션 메뉴 ⋮ 를 클릭하여 **라벨 편집**,**주석 편집** 또는 **삭제를** 선택합니다.

DataSource를 클릭하여 **DataSource** 세부 정보 페이지를 확인합니다.

4.5.1. 데이터 소스 세부 정보 페이지

DataSource 세부 정보 페이지에서 **DataSource**를 구성할 수 있습니다.

예 **4.33.** 데이터 소스 세부 정보 페이지

요소	설명
세부 정보 탭	양식을 편집하여 DataSource를 구성합니다.
YAML 탭	YAML 구성 파일을 편집하여 DataSource를 구성합니다.
작업 메뉴	**레이블 편집**,**주석 편집** 또는 **삭제**를 선택합니다.
이름	데이터 소스 이름입니다.
네임스페이스	데이터 소스 네임스페이스.
라벨	편집 아이콘을 클릭하여 레이블을 편집합니다.
주석	편집 아이콘을 클릭하여 주석을 편집합니다.
조건	DataSource의 상태 조건을 표시합니다.

4.6. MIGRATIONPOLICIES 페이지

MigrationPolicies 페이지에서 워크로드의 **MigrationPolicies**를 관리할 수 있습니다.

예 **4.34. MigrationPolicies** 페이지

요소	설명
MigrationPolicy → 양식 생성	구성 및 레이블을 양식에 입력하여 MigrationPolicy를 생성합니다.
MigrationPolicy → YAML을 사용하여 생성	YAML 구성 파일을 편집하여 MigrationPolicy를 생성합니다.
이름 | 라벨 검색 필드	이름 또는 라벨별로 MigrationPolicy를 검색합니다.
MigrationPolicies 테이블	MigrationPolicies 목록. MigrationPolicy 옆에 있는 옵션 메뉴 ⋮ 를 클릭하여 **편집** 또는 **삭제** 를 선택합니다.

MigrationPolicy를 클릭하여 **MigrationPolicy** 세부 정보 페이지를 확인합니다.

4.6.1. MigrationPolicy 세부 정보 페이지

MigrationPolicy 세부 정보 페이지에서 **MigrationPolicy**를 구성할 수 있습니다.

예 **4.35. MigrationPolicy** 세부 정보 페이지

요소	설명
세부 정보 탭	양식을 편집하여 MigrationPolicy를 구성합니다.
YAML 탭	YAML 구성 파일을 편집하여 MigrationPolicy를 구성합니다.
작업 메뉴	**편집** 또는 **삭제를** 선택합니다.
이름	MigrationPolicy 이름입니다.
설명	MigrationPolicy 설명.
구성	편집 아이콘을 클릭하여 MigrationPolicy 구성을 업데이트합니다.
마이그레이션당 대역폭	마이그레이션당 대역폭 요청입니다. 무제한 대역폭의 경우 값을 **0** 으로 설정합니다.
자동 통합	자동 통합 정책.
복사 후	복사 후 정책입니다.
완료 제한	완료 시간(초)입니다.
프로젝트 라벨	**Edit** (편집)를 클릭하여 프로젝트 레이블을 편집합니다.
VirtualMachine 라벨	**편집** 을 클릭하여 VirtualMachine 레이블을 편집합니다.

5장. OPENSHIFT VIRTUALIZATION 릴리스 정보

5.1. 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.

5.2. 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(Container Network Interface) 네트워크 공급자와 함께 OpenShift Virtualization을 사용할 수 있습니다.

OpenShift Virtualization으로 수행할 수 있는 작업에 대해 자세히 알아보십시오.

OpenShift Virtualization 아키텍처 및 배포에 대해 자세히 알아보십시오.

OpenShift Virtualization을 위한 클러스터를 준비합니다.

5.2.1. OpenShift Virtualization 지원 클러스터 버전

OpenShift Container Platform 4.12 클러스터에서 사용할 수 있도록 OpenShift Virtualization 4.12가 지원됩니다. OpenShift Virtualization의 최신 z-stream 릴리스를 사용하려면 먼저 최신 버전의 OpenShift Container Platform으로 업그레이드해야 합니다.

5.2.2. 지원되는 게스트 운영 체제

OpenShift Virtualization에서 지원되는 게스트 운영 체제를 보려면 Certified Guest Operating Systems in Red Hat OpenStack Platform, Red Hat Virtualization 및 OpenShift Virtualization 을 참조하십시오.

5.3. 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 Virtualization에서는 더 이상 로고를 사용하지 않습니다. OpenShift Virtualization은 이제 버전 4.9 이상에 대한 로고로 표시됩니다.
- virtctl memory-dump 명령을 사용하여 포렌식 분석에 대한 VM 메모리 덤프를 생성할 수 있습니다.
- virtctl vm export 명령을 사용하거나 VirtualMachineExport 사용자 정의 리소스를 생성하여 다른 클러스터 또는 동일한 클러스터의 다른 네임스페이스에서 VM(가상 머신), VM 스냅샷 또는 PVC(영구 볼륨 클레임)에서 볼륨을 내보내고 다운로드할 수 있습니다. 또한 법의학 분석을 위해 메모리 덤프를 내보낼 수 있습니다.
- 웹 콘솔 개요 설명서를 참조하여 OpenShift Virtualization 웹 콘솔의 기능 및 조직에 대해 알아볼 수 있습니다.
- virtctl ssh 명령을 사용하여 로컬 SSH 클라이언트를 사용하거나 OpenShift Container Platform 웹 콘솔에서 SSH 명령을 복사하여 가상 머신에 SSH트래픽을 전달할 수 있습니다.
- 독립 실행형 데이터 볼륨과 dataVolumeTemplate 을 사용하여 VM용 디스크를 준비할 때 생성되는 데이터 볼륨은 더 이상 시스템에 저장되지 않습니다. 이제 PVC가 생성된 후 데이터 볼륨이 자동으로 수집되고 삭제됩니다.
- OpenShift Virtualization은 이제 OpenShift Container Platform 모니터링 대시보드를 사용하여 액세스할 수 있는 실시간 마이그레이션 메트릭 을 제공합니다.
- OpenShift Virtualization Operator는 이제 APIServer 사용자 정의 리소스에서 클러스터 전체 TLS 보안 프로필 을 읽고 가상화, 스토리지, 네트워킹 및 인프라를 포함하여 OpenShift Virtualization 구성 요소로 전달합니다.
- OpenShift Virtualization에는 경고를 트리거하는 문제를 해결하는 데 도움이 되는 runbooks 가 있습니다. 웹 콘솔의 가상화 → 개요 페이지에 경고가 표시됩니다. 각 runbook은 경고를 정의하고 문제를 진단하고 해결하는 단계를 제공합니다. 이 기능은 이전에 기술 프리뷰로 소개되었으며 현재 일반적으로 사용 가능합니다.

5.3.1. 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 virtualization 키워드를 입력하여 사용 가능한 둘러보기를 필터링할 수 있습니다.

5.3.2. 네트워킹

- 이제 OpenShift Container Platform 클러스터 점검을 실행할 네임스페이스를 지정할 수 있습니다.
- 이제 계층 2 모드에서 MetalLB Operator 를 사용하여 로드 밸런싱 서비스를 구성할 수 있습니다.

5.3.3. 웹 콘솔

- 가상화 → 개요 페이지에는 다음과 같은 사용성이 향상되었습니다.
  - Download virtctl 링크를 사용할 수 있습니다.

  - 리소스 정보는 관리자 및 관리자가 아닌 사용자를 위해 사용자 지정됩니다. 예를 들어 관리자가 아닌 사용자는 VM만 볼 수 있습니다.
  - 개요 탭에는 지난 7일의 추세를 보여주는 차트가 포함된 VM 수 및 vCPU, 메모리, 스토리지 사용량이 표시됩니다.
  - 개요 탭의 경고 카드에는 심각도별로 그룹화된 경고가 표시됩니다.
  - Top Consumers 탭에는 구성 가능한 기간 동안 CPU, 메모리 및 스토리지 사용량의 상위 소비자가 표시됩니다.
  - Migrations (마이그레이션) 탭에 VM 마이그레이션의 진행 상황이 표시됩니다.
  - 설정 탭에는 실시간 마이그레이션 제한, 실시간 마이그레이션 네트워크 및 templates 프로젝트를 포함한 클러스터 전체 설정이 표시됩니다.
- 가상화 → MigrationPolicies 페이지의 단일 위치에서 실시간 마이그레이션 정책을 생성하고 관리할 수 있습니다.
- VirtualMachine details 페이지의 Metrics 탭에는 구성 가능한 기간 동안 VM의 메모리, CPU, 스토리지, 네트워크 및 마이그레이션 메트릭이 표시됩니다.
- VM을 생성하도록 템플릿을 사용자 지정할 때 각 VM 구성 탭에서 YAML 스위치를 ON 으로 설정하여 양식과 함께 YAML 구성 파일의 실시간 변경 사항을 볼 수 있습니다.
- 가상화 → 개요 페이지 의 마이그레이션 탭에 구성 가능한 기간 동안 가상 머신 인스턴스 마이그레이션 진행 상황이 표시됩니다.
- 테넌트 워크로드 중단을 최소화하기 위해 실시간 마이그레이션을 위해 전용 네트워크를 정의할 수 있습니다. 네트워크를 선택하려면 가상화 → 개요 → 설정 → 실시간 마이그레이션 로 이동합니다.

5.3.4. 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.

5.3.5. 삭제된 기능

제거된 기능은 현재 릴리스에서 지원되지 않습니다.

- 레거시 HPP 사용자 지정 리소스 및 관련 스토리지 클래스에 대한 지원이 모두 새로운 배포에 대해 제거되었습니다. OpenShift Virtualization 4.12에서 HPP Operator는 Kubernetes CSI(Container Storage Interface) 드라이버를 사용하여 로컬 스토리지를 구성합니다. 레거시 HPP 사용자 지정 리소스는 이전 버전의 OpenShift Virtualization에 설치된 경우에만 지원됩니다.
- OpenShift Virtualization 4.11에서는 다음 오브젝트를 포함하여 nmstate 에 대한 지원을 제거했습니다.
  - **NodeNetworkState**
  - **NodeNetworkConfigurationPolicy**
  - **NodeNetworkConfigurationEnactment**

  기존 nmstate 구성을 유지하고 지원하려면 OpenShift Virtualization 4.11로 업데이트하기 전에 Kubernetes NMState Operator 를 설치합니다. EUS(Extended Update Support) 버전의 4.12의 경우 4.12로 업데이트한 후 Kubernetes NMState Operator를 설치합니다. OpenShift Container

Platform 웹 콘솔의 OperatorHub 에서 또는 OpenShift CLI(oc)를 사용하여 Operator를 설치할 수 있습니다.

- Node Maintenance Operator (NMO)는 더 이상 OpenShift Virtualization과 함께 제공되지 않습니다. OpenShift Container Platform 웹 콘솔의 OperatorHub 에서 또는 OpenShift CLI(oc)를 사용하여 NMO를 설치할 수 있습니다.
  OpenShift Virtualization 4.10.2 이상 릴리스에서 OpenShift Virtualization 4.11로 업데이트하기 전에 다음 작업 중 하나를 수행해야 합니다. EUS (Extended Update Support) 버전의 경우 4.10.2 이상 4.10 릴리스에서 OpenShift Virtualization 4.12로 업데이트하기 전에 다음 작업을 수행해야 합니다.
  - 모든 노드를 유지보수 모드에서 이동합니다.
  - 독립 실행형 NMO를 설치하고 **nodemaintenances.nodemaintenance.kubevirt.io** CR(사용자 정의 리소스)을 **nodemaintenances.nodemaintenance.medik8s.io** CR로 교체합니다.

5.4. 기술 프리뷰 기능

이 릴리스의 일부 기능은 현재 기술 프리뷰 단계에 있습니다. 이러한 실험적 기능은 프로덕션용이 아닙니다. 해당 기능은 Red Hat Customer Portal의 지원 범위를 참조하십시오.

기술 프리뷰 기능 지원 범위

- OpenShift Container Platform 클러스터 점검을 실행하여 VM 간 네트워크 대기 시간을 측정할 수 있습니다.
- Tekton Tasks Operator(TTO)는 이제 OpenShift Virtualization과 Red Hat OpenShift Pipelines 를 통합합니다. TTO에는 다음을 수행할 수 있는 클러스터 작업 및 예제 파이프라인이 포함되어 있습니다.
  - VM(가상 머신), PVC(영구 볼륨 클레임) 및 데이터 볼륨을 생성하고 관리합니다.
  - VM에서 명령을 실행합니다.
  - **libguestfs** 툴을 사용하여 디스크 이미지를 조작합니다.
  - Windows 10을 Windows 설치 이미지(ISO 파일)에서 새 데이터 볼륨에 설치합니다.
  - 기본 Windows 10 설치를 사용자 지정한 다음 새 이미지와 템플릿을 만듭니다.
- 게스트 에이전트 ping 프로브를 사용하여 QEMU 게스트 에이전트가 가상 머신에서 실행 중인지 확인할 수 있습니다.
- 이제 Microsoft Windows 11을 게스트 운영 체제로 사용할 수 있습니다. 그러나 OpenShift Virtualization 4.12는 USB 디스크를 지원하지 않습니다.이 디스크는ECDHE 복구의 중요한 기능에 필요합니다. 복구 키를 보호하려면ECDHE 복구 가이드에 설명된 다른 방법을 사용하십시오.
- 대역폭 사용량, 최대 병렬 마이그레이션 및 시간 초과와 같은 특정 매개 변수로 실시간 마이그레이션 정책을 생성하고 가상 머신 및 네임스페이스 레이블을 사용하여 가상 머신 그룹에 정책을 적용할 수 있습니다.

5.5. 버그 수정

- 드라이버 설치 후 새 장치 구성을 손실하지 않고 드라이버를 설치하기 전에 중재 장치를 사용하도록 **HyperConverged** CR을 구성할 수 있습니다. (BZ#2046298)

- **NodePort** 서비스를 많이 생성하는 경우 **OVN-Kubernetes** 클러스터 네트워크 공급자가 더 이상 최대 **RAM** 및 **CPU** 사용량에서 충돌하지 않습니다. (OCPBUGS-1940)
- **Red Hat Ceph Storage** 또는 **Red Hat OpenShift Data Foundation Storage**를 사용하면 한 번에 100개 이상의 **VM** 복제가 더 이상 간헐적으로 실패하지 않습니다. (BZ#1989527)

5.6. 확인된 문제

- 단일 스택 **IPv6** 클러스터에서는 **OpenShift Virtualization**을 실행할 수 없습니다. (BZ#2193267)
- 다른 컴퓨팅 노드가 있는 이기종 클러스터에서는 **TSC(Timest-counter scaling)**를 지원하지 않거나 적절한 **TSC** 빈도를 갖는 노드에 **HyperV Reenlightenment**가 활성화된 가상 머신을 예약할 수 없습니다. (BZ#2151169)
- 다른 **SELinux** 컨텍스트가 있는 두 개의 **Pod**를 사용하면 **ocs-storagecluster-cephfs** 스토리지 클래스가 있는 **VM**이 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 Storage**를 사용하는 클러스터에서 **VM**을 실시간 마이그레이션합니다.
- **OpenShift Virtualization 4.12**에서 **TopoLVM** 이름 문자열이 변경되었습니다. 결과적으로 다음 오류 메시지와 함께 운영 체제 이미지의 자동 가져오기가 실패할 수 있습니다(BZ#2158521).

  > **DataVolume.storage spec is missing accessMode and volumeMode, cannot get access mode from StorageProfile.**

  - 해결 방법:

    1. 스토리지 프로파일의 **claimPropertySets** 배열을 업데이트합니다.

    ```
    $ oc patch storageprofile <storage_profile> --type=merge -p '{"spec": {"claimPropertySets": [{"accessModes": ["ReadWriteOnce"], "volumeMode": "Block"}, \
      {"accessModes": ["ReadWriteOnce"], "volumeMode": "Filesystem"}]}}'
    ```

    2. **openshift-virtualization-os-images** 네임스페이스에서 영향을 받는 데이터 볼륨을 삭제합니다. 업데이트된 스토리지 프로필에서 액세스 모드 및 볼륨 모드로 다시 생성됩니다.
- 바인딩 모드가 **WaitForFirstConsumer** 인 스토리지에 대한 **VM** 스냅샷을 복원할 때 복원된 **PVC**는 **Pending** 상태로 유지되고 복원 작업이 진행되지 않습니다.
  - 이 문제를 해결하려면 복원된 **VM**을 시작한 다음 중지한 다음 다시 시작합니다. **VM**이 예약되고 **PVC**가 **Bound** 상태가 되고 복원 작업이 완료됩니다. (BZ#2149654)
- **SNO(Single Node OpenShift)** 클러스터에서 공통 템플릿으로 생성된 **VM**은 템플릿의 기본 제거 전략이 **LiveMigrate** 이므로 **VMCannotBeEvicted** 경고를 표시합니다. **VM**의 제거 전략을 업데이트하여 이 경고를 무시하거나 경고를 제거할 수 있습니다. (BZ#2092412)
- **OpenShift Virtualization**을 설치 제거해도 **OpenShift Virtualization**에서 생성한 **feature.node.kubevirt.io** 노드 레이블은 제거되지 않습니다. 라벨을 수동으로 제거해야 합니다. (CNV-22036)
- **CDI(Containerized Data Importer)**로 생성된 일부 **PVC**(영구 볼륨 클레임) 주석은 가상 머신 스냅샷 복원 작업이 무기한 중단될 수 있습니다. (BZ#2070366)

- 이 문제를 해결하려면 주석을 수동으로 제거할 수 있습니다.

  1. VirtualMachineSnapshotContent CR(사용자 정의 리소스) 이름을 **VirtualMachineSnapshot** CR CR의 **status.virtualMachineSnapshotContentName** 값에서 가져옵니다.

  2. **VirtualMachineSnapshotContent** CR을 편집하고 **k8s.io/cloneRequest** 가 포함된 모든 행을 제거합니다.

  3. **VirtualMachine** 오브젝트에서 **spec.dataVolumeTemplates** 값을 지정하지 않은 경우 다음 두 조건이 모두 **true**인 이 네임스페이스의 모든 **DataVolume** 및 **PersistentVolumeClaim** 오브젝트를 삭제합니다.

     a. 오브젝트의 이름은 **restore**로 시작합니다.

     b. 가상 머신에서 오브젝트를 참조하지 않습니다. **spec.dataVolumeTemplates** 에 값을 지정한 경우 이 단계는 선택 사항입니다.

  4. 업데이트된 **VirtualMachineSnapshot** CR을 사용하여 복원 작업을 반복합니다.

- **Windows 11** 가상 머신은 FIPS 모드에서 실행되는 클러스터에서 부팅되지 않습니다. **Windows 11**에는 기본적으로 **TPM**(신뢰할 수 있는 플랫폼 모듈) 장치가 필요합니다. 그러나 **swtpm** (software **TPM** 에뮬레이터) 패키지는 **FIPS**와 호환되지 않습니다. (BZ#2089301)

- **OpenShift Container Platform** 클러스터에서 **OVN-Kubernetes**를 기본 **CNI**(Container Network Interface) 공급자로 사용하는 경우 **OVN-Kubernetes**의 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.

- 경우에 따라 여러 가상 머신이 동일한 **PVC**를 읽기-쓰기 모드로 마운트할 수 있으므로 데이터가 손상될 수 있습니다. (BZ#1992753)

  - 이 문제를 해결하려면 여러 **VM**에서 읽기-쓰기 모드에서 단일 **PVC**를 사용하지 마십시오.

- **Pod Disruption Budget**(**PDB**)은 **Pod**가 모호한 가상 머신 이미지에 대한 **Pod** 중단을 방지합니다. **PDB**에서 **Pod** 중단을 감지하면 **openshift-monitoring** 는 **LiveMigrate** 제거 전략을 사용하는 가상 머신 이미지에 대해 60분마다 **PodDisruptionBudgetAtLimit** 경고를 보냅니다. (BZ#2026733)

  - 해결 방법으로 경고는 음소거 합니다.

- **OpenShift Virtualization**은 **Pod**에서 사용하는 서비스 계정 토큰을 해당 특정 포드에 연결합니다. **OpenShift Virtualization**은 토큰이 포함된 디스크 이미지를 생성하여 서비스 계정 볼륨을 구현합니다. **VM**을 마이그레이션하는 경우 서비스 계정 볼륨이 잘못되었습니다. (BZ#2037611)

  - 이 문제를 해결하려면 사용자 계정 토큰이 특정 **Pod**에 바인딩되지 않으므로 서비스 계정 대신 사용자 계정을 사용하십시오.

- **csi-clone** 복제 전략을 사용하여 **VM** 100개를 복제하는 경우 **Ceph CSI**에서 복제본을 제거하지 못할 수 있습니다. 복제본을 수동으로 삭제하면 실패할 수도 있습니다. (BZ#2055595)

  - 이 문제를 해결하려면 **ceph-mgr** 을 다시 시작하여 **VM** 복제본을 제거할 수 있습니다.

- 블록 스토리지 장치와 함께 **LVM**(**Logical Volume Management**)을 사용하는 **VM**에는 **RHCOS**(**Red Hat Enterprise Linux CoreOS**) 호스트와의 충돌을 방지하기 위해 추가 구성이 필요합니다.

- 이 문제를 해결하려면 VM을 생성하고 LVM을 프로비저닝한 후 VM을 다시 시작할 수 있습니다. 이렇게 하면 빈 **system.lvmdevices** 파일이 생성됩니다. (OCPBUGS-5223)

6장. 설치

6.1. 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 Platform**을 배포할 수 있습니다. 그러나 설치 방법과 클러스터 토폴로지는 스냅샷 또는 실시간 마이그레이션과 같은 **OpenShift Virtualization** 기능에 영향을 줄 수 있습니다.

FIPS 모드

FIPS 모드에서 클러스터를 설치하는 경우 **OpenShift Virtualization**에 추가 설정이 필요하지 않습니다.

IPv6

단일 스택 **IPv6** 클러스터에서는 **OpenShift Virtualization**을 실행할 수 없습니다. (BZ#2193267)

6.1.1. 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 인스턴스에 **OpenShift Virtualization**을 설치하거나 **IBM Cloud Bare Metal Server**에 설치하는 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.

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

CPU 요구사항

- **RHEL (Red Hat Enterprise Linux) 8**에서 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.

- 클러스터에서 다른 CPU가 있는 작업자 노드를 사용하는 경우 다른 CPU의 기능이 다르기 때문에 실시간 마이그레이션 오류가 발생할 수 있습니다. 이러한 오류를 방지하려면 각 노드에 적절한 용량이 있는 CPU를 사용하고 가상 머신에 노드 선호도를 설정하여 마이그레이션을 성공적으로 수행합니다. 자세한 내용은 필수 노드 유사성 규칙 구성을 참조하십시오.

추가 리소스

- RHCOS 정보.
- 지원되는 CPU를 위한 Red Hat Ecosystem Catalog.
- 지원되는 스토리지.

6.1.2. 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.

6.1.2.1. 메모리 오버헤드

아래 식을 사용하여 **OpenShift Virtualization**의 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 **2179MiB**의 **RAM**이 필요합니다.

가상 머신 메모리 오버헤드

```
Memory overhead per virtual machine ≈ (1.002 × requested memory) \
        + 218 MiB \ 1
        + 8 MiB × (number of vCPUs) \ 2
        + 16 MiB × (number of graphics devices) \ 3
        + (additional memory overhead) 4
```

1. **virt-launcher Pod**에서 실행되는 프로세스에 필요합니다.
2. 가상 머신에서 요청한 가상 **CPU** 수입니다.
3. 가상 머신에서 요청한 가상 그래픽 카드 수입니다.
4. 추가 메모리 오버헤드:
    - 환경에 **SR-IOV(Single Root I/O Virtualization)** 네트워크 장치 또는 **GPU(Graphics Processing Unit)**가 포함된 경우 각 장치에 대해 **1GiB**의 추가 메모리 오버헤드를 할당합니다.

6.1.2.2. CPU 오버헤드

아래 식을 사용하여 **OpenShift Virtualization**에 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 (**4000**밀리코어)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 **2**개의 추가 코어(**2000**밀리코어)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.

6.1.2.3. 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.

6.1.2.4. 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.

6.1.3. 오브젝트 최대값

클러스터를 계획할 때 다음과 같은 테스트된 오브젝트 최대값을 고려해야 합니다.

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

6.1.4. 제한된 네트워크 환경

인터넷 연결이 없는 제한된 환경에 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 구성할 수 있습니다.

6.1.5. 실시간 마이그레이션

실시간 마이그레이션에는 다음과 같은 요구 사항이 있습니다.

- RWX(ReadWriteMany) 액세스 모드를 사용한 공유 스토리지.
- 충분한 RAM 및 네트워크 대역폭.
- 가상 머신에서 호스트 모델 CPU를 사용하는 경우 노드에서 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.

6.1.6. 스냅샷 및 복제

스냅샷 및 복제 요구 사항은 OpenShift Virtualization 스토리지 기능을 참조하십시오.

6.1.7. 클러스터 고가용성 옵션

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

- 머신 상태 점검을 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 머신 의 RunStrategies 정보를 참조하십시오.

- OpenShift Container Platform 클러스터에서 Node Health Check Operator를 사용하여 NodeHealthCheck 컨트롤러를 배포하면 IPI 및 비 IPI에 대한 자동 고가용성을 사용할 수 있습니다. 컨트롤러는 비정상 노드를 식별하고 Self Node Remediation Operator 또는 Fence Agents Remediation Operator와 같은 수정 공급자를 사용하여 비정상 노드를 수정합니다. 노드 수정, 펜싱 및 유지 관리에 대한 자세한 내용은 Workload Availability for Red Hat OpenShift설명서를 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.

6.2. 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.

6.2.1. 가상화 구성 요소를 위한 노드 배치 정보

다음이 수행되도록 **OpenShift Virtualization**이 구성 요소를 배포하는 위치를 사용자 지정하는 것이 좋습니다.

- 가상 머신은 가상화 워크로드를 위한 노드에만 배포됩니다.
- **Operator**는 인프라 노드에만 배포됩니다.
- 특정 노드는 **OpenShift Virtualization**의 영향을 받지 않습니다. 예를 들어, 클러스터에서 실행되는 가상화와 관련이 없는 워크로드가 있으며 해당 워크로드가 **OpenShift Virtualization**과 격리되기를 원하는 경우가 이에 해당합니다.

6.2.1.1. 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 Operator**의 경우, **OLM** 서브스크립션 오브젝트를 직접 편집합니다. 현재는 웹 콘솔을 사용하여서브스크립션 오브젝트에 대한 노드 배치 규칙을 구성할 수 없습니다.
- **OpenShift Virtualization Operator**가 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 **Pod**를 실행할 수 없습니다.

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

nodeSelector

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

유사성

더 많은 표현 구문을 사용하여 노드와 **Pod**의 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 **Pod**를 예약할 수 있습니다.

허용 오차

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

6.2.1.2. 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.v4.12.13
  channel: "stable"
  config: (1)
```

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

6.2.1.3. 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 포함할 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.

6.2.1.4. 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.

6.2.1.5. 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
- **CLI**를 사용한 **OpenShift Virtualization** 설치
- 웹 콘솔을 사용한 **OpenShift Virtualization** 설치
- 가상 머신 로컬 스토리지 구성

6.2.2. 예시 매니페스트

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

6.2.2.1. Operator Lifecycle Manager 서브스크립션 오브젝트

6.2.2.1.1. 예: OLM 서브스크립션 오브젝트에서 nodeSelector를 사용한 노드 배치

이 예에서는 **example.io/example-infra-key = example-infra-value**로 라벨이 지정된 노드에 **OLM**이 **OpenShift Virtualization Operator**를 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.v4.12.13
  channel: "stable"
  config:
    nodeSelector:
      example.io/example-infra-key: example-infra-value
```

–

6.2.2.1.2. 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.v4.12.13
  channel: "stable"
  config:
    tolerations:
    - key: "key"
      operator: "Equal"
      value: "virtualization"
      effect: "NoSchedule"
```

6.2.2.2. HyperConverged 오브젝트

6.2.2.2.1. 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
```

6.2.2.2.2. 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
```

6.2.2.2.3. 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"
```

6.2.2.3. HostPathProvisioner 오브젝트

6.2.2.3.1. 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
```

6.3. 웹 콘솔을 사용한 OPENSHIFT VIRTUALIZATION 설치

OpenShift Virtualization을 설치하여 OpenShift Container Platform 클러스터에 가상화 기능을 추가합니다.

OpenShift Container Platform 4.12 웹 콘솔 을 사용하여 OpenShift Virtualization Operator를 구독하고 배포할 수 있습니다.

6.3.1. OpenShift Virtualization Operator 설치

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

사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.12를 설치합니다.
- OpenShift Container Platform 웹 콘솔에 **cluster-admin** 권한이 있는 사용자로 로그인합니다.

프로세스

1. 관리자로 Operator → OperatorHub를 클릭합니다.
2. 키워드로 필터링 필드에 가상화를 입력합니다.
3. Red Hat 소스 라벨을 사용하여 **{CNVOperator CryostatName}** 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 **Operator**가 설치되고, 해당 네임스페이스가 존재하지 않는 경우 자동으로 생성됩니다.

> 주의
>
> **openshift-cnv** 이외의 네임스페이스에 **OpenShift Virtualization Operator**를 설치하려고 하면 설치가 실패합니다.

c. 승인 전략의 경우 기본값인 자동을 선택하여 **OpenShift Virtualization**이 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 Virtualization**을 지원하므로 누락된 **OpenShift Virtualization** 업데이트가 없으면 클러스터가 지원되지 않을 수 있습니다.

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

검증

- 워크로드 → **Pods** 페이지로 이동하여 모두실행 중 상태가 될 때까지 **OpenShift Virtualization Pod**를 모니터링합니다. 모든 **Pod**에 실행 중 상태가 표시되면 **OpenShift Virtualization**을 사용할 수 있습니다.

6.3.2. 다음 단계

다음 구성 요소를 추가로 구성하는 것이 좋습니다.

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

6.4. 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.

6.4.1. 사전 요구 사항

- 클러스터에 OpenShift Container Platform 4.12를 설치합니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 로그인합니다.

6.4.2. 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.v4.12.13
  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 매개변수를 구성할 수 있습니다.

6.4.3. 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.v4.12.13   OpenShift Virtualization   4.12.13
Succeeded
```

6.4.4. 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.

6.5. VIRTCTL 클라이언트 설치

virtctl 클라이언트는 OpenShift Virtualization 리소스를 관리하는 명령줄 유틸리티입니다. Linux, Windows 및 macOS에서 사용할 수 있습니다.

6.5.1. Linux, Windows 및 macOS에 virtctl 클라이언트 설치

운영 체제에 대한 **virtctl** 클라이언트를 다운로드하여 설치합니다.

절차

1. OpenShift Container Platform 웹 콘솔에서 **Virtualization > Overview** 로 이동합니다.
2. 페이지 오른쪽 상단에 있는 **Download virtctl** 링크를 클릭하고 운영 체제용 **virtctl** 클라이언트를 다운로드합니다.
3. **install virtctl**:
   - Linux의 경우:
     a. 아카이브 파일의 압축을 풉니다.

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

     b. 다음 명령을 실행하여 **virtctl** 바이너리를 실행할 수 있도록 합니다.

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

     c. **virtctl** 바이너리를 **PATH** 환경 변수의 디렉터리로 이동합니다.
     다음 명령을 실행하여 경로를 확인할 수 있습니다.

     ```
     $ echo $PATH
     ```

     d. **KUBECONFIG** 환경 변수를 설정합니다.

     ```
     $ export KUBECONFIG=/home/<user>/clusters/current/auth/kubeconfig
     ```

   - Windows의 경우:
     a. 아카이브 파일의 압축을 풉니다.
     b. 추출된 폴더 계층 구조로 이동하고 **virtctl** 실행 파일을 두 번 클릭하여 클라이언트를 설치합니다.
     c. **virtctl** 바이너리를 **PATH** 환경 변수의 디렉터리로 이동합니다.
     다음 명령을 실행하여 경로를 확인할 수 있습니다.

```
C:\> path
```

- macOS의 경우:
  a. 아카이브 파일의 압축을 풉니다.
  b. **virtctl** 바이너리를 **PATH** 환경 변수의 디렉터리로 이동합니다.
     다음 명령을 실행하여 경로를 확인할 수 있습니다.

```
echo $PATH
```

6.5.2. virtctl을 RPM으로 설치

OpenShift Virtualization 리포지토리를 활성화한 후 RHEL(Red Hat Enterprise Linux)에 **virtctl** 클라이언트를 RPM으로 설치할 수 있습니다.

6.5.2.1. OpenShift Virtualization 리포지토리 활성화

RHEL(Red Hat Enterprise Linux) 버전에 대해 OpenShift Virtualization 리포지토리를 활성화합니다.

사전 요구 사항

- "Red Hat Container Native Virtualization" 인타이틀먼트에 대한 활성 서브스크립션이 있는 Red Hat 계정에 시스템이 등록됩니다.

절차

- **subscription-manager** CLI 툴을 사용하여 운영 체제에 적합한 OpenShift Virtualization 리포지토리를 활성화합니다.
  - RHEL 8용 리포지토리를 활성화하려면 다음을 실행합니다.

```
# subscription-manager repos --enable cnv-4.12-for-rhel-8-x86_64-rpms
```

  - RHEL 7용 리포지토리를 활성화하려면 다음을 실행합니다.

```
# subscription-manager repos --enable rhel-7-server-cnv-4.12-rpms
```

6.5.2.2. yum 유틸리티를 사용하여 virtctl 클라이언트 설치

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

사전 요구 사항

- RHEL(Red Hat Enterprise Linux) 시스템에서 OpenShift Virtualization 리포지토리를 활성화했습니다.

절차

- **kubevirt-virtctl** 패키지를 설치합니다.

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

6.5.3. 추가 리소스

- OpenShift Virtualization 에 CLI 툴 사용.

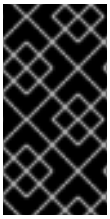
6.6. OPENSHIFT VIRTUALIZATION 설치 제거

웹 콘솔 또는 CLI(명령줄 인터페이스)를 사용하여 OpenShift Virtualization을 설치 제거하여 OpenShift Virtualization 워크로드, Operator 및 해당 리소스를 삭제합니다.

6.6.1. 웹 콘솔을 사용하여 OpenShift Virtualization 설치 제거

웹 콘솔 을 사용하여 다음 작업을 수행하여 OpenShift Virtualization을 설치 제거합니다.

1. **HyperConverged CR**을 삭제합니다.
2. OpenShift Virtualization Operator를 삭제합니다.
3. **openshift-cnv** 네임스페이스를 삭제합니다.
4. OpenShift Virtualization CRD(사용자 정의 리소스 정의)를 삭제합니다.



중요

먼저 모든 가상 머신 및 가상 머신 인스턴스를삭제해야 합니다.

워크로드가 클러스터에 남아 있는 동안 OpenShift Virtualization을 설치 제거할 수 없습니다.

6.6.1.1. HyperConverged 사용자 정의 리소스 삭제

OpenShift Virtualization을 설치 제거하려면 먼저 **HyperConverged** CR(사용자 정의 리소스)을 삭제합니다.

사전 요구 사항

- **cluster-admin** 권한이 있는 계정을 사용하여 OpenShift Container Platform 클러스터에 액세스할 수 있습니다.

절차

1. Operator → 설치된 Operator 페이지로 이동합니다.
2. OpenShift Virtualization Operator를 선택합니다.
3. OpenShift Virtualization 배포 탭을 클릭합니다.
4. **kubevirt-hyperconverged** 옆에 있는 옵션 메뉴 ⋮ 를 클릭하고 HyperConverged 삭제 를 선택합니다.
5. 확인 창에서 삭제를 클릭합니다.

6.6.1.2. 웹 콘솔을 사용하여 클러스터에서 Operator 삭제

클러스터 관리자는 웹 콘솔을 사용하여 선택한 네임스페이스에서 설치된 **Operator**를 삭제할 수 있습니다.

사전 요구 사항

- **cluster-admin** 권한이 있는 계정을 사용하여 **OpenShift Container Platform** 클러스터 웹 콘솔에 액세스할 수 있습니다.

절차

1. **Operator** → 설치된 **Operator** 페이지로 이동합니다.
2. 제거하려는 **Operator**를 찾으려면 이름으로 필터링 필드에 키워드를 스크롤하거나 입력합니다. 그런 다음 해당 **Operator**를 클릭합니다.
3. **Operator** 세부 정보 페이지 오른쪽에 있는 작업 목록에서 **Operator** 설치 제거를 선택합니다. **Operator**를 설치 제거하시겠습니까? 대화 상자가 표시됩니다.
4. 설치 제거를 선택하여 **Operator, Operator** 배포 및 **Pod**를 제거합니다. 이 작업 후에 **Operator**는 실행을 중지하고 더 이상 업데이트가 수신되지 않습니다.

참고

이 작업은 **CRD**(사용자 정의 리소스 정의) 및 **CR**(사용자 정의 리소스)을 포함하여 **Operator**에서 관리하는 리소스를 제거하지 않습니다. 웹 콘솔에서 활성화된 대시보드 및 탐색 항목과 계속 실행되는 클러스터 외부 리소스는 수동 정리가 필요할 수 있습니다. **Operator**를 설치 제거한 후 해당 항목을 제거하려면 **Operator CRD**를 수동으로 삭제해야 할 수 있습니다.

6.6.1.3. 웹 콘솔을 사용하여 네임스페이스 삭제

OpenShift Container Platform 웹 콘솔을 사용하여 네임스페이스를 삭제할 수 있습니다.

사전 요구 사항

- **cluster-admin** 권한이 있는 계정을 사용하여 **OpenShift Container Platform** 클러스터에 액세스할 수 있습니다.

절차

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

6.6.1.4. OpenShift Virtualization 사용자 정의 리소스 정의 삭제

웹 콘솔을 사용하여 **OpenShift Virtualization CRD**(사용자 정의 리소스 정의)를 삭제할 수 있습니다.

사전 요구 사항

- **cluster-admin** 권한이 있는 계정을 사용하여 OpenShift Container Platform 클러스터에 액세스할 수 있습니다.

절차

1. Administration → CustomResourceDefinitions 로 이동합니다.
2. Label 필터를 선택하고 검색 필드에 **operators.coreos.com/kubevirt-hyperconverged.openshift-cnv** 를 입력하여 OpenShift Virtualization CRD를 표시합니다.
3. 각 CRD 옆에 있는 옵션 메뉴 ⋮ 를 클릭하고 CustomResourceDefinition 삭제 를 선택합니다.

6.6.2. CLI를 사용하여 OpenShift Virtualization 설치 제거

OpenShift CLI(**oc**)를 사용하여 OpenShift Virtualization을 설치 제거할 수 있습니다.

사전 요구 사항

- **cluster-admin** 권한이 있는 계정을 사용하여 OpenShift Container Platform 클러스터에 액세스할 수 있습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- 모든 가상 머신 및 가상 머신 인스턴스를 삭제했습니다. 워크로드가 클러스터에 남아 있는 동안 OpenShift Virtualization을 설치 제거할 수 없습니다.

절차

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

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

2. OpenShift Virtualization Operator 서브스크립션을 삭제합니다.

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

3. OpenShift Virtualization **ClusterServiceVersion** 리소스를 삭제합니다.

```
$ oc delete csv -n openshift-cnv -l operators.coreos.com/kubevirt-
hyperconverged.openshift-cnv
```

4. OpenShift Virtualization 네임스페이스를 삭제합니다.

```
$ oc delete namespace openshift-cnv
```

5. 시험 실행 옵션을 사용하여 **oc delete crd** 명령을 실행하여 OpenShift Virtualization CRD(사용자 정의 리소스 정의) 를 나열합니다.

```
$ oc delete crd --dry-run=client -l operators.coreos.com/kubevirt-
hyperconverged.openshift-cnv
```

출력 예

```
customresourcedefinition.apiextensions.k8s.io "cdis.cdi.kubevirt.io" deleted (dry run)
customresourcedefinition.apiextensions.k8s.io
"hostpathprovisioners.hostpathprovisioner.kubevirt.io" deleted (dry run)
customresourcedefinition.apiextensions.k8s.io "hyperconvergeds.hco.kubevirt.io" deleted
(dry run)
customresourcedefinition.apiextensions.k8s.io "kubevirts.kubevirt.io" deleted (dry run)
customresourcedefinition.apiextensions.k8s.io
"networkaddonsconfigs.networkaddonsoperator.network.kubevirt.io" deleted (dry run)
customresourcedefinition.apiextensions.k8s.io "ssps.ssp.kubevirt.io" deleted (dry run)
customresourcedefinition.apiextensions.k8s.io "tektontasks.tektontasks.kubevirt.io" deleted
(dry run)
```

6. **dry-run** 옵션 없이 **oc delete crd** 명령을 실행하여 **CRD**를 삭제합니다.

```
$ oc delete crd -l operators.coreos.com/kubevirt-hyperconverged.openshift-cnv
```

추가 리소스

- 가상 머신 삭제
- 가상 머신 인스턴스 삭제

7장. OPENSHIFT VIRTUALIZATION 업데이트

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

참고

- Node Maintenance Operator (NMO)는 더 이상 OpenShift Virtualization과 함께 제공되지 않습니다. OpenShift Container Platform 웹 콘솔의 OperatorHub 에서 NMO를 설치하거나 OpenShift CLI(oc)를 사용하여 설치할 수 있습니다. 노드 수정, 펜싱 및 유지 관리에 대한 자세한 내용은 Workload Availability for Red Hat OpenShift 설명서를 참조하십시오.
  OpenShift Virtualization 4.10.2 이상 릴리스에서 OpenShift Virtualization 4.11로 업데이트하기 전에 다음 작업 중 하나를 수행해야 합니다.
  - 모든 노드를 유지보수 모드에서 이동합니다.
  - 독립 실행형 NMO를 설치하고 **nodemaintenances.nodemaintenance.kubevirt.io** CR(사용자 정의 리소스)을 **nodemaintenances.nodemaintenance.medik8s.io** CR로 교체합니다.

7.1. 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 업데이트할 때 마이너 버전 업데이트가 제공됩니다. 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 설정되어 있으면 **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 4.12에서 OpenShift Virtualization 4.12를 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 Virtualization을 업데이트하면 네트워크 연결이 중단되지 않습니다.
- 데이터 볼륨 및 관련 영구 볼륨 클레임은 업데이트 중에 보존됩니다.

중요

hostpath 프로비전 프로그램을 사용하는 가상 머신이 실행 중인 경우 실시간으로 마이그레이션할 수 없으며 **OpenShift Container Platform** 클러스터 업데이트가 차단될 수 있습니다.

해결 방법으로 클러스터를 업데이트하는 동안 전원이 자동으로 꺼지도록 가상 머신을 재구성할 수 있습니다. **evictionStrategy: LiveMigrate** 필드를 제거하고 **runStrategy** 필드를 **Always**로 설정합니다.

7.1.1. 워크로드 업데이트 정보

OpenShift Virtualization을 업데이트하면 **libvirt,virt-launcher, qemu** 를 포함한 가상 머신 워크로드가 실시간 마이그레이션을 지원하는 경우 자동으로 업데이트됩니다.

참고

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

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

Evict 방법이 **VMI Pod**를 종료하므로**LiveMigrate** 업데이트 전략만 기본적으로 활성화됩니다.

LiveMigrate 가 활성화된 유일한 업데이트 전략인 경우:

- 실시간 마이그레이션을 지원하는 **VMI**가 업데이트 프로세스 중에 마이그레이션됩니다. **VM** 게스트는 업데이트된 구성 요소가 활성화된 새 **Pod**로 이동합니다.
- 실시간 마이그레이션을 지원하지 않는 **VMI**는 중단되거나 업데이트되지 않습니다.
  - **VMI**에 **LiveMigrate** 제거 전략이 있지만 실시간 마이그레이션을 지원하지 않는 경우 업데이트되지 않습니다.

LiveMigrate 및 **Evict** 를 모두 사용할 수 있는 경우:

- 실시간 마이그레이션을 지원하는 **VMI**는 **LiveMigrate** 업데이트 전략을 사용합니다.
- 실시간 마이그레이션을 지원하지 않는 **VMI**는 **Evict** 업데이트 전략을 사용합니다. **VMI**가항상 **runStrategy** 값이 있는**VirtualMachine** 오브젝트에 의해 제어되는 경우 업데이트된 구성 요소가 있는 새 **Pod**에 새 **VMI**가 생성됩니다.

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

5분

Pod가 보류 중이므로 **Unschedul**을 사용할 수 있습니다.

15분

어떤 이유로든 **Pod**가 보류 중 상태에 있는 경우

VMI가 마이그레이션되지 않으면 **virt-controller** 에서 다시 마이그레이션하려고 합니다. 이 프로세스는 모든 migratable VMI가 새 **virt-launcher** Pod에서 실행될 때까지 이 프로세스를 반복합니다. 그러나 VMI가 부적절하게 구성된 경우 이러한 시도는 무기한 반복할 수 있습니다.

참고

각 시도는 마이그레이션 오브젝트에 해당합니다. 최근 5개의 시도만 버퍼에 저장됩니다. 이렇게 하면 디버깅을 위한 정보를 유지하면서 마이그레이션 오브젝트가 시스템에서 누적되는 것을 방지할 수 있습니다.

7.1.2. EUS-to-EUS 업데이트 정보

4.10 및 4.12를 포함한 OpenShift Container Platform의 모든 짝수 마이너 버전은 EUS (Extended Update Support) 버전입니다. 그러나 Kubernetes 설계에는 직렬 마이너 버전 업데이트가 필요하므로 하나의 EUS 버전에서 다음 버전으로 직접 업데이트할 수 없습니다.

소스 EUS 버전에서 다음 홀수의 마이너 버전으로 업데이트한 후 업데이트 경로에 있는 해당 마이너 버전의 모든 z-stream 릴리스로 OpenShift Virtualization을 순차적으로 업데이트해야 합니다. 해당 최신 z-stream 버전으로 업그레이드한 후 OpenShift Container Platform을 대상 EUS 마이너 버전으로 업데이트할 수 있습니다.

OpenShift Container Platform 업데이트가 성공하면 OpenShift Virtualization에 대한 해당 업데이트를 사용할 수 있습니다. 이제 OpenShift Virtualization을 대상 EUS 버전으로 업데이트할 수 있습니다.

7.1.2.1. 업데이트 준비

EUS-to-EUS 업데이트를 시작하기 전에 다음을 수행해야 합니다.

- EUS-to-EUS 업데이트를 시작하기 전에 작업자 노드의 머신 구성 풀을 일시 중지하여 작업자가 두 번 재부팅되지 않도록 합니다.
- 업데이트 프로세스를 시작하기 전에 자동 워크로드 업데이트를 비활성화합니다. 이는 대상 EUS 버전으로 업데이트할 때까지 OpenShift Virtualization이 VM(가상 머신)을 마이그레이션하거나 제거하지 않도록하기 위한 것입니다.

참고

기본적으로 OpenShift Virtualization은 OpenShift Virtualization Operator를 업데이트할 때 **virt-launcher** Pod와 같은 워크로드를 자동으로 업데이트합니다.**HyperConverged** 사용자 정의 리소스의 **spec.workloadUpdateStrategy** 스탠자에서 이 동작을 구성할 수 있습니다.

EUS에서 EUS로의 업데이트 수행 준비에 대해 자세히 알아보십시오.

7.2. EUS-TO-EUS 업데이트 중 워크로드 업데이트 방지

EUS (Extended Update Support) 버전에서 다음 버전으로 업데이트하는 경우 OpenShift Virtualization이 업데이트 프로세스 중 워크로드를 마이그레이션하거나 제거하지 않도록 자동 워크로드 업데이트를 수동으로 비활성화해야 합니다.

사전 요구 사항

- OpenShift Container Platform의 EUS 버전을 실행 중이며 다음 EUS 버전으로 업데이트하려고 합니다. 그 사이에 홀수의 버전으로 아직 업데이트하지 않았습니다.

- "EUS-to-EUS 업데이트를 수행하기 위한 준비"를 읽고 OpenShift Container Platform 클러스터와 관련된 경고 및 요구 사항을 알아봅니다.
- OpenShift Container Platform 설명서의 지시에 따라 작업자 노드의 머신 구성 풀을 일시 중지했습니다.
- 기본 자동 승인 전략을 사용하는 것이 좋습니다.수동 승인 전략을 사용하는 경우 웹 콘솔에서 보류 중인 모든 업데이트를 승인해야 합니다. 자세한 내용은 "Manually a pending Operator update" 섹션을 참조하십시오.

절차

1. 다음 명령을 실행하여 현재 **workloadUpdateMethods** 구성을 백업합니다.

```
$ WORKLOAD_UPDATE_METHODS=$(oc get kv kubevirt-kubevirt-hyperconverged -n
openshift-cnv -o jsonpath='{.spec.workloadUpdateStrategy.workloadUpdateMethods}')
```

2. 다음 명령을 실행하여 모든 워크로드 업데이트 방법을 끕니다.

```
$ oc patch hco kubevirt-hyperconverged -n openshift-cnv --type json -p
'[{"op":"replace","path":"/spec/workloadUpdateStrategy/workloadUpdateMethods",
"value":[]}]'
```

출력 예

```
hyperconverged.hco.kubevirt.io/kubevirt-hyperconverged patched
```

3. 계속하기 전에 **HyperConverged** Operator를 업그레이드할 수 있는지 확인합니다. 다음 명령을 입력하고 출력을 모니터링합니다.

```
$ oc get hco kubevirt-hyperconverged -n openshift-cnv -o json | jq
".status.conditions"
```

예 7.1. 출력 예

```
[
  {
    "lastTransitionTime": "2022-12-09T16:29:11Z",
    "message": "Reconcile completed successfully",
    "observedGeneration": 3,
    "reason": "ReconcileCompleted",
    "status": "True",
    "type": "ReconcileComplete"
  },
  {
    "lastTransitionTime": "2022-12-09T20:30:10Z",
    "message": "Reconcile completed successfully",
    "observedGeneration": 3,
    "reason": "ReconcileCompleted",
    "status": "True",
    "type": "Available"
  },
  {
    "lastTransitionTime": "2022-12-09T20:30:10Z",
```

```
    "message": "Reconcile completed successfully",
    "observedGeneration": 3,
    "reason": "ReconcileCompleted",
    "status": "False",
    "type": "Progressing"
  },
  {
    "lastTransitionTime": "2022-12-09T16:39:11Z",
    "message": "Reconcile completed successfully",
    "observedGeneration": 3,
    "reason": "ReconcileCompleted",
    "status": "False",
    "type": "Degraded"
  },
  {
    "lastTransitionTime": "2022-12-09T20:30:10Z",
    "message": "Reconcile completed successfully",
    "observedGeneration": 3,
    "reason": "ReconcileCompleted",
    "status": "True",
    "type": "Upgradeable" (1)
  }
]
```

1. OpenShift Virtualization Operator의 업그레이드 가능 상태입니다.

4. 소스 EUS 버전에서 다음 마이너 버전의 OpenShift Container Platform으로 클러스터를 수동으로 업데이트합니다.

```
$ oc adm upgrade
```

검증

- 다음 명령을 실행하여 현재 버전을 확인합니다.

```
$ oc get clusterversion
```

참고

OpenShift Container Platform을 다음 버전으로 업데이트하는 것은 OpenShift Virtualization을 업데이트하기 위한 사전 요구 사항입니다. 자세한 내용은 OpenShift Container Platform 설명서의 "클러스터 업데이트" 섹션을 참조하십시오.

5. OpenShift Virtualization을 업데이트합니다.

- 기본 자동 승인 전략을 사용하면 OpenShift Container Platform을 업데이트한 후 OpenShift Virtualization이 해당 버전으로 자동으로 업데이트됩니다.
- 수동 승인 전략을 사용하는 경우 웹 콘솔을 사용하여 보류 중인 업데이트를 승인합니다.

6. 다음 명령을 실행하여 OpenShift Virtualization 업데이트를 모니터링합니다.

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

7. OpenShift Virtualization을 EUS 이외의 마이너 버전에서 사용할 수 있는 모든 z-stream 버전으로 업데이트하여 이전 단계에 표시된 명령을 실행하여 각 업데이트를 모니터링합니다.

8. 다음 명령을 실행하여 OpenShift Virtualization이 EUS 이외의 버전의 최신 z-stream 릴리스로 성공적으로 업데이트되었는지 확인합니다.

```
$ oc get hco kubevirt-hyperconverged -n openshift-cnv -o json | jq ".status.versions"
```

출력 예

```
[
  {
    "name": "operator",
    "version": "4.12.13"
  }
]
```

9. 다음 업데이트를 수행하기 전에 HyperConverged Operator에 Upgradeable 상태가 될 때까지 기다립니다. 다음 명령을 입력하고 출력을 모니터링합니다.

```
$ oc get hco kubevirt-hyperconverged -n openshift-cnv -o json | jq
".status.conditions"
```

10. OpenShift Container Platform을 대상 EUS 버전으로 업데이트합니다.

11. 클러스터 버전을 확인하여 업데이트가 성공했는지 확인합니다.

```
$ oc get clusterversion
```

12. OpenShift Virtualization을 대상 EUS 버전으로 업데이트합니다.

    - 기본 자동 승인 전략을 사용하면 OpenShift Container Platform을 업데이트한 후 OpenShift Virtualization이 해당 버전으로 자동으로 업데이트됩니다.

    - 수동 승인 전략을 사용하는 경우 웹 콘솔을 사용하여 보류 중인 업데이트를 승인합니다.

13. 다음 명령을 실행하여 OpenShift Virtualization 업데이트를 모니터링합니다.

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

VERSION 필드가 대상 EUS 버전과 일치하고PHASE 필드에 Succeeded 가 표시되면 업데이트가 완료됩니다.

14. 백업한 워크로드 업데이트 방법 구성을 복원합니다.

```
$ oc patch hco kubevirt-hyperconverged -n openshift-cnv --type json -p "
[{\"op\":\"add\",\"path\":\"/spec/workloadUpdateStrategy/workloadUpdateMethods\",
\"value\":$WORKLOAD_UPDATE_METHODS}]"
```

출력 예

```
hyperconverged.hco.kubevirt.io/kubevirt-hyperconverged patched
```

검증

- 다음 명령을 실행하여 VM 마이그레이션의 상태를 확인합니다.

```
$ oc get vmim -A
```

다음 단계

- 이제 작업자 노드의 머신 구성 풀의 일시 정지를 해제할 수 있습니다.

7.3. 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 않는 경우 **VMI**가 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 **VMI**에 실시간 마이그레이션 및 **Evict**를 지원하는 **VMI**

에 **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 **Pod**로 마이그레이션하여 실시간 마이그레이션을 지원하는 **VMI**가 업데이트됩니다. **LiveMigrate**가 나열된 유일한 워크로드 업데이트 방법인 경우 실시간 마이그레이션을 지원하지 않는 **VMI**는 중단되거나 업데이트되지 않습니다.

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

4. **Evict** 방법을 사용하여 한 번에 업데이트하도록 강제 수행할 수 있는 **VMI** 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.

7.4. 보류 중인 OPERATOR 업데이트 승인

7.4.1. 보류 중인 Operator 업데이트 수동 승인

설치된 **Operator**의 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 **Operator**입니다.

절차

1. **OpenShift Container Platform** 웹 콘솔의 관리자 관점에서 **Operator → 설치된 Operator**로 이동합니다.

2. 보류 중인 업데이트가 있는 **Operator**에 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.

7.5. 업데이트 상태 모니터링

7.5.1. 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
```

7.5.2. 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.

7.6. 추가 리소스

- EUS에서 EUS로의 업데이트 수행 준비
- 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

8장. 보안 정책

VM(가상 머신) 워크로드는 권한이 없는 Pod로 실행됩니다. VM에서 OpenShift Virtualization 기능을 사용할 수 있도록 일부 포드에는 다른 Pod 소유자가 사용할 수 없는 사용자 정의 보안 정책이 부여됩니다.

- 확장된 **container_t** SELinux 정책은 **virt-launcher** Pod에 적용됩니다.
- SCC(보안 컨텍스트 제약 조건)는 **kubevirt-controller** 서비스 계정에 대해 정의됩니다.

8.1. 워크로드 보안 정보

기본적으로 VM(가상 머신) 워크로드는 OpenShift Virtualization에서 루트 권한으로 실행되지 않습니다.

virt-launcher Pod는 각 VM에 대해 ***세션 모드에서*** **libvirt** 인스턴스를 실행하여 VM 프로세스를 관리합니다. 세션 모드에서 **libvirt** 데몬은 루트가 아닌 사용자 계정으로 실행되며 동일한 사용자 식별자(UID)에서 실행 중인 클라이언트의 연결만 허용합니다. 따라서 VM은 권한이 없는 Pod로 실행되며 최소 권한의 보안 원칙을 준수합니다.

루트 권한이 필요한 OpenShift Virtualization 기능은 지원되지 않습니다. 기능에 **root**가 필요한 경우 OpenShift Virtualization에서 사용할 수 없습니다.

8.2. 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 **NFS**에 액세스할 수 있습니다.
  - **allow process nfs_t(dir(mounton))**

  - **allow process proc_t (dir) (mounton)**
  - **allow process proc_t (filesystem (mount unmount))**

- 다음 정책은 패스트 네트워킹을 활성화하는 업스트림 **Kubevirt**에서 상속됩니다.

  - **allow process tmpfs_t (filesystem (mount))**

참고

OpenShift Virtualization은 현재 패스를 지원하지 않습니다.

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

SCC(보안 컨텍스트 제약 조건)는 **Pod**에 대한 권한을 제어합니다. 이러한 권한에는 컨테이너 모음인 **Pod**에서 수행할 수 있는 작업과 액세스할 수 있는 리소스가 포함됩니다. **Pod**가 시스템에 수용되려면 일련의 조건을 함께 실행해야 하는데, **SCC**를 사용하여 이러한 조건을 정의할 수 있습니다.

virt-controller 는 클러스터의 가상 머신에 대해 **virt-launcher Pod**를 생성하는 클러스터 컨트롤러입니다. 이러한 **Pod**에는 **kubevirt-controller** 서비스 계정에서 권한이 부여됩니다.

kubevirt-controller 서비스 계정에는 적절한 권한으로 **virt-launcher Pod**를 생성할 수 있도록 추가 **SCC** 및 **Linux** 기능이 부여됩니다. 이러한 확장된 권한을 통해 가상 머신에서 일반적인 **Pod**의 범위를 벗어나는 **OpenShift Virtualization** 기능을 사용할 수 있습니다.

kubevirt-controller 서비스 계정에는 다음 **SCC**가 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{"SYS_NICE", "NET_BIND_SERVICE", "SYS_PTRACE"}**

- SYS_NICE 를 사용하면 CPU 선호도를 설정할 수 있습니다.
- NET_BIND_SERVICE 를 사용하면 DHCP 및 Slirp 작업을 수행할 수 있습니다.
- SYS_PTRACE 를 사용하면 특정 버전의 libvirt 가 소프트웨어 TPM(Trusted Platform Module) 에뮬레이터인 swtpm 의 PID(프로세스 ID)를 찾을 수 있습니다.

8.3.1. kubevirt-controller에 대한 SCC 및 RBAC 정의 보기

oc 툴을 사용하여 kubevirt-controller에 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
```

8.4. 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

9장. CLI 툴 사용

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

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

9.1. 사전 요구 사항

- **virtctl** 클라이언트를 설치해야 합니다.

9.2. 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 **9.1. 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.

명령	설명
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 내용은 **OpenShift Container Platform CLI** 툴 설명서를 참조하십시오.

9.3. VIRTCTL 명령

virtctl 클라이언트는 **OpenShift Virtualization** 리소스를 관리하는 명령줄 유틸리티입니다.

표 **9.2. virtctl** 일반 명령

명령	설명
virtctl version	**virtctl** 클라이언트 및 서버 버전을 확인합니다.
virtctl help	**virtctl** 명령 목록을 확인합니다.
virtctl <command> -h|--help	특정 명령의 옵션 목록을 확인합니다.
virtctl options	**virtctl** 명령의 글로벌 명령 옵션 목록을 확인합니다.

9.3.1. VM 및 VMI 관리 명령

virtctl 을 사용하여 **VM**(가상 머신) 또는 **VMI**(가상 머신 인스턴스) 상태를 관리하고 **VM**을 마이그레이션할 수 있습니다.

표 **9.3. virtctl VM** 관리 명령

명령	설명
virtctl start <vm_name>	VM을 시작합니다.
virtctl start --paused <vm_name>	일시 중지된 상태에서 VM을 시작합니다. 이 옵션을 사용하면 VNC 콘솔에서 부팅 프로세스를 중단할 수 있습니다.
virtctl stop <vm_name>	VM을 중지합니다.

명령	설명
virtctl stop <vm_name> --grace-period 0 --force	VM을 강제 중지합니다. 이 옵션을 사용하면 데이터 불일치 또는 데이터 손실이 발생할 수 있습니다.
virtctl pause vm|vmi <vm_name>	VM 또는 VMI를 일시 중지합니다. 머신 상태는 메모리에 유지됩니다.
virtctl unpause vm|vmi <vm_name>	VM 또는 VMI의 일시 정지를 해제합니다.
virtctl migrate <vm_name>	VM을 마이그레이션합니다.
virtctl restart <vm_name>	VM을 다시 시작합니다.

9.3.2. VM 및 VMI 연결 명령

virtctl 을 사용하여 직렬 콘솔에 연결하고, 포트를 노출하고, 프록시 연결을 설정하고, 포트를 지정하고, **VM**에 대한 **VNC** 연결을 열 수 있습니다.

표 **9.4. virtctl console,expose, vnc** 명령

명령	설명
virtctl console <vmi_name>	VMI의 직렬 콘솔에 연결합니다.
virtctl expose <vm_name>	VM 또는 VMI의 지정된 포트를 전달하고 서비스를 노드의 지정된 포트에 노출하는 서비스를 생성합니다.
virtctl vnc --kubeconfig=$KUBECONFIG <vmi_name>	VMI에 대한 VNC(Virtual Network Client) 연결을 엽니다. VNC를 통해 VMI의 그래픽 콘솔에 액세스하려면 로컬 머신에 원격 뷰어가 필요합니다.
virtctl vnc --kubeconfig=$KUBECONFIG --proxy-only=true <vmi_name>	포트 번호를 표시하고 VNC 연결을 통해 뷰어를 사용하여 VMI에 수동으로 연결합니다.
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.

9.3.3. VM 볼륨 내보내기 명령

virtctl vmexport 명령을 사용하여 **VM, VM** 스냅샷 또는 **PVC(**영구 볼륨 클레임**)**에서 내보낸 볼륨을 생성**,** 다운로드 또는 삭제할 수 있습니다**.**

표 **9.5. virtctl vmexport** 명령

명령	설명
virtctl vmexport create <vmexport_name> --vm|snapshot|pvc= <object_name>	**VirtualMachineExport** CR(사용자 정의 리소스)을 생성하여 VM, VM 스냅샷 또는 PVC에서 볼륨을 내보냅니다. ● **--VM**: VM의 PVC를 내보냅니다. ● **--snapshot**: **VirtualMachineSnapshot** CR에 포함된 PVC를 내보냅니다. ● **--PVC**: PVC를 내보냅니다. ● 선택 사항: **--ttl=1h** 는 라이브 시간을 지정합니다. 기본 기간은 2 시간입니다.
virtctl vmexport delete <vmexport_name>	**VirtualMachineExport** CR을 수동으로 삭제합니다.
virtctl vmexport 다운로드 **<vmexport_name> --output= <output_file> --volume= <volume_name>**	**VirtualMachineExport** CR에 정의된 볼륨을 다운로드합니다. ● **--output** 은 파일 형식을 지정합니다. 예:**disk.img.gz**. ● **--volume** 은 다운로드할 볼륨을 지정합니다. 하나의 볼륨만 사용할 수 있는 경우 이 플래그는 선택 사항입니다. 선택 사항: ● **--keep-vme** 는 다운로드 후 **VirtualMachineExport** CR을 유지합니다. 기본 동작은 다운로드 후 **VirtualMachineExport** CR을 삭제하는 것입니다. ● **--insecure** 는 비보안 HTTP 연결을 활성화합니다.
virtctl vmexport 다운로드 **<vmexport_name> -- <vm|snapshot|pvc>= <object_name> --output= <output_file> --volume= <volume_name>**	**VirtualMachineExport** CR을 생성한 다음 CR에 정의된 볼륨을 다운로드합니다.

9.3.4. VM 메모리 덤프 명령

virtctl memory-dump 명령을 사용하여 **PVC**에서 **VM(**가상 머신**)** 메모리 덤프를 출력할 수 있습니다**.** 기존 **PVC**를 지정하거나 **--create-claim** 플래그를 사용하여 새 **PVC**를 생성할 수 있습니다**.**

사전 요구 사항

- **PVC** 볼륨 모드는 **FileSystem** 여야 합니다.
- **PVC**는 메모리 덤프를 포함할 수 있을 만큼 커야 합니다.

  PVC 크기를 계산하는 공식은 **(VMMemorySize + 100Mi) * FileSystemOverhead** 입니다. 여기서 **100Mi** 는 메모리 덤프 오버헤드입니다.
- 다음 명령을 실행하여 **HyperConverged** 사용자 정의 리소스에서 핫플러그 기능 게이트를 활성화해야 합니다.

```
$ oc patch hco kubevirt-hyperconverged -n openshift-cnv \
  --type json -p '[{"op": "add", "path": "/spec/featureGates", \
  "value": "HotplugVolumes"}]'
```

메모리 덤프 다운로드

메모리 덤프를 다운로드하려면 **virtctl vmexport download** 명령을 사용해야 합니다.

```
$ virtctl vmexport download <vmexport_name> --vm\|pvc=<object_name> \
  --volume=<volume_name> --output=<output_file>
```

표 **9.6. virtctl memory-dump** 명령

명령	설명
virtctl memory-dump get <vm_name> --claim-name= <pvc_name>	PVC에 VM의 메모리 덤프를 저장합니다. 메모리 덤프 상태는 **VirtualMachine** 리소스의 **status** 섹션에 표시됩니다. 선택 사항: ● **--create-claim** 은 적절한 크기로 새 PVC를 생성합니다. 이 플래그에는 다음 옵션이 있습니다. ○ **--storage-class=<storage_class&** gt; : PVC의 스토리지 클래스를 지정합니다. ○ **--access-mode=<access_mode&gt**; : **ReadWriteOnce** 또는 **ReadWriteMany** 를 지정합니다.
virtctl memory-dump get <vm_name>	동일한 PVC를 사용하여 **virtctl memory-dump** 명령을 재실행합니다. 이 명령은 이전 메모리 덤프를 덮어씁니다.

명령	설명
virtctl memory-dump remove <vm_name>	메모리 덤프를 제거합니다. 대상 PVC를 변경하려면 메모리 덤프를 수동으로 제거해야 합니다. 이 명령은 VM과 PVC 간의 연결을 제거하여 메모리 덤프가 **VirtualMachine** 리소스의 **status** 섹션에 표시되지 않도록 합니다. PVC는 영향을 받지 않습니다.

9.3.5. 이미지 업로드 명령

virtctl image-upload 명령을 사용하여 **VM** 이미지를 데이터 볼륨에 업로드할 수 있습니다.

표 **9.7. virtctl image-upload** 명령

명령	설명
virtctl image-upload dv <datavolume_name> --image-path= </path/to/image> --no-create	VM 이미지를 이미 존재하는 데이터 볼륨에 업로드합니다.
virtctl image-upload dv <datavolume_name> --size= <datavolume_size> --image-path=</path/to/image>	VM 이미지를 지정된 요청된 크기의 새 데이터 볼륨에 업로드합니다.

9.3.6. 환경 정보 명령

virtctl 을 사용하여 버전, 파일 시스템, 게스트 운영 체제 및 로그인한 사용자에 대한 정보를 볼 수 있습니다.

표 **9.8. virtctl** 환경 정보 명령

명령	설명
virtctl fslist <vmi_name>	게스트 시스템에서 사용 가능한 파일 시스템을 확인합니다.
virtctl guestosinfo <vmi_name>	게스트 머신의 운영 체제에 대한 정보를 봅니다.
virtctl userlist <vmi_name>	게스트 머신에서 로그인한 사용자를 확인합니다.

9.4. 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 **PVC**를 마운트하고 쉘을 연결하려면 다음 명령을 실행합니다.

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

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

9.5. 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 **PVC**에서 **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.

명령	설명

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

또한 명령은 다른 **pod**에서 **PVC**를 사용 중인지 확인합니다. 이 경우 오류 메시지가 표시됩니다. 그러나 **libguestfs-tools** 프로세스가 시작되면 동일한 **PVC**를 사용하는 새 **Pod**를 방지할 수 없습니다. 동일한 **PVC**에 액세스하는 **VM**을 시작하기 전에 활성 **virtctl guestfs Pod**가 없는지 확인해야 합니다.

참고

virtctl guestfs 명령은 대화형 **Pod**에 연결된 단일 **PVC**만 허용합니다.

9.6. 추가 리소스

- **libguestfs:** 가상 머신 디스크 이미지에 액세스하고 수정하기 위한 툴입니다.

10장. 가상 머신

10.1. 가상 머신 생성

가상 머신을 생성하려면 다음 절차 중 하나를 사용하십시오.

- 빠른 시작 기능 둘러보기
- 카탈로그에서빠른 생성
- 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 추가하는 단계를 완료하면 이러한 템플릿을 사용할 수 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.

10.1.1. 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.

10.1.2. 가상 머신 생성 빠른 생성

사용 가능한 부팅 소스와 함께 템플릿을 사용하여 **VM(**가상 머신**)**을 빠르게 생성할 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → 카탈로그 를 클릭합니다.
2. 부팅 소스가 있는 템플릿을 필터링하려면 사용 가능한 부팅 소스를 클릭합니다.

참고

기본적으로 템플릿 목록은 기본 템플릿 만 표시합니다. 필터링 시 모든 항목을 클릭하여 선택한 필터에 사용 가능한 모든 템플릿을 확인합니다.

3. 템플릿을 클릭하여 세부 정보를 확인합니다.

4. 빠른 생성 **VirtualMachine** 를 클릭하여 템플릿에서 **VM**을 생성합니다.

   가상 머신 세부 정보 페이지가 배포 상태와 함께 표시됩니다.

검증

1. 이벤트를 클릭하여 **VM**이 프로비저닝될 때 이벤트 스트림을 확인합니다.

2. 콘솔을 클릭하여 **VM**이 성공적으로 부팅되었는지 확인합니다.

10.1.3. 사용자 지정된 템플릿에서 가상 머신 생성

일부 템플릿에는 추가 매개변수(예: 부팅 소스가 있는 **PVC**)가 필요합니다. 템플릿에서 선택한 매개변수를 사용자 지정하여 **VM**(가상 머신)을 생성할 수 있습니다.

절차

1. 웹 콘솔에서 템플릿을 선택합니다.

   a. 사이드 메뉴에서 가상화 → 카탈로그 를 클릭합니다.

   b. 선택 사항: 프로젝트, 키워드, 운영 체제 또는 워크로드 프로필로 템플릿을 필터링합니다.

   c. 사용자 지정할 템플릿을 클릭합니다.

2.

2. **Customize VirtualMachine** 를 클릭합니다.

3. **Name** 및 **Disk** 소스를 포함하여 **VM**의 매개변수를 지정합니다. 선택적으로 복제할 데이터 소스를 지정할 수 있습니다.

검증

1. 이벤트를 클릭하여 **VM**이 프로비저닝될 때 이벤트 스트림을 확인합니다.

2. 콘솔을 클릭하여 **VM**이 성공적으로 부팅되었는지 확인합니다.

웹 콘솔에서 **VM**을 생성할 때 가상 머신 필드 섹션을 참조하십시오.

10.1.3.1. 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: 브리지 ● SR-IOV 네트워크: **SR-IOV**
MAC 주소	네트워크 인터페이스 컨트롤러의 MAC 주소입니다. MAC 주소를 지정하지 않으면 주소가 자동으로 할당됩니다.

10.1.3.2. 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.
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,URL**을 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** 이상에서 시스템은 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.

10.1.3.3. Cloud-init 필드

이름	설명
승인된 SSH 키	가상 머신의 *~/.ssh/authorized_keys*에 복사되는 사용자의 공개 키입니다.

이름	설명
사용자 정의 스크립트	기타 옵션을 사용자 정의 cloud-init 스크립트를 붙여넣는 필드로 교체합니다.

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

10.1.3.4. 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 → **VirtualMachine**를 클릭합니다.
2. 생성을 클릭하고 **YAML** 사용 을 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 머신이 **VirtualMachines** 페이지에 나열됩니다.

10.1.4. CLI를 사용하여 가상 머신 생성

virtualMachine 매니페스트에서 가상 머신을 생성할 수 있습니다.

절차

1. **VM**의 **VirtualMachine** 매니페스트를 편집합니다. 예를 들어 다음 매니페스트에서는 **RHEL(Red Hat Enterprise Linux) VM**을 구성합니다.

예 **10.1. RHEL VM**의 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-user**의 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>
```

10.1.5. 가상 머신 스토리지 볼륨 유형

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.

10.1.6. 가상 머신 RunStrategies 정보

가상 머신에 대한 **RunStrategy**는 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 **VMI**를 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 **VMI**가 항상 존재합니다. 어떠한 이유로 원본이 중지되면 새 **VMI**가 생성되는데, 이러한 동작은 **spec.running: true**와 동일합니다.

RerunOnFailure

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

Manual

start, stop, restart virtctl 클라이언트 명령을 사용하여 **VMI**의 상태 및 존재를 제어할 수 있습니다.

Halted

가상 머신이 생성될 때 **VMI**가 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 **VM**의 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 **MachineHealthCheck**에 실패하여 클러스터에서 노드를 사용할 수 없는 경우, 새 노드에 **RunStrategy**가 **Always** 또는 **RerunOnFailure**인 **VM**이 다시 예약됩니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
```

```
RunStrategy: Always (1)
template:
...
```

(1) VMI의 현재 **RunStrategy** 설정입니다.

10.1.7. 추가 리소스

- **KubeVirt v0.58.0 API** 참조 의 **VirtualMachineSpec** 정의에는 가상 머신 사양의 매개변수 및 계층 구조에 대한 광범위한 컨텍스트가 있습니다.

  참고

  KubeVirt API 참조는 업스트림 프로젝트 참조이며 **OpenShift Virtualization** 에서 지원되지 않는 매개변수를 포함할 수 있습니다.

- **CPU** 관리자를 활성화하여 고성능 워크로드 프로필을 사용합니다.

- 가상 머신에 **containerDisk** 볼륨으로 추가할 컨테이너 디스크 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

10.2. 가상 머신 편집

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

10.2.1. 웹 콘솔에서 가상 머신 편집

OpenShift Container Platform 웹 콘솔 또는 명령줄 인터페이스를 사용하여 가상 머신을 편집할 수 있습니다.

절차

1. 웹 콘솔에서 가상화 → **VirtualMachines** 로 이동합니다.

2. 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.

3. 필드를 편집할 수 있음을 나타내는 연필 아이콘이 있는 필드를 클릭합니다. 예를 들어 **BIOS** 또는 **UEFI**와 같은 현재 부팅 모드 설정을 클릭하여 부팅 모드 창을 열고 목록에서 옵션을 선택합니다.

4. 저장을 클릭합니다.

참고

가상 머신이 실행 중인 경우 가상 머신을 재시작해야 **Boot Order** 또는 **Flavor**에 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.

10.2.2. 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 → **VirtualMachine**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.

10.2.3. 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 CLI**를 설치했습니다.

절차

1. 다음 명령을 실행하여 가상 머신 구성을 업데이트합니다.

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

2. 오브젝트 구성을 엽니다.

3. **YAML**을 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>
```

10.2.4. 가상 머신에 가상 디스크 추가

가상 디스크를 가상 머신에 추가하려면 다음 절차를 사용하십시오.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachine**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 및 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** 설정을 지워 지워 가상 디스크의 볼륨 모드 및 액세스 모드 를 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 프로필 사용자 지정을 참조하십시오.

10.2.4.1. VirtualMachines의 CD-ROM 편집

가상 머신을 위한 **CD-ROM**을 편집하려면 다음 절차를 사용하십시오.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachine**를 클릭합니다.

2. 가상 머신을 선택하여 **VirtualMachine** 세부 정보 화면을 엽니다.

3. 디스크 탭을 클릭합니다.

4. 편집하려는 **CD-ROM**의 옵션 메뉴 ⋮ 를 클릭하고 편집을 선택합니다.

5. **CD-ROM** 편집 창에서 소스, 영구 볼륨 클레임, 이름, 유형 및 인터페이스 필드를 편집합니다.

6.

6. 저장을 클릭합니다.

10.2.4.2. 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.
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**,**URL**을 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** 이상에서 시스템은 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.

10.2.5. 가상 머신에 네트워크 인터페이스 추가

가상 머신에 네트워크 인터페이스를 추가하려면 다음 절차를 사용하십시오.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachine**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.

10.2.5.1. 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.

이름	설명
유형	사용 가능한 바인딩 방법 목록입니다. 네트워크 인터페이스에 적합한 바인딩 방법을 선택합니다. ● 기본 Pod 네트워크: **masquerade** ● Linux 브리지 네트워크: 브리지 ● SR-IOV 네트워크: **SR-IOV**
MAC 주소	네트워크 인터페이스 컨트롤러의 MAC 주소입니다. MAC 주소를 지정하지 않으면 주소가 자동으로 할당됩니다.

10.2.6. 추가 리소스

- 스토리지 프로파일 사용자 정의

10.3. 부팅 순서 편집

웹 콘솔 또는 **CLI**를 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 **NIC**를 제거하고 부팅 가능 소스 인벤토리로 반환합니다.

10.3.1. 웹 콘솔에서 부팅 순서 목록에 항목 추가

웹 콘솔을 사용하여 부팅 순서 목록에 항목을 추가합니다.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachine**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. **VM**은 **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 **NIC**를 추가합니다.

7. 저장을 클릭합니다.

참고

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

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

10.3.2. 웹 콘솔에서 부팅 순서 목록 편집

웹 콘솔에서 부팅 순서 목록을 편집합니다.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachine**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 있는 보류 중인 변경 사항 보기를 클릭하면 보류 중인 변경 사항을 볼 수 있습니다. 페이지 상단의 보류 중인 변경 사항 배너에는 가상 머신이 재시작될 때 적용되는 모든 변경 사항 목록이 표시됩니다.

10.3.3. 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.

10.3.4. 웹 콘솔의 부팅 순서 목록에서 항목 제거

웹 콘솔을 사용하여 부팅 순서 목록에서 항목을 제거합니다.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachine**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. **VM**은 **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 있는 보류 중인 변경 사항 보기를 클릭하면 보류 중인 변경 사항을 볼 수 있습니다. 페이지 상단의 보류 중인 변경 사항 배너에는 가상 머신이 재시작될 때 적용되는 모든 변경 사항 목록이 표시됩니다.

10.4. 가상 머신 삭제

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

10.4.1. 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** 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 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.

10.4.2. 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.

10.5. 가상 머신 내보내기

VM(가상 머신) 및 관련 디스크를 내보내 **VM**을 다른 클러스터로 가져오거나 법의학 목적으로 볼륨을 분석할 수 있습니다.

명령줄 인터페이스를 사용하여 **VirtualMachineExport CR**(사용자 정의 리소스)을 생성합니다.

또는 **virtctl vmexport** 명령을 사용하여 **VirtualMachineExport CR**을 생성하고 내보낸 볼륨을 다운로드할 수 있습니다.

10.5.1. VirtualMachineExport 사용자 정의 리소스 생성

VirtualMachineExport 사용자 정의 리소스(**CR**)를 생성하여 다음 오브젝트를 내보낼 수 있습니다.

- **VM**(가상 머신): 지정된 **VM**의 **PVC**(영구 볼륨 클레임)를 내보냅니다.
- **VM** 스냅샷: **VirtualMachineSnapshot CR**에 포함된 **PVC**를 내보냅니다.
- **PVC: PVC**를 내보냅니다. **virt-launcher Pod**와 같은 다른 **Pod**에서 **PVC**를 사용하는 경우 **PVC**가 더 이상 사용되지 않을 때까지 내보내기가 **Pending** 상태로 유지됩니다.

VirtualMachineExport CR은 내보낸 볼륨에 대한 내부 및 외부 링크를 생성합니다. 내부 링크는 클러스터 내에서 유효합니다. 외부 링크는 **Ingress** 또는 경로를 사용하여 액세스할 수 있습니다.

내보내기 서버는 다음 파일 형식을 지원합니다.

- **Raw :** 원시 디스크 이미지 파일.

- **gzip:** 압축 디스크 이미지 파일
- **dir: PVC** 디렉토리 및 파일
- **tar.gz:** 압축된 **PVC** 파일

사전 요구 사항

- **VM** 내보내기에 대해 **VM**을 종료해야 합니다.

절차

1. 다음 예에 따라 **VirtualMachine , VirtualMachine Snapshot** 또는 **PersistentVolumeClaim CR**에서 볼륨을 내보내고 **example-export.yaml** 로 저장할 **VirtualMachineExport** 매니페스트를 생성합니다.

VirtualMachineExport 예

```
apiVersion: export.kubevirt.io/v1alpha1
kind: VirtualMachineExport
metadata:
  name: example-export
spec:
  source:
    apiGroup: "kubevirt.io" (1)
    kind: VirtualMachine (2)
    name: example-vm
  ttlDuration: 1h (3)
```

(1) 적절한 **API** 그룹을 지정합니다.

- **VirtualMachine** 의 **"kubevirt.io ".**

- **VirtualMachineSnapshot** 의 **"snapshot.kubevirt.io ".**
- **PersistentVolumeClaim** 의 **"".**

2. **VirtualMachine,VirtualMachineSnapshot** 또는 **PersistentVolumeClaim** 을 지정합니다.

3. 선택사항입니다. 기본 기간은 **2** 시간입니다.

2. **VirtualMachineExport CR**을 생성합니다.

```
$ oc create -f example-export.yaml
```

3. **VirtualMachineExport CR**을 가져옵니다.

```
$ oc get vmexport example-export -o yaml
```

내보낸 볼륨의 내부 및 외부 링크는 상태 스탠자에 표시됩니다.

출력 예

```
apiVersion: export.kubevirt.io/v1alpha1
kind: VirtualMachineExport
metadata:
  name: example-export
  namespace: example
spec:
  source:
    apiGroup: ""
    kind: PersistentVolumeClaim
    name: example-pvc
  tokenSecretRef: example-token
status:
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2022-06-21T14:10:09Z"
```

```
    reason: podReady
    status: "True"
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2022-06-21T14:09:02Z"
    reason: pvcBound
    status: "True"
    type: PVCReady
  links:
    external: (1)
      cert: |-
        -----BEGIN CERTIFICATE-----
        ...
        -----END CERTIFICATE-----
      volumes:
      - formats:
        - format: raw
          url: https://vmexport-
proxy.test.net/api/export.kubevirt.io/v1alpha1/namespaces/example/virtualmachineexp
orts/example-export/volumes/example-disk/disk.img
        - format: gzip
          url: https://vmexport-
proxy.test.net/api/export.kubevirt.io/v1alpha1/namespaces/example/virtualmachineexp
orts/example-export/volumes/example-disk/disk.img.gz
        name: example-disk
    internal: (2)
      cert: |-
        -----BEGIN CERTIFICATE-----
        ...
        -----END CERTIFICATE-----
      volumes:
      - formats:
        - format: raw
          url: https://virt-export-example-export.example.svc/volumes/example-
disk/disk.img
        - format: gzip
          url: https://virt-export-example-export.example.svc/volumes/example-
disk/disk.img.gz
        name: example-disk
  phase: Ready
  serviceName: virt-export-example-export
```

(1) 외부 링크는 **Ingress** 또는 경로를 사용하여 클러스터 외부에서 액세스할 수 있습니다.

(2) 내부 링크는 클러스터 내에서만 유효합니다.

10.6. 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 **VM**을 일시 중지하거나 포트를 노출할 수 있습니다**.**

10.6.1. 가상 머신 인스턴스 정보

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

독립 실행형 **VMI**는 스크립트**,** 자동화 또는 **CLI**의 다른 방법을 통해 독립적으로 생성 및 시작됩니다**. OpenShift Virtualization** 환경 외부에서 개발 및 시작된 독립 실행형 **VMI**가 사용자 환경에 있을 수 있습니다**. CLI**를 사용하여 이러한 독립 실행형 **VMI**를 계속 관리할 수 있습니다**.** 다음과 같이 독립 실행형 **VMI** 와 관련된 특정 작업에 웹 콘솔을 사용할 수도 있습니다**.**

- 독립 실행형 **VMI** 및 세부 정보를 나열합니다**.**
- 독립 실행형 **VMI**의 라벨 및 주석을 편집합니다**.**
- 독립 실행형 **VMI**를 삭제합니다**.**

VM을 삭제하면 관련 **VMI**가 자동으로 삭제됩니다**.** 독립 실행형 **VMI**는 **VM** 또는 다른 오브젝트에 속하지 않기 때문에 직접 삭제합니다**.**

참고

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

10.6.2. CLI를 사용하여 모든 가상 머신 인스턴스 나열

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

절차

- 다음 명령을 실행하여 **VMI**를 모두 나열합니다**.**

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

10.6.3. 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 **VMI**는 웹 콘솔에 표시되지 않습니다**.** 웹 콘솔에는 독립 실행형 **VMI**만 표시됩니다**.** 클러스터의 모든 **VMI**를 나열하려면 **CLI**를 사용해야 합니다**.**

절차

- 사이드 메뉴에서 가상화 → **VirtualMachine**를 클릭합니다**.**

  이름 옆에 있는 다크색 배지로 독립 실행형 **VMI**를 확인할 수 있습니다**.**

10.6.4. 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**.** 다른 필드는 편집할 수 없습니다**.**

절차

1. **OpenShift Container Platform** 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 를 클릭합니다**.**

2. **VirtualMachineInstance** 세부 정보 페이지를 열 수 있도록 독립 실행형 **VMI**를 선택합니다**.**

3. 세부 정보 탭에서 주석 또는 라벨 옆에 있는 연필 아이콘을 클릭합니다.

4. 관련 사항을 변경하고 저장을 클릭합니다.

10.6.5. 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 **VMI**의 이름을 확인합니다.

절차

- 다음 명령을 실행하여 **VMI**를 삭제합니다.

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

10.6.6. 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 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 를 클릭합니다.

2. 작업 → **VirtualMachineInstance** 삭제를 클릭합니다.

3. 확인 팝업 창에서 삭제를 클릭하여 독립 실행형 **VMI**를 영구적으로 삭제합니다.

10.7. 가상 머신 상태 제어

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

virtctl 을 사용하여 가상 머신 상태를 관리하고 **CLI**에서 다른 작업을 수행할 수 있습니다. 예를 들어 **virtctl** 을 사용하여 **VM**을 강제로 중지하거나 포트를 노출할 수 있습니다.

10.7.1. 가상 머신 시작

웹 콘솔에서 가상 머신을 시작할 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachine**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 머신의 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 상태가 가져오는 중이 되고 **OpenShift Virtualization**은 **URL** 끝점에서 컨테이너를 가져옵니다. 이미지 크기에 따라 이 프로세스에 몇 분이 걸릴 수 있습니다.

10.7.2. 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 → **VirtualMachine**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 머신의 이름을 클릭하여 **VirtualMachine** 세부 정보 페이지에 액세스합니다.

b. 작업 → 재시작 을 클릭합니다.

4. 확인 창에서 재시작을 클릭하여 가상 머신을 재시작합니다.

10.7.3. 가상 머신 중지

웹 콘솔에서 가상 머신을 중지할 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachine**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 머신의 이름을 클릭하여 **VirtualMachine** 세부 정보 페이지에 액세스합니다.

  b. 작업 → 중지 를 클릭합니다.

4. 확인 창에서 중지를 클릭하여 가상 머신을 중지합니다.

10.7.4. 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 → **VirtualMachine**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 머신의 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.

10.8. 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.

참고

단일 가상 머신에 대한 동시 **VNC** 연결 실행은 현재 지원되지 않습니다.

10.8.1. 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.

10.8.1.1. 직렬 콘솔 연결

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

절차

1. **OpenShift Container Platform** 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 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 열려 있는지 확인합니다. 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.

10.8.1.2. VNC 콘솔에 연결

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

절차

1. **OpenShift Container Platform** 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 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 키 조합을 가상 머신에 보냅니다.

6. 콘솔 창 외부에서 클릭한 다음 연결 끊기 를 클릭하여 세션을 종료합니다.

10.8.1.3. RDP를 사용하여 Windows 가상 머신에 연결

RDP(Remote Desktop Protocol)를 사용하는 데스크탑 뷰어 콘솔은 **Windows** 가상 머신 연결을 위해 개선된 콘솔 환경을 제공합니다.

RDP를 사용하여 **Windows** 가상 머신에 연결하려면 웹 콘솔의 **VirtualMachine** 세부 정보 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.
- **RDP** 클라이언트가 **Windows** 가상 머신과 동일한 네트워크의 머신에 설치되어 있습니다.

절차

1. **OpenShift Container Platform** 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 를 클릭합니다.
2. **Windows** 가상 머신을 클릭하여 **VirtualMachine** 세부 정보 페이지를 엽니다.
3. 콘솔 탭을 클릭합니다.
4. 콘솔 목록에서 데스크탑 뷰어를 선택합니다.
5. 원격 데스크탑 시작을 클릭하여 **console.rdp** 파일을 다운로드합니다.
6. 선호하는 **RDP** 클라이언트의 **console.rdp** 파일을 참조하여 **Windows** 가상 머신에 연결합니다.

10.8.1.4. 가상 머신 간 전환

Windows 가상 머신(**VM**)에 **vGPU**가 연결된 경우 웹 콘솔을 사용하여 기본 디스플레이와 **vGPU** 표시 간에 전환할 수 있습니다.

사전 요구 사항

- 중재된 장치는 **HyperConverged** 사용자 정의 리소스에서 구성되며 **VM**에 할당됩니다.

- **VM**이 실행 중입니다.

절차

1. **OpenShift Container Platform** 콘솔에서 가상화 → **VirtualMachine**를클릭합니다.

2. **Windows** 가상 머신을 선택하여 개요 화면을 엽니다.

3. 콘솔 탭을 클릭합니다.

4. 콘솔 목록에서 **VNC** 콘솔 을 선택합니다.

5. **Send Key** 목록에서 적절한 키 조합을 선택합니다.

   a. 기본 **VM** 디스플레이에 액세스하려면 **Ctl + Alt+ 1** 을 선택합니다.

   b. **vGPU** 디스플레이에 액세스하려면 **Ctl + Alt + 2** 를 선택합니다.

추가 리소스

- 중재된 장치 구성

10.8.1.5. 웹 콘솔을 사용하여 SSH 명령 복사

SSH를 통해 **VM(**가상 머신**)** 터미널에 연결하도록 명령을 복사합니다.

절차

1. **OpenShift Container Platform** 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 를 클릭합니다.

2. 가상 머신의 옵션 메뉴 ⋮ 를 클릭하고 **SSH** 복사 명령을 선택합니다.

3. 터미널에 붙여넣어 **VM**에 액세스합니다.

10.8.2. CLI 명령을 사용하여 가상 머신 콘솔에 액세스

10.8.2.1. virtctl을 사용하여 SSH를 통해 가상 머신에 액세스

virtctl ssh 명령을 사용하여 로컬 **SSH** 클라이언트를 사용하여 **VM(**가상 머신**)**으로 **SSH** 트래픽을 전달할 수 있습니다**. VM**을 사용하여 **SSH** 키 인증을 구성한 경우 **1**단계가 필요하지 않기 때문에 절차의 **2**단계로 건너뜁니다**.**

참고

컨트롤 플레인에서 **SSH** 트래픽이 많은 경우 **API** 서버의 속도가 느려질 수 있습니다**.** 많은 연결이 정기적으로 필요한 경우 전용 **Kubernetes** 서비스 오브젝트를 사용하여 가상 머신에 액세스합니다**.**

사전 요구 사항

- **OpenShift CLI(oc)**가 설치되어 있습니다**.**
- **virtctl** 클라이언트가 설치되어 있어야 합니다**.**
- 액세스하려는 가상 머신이 실행 중입니다**.**
- **VM**과 동일한 프로젝트에 있습니다**.**

절차

1. **SSH** 키 인증을 구성합니다**.**

a. **ssh-keygen** 명령을 사용하여 **SSH** 공개 키 쌍을 생성합니다.

```
$ ssh-keygen -f <key_file> (1)
```

(1) 키를 저장할 파일을 지정합니다.

b. **VM**에 액세스하기 위한 **SSH** 공개 키가 포함된 **SSH** 인증 보안을 생성합니다.

```
$ oc create secret generic my-pub-key --from-file=key1=<key_file>.pub
```

c. **VirtualMachine** 매니페스트의 보안에 대한 참조를 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: testvm
spec:
  running: true
  template:
    spec:
      accessCredentials:
      - sshPublicKey:
          source:
            secret:
              secretName: my-pub-key (1)
          propagationMethod:
            configDrive: {} (2)
# ...
```

(1) **SSH** 인증 보안 오브젝트에 대한 참조 입니다.

(2) **SSH** 공개 키는 **VM**에 **configDrive** 공급자를 사용하여 **cloud-init** 메타데이터로 삽입됩니다.

d. **VM**을 다시 시작하여 변경 사항을 적용합니다.

2. **SSH**를 통해 **VM**에 연결합니다.

   a. 다음 명령을 실행하여 **SSH**를 통해 **VM**에 액세스합니다.

   ```
   $ virtctl ssh -i <key_file> <vm_username>@<vm_name>
   ```

   b. 선택 사항**: VM**으로 파일을 안전하게 전송하거나 **VM**에서 전송하려면 다음 명령을 사용합니다.

   시스템에서 **VM**으로 파일을 복사

   ```
   $ virtctl scp -i <key_file> <filename> <vm_username>@<vm_name>:
   ```

   VM에서 시스템으로 파일을 복사

   ```
   $ virtctl scp -i <key_file> <vm_username@<vm_name>:<filename> .
   ```

추가 리소스

- 가상 머신을 노출하는 서비스 생성
- 보안 이해

10.8.2.2. OpenSSH 및 virtctl port-forward 사용

로컬 **OpenSSH** 클라이언트와 **virtctl port-forward** 명령을 사용하여 실행 중인 **VM(**가상 머신**)**에 연결할 수 있습니다**.** 이 방법을 **Ansible**과 함께 사용하여 **VM** 구성을 자동화할 수 있습니다**.**

이 방법은 컨트롤 플레인을 통해 포트 전달 트래픽이 전송되므로 트래픽이 적은 애플리케이션에 권장됩니다. 이 방법은 **API** 서버에 많은 부담을 주기 때문에 **Rsync** 또는 원격 데스크탑 프로토콜과 같은 트래픽이 많은 애플리케이션에 권장되지 않습니다.

사전 요구 사항

- **virtctl** 클라이언트가 설치되어 있어야 합니다.
- 액세스하려는 가상 머신이 실행 중입니다.
- **virtctl** 툴을 설치한 환경에는 **VM**에 액세스하는 데 필요한 클러스터 권한이 있습니다. 예를 들어 **oc login** 을 실행하거나 **KUBECONFIG** 환경 변수를 설정합니다.

절차

1. 클라이언트 머신의 **~/.ssh/config** 파일에 다음 텍스트를 추가합니다.

```
Host vm/*
  ProxyCommand virtctl port-forward --stdio=true %h %p
```

2. 다음 명령을 실행하여 **VM**에 연결합니다.

```
$ ssh <user>@vm/<vm_name>.<namespace>
```

10.8.2.3. 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>
```

10.8.2.4. 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 **SSH**를 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
```

10.8.2.5. RDP 콘솔을 사용하여 Windows 가상 머신에 연결

로컬 **RDP(Remote Desktop Protocol)** 클라이언트를 사용하여 **Windows VM(**가상 머신**)**에 연결할 **Kubernetes** 서비스 오브젝트를 생성합니다.

사전 요구 사항

- **Windows** 가상 머신이 실행 중이고 **QEMU** 게스트 에이전트가 설치되어 있습니다. **qemu-guest-agent** 오브젝트는 **VirtIO** 드라이버에 포함되어 있습니다.

- 로컬 머신에 **RDP** 클라이언트가 설치되어 있어야 합니다.

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 **Pod**로 전달됩니다. **special: key** 레이블은 서비스 매니페스트의 **spec.selector** 특성의 레이블과 일치해야 합니다.

2. **VirtualMachine** 매니페스트 파일을 저장하여 변경 사항을 적용합니다.

3. **VM**을 노출할 서비스 매니페스트를 생성합니다.

```
apiVersion: v1
kind: Service
metadata:
  name: rdpservice (1)
  namespace: example-namespace (2)
```

```
spec:
  ports:
  - targetPort: 3389 (3)
    protocol: TCP
  selector:
    special: key (4)
  type: NodePort (5)
# ...
```

1. **Service** 오브젝트의 이름입니다.

2. **Service** 오브젝트가 있는 네임스페이스입니다. 이는 **VirtualMachine** 매니페스트의 **metadata.namespace** 필드와 일치해야 합니다.

3. 서비스에서 노출할 **VM** 포트입니다. 포트 목록이 **VM** 매니페스트에 정의된 경우 열려 있는 포트를 참조해야 합니다.

4. **VirtualMachine** 매니페스트의 **spec.template.metadata.labels** 스탠자에 추가한 라벨 참조입니다.

5. 서비스 유형입니다.

4. 서비스 매니페스트 파일을 저장합니다.

5. 다음 명령을 실행하여 서비스를 생성합니다.

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

6. **VM**을 시작합니다. **VM**이 이미 실행 중인 경우 다시 시작합니다.

7. **Service** 오브젝트를 쿼리하여 사용할 수 있는지 확인합니다.

```
$ oc get service -n example-namespace
```

NodePort 서비스의 출력 예

```
NAME        TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
rdpservice  NodePort   172.30.232.73  <none>        3389:30000/TCP   5m
```

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  v1.24.0  192.168.55.101  <none>
```

9. 원하는 **RDP** 클라이언트에 노드 **IP** 주소와 할당된 포트를 지정합니다.

10. 사용자 이름과 암호를 입력하여 **Windows** 가상 머신에 연결합니다.

10.9. SYSPREP로 WINDOWS 설치 자동화

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

10.9.1. Windows DVD를 사용하여 VM 디스크 이미지 생성

Microsoft는 다운로드를 위한 디스크 이미지를 제공하지 않지만 **Windows DVD**를 사용하여 디스크 이미지를 만들 수 있습니다. 그러면 이 디스크 이미지를 사용하여 가상 머신을 생성할 수 있습니다.

절차

1. **OpenShift Virtualization** 웹 콘솔에서 스토리지 → **PersistentVolumeClaims** → 데이터 업로드 양식을 사용하여 **PersistentVolumeClaim** 생성을 클릭합니다.

2. 원하는 프로젝트를 선택합니다.

3. 영구 볼륨 클레임 이름을 으로 설정합니다.

4. **Windows DVD**에서 **VM** 디스크 이미지를 업로드합니다. 이제 이미지를 부팅 소스로 사용하여 새 **Windows VM**을 생성할 수 있습니다.

10.9.2. 디스크 이미지를 사용하여 Windows 설치

디스크 이미지를 사용하여 가상 머신에 **Windows**를 설치할 수 있습니다.

사전 요구 사항

- **Windows DVD**를 사용하여 디스크 이미지를 만들어야 합니다.

- **autounattend.xml** 응답 파일을 만들어야 합니다. 자세한 내용은 **Microsoft** 설명서를 참조하십시오.

절차

1. **OpenShift Container Platform** 콘솔의 사이드 메뉴에서 가상화 → 카탈로그 를 클릭합니다.

2. **Windows** 템플릿을 선택하고 사용자 지정 **VirtualMachine** 를 클릭합니다.

3. 디스크 소스 목록에서 **Upload a new file to a PVC(**새 파일을 **PVC**로 업로드**)** 를 선택하고 **DVD** 이미지로 이동합니다.

4. 검토를 클릭하고 **VirtualMachine**를 생성합니다.

5. 이 가상 머신에 사용 가능한 운영 체제의 명확한 복제.

6. 생성 후 이 **VirtualMachine** 를 지웁니다.

7. 스크립트 탭의 **Sysprep** 섹션에서 편집 을 클릭합니다.

8. **autountend.xml** 응답 파일로 이동하여 저장을 클릭합니다.

9. **Create VirtualMachine** 를 클릭합니다.

10. **YAML** 탭에서 **running:false** 를 **runStrategy: RerunOnFailure** 로 교체하고 저장을 클릭합니다.

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

10.9.3. sysprep을 사용하여 Windows VM 일반화

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

VM을 일반화하기 전에 시스템 시스템 설치 후 **sysprep** 툴에서 응답 파일을 탐지할 수 없는지 확인해야 합니다.

절차

1. **OpenShift Container Platform** 콘솔에서 가상화 → **VirtualMachine**를 클릭합니다.

2. **Windows VM**을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.

3. 디스크 탭을 클릭합니다.

4.

sysprep 디스크의 옵션 메뉴

⋮

를 클릭하고 분리를 선택합니다.

5. **Detach** (분리)를 클릭합니다.

6. **sysprep** 툴로 탐지되지 않도록 **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 VM**이 종료됩니다. 이제 **VM**의 디스크 이미지를 **Windows VM**의 설치 이미지로 사용할 수 있습니다.

이제 **VM**을 특수화할 수 있습니다.

10.9.4. Windows 가상 머신 전문 설정

가상 머신(**VM**)을 전문으로 설정하면 일반 **Windows** 이미지에서 **VM**으로 컴퓨터 관련 정보를 구성합니다.

사전 요구 사항

- 일반적인 **Windows** 디스크 이미지가 있어야 합니다.
- **unattend.xml** 응답 파일을 만들어야 합니다. 자세한 내용은 **Microsoft** 설명서를 참조하십시오.

절차

1. **OpenShift Container Platform** 콘솔에서 가상화 → 카탈로그 를 클릭합니다.

2. **Windows** 템플릿을 선택하고 사용자 지정 **VirtualMachine** 를 클릭합니다.

3. **Disk** 소스 목록에서 **PVC(clone PVC)** 를 선택합니다.

4. 영구 볼륨 클레임 프로젝트 및 일반화된 **Windows** 이미지의 영구 볼륨 클레임 이름을 지정합니다.

5. 검토를 클릭하고 **VirtualMachine**를 생성합니다.

6. 스크립트 탭을 클릭합니다.

7. **Sysprep** 섹션에서 편집 을 클릭하고 **unattend.xml** 응답 파일을 찾은 다음 저장을 클릭합니다.

8. **Create VirtualMachine** 를 클릭합니다.

초기 부팅 중에 **Windows**는 **unattend.xml** 응답 파일을 사용하여 **VM**을 전문으로 설정합니다. 이제 **VM**을 사용할 준비가 되었습니다.

10.9.5. 추가 리소스

- 가상 머신 생성
- **Microsoft, Sysprep (Generalize) Windows** 설치
- **Microsoft, generalize**
- **Microsoft, Specialize**

10.10. 실패한 노드를 해결하여 가상 머신 장애 조치 트리거

노드가 실패하고 머신 상태 점검 이 클러스터에 배포되지 않으면 **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 인프라 를 사용하여 클러스터를 설치하고 머신 상태 점검을 올바르게 구성한 경우 다음을 수행합니다.

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

10.10.1. 사전 요구 사항

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

10.10.2. 베어 메탈 클러스터에서 노드 삭제

CLI를 사용하여 노드를 삭제하면 **Kubernetes**에서 노드 오브젝트가 삭제되지만 노드에 존재하는 **Pod**는 삭제되지 않습니다. 복제 컨트롤러에서 지원하지 않는 기본 **Pod**는 **OpenShift Container Platform**에 액세스할 수 없습니다. 복제 컨트롤러에서 지원하는 **Pod**는 사용 가능한 다른 노드로 다시 예약됩니다. 로컬 매니페스트 **Pod**를 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 **Pod**를 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.

10.10.3. 가상 머신 장애 조치 확인

비정상 노드에서 모든 리소스가 종료되면 **VM**이 재배치될 때마다 정상 노드에 새 **VMI**(가상 머신 인스턴스)가 자동으로 생성됩니다. **VMI**가 생성되었는지 확인하려면 **oc CLI**를 사용하여 모든 **VMI**를 확인합니다.

10.10.3.1. CLI를 사용하여 모든 가상 머신 인스턴스 나열

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

절차

- 다음 명령을 실행하여 **VMI**를 모두 나열합니다.

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

10.11. 가상 머신에 QEMU 게스트 에이전트 설치

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

10.11.1. 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 AgentConnected가 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
```

10.11.2. 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**) VM**의 스냅샷을 생성하려면 **QEMU** 게스트 에이전트를 설치합니다.

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

10.11.2.1. 기존 **Windows** 가상 머신에 **VirtIO** 드라이버 설치

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

참고

다음 절차에서는 일반적인 방법을 사용하여 **Windows**에 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 버튼

으로 클릭하고 속성을 선택합니다.

b. 세부 정보 탭을 클릭하고 속성 목록에서 하드웨어 **ID**를 선택합니다.

c. 하드웨어 **ID**의 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.

10.11.2.2. 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.

10.12. 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.

10.12.1. 사전 요구 사항

- 가상 머신에 **QEMU** 게스트 에이전트 를 설치합니다.

10.12.2. 웹 콘솔의 QEMU 게스트 에이전트 정보

QEMU 게스트 에이전트가 설치되면 **VirtualMachine** 세부 정보 페이지의 개요 및 세부 정보 탭에 호스트 이름, 운영 체제, 시간대 및 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.

10.12.3. 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 → **VirtualMachine**를 클릭합니다.
2. 가상 머신 이름을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.
3. 세부 정보 탭을 클릭하여 활성 사용자를 확인합니다.
4. 파일 시스템에 대한 정보를 보려면 디스크 탭을 클릭합니다.

10.13. 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.

10.13.1. 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 → **VirtualMachine**를 클릭합니다.
2. 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.
3. 환경 탭에서 구성 맵, 시크릿 또는 서비스 계정 추가를 클릭합니다.
4. **Select a resource** 를 클릭하고 목록에서 리소스를 선택합니다. 선택한 리소스에 대해 **6**자리 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 머신을 재시작합니다.

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

10.13.2. 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 → **VirtualMachine**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.

10.13.3. 추가 리소스

- **Pod**에 민감한 데이터 제공
- 서비스 계정 이해 및 생성
- 구성 맵 이해

10.14. 기존 WINDOWS 가상 머신에 VIRTIO 드라이버 설치

10.14.1. VirtIO 드라이버 정보

VirtIO 드라이버는 **Microsoft Windows** 가상 머신을 **OpenShift Virtualization**에서 실행하는 데 필요한 준가상화 장치 드라이버입니다. 지원되는 드라이버는 **Red Hat Ecosystem Catalog**의 **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 **Windows**를 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 설치 도 참조하십시오.

10.14.2. Microsoft Windows 가상 머신에 지원되는 VirtIO 드라이버

표 **10.1.** 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.
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.

10.14.3. 가상 머신에 VirtIO 드라이버 컨테이너 디스크 추가

OpenShift Virtualization에서는 **Microsoft Windows**용 **VirtIO** 드라이버를 컨테이너 디스크로 배포하며, **Red Hat Ecosystem Catalog**에서 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 Catalog**에서 **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 Virtualization**은 **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 **CLI**에 **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.

10.14.4. 기존 Windows 가상 머신에 VirtIO 드라이버 설치

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

참고

다음 절차에서는 일반적인 방법을 사용하여 **Windows**에 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 **ID**를 선택합니다.

   c. 하드웨어 **ID**의 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.

10.14.5. 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.

10.15. 새로운 WINDOWS 가상 머신에 VIRTIO 드라이버 설치

10.15.1. 사전 요구 사항

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

10.15.2. VirtIO 드라이버 정보

VirtIO 드라이버는 **Microsoft Windows** 가상 머신을 **OpenShift Virtualization**에서 실행하는 데 필요한 준가상화 장치 드라이버입니다. 지원되는 드라이버는 **Red Hat Ecosystem Catalog**의 **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 **Windows**를 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 설치 도 참조하십시오.

10.15.3. Microsoft Windows 가상 머신에 지원되는 VirtIO 드라이버

표 **10.2.** 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.
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.

10.15.4. 가상 머신에 VirtIO 드라이버 컨테이너 디스크 추가

OpenShift Virtualization에서는 **Microsoft Windows**용 **VirtIO** 드라이버를 컨테이너 디스크로 배포하며, **Red Hat Ecosystem Catalog**에서 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 Catalog**에서 **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 **CLI**에 **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.

10.15.5. 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.

10.15.6. 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.

10.16. 가상 신뢰할 수 있는 플랫폼 모듈 장치 사용

VirtualMachine (VM) 또는 **VirtualMachine Instance (VM)** 매니페스트를 편집하여 **vTPM(**가상 신뢰할 수 있는 플랫폼 모듈**)** 장치를 새 또는 기존 가상 머신에 추가합니다.

10.16.1. vTPM 장치 정보

TPM(가상 신뢰할 수 있는 플랫폼 모듈**)** 장치 기능은**TPM(**물리적으로 신뢰할 수 있는 플랫폼 모듈**)** 하드웨어 칩과 같습니다.

모든 운영 체제와 함께 **vTPM** 장치를 사용할 수 있지만 **Windows 11**은 설치 또는 부팅하기 위해 **TPM** 칩이 있어야 합니다. **vTPM** 장치를 사용하면 **Windows 11** 이미지에서 생성된 **VM**이 물리적 **TPM** 칩 없이 작동할 수 있습니다.

vTPM을 활성화하지 않으면 노드에 하나씩 있어도 **VM**에서 **TPM** 장치를 인식하지 않습니다.

또한 **vTPM** 장치는 물리적 하드웨어 없이 시크릿을 일시적으로 저장하여 가상 머신을 보호합니다. 그러나 영구 시크릿 스토리지에 **vTPM** 사용은 현재 지원되지 않습니다. **vTPM**은 **VM**이 종료된 후 저장된 시크릿을 삭제합니다.

10.16.2. 가상 머신에 vTPM 장치 추가

vTPM(가상 신뢰할 수 있는 플랫폼 모듈**)** 장치를 **VM(**가상 머신**)**에 추가하면 물리적 **TPM** 장치 없이 **Windows 11** 이미지에서 생성된 **VM**을 실행할 수 있습니다. **vTPM** 장치는 해당 **VM**의 시크릿도 일시적으로 저장합니다.

절차

1. 다음 명령을 실행하여 **VM** 구성을 업데이트합니다.

```
$ oc edit vm <vm_name>
```

2. **tpm: {}** 행을 포함하도록 **VM** 사양 을 편집합니다. 예를 들면 다음과 같습니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: example-vm
spec:
 template:
  spec:
   domain:
    devices:
     tpm: {} (1)
...
```

(1) **TPM** 장치를 **VM**에 추가합니다.

3. 변경 사항을 적용하려면 편집기를 저장하고 종료합니다.

4. 선택 사항: 실행 중인 가상 머신을 편집한 경우 변경 사항을 적용하려면 가상 머신을 다시 시작해야 합니다.

10.17. OPENSHIFT PIPELINES를 사용하여 가상 머신 관리

Red Hat OpenShift Pipelines 는 **Kubernetes** 네이티브 **CI/CD** 프레임워크로, 개발자가 자체 컨테이너에서 **CI/CD** 파이프라인의 각 단계를 설계 및 실행할 수 있습니다.

Tekton Tasks Operator(TTO)는 **OpenShift Virtualization**을 **OpenShift Pipelines**와 통합합니다. **TTO**에는 다음을 수행할 수 있는 클러스터 작업 및 예제 파이프라인이 포함되어 있습니다.

- **VM**(가상 머신), **PVC**(영구 볼륨 클레임) 및 데이터 볼륨 생성 및 관리
- **VM**에서 명령 실행
- **libguestfs** 툴을 사용하여 디스크 이미지 조작

중요

Red Hat OpenShift Pipelines를 사용하여 가상 머신을 관리하는 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.

10.17.1. 사전 요구 사항

- **cluster-admin** 권한이 있는 **OpenShift Container Platform** 클러스터에 액세스할 수 있습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.
- **OpenShift Pipelines**를 설치했습니다.

10.17.2. Tekton Tasks Operator 리소스 배포

OpenShift Virtualization을 설치할 때 **TTO(Tekton Tasks Operator)** 클러스터 작업 및 예제 파이프라인은 기본적으로 배포되지 않습니다. **TTO** 리소스를 배포하려면 **HyperConverged CR(**사용자 정의 리소스**)**에서 **deployTektonTaskResources** 기능 게이트를 활성화합니다.

절차

1. 다음 명령을 실행하여 기본 편집기에서 **HyperConverged CR**을 엽니다.

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

2. **spec.featureGates.deployTektonTaskResources** 필드를 **true** 로 설정합니다.

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

```
spec:
  tektonPipelinesNamespace: <user_namespace> (1)
  featureGates:
    deployTektonTaskResources: true (2)
#...
```

(1) 파이프라인을 실행할 네임스페이스입니다.

(2) **TTO** 리소스를 배포하기 위해 활성화할 기능 게이트입니다.

참고

나중에 기능 게이트를 비활성화해도 클러스터 작업 및 예제 파이프라인은 계속 사용할 수 있습니다.

3. 변경 사항을 저장하고 편집기를 종료합니다.

10.17.3. Tekton Tasks Operator에서 지원하는 가상 머신 작업

다음 표에서는 **Tekton Tasks Operator**의 일부로 포함된 클러스터 작업을 보여줍니다.

표 **10.3. Tekton Tasks Operator**에서 지원하는 가상 머신 작업

Task	설명
create-vm-from-template	템플릿에서 가상 머신을 생성합니다.
copy-template	가상 머신 템플릿을 복사합니다.
modify-vm-template	가상 머신 템플릿을 수정합니다.
modify-data-object	데이터 볼륨 또는 데이터 소스를 생성하거나 삭제합니다.
cleanup-vm	가상 머신에서 스크립트 또는 명령을 실행하고 나중에 가상 머신을 중지하거나 삭제합니다.
disk-virt-customize	**virt-customize** 툴을 사용하여 대상 PVC에서 사용자 지정 스크립트를 실행합니다.

Task	설명
disk-virt-sysprep	**virt-sysprep** 도구를 사용하여 대상 PVC에서 sysprep 스크립트를 실행합니다.
wait-for-vmi-status	가상 머신 인스턴스의 특정 상태를 기다린 후 상태에 따라 실패하거나 성공합니다.

10.17.4. 파이프라인 예

Tekton Tasks Operator에는 다음 예제 **Pipeline** 매니페스트가 포함되어 있습니다. 웹 콘솔 또는 **CLI**를 사용하여 예제 파이프라인을 실행할 수 있습니다.

Windows 10 설치 프로그램 파이프라인

이 파이프라인은 **Windows** 설치 이미지(**ISO** 파일)에서 **Windows 10**을 새 데이터 볼륨에 설치합니다. 사용자 정의 응답 파일은 설치 프로세스를 실행하는 데 사용됩니다.

Windows 10 사용자 정의 파이프라인

이 파이프라인은 기본 **Windows 10** 설치의 데이터 볼륨을 복제하고 **Microsoft SQL Server Express**를 설치하여 사용자 지정한 다음 새 이미지와 템플릿을 만듭니다.

10.17.4.1. 웹 콘솔을 사용하여 예제 파이프라인 실행

웹 콘솔의 파이프라인 메뉴에서 예제 파이프라인 을 실행할 수 있습니다.

절차

1. 사이드 메뉴에서 파이프라인 → 파이프라인 을 클릭합니다.
2. 파이프라인 세부 정보 페이지를 열려면 파이프라인 을 선택합니다.
3. 작업 목록에서 시작을 선택합니다. **Start Pipeline** 대화 상자가 표시됩니다.
4. 매개변수의 기본값을 유지하고 시작을 클릭하여 파이프라인을 실행합니다. 세부 정보 탭은 각 작업의 진행 상황을 추적하고 파이프라인 상태를 표시합니다.

10.17.4.2. CLI를 사용하여 예제 파이프라인 실행

PipelineRun 리소스를 사용하여 예제 파이프라인을 실행합니다. **PipelineRun** 오브젝트는 파이프라인에서 실행 중인 인스턴스입니다. 클러스터에서 특정 입력, 출력 및 실행 매개변수를 사용하여 실행할 파이프라인을 인스턴스화합니다. 또한 파이프라인의 각 작업에 대해 **TaskRun** 오브젝트를 생성합니다.

절차

1. **Windows 10** 설치 프로그램 파이프라인을 실행하려면 다음 **PipelineRun** 매니페스트를 생성합니다.

```
apiVersion: tekton.dev/v1beta1
kind: PipelineRun
metadata:
  generateName: windows10-installer-run-
  labels:
    pipelinerun: windows10-installer-run
spec:
  params:
  - name: winImageDownloadURL
    value: <link_to_windows_10_iso> (1)
  pipelineRef:
    name: windows10-installer
  taskRunSpecs:
    - pipelineTaskName: copy-template
      taskServiceAccountName: copy-template-task
    - pipelineTaskName: modify-vm-template
      taskServiceAccountName: modify-vm-template-task
    - pipelineTaskName: create-vm-from-template
      taskServiceAccountName: create-vm-from-template-task
    - pipelineTaskName: wait-for-vmi-status
      taskServiceAccountName: wait-for-vmi-status-task
    - pipelineTaskName: create-base-dv
      taskServiceAccountName: modify-data-object-task
    - pipelineTaskName: cleanup-vm
      taskServiceAccountName: cleanup-vm-task
  status: {}
```

(1) **Windows 10 64**비트 **ISO** 파일의 **URL**을 지정합니다. 제품 언어는 영어**(United States)**여야 합니다.

2. **PipelineRun** 매니페스트를 적용합니다.

```
$ oc apply -f windows10-installer-run.yaml
```

3. **Windows 10** 사용자 지정 파이프라인을 실행하려면 다음 **PipelineRun** 매니페스트를 생성

합니다.

```
apiVersion: tekton.dev/v1beta1
kind: PipelineRun
metadata:
  generateName: windows10-customize-run-
  labels:
    pipelinerun: windows10-customize-run
spec:
  params:
    - name: allowReplaceGoldenTemplate
      value: true
    - name: allowReplaceCustomizationTemplate
      value: true
  pipelineRef:
    name: windows10-customize
  taskRunSpecs:
    - pipelineTaskName: copy-template-customize
      taskServiceAccountName: copy-template-task
    - pipelineTaskName: modify-vm-template-customize
      taskServiceAccountName: modify-vm-template-task
    - pipelineTaskName: create-vm-from-template
      taskServiceAccountName: create-vm-from-template-task
    - pipelineTaskName: wait-for-vmi-status
      taskServiceAccountName: wait-for-vmi-status-task
    - pipelineTaskName: create-base-dv
      taskServiceAccountName: modify-data-object-task
    - pipelineTaskName: cleanup-vm
      taskServiceAccountName: cleanup-vm-task
    - pipelineTaskName: copy-template-golden
      taskServiceAccountName: copy-template-task
    - pipelineTaskName: modify-vm-template-golden
      taskServiceAccountName: modify-vm-template-task
status: {}
```

4. **PipelineRun** 매니페스트를 적용합니다.

```
$ oc apply -f windows10-customize-run.yaml
```

10.17.5. 추가 리소스

- **Red Hat OpenShift Pipelines**를 사용하여 애플리케이션용 **CI/CD** 솔루션 생성

10.18. 고급 가상 머신 관리

10.18.1. 가상 머신의 리소스 할당량 작업

가상 시스템의 리소스 할당량을 생성하고 관리합니다.

10.18.1.1. 가상 머신의 리소스 할당량 제한 설정

요청만 사용하는 리소스 할당량은 VM(가상 머신)에서 자동으로 작동합니다. 리소스 할당량이 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 큰 100Mi 이상이므로 이 구성이 지원됩니다.

2. VirtualMachine 매니페스트를 저장합니다.

10.18.1.2. 추가 리소스

- 프로젝트당 리소스 할당량
- 다중 프로젝트의 리소스 할당량

10.18.2. 가상 머신용 노드 지정

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

10.18.2.1. 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 **VM**이 있습니다. 내결함성을 보장하기 위해 서로 다른 노드에서 실행하려고 합니다.
- 두 개의 가상 머신이 있습니다. 중복 노드 간 라우팅을 방지하기 위해 **VM**을 동일한 노드에서 실행하려고 합니다.
- **VM**에는 사용 가능한 모든 노드에 존재하지 않는 특정 하드웨어 기능이 필요합니다.
- 노드에 기능을 추가하는 **Pod**가 있으며 해당 노드에 **VM**을 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 지원

됩니다. **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 Platform**은 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.

10.18.2.2. 노드 배치의 예

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

10.18.2.2.1. 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
...
```

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

이 예에서는 **example-key-1 = example-value-1** 레이블이 있는 실행 중인 **pod**가 있는 노드에 **VM**을 예약해야 합니다. 노드에 실행 중인 **Pod**가 없는 경우 **VM**은 예약되지 않습니다.

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

VM 매니페스트 예

```
metadata:
  name: example-vm-pod-affinity
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  template:
    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 **VM**이 예약되지 않습니다.

2

preferredDuringSchedulingIgnoredDuringExecution 규칙 유형을 사용하는 경우 필요한 모든 제약 조건이 충족되면 여전히 **VM**이 예약됩니다.

10.18.2.2.3. 예: 노드 선호도를 사용한 **VM** 노드 배치

이 예에서 **VM**은 **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 **VM**이 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:
  template:
    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 **VM**이 예약되지 않습니다.

2

preferredDuringSchedulingIgnoredDuringExecution 규칙 유형을 사용하는 경우 필요한 모든 제약 조건이 충족되면 여전히 **VM**이 예약됩니다.

10.18.2.2.4. 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"
...
```

10.18.2.3. 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

10.18.3. 인증서 교체 구성

기존 인증서를 교체하도록 인증서 교체 매개 변수를 구성합니다.

10.18.3.1. 인증서 교체 구성

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

절차

1. 다음 명령을 실행하여 **HyperConverged CR**을 엽니다.

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

2. 다음 예와 같이 **spec.certConfig** 필드를 편집합니다. 시스템 과부하를 방지하려면 모든 값이 **10**분 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.renewBefore**의 값은 **ca.duration** 값보다 작거나 같아야 합니다.

2. **server.duration**의 값은 **ca.duration** 값보다 작거나 같아야 합니다.

3. **server.renewBefore**의 값은 **server.duration** 값보다 작거나 같아야 합니다.

3. **YAML** 파일을 클러스터에 적용합니다.

10.18.3.2. 인증서 교체 매개변수 문제 해결

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

- **ca.renewBefore**의 값은 **ca.duration** 값보다 작거나 같아야 합니다.
- **server.duration**의 값은 **ca.duration** 값보다 작거나 같아야 합니다.
- **server.renewBefore**의 값은 **server.duration** 값보다 작거나 같아야 합니다.

기본값이 이러한 조건과 충돌하면 오류가 발생합니다.

다음 예제에서 **server.duration** 값을 제거하는 경우 기본값 **24h0m0s**는 **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.

10.18.4. 가상 머신에 UEFI 모드 사용

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

10.18.4.1. 가상 머신의 UEFI 모드 정보

레거시 **BIOS**와 같은 **UEFI(Unified Extensible Firmware Interface)**는 컴퓨터가 시작될 때 하드웨어 구성 요소 및 운영 체제 이미지 파일을 초기화합니다. **UEFI**는 **BIOS**보다 최신 기능 및 사용자 정의 옵션을 지원하므로 부팅 시간이 단축됩니다.

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

10.18.4.2. 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 설정합니다.

   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 Virtualization**에서는 **UEFI** 모드에서 **Secure Boot**를 사용하려면**SMM(**시스템 관리 모드**)**을 활성화해야 합니다.

2

OpenShift Virtualization에서는 **UEFI** 모드를 사용할 때 **Secure Boot**가 있거나 없는 **VM**을 지원합니다. **Secure Boot**가 활성화된 경우 **UEFI** 모드가 필요합니다. 그러나 **Secure Boot**를 사용하지 않고 **UEFI** 모드를 활성화할 수 있습니다.

2. 다음 명령을 실행하여 클러스터에 매니페스트를 적용합니다.

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

10.18.5. 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.

10.18.5.1. 사전 요구 사항

- **Linux** 브리지가 연결되어 있어야 합니다.
- **PXE** 서버는 브리지와 동일한 **VLAN**에 연결되어 있어야 합니다.

10.18.5.2. 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 **VLAN**에 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. **VNC**를 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 Platform**에서 **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
```

10.18.5.3. 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 Virtualization**에서는 **CNI** 플러그인을 사용하여 기본 **Kubernetes** 네트워킹 기능을 기반으로 빌드합니다.

Multus

Pod 또는 가상 머신에서 필요한 인터페이스를 사용할 수 있도록 여러 **CNI**가 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 **CRD**입니다.

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.

10.18.6. 가상 머신에서 대규모 페이지 사용

대규모 페이지를 클러스터의 가상 머신 백업 메모리로 사용할 수 있습니다.

10.18.6.1. 사전 요구 사항

- 노드에 사전 할당된 대규모 페이지가 구성되어 있어야 합니다.

10.18.6.2. 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.

10.18.6.3. 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 1Gi인 500Mi의 메모리는 요청할 수 없습니다.

참고

호스트와 게스트 **OS**의 메모리 레이아웃은 관련이 없습니다. 가상 머신 매니페스트에서 요청된 대규모 페이지가 **QEMU**에 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
```

10.18.7. 가상 머신 전용 리소스 사용

성능 향상을 위해 **CPU**와 같은 노드 리소스를 가상 머신에 전용으로 지정할 수 있습니다.

10.18.7.1. 전용 리소스 정보

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

10.18.7.2. 사전 요구 사항

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

10.18.7.3. 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** 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 를 클릭합니다.
2. 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.
3. **Scheduling** 탭에서 전용 리소스 옆에 있는 연필 아이콘을 클릭합니다.
4. 전용 리소스(보장된 정책)를 사용하여 이 워크로드 예약을 선택합니다.

-

5. 저장을 클릭합니다.

10.18.8. 가상 머신 예약

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

10.18.8.1. 정책 특성

VM(가상 머신)을 노드에 예약할 때 호환성을 위해 일치하는 정책 특성과 **CPU** 기능을 지정하면 **VM**을 예약할 수 있습니다. **VM**에 지정되는 정책 특성에 따라 **VM**이 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.

10.18.8.2. 정책 특성 및 CPU 기능 설정

각 **VM**(가상 머신)에 대한 정책 특성 및 **CPU** 기능을 설정하면 정책 및 기능에 따라 노드에 **VM**을 예약할 수 있습니다. 설정한 **CPU** 기능은 호스트 **CPU**의 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.

10.18.8.3. 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.

10.18.8.4. 호스트 모델을 사용하여 가상 머신 예약

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

절차

- VM 구성 파일의 domain 사양을 편집합니다. 다음 예제에서는 가상 머신에 지정된 host-model을 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.

10.18.9. 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.

10.18.9.1. 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 CR**에서 **PCI** 장치 정보를 삭제합니다.

10.18.9.1.1. 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.
- **BIOS**의 **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 CPU**에 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)

name은 머신 구성 및 그 용도 중 이 커널 인수의 순위**(100)**를 나타냅니다**. AMD CPU**가 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
```

10.18.9.1.2. 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 Operator**는 **PCI** 장치가 있는 노드에 **/etc/modprobe.d/vfio.conf**를 생성하고 **PCI** 장치를 **VFIO** 드라이버에 바인딩합니다**.**

사전 요구 사항

- **CPU**에 **IOMMU**를 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 **"Butane** 을 사용하여 머신 구성 생성**"**을 참조하십시오**.**

예

```
variant: openshift
version: 4.12.0
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)

3 작업자 노드에서 **vfio-pci** 커널 모듈을 로드하는 파일입니다.

3. **Butane**을 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
```

10.18.9.1.3. 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 CR**을 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 Virtualization**에서는 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 CR**에서 허용된 호스트 장치 목록에 추가된 두 개의 **PCI** 호스트 장치를 보여줍니다. 이러한 장치는 **OpenShift Virtualization**에서 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
```

10.18.9.1.4. 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 CR**을 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 **0**개 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
```

10.18.9.2. 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.

10.18.9.2.1. 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)
```

10.18.9.3. 추가 리소스

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

10.18.10. vGPU passthrough 구성

가상 머신은 **vGPU(**가상 **GPU)** 하드웨어에 액세스할 수 있습니다**.** 가상 머신에 **vGPU**를 할당하면 다음을 수행할 수 있습니다**.**

- 기본 하드웨어의 **GPU** 중 일부에 액세스하여 가상 머신에서 고성능 이점을 실현할 수 있습니다**.**
- 리소스 집약적 **I/O** 작업을 간소화합니다**.**

중요

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

10.18.10.1. 가상 머신에 **vGPU** 패스스루 장치 할당

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

사전 요구 사항

- 가상 머신을 중지해야 합니다**.**

절차

1. **OpenShift Container Platform** 웹 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 를 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 생성된 가상 머신에 하드웨어 장치를 추가할 수 있습니다. **Windows 10** 또는 **RHEL 7**과 같은 특정 운영 체제의 사전 제공 부팅 소스 템플릿에 장치를 추가할 수 없습니다.

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

10.18.10.2. 추가 리소스

- 가상 머신 생성
- 가상 머신 템플릿 생성

10.18.11. 중재된 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.

10.18.11.1. NVIDIA GPU Operator 사용 정보

NVIDIA GPU Operator는 **OpenShift Container Platform** 클러스터에서 **NVIDIA GPU** 리소스를 관리하고 부트스트랩 **GPU** 노드와 관련된 작업을 자동화합니다. **GPU**는 클러스터의 특수 리소스이므로 애플리케이션 워크로드를 **GPU**에 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 **NVIDIA**에서만 지원됩니다. **NVIDIA**에서 지원을 얻는 방법에 대한 자세한 내용은 **NVIDIA**에서 지원을 참조하십시오.

OpenShift Container Platform OpenShift Virtualization에서 **GPU**를 활성화하는 방법에는 여기에 설명된 **OpenShift Container Platform** 네이티브 방법과 **NVIDIA GPU Operator**를 사용하는 두 가지 방법이 있습니다.

NVIDIA GPU Operator는 **OpenShift Container Platform OpenShift Virtualization**이 **OpenShift Container Platform**에서 실행되는 가상화된 워크로드에 **GPU**를 노출할 수 있는 **Kubernetes Operator** 입니다. 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 **VM**을 실행하기 위한 작업자 노드를 프로비저닝하는 방법에 대한 자세한 내용은 **OpenShift Virtualization**을 사용하는 **NVIDIA GPU Operator**를 참조하십시오.

10.18.11.2. 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 제공하면 **OpenShift Virtualization**에서 **vGPUs** 및 기타 중재 장치를 자동으로 생성할 수 있습니다. 이 자동화는 특히 대규모 클러스터에 유용합니다.

참고

기능 및 지원 세부 정보는 하드웨어 벤더의 설명서를 참조하십시오.

중재된 장치

하나 이상의 가상 장치로 분할되는 물리적 장치입니다. **vGPU**는 중재된 장치(**mdev**) 유형입니다. 물리 **GPU**의 성능은 가상 장치 간에 나뉩니다. 하나 이상의 **VM**(가상 머신)에 미디어화된 장치를 할당할 수 있지만 게스트 수는 **GPU**와 호환되어야 합니다. 일부 **GPU**는 여러 게스트를 지원하지 않습니다.

10.18.11.2.1. 사전 요구 사항

- 하드웨어 벤더가 드라이버를 제공하는 경우 중재 장치를 생성하려는 노드에 설치합니다.
  - **NVIDIA** 카드를 사용하는 경우 **NVIDIA GRID** 드라이버를 설치했습니다.

10.18.11.2.2. 구성 개요

중재된 장치를 구성할 때 관리자는 다음 작업을 완료해야 합니다.

- 중재된 장치를 만듭니다.
- 중재된 장치를 클러스터에 노출합니다.

HyperConverged CR에는 두 작업을 모두 수행하는 **API**가 포함되어 있습니다.

중재된 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 그룹의 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.

중재된 장치를 클러스터로 노출

```
...
 permittedHostDevices:
  mediatedDevices:
  - mdevNameSelector: GRID T4-2Q (1)
    resourceName: nvidia.com/GRID_T4-2Q (2)
...
```

1

호스트의 이 값에 매핑되는 중재 장치를 노출합니다.

참고

/sys/bus/pci/devices/<slot>:<bus>:<domain>.<function>/mdev_supported_types/<type>/name 의 내용을 보고 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"))'
```

10.18.11.2.3. 노드에 vGPU를 할당하는 방법

각 물리적 장치에 대해 **OpenShift Virtualization**은 다음 값을 구성합니다.

- 단일 **mdev** 유형.
- 선택한 **mdev** 유형의 최대 인스턴스 수입니다.

클러스터 아키텍처는 장치를 생성하고 노드에 할당하는 방법에 영향을 미칩니다.

노드당 여러 개의 카드가 있는 대규모 클러스터

유사한 **vGPU** 유형을 지원할 수 있는 여러 카드가 있는 노드에서 관련 장치 유형이 라운드 로빈 방식으로 생성됩니다. 예를 들면 다음과 같습니다.

```
...
mediatedDevicesConfiguration:
  mediatedDevicesTypes:
  - nvidia-222
  - nvidia-228
  - nvidia-105
  - nvidia-108
...
```

이 시나리오에서는 각 노드에 다음 **vGPU** 유형을 지원하는 두 개의 카드가 있습니다.

```
nvidia-105
...
nvidia-108
nvidia-217
nvidia-299
...
```

각 노드에서 **OpenShift Virtualization**은 다음과 같은 **vGPU**를 생성합니다.

- 첫 번째 카드에 **nvidia-105** 유형의 **16 vGPU.**
- 두 번째 카드에서 **nvidia-108** 유형의 **vGPU.**

한 노드에는 하나 이상의 요청된 **vGPU** 유형을 지원하는 단일 카드가 있습니다.

OpenShift Virtualization은 **mediatedDevicesTypes** 목록에서 먼저 제공되는 지원되는 유형을 사용합니다.

예를 들어 노드 카드의 카드는 **nvidia-223** 및 **nvidia-224** 를 지원합니다. 다음 **mediatedDevicesTypes** 목록이 구성됩니다.

```
...
mediatedDevicesConfiguration:
  mediatedDevicesTypes:
  - nvidia-22
  - nvidia-223
  - nvidia-224
...
```

이 예에서 **OpenShift Virtualization**은 **nvidia-223** 유형을 사용합니다.

10.18.11.2.4. 미디어된 장치 변경 및 제거 정보

클러스터의 중재 장치 구성은 다음을 통해 **OpenShift Virtualization**으로 업데이트할 수 있습니다.

- **HyperConverged CR**을 편집하고 **mediatedDevicesTypes** 스탠자의 내용을 변경합니다.
- **node MediatedDeviceTypes** 노드 선택기와 일치하는 노드 레이블 변경
- **HyperConverged CR**의 **spec.mediatedDevicesConfiguration** 및 **spec.permittedHostDevices** 스탠자에서 장치 정보를 제거합니다.

참고

spec.permittedHostDevices 스탠자에서도 **spec.mediatedDevicesConfiguration** 스탠자에서 장치 정보를 제거하지 않으면 동일한 노드에 새 중재 장치 유형을 생성할 수 없습니다. 중재된 장치를 올바르게 제거하려면 두 스탠자에서 장치 정보를 제거합니다.

이러한 작업은 특정 변경 사항에 따라 **OpenShift Virtualization**에서 미디어를 재구성하거나 클러스터 노드에서 해당 장치를 제거합니다.

10.18.11.2.5. 중재된 장치를 위한 호스트 준비

중재 장치를 구성하려면 먼저 **IOMMU(Input-Output Memory Management Unit)** 드라이버를 활성화해야 합니다.

10.18.11.2.5.1. 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.
- **BIOS**의 **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 CPU**에 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 CPU**가 있는 경우 커널 인수를 **amd_iommu=on**으로 지정합니다.

(3) **Intel CPU**의 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
```

10.18.11.2.6. 미디어된 장치 추가 및 제거

중재된 장치를 추가하거나 제거할 수 있습니다.

10.18.11.2.6.1. 미디어된 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 CR**을 편집합니다.

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

2. 중재된 장치 정보를 **HyperConverged CR** 사양에 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>
```

10.18.11.2.6.2. CLI를 사용하여 클러스터에서 중재된 장치 제거

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

절차

1. 다음 명령을 실행하여 기본 편집기에서 **HyperConverged CR**을 편집합니다.

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

2. **HyperConverged CR**의 **spec.mediatedDevicesConfiguration** 및 **spec.permittedHostDevices** 스탠자에서 장치 정보를 제거합니다. 두 항목을 모두 제거하면 나중에 동일한 노드에 새로 중재된 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** 필드 및 해당 **resourceName** 필드를 삭제합니다.

3. 변경 사항을 저장하고 편집기를 종료합니다.

10.18.11.3. 중재 장치 사용

vGPU는 중재된 장치 유형입니다. 물리적 **GPU**의 성능은 가상 장치로 나뉩니다. 중재 장치를 하나 이상의 가상 머신에 할당할 수 있습니다.

10.18.11.3.1. 가상 머신에 중재 장치 할당

가상 **GPU(**가상 **GPU)**와 같은 중재 장치를 가상 머신에 할당합니다.

사전 요구 사항

- 중재된 장치는 **HyperConverged** 사용자 정의 리소스에서 구성됩니다.

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 연관된 리소스 이름입니다.

2. **VM**에서 장치를 식별하는 이름입니다.

검증

- 가상 머신에서 장치를 사용할 수 있는지 확인하려면 **VirtualMachine** 매니페스트의 **deviceName** 값을 사용하여 **< device_name >**을 대체하는 다음 명령을 실행합니다.

```
$ lspci -nnk | grep <device_name>
```

10.18.11.4. 추가 리소스

- **BIOS**에서 **Intel VT-X** 및 **AMD-V** 가상화 하드웨어 확장 활성화

10.18.12. 워치독 구성

워치독 장치에 대해 **VM(**가상 머신**)**을 구성하고**,** 워치독을 설치한 후 워치독 서비스를 시작하여 워치독을 노출합니다**.**

10.18.12.1. 사전 요구 사항

- 가상 머신에는 **i6300esb** 워치독 장치에 대한 커널 지원이 있어야 합니다**. RHEL(Red Hat Enterprise Linux)** 이미지는 **i6300esb**를 지원합니다**.**

10.18.12.2. 워치독 장치 정의

운영 체제(OS)가 더 이상 응답하지 않을 때 워치독이 진행되는 방식을 정의합니다.

표 10.4. 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 VM**에서 **i6300esb** 워치독 장치를 구성하고 장치를 **/dev/watchdog**로 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 **VM**이 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
```

10.18.12.3. 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
```

10.18.12.4. 추가 리소스

- 상태 점검을 사용하여 애플리케이션 상태 모니터링

10.18.13. 사전 정의된 부팅 소스 자동 가져오기 및 업데이트

시스템 정의이고 OpenShift Virtualization 또는 사용자가 생성한 사용자 정의 에 포함된 부팅 소스를 사용할 수 있습니다. 시스템 정의 부팅 소스 가져오기 및 업데이트는 제품 기능 게이트에 의해 제어됩니다. 기능 게이트를 사용하여 업데이트를 사용, 비활성화 또는 다시 활성화할 수 있습니다. 사용자 정의 부팅 소스는 제품 기능 게이트에 의해 제어되지 않으며 자동 가져오기 및 업데이트를 선택하거나 옵트아웃하기 위해 개별적으로 관리되어야 합니다.

중요

버전 **4.10**부터 **OpenShift Virtualization**은 수동으로 비활성화하거나 기본 스토리지 클래스를 설정하지 않는 한 부팅 소스를 자동으로 가져오고 업데이트합니다.

버전 **4.10**으로 업그레이드하는 경우, 버전 **4.9** 또는 이전 버전에서 부팅 소스에 대한 자동 가져오기 및 업데이트를 수동으로 활성화해야 합니다.

10.18.13.1. 자동 부팅 소스 업데이트 활성화

OpenShift Virtualization 4.9 또는 이전 버전에서 부팅 소스가 있는 경우 이러한 부팅 소스에 대한 자동 업데이트를 수동으로 활성화해야 합니다. **OpenShift Virtualization 4.10** 이상의 모든 부팅 소스는 기본적으로 자동으로 업데이트됩니다.

자동 부팅 소스 가져오기 및 업데이트를 활성화하려면 자동으로 업데이트하려는 각 부팅 소스에 대해 **cdi.kubevirt.io/dataImportCron** 필드를 **true** 로 설정합니다.

절차

- 부팅 소스에 대한 자동 업데이트를 활성화하려면 다음 명령을 사용하여 데이터 소스에 **dataImportCron** 레이블을 적용합니다.

```
$ oc label --overwrite DataSource rhel8 -n openshift-virtualization-os-images
cdi.kubevirt.io/dataImportCron=true (1)
```

(1) **true** 를 지정하면 **rhel8** 부팅 소스의 자동 업데이트가 실행됩니다.

10.18.13.2. 자동 부팅 소스 업데이트 비활성화

자동 부팅 소스 가져오기 및 업데이트를 비활성화하면 연결이 끊긴 환경의 로그 수를 줄이거나 리소스 사용량을 줄이는 데 유용할 수 있습니다.

자동 부팅 소스 가져오기 및 업데이트를 비활성화하려면 **HyperConverged CR**(사용자 정의 리소스)의 **spec.featureGates.enableCommonBootImageImport** 필드를 **false** 로 설정합니다.

참고

사용자 정의 부팅 소스는 이 설정의 영향을 받지 않습니다.

절차

- 다음 명령을 사용하여 자동 부팅 소스 업데이트를 비활성화합니다.

```
$ oc patch hco kubevirt-hyperconverged -n openshift-cnv \
 --type json -p '[{"op": "replace", "path":
"/spec/featureGates/enableCommonBootImageImport", \
 "value": false}]'
```

10.18.13.3. 자동 부팅 소스 업데이트 다시 활성화

이전에 자동 부팅 소스 업데이트를 비활성화한 경우 해당 기능을 수동으로 다시 활성화해야 합니다. **HyperConverged CR**(사용자 정의 리소스)의 **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}]'
```

10.18.13.4. 사용자 정의 부팅 소스 업데이트를 위한 스토리지 클래스 구성

사용자 정의 부팅 소스에 대해 자동 가져오기 및 업데이트를 허용하는 스토리지 클래스를 구성할 수 있습니다.

절차

1. **HyperConverged CR(**사용자 정의 리소스**)**을 편집하여 새 **storageClassName** 을 정의합니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
spec:
  dataImportCronTemplates:
  - metadata:
      name: rhel8-image-cron
    spec:
      template:
        spec:
          storageClassName: <appropriate_class_name>
...
```

2. 다음 명령을 실행하여 새 기본 스토리지 클래스를 설정합니다.

```
$ oc patch storageclass <current_default_storage_class> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

```
$ oc patch storageclass <appropriate_storage_class> -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

10.18.13.5. 사용자 정의 부팅 소스에 대한 자동 업데이트 활성화

OpenShift Virtualization은 기본적으로 시스템 정의 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 수동으로 활성화해야 합니다.

절차

1. 다음 명령을 사용하여 편집할 **HyperConverged CR**을 엽니다.

```
$ oc edit -n openshift-cnv HyperConverged
```

2. **HyperConverged CR**을 편집하여 **dataImportCronTemplates** 섹션에 적절한 템플릿 및 부팅 소스를 추가합니다. 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) **volumeBindingMode** 가 있는 스토리지 클래스에는 **WaitForFirstConsumer** 가 설정되어 있는 스토리지 클래스에 필요합니다.

(2)

cron 형식으로 지정된 작업의 스케줄입니다.

3

을 사용하여 레지스트리 소스에서 데이터 볼륨을 생성합니다. 노드 **docker** 캐시를 기반으로 하는 기본 **pod pullMethod** 및 **node pullMethod** 를 사용합니다. 노드 **Docker** 캐시는 **Container.Image** 를 통해 레지스트리 이미지를 사용할 수 있지만 **CDI** 가져오기자는 액세스할 수 있는 권한이 없는 경우에 유용합니다.

4

사용자 지정 이미지를 사용 가능한 부팅 소스로 감지하려면 이미지의 **managedDataSource** 의 이름이 **VM** 템플릿 **YAML** 파일의 **spec.dataVolumeTemplates.spec.sourceRef.name** 아래에 있는 템플릿의 **DataSource** 와 일치해야 합니다.

5

cron 작업이 삭제될 때 **All** 을 사용하여 데이터 볼륨 및 데이터 소스를 유지합니다. **None** 을 사용하여 **cron** 작업이 삭제될 때 데이터 볼륨 및 데이터 소스를 삭제합니다.

10.18.13.6. 시스템 정의 또는 사용자 정의 부팅 소스에 대한 자동 업데이트 비활성화

사용자 정의 부팅 소스와 시스템 정의 부팅 소스에 대한 자동 가져오기 및 업데이트를 비활성화할 수 있습니다.

시스템 정의 부팅 소스는 **HyperConverged CR**(사용자 정의 리소스)의 **spec.dataImportCronTemplates** 에 기본적으로 나열되지 않으므로 부팅 소스를 추가하고 자동 가져오기 및 업데이트를 비활성화해야 합니다.

절차

- 사용자 정의 부팅 소스에 대한 자동 가져오기 및 업데이트를 비활성화하려면 사용자 정의 리소스 목록의 **spec.dataImportCronTemplates** 필드에서 부팅 소스를 제거합니다.
- 시스템 정의 부팅 소스에 대한 자동 가져오기 및 업데이트를 비활성화하려면 다음을 수행합니다.
  - **HyperConverged CR**을 편집하고 **spec.dataImportCronTemplates** 에 부팅 소스를 추가합니다.

- **dataimportcrontemplate.kubevirt.io/enable** 주석을 **false** 로 설정하여 자동 가져오기 및 업데이트를 비활성화합니다. 예를 들면 다음과 같습니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
spec:
  dataImportCronTemplates:
  - metadata:
      annotations:
        dataimportcrontemplate.kubevirt.io/enable: false
      name: rhel8-image-cron
...
```

10.18.13.7. 부팅 소스 상태 확인

부팅 소스가 시스템 정의인지 사용자 정의인지 확인할 수 있습니다.

HyperConverged CR의 **status.dataImportChronTemplates** 필드에 나열된 각 부팅 소스의 **status** 섹션은 부팅 소스 유형을 나타냅니다. 예를 들어, **commonTemplate: true** 는 시스템 정의 **(CommonTemplate)** 부팅 소스 및 **status: {}** 를 나타냅니다. **{}**는 사용자 정의 부팅 소스를 나타냅니다.

절차

1. **oc get** 명령을 사용하여 **HyperConverged CR**의 **dataImportChronTemplates** 를 나열합니다.

2. 부팅 소스의 상태를 확인합니다.

출력 예

```
...
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
...
spec:
  ...
status: (1)
  ...
  dataImportCronTemplates: (2)
  - metadata:
```

```
    annotations:
      cdi.kubevirt.io/storage.bind.immediate.requested: "true"
    name: centos-7-image-cron
  spec:
    garbageCollect: Outdated
    managedDataSource: centos7
    schedule: 55 8/12 * * *
    template:
      metadata: {}
      spec:
        source:
          registry:
            url: docker://quay.io/containerdisks/centos:7-2009
        storage:
          resources:
            requests:
              storage: 30Gi
      status: {}
  status:
    commonTemplate: true 3
  ...
- metadata:
    annotations:
      cdi.kubevirt.io/storage.bind.immediate.requested: "true"
    name: user-defined-dic
  spec:
    garbageCollect: Outdated
    managedDataSource: user-defined-centos-stream8
    schedule: 55 8/12 * * *
    template:
      metadata: {}
      spec:
        source:
          registry:
            pullMethod: node
            url: docker://quay.io/containerdisks/centos-stream:8
        storage:
          resources:
            requests:
              storage: 30Gi
      status: {}
  status: {} 4
...
```

1 HyperConverged CR의 status 필드입니다.

2

모든 정의된 부팅 소스를 나열하는 **dataImportCronTemplates** 필드

3

시스템 정의 부팅 소스를 나타냅니다.

4

사용자 정의 부팅 소스를 나타냅니다.

10.18.14. 가상 머신에서 Descheduler 제거 활성화

Descheduler를 사용하여 **Pod**를 제거하여 더 적절한 노드에 **Pod**를 다시 예약할 수 있습니다. **Pod**가 가상 머신인 경우 **Pod** 제거를 통해 가상 머신이 다른 노드로 실시간 마이그레이션됩니다.

중요

가상 머신의 **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.

10.18.14.1. Descheduler 프로필

기술 프리뷰 **DevPreviewLongLifecycle** 프로필을 사용하여 가상 시스템에서 **Descheduler**를 활성화합니다. 현재 **OpenShift Virtualization**에서 사용할 수 있는 유일한 **Descheduler** 프로필입니다. 적절한 스케줄링을 위해 예상되는 부하에 대한 **CPU** 및 메모리 요청이 있는 **VM**을 생성합니다.

DevPreviewLongLifecycle

이 프로필은 노드 간 리소스 사용량의 균형을 조정하고 다음 전략을 활성화합니다.

- **RemovePodsHavingTooManyRestarts:** 컨테이너가 너무 여러 번 재시작된 **Pod**와 모든 컨테이너에 대한 재시작 횟수(**Init Containers** 포함)가 **100**개 이상인 **Pod**를 제거합니다. **VM** 게스트 운영 체제를 다시 시작해도 이 수가 늘어나지 않습니다.

- **LowNodeUtilization:** 활용도가 낮은 노드가 있는 경우 활용도가 높은 노드에서 **Pod**를 제거합니다. 제거된 **Pod**의 대상 노드는 스케줄러에 의해 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.

10.18.14.2. Descheduler 설치

Descheduler는 기본적으로 사용할 수 없습니다. **Descheduler**를 활성화하려면 **OperatorHub**에서 **Kube Descheduler Operator**를 설치하고 **Descheduler** 프로필을 한 개 이상 활성화해야 합니다.

기본적으로 **Descheduler**는 예측 모드에서 실행되므로 **Pod** 제거만 시뮬레이션합니다. **Descheduler**가 **Pod** 제거를 수행할 수 있도록 모드를 자동으로 변경해야 합니다.

중요

클러스터에서 호스트된 컨트롤 플레인을 활성화한 경우 사용자 정의 우선순위 임계값을 설정하여 호스트된 컨트롤 플레인 네임스페이스의 **Pod**가 제거될 가능성을 낮춥니다. 호스팅된 컨트롤 플레인 우선 순위 클래스 클래스의 가장 낮은 우선 순위 값(**100000000**)이 있으므로 우선순위 임계값 클래스 이름을 **hypershift-control-plane** 으로 설정합니다.

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 Operator**에 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 다음 생성을 클릭합니다.

3. **Kube Descheduler Operator**를 설치합니다.

   a. **Operators** → **OperatorHub**로 이동합니다.

   b. 필터 박스에 **Kube Descheduler Operator**를 입력합니다.

   c. **Kube Descheduler Operator**를 선택하고 설치를 클릭합니다.

   d. **Operator** 설치 페이지에서 클러스터의 특정 네임스페이스를 선택합니다. 드롭다운 메뉴에서 **openshift-kube-descheduler-operator**를 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 Operator**를 클릭합니다.

   b. **Kube Descheduler** 탭을 선택하고 **KubeDescheduler** 생성을 클릭합니다.

c. 필요에 따라 설정을 편집합니다.

i. 제거를 시뮬레이션하는 대신 **Pod**를 제거하려면 **Mode** 필드를 자동으로 변경합니다.

ii. **Profiles** 섹션을 확장하고 **DevPreviewLongLifecycle** 를 선택합니다. **AffinityAndTaints** 프로필은 기본적으로 활성화되어 있습니다.

중요

OpenShift Virtualization에서 현재 사용할 수 있는 유일한 프로필은 **DevPreviewLongLifecycle** 입니다.

나중에 **OpenShift CLI(oc)**를 사용하여 **Descheduler**에 대한 프로필 및 설정을 구성할 수도 있습니다.

10.18.14.3. VM(가상 머신)에서 Descheduler 제거 활성화

Descheduler가 설치되면 **VirtualMachine CR(**사용자 정의 리소스**)**에 주석을 추가하여 **VM**에서 **Descheduler** 제거를 활성화할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 웹 콘솔 또는 **OpenShift CLI(oc)**에 **Descheduler**를 설치합니다.
- **VM**이 실행되고 있지 않은지 확인합니다.

절차

1. **VM**을 시작하기 전에 **VirtualMachine CR**에 **Descheduler.alpha.kubernetes.io/evict** 주석을 추가합니다.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  template:
```

```
metadata:
  annotations:
    descheduler.alpha.kubernetes.io/evict: "true"
```

2. 설치하는 동안 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
  mode: Predictive (1)
```

(1) 기본적으로 **Descheduler**는 **Pod**를 제거하지 않습니다. **Pod**를 제거하려면 **mode** 를 **Automatic** 으로 설정합니다.

이제 **VM**에서 **Descheduler**가 활성화됩니다.

10.18.14.4. 추가 리소스

- **Descheduler**를 사용하여 **Pod** 제거

10.19. 가상 머신 가져오기

10.19.1. 데이터 볼륨 가져오기에 필요한 TLS 인증서

10.19.1.1. 데이터 볼륨 가져오기 인증을 위한 TLS 인증서 추가

레지스트리 또는 **HTTPS**에서 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 map**을 생성합니다.

```
$ oc create configmap <configmap-name> --from-file=</path/to/file/ca.pem>
```

10.19.1.2. 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-----
```

10.19.2. 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 **PVC**로 가져오면 **PVC**에 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.

10.19.2.1. 사전 요구 사항

- 끝점에 **TLS** 인증서가 필요한 경우 인증서가 데이터 볼륨과 동일한 네임스페이스의 구성 맵에 포함되고 데이터 볼륨 구성에서 참조되어야 합니다.
- 컨테이너 디스크를 가져오려면 다음을 수행합니다.
  - 가져오기 전에 가상 머신 이미지에서 컨테이너 디스크를 준비하여 컨테이너 레지스트리에 저장해야 할 수 있습니다.
  - 컨테이너 레지스트리에 **TLS**가 없는 경우 컨테이너 디스크 를 가져오기 전에 **HyperConverged** 사용자 정의 리소스의 **insecureRegistries** 필드에 레지스트리를 추가해야 합니다.
- 이 작업을 완료하려면 스토리지 클래스를 정의하거나 **CDI** 스크래치 공간을 준비해야 할 수 있습니다.

10.19.2.2. 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*

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

참고

이제 **CDI**에서 **OpenShift Container Platform** 클러스터 전체 프록시 구성 을 사용합니다.

10.19.2.3. 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. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM(**가상 머신**)** 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다.

참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다. **VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다.

10.19.2.4. 데이터 볼륨을 사용하여 가상 머신 이미지를 스토리지로 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 파일에서 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 빌드되고 컨테이너 레지스트리에 저장된 가상 머신 이미지가 있어야 합니다.
- 가상 머신이 자체 서명된 인증서 또는 시스템 **CA** 번들에서 서명되지 않은 인증서를 사용하는 서버와 통신해야 하는 경우 데이터 볼륨과 동일한 네임스페이스에 구성 맵을 생성해야 합니다.

절차

1. 데이터 소스에 인증이 필요한 경우 데이터 소스 인증 정보를 지정하여 **Secret** 매니페스트를 생성하고 이를 **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)

2. **Base64**로 인코딩된 키 **ID** 또는 사용자 이름을 지정합니다.

3. **Base64**로 인코딩된 보안 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 편집하고 **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** 엔드포인트에 **http** 를 지정합니다. 레지스트리에서 가져온 컨테이너 디스크 이미지의 레지스트리를 지정합니다.

4. 가져올 가상 머신 이미지의 **URL** 또는 레지스트리 끝점을 지정합니다. 이 예제에서는 **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** 컨트롤러에서 올바른 주석을 사용하여 기본 **PVC**를 생성하고 가져오기 프로세스가 시작됩니다. 가져오기가 완료되면 데이터 볼륨 상태가 **Succeeded** 로 변경됩니다. 가상 머신을 시작할 수 있습니다.

데이터 볼륨 프로비저닝은 백그라운드에서 이루어지므로 프로세스를 모니터링할 필요가 없습니다.

검증

1. 가져오기 **Pod**는 지정된 **URL**에서 가상 머신 이미지 또는 컨테이너 디스크를 다운로드하여 프로비저닝된 **PV**에 저장합니다. 다음 명령을 실행하여 가져오기 **Pod**의 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. 직렬 콘솔에 액세스하여 프로비저닝이 완료되고 가상 머신이 시작되었는지 확인합니다.

```
$ virtctl console vm-fedora-datavolume
```

10.19.2.5. 추가 리소스

- 데이터 볼륨 작업에 대한 쓰기 성능을 개선하도록 사전 할당 모드를 구성합니다.

10.19.3. 데이터 볼륨을 사용하여 가상 머신 이미지를 블록 스토리지로 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 **PVC**로 가져오면 **PVC**에 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.

10.19.3.1. 사전 요구 사항

- **CDI** 지원 작업 매트릭스에 따라 스크래치 공간이 필요한 경우 먼저 이 작업을 완료하려면 스토리지 클래스를 정의하거나 **CDI** 스크래치 공간을 준비해야 합니다.

10.19.3.2. 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. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM(**가상 머신**)** 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다.

참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다. **VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다.

10.19.3.3. 블록 영구 볼륨 정보

PV(블록 영구 볼륨**)**는 원시 블록 장치에서 지원하는 **PV**입니다. 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.

10.19.3.4. 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 **PV**를 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 **PV**임을 나타냅니다.

3. 선택 사항: **PV**의 스토리지 클래스를 설정합니다. 생략하면 클러스터 기본값이 사용됩니다.

4. 블록 장치가 마운트된 노드입니다.

5. 블록 **PV**를 생성합니다.

```
# oc create -f <local-block-pv10.yaml> (1)
```

1. 이전 단계에서 생성한 영구 볼륨의 파일 이름입니다.

10.19.3.5. 데이터 볼륨을 사용하여 가상 머신 이미지를 블록 스토리지로 가져오기

데이터 볼륨을 사용하여 가상 머신 이미지를 블록 스토리지로 가져올 수 있습니다. 가상 머신을 생성하기 전에 **VirtualMachine** 매니페스트의 데이터 볼륨을 참조합니다.

사전 요구 사항

- **RAW, ISO** 또는 **QCOW2** 형식의 가상 머신 디스크 이미지(필요한 경우 **xz** 또는 **gz**를 사용하

여 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 인증 정보를 지정하여 **Secret** 매니페스트를 생성하고 이를 **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) **Secret** 의 이름을 지정합니다.

(2) **Base64**로 인코딩된 키 **ID** 또는 사용자 이름을 지정합니다.

(3) **Base64**로 인코딩된 보안 키 또는 암호를 지정합니다.

2. 보안 매니페스트 를 적용합니다.

```
$ oc apply -f endpoint-secret.yaml
```

3. **DataVolume** 매니페스트를 생성하여 가상 머신 이미지의 데이터 소스 및 **storage.volumeMode** 의 경우 **Block** 을 지정합니다.

```
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 URL**을 지정합니다.

4. 데이터 소스에 대한 보안을 생성한 경우 시크릿 이름을 지정합니다.

5. 볼륨 모드 및 액세스 모드는 알려진 스토리지 프로비저너에서 자동으로 탐지됩니다. 그렇지 않으면 **Block** 을 지정합니다.

4. 데이터 볼륨을 생성하여 가상 머신 이미지를 가져옵니다.

```
$ oc create -f import-pv-datavolume.yaml
```

가상 머신을 생성하기 전에 **VirtualMachine** 매니페스트에서 이 데이터 볼륨을 참조할 수 있습니다.

10.19.3.6. 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 **CDI**에서 **OpenShift Container Platform** 클러스터 전체 프록시 구성 을 사용합니다.

10.19.3.7. 추가 리소스

- 데이터 볼륨 작업에 대한 쓰기 성능을 개선하도록 사전 할당 모드를 구성합니다.

10.20. 가상 머신 복제

10.20.1. 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.

10.20.1.1. 사전 요구 사항

- **cluster-admin** 역할의 사용자만 클러스터 역할을 생성할 수 있습니다.

10.20.1.2. 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. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM(**가상 머신**)** 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다.

참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다. **VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다.

10.20.1.3. 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.

10.20.2. 가상 머신 디스크를 새 데이터 볼륨으로 복제

데이터 볼륨 구성 파일에서 소스 **PVC(**영구 볼륨 클레임**)**를 참조하여 가상 머신 디스크의 **PVC**를 새 데이터 볼륨으로 복제할 수 있습니다.

> 주의
>
> **volumeMode: Block**이 있는 **PV(**영구 볼륨**)**에서 **volumeMode: Filesystem**인 **PV**로 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.

10.20.2.1. 사전 요구 사항

- 사용자는 가상 머신 디스크의 **PVC**를 다른 네임스페이스에 복제하려면 추가 권한이 필요합니다.

10.20.2.2. 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. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM(**가상 머신**)** 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다.

참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다. **VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다.

10.20.2.3. 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 **PVC**가 삭제되지 않습니다.

사전 요구 사항

- 사용할 기존 가상 머신 디스크의 **PVC**를 결정합니다. **PVC**와 연결된 가상 머신의 전원을 꺼야 복제할 수 있습니다.
- **OpenShift CLI(oc)**를 설치합니다.

절차

1. 복제하려는 가상 머신 디스크를 검사하여 연결된 **PVC**의 이름과 네임스페이스를 확인합니다.
2. 데이터 볼륨에 대해 새 데이터 볼륨의 이름, 소스 **PVC**의 이름과 네임스페이스, 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)
```

1. 새 데이터 볼륨의 이름입니다.

2. 소스 **PVC**가 존재하는 네임스페이스입니다.

3. 소스 **PVC**의 이름입니다.

4. 새 데이터 볼륨의 크기입니다. 충분한 공간을 할당해야 합니다. 그러지 않으면 복제 작업이 실패합니다. 크기는 소스 **PVC**와 같거나 커야 합니다.

3. 데이터 볼륨을 생성하여 **PVC** 복제를 시작합니다.

```
$ oc create -f <cloner-datavolume>.yaml
```

참고

데이터 볼륨이 있으면 **PVC**가 준비될 때까지 가상 머신이 시작되지 않으므로 **PVC**가 복제되는 동안 새 데이터 볼륨을 참조하는 가상 머신을 생성할 수 있습니다.

10.20.2.4. 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

10.20.3. 데이터 볼륨 템플릿을 사용하여 가상 머신 복제

기존 **VM**의 **PVC(**영구 볼륨 클레임**)**를 복제하여 새 가상 머신을 생성할 수 있습니다**.** 가상 머신 구성 파일에 **dataVolumeTemplate**을 포함하여 원래 **PVC**에서 새 데이터 볼륨을 생성합니다**.**

주의

volumeMode: Block이 있는 **PV(**영구 볼륨**)**에서 **volumeMode: Filesystem**인 **PV**로 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**.**

10.20.3.1. 사전 요구 사항

- 사용자는 가상 머신 디스크의 **PVC**를 다른 네임스페이스에 복제하려면 추가 권한이 필요합니다**.**

10.20.3.2. 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**.** 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM(**가상 머신**)** 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다**.**

참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다**. VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다**.**

10.20.3.3. 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 PVC**가 데이터 볼륨에 복제되어 가

상 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 **PVC**도 삭제됩니다.

사전 요구 사항

- 사용할 기존 가상 머신 디스크의 **PVC**를 결정합니다. **PVC**와 연결된 가상 머신의 전원을 꺼야 복제할 수 있습니다.
- **OpenShift CLI(oc)**를 설치합니다.

절차

1. 복제하려는 가상 머신을 검사하여 연결된 **PVC**의 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
```

10.20.3.4. 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

10.20.4. 가상 머신 디스크를 새 블록 스토리지 데이터 볼륨에 복제

데이터 볼륨 구성 파일의 소스 **PVC(**영구 볼륨 클레임**)**를 참조하여 가상 머신 디스크의 **PVC**를 새 블록 데이터 볼륨에 복제할 수 있습니다**.**

주의

volumeMode: Block이 있는 **PV(**영구 볼륨**)**에서 **volumeMode: Filesystem**인 **PV**로 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**.**

10.20.4.1. 사전 요구 사항

- 사용자는 가상 머신 디스크의 **PVC**를 다른 네임스페이스에 복제하려면 추가 권한이 필요합니다**.**

10.20.4.2. 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. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM(**가상 머신**)** 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다.

참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다. **VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다.

10.20.4.3. 블록 영구 볼륨 정보

PV(블록 영구 볼륨**)**는 원시 블록 장치에서 지원하는 **PV**입니다. 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.

10.20.4.4. 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 **PV**를 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 **PV**임을 나타냅니다.

(3) 선택 사항: **PV**의 스토리지 클래스를 설정합니다. 생략하면 클러스터 기본값이 사용됩니다.

4. 블록 장치가 마운트된 노드입니다.

5. 블록 **PV**를 생성합니다.

```
# oc create -f <local-block-pv10.yaml> 1
```

1. 이전 단계에서 생성한 영구 볼륨의 파일 이름입니다.

10.20.4.5. 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 **PVC**가 삭제되지 않습니다**.**

사전 요구 사항

- 사용할 기존 가상 머신 디스크의 **PVC**를 결정합니다**. PVC**와 연결된 가상 머신의 전원을 꺼야 복제할 수 있습니다**.**
- **OpenShift CLI(oc)**를 설치합니다**.**
- 소스 **PVC**와 크기가 같거나 더 큰 블록 **PV(**영구 볼륨**)**가 한 개 이상 사용 가능합니다**.**

절차

1. 복제하려는 가상 머신 디스크를 검사하여 연결된 **PVC**의 이름과 네임스페이스를 확인합니다**.**

2. 데이터 볼륨에 대해 새 데이터 볼륨의 이름, 소스 **PVC**의 이름과 네임스페이스, 사용 가능한 블록 **PV**를 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 **PVC**가 존재하는 네임스페이스입니다.

(3) 소스 **PVC**의 이름입니다.

(4) 새 데이터 볼륨의 크기입니다. 충분한 공간을 할당해야 합니다. 그러지 않으면 복제 작업이 실패합니다. 크기는 소스 **PVC**와 같거나 커야 합니다.

(5) 대상이 블록 **PV**임을 나타냅니다.

3. 데이터 볼륨을 생성하여 **PVC** 복제를 시작합니다.

```
$ oc create -f <cloner-datavolume>.yaml
```

참고

데이터 볼륨이 있으면 **PVC**가 준비될 때까지 가상 머신이 시작되지 않으므로 **PVC**가 복제되는 동안 새 데이터 볼륨을 참조하는 가상 머신을 생성할 수 있습니다.

10.20.4.6. 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

10.21. 가상 머신 네트워킹

10.21.1. 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.

10.21.1.1. 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 **DHCP**를 사용하여 **IPv4** 주소를 가져오도록 구성해야 합니다. 아래 예제는 **DHCP**를 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: 포트 필드에 지정된 각 포트를 가상 머신에서 노출하려는 포트 를 나열합니다. 포트 값은 **0**에서 **65536** 사이의 숫자여야 합니다. 포트 배열을 사용하지 않으면 유효한 범위의 모든 포트가 들어오는 트래픽에 대해 열려 있습니다. 이 예에서 들어오는 트래픽은 포트 **80** 에서 허용됩니다.

참고

포트 **49152** 및 **49153**은 **libvirt** 플랫폼에서 사용하도록 예약되어 있으며 이러한 포트에 대한 기타 모든 들어오는 트래픽은 삭제됩니다.

2. 가상 머신을 생성합니다.

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

10.21.1.2. 듀얼 스택**(IPv4** 및 **IPv6)**을 사용하여 가상 모드 구성

cloud-init를 사용하여 기본 **Pod** 네트워크에서 **IPv6** 및 **IPv4**를 모두 사용하도록 새 **VM(**가상 머신**)**을 구성할 수 있습니다.

가상 머신 인스턴스 구성의 **Network.pod.vmIPv6NetworkCIDR** 필드에 따라 **VM**의 정적 **IPv6** 주소와 게이트웨이 **IP** 주소가 결정됩니다. 이는 **virt-launcher Pod**에서 **IPv6** 트래픽을 가상 머신으로 라우팅하는데 사용되며 외부적으로 사용되지 않습니다. **Network.pod.vmIPv6NetworkCIDR** 필드는 **CIDR(Classless Inter-Domain Routing)** 표기법에서 **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 Pod**의 **IPv4** 주소와 고유한 **IPv6** 주소로 라우팅됩니다. 그런 다음 **virt-launcher Pod**는 **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 플러그인을 사용해야 합니다.

절차

1. 새 가상 시스템 구성에서 **masquerade**가 있는 인터페이스를 포함하고 **cloud-init**를 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 **80**에서 가상 머신으로 들어오는 트래픽을 허용합니다.

(3) 가상 머신 인스턴스 구성의 **Network.pod.vmIPv6NetworkCIDR** 필드에 의해 결정된 정적 **IPv6** 주소입니다. 기본값은 **fd10:0:2::2/120** 입니다.

(4) 가상 머신 인스턴스 구성의 **Network.pod.vmIPv6NetworkCIDR** 필드에 의해 결정된 게이트웨이 **IP** 주소입니다. 기본값은 **fd10:0:2::1** 입니다.

2. 네임스페이스에서 가상 머신을 생성합니다.

```
$ oc create -f example-vm-ipv6.yaml
```

검증

- **IPv6**가 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}"
```

10.21.2. 가상 머신 노출 서비스 생성

Service 오브젝트를 사용하여 클러스터 내에서 또는 클러스터 외부에 가상 머신을 노출할 수 있습니다.

10.21.2.1. 서비스 정보

Kubernetes ***서비스는*** 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 서비스 유형 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.

참고

온프레미스 클러스터의 경우 **MetalLB Operator**를 배포하여 로드 밸런싱 서비스를 구성할 수 있습니다.

추가 리소스

- **MetalLB Operator** 설치
- **MetalLB**를 사용하도록 서비스 구성

10.21.2.1.1. 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]**

10.21.2.2. 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 **Pod**로 전달됩니다. **special: key** 레이블은 서비스 매니페스트의 **spec.selector** 특성의 레이블과 일치해야 합니다.

2. **VirtualMachine** 매니페스트 파일을 저장하여 변경 사항을 적용합니다.

3. **VM**을 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 않으면 포트 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. **VM**을 시작합니다. **VM**이 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 **VM**에 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 **VM**에 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.

10.21.2.3. 추가 리소스

- **NodePort**를 사용하여 수신 클러스터 트래픽 구성
- 로드 밸런서를 사용하여 수신 클러스터 트래픽 구성

10.21.3. Linux 브리지 네트워크에 가상 머신 연결

기본적으로 **OpenShift Virtualization**은 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.

10.21.3.1. 네트워크 연결 정의를 통해 네트워크에 연결

10.21.3.1.1. Linux 브리지 노드 네트워크 구성 정책 생성

NodeNetworkConfigurationPolicy 매니페스트 **YAML** 파일을 사용하여 **Linux** 브리지를 만듭니다.

사전 요구 사항

- **Kubernetes NMState Operator**를 설치했습니다.

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 **IPv4**를 비활성화합니다.
7. 이 예에서 **STP**를 비활성화합니다.
8. 브리지가 연결된 노드 **NIC**입니다.

10.21.3.2. Linux 브리지 네트워크 연결 정의 생성

> 주의
>
> 가상 머신의 네트워크 연결 정의에서 **IPAM(IP** 주소 관리**)** 구성은 지원되지 않습니다.

10.21.3.2.1. 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 ID**가 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 **Pod**를 종료할 수 있도록 허용하여 **MAC** 스푸핑 공격에 대한 보안을 제공합니다.

8. 생성을 클릭합니다.

   참고

   Linux 브리지 네트워크 연결 정의는 가상 머신을 **VLAN**에 연결하는 가장 효율적인 방법입니다.

10.21.3.2.2. 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 **nftables**를 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": 100, (7)
    "preserveDefaultVlan": false (8)
  }'
```

(1) **NetworkAttachmentDefinition** 개체의 이름입니다.

(2) 선택 사항: 노드 선택의 주석 키-값 쌍입니다. 여기서 **bridge-interface** 는 일부 노드에 구성된 브리지의 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 **CNI**를 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 **Pod**를 종료할 수 있도록 허용하여 **MAC** 스푸핑 공격에 대한 보안을 제공합니다.

7

선택 사항: **VLAN** 태그. 노드 네트워크 구성 정책에 추가 **VLAN** 구성이 필요하지 않습니다.

8

선택 사항: **VM**이 기본 **VLAN**을 통해 브리지에 연결되는지 여부를 나타냅니다. 기본값은 **true**입니다.

참고

Linux 브리지 네트워크 연결 정의는 가상 머신을 **VLAN**에 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>
```

10.21.3.3. Linux 브리지 네트워크에 대한 가상 머신 구성

10.21.3.3.1. 웹 콘솔에서 가상 머신의 NIC를 생성

웹 콘솔에서 추가 **NIC**를 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 → **VirtualMachine** 를 클릭합니다.
2. 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.
3. 네트워크 인터페이스 탭을 클릭하여 가상 머신에 이미 연결된 **NIC**를 확인합니다.
4. 네트워크 인터페이스 추가를 클릭하여 목록에 새 슬롯을 만듭니다.
5. 추가 네트워크의 네트워크 목록에서 네트워크 연결 정의를 선택합니다.
6. 새 **NIC**의 이름, 모델, 유형, **MAC** 주소를 입력합니다.
7. 저장을 클릭하여 **NIC**를 저장하고 가상 머신에 연결합니다.

10.21.3.3.2. 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: 브리지 ● SR-IOV 네트워크: **SR-IOV**
MAC 주소	네트워크 인터페이스 컨트롤러의 MAC 주소입니다. MAC 주소를 지정하지 않으면 주소가 자동으로 할당됩니다.

10.21.3.3.3. 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 **VM**을 생성할 동일한 네임스페이스여야 합니다. 이 경우 **multus** 가 사용됩니다. **Multus**는 **Pod** 또는 가상 머신에서 필요한 인터페이스를 사용할 수 있도록 여러 **CNI**가 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.

10.21.4. 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.

10.21.4.1. 사전 요구 사항

- 호스트의 펌웨어에 글로벌 SR-IOV 및 VT-d 설정이 활성화되어 있어야 합니다.
- SR-IOV Network Operator가 설치되어 있어야 합니다.

10.21.4.2. 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 Operator**가 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 **YAML**을 **<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)

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: **0**에서 **99** 사이의 정수 값을 지정합니다. 숫자가 작을수록 우선 순위가 높아지므로 우선 순위 **10**은 우선 순위 **99**보다 높습니다. 기본값은 **99**입니다.

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 **VF**보다 클 수 없습니다. **Mellanox NIC**의 경우 **VF** 수는 **127** 보다 클 수 없습니다.

8

nicSelector 매핑은 **Operator**가 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. **rootDevices**를 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 **16**진 코드를 지정합니다. 허용되는 유일한 값은 **8086** 또는 **15b3**입니다.

10

선택 사항: **SR-IOV** 네트워크 장치의 장치 **16**진수 코드를 지정합니다. 허용되는 값은 **158b, 1015, 1017**입니다.

11

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 Virtualization**의 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 **VF**를 일반 네트워크 장치로 계속 사용할 수 있습니다. 어느 모드에서나 장치를 사용할 수 있습니다.

2. 선택 사항**: SriovNetworkNodePolicy.Spec.NodeSelector** 로 레이블이 없는 경우 **SR-IOV** 가능 클러스터 노드에 레이블을 지정합니다. 노드에 레이블을 지정하는 방법에 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 **Pod**가 **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}'
```

10.21.4.3. SR-IOV 추가 네트워크 구성

SriovNetwork 오브젝트를 생성하여 **SR-IOV** 하드웨어를 사용하는 추가 네트워크를 구성할 수 있습니다.

SriovNetwork 오브젝트를 생성하면 **SR-IOV Network Operator**가 **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 **YAML**을 **<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 Operator**는 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 **SriovNetwork**의 대상 네임스페이스로 바꿉니다. 대상 네임스페이스의 **pod** 또는 가상 머신만 **SriovNetwork**에 연결할 수 있습니다.

5

선택 사항: **<vlan>**을 추가 네트워크의 **VLAN(Virtual LAN) ID**로 바꿉니다. 정수 값은 **0**에서 **4095** 사이여야 합니다. 기본값은 **0**입니다.

6

선택 사항: **<spoof_check>**를 **VF**의 위조 확인 모드로 바꿉니다. 허용되는 값은 문자열 **"on"** 및 **"off"**입니다.

중요

SR-IOV Network Operator가 지정한 값을 따옴표로 묶거나 **CR**을 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: **VF**의 경우 **<max_tx_rate>**를 최대 전송 속도(**Mbps**)로 바꿉니다.

9

참고

인텔 **NIC**는 **minTxRate** 매개변수를 지원하지 않습니다. 자세한 내용은 **BZ#1772847**에서 참조하십시오.

10

선택 사항: **<vlan_qos>**를 **VF**의 **IEEE 802.1p** 우선 순위 레벨로 바꿉니다. 기본값은 **0**입니다.

11

선택 사항: **<trust_vf>**를 **VF**의 신뢰 모드로 바꿉니다. 허용되는 값은 문자열 **"on"** 및 **"off"**입니다.

중요

SR-IOV Network Operator가 지정한 값을 따옴표로 묶거나 **CR**을 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>
```

10.21.4.4. 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.

10.21.5. 서비스 메시에 가상 머신 연결

OpenShift Virtualization은 **OpenShift Service Mesh**와 통합되었습니다. **IPv4**를 사용하여 기본 **Pod** 네트워크에서 가상 머신 워크로드를 실행하는 **Pod** 간 트래픽을 모니터링, 시각화 및 제어할 수 있습니다.

10.21.5.1. 사전 요구 사항

- **Service Mesh Operator**를 설치하고 서비스 메시 컨트롤 플레인을 배포해야 합니다.
- 가상 머신이 생성된 네임스페이스를 서비스 메시 멤버 롤에 추가해야 합니다.
- 기본 **Pod** 네트워크에는 **masquerade** 바인딩 방법을 사용해야 합니다.

10.21.5.2. 서비스 메시에 대한 가상 머신 구성

VM(가상 머신) 워크로드를 서비스 메시에 추가하려면 **sidecar.istio.io/inject** 주석을 **true** 로 설정하여 **VM** 구성 파일에서 자동 사이드카 삽입을 활성화합니다. 그런 다음 **VM**을 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** 및 **15090**이 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(라벨)입니다.

2.

3. 기본 **Pod** 네트워크와 함께 사용할 바인딩 방법(**masquerade** 모드)입니다.

2. **VM** 구성을 적용합니다.

```
$ oc apply -f <vm_name>.yaml (1)
```

1. 가상 머신 **YAML** 파일의 이름입니다.

3. 서비스 메시에 **VM**을 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** 레이블이 있는 모든 포드에서 **TCP** 포트 **8080**을 대상으로 합니다.

4. 서비스를 생성합니다.

```
$ oc create -f <service_name>.yaml (1)
```

1. 서비스 **YAML** 파일의 이름입니다.

10.21.6. 가상 머신용 IP 주소 구성

가상 머신의 고정 및 동적 IP 주소를 구성할 수 있습니다.

10.21.6.1. cloud-init를 사용하여 새 가상 머신의 IP 주소 구성

VM(가상 머신)을 생성할 때 cloud-init를 사용하여 보조 NIC의 IP 주소를 구성할 수 있습니다. IP 주소는 동적으로 또는 정적으로 프로비저닝될 수 있습니다.

참고

VM이 Pod 네트워크에 연결되어 있으면 업데이트하지 않는 한 Pod 네트워크 인터페이스가 기본 경로입니다.

사전 요구 사항

- 가상 머신이 보조 네트워크에 연결되어 있습니다.
- 가상 머신의 동적 IP를 구성하기 위해 보조 네트워크에 DHCP 서버를 사용할 수 있습니다.

절차

- 가상 머신 구성의 spec.template.spec.volumes.cloudInitNoCloud.networkData 스탠자를 편집합니다.
  - 동적 IP 주소를 구성하려면 인터페이스 이름을 지정하고 DHCP를 활성화합니다.

```
kind: VirtualMachine
spec:
# ...
  template:
  # ...
    spec:
      volumes:
      - cloudInitNoCloud:
          networkData: |
            version: 2
            ethernets:
              eth1: (1)
                dhcp4: true
```

1

인터페이스 이름을 지정합니다.

- 고정 **IP**를 구성하려면 인터페이스 이름과 **IP** 주소를 지정합니다.

```
kind: VirtualMachine
spec:
# ...
  template:
  # ...
    spec:
      volumes:
      - cloudInitNoCloud:
          networkData: |
            version: 2
            ethernets:
              eth1: (1)
                addresses:
                - 10.10.10.14/24 (2)
```

1

인터페이스 이름을 지정합니다.

2

고정 **IP** 주소를 지정합니다.

10.21.7. 가상 머신에서 NIC의 IP 주소 보기

웹 콘솔 또는 **oc** 클라이언트를 사용하여 **NIC**(네트워크 인터페이스 컨트롤러)의 **IP** 주소를 볼 수 있습니다. **QEMU** 게스트 에이전트 는 가상 머신의 보조 네트워크에 대한 추가 정보를 표시합니다.

10.21.7.1. 사전 요구 사항

- 가상 머신에 **QEMU** 게스트 에이전트를 설치합니다.

10.21.7.2. 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
```

10.21.7.3. 웹 콘솔에서 가상 머신 인터페이스의 **IP** 주소 보기

IP 정보는 가상 머신의 **VirtualMachine** 세부 정보 페이지에 표시됩니다.

절차

1. **OpenShift Container Platform** 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 를 클릭합니다.

2. 가상 머신 이름을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.

연결된 각 **NIC**에 대한 정보는 세부 정보 탭의 **IP** 주소 아래에 표시됩니다.

10.21.8. 가상 머신의 MAC 주소 풀 사용

KubeMacPool 구성 요소는 네임스페이스의 가상 머신 **NIC**에 대한 **MAC** 주소 풀 서비스를 제공합니다.

10.21.8.1. About KubeMacPool

KubeMacPool은 네임스페이스당 **MAC** 주소 풀을 제공하고 풀의 가상 머신 **NIC**에 **MAC** 주소를 할당합니다. 이렇게 하면 다른 가상 머신의 **MAC** 주소와 충돌하지 않는 고유한 **MAC** 주소가 **NIC**에 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 **KubeMacPool**은 기본적으로 활성화됩니다. 네임스페이스에 **mutatevirtualmachines.kubemacpool.io=ignore** 레이블을 추가하여 네임스페이스의 **MAC** 주소 풀을 비활성화합니다. 레이블을 제거하여 네임스페이스에 대해 **KubeMacPool**을 다시 활성화합니다.

10.21.8.2. 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
```

10.21.8.3. CLI에서 네임스페이스의 MAC 주소 풀을 다시 활성화

네임스페이스에 대해 **KubeMacPool**을 비활성화하고 다시 활성화하려면 네임스페이스에서 **mutatevirtualmachines.kubemacpool.io=ignore** 레이블을 제거합니다.

참고

이전 버전의 **OpenShift Virtualization**에서는 **mutatevirtualmachines.kubemacpool.io=allocate** 레이블을 사용하여 네임스페이스에 **KubeMacPool**을 활성화했습니다. 이는 여전히 지원되지만 **KubeMacPool**의 중복은 기본적으로 활성화되어 있습니다.

절차

- 네임스페이스에서 **KubeMacPool** 라벨을 제거합니다. 다음 예제에서는 **<namespace1>** 및 **<namespace2>** 네임스페이스에 **KubeMacPool**을 다시 사용하도록 설정합니다.

```
$ oc label namespace <namespace1> <namespace2>
mutatevirtualmachines.kubemacpool.io-
```

10.22. 가상 머신 디스크

10.22.1. 스토리지 기능

다음 표를 사용하여 **OpenShift Virtualization**의 로컬 및 공유 영구 스토리지에 대한 기능 가용성을 확인합니다.

10.22.1.1. OpenShift Virtualization 스토리지 기능 매트릭스

표 **10.5. 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. **PVC**에서 **ReadWriteMany** 액세스 모드를 요청해야 합니다.

2. 스토리지 공급자는 **Kubernetes** 및 **CSI Snapshot API**를 모두 지원해야 합니다.

참고

다음을 사용하는 가상 머신은 실시간 마이그레이션할 수 없습니다.

- **RWO(ReadWriteOnce)** 액세스 모드를 사용하는 스토리지 클래스
- **GPU**와 같은 패스스루**(passthrough)** 기능

이러한 가상 머신의 경우 **evictionStrategy** 필드를 **LiveMigrate**로 설정하지 않도록 합니다.

10.22.2. 가상 머신 로컬 스토리지 구성

hostpath 프로비전 프로그램**(HPP)**을 사용하여 가상 머신의 로컬 스토리지를 구성할 수 있습니다.

OpenShift Virtualization Operator를 설치하면 **HPP(Hostpath Provisioner) Operator**가 자동으로 설치됩니다. **HPP**는 **Hostpath Provisioner Operator**가 생성한 **OpenShift Virtualization**용으로 설계된

로컬 스토리지 프로비전 프로그램입니다. **HPP**를 사용하려면 **HPP** 사용자 정의 리소스(**CR**)를 생성해야 합니다.

10.22.2.1. 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 디렉터리에 읽기/쓰기 액세스 권한이 있어야 합니다.
- 스토리지 풀은 운영 체제와 동일한 파티션에 있지 않아야 합니다. 그렇지 않으면 운영 체제 파티션이 용량으로 채워질 수 있으며 성능에 영향을 미치거나 노드를 불안정하게 만들거나 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 **HPP**를 만듭니다.

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

10.22.2.1.1. 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**.**

hostpath 프로비전 프로그램**(HPP)**을 사용하려면 **storagePools** 스탠자를 사용하여 **CSI** 드라이버에 대한 관련 스토리지 클래스를 생성해야 합니다**.**

참고

가상 머신은 로컬 **PV**를 기반으로 하는 데이터 볼륨을 사용합니다**.** 로컬 **PV**는 특정 노드에 바인딩됩니다**.** 디스크 이미지는 가상 머신에서 사용할 수 있는 반면 가상 머신은 이전에 로컬 스토리지 **PV**가 고정된 노드에 예약할 수 없습니다**.**

이 문제를 해결하려면 **Kubernetes Pod** 스케줄러를 사용하여 **PVC(**영구 볼륨 클레임**)**를 올바른 노드의 **PV**에 바인딩합니다**. volumeBindingMode** 매개변수가 **WaitForFirstConsumer** 로 설정된 **StorageClass** 값을 사용하면 **PVC**를 사용하여 **Pod**가 생성될 때까지 **PV**의 바인딩 및 프로비저닝이 지연됩니다**.**

10.22.2.1.2. storagePools 스탠자를 사용하여 **CSI** 드라이버의 스토리지 클래스 생성

HPP(Hostpath 프로비전 프로그램**) CSI** 드라이버에 대한 스토리지 클래스 **CR(**사용자 정의 리소스**)**을 생성합니다**.**

절차

1. **storageclass_csi.yaml** 파일을 생성하여 스토리지 클래스를 정의합니다**.**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: hostpath-csi
```

```
provisioner: kubevirt.io.hostpath-provisioner
reclaimPolicy: Delete (1)
volumeBindingMode: WaitForFirstConsumer (2)
parameters:
  storagePool: my-storage-pool (3)
```

(1)

reclaimPolicy에 사용할 수 있는 값은 **Delete** 및 **Retain** 두 가지입니다. 값을 지정하지 않으면 기본값은 **Delete** 입니다.

(2)

volumeBindingMode 매개변수는 동적 프로비저닝 및 볼륨 바인딩이 발생하는 시기를 결정합니다. **PVC**(영구 볼륨 클레임)를 사용하는 **Pod**가 생성될 때까지 **WaitForFirstConsumer** 를 지정하여 **PV**(영구 볼륨)의 바인딩 및 프로비저닝을 지연합니다. 이렇게 하면 **PV**에서 **Pod**의 스케줄링 요구 사항을 충족할 수 있습니다.

(3)

HPP CR에 정의된 스토리지 풀의 이름을 지정합니다.

1. 파일을 저장하고 종료합니다.

2. 다음 명령을 실행하여 **StorageClass** 오브젝트를 만듭니다.

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

10.22.2.2. PVC 템플릿으로 생성된 스토리지 풀 정보

대용량 영구 볼륨(**PV**)이 있는 경우 **hostpath** 프로비전 프로그램(**HPP**) 사용자 정의 리소스(**CR**)에 **PVC** 템플릿을 정의하여 스토리지 풀을 생성할 수 있습니다.

PVC 템플릿으로 생성된 스토리지 풀에는 여러 개의 **HPP** 볼륨이 포함될 수 있습니다. **PV**를 작은 볼륨으로 분할하면 데이터 할당에 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 **PV**에만 필요합니다.

HPP CR에서 **pvcTemplate** 사양을 사용하여 스토리지 풀을 정의합니다. **Operator**는 **HPP CSI** 드라이버가 포함된 각 노드의 **pvcTemplate** 사양에서 **PVC**를 생성합니다. **PVC** 템플릿에서 생성된 **PVC**는 하나의 큰 **PV**를 사용하므로 **HPP**에서 더 작은 동적 볼륨을 생성할 수 있습니다.

기본 스토리지 풀을 **PVC** 템플릿에서 생성한 스토리지 풀과 결합할 수 있습니다.

10.22.2.2.1. PVC 템플릿을 사용하여 스토리지 풀 생성

HPP CR(사용자 정의 리소스)에 **PVC** 템플릿을 지정하여 여러 **HBA(Hostpath** 프로비전 프로그램) 볼륨의 스토리지 풀을 생성할 수 있습니다.

사전 요구 사항

- **spec.storagePools.path** 에 지정된 디렉터리에 읽기/쓰기 액세스 권한이 있어야 합니다.
- 스토리지 풀은 운영 체제와 동일한 파티션에 있지 않아야 합니다. 그렇지 않으면 운영 체제 파티션이 용량으로 채워질 수 있으며 성능에 영향을 미치거나 노드를 불안정하게 만들거나 사용할 수 없게 됩니다.

절차

1. 다음 예에 따라 **storagePools** 스탠자에서 **PVC(**영구 볼륨**)** 템플릿을 지정하는 **HPP CR**에 대한 **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 **Block** 또는 **Filesystem** 일 수 있습니다**.** 값을 지정하지 않으면 기본값은 **Filesystem** 입니다**. volumeMode** 가 **Block** 인 경우 마운트 **Pod**는 마운트하기 전에 블록 볼륨에 **XFS** 파일 시스템을 생성합니다**.**

(4) **storageClassName** 매개변수가 생략되면 기본 스토리지 클래스가 **PVC**를 생성하는데 사용됩니다**. storageClassName** 을 생략하면 **HPP** 스토리지 클래스가 기본 스토리지 클래스가 아닌지 확인합니다**.**

(5)

2. 파일을 저장하고 종료합니다.

3. 다음 명령을 실행하여 스토리지 풀로 **HPP**를 만듭니다.

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

추가 리소스

- 스토리지 프로파일 사용자 정의

10.22.3. 데이터 볼륨 생성

PVC 또는 스토리지 **API**를 사용하여 데이터 볼륨을 생성할 수 있습니다.

중요

OpenShift Container Platform Container Storage와 함께 **OpenShift Virtualization**을 사용하는 경우 가상 머신 디스크를 생성할 때 **RBD** 블록 모드 **PVC(**영구 볼륨 클레임**)**를 지정합니다. 가상 머신 디스크를 사용하면 **RBD** 블록 모드 볼륨이 더 효율적이고 **Ceph FS** 또는 **RBD** 파일 시스템 모드 **PVC**보다 더 나은 성능을 제공합니다.

RBD 블록 모드 **PVC**를 지정하려면 **'ocs-storagecluster-ceph-rbd'** 스토리지 클래스와 **VolumeMode: Block** 을 사용하십시오.

작은 정보

가능한 경우 스토리지 **API**를 사용하여 공간 할당을 최적화하고 성능을 극대화합니다.

스토리지 프로필은 **CDI**가 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 **CDI**는 **PVC** 생성을 최적화하는 값을 제공합니다. 그러나 스토리지 프로필을 사용자 지정하는 경우 스토리지 클래스에 대한 자동 설정을 구성할 수 있습니다.

10.22.3.1. 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. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM(**가상 머신**)** 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다.

참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다. **VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다.

10.22.3.2. 스토리지 API를 사용하여 데이터 볼륨 생성

스토리지 **API**를 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 RBD**를 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 **CDI**에서 자동으로 더 많은 공간을 요청합니다.

다음 **YAML**에서 스토리지 **API**를 사용하면 사용 가능한 공간이 **2GB**인 데이터 볼륨을 요청합니다. 사용자가 필요한 **PVC(**영구 볼륨 클레임**)** 크기를 올바르게 추정하기 위해 **volumeMode**를 알 필요가 없습니다. **CDI**는 **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 **PVC**가 존재하는 네임스페이스입니다.

4. 소스 **PVC**의 이름입니다.

5. 스토리지 **API**를 사용하여 할당을 나타냅니다.

6.

7. 선택 사항: 스토리지 클래스의 이름입니다. 스토리지 클래스를 지정하지 않으면 시스템 기본 스토리지 클래스가 사용됩니다.

10.22.3.3. 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 **YAML**에서 **PVC API**를 사용하면 **2GB**의 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 **PVC**가 존재하는 네임스페이스입니다.

4

소스 **PVC**의 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 **PVC**임을 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 **API**를 사용하는 경우 **CDI**는 파일 시스템 오버헤드를 인수하고 더 큰 **PVC(**영구 볼륨 클레임**)**를 요청하여 할당 요청이 성공했는지 확인합니다.

10.22.3.4. 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. **CDI**에서 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 **CDI**는 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.

10.22.3.4.1. 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 **Pod**와 대상 **Pod**를 사용하여 소스 볼륨의 데이터를 대상 볼륨으로 복사합니다. 호스트 지원 복제는 가장 효율적인 복제 방법입니다.

- **CSI-clone -** 이 방법은 **CSI** 복제 **API**를 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 사양 섹션의 기본 **claimPropertySets** 를 수정하지 않고 **CLI**를 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.

10.22.3.5. 추가 리소스

- 스토리지 클래스 생성 정보
- 기본 파일 시스템 오버헤드 값 덮어쓰기
- 스마트 복제를 사용하여 데이터 볼륨 복제

10.22.4. 파일 시스템 오버헤드의 PVC 공간 예약

기본적으로 **OpenShift Virtualization**은 **Filesystem** 볼륨 모드를 사용하는 **PVC**(영구 볼륨 클레임)의

파일 시스템 오버헤드 데이터를 위해 공간을 예약합니다. 전역 및 특정 스토리지 클래스에 대해 이 용도의 공간을 예약하도록 백분율을 설정할 수 있습니다.

10.22.4.1. 파일 시스템 오버헤드가 가상 머신 디스크의 공간에 미치는 영향

Filesystem 볼륨 모드를 사용하는 **PVC(**영구 볼륨 클레임**)**에 가상 머신 디스크를 추가하는 경우 **PVC**에 다음을 위한 충분한 공간이 있는지 확인해야 합니다.

- 가상 머신 디스크
- 메타데이터와 같은 파일 시스템 오버헤드용으로 예약된 공간

기본적으로 **OpenShift Virtualization**은 **5.5%**의 **PVC** 공간을 오버헤드로 예약하므로 해당 용량에 따라 가상 머신 디스크에 사용 가능한 공간을 줄일 수 있습니다.

HCO 오브젝트를 편집하여 다른 오버헤드 값을 구성할 수 있습니다. 전체적으로 값을 변경하고 특정 스토리지 클래스의 값을 지정할 수 있습니다.

10.22.4.2. 기본 파일 시스템 오버헤드 값 덮어쓰기

HCO 오브젝트의 **spec.filesystemOverhead** 속성을 편집하여 **OpenShift Virtualization**에서 파일 시스템 오버헤드를 위해 예약하는 **PVC(**영구 볼륨 클레임**)** 공간을 변경합니다.

사전 요구 사항

- **OpenShift CLI(oc)**를 설치합니다.

절차

1. 다음 명령을 실행하여 편집할 **HCO** 오브젝트를 엽니다.

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

2. **spec.filesystemOverhead** 필드를 편집하여 선택한 값으로 채웁니다.

```
...
```

```
spec:
  filesystemOverhead:
    global: "<new_global_value>" (1)
    storageClass:
      <storage_class_name>: "<new_value_for_this_storage_class>" (2)
```

(1) **set** 값이 없는 모든 스토리지 클래스에 사용되는 기본 파일 시스템 오버헤드 백분율입니다. 예를 들어, **global: "0.07"**은 파일 시스템 오버헤드용으로 **PVC**의 **7%**를 예약합니다.

(2) 지정된 스토리지 클래스의 파일 시스템 오버헤드 백분율입니다. 예를 들어, **mystorageclass: "0.04"**는 **mystorageclass** 스토리지 클래스의 **PVC**의 기본 오버헤드 값을 **4%**로 변경합니다.

3. 편집기를 저장하고 종료하여 **HCO** 오브젝트를 업데이트합니다.

검증

- 다음 명령 중 하나를 실행하여 **CDIConfig** 상태를 보고 변경 사항을 확인합니다.

  일반적으로 **CDIConfig** 변경 사항을 확인하려면 다음을 수행합니다.

```
$ oc get cdiconfig -o yaml
```

  CDIConfig 에 대한 특정 변경 사항을 보려면 다음을 수행합니다.

```
$ oc get cdiconfig -o jsonpath='{.items..status.filesystemOverhead}'
```

10.22.5. 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.

10.22.5.1. 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 리소스 요구 사항 없이 **CDI**에서 생성한 **Pod**에 기본값을 제공하고 할당량으로 제한되는 네임스페이스에서 해당 **Pod**를 실행할 수 있습니다**.**

10.22.5.2. 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 CR**을 편집합니다**.**

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

2. **spec.resourceRequirements.storageWorkloads** 스탠자를 **CR**에 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 CR**을 업데이트합니다**.**

10.22.5.3. 추가 리소스

- 프로젝트당 리소스 할당량

10.22.6. 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.

10.22.6.1. 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-network라는 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

10.22.7. 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.

10.22.7.1. 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 **CDI**는 기본 파일 시스템 및 장치 유형에 따라 더 나은 사전 할당 방법을 사용합니다.

fallocate

파일 시스템이 이를 지원하는 경우, **CDI**는 **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 **0**으로 만들 수 있습니다.

10.22.7.2. 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 **false**인 부울입니다.

10.22.8. 웹 콘솔을 사용하여 로컬 디스크 이미지 업로드

웹 콘솔을 사용하여 로컬에 저장된 디스크 이미지 파일을 업로드할 수 있습니다.

10.22.8.1. 사전 요구 사항

- 가상 머신 이미지 파일이 **IMG, ISO** 또는 **QCOW2** 형식이어야 합니다.
- **CDI** 지원 작업 매트릭스에 따라 스크래치 공간이 필요한 경우 먼저 이 작업을 완료하려면 스토리지 클래스를 정의하거나 **CDI** 스크래치 공간을 준비해야 합니다.

10.22.8.2. 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*

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

10.22.8.3. 웹 콘솔을 사용하여 이미지 파일 업로드

웹 콘솔을 사용하여 이미지 파일을 새 **PVC(**영구 볼륨 클레임**)**에 업로드합니다**.** 나중에 이 **PVC**를 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 연결 확인란을 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 **PVC**에 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.

10.22.8.4. 추가 리소스

- 데이터 볼륨 작업에 대한 쓰기 성능을 개선하도록 사전 할당 모드를 구성합니다.

10.22.9. virtctl 툴을 사용하여 로컬 디스크 이미지 업로드

virtctl 명령줄 유틸리티를 사용하여 로컬에 저장된 디스크 이미지를 신규 또는 기존 데이터 볼륨에 업로드할 수 있습니다.

10.22.9.1. 사전 요구 사항

- **virtctl**을 설치합니다.

- **CDI** 지원 작업 매트릭스에 따라 스크래치 공간이 필요한 경우 먼저 이 작업을 완료하려면 스토리지 클래스를 정의하거나 **CDI** 스크래치 공간을 준비해야 합니다.

10.22.9.2. 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. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM(**가상 머신**)** 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다.

참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다. **VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다.

10.22.9.3. 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
```

10.22.9.4. 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 **DV**를 사용하거나 이 절차 중에 새 **DV**를 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 **PVC**에 업로드할 때 **PVC** 크기는 압축되지 않은 가상 디스크의 크기보다 커야 합니다.
- **HTTPS**를 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
```

10.22.9.5. 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

10.22.9.6. 추가 리소스

- 데이터 볼륨 작업에 대한 쓰기 성능을 개선하도록 사전 할당 모드를 구성합니다.

10.22.10. 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.

10.22.10.1. 사전 요구 사항

- **virtctl**을 설치합니다.
- **CDI** 지원 작업 매트릭스에 따라 스크래치 공간이 필요한 경우 먼저 이 작업을 완료하려면 스토리지 클래스를 정의하거나 **CDI** 스크래치 공간을 준비해야 합니다.

10.22.10.2. 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. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM(**가상 머신**)** 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다.

참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다. **VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다.

10.22.10.3. 블록 영구 볼륨 정보

PV(블록 영구 볼륨)는 원시 블록 장치에서 지원하는 **PV**입니다. 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.

10.22.10.4. 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 **PV**를 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 **PV**임을 나타냅니다.

3. 선택 사항: **PV**의 스토리지 클래스를 설정합니다. 생략하면 클러스터 기본값이 사용됩니다.

4. 블록 장치가 마운트된 노드입니다.

5. 블록 **PV**를 생성합니다.

```
# oc create -f <local-block-pv10.yaml> (1)
```

1. 이전 단계에서 생성한 영구 볼륨의 파일 이름입니다.

10.22.10.5. 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
```

10.22.10.6. 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 **DV**를 사용하거나 이 절차 중에 새 **DV**를 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 **PVC**에 업로드할 때 **PVC** 크기는 압축되지 않은 가상 디스크의 크기보다 커야 합니다.
- **HTTPS**를 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
```

10.22.10.7. 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

10.22.10.8. 추가 리소스

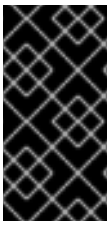
- 데이터 볼륨 작업에 대한 쓰기 성능을 개선하도록 사전 할당 모드를 구성합니다.

10.22.11. 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 API**를 지원하는 **CSI(Container Storage Interface)** 드라이버가 있는 기타 클라우드 스토리지 공급자

온라인 스냅샷에는 필요한 경우 변경할 수 있는 5분(5m)의 기본 기한이 있습니다.



중요

온라인 스냅샷은 핫플러그 가상 디스크가 있는 가상 머신에 지원됩니다. 그러나 가상 머신 사양에 없는 핫플러그 디스크는 스냅샷에 포함되어 있지 않습니다.

참고

가장 높은 무결성을 가진 온라인(실행 상태) **VM**의 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/O**가 디스크에 기록됩니다. 게스트 에이전트가 없으면 정지를 수행할 수 없으며 최상의 스냅샷을 생성합니다. 스냅샷이 수행된 조건은 웹 콘솔 또는 **CLI**에 표시되는 스냅샷 표시에 반영됩니다.

10.22.11.1. 가상 머신 스냅샷 정보

스냅샷은 특정 시점의 **VM**(가상 머신) 상태 및 데이터를 나타냅니다. 스냅샷을 사용하면 백업 및 재해 복구를 위해 기존 **VM**을 (스냅샷에 표시된) 이전 상태로 복원하거나 이전 개발 버전으로 신속하게 롤백할 수 있습니다.

VM 스냅샷은 전원이 꺼진(중지됨 상태) 또는 전원이 꺼진(실행 중 상태) **VM**에서 생성됩니다.

실행 중인 **VM**의 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 **VM**에 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 **VM**에 연결된 모든 스냅샷 나열
- 스냅샷에서 **VM** 복원

- 기존 **VM** 스냅샷 삭제

10.22.11.1.1. 가상 머신 스냅샷 컨트롤러 및 CRD(사용자 정의 리소스 정의)

스냅샷 관리를 위해 **VM** 스냅샷 기능에 다음과 같이 **CRD**로 정의되는 새 **API** 오브젝트 세 가지가 도입되었습니다.

- **VirtualMachineSnapshot:** 스냅샷을 생성하라는 사용자 요청을 나타냅니다. 여기에는 **VM**의 현재 상태 정보가 포함됩니다.
- **VirtualMachineSnapshotContent:** 클러스터의 프로비저닝 리소스를 나타냅니다(스냅샷). **VM** 스냅샷 컨트롤러에서 생성하며 **VM**을 복원하는 데 필요한 모든 리소스에 대한 참조를 포함합니다.
- **VirtualMachineRestore:** 스냅샷에서 **VM**을 복원하라는 사용자 요청을 나타냅니다.

VM 스냅샷 컨트롤러는 **VirtualMachineSnapshot** 오브젝트와 이 오브젝트에 대해 생성된 **VirtualMachineSnapshotContent** 오브젝트를 일대일 매핑으로 바인딩합니다.

10.22.11.2. 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) **VM**의 스냅샷을 생성하려면 **QEMU** 게스트 에이전트를 설치합니다.

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

절차

1. 콘솔 중 하나 또는 **SSH**를 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
```

10.22.11.3. 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) **VM**의 스냅샷을 생성하려면 **QEMU** 게스트 에이전트를 설치합니다.

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

10.22.11.3.1. 기존 Windows 가상 머신에 VirtIO 드라이버 설치

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

참고

다음 절차에서는 일반적인 방법을 사용하여 **Windows**에 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 **ID**를 선택합니다.

   c. 하드웨어 **ID**의 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.

10.22.11.3.2. 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.

10.22.11.4. 웹 콘솔에서 가상 머신 스냅샷 생성

웹 콘솔을 사용하여 **VM**(가상 머신) 스냅샷을 생성할 수 있습니다.

참고

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

QEMU 게스트 에이전트는 시스템 워크로드에 따라 **VM**의 파일 시스템을 가능한 한 많이 정지하여 일관된 스냅샷을 사용합니다. 이렇게 하면 스냅샷을 생성하기 전에 진행 중인 **I/O**가 디스크에 기록됩니다. 게스트 에이전트가 없으면 정지를 수행할 수 없으며 최상의 스냅샷을 생성합니다. 스냅샷이 수행된 조건은 웹 콘솔 또는 **CLI**에 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 → **VirtualMachine**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 클릭한 후 스냅샷 찍기를 클릭합니다.

5. 스냅샷 이름 및 선택 사항으로 설명 필드를 작성합니다.

6. 이 스냅샷에 포함된 디스크를 확장하여 스냅샷에 스토리지 볼륨이 포함되어 있는지 확인합니다.

7. **VM**에 스냅샷에 포함할 수 없는 디스크가 있고 계속 진행하려면 이 경고에 대해 인식하고 있으며 계속 진행하겠습니다. 확인란을 선택합니다.

8. 저장을 클릭합니다.

10.22.11.5. CLI에서 가상 머신 스냅샷 생성

VirtualMachineSnapshot 오브젝트를 생성하여 오프라인 또는 온라인 **VM**에 대한 **VM(**가상 머신**)** 스냅샷을 생성할 수 있습니다. **kubevirt**는 **QEMU** 게스트 에이전트와 조정하여 온라인 **VM**의 스냅샷을 생성합니다.

참고

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

QEMU 게스트 에이전트는 시스템 워크로드에 따라 **VM**의 파일 시스템을 가능한 한 많이 정지하여 일관된 스냅샷을 사용합니다. 이렇게 하면 스냅샷을 생성하기 전에 진행 중인 **I/O**가 디스크에 기록됩니다. 게스트 에이전트가 없으면 정지를 수행할 수 없으며 최상의 스냅샷을 생성합니다. 스냅샷이 수행된 조건은 웹 콘솔 또는 **CLI**에 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 기간은 **5**분**(5**분**)**입니다**. 5**분 내에 스냅샷이 성공적으로 완료되지 않으면 상태가 **failed**로 설정됩니다**.** 나중에 파일 시스템이 손상되고 **VM**이 수정되지 않지만 **failed** 스냅샷 이미지를 삭제할 때까지 상태는 실패로 유지됩니다**.**

기본 시간 데드라인을 변경하려면 스냅샷 작업이 시간 초과되기 전에 지정할 시간**(m)** 또는 초**(**초**)**로 **FailureDeadline** 속성을 **VM** 스냅샷 사양에 추가합니다**.**

시간 기한을 설정하지 않으려면 **0**을 지정할 수 있지만 **0**은 응답하지 않는 **VM**이 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 **PVC**가 스냅샷에 포함되어 있는지 확인합니다.

10.22.11.6. 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**.**

   - **CLI**를 사용하여 생성된 스냅샷의 경우 **status** 필드의 **VirtualMachineSnapshot** 오브젝트 **YAML**의 표시기 출력을 확인합니다**.**

   - 웹 콘솔을 사용하여 생성한 스냅샷의 경우 스냅샷 세부 정보 화면에서 가상 머신 스냅샷 **>** 상태를 클릭합니다**.**

2. 온라인 **VM** 스냅샷의 상태를 확인합니다**.**

   - **Online**에서는 온라인 스냅샷 생성 중에 **VM**이 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.

10.22.11.7. 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 → **VirtualMachine**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 클릭합니다. 페이지에는 가상 머신과 연결된 스냅샷 목록이 표시됩니다.
5. **VM** 스냅샷을 복원하는 다음 방법 중 하나를 선택합니다.
   a. **VM**을 복원하기 위해 소스로 사용할 스냅샷에서 복원을 클릭합니다.
   b. 스냅샷을 선택하여 스냅샷 세부 정보 화면을 열고 작업 → **VirtualMachineSnapshot** 복원 을 클릭합니다.
6. 확인 팝업 창에서 복원을 클릭하여 스냅샷에 표시된 **VM**을 이전 구성으로 복원합니다.

10.22.11.8. 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 **true**로 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. **VM**을 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.

10.22.11.9. 웹 콘솔에서 가상 머신 스냅샷 삭제

웹 콘솔을 사용하여 기존 가상 머신 스냅샷을 삭제할 수 있습니다.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachine**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.

10.22.11.10. 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 **VirtualMachineSnapshot**을 연결된 **VirtualMachineSnapshotContent** 오브젝트와 함께 삭제합니다.

```
$ oc delete vmsnapshot <my-vmsnapshot>
```

검증

- 스냅샷이 삭제되고 더 이상 이 **VM**에 연결되어 있지 않은지 확인합니다.

```
$ oc get vmsnapshot
```

10.22.11.11. 추가 리소스

- **CSI** 볼륨 스냅샷

10.22.12. 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 사용자에게는 네임스페이스에 볼륨을 복제할 수 있는 추가 사용자 권한이 필요합니다.

10.22.12.1. 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 **PV**에 고유한 라벨을 적용하십시오.

참고

대상 **PV**는 소스 **PVC**와 크기가 같거나 커야 합니다. 대상 **PV**가 소스 **PVC**보다 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 **PV**를 생성하거나 노드의 기존 로컬 **PV**를 확인합니다.

- **nodeAffinity.nodeSelectorTerms** 매개변수를 포함하는 로컬 **PV**를 생성합니다. 다음 매니페스트에서는 **node01**에 **10Gi** 로컬 **PV**를 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) **PV**의 이름입니다.

(2) **PV**의 크기입니다. 충분한 공간을 할당해야 합니다. 그러지 않으면 복제 작업이 실패합니다. 크기는 소스 **PVC**와 같거나 커야 합니다.

(3) 노드의 마운트 경로입니다.

(4) **PV**를 생성하려는 노드의 이름입니다.

-

대상 노드에 이미 존재하는 **PV**를 확인합니다. 구성에서 **nodeAffinity** 필드를 확인하여 **PV**가 프로비저닝되는 노드를 확인할 수 있습니다.

```
$ oc get pv <destination-pv> -o yaml
```

다음 스니펫은 **PV**가 **node01**에 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. **PV**에 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 **PV**에 적용한 라벨

- 대상 **PV**의 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 **PVC**의 이름입니다. **PVC** 이름을 모르는 경우 가상 머신 구성의 **spec.volumes.persistentVolumeClaim.claimName**에서 확인할 수 있습니다.

(3) 소스 **PVC**가 존재하는 네임스페이스입니다.

(4) 이전 단계에서 **PV**에 적용한 라벨입니다.

(5) 대상 **PV**의 크기

4

7. 클러스터에 데이터 볼륨 매니페스트를 적용하여 복제 작업을 시작합니다.

```
$ oc apply -f <clone-datavolume.yaml>
```

데이터 볼륨은 가상 머신의 **PVC**를 특정 노드의 **PV**에 복제합니다.

10.22.13. 빈 디스크 이미지를 추가하여 가상 스토리지 확장

OpenShift Virtualization에 빈 디스크 이미지를 추가하여 스토리지 용량을 늘리거나 새 데이터 파티션을 만들 수 있습니다.

10.22.13.1. 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. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM(**가상 머신**)** 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다.

참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다. **VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다.

10.22.13.2. 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 **1**개 이상 있습니다.
- **OpenShift CLI(oc)**를 설치합니다.

절차

1.

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
```

10.22.13.3. 추가 리소스

- 데이터 볼륨 작업에 대한 쓰기 성능을 개선하도록 사전 할당 모드를 구성합니다.

10.22.14. 스마트 복제를 사용하여 데이터 볼륨 복제

스마트 복제는 **Red Hat OpenShift Data Foundation**의 기본 제공 기능입니다. 스마트 복제는 호스트 지원 복제보다 더 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.

10.22.14.1. 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. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM(**가상 머신**)** 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다.

참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다. **VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다.

10.22.14.2. 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.

10.22.14.3. 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 **PVC**는 동일한 **volumeMode**를 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 **PVC**의 이름과 네임스페이스를 지정하는 **YAML** 파일을 생성합니다. 이 예제에서는 스토리지 **API**를 지정하므로 **accessModes** 또는 **volumeMode**를 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 **PVC**가 존재하는 네임스페이스입니다.

(3) 소스 **PVC**의 이름입니다.

(4) 스토리지 **API**로 할당을 지정합니다.

(5) 새 데이터 볼륨의 크기입니다.

2. 데이터 볼륨을 생성하여 **PVC** 복제를 시작합니다.

```
$ oc create -f <cloner-datavolume>.yaml
```

참고

데이터 볼륨이 있으면 **PVC**가 준비될 때까지 가상 머신이 시작되지 않으므로 **PVC**가 복제되는 동안 새 데이터 볼륨을 참조하는 가상 머신을 생성할 수 있습니다.

10.22.14.4. 추가 리소스

- 가상 머신 디스크의 영구 볼륨 클레임을 새 데이터 볼륨으로 복제
- 데이터 볼륨 작업에 대한 쓰기 성능을 개선하도록 사전 할당 모드를 구성합니다.
- 스토리지 프로파일 사용자 정의

10.22.15. 부팅 소스 생성 및 사용

부팅 소스에는 부팅 가능한 운영 체제(**OS**)와 **OS**의 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.

빠른 시작 둘러보기는 사용자 정의 부팅 소스 생성, 부팅 소스 업로드 및 기타 작업을 지원하기 위해 **OpenShift Container Platform** 웹 콘솔에서 사용할 수 있습니다. 도움말 메뉴에서 빠른 시작을 선택하여 빠른 시작 둘러보기를 확인합니다.

10.22.15.1. 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 소스는 운영 체제의 최신 버전으로 자동으로 업데이트됩니다. 자동 업데이트 부팅 소스의 경우 **PVC(**영구 볼륨 클레임**)**는 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.

10.22.15.2. RHEL 이미지를 부팅 소스로 가져오기

이미지 **URL**을 지정하여 **RHEL(Red Hat Enterprise Linux)** 이미지를 부팅 소스로 가져올 수 있습니다.

사전 요구 사항

- 운영 체제 이미지가 있는 웹 페이지에 액세스해야 합니다. 예를 들어 이미지를 사용하여 **Red Hat Enterprise Linux** 웹 페이지를 다운로드합니다.

절차

1. **OpenShift Container Platform** 콘솔의 사이드 메뉴에서 가상화 → 템플릿 을 클릭합니다.
2. 부팅 소스를 구성할 **RHEL** 템플릿을 확인하고 소스 추가를 클릭합니다.
3. 템플릿에 부팅 소스 추가 창의 부팅 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 **URL**을 붙여넣고 저장 및 가져오기를 클릭합니다.

검증

1. 템플릿이 템플릿 페이지의 부팅 소스 열에 녹색 확인 표시를 표시하는지 확인합니다.

이제 이 템플릿을 사용하여 **RHEL** 가상 머신을 생성할 수 있습니다.

10.22.15.3. 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 구성되면 템플릿 페이지에서 사용 가능한 **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 필요하

지 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 → 템플릿 을 클릭합니다.
2. 템플릿 옆에 있는 옵션 메뉴를 클릭하고 부팅 소스 편집 을 선택합니다.
3. 디스크 추가를 클릭합니다.
4. 디스크 추가 창에서 부팅 소스로 이 디스크 사용을 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**: Disk** 또는 **CD-ROM)**을 선택합니다.
8. 선택 사항**:** 스토리지 클래스 를 클릭하고 디스크를 생성하는 데 사용되는 스토리지 클래스를 선택합니다. 일반적으로 이 스토리지 클래스는 모든 **PVC**에서 사용하기 위해 생성되는 기본 스토리지 클래스입니다.

참고

제공된 부팅 소스는 운영 체제의 최신 버전으로 자동으로 업데이트됩니다. 자동 업데이트 부팅 소스의 경우 **PVC(**영구 볼륨 클레임**)**는 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 지워 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.

부팅 소스가 포함된 사용자 정의 가상 머신 템플릿은 카탈로그 페이지에 나열됩니다. 이 템플릿을 사용하여 가상 머신을 생성할 수 있습니다.

10.22.15.4. 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** 를 클릭합니다.

VirtualMachine 세부 정보는 **Starting** 상태로 표시됩니다.

10.22.15.5. 추가 리소스

- 가상 머신 템플릿 생성
- 사전 정의된 부팅 소스 자동 가져오기 및 업데이트

10.22.16. 가상 디스크 핫플러그

VM(가상 머신) 또는 VMI(가상 머신 인스턴스)를 중지하지 않고 가상 디스크를 추가하거나 제거할 수 있습니다.

10.22.16.1. 가상 디스크 핫플러그 정보

가상 디스크를 ***핫플러그*** 하면 가상 머신이 실행되는 동안 가상 디스크를 가상 머신 인스턴스에 연결합니다.

가상 디스크를 ***핫 플러그*** 해제하면 가상 머신이 실행되는 동안 가상 머신 인스턴스에서 가상 디스크를 분리합니다.

데이터 볼륨 및 PVC(영구 볼륨 클레임)만 핫플러그 및 핫 플러그 해제할 수 있습니다. 컨테이너 디스크를 핫 플러그 또는 핫 플러그 해제할 수 없습니다.

가상 디스크를 핫플러그한 후에는 가상 머신을 재시작하더라도 연결을 끊을 때까지 계속 연결됩니다.

10.22.16.2. virtio-scsi 정보

OpenShift Virtualization에서 각 VM(가상 머신)에는 virtio-scsi 컨트롤러가 있으므로 핫플러그된 디스크가 scsi 버스를 사용할 수 있습니다. virtio-scsi 컨트롤러는 성능 이점을 유지하면서 virtio 의 제한을 극복합니다. 확장성이 뛰어나고 4만 개 이상의 디스크 핫플러그를 지원합니다.

확장 불가능합니다. 각 virtio 디스크는 VM에서 제한된 PCI Express(PCIe) 슬롯 중 하나를 사용합니다. PCIe 슬롯은 다른 장치에서도 사용되며 사전에 예약해야 하므로 필요에 따라 슬롯을 사용할 수 없습니다.

10.22.16.3. 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**.**
- 핫플러그에 사용할 수 있는 데이터 볼륨 또는 **PVC(**영구 볼륨 클레임**)**가 하나 이상 있어야 합니다**.**

절차

- 다음 명령을 실행하여 가상 디스크를 핫플러그합니다**.**

```
$ 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 핫 플러그를 연결하거나 가상 디스크를 핫 플러그 해제할 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 머신에서 핫플러그 디스크를 식별하는 데 도움이 됩니다**.** 이 옵션을 지정하지 않으면 레이블은 기본적으로 핫플러그 데이터 볼륨 또는 **PVC**의 이름으로 설정됩니다**.**

10.22.16.4. CLI를 사용하여 가상 디스크 핫 플러그 연결

가상 머신이 실행되는 동안 **VMI(**가상 머신 인스턴스**)**에서 분리할 가상 디스크의 핫 플러그를 해제합니다**.**

사전 요구 사항

- 가상 머신이 실행 중이어야 합니다**.**
- 하나 이상의 데이터 볼륨 또는 **PVC(**영구 볼륨 클레임**)**를 사용할 수 있고 핫플러그해야 합니

다.

절차

- 다음 명령을 실행하여 가상 디스크의 핫 플러그를 해제합니다.

```
$ virtctl removevolume <virtual-machine|virtual-machine-instance> --volume-name=
<datavolume|PVC>
```

10.22.16.5. 웹 콘솔을 사용하여 가상 디스크 핫플러그

가상 머신이 실행되는 동안 **VMI(**가상 머신 인스턴스**)**에 연결하려는 가상 디스크를 핫플러그합니다. 가상 디스크를 핫플러그하면 연결 해제될 때까지 **VMI**에 남아 있습니다.

사전 요구 사항

- 가상 디스크를 핫 플러그하려면 실행 중인 가상 머신이 있어야 합니다.

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachine**를 클릭합니다.
2. 가상 디스크를 핫플러그할 실행 중인 가상 머신을 선택합니다.
3. **VirtualMachine** 세부 정보 페이지에서 디스크 탭을 클릭합니다.
4. 디스크 추가를 클릭합니다.
5. 디스크 추가**(hot plugged)** 창에서 핫 플러그로 연결하려는 가상 디스크에 대한 정보를 입력합니다.
6. 저장을 클릭합니다.

10.22.16.6. 웹 콘솔을 사용하여 가상 디스크 핫 플러그 연결

가상 머신이 실행되는 동안 **VMI(**가상 머신 인스턴스**)**에서 분리할 가상 디스크의 핫 플러그를 해제합니다**.**

사전 요구 사항

- 가상 머신이 핫플러그 디스크로 연결되어 있어야 합니다**.**

절차

1. 사이드 메뉴에서 가상화 → **VirtualMachine**를 클릭합니다**.**

2. 핫 플러그 해제하려는 디스크로 실행 중인 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다**.**

3. 디스크 탭에서 핫 플러그 해제할 가상 디스크의 옵션 메뉴 ⋮ 를 클릭합니다**.**

4. **Detach (**분리**)**를 클릭합니다**.**

10.22.17. 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 작업자 노드에 영향을 미치기 위해 **I/O** 트래픽이 증가할 수 있습니다. 이로 인해 노드를 사용할 수 없게 될 수 있습니다. 다음을 수행하여 이 문제를 해결할 수 있습니다.

- **DeploymentConfig** 오브젝트 정리
- 가비지 컬렉션 구성

10.22.17.1. 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.

10.22.17.1.1. 데이터 볼륨을 사용하여 컨테이너 디스크를 **PVC**로 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.

10.22.17.1.2. 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.

10.22.17.2. 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 **TLS**가 없는 경우 컨테이너 디스크를 영구 스토리지로 가져오기 전에 이를 비보안 레지스트리로 추가해야 합니다.

10.22.17.3. 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.

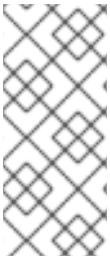
10.22.17.4. 다음 단계

- 컨테이너 디스크를 가상 머신의 영구 스토리지로 가져옵니다.
- 임시 스토리지에 **containerDisk** 볼륨을 사용하는 가상 머신을 생성합니다.

10.22.18. CDI 스크래치 공간 준비

10.22.18.1. 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. 독립 실행형 리소스로 데이터 볼륨을 생성하거나 **VM**(가상 머신) 사양의 **dataVolumeTemplate** 필드를 사용하여 생성할 수 있습니다.



참고

- 독립 실행형 데이터 볼륨을 사용하여 준비된 **VM** 디스크 **PVC**는 **VM**에서 독립 라이프사이클을 유지합니다. **VM** 사양에서 **dataVolumeTemplate** 필드를 사용하여 **PVC**를 준비하는 경우 **PVC**는 **VM**과 동일한 라이프사이클을 공유합니다.

10.22.18.2. 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

PVC를 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.

10.22.18.3. 스크래치 공간이 필요한 CDI 작업

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.

10.22.18.4. 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 CR**을 편집합니다**.**

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

2. **spec.scratchSpaceStorageClass** 필드를 **CR**에 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**.**

10.22.18.5. 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

10.22.18.6. 추가 리소스

- 동적 프로비저닝

10.22.19. 영구 볼륨 다시 사용

정적으로 프로비저닝된 **PV**(영구 볼륨)를 다시 사용하려면 먼저 볼륨을 회수해야 합니다. 이를 위해서는 스토리지 구성을 다시 사용할 수 있도록 **PV**를 삭제해야 합니다.

10.22.19.1. 정적으로 프로비저닝된 영구 볼륨 회수 정보

PV(영구 볼륨)를 회수할 때는 **PVC**(영구 볼륨 클레임)에서 **PV**를 바인딩 해제하고 **PV**를 삭제합니다. 기본 스토리지에 따라 공유 스토리지를 수동으로 삭제해야 할 수도 있습니다.

그런 다음 **PV** 구성을 다시 사용하여 다른 이름으로 **PV**를 생성할 수 있습니다.

정적으로 프로비저닝된 **PV**를 회수하려면 **PV**에 **Retain**이라는 회수 정책이 있어야 합니다. 회수 정책

이 없으면 **PVC**를 **PV**에서 바인딩 해제할 때 **PV**의 상태가 실패가 됩니다.

중요

OpenShift Container Platform 4에서는 **Recycle** 회수 정책이 사용되지 않습니다.

10.22.19.2. 정적으로 프로비저닝된 영구 볼륨 회수

PVC(영구 볼륨 클레임**)**를 바인딩 해제하고 **PV**를 삭제하여 정적으로 프로비저닝된 **PV(**영구 볼륨**)**를 회수합니다**.** 공유 스토리지를 수동으로 삭제해야 할 수도 있습니다**.**

정적으로 프로비저닝된 **PV**를 회수하는 방법은 기본 스토리지에 따라 다릅니다**.** 이 절차에서는 일반적인 접근법을 제공하며 사용 중인 스토리지에 따라 사용자 정의가 필요할 수 있습니다**.**

절차

1. **PV**의 회수 정책이 **Retain**으로 설정되어 있는지 확인합니다**.**

   a. **PV**의 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.

10.22.20. 가상 머신 디스크 확장

가상 머신의 **(VM)** 디스크의 크기를 확대하여 디스크의 **PVC(**영구 볼륨 클레임**)**의 크기를 조정하여 더

큰 스토리지 용량을 제공할 수 있습니다.

그러나 **VM** 디스크의 크기를 줄일 수 없습니다.

10.22.20.1. 가상 머신 디스크 설명

VM 디스크를 사용하면 가상 머신에서 추가 공간을 사용할 수 있습니다. 그러나 **VM** 소유자가 스토리지를 사용하는 방법을 결정하는 것은 책임이 있습니다.

디스크가 파일 시스템 **PVC**인 경우 일치하는 파일은 파일 시스템 오버헤드의 일부 공간을 예약하면서 나머지 크기로 확장됩니다.

절차

1. 확장하려는 **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) 늘릴 수 있는 **VM** 디스크 크기

10.22.20.2. 추가 리소스

- **Windows**에서 기본 볼륨 확장.

- **Red Hat Enterprise Linux**에서 데이터를 삭제하지 않고 기존 파일 시스템 파티션을 확장합니다.

- **Red Hat Enterprise Linux**에서 온라인 논리 볼륨 및 파일 시스템 확장.

11장. 가상 머신 템플릿

11.1. 가상 머신 템플릿 생성

11.1.1. 가상 머신 템플릿 정보

사전 구성된 **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 다음과 같은 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, 워크로드와 같은 속성별로 템플릿을 정렬할 수 있습니다.

Red Hat 지원 또는 **Red Hat** 제공 템플릿을 편집하거나 삭제할 수 없습니다. 템플릿을 복제하고 사용자 지정 가상 머신 템플릿으로 저장한 다음 편집할 수 있습니다.

YAML 파일 예제를 편집하여 사용자 정의 가상 머신 템플릿을 생성할 수도 있습니다.

11.1.2. 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 소스는 운영 체제의 최신 버전으로 자동으로 업데이트됩니다. 자동 업데이트 부팅 소스의 경우 **PVC(**영구 볼륨 클레임**)**는 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.

11.1.3. 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 → 템플릿 을 클릭합니다.

2. 선택 사항**:** 프로젝트 드롭다운 메뉴를 사용하여 새 템플릿과 관련된 프로젝트를 변경합니다. 모든 템플릿은 기본적으로 **openshift** 프로젝트에 저장됩니다.

3. 템플릿 생성을 클릭합니다.

4. **YAML** 파일을 편집하여 템플릿 매개 변수를 지정합니다.

–

5. 생성을 클릭합니다.

   템플릿은 템플릿 페이지에 표시됩니다.

6. 선택 사항: **YAML** 파일을 다운로드하여 저장하려면 다운로드를 클릭합니다.

11.1.4. 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 구성되면 템플릿 페이지에서 사용 가능한 **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 → 템플릿 을 클릭합니다.
2. 템플릿 옆에 있는 옵션 메뉴를 클릭하고 부팅 소스 편집 을 선택합니다.
3. 디스크 추가를 클릭합니다.
4. 디스크 추가 창에서 부팅 소스로 이 디스크 사용을 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**: Disk** 또는 **CD-ROM)**을 선택합니다**.**
8. 선택 사항**:** 스토리지 클래스 를 클릭하고 디스크를 생성하는 데 사용되는 스토리지 클래스를 선택합니다**.** 일반적으로 이 스토리지 클래스는 모든 **PVC**에서 사용하기 위해 생성되는 기본 스토리지 클래스입니다**.**

참고

제공된 부팅 소스는 운영 체제의 최신 버전으로 자동으로 업데이트됩니다**.** 자동 업데이트 부팅 소스의 경우 **PVC(**영구 볼륨 클레임**)**는 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 지워 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.

부팅 소스가 포함된 사용자 정의 가상 머신 템플릿은 카탈로그 페이지에 나열됩니다. 이 템플릿을 사용하여 가상 머신을 생성할 수 있습니다.

11.1.4.1. 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 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 끝점에서 사용할 수 있는 이미지에서 콘텐츠를 가져옵니다. 이미지 다운로드를 사용할 수 있는 웹 페이지에서 다운로드 링크 URL을 가져와 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 생성(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.

11.1.5. 추가 리소스

- 부팅 소스 생성 및 사용
- 스토리지 프로파일 사용자 정의

11.2. 가상 머신 템플릿 편집

웹 콘솔에서 가상 머신 템플릿을 편집할 수 있습니다.

참고

Red Hat Virtualization Operator에서 제공하는 템플릿을 편집할 수 없습니다. 템플릿을 복제하는 경우 템플릿을 편집할 수 있습니다.

11.2.1. 웹 콘솔에서 가상 머신 템플릿 편집

OpenShift Container Platform 웹 콘솔 또는 명령줄 인터페이스를 사용하여 가상 머신 템플릿을 편집할 수 있습니다.

가상 머신 템플릿을 편집해도 해당 템플릿에서 이미 생성된 가상 머신에는 영향을 미치지 않습니다.

절차

1. 웹 콘솔에서 가상화 → 템플릿 으로 이동합니다.

2. 가상 머신 템플릿 옆에 있는 ⋮ 옵션 메뉴를 클릭하고 편집할 오브젝트를 선택합니다.

3. **Red Hat** 템플릿을 편집하려면 ⋮ 옵션 메뉴를 클릭하고 복제 를 선택하여 사용자 지정 템플릿을 생성한 다음 사용자 지정 템플릿을 편집합니다.

   참고

   템플릿의 데이터 소스를 **DataImportCron** 사용자 정의 리소스에서 관리하거나 템플릿에 데이터 볼륨 참조가 없는 경우 부팅 소스 참조가 비활성화됩니다.

4. 저장을 클릭합니다.

11.2.1.1. 가상 머신 템플릿에 네트워크 인터페이스 추가

네트워크 인터페이스를 가상 머신 템플릿에 추가하려면 이 절차를 사용하십시오.

절차

1. 사이드 메뉴에서 가상화 → 템플릿 을 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.

11.2.1.2. 가상 머신 템플릿에 가상 디스크 추가

가상 디스크를 가상 머신 템플릿에 추가하려면 이 절차를 사용하십시오.

절차

1. 사이드 메뉴에서 가상화 → 템플릿 을 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 및 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** 설정을 지워 지워 가상 디스크의 볼륨 모드 및 액세스 모드 를 변경할 수 있습니다. 이러한 매개변수를 지정하지 않으면 **kubevirt-storage-class-defaults** 구성 맵의 기본값이 사용됩니다.

5. 추가를 클릭합니다.

11.2.1.3. 템플릿용 CD-ROM 편집

가상 머신 템플릿에 대해 **CD-ROM**을 편집하려면 다음 절차를 사용하십시오.

절차

1. 사이드 메뉴에서 가상화 → 템플릿 을 클릭합니다.
2. 가상 머신 템플릿을 선택하여 템플릿 세부 정보 화면을 엽니다.
3. 디스크 탭을 클릭합니다.
4. 편집하려는 **CD-ROM**의 옵션 메뉴 ⋮ 를 클릭하고 편집을 선택합니다.
5. **CD-ROM** 편집 창에서 소스, 영구 볼륨 클레임, 이름, 유형 및 인터페이스 필드를 편집합니다.
6. 저장을 클릭합니다.

11.3. 가상 머신 템플릿 전용 리소스 활성화

가상 머신은 성능 향상을 위해 **CPU**와 같은 노드 리소스를 전용으로 사용할 수 있습니다.

11.3.1. 전용 리소스 정보

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

11.3.2. 사전 요구 사항

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

11.3.3. 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 → 템플릿 을 클릭합니다.
2. 가상 머신 템플릿을 선택하여 템플릿 세부 정보 페이지를 엽니다.
3. **Scheduling** 탭에서 전용 리소스 옆에 있는 연필 아이콘을 클릭합니다.
4. 전용 리소스(보장된 정책)를 사용하여 이 워크로드 예약을 선택합니다.
5. 저장을 클릭합니다.

11.4. 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.

11.4.1. 템플릿에 대한 사용자 정의 네임스페이스 생성

해당 템플릿에 액세스할 수 있는 권한이 있는 모든 사용자가 사용할 가상 머신 템플릿을 배포하는 데 사용되는 사용자 지정 네임스페이스를 생성할 수 있습니다. 사용자 지정 네임스페이스에 템플릿을 추가하려면 **HyperConverged CR**(사용자 정의 리소스)을 편집하고, **spec**에 **commonTemplatesNamespace**를 추가하고, 가상 머신 템플릿의 사용자 지정 네임스페이스를 지정합니다. **HyperConverged CR**이 수정되면 **ssp-operator** 에서 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>
```

11.4.2. 사용자 정의 네임스페이스에 템플릿 추가

ssp-operator 는 기본적으로 가상 머신 템플릿을 **openshift** 네임스페이스에 배포합니다. **openshift** 네임스페이스의 템플릿은 모든 사용자에게 공개적으로 사용할 수 있습니다. 사용자 정의 네임스페이스가 생성되고 해당 네임스페이스에 템플릿이 추가되면 **openshift** 네임스페이스에서 가상 머신 템플릿을 수정하거나 삭제할 수 있습니다. 사용자 지정 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 사용 가능한 가상 머신 템플릿 목록을 확인합니다.

```
$ oc get templates -n customnamespace
```

4. **commonTemplatesNamespace** 특성을 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 지정 네임스페이스입니다.

5. 변경 사항을 저장하고 편집기를 종료합니다. **ssp-operator** 는 기본 **openshift** 네임스페이스에 존재하는 가상 머신 템플릿을 사용자 정의 네임스페이스에 추가합니다.

11.4.2.1. 사용자 정의 네임스페이스에서 템플릿 삭제

사용자 지정 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. 제거된 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
```

11.4.2.2. 추가 리소스

- 가상 머신 템플릿 생성

11.5. 가상 머신 템플릿 삭제

웹 콘솔을 사용하여 **Red Hat** 템플릿을 기반으로 사용자 지정 가상 머신 템플릿을 삭제할 수 있습니다.

Red Hat 템플릿을 삭제할 수 없습니다.

11.5.1. 웹 콘솔에서 가상 머신 템플릿 삭제

가상 머신 템플릿을 삭제하면 클러스터에서 해당 템플릿이 영구적으로 제거됩니다.

참고

사용자 지정 가상 머신 템플릿을 삭제할 수 있습니다. **Red Hat** 제공 템플릿은 삭제할 수 없습니다.

절차

1. **OpenShift Container Platform** 콘솔의 사이드 메뉴에서 가상화 → 템플릿 을 클릭합니다.

2. 템플릿의 옵션 메뉴

   ⋮

   를 클릭하고 템플릿 삭제 를 선택합니다.

3. 삭제를 클릭합니다.

12장. 실시간 마이그레이션

12.1. 가상 머신 실시간 마이그레이션

12.1.1. 실시간 마이그레이션 정보

실시간 마이그레이션은 가상 워크로드 또는 액세스를 중단하지 않고 실행 중인 **VMI(**가상 머신 인스턴스**)**를 클러스터의 다른 노드로 이동하는 프로세스입니다**. VMI**에서 **LiveMigrate** 제거 전략을 사용하는 경우 **VMI**가 유지보수 모드로 실행되는 노드가 유지보수 모드에 배치될 때 자동으로 마이그레이션됩니다**.** 마이그레이션할 **VMI**를 선택하여 실시간 마이그레이션을 수동으로 시작할 수도 있습니다**.**

다음 조건이 충족되면 실시간 마이그레이션을 사용할 수 있습니다**.**

- **RWX(ReadWriteMany)** 액세스 모드를 사용한 공유 스토리지**.**
- 충분한 **RAM** 및 네트워크 대역폭**.**
- 가상 머신에서 호스트 모델 **CPU**를 사용하는 경우 노드에서 가상 머신의 호스트 모델 **CPU**를 지원해야 합니다**.**

기본적으로 실시간 마이그레이션 트래픽은 **TLS(Transport Layer Security)**를 사용하여 암호화됩니다**.**

12.1.2. 추가 리소스

- 가상 머신 인스턴스를 다른 노드로 마이그레이션
- 실시간 마이그레이션 모니터링
- 실시간 마이그레이션 제한
- 스토리지 프로파일 사용자 정의

12.2. 실시간 마이그레이션 제한 및 타임아웃

마이그레이션 프로세스에서 클러스터를 전부 사용하지 않도록 실시간 마이그레이션 제한 및 타임아웃을 적용합니다. **HyperConverged CR**(사용자 정의 리소스)을 편집하여 이러한 설정을 구성합니다.

12.2.1. 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

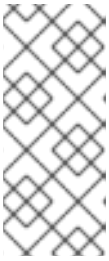
- **HyperConverged CR**을 편집하고 필요한 실시간 마이그레이션 매개변수를 추가합니다.

```
$ 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: 150**을 복원합니다.

12.2.2. 클러스터 수준의 실시간 마이그레이션 제한 및 타임아웃

표 **12.1.** 마이그레이션 매개변수

매개변수	설명	기본
parallelMigrationsPerCluster	클러스터에서 병렬로 실행되고 있는 마이그레이션의 수입니다.	5
parallelOutboundMigrationsPerNode	노드당 최대 아웃바운드 마이그레이션의 수입니다.	2
bandwidthPerMigration	각 마이그레이션의 대역폭 제한입니다. 여기서 값은 초당 바이트 수입니다. 예를 들어, **2048Mi** 값은 2048MiB/s를 의미합니다.	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.

12.3. 가상 머신 인스턴스를 다른 노드로 마이그레이션

웹 콘솔 또는 **CLI**를 사용하여 다른 노드로의 가상 머신 인스턴스 실시간 마이그레이션을 수동으로 시작합니다.

참고

가상 머신에서 호스트 모델 **CPU**를 사용하는 경우 호스트 **CPU** 모델을 지원하는 노드 간에만 해당 가상 머신을 실시간 마이그레이션할 수 있습니다.

12.3.1. 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** 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 를 클릭합니다.

2. 이 페이지에서 마이그레이션을 시작하면 동일한 페이지에서 여러 가상 머신에 대한 작업을 더 쉽게 수행할 수 있습니다. **VirtualMachine** 세부 정보 페이지에서 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.

12.3.2. 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.

12.3.3. 추가 리소스

- 실시간 마이그레이션 모니터링
- 가상 머신 인스턴스의 실시간 마이그레이션 취소

12.4. 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.

12.4.1. 가상 머신 실시간 마이그레이션을 위한 전용 보조 네트워크 구성

실시간 마이그레이션을 위해 전용 보조 네트워크를 구성하려면 먼저 **CLI**를 사용하여 **openshift-cnv** 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 사용되는 **NIC**는 동일한 **VLAN**에 연결되어 있습니다.
- 가상 머신(**VM**)은 **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 (2)
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "migration-bridge",
    "type": "macvlan",
    "master": "eth1", (3)
    "mode": "bridge",
    "ipam": {
      "type": "whereabouts", (4)
      "range": "10.200.5.0/24" (5)
    }
  }'
```

(1) **NetworkAttachmentDefinition** 오브젝트의 이름입니다.

2. **NetworkAttachmentDefinition** 오브젝트가 있는 네임스페이스입니다. **openshift-cnv** 여야 합니다.

3. 실시간 마이그레이션에 사용할 **NIC**의 이름입니다.

4. 이 네트워크 연결 정의에 대한 네트워크를 제공하는 **CNI** 플러그인의 이름입니다.

5. 보조 네트워크의 **IP** 주소 범위입니다. 이 범위는 기본 네트워크의 **IP** 주소와 겹치지 않아야 합니다.

2. 다음 명령을 실행하여 기본 편집기에서 **HyperConverged CR**을 엽니다.

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

3. **NetworkAttachmentDefinition** 오브젝트의 이름을 **HyperConverged CR**의 **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 시작하고 보조 네트워크에 연결합니다.

검증

- 가상 머신이 실행되는 노드가 유지 관리 모드로 전환되면 **VM**이 클러스터의 다른 노드로 자동 마이그레이션됩니다. **VMI(**가상 머신 인스턴스**)** 메타데이터에서 대상 **IP** 주소를 확인하여 마이그레이션이 기본 **Pod** 네트워크가 아닌 보조 네트워크를 통해 마이그레이션이 발생했는지 확인할 수 있습니다.

```
oc get vmi <vmi_name> -o jsonpath='{.status.migrationState.targetNodeAddress}'
```

12.4.2. 웹 콘솔을 사용하여 전용 네트워크 선택

OpenShift Container Platform 웹 콘솔을 사용하여 실시간 마이그레이션 전용 네트워크를 선택할 수 있습니다.

사전 요구 사항

- 실시간 마이그레이션을 위해 **Multus** 네트워크를 구성하셨습니다.

절차

1. **OpenShift Container Platform** 웹 콘솔에서 **Virtualization > Overview** 로 이동합니다.

2. 설정 탭을 클릭한 다음 실시간 마이그레이션 을 클릭합니다.

3. **Live migration network** 목록에서 네트워크를 선택합니다.

12.4.3. 추가 리소스

- 실시간 마이그레이션 제한 및 타임아웃

12.5. 가상 머신 인스턴스의 실시간 마이그레이션 취소

가상 머신 인스턴스가 원래 노드에 남아 있도록 실시간 마이그레이션을 취소합니다.

웹 콘솔 또는 **CLI**에서 실시간 마이그레이션을 취소할 수 있습니다.

12.5.1. 웹 콘솔에서 가상 머신 인스턴스의 실시간 마이그레이션 취소

웹 콘솔에서 가상 머신 인스턴스의 실시간 마이그레이션을 취소할 수 있습니다.

절차

1. **OpenShift Container Platform** 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 를 클릭합니다.

2. 가상 머신 옆에 있는 옵션 메뉴

   ⋮

   를 클릭하고 마이그레이션 취소 를 선택합니다.

12.5.2. 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
```

12.6. 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.

12.6.1. 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>
```

12.7. 실시간 마이그레이션 정책 구성

실시간 마이그레이션 정책을 사용하여 지정된 **VMI(**가상 머신 인스턴스**)** 그룹에 대한 다양한 마이그레이션 구성을 정의할 수 있습니다.

중요

실시간 마이그레이션 정책은 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.

웹 콘솔을 사용하여 실시간 마이그레이션 정책을 구성하려면 **MigrationPolicies** 페이지 설명서를 참조하십시오.

12.7.1. 명령줄에서 실시간 마이그레이션 정책 구성

MigrationPolicy CRD(사용자 정의 리소스 정의**)**를 사용하여 선택한 **VMI(**가상 머신 인스턴스**)** 그룹의 하나 이상의 마이그레이션 정책을 정의합니다.

다음 조합을 사용하여 **VMI** 그룹을 지정할 수 있습니다.

- 크기**,os,gpu** 및 기타 **VMI** 라벨과 같은 가상 머신 인스턴스 레이블입니다.
- 우선순위**,**대역폭**,hpc-workload** 및 기타 네임스페이스 라벨과 같은 네임스페이스 레이블입니다.

정책이 특정 **VMI** 그룹에 적용되려면 **VMI** 그룹의 모든 라벨이 정책의 라벨과 일치해야 합니다.

참고

VMI에 여러 실시간 마이그레이션 정책이 적용되는 경우 일치하는 라벨이 가장 많은 정책이 우선합니다. 여러 정책이 이 기준을 충족하는 경우 일치하는 라벨 키의 사전순으로 정책이 정렬되며 첫 번째 정책이 우선합니다.

절차

1.

지정된 **VMI** 그룹에 대한 **MigrationPolicy CRD**를 생성합니다. 다음 예제 **YAML**은 **hpc-workloads:true,xyz-workloads-type: "", workload-type: db, operating-system: ""** 라벨이 있는 그룹을 구성합니다.

```
apiVersion: migrations.kubevirt.io/v1alpha1
kind: MigrationPolicy
metadata:
  name: my-awesome-policy
spec:
  # Migration Configuration
  allowAutoConverge: true
  bandwidthPerMigration: 217Ki
  completionTimeoutPerGiB: 23
  allowPostCopy: false

  # Matching to VMIs
  selectors:
    namespaceSelector: (1)
      hpc-workloads: "True"
      xyz-workloads-type: ""
    virtualMachineInstanceSelector: (2)
      workload-type: "db"
      operating-system: ""
```

(1) **namespaceSelector** 를 사용하여 네임스페이스 라벨을 사용하여 **VMI** 그룹을 정의합니다.

(2) **virtualMachineInstanceSelector** 를 사용하여 **VMI** 라벨을 사용하여 **VMI** 그룹을 정의합니다.

13장. 노드 유지보수

13.1. 노드 유지보수 정보

13.1.1. 노드 유지보수 모드 정보

노드는 **oc adm** 유틸리티 또는 **NodeMaintenance** 사용자 정의 리소스 **(CR)**를 사용하여 유지보수 모드로 전환할 수 있습니다.

참고

node-maintenance-operator (NMO)는 더 이상 **OpenShift Virtualization**과 함께 제공되지 않습니다. 이제 **OpenShift Container Platform** 웹 콘솔의 **OperatorHub** 에서 독립형 **Operator**로 배포하거나 **OpenShift CLI(oc)**를 사용하여 배포할 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.

Node Maintenance Operator는 신규 또는 삭제된 **NodeMaintenance CR**을 감시합니다. 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.

13.1.2. 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 환경에서와 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 베에메탈 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.

13.1.3. 추가 리소스

- **CLI**를 사용하여 **Node Maintenance Operator** 설치
- 노드를 유지보수 모드로 설정
- 유지관리 모드에서 노드 재시작
- 가상 머신 **RunStrategies** 정보
- 가상 머신 실시간 마이그레이션
- 가상 머신 제거 전략 구성

13.2. TLS 인증서 자동 갱신

OpenShift Virtualization 구성 요소에 대한 모든 **TLS** 인증서는 자동으로 갱신되고 순환됩니다. 수동

으로 새로 고치지 않아도 됩니다.

13.2.1. 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

13.3. 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.

13.3.1. 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 **13.1.** 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.

13.3.2. 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 **13.2. Penryn**에서 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 **13.3. Haswell**에서 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 **13.4. 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
```

13.3.3. 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 **CR**에 표시되지 않습니다.

(2) 이 값을 기본 **CPU** 기능에 사용할 최소 **CPU** 모델로 바꿉니다. 값을 지정하지 않으면 기본적으로 **Penryn**이 사용됩니다.

13.4. 노드 조정 방지

node-labeller가 노드를 조정하지 못하도록 하려면 **skip-node** 주석을 사용합니다.

13.4.1. skip-node 주석 사용

node-labeller에서 노드를 건너뛰려면 **oc CLI**를 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.

13.4.2. 추가 리소스

- 더 이상 사용되지 않는 **CPU** 모델에 대한 노드 라벨링 관리

14장. 로깅, 이벤트, 모니터링

14.1. 가상화 개요 페이지

가상화 개요 페이지에서는 가상화 리소스, 세부 정보, 상태 및 상위 소비자에 대한 포괄적인 보기를 제공합니다.

- 개요 탭에는 시작하기 리소스, 세부 정보, 인벤토리, 경고 및 **OpenShift Virtualization** 환경에 대한 기타 정보가 표시됩니다.
- 상위 소비자 탭에는 프로젝트, 가상 머신 또는 노드별 특정 리소스의 사용률이 높습니다.
- 마이그레이션 탭에 실시간 마이그레이션 상태가 표시됩니다.
- 설정 탭에는 실시간 마이그레이션 설정 및 사용자 권한을 포함하여 클러스터 전체 설정이 표시됩니다.

OpenShift Virtualization의 전반적인 상태에 대한 통찰력을 확보하면 데이터를 검사하여 확인된 특정 문제를 해결하는 데 개입이 필요한지 확인할 수 있습니다.

14.1.1. 상위 소비자 검토

가상화 개요 페이지의 상위 소비자 탭에서 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.
- 상위 소비자 탭에서 **vCPU** 대기 지표를 사용하려면 **MachineConfig** 오브젝트에 **schedstats=enable** 커널 인수를 적용해야 합니다.

절차

1. **OpenShift Container Platform** 웹 콘솔의 관리자 관점에서 가상화 → 개요 로 이동합니다.

2. 상위 소비자 탭을 클릭합니다.

3. 선택 사항: 기간을 선택하거나 **5** 또는 **10**개의 상위 소비자를 선택하여 결과를 필터링할 수 있습니다.

14.1.2. 추가 리소스

- 노드에 커널 인수 추가
- 모니터링 개요
- 모니터링 대시보드 검토
- 대시보드

14.2. OPENSHIFT VIRTUALIZATION 로그 보기

웹 콘솔 또는 **oc CLI**를 사용하여 **OpenShift Virtualization** 구성 요소 및 가상 머신의 로그를 볼 수 있습니다. **virt-launcher Pod**에서 가상 머신 로그를 검색할 수 있습니다. 로그 세부 정보 표시를 제어하려면 **HyperConverged** 사용자 정의 리소스를 편집합니다.

14.2.1. CLI를 사용하여 OpenShift Virtualization 로그 보기

HyperConverged CR(사용자 정의 리소스)을 편집하여 **OpenShift Virtualization** 구성 요소에 대한 로그 상세 정보를 구성합니다. 그런 다음 **oc CLI** 툴을 사용하여 구성 요소 **Pod**의 로그를 확인합니다.

절차

1. 특정 구성 요소에 대한 로그 상세 정보를 설정하려면 다음 명령을 실행하여 기본 텍스트 편집기에서 **HyperConverged CR**을 엽니다.

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

2. **spec.logVerbosityConfig** 스탠자를 편집하여 하나 이상의 구성 요소에 대한 로그 수준을 설정합니다. 예를 들면 다음과 같습니다.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
spec:
  logVerbosityConfig:
    kubevirt:
      virtAPI: 5 (1)
      virtController: 4
      virtHandler: 3
      virtLauncher: 2
      virtOperator: 6
```

(1) 로그 세부 정보 표시 값은 **1-9** 범위의 정수여야 합니다. 여기서 숫자가 더 자세한 로그를 나타냅니다. 이 예에서는 우선순위 수준이 **5** 이상인 경우 **virtAPI** 구성 요소 로그가 노출됩니다.

3. 편집기를 저장하고 종료하여 변경 사항을 적용합니다.

4. 다음 명령을 실행하여 **OpenShift Virtualization** 네임스페이스에서 **Pod** 목록을 확인합니다.

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

예 **14.1.** 출력 예

```
NAME                                READY   STATUS    RESTARTS   AGE
disks-images-provider-7gqbc         1/1     Running   0          32m
disks-images-provider-vg4kx         1/1     Running   0          32m
virt-api-57fcc4497b-7qfmc           1/1     Running   0          31m
virt-api-57fcc4497b-tx9nc           1/1     Running   0          31m
virt-controller-76c784655f-7fp6m    1/1     Running   0          30m
virt-controller-76c784655f-f4pbd    1/1     Running   0          30m
virt-handler-2m86x                  1/1     Running   0          30m
virt-handler-9qs6z                  1/1     Running   0          30m
virt-operator-7ccfdbf65f-q5snk      1/1     Running   0          32m
virt-operator-7ccfdbf65f-vllz8      1/1     Running   0          32m
```

5. 구성 요소 **Pod**의 로그를 보려면 다음 명령을 실행합니다.

```
$ oc logs -n openshift-cnv <pod_name>
```

예를 들면 다음과 같습니다.

```
$ oc logs -n openshift-cnv virt-handler-2m86x
```

참고

Pod를 시작하지 못하면 **--previous** 옵션을 사용하여 마지막 시도에서 로그를 볼 수 있습니다.

로그 출력을 실시간으로 모니터링하려면 **-f** 옵션을 사용합니다.

예 **14.2.** 출력 예

```
{"component":"virt-handler","level":"info","msg":"set verbosity to 2","pos":"virt-handler.go:453","timestamp":"2022-04-17T08:58:37.373695Z"}
{"component":"virt-handler","level":"info","msg":"set verbosity to 2","pos":"virt-handler.go:453","timestamp":"2022-04-17T08:58:37.373726Z"}
{"component":"virt-handler","level":"info","msg":"setting rate limiter to 5 QPS and 10 Burst","pos":"virt-handler.go:462","timestamp":"2022-04-17T08:58:37.373782Z"}
{"component":"virt-handler","level":"info","msg":"CPU features of a minimum baseline CPU model: map[apic:true clflush:true cmov:true cx16:true cx8:true de:true fpu:true fxsr:true lahf_lm:true lm:true mca:true mce:true mmx:true msr:true mtrr:true nx:true pae:true pat:true pge:true pni:true pse:true pse36:true sep:true sse:true sse2:true sse4.1:true ssse3:true syscall:true tsc:true]","pos":"cpu_plugin.go:96","timestamp":"2022-04-17T08:58:37.390221Z"}
{"component":"virt-handler","level":"warning","msg":"host model mode is expected to contain only one model","pos":"cpu_plugin.go:103","timestamp":"2022-04-17T08:58:37.390263Z"}
{"component":"virt-handler","level":"info","msg":"node-labeller is running","pos":"node_labeller.go:94","timestamp":"2022-04-17T08:58:37.391011Z"}
```

14.2.2. 웹 콘솔에서 가상 머신 로그 보기

연결된 가상 머신 시작 관리자 **Pod**에서 가상 머신 로그를 가져옵니다.

절차

1. **OpenShift Container Platform** 콘솔 의 사이드 메뉴에서 가상화 → **VirtualMachine** 를 클릭합니다.

2. 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.

3. 세부 정보 탭을 클릭합니다.

4. **Pod** 섹션에서 **virt-launcher-<name> Pod**를 클릭하여 **Pod** 세부 정보 페이지를 엽니다.

5. **Logs (로그)** 탭을 클릭하여 포드 로그를 확인합니다.

14.2.3. 일반적인 오류 메시지

OpenShift Virtualization 로그에 다음과 같은 오류 메시지가 표시될 수 있습니다.

ErrImagePull 또는 **ImagePullBackOff**

는 잘못된 배포 구성 또는 참조되는 이미지 관련 문제를 나타냅니다.

14.3. 이벤트 보기

14.3.1. 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 **CLI**에서 볼 수 있습니다.

OpenShift Container Platform 클러스터에서 시스템 이벤트 정보 보기도 참조하십시오.

14.3.2. 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 → **VirtualMachine**를 클릭합니다.

2. 가상 머신을 선택하여 **VirtualMachine** 세부 정보 페이지를 엽니다.

3. 이벤트 탭을 클릭하여 가상 머신의 스트리밍 이벤트를 확인합니다.

   - **tektontekton** 버튼은 이벤트 스트림을 일시 중지합니다.
   - ▶ 버튼은 일시 정지된 이벤트 스트림을 다시 시작합니다.

14.3.3. CLI에서 네임스페이스 이벤트 보기

OpenShift Container Platform 클라이언트를 사용하여 네임스페이스에 대한 이벤트를 가져옵니다.

절차

- 네임스페이스에서 **oc get** 명령을 사용합니다.

```
$ oc get events
```

14.3.4. 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>
```

14.4. 실시간 마이그레이션 모니터링

웹 콘솔 또는 **CLI**에서 실시간 마이그레이션 진행 상황을 모니터링할 수 있습니다.

14.4.1. 웹 콘솔을 사용하여 실시간 마이그레이션 모니터링

웹 콘솔에서 개요 → 마이그레이션 탭에서 모든 실시간 마이그레이션의 진행 상황을 모니터링할 수 있습니다.

웹 콘솔의 **VirtualMachine** 세부 정보 → 메트릭 탭에서 가상 머신의 마이그레이션 메트릭 을 볼 수 있습니다.

14.4.2. 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
```

14.4.3. 메트릭

Prometheus 쿼리 를 사용하여 실시간 마이그레이션을 모니터링할 수 있습니다.

14.4.3.1. 실시간 마이그레이션 메트릭

실시간 마이그레이션 상태를 표시하려면 다음 메트릭을 쿼리할 수 있습니다.

kubevirt_migrate_vmi_data_processed_bytes

새 **VM**(가상 머신)으로 마이그레이션된 게스트 운영 체제(**OS**) 데이터의 양입니다. 유형: 게이지.

kubevirt_migrate_vmi_data_remaining_bytes

마이그레이션 중인 게스트 **OS** 데이터의 양입니다. 유형: 게이지.

kubevirt_migrate_vmi_dirty_memory_rate_bytes

게스트 **OS**에서 메모리가 더러워지는 속도입니다. 더러움 메모리는 변경되었지만 아직 디스크에 기록되지 않은 데이터입니다. 유형: 게이지.

kubevirt_migrate_vmi_pending_count

보류 중인 마이그레이션 수입니다. 유형: 게이지.

kubevirt_migrate_vmi_scheduling_count

스케줄링 마이그레이션 수입니다. 유형: 게이지.

kubevirt_migrate_vmi_running_count

실행 중인 마이그레이션 수입니다. 유형: 게이지.

kubevirt_migrate_vmi_succeeded

성공적으로 완료된 마이그레이션 수입니다. 유형: 게이지.

kubevirt_migrate_vmi_failed

실패한 마이그레이션 수입니다. 유형: 게이지.

14.5. 이벤트 및 조건을 사용하여 데이터 볼륨 진단

oc describe 명령을 사용하여 데이터 볼륨 문제를 분석하고 해결합니다.

14.5.1. 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 **Types**이 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 **Events**가 포함되는 것은 아닙니다.

Status, Reason 또는 **Message**가 변경되면 이벤트가 생성됩니다. 조건과 이벤트는 모두 데이터 볼륨의 상태 변화에 반응합니다.

예를 들어 가져오기 작업 중에 **URL**을 잘못 입력하면 가져오기 작업에서 **404** 메시지를 생성합니다. 이러한 메시지 변경으로 인해 원인이 포함된 이벤트가 생성됩니다. **Conditions** 섹션의 출력도 업데이트됩니다.

14.5.2. 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 **PVC**가 표시됩니다.

  Type은 **Bound**이므로 **Status**가 **True**입니다. **PVC**가 바인딩되지 않은 경우 **Status**는 **False**입니다.

  PVC가 바인딩되면 **PVC**가 바인딩되었음을 알리는 이벤트가 생성됩니다. 이 경우 **Reason**은 **Bound**이고 **Status**는 **True**입니다. **Message**는 데이터 볼륨이 속한 **PVC**를 나타냅니다.

  Events 섹션의 **Message**에서는 **PVC**가 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 – Type이 **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
```

14.6. 가상 머신 워크로드에 대한 정보 보기

OpenShift Container Platform 웹 콘솔의 가상 머신 대시보드를 사용하여 가상 머신에 대한 개괄적인 정보를 볼 수 있습니다.

14.6.1. 가상 머신 대시보드

Virtualization → **VirtualMachines** 페이지로 이동한 후 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
  - 상태
  - 작성일

- 운영 체제
- **CPU** 및 메모리
- 호스트 이름
- 템플릿

VM이 실행 중인 경우 활성 **VNC** 프리뷰 창과 **VNC** 웹 콘솔을 여는 링크가 있습니다. 세부 정보 카드의 옵션 메뉴 ⋮ 는 **VM**을 중지하거나 일시 중지하고 **SSH** 터널링을 위해 **nodeport** 명령을 통해 **ssh** 를 복사하는 옵션을 제공합니다.

- 경고 는 세 가지 심각도 수준이 있는 **VM** 경고를 나열합니다.
  - 심각
  - 경고
  - 정보
- 스냅샷 은 **VM** 스냅샷에 대한 정보와 스냅샷을 생성하는 기능을 제공합니다. 나열된 각 스냅샷에 대해 **Snapshots** 카드에 다음이 포함됩니다.
  - 스냅샷 상태가 성공적으로 생성되었거나 아직 진행 중이거나 실패한 경우 시각적 지표가 표시됩니다.
  - 스냅샷을 복원하거나 삭제하는 옵션이 있는 옵션 메뉴 ⋮

- 네트워크 인터페이스 는 다음을 포함하여 **VM**의 네트워크 인터페이스에 대한 정보를 제공합니다.
  - 이름(네트워크 및 유형)
  - **IP** 주소를 클립보드에 복사할 수 있는 **IP** 주소
- 디스크에 는 다음을 포함한 **VM** 디스크 세부 정보가 나열됩니다.
  - 이름
  - 드라이브
  - 크기
- 사용률에는 다음에 대한 사용 데이터가 표시되는 차트가 포함됩니다.
  - **CPU**
  - 메모리
  - 스토리지
  - 네트워크 전송

참고

드롭다운 목록을 사용하여 사용률 데이터의 기간을 선택합니다. 사용 가능한 옵션은 **5**분,**1**시간,**6**시간 및 **24**시간 입니다.

- 하드웨어 장치는 다음을 포함하여 **GPU** 및 호스트 장치에 대한 정보를 제공합니다.
  - 리소스 이름
  - 하드웨어 장치 이름

14.7. 가상 머신 상태 모니터링

VMI(가상 머신 인스턴스**)**는 연결 해제**,** 교착 상태 또는 외부 종속성 문제와 같은 일시적인 문제로 인해 **VMI**가 비정상 상태가 될 수 있습니다. 상태 점검은 준비 및 활성 프로브의 조합을 사용하여 **VMI**에서 정기적으로 진단을 수행합니다.

14.7.1. 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 **VMI**에 도달하지 않고 **VMI**가 응답하지 않을 때 새 인스턴스가 생성되도록 할 수 있습니다.

준비 상태 프로브는 **VMI**가 서비스 요청을 수락할 준비가 되었는지 확인합니다. 프로브가 실패하면 **VMI**가 준비될 때까지 **VMI**가 사용 가능한 엔드포인트 목록에서 **VMI**가 제거됩니다.

활성 프로브는 **VMI**의 응답 여부를 결정합니다. 프로브가 실패하면 **VMI**가 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 **VMI**의 상태를 결정합니다. **HTTP** 응답 코드가 **200**에서 **399** 사이인 경우 테스트에 성공합니다. **HTTP GET** 테스트는 **HTTP** 상태 코드를 완전히 초기화할 때 반환하는 애

플리케이션에 사용할 수 있습니다.

TCP 소켓

프로브는 **VMI**에 대한 소켓을 열려고 시도합니다. **VMI**는 프로브에서 연결을 설정할 수 있는 경우에만 정상으로 간주됩니다. **TCP** 소켓 테스트는 초기화가 완료된 후 수신 대기를 시작하는 애플리케이션에 사용할 수 있습니다.

게스트 에이전트 **ping**

프로브는 **guest-ping** 명령을 사용하여 **QEMU** 게스트 에이전트가 가상 머신에서 실행 중인지 확인합니다.

14.7.2. 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)

2. 프로브에서 쿼리하는 **VMI**의 포트입니다. 위의 예에서 프로브는 포트 **1500**을 쿼리합니다.

3. **HTTP** 서버에서 액세스할 경로입니다. 위의 예에서 서버의 **/healthz** 경로에 대한 핸들러가 성공 코드를 반환하면 **VMI**가 정상으로 간주됩니다. 핸들러에서 실패 코드를 반환하면 **VMI**가 사용 가능한 엔드포인트 목록에서 제거됩니다.

4. 준비 프로브가 시작되기 전에 **VMI**가 시작된 후 시간(초)입니다.

5. 프로브 수행 사이의 지연 시간(초)입니다. 기본 지연 시간은 **10**초입니다. 이 값은 **timeoutSeconds** 보다 커야 합니다.

6. 프로브가 시간 초과되고 **VMI**가 실패한 것으로 간주되는 비활성 시간(초)입니다. 기본값은 **1**입니다. 이 값은 **periodSeconds** 보다 작아야 합니다.

7. 프로브가 실패할 수 있는 횟수입니다. 기본값은 **3**입니다. 지정된 횟수의 시도 후 **Pod**가 **Unready**로 표시됩니다.

8. 프로브에서 실패 후 성공한 것으로 간주하기 위해 성공으로 보고해야 하는 횟수입니다. 기본값은 **1**입니다.

2. 다음 명령을 실행하여 **VMI**를 생성합니다.

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

14.7.3. 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 **VMI**가 시작된 후 시간(초)입니다.

(2) 프로브 수행 사이의 지연 시간(초)입니다. 기본 지연 시간은 **10**초입니다. 이 값은 **timeoutSeconds** 보다 커야 합니다.

(3) 수행할 **TCP** 작업입니다.

(4) 프로브에서 쿼리하는 **VMI**의 포트입니다.

(5) 프로브가 시간 초과되고 **VMI**가 실패한 것으로 간주되는 비활성 시간(초)입니다. 기본 값은 **1**입니다. 이 값은 **periodSeconds** 보다 작아야 합니다.

2. 다음 명령을 실행하여 **VMI**를 생성합니다.

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

14.7.4. 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 **VMI**가 시작된 후 시간(초)입니다.

(2) 프로브 수행 사이의 지연 시간(초)입니다. 기본 지연 시간은 **10**초입니다. 이 값은 **timeoutSeconds** 보다 커야 합니다.

3. **VMI** 연결에 수행할 **HTTP GET** 요청입니다.

4. 프로브에서 쿼리하는 **VMI**의 포트입니다. 위의 예에서 프로브는 포트 **1500**을 쿼리합니다. **VMI**는 **cloud-init**를 통해 포트 **1500**에 최소 **HTTP** 서버를 설치하고 실행합니다.

5. **HTTP** 서버에서 액세스할 경로입니다. 위의 예에서 서버의 **/healthz** 경로에 대한 핸들러가 성공 코드를 반환하면 **VMI**가 정상으로 간주됩니다. 핸들러에서 실패 코드를 반환하면 **VMI**가 삭제되고 새 인스턴스가 생성됩니다.

6. 프로브가 시간 초과되고 **VMI**가 실패한 것으로 간주되는 비활성 시간(초)입니다. 기본값은 **1**입니다. 이 값은 **periodSeconds** 보다 작아야 합니다.

2. 다음 명령을 실행하여 **VMI**를 생성합니다.

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

14.7.5. 게스트 에이전트 ping 프로브 정의

VMI(가상 머신 인스턴스) 구성의 **spec.readinessProbe.guestAgentPing** 필드를 설정하여 게스트 에이전트 **ping** 프로브를 정의합니다.

중요

게스트 에이전트 **ping** 프로브는 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

- **QEMU** 게스트 에이전트를 가상 머신에 설치하고 활성화해야 합니다.

절차

1. **VMI** 구성 파일에 게스트 에이전트 **ping** 프로브에 대한 세부 정보를 포함합니다. 예를 들면 다음과 같습니다.

게스트 에이전트 **ping** 프로브 샘플

```
# ...
spec:
  readinessProbe:
    guestAgentPing: {} (1)
    initialDelaySeconds: 120 (2)
    periodSeconds: 20 (3)
    timeoutSeconds: 10 (4)
    failureThreshold: 3 (5)
    successThreshold: 3 (6)
# ...
```

(1) **VMI**에 연결하는 게스트 에이전트 **ping** 프로브입니다.

(2) 선택 사항: 게스트 에이전트 프로브를 시작하기 전에 **VMI**가 시작된 후의 시간(초)입니다.

(3) 선택 사항: 프로브 수행 사이의 지연 시간(초)입니다. 기본 지연 시간은 **10**초입니다. 이 값은 **timeoutSeconds** 보다 커야 합니다.

(4) 선택 사항: 프로브가 타임아웃되고 **VMI**가 실패한 것으로 간주되는 비활성 시간(초)입니다. 기본값은 **1**입니다. 이 값은 **periodSeconds** 보다 작아야 합니다.

(5)

6 선택 사항: 실패 후 성공으로 간주하려면 프로브에서 성공을 보고해야 하는 횟수입니다. 기본값은 **1**입니다.

2. 다음 명령을 실행하여 **VMI**를 생성합니다.

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

14.7.6. 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
```

14.7.7. 추가 리소스

- 상태 점검을 사용하여 애플리케이션 상태 모니터링

14.8. 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.

14.8.1. OpenShift Container Platform 대시 보드 페이지 정보

OpenShift Container Platform 대시보드에 액세스합니다. 이 대시보드는 **OpenShift Container Platform** 웹 콘솔에서 홈 → 개요 로 이동하여 클러스터에 대한 고급 정보를 캡처합니다.

OpenShift Container Platform 대시 보드는 별도의 대시 보드 카드에 표시되는 다양한 클러스터 정보를 제공합니다.

OpenShift Container Platform 대시 보드는 다음 카드로 구성됩니다.

- **Details**는 클러스터 정보에 대한 간략한 개요를 표시합니다.

  상태에는 **ok, error, warning, progress** 및 **unknown**이 포함되어 있습니다. 리소스는 사용자 정의 상태 이름을 추가 할 수 있습니다.

- 클러스터 **ID**
- 공급자
- 버전

- **Cluster Inventory**는 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 Virtualization**이 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 Virtualization**이 설치된 경우 **OpenShift Virtualization**의 전반적인 상태도 진단됩니다. 하위 시스템이 한 개 이상 있는 경우 모두 보기를 클릭하여 각 하위 시스템의 상태를 확인하십시오.
  - 상태에 따라 나열된 클러스터의 베어 메탈 호스트 (**metal3** 환경에서만 사용 가능)

- **Status** 는 관리자가 클러스터 리소스를 사용하는 방법을 이해하는 데 도움이 됩니다. 리소스를 클릭하면 지정된 클러스터 리소스(**CPU,** 메모리 또는 스토리지)를 가장 많이 사용하는 **Pod**와 노드가 나열된 세부 정보 페이지로 이동합니다.

- 클러스터 사용률 은 관리자가 다음과 같은 정보를 포함하여 높은 리소스 소비의 규모와 빈도를 이해하는 데 도움이 되도록 지정된 기간 동안 다양한 리소스의 용량을 표시합니다.

- **CPU** 시간
- 메모리 할당
- 소비된 스토리지
- 소비된 네트워크 리소스
- **Pod** 수

- 활동에는 **Pod** 생성 또는 다른 호스트로의 가상 머신 마이그레이션과 같은 클러스터의 최근 활동과 관련된 메시지가 나열됩니다.

14.9. 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 **pod**의 리소스 사용에 대한 데이터를 제공합니다. **OpenShift Virtualization** 대시보드에 표시되는 시각화 지표는 **PromQL(Prometheus Query Language)** 쿼리를 기반으로 합니다.

OpenShift Virtualization 대시보드에서 사용자 정의 네임스페이스를 모니터링하려면 모니터링 역할이 필요합니다.

웹 콘솔의 **VirtualMachine** 세부 정보 페이지 → 메트릭 탭에서 특정 가상 머신의 리소스 사용량을 볼 수 있습니다.

14.9.1. 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**입

니다.

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 **5**개의 소비자로 제한됩니다.

14.9.2. 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.

14.9.3. 추가 리소스

- 모니터링 개요
- 모니터링 대시보드 검토

14.10. OPENSHIFT CONTAINER PLATFORM 클러스터 모니터링, 로깅, TELEMETRY

OpenShift Container Platform은 클러스터 수준에서 모니터링에 필요한 다양한 리소스를 제공합니다.

14.10.1. OpenShift Container Platform 모니터링 정보

OpenShift Container Platform에는 핵심 플랫폼 구성 요소를 모니터링할 수 있는 사전 구성, 사전 설치 및 자체 업데이트 모니터링 스택이 포함되어 있습니다. **OpenShift Container Platform**은 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 4.12를 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 **Pod**를 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.

14.10.2. 로깅 아키텍처

로깅의 주요 구성 요소는 다음과 같습니다.

수집기

수집기는 데몬 세트로 각 **OpenShift Container Platform** 노드에 **Pod**를 배포합니다. 각 노드에서 로그 데이터를 수집하고 데이터를 변환한 다음 구성된 출력으로 전달합니다. **Vector** 수집기 또는 레거시 **Fluentd** 수집기를 사용할 수 있습니다.

참고

Fluentd는 더 이상 사용되지 않으며 향후 릴리스에서 제거될 예정입니다. **Red Hat**은 현재 릴리스 라이프사이클 동안 이 기능에 대한 버그 수정 및 지원을 제공하지만 이 기능은 더 이상 개선 사항을 받지 않습니다. **Fluentd** 대신 **Vector**를 사용할 수 있습니다.

로그 저장소

로그 저장소는 분석을 위해 로그 데이터를 저장하고 로그 전달자의 기본 출력입니다. 기본 **LokiStack** 로그 저장소, 레거시 **Elasticsearch** 로그 저장소를 사용하거나 로그를 추가 외부 로그 저장소로 전달할 수 있습니다.

참고

로깅 **5.9** 릴리스에는 업데이트된 **OpenShift Elasticsearch Operator** 버전이 포함되어 있지 않습니다. 현재 **Logging 5.8**과 함께 릴리스된 **OpenShift Elasticsearch Operator**를 사용하는 경우 로깅 **5.8**의 **EOL**까지 로깅에서 계속 작동합니다. **OpenShift Elasticsearch Operator**를 사용하여 기본 로그 스토리지를 관리하는 대신 **Loki Operator**를 사용할 수 있습니다. 로깅 라이프사이클 날짜에 대한 자세한 내용은 **Platform Agnostic Operators** 를 참조하십시오.

시각화

UI 구성 요소를 사용하여 로그 데이터의 시각적 표현을 볼 수 있습니다. **UI**는 저장된 로그를 검색, 쿼리 및 볼 수 있는 그래픽 인터페이스를 제공합니다. **OpenShift Container Platform** 웹 콘솔 **UI**는

OpenShift Container Platform 콘솔 플러그인을 활성화하면 제공됩니다.

참고

Kibana 웹 콘솔은 향후 로깅 릴리스에서 더 이상 사용되지 않습니다.

로깅은 컨테이너 로그 및 노드 로그를 수집합니다. 이러한 유형은 다음과 같이 분류됩니다.

애플리케이션 로그

인프라 컨테이너 애플리케이션을 제외하고 클러스터에서 실행 중인 사용자 애플리케이션에 의해 생성된 컨테이너 로그입니다.

인프라 로그

인프라 네임스페이스에서 생성된 컨테이너 로그: **openshift***, **kube*** 또는 **default** 및 노드의 **journald** 메시지입니다.

감사 로그

/var/log/audit/audit.log 파일에 저장되는 노드 감사 시스템인 **auditd**에서 생성된 로그와 **auditd,kube-apiserver,openshift-apiserver** 서비스 및 활성화된 경우 **ovn** 프로젝트의 로그입니다.

OpenShift 로깅에 대한 자세한 내용은 **OpenShift Logging** 설명서를 참조하십시오.

14.10.3. Telemetry 정보

Telemetry는 엄선된 클러스터 모니터링 지표의 일부를 **Red Hat**으로 보냅니다. **Telemeter Client**는 **4**분 **30**초마다 메트릭 값을 가져와 **Red Hat**에 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 Hat**은 **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 Hat**은 연결된 모든 클러스터 정보를 사용하여 **OpenShift Container Platform**을 개선하고 사용 편의성을 높입니다.

14.10.3.1. Telemetry에서 수집하는 정보

Telemetry에서 수집되는 정보는 다음과 같습니다.

14.10.3.1.1. 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 **Operator**에 대해 "관련 개체"로 나열된 모든 네임스페이스에 대한 이벤트
- 성능 저하 소프트웨어에 대한 정보
- 인증서의 유효성에 대한 정보
- **OpenShift Container Platform**이 배포된 공급자 플랫폼의 이름 및 데이터 센터 위치

14.10.3.1.2. 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

14.10.3.1.3. 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 Hat**은 개인 정보를 수집하지 않습니다. 개인 정보가 의도하지 않게 **Red Hat**에 수신된 경우 **Red Hat**은 이러한 정보를 삭제합니다. **Telemetry** 데이터가 개인 정보를 구성하는 범위까지, **Red Hat**의 개인정보 보호정책에 대한 자세한 내용은 **Red Hat** 개인정보처리방침을 참조하십시오.

14.10.4. CLI 문제 해결 및 디버깅 명령

oc 클라이언트 문제 해결 및 디버깅 명령 목록은 **OpenShift Container Platform CLI** 툴 설명서를 참조하십시오.

14.11. 클러스터 검사 실행

OpenShift Virtualization에는 클러스터 유지 관리 및 문제 해결에 사용할 수 있는 사전 정의된 점검이 포함되어 있습니다.

중요

OpenShift Container Platform 클러스터 검사 프레임워크는 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.

14.11.1. OpenShift Container Platform 클러스터 검사 프레임워크 정보

점검 은 특정 클러스터 기능이 예상대로 작동하는지 확인할 수 있는 자동화된 테스트 워크로드입니다. 클러스터 점검 프레임워크는 네이티브 **Kubernetes** 리소스를 사용하여 검사를 구성하고 실행합니다.

클러스터 관리자와 개발자는 사전 정의된 점검을 사용하여 클러스터 유지 관리를 개선하고, 예기치 않은 동작을 해결하고, 오류를 최소화하며, 시간을 절약할 수 있습니다. 또한 수표의 결과를 검토하고 추가 분석을 위해 전문가와 공유할 수 있습니다. 벤더는 제공하는 기능 또는 서비스에 대한 점검 확인을 작성하고 게시하여 고객 환경이 올바르게 구성되었는지 확인할 수 있습니다.

기존 네임스페이스에서 사전 정의된 점검을 실행하려면 점검에 대한 서비스 계정을 설정하고, 서비스 계정에 대한 **Role** 및 **RoleBinding** 오브젝트를 생성하며, 검사에 대한 권한을 활성화하며, 입력 구성 맵 및 검사 작업을 생성해야 합니다. 검사를 여러 번 실행할 수 있습니다.

중요

항상 다음을 수행해야 합니다.

- 이미지를 적용하기 전에 이미지 점검 소스의 출처가 있는지 확인합니다.
- **Role** 및 **RoleBinding** 오브젝트를 생성하기 전에 점검 권한을 검토합니다.

14.11.2. 보조 네트워크에서 가상 머신의 네트워크 연결 및 대기 시간 확인

사전 정의된 점검을 사용하여 네트워크 연결을 확인하고 보조 네트워크 인터페이스에 연결된 두 가상 머신**(VM)** 간의 대기 시간을 측정합니다.

처음으로 검사를 실행하려면 절차의 단계를 따르십시오.

이전에 검사를 실행한 경우 프레임워크를 설치하고 점검에 대한 권한을 활성화하는 단계가 필요하기 때문에 **5**단계의 절차로 건너뜁니다.

사전 요구 사항

- **OpenShift CLI(oc)**를 설치합니다.
- 클러스터에는 두 개 이상의 작업자 노드가 있습니다.
- **Multus CNI(Container Network Interface)** 플러그인이 클러스터에 설치되어 있습니다.
- 네임스페이스에 대한 네트워크 연결 정의를 구성하셨습니다.

절차

1. 검사에 클러스터 액세스에 필요한 권한이 있는 **ServiceAccount,Role, RoleBinding** 오브젝트가 포함된 매니페스트 파일을 생성합니다.

예 **14.3.** 역할 매니페스트 파일 예

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: vm-latency-checkup-sa
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: kubevirt-vm-latency-checker
rules:
- apiGroups: ["kubevirt.io"]
  resources: ["virtualmachineinstances"]
  verbs: ["get", "create", "delete"]
- apiGroups: ["subresources.kubevirt.io"]
  resources: ["virtualmachineinstances/console"]
  verbs: ["get"]
- apiGroups: ["k8s.cni.cncf.io"]
  resources: ["network-attachment-definitions"]
  verbs: ["get"]
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: kubevirt-vm-latency-checker
subjects:
- kind: ServiceAccount
  name: vm-latency-checkup-sa
roleRef:
  kind: Role
  name: kubevirt-vm-latency-checker
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: kiagnose-configmap-access
rules:
- apiGroups: [ "" ]
  resources: [ "configmaps" ]
  verbs: ["get", "update"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: kiagnose-configmap-access
subjects:
- kind: ServiceAccount
  name: vm-latency-checkup-sa
roleRef:
  kind: Role
  name: kiagnose-configmap-access
  apiGroup: rbac.authorization.k8s.io
```

2. 검사 역할 매니페스트를 적용합니다.

```
$ oc apply -n <target_namespace> -f <latency_roles>.yaml (1)
```

(1) **<target_namespace >**는 점검을 실행할 네임스페이스입니다. **NetworkAttachmentDefinition** 오브젝트가 있는 기존 네임스페이스여야 합니다.

3. 확인에 대한 입력 매개변수가 포함된 **ConfigMap** 매니페스트를 생성합니다. 구성 맵은 검사를 실행하고 점검 결과를 저장할 프레임워크에 대한 입력을 제공합니다.

입력 구성 맵의 예

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kubevirt-vm-latency-checkup-config
data:
  spec.timeout: 5m
  spec.param.network_attachment_definition_namespace: <target_namespace>
  spec.param.network_attachment_definition_name: "blue-network" (1)
  spec.param.max_desired_latency_milliseconds: "10" (2)
  spec.param.sample_duration_seconds: "5" (3)
  spec.param.source_node: "worker1" (4)
  spec.param.target_node: "worker2" (5)
```

1. **NetworkAttachmentDefinition** 오브젝트의 이름입니다.

2. 선택 사항: 가상 머신 간에 필요한 최대 대기 시간(밀리초)입니다. 측정된 대기 시간이 이 값을 초과하면 검사에 실패합니다.

3. 선택 사항: 대기 시간 점검 기간(초)입니다.

4. 선택 사항: 지정된 경우 대기 시간이 이 노드에서 대상 노드로 측정됩니다. 소스 노드를 지정하면 **spec.param.target_node** 필드가 비어 있을 수 없습니다.

5. 선택 사항: 지정된 경우 소스 노드에서 이 노드로 대기 시간이 측정됩니다.

4. 대상 네임스페이스에 구성 맵 매니페스트를 적용합니다.

```
$ oc apply -n <target_namespace> -f <latency_config_map>.yaml
```

5. 검사를 실행할 **Job** 오브젝트를 생성합니다.

작업 매니페스트 예

```
apiVersion: batch/v1
kind: Job
metadata:
  name: kubevirt-vm-latency-checkup
spec:
  backoffLimit: 0
  template:
    spec:
      serviceAccountName: vm-latency-checkup-sa
      restartPolicy: Never
      containers:
        - name: vm-latency-checkup
          image: registry.redhat.io/container-native-virtualization/vm-network-latency-
checkup:v4.12.0
          securityContext:
            allowPrivilegeEscalation: false
            capabilities:
              drop: ["ALL"]
            runAsNonRoot: true
            seccompProfile:
              type: "RuntimeDefault"
          env:
            - name: CONFIGMAP_NAMESPACE
              value: <target_namespace>
            - name: CONFIGMAP_NAME
              value: kubevirt-vm-latency-checkup-config
```

6. 작업 매니페스트를 적용합니다. 점검에서는 **ping** 유틸리티를 사용하여 연결을 확인하고 대기 시간을 측정합니다.

```
$ oc apply -n <target_namespace> -f <latency_job>.yaml
```

7. 작업이 완료될 때까지 기다립니다.

```
$ oc wait job kubevirt-vm-latency-checkup -n <target_namespace> --for
condition=complete --timeout 6m
```

8.

다음 명령을 실행하여 대기 시간 점검 결과를 검토합니다. 최대 측정 대기 시간이 **spec.param.max_desired_latency_milliseconds** 특성 값보다 크면 검사에 실패하고 오류를 반환합니다.

```
$ oc get configmap kubevirt-vm-latency-checkup-config -n <target_namespace> -o
yaml
```

출력 구성 맵(**success**)의 예

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kubevirt-vm-latency-checkup-config
  namespace: <target_namespace>
data:
  spec.timeout: 5m
  spec.param.network_attachment_definition_namespace: <target_namespace>
  spec.param.network_attachment_definition_name: "blue-network"
  spec.param.max_desired_latency_milliseconds: "10"
  spec.param.sample_duration_seconds: "5"
  spec.param.source_node: "worker1"
  spec.param.target_node: "worker2"
  status.succeeded: "true"
  status.failureReason: ""
  status.completionTimestamp: "2022-01-01T09:00:00Z"
  status.startTimestamp: "2022-01-01T09:00:07Z"
  status.result.avgLatencyNanoSec: "177000"
  status.result.maxLatencyNanoSec: "244000" (1)
  status.result.measurementDurationSec: "5"
  status.result.minLatencyNanoSec: "135000"
  status.result.sourceNode: "worker1"
  status.result.targetNode: "worker2"
```

(1) 나노초 단위의 최대 측정 대기 시간입니다.

9. 선택 사항: 검사 실패 시 자세한 작업 로그를 보려면 다음 명령을 사용합니다.

```
$ oc logs job.batch/kubevirt-vm-latency-checkup -n <target_namespace>
```

10. 다음 명령을 실행하여 이전에 생성한 작업 및 구성 맵 리소스를 삭제합니다.

```
$ oc delete job -n <target_namespace> kubevirt-vm-latency-checkup
```

```
$ oc delete config-map -n <target_namespace> kubevirt-vm-latency-checkup-config
```

11. 선택 사항: 다른 점검을 실행하지 않는 경우 점검 역할 및 프레임워크 매니페스트 파일을 삭제합니다.

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

14.11.3. 추가 리소스

- 여러 네트워크에 가상 머신 연결

14.12. 가상 리소스에 대한 PROMETHEUS 쿼리

OpenShift Virtualization에서는 vCPU, 네트워크, 스토리지, 게스트 메모리 스왑을 포함하여 클러스터 인프라 리소스의 사용을 모니터링하는 데 사용할 수 있는 메트릭을 제공합니다. 메트릭을 사용하여 실시간 마이그레이션 상태를 쿼리할 수도 있습니다.

OpenShift Container Platform 모니터링 대시보드를 사용하여 가상화 메트릭을 쿼리합니다.

14.12.1. 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 활성화되고 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.

14.12.2. 메트릭 쿼리 정보

OpenShift Container Platform 모니터링 대시보드를 사용하면 Pacemaker에서 표시되는 메트릭을 검사하기 위해 Prometheus Query Language(PromQL) 쿼리를 실행할 수 있습니다. 이 기능을 사용하

면 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.

14.12.2.1. 클러스터 관리자로서 모든 프로젝트의 메트릭 쿼리

클러스터 관리자 또는 모든 프로젝트에 대한 보기 권한이 있는 사용자로 **Metrics UI**에서 모든 기본 **OpenShift Container Platform** 및 사용자 정의 프로젝트에 대한 메트릭에 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 화면에서 모니터링 → 메트릭 으로 이동합니다.
2. 하나 이상의 쿼리를 추가하려면 다음 작업을 수행합니다.

옵션	설명

옵션	설명
사용자 지정 쿼리를 생성합니다.	**표현식** 필드에 PromQL(Prometheus Query Language) 쿼리를 추가합니다. PromQL 표현식을 입력하면 목록에 자동 완성 제안 사항이 표시됩니다. 이러한 제안에는 함수, 메트릭, 라벨, 시간 토큰이 있습니다. 키보드 화살표를 사용하여 이러한 제안된 항목 중 하나를 선택한 다음 Enter를 눌러 식에 항목을 추가할 수 있습니다.You can use the keyboard arrows to select one of these suggested items and then press Enter to add the item to your expression. 또한 제안된 항목에 마우스 포인터를 이동하여 해당 항목에 대한 간략한 설명을 볼 수도 있습니다.
여러 쿼리를 추가합니다.	**쿼리 추가를** 클릭합니다.
기존 쿼리를 복제합니다.	쿼리 옆에 있는 옵션 메뉴 ⋮ 를 클릭하고 **중복 쿼리** 를 선택합니다.
쿼리를 삭제합니다.	쿼리 옆에 있는 옵션 메뉴 ⋮ 를 클릭하고 **쿼리 삭제** 를 선택합니다.
쿼리가 실행되지 않도록 비활성화합니다.	쿼리 옆에 있는 옵션 메뉴 ⋮ 를 클릭하고 **쿼리 비활성화** 를 선택합니다.

3. 생성한 쿼리를 실행하려면 쿼리 실행을 클릭합니다. 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 숨기기 를 클릭하고 메트릭 테이블을 사용하여 쿼리를 조정합니다. 가능한 쿼리를 찾은 후 플롯을 활성화하여 그래프를 그립니다.

4. 선택 사항: 이제 페이지 **URL**에 실행한 쿼리가 포함되어 있습니다. 나중에 이 쿼리 세트를 다시 사용하려면 이 **URL**을 저장합니다.

14.12.2.2. 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 UI**에는 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 웹 콘솔의 사용자 정의 프로젝트에 대해 한 번에 한 프로젝트의 메트릭 만 쿼리할 수 있습니다**.**

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 콘솔에서 개발자 화면을 선택합니다**.**
2. **Observe → Metrics** 를 선택합니다**.**
3. **Project:** 목록에서 메트릭을 보려는 프로젝트를 선택합니다**.**
4.

쿼리 선택 목록에서 쿼리를 선택하거나 **PromQL** 표시를 선택하여 선택한 쿼리에 따라 사용자 정의 **PromQL** 쿼리를 만듭니다.

5. 선택 사항: 쿼리 선택 목록에서 사용자 지정 쿼리를 선택하여 새 쿼리를 입력합니다. 입력하는 경우 드롭다운 목록에 자동 완성 제안이 표시됩니다. 이러한 제안에는 함수 및 메트릭이 포함됩니다. 제안된 항목을 클릭하여 선택합니다.

참고

개발자 관점에서는 한 번에 하나의 쿼리만 실행할 수 있습니다.

14.12.3. 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.

14.12.3.1. 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. 유형:

'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 **6**분 동안 매 순간에 **I/O**를 기다리는 상위 **3**개의 **VM**을 반환합니다.

14.12.3.2. 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. 유형:

kubevirt_vmi_network_transmit_bytes_total

가상 시스템의 네트워크에서 전송된 총 트래픽(바이트 단위)을 반환합니다. 유형:

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 **6**분 동안 매번 가장 많은 네트워크 트래픽을 전송하는 상위 **3** 개의 **VM**을 반환합니다.

14.12.3.3. 스토리지 메트릭

14.12.3.3.1. 스토리지 관련 트래픽

다음 쿼리는 대량의 데이터를 작성하는 **VM**을 식별할 수 있습니다.

kubevirt_vmi_storage_read_traffic_bytes_total

가상 머신의 스토리지 관련 트래픽의 총 양(바이트)을 반환합니다. 유형:

kubevirt_vmi_storage_write_traffic_bytes_total

가상 시스템의 스토리지 관련 트래픽의 총 스토리지 쓰기(바이트)를 반환합니다. 유형:

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 **6**분 동안 매번 가장 많은 스토리지 트래픽을 수행하는 상위 **3** 개의 **VM**을 반환합니다.

14.12.3.3.2. 스토리지 스냅샷 데이터

kubevirt_vmsnapshot_disks_restored_from_source_total

소스 가상 머신에서 복원된 총 가상 머신 디스크 수를 반환합니다. 유형: 게이지.

kubevirt_vmsnapshot_disks_restored_from_source_bytes

소스 가상 시스템에서 복원된 공간(바이트)을 반환합니다. 유형: 게이지.

스토리지 스냅샷 데이터 쿼리의 예

```
kubevirt_vmsnapshot_disks_restored_from_source_total{vm_name="simple-vm",
vm_namespace="default"} (1)
```

1

이 쿼리는 소스 가상 머신에서 복원된 총 가상 머신 디스크 수를 반환합니다.

```
kubevirt_vmsnapshot_disks_restored_from_source_bytes{vm_name="simple-vm",
vm_namespace="default"} 1
```

1

이 쿼리는 소스 가상 시스템에서 복원된 공간(바이트)을 반환합니다.

14.12.3.3.3. 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. 유형:

kubevirt_vmi_storage_iops_write_total

가상 시스템이 초당 수행하는 읽기 I/O 작업의 양을 반환합니다. 유형:

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.

14.12.3.4. 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. 유형: 게이지.

kubevirt_vmi_memory_swap_out_traffic_bytes_total

가상 게스트가 스왑 아웃하는 메모리의 총 양(바이트 단위)을 반환합니다. 유형: 게이지.

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.

14.12.4. 실시간 마이그레이션 메트릭

실시간 마이그레이션 상태를 표시하려면 다음 메트릭을 쿼리할 수 있습니다.

kubevirt_migrate_vmi_data_processed_bytes

새 VM(가상 머신)으로 마이그레이션된 게스트 운영 체제(OS) 데이터의 양입니다. 유형: 게이지.

kubevirt_migrate_vmi_data_remaining_bytes

마이그레이션 중인 게스트 **OS** 데이터의 양입니다. 유형: 게이지.

kubevirt_migrate_vmi_dirty_memory_rate_bytes

게스트 **OS**에서 메모리가 더러워지는 속도입니다. 더러움 메모리는 변경되었지만 아직 디스크에 기록되지 않은 데이터입니다. 유형: 게이지.

kubevirt_migrate_vmi_pending_count

보류 중인 마이그레이션 수입니다. 유형: 게이지.

kubevirt_migrate_vmi_scheduling_count

스케줄링 마이그레이션 수입니다. 유형: 게이지.

kubevirt_migrate_vmi_running_count

실행 중인 마이그레이션 수입니다. 유형: 게이지.

kubevirt_migrate_vmi_succeeded

성공적으로 완료된 마이그레이션 수입니다. 유형: 게이지.

kubevirt_migrate_vmi_failed

실패한 마이그레이션 수입니다. 유형: 게이지.

14.12.5. 추가 리소스

- 모니터링 개요
- **Prometheus** 쿼리
- **Prometheus** 쿼리 예

14.13. 가상 머신의 사용자 정의 메트릭 노출

OpenShift Container Platform에는 핵심 플랫폼 구성 요소에 대한 모니터링을 제공하는 사전 구성된 사전 설치된 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.

14.13.1. 노드 내보내기 서비스 구성

node-exporter 에이전트는 메트릭을 수집하려는 클러스터의 모든 가상 머신에 배포됩니다. 가상 머신과 연결된 내부 지표 및 프로세스를 노출하도록 **node-exporter** 에이전트를 서비스로 구성합니다.

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 예제에서는 파일을 **node-exporter-service.yaml** 이라고 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 머신의 지표를 표시하는 **node-exporter** 서비스입니다.

2. 서비스가 생성되는 네임스페이스입니다.

3. 서비스의 레이블입니다. **ServiceMonitor** 는 이 라벨을 사용하여 이 서비스와 일치시킵니다.

4. **ClusterIP** 서비스의 포트 **9100**에서 메트릭을 노출하는 포트에 지정된 이름입니다.

5. **node-exporter-service** 가 요청을 수신 대기하는 데 사용하는 대상 포트입니다.

6. **monitor** 레이블로 구성된 가상 시스템의 **TCP** 포트 번호입니다.

7. 가상 머신의 **Pod**와 일치하는 데 사용되는 레이블입니다. 이 예에서는 라벨이 **monitor** 인 가상 머신의 **Pod**와 값 지표 가 일치합니다.

2. **node-exporter** 서비스를 생성합니다.

```
$ oc create -f node-exporter-service.yaml
```

14.13.2. 노드 내보내기 서비스를 사용하여 가상 머신 구성

node-exporter 파일을 가상 머신에 다운로드합니다. 그런 다음 가상 머신이 부팅될 때 **node-exporter** 서비스를 실행하는 **systemd** 서비스를 생성합니다.

사전 요구 사항

- 구성 요소의 **Pod**는 **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** 파일의 버전에 적용되는 디렉터리 경로를 사용하여 가상 머신에 **node-exporter** 파일을 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 생성합니다. **/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
```

14.13.3. 가상 머신의 사용자 정의 모니터링 레이블 생성

단일 서비스에서 여러 가상 머신에 대한 쿼리를 활성화하려면 가상 머신의 **YAML** 파일에 사용자 지정 레이블을 추가합니다.

사전 요구 사항

- **OpenShift Container Platform CLI oc**를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 로그인합니다.
- 웹 콘솔에 액세스하여 중지한 후 가상 머신을 재시작합니다.

절차

1. 가상 머신 구성 파일의 **template** 사양을 편집합니다. 이 예에서 레이블 **monitor** 에는 값 메트릭이 있습니다.

```
spec:
  template:
    metadata:
      labels:
        monitor: metrics
```

2. 가상 머신을 중지하고 다시 시작하여 **monitor** 레이블에 지정된 레이블 이름으로 새 **Pod**를 생성합니다.

14.13.3.1. 메트릭에 대해 node-exporter 서비스 쿼리

지표는 **/metrics** 표준 이름 아래에 **HTTP** 서비스 끝점을 통해 가상 머신에 대해 노출됩니다. 메트릭을 쿼리할 때 **Prometheus**는 가상 머신에서 노출하는 지표 끝점에서 직접 메트릭을 스크랩하고 이러한 메트릭을 표시하도록 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
```

14.13.4. 노드 내보내기 서비스에 대한 ServiceMonitor 리소스 생성

Prometheus 클라이언트 라이브러리를 사용하고 **/metrics** 끝점에서 메트릭을 스크랩하여 **node-exporter** 서비스에서 노출하는 지표에 액세스하고 볼 수 있습니다. **ServiceMonitor CRD(**사용자 정의 리

소스 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) **ServiceMonitor** 의 이름입니다.

(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
```

14.13.4.1. 클러스터 외부에서 노드 내보내기 서비스 액세스

클러스터 외부에서 node-exporter 서비스에 액세스하고 노출된 메트릭을 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. 경로에 대한 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
```

14.13.5. 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

14.14. OPENSHIFT VIRTUALIZATION RUNBOOK

OpenShift Virtualization Operator용 **Runbook**은 **openshift/runbooks Git** 리포지토리에서 유지 관

리되며 **GitHub**에서 볼 수 있습니다. **OpenShift Virtualization** 경고를 트리거하는 문제를 진단하고 해결하려면 **runbooks**의 절차를 따르십시오.

OpenShift Virtualization 경고는 웹 콘솔의 가상화 → 개요 탭에 표시됩니다.

14.14.1. CDIDataImportCronOutdated

- **CDIDataImportCronOutdated** 경고의 실행북 을 확인합니다.

14.14.2. CDIDataVolumeUnusualRestartCount

- **CDIDataVolumeUnusualRestartCount** 경고의 실행북 을 확인합니다.

14.14.3. CDIDefaultStorageClassDegraded

- **CDIDefaultStorageClassDegraded** 경고 의 **runbook** 을 확인합니다.

14.14.4. CDIMultipleDefaultVirtStorageClasses

- **CDIMultipleDefaultVirtStorageClasses** 경고의 **runbook** 을 확인합니다.

14.14.5. CDINoDefaultStorageClass

- **CDINoDefaultStorageClass** 경고 의 **runbook** 을 확인합니다.

14.14.6. CDINotReady

- **CDINotReady** 경고 의 **runbook** 을 확인합니다.

14.14.7. CDIOperatorDown

- **CDIOperatorDown** 경고 의 **runbook** 을 확인합니다.

14.14.8. CDIStorageProfilesIncomplete

- **CDIStorageProfilesIncomplete** 경고의 **runbook** 을 확인합니다.

14.14.9. CnaoDown

- **CnaoDown** 경고 의 **runbook**을 봅니다.

14.14.10. CnaoNMstateMigration

- **CnaoNMstateMigration** 경고 의 실행북을 확인합니다.

14.14.11. HCOInstallationIncomplete

- **HCOInstallationIncomplete** 경고 의 실행북을 봅니다.

14.14.12. HPPNotReady

- **HPPNotReady** 경고 의 실행북 을 확인합니다.

14.14.13. HPPOperatorDown

- **HPPOperatorDown** 경고 의 실행북 을 확인합니다.

14.14.14. HPPSharingPoolPathWithOS

- **HPPSharingPoolPathWithOS** 경고의 **runbook** 을 확인합니다.

14.14.15. KubemacpoolDown

- **KubemacpoolDown** 경고 의 **runbook** 을 확인합니다.

14.14.16. KubeMacPoolDuplicateMacsFound

- **KubeMacPoolDuplicateMacsFound** 경고의 **runbook** 을 확인합니다.

14.14.17. KubeVirtComponentExceedsRequestedCPU

- **KubeVirtComponentExceedsRequestedCPU** 경고가 더 이상 사용되지 않습니다.

14.14.18. KubeVirtComponentExceedsRequestedMemory

- **KubeVirtComponentExceedsRequestedMemory** 경고가 더 이상 사용되지 않습니다.

14.14.19. KubeVirtCRModified

- **KubeVirtCR Cryostat** 경고 의 실행북 을 확인합니다.

14.14.20. KubeVirtDeprecatedAPIRequested

- **KubeVirtDeprecatedAPIRequested** 경고 의 실행북 을 확인합니다.

14.14.21. KubeVirtNoAvailableNodesToRunVMs

- **KubeVirtNoAvailableNodesToRunVMs** 경고의 **runbook** 을 확인합니다.

14.14.22. KubevirtVmHighMemoryUsage

- **KubevirtVmHighMemoryUsage** 경고의 **runbook** 을 확인합니다.

14.14.23. KubeVirtVMIExcessiveMigrations

- **KubeVirtVMIExcessiveMigrations** 경고의 **runbook** 을 확인합니다.

14.14.24. LowKVMNodesCount

- **LowKVMNodesCount** 경고 의 실행북 을 확인합니다.

14.14.25. LowReadyVirtControllersCount

- **LowReadyVirtControllersCount** 경고의 **runbook** 을 확인합니다.

14.14.26. LowReadyVirtOperatorsCount

- **LowReadyVirtOperatorsCount** 경고의 **runbook** 을 확인합니다.

14.14.27. LowVirtAPICount

- **LowVirtAPICount** 경고 의 실행북 을 확인합니다.

14.14.28. LowVirtControllersCount

- **LowVirtControllersCount** 경고의 실행북 을 확인합니다.

14.14.29. LowVirtOperatorCount

- **LowVirtOperatorCount** 경고 의 **runbook** 을 확인합니다.

14.14.30. NetworkAddonsConfigNotReady

- **NetworkAddonsConfigNotReady** 경고의 **runbook** 을 확인합니다.

14.14.31. NoLeadingVirtOperator

- **NoLeadingVirtOperator** 경고 의 **runbook** 을 확인합니다.

14.14.32. NoReadyVirtController

- **NoReadyVirtController** 경고 의 **runbook** 을 확인합니다.

14.14.33. NoReadyVirtOperator

- **NoReadyVirtOperator** 경고 의 **runbook**을 봅니다.

14.14.34. OrphanedVirtualMachineInstances

- **Orphaned CryostatInstances** 경고 의 **runbook** 을 확인합니다.

14.14.35. OutdatedVirtualMachineInstanceWorkloads

- **Outdated CryostatInstanceWorkloads** 경고의 **runbook** 을 확인합니다.

14.14.36. SingleStackIPv6Unsupported

- **SingleStackIPv6Unsupported** 경고의 실행북 을 확인합니다.

14.14.37. SSPCommonTemplatesModificationReverted

- **SSPCommonTemplatesModificationReverted** 경고의 실행북 을 확인합니다.

14.14.38. SSPDown

- **SSPDown** 경고 의 **runbook**을 봅니다.

14.14.39. SSPFailingToReconcile

- **SSPFailingToReconcile** 경고의 실행북 을 확인합니다.

14.14.40. SSPHighRateRejectedVms

- **SSPHighRateRejectedVms** 경고의 실행북 을 확인합니다.

14.14.41. SSPTemplateValidatorDown

- **SSPTemplateValidatorDown** 경고 의 **runbook** 을 확인합니다.

14.14.42. 지원되지 않는HCOModification

- **UnsupportedHCOModification** 경고에 대한 **runbook** 을 확인합니다.

14.14.43. VirtAPIDown

- **VirtAPIDown** 경고 의 실행북 을 확인합니다.

14.14.44. VirtApiRESTErrorsBurst

- **VirtApiRESTErrorsBurst** 경고 의 실행북 을 확인합니다.

14.14.45. VirtApiRESTErrorsHigh

- **VirtApiRESTErrorsHigh** 경고 의 실행북 을 확인합니다.

14.14.46. VirtControllerDown

- **VirtControllerDown** 경고 의 실행북 을 확인합니다.

14.14.47. VirtControllerRESTErrorsBurst

- **VirtControllerRESTErrorsBurst** 경고 의 실행북 을 확인합니다.

14.14.48. VirtControllerRESTErrorsHigh

- **VirtControllerRESTErrorsHigh** 경고 의 실행북 을 확인합니다.

14.14.49. VirtHandlerDaemonSetRolloutFailing

- **VirtHandlerDaemonSetRolloutFailing** 경고의 실행북 을 확인합니다.

14.14.50. VirtHandlerRESTErrorsBurst

- **VirtHandlerRESTErrorsBurst** 경고 의 실행북 을 확인합니다.

14.14.51. VirtHandlerRESTErrorsHigh

- **VirtHandlerRESTErrorsHigh** 경고 의 실행북 을 확인합니다.

14.14.52. VirtOperatorDown

- **VirtOperatorDown** 경고 의 실행북 을 확인합니다.

14.14.53. VirtOperatorRESTErrorsBurst

- **VirtOperatorRESTErrorsBurst** 경고 의 실행북 을 확인합니다.

14.14.54. VirtOperatorRESTErrorsHigh

- **VirtOperatorRESTErrorsHigh** 경고 의 실행북 을 확인합니다.

14.14.55. VirtualMachineCRCErrors

- 경고의 이름이 **VMStorageClassWarning** 으로 변경되었기 때문에 **VirtualMachineCRCErrors** 경고의 **runbook**이 더 이상 사용되지 않습니다.
  - **VMStorageClassWarning** 경고 의 실행북 을 확인합니다.

14.14.56. VMCannotBeEvicted

- **VMCannotBeEvicted** 경고의 실행북 을 확인합니다.

14.14.57. VMStorageClassWarning

- **VMStorageClassWarning** 경고 의 실행북 을 확인합니다.

14.15. RED HAT 지원을 위한 데이터 수집

Red Hat 지원에 지원 케이스 를 제출할 때 다음 툴을 사용하여 **OpenShift Container Platform** 및 **OpenShift Virtualization**에 대한 디버깅 정보를 제공하는 것이 좋습니다.

must-gather 툴

must-gather 툴은 리소스 정의 및 서비스 로그를 포함하여 진단 정보를 수집합니다.

Prometheus

Prometheus는 시계열 데이터베이스이며 메트릭에 대한 규칙 평가 엔진입니다. **Prometheus**는 처리를 위해 **Alertmanager**에 경고를 보냅니다.

Alertmanager

Alertmanager 서비스는 **Prometheus**에서 수신한 경고를 처리합니다. **Alertmanager**는 또한 외부 알림 시스템으로 경고를 보냅니다.

14.15.1. 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 **7**일로 설정합니다.
- 관련 경고를 캡처하고 클러스터 외부에서 보고 유지할 수 있도록 전용 메일로 전송하도록 **Alertmanager**를 구성합니다.
- 영향을 받는 노드 및 가상 머신의 정확한 수를 기록합니다.

절차

1. 기본 **must-gather** 이미지를 사용하여 클러스터의 **must-gather** 데이터를 수집합니다.
2. 필요한 경우 **Red Hat OpenShift Data Foundation**의 **must-gather** 데이터를 수집합니다.
3. **OpenShift Virtualization must-gather** 이미지를 사용하여 **OpenShift Virtualization**의 **must-gather** 데이터를 수집합니다.
4. 클러스터에 대한 **Prometheus** 지표를 수집합니다.

14.15.1.1. 추가 리소스

- **Prometheus** 메트릭 데이터의 보존 시간 구성
- 경고 알림을 외부 시스템에 보내도록 **Alertmanager** 구성
- **OpenShift Container Platform**에 대한 **must-gather** 데이터 수집
- **Red Hat OpenShift Data Foundation**의 **must-gather** 데이터 수집
- **OpenShift Virtualization**에 대한 **must-gather** 데이터 수집
- 클러스터 관리자로 모든 프로젝트 의 **Prometheus** 메트릭 수집

14.15.2. 가상 머신에 대한 데이터 수집

가상 머신 장애 복구**(VM)**에 대한 데이터를 수집하면 근본 원인을 분석하고 결정하는 데 필요한 시간이 최소화됩니다**.**

사전 요구 사항

- **Windows VM:**
  - **Red Hat** 지원에 대한 **Windows** 패치 업데이트 세부 정보를 기록합니다**.**
  - **VirtIO** 드라이버의 최신 버전을 설치합니다**. VirtIO** 드라이버에는 **QEMU** 게스트 에이전트가 포함되어 있습니다**.**
  - **RDP(Remote Desktop Protocol)**가 활성화된 경우**, RDP**를 사용하여 **VM**에 연결하여 연결 소프트웨어에 문제가 있는지 확인합니다**.

절차

1. **VM** 손상에 대한 자세한 **must-gather** 데이터를 수집합니다.

2. 재시작하기 전에 충돌한 **VM**의 스크린샷을 수집합니다.

3. **VM**이 손상되는 요인을 기록하십시오. 예를 들어 **VM**에는 동일한 호스트 또는 네트워크가 있습니다.

14.15.2.1. 추가 리소스

- **Windows VM**에 **VirtIO** 드라이버 설치
- 호스트 액세스없이 **Windows VM**에 **VirtIO** 드라이버 다운로드 및 설치
- 웹 콘솔 또는 명령줄을 사용하여 **RDP**를 사용하여 **Windows VM**에 연결
- 가상 머신에대한 **must-gather** 데이터 수집

14.15.3. 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 Virtualization**에 대한 데이터를 수집합니다.

```
$ oc adm must-gather --image-stream=openshift/must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.12.13
```

14.15.3.1. must-gather 툴 옵션

다음 옵션에 대한 스크립트 및 환경 변수 조합을 지정할 수 있습니다.

- 네임스페이스에서 자세한 **VM(**가상 머신**)** 정보 수집
- 지정된 **VM**에 대한 세부 정보 수집
- 이미지**,** 이미지 스트림 및 **image-stream-tags** 정보 수집
- **must-gather** 툴에서 사용하는 최대 병렬 프로세스 수 제한

14.15.3.1.1. 매개 변수

환경 변수

호환되는 스크립트의 환경 변수를 지정할 수 있습니다.

NS=<namespace_name>

지정하는 네임스페이스에서 **virt-launcher Pod** 세부 정보를 포함하여 가상 머신 정보를 수집합니다**. VirtualMachine** 및 **VirtualMachineInstance CR** 데이터는 모든 네임스페이스에 대해 수집됩니다**.**

VM=<vm_name>

특정 가상 머신에 대한 세부 정보를 수집합니다**.** 이 옵션을 사용하려면 **NS** 환경 변수를 사용하여 네임스페이스도 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.

scripts

각 스크립트는 특정 환경 변수 조합과만 호환됩니다.

/usr/bin/gather

모든 네임스페이스에서 클러스터 데이터를 수집하고 기본 **VM** 정보만 포함하는 기본 **must-gather** 스크립트를 사용합니다. 이 스크립트는 **PROS** 변수와만 호환됩니다.

/usr/bin/gather --vms_details

VM 로그 파일, **VM** 정의, 컨트롤 플레인 로그 및 **OpenShift Virtualization** 리소스에 속하는 네임스페이스를 수집합니다. 네임스페이스를 지정하면 해당 하위 오브젝트가 포함됩니다. 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.

/usr/bin/gather --images

이미지, 이미지 스트림 및 **image-stream-tags** 사용자 정의 리소스 정보를 수집합니다. 이 스크립트는 **PROS** 변수와만 호환됩니다.

14.15.3.1.2. 사용법 및 예

환경 변수는 선택 사항입니다. 자체적으로 또는 하나 이상의 호환 환경 변수를 사용하여 스크립트를 실행할 수 있습니다.

표 **14.1.** 호환 매개변수

스크립트	호환 가능한 환경 변수
/usr/bin/gather	• **PROS=<number_of_processes>**

스크립트	호환 가능한 환경 변수
/usr/bin/gather --vms_details	• 네임스페이스의 경우: **NS=<namespace_name>** • VM: **VM=<vm_name> NS=<namespace_name>** • **PROS=<number_of_processes>**
/usr/bin/gather --images	• **PROS=<number_of_processes>**

구문

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.12.13 \
  -- <environment_variable_1> <environment_variable_2> <script_name>
```

기본 데이터 수집 병렬 프로세스

기본적으로 **5**개의 프로세스가 병렬로 실행됩니다.

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.12.13 \
  -- PROS=5 /usr/bin/gather (1)
```

(1) 기본값을 변경하여 병렬 프로세스 수를 수정할 수 있습니다.

VM 세부 정보

다음 명령은 **mynamespace** 네임스페이스에서 **my-vm VM**에 대한 자세한 **VM** 정보를 수집합니다.

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.12.13 \
  -- NS=mynamespace VM=my-vm /usr/bin/gather --vms_details 1
```

1 **VM** 환경 변수를 사용하는 경우 **NS** 환경 변수는 필수입니다.

image, image-stream 및 **image-stream-tags** 정보

다음 명령은 클러스터에서 이미지, 이미지 스트림 및 **image-stream-tags** 정보를 수집합니다.

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.12.13 \
  -- /usr/bin/gather --images
```

14.15.3.2. 추가 리소스

- **must-gather** 툴 정보

15장. 백업 및 복원

15.1. OADP 설치 및 구성

클러스터 관리자는 **OADP Operator**를 설치하여 **OADP(Data Protection)**용 **OpenShift API**를 설치합니다. **Operator**는 **Velero 1.12** 를 설치합니다.

백업 스토리지 공급자에 대한 기본 보안을 생성한 다음 데이터 보호 애플리케이션을 설치합니다.

15.1.1. OADP Operator 설치

OLM(Operator Lifecycle Manager)을 사용하여 **OpenShift Container Platform 4.12**에 **OADP(Data Protection) Operator**를 설치합니다.

OADP Operator는 **Velero 1.12** 를 설치합니다.

사전 요구 사항

- **cluster-admin** 권한이 있는 사용자로 로그인해야 합니다.

절차

1. **OpenShift Container Platform** 웹 콘솔에서 **Operator** → **OperatorHub**를 클릭합니다.
2. 키워드로 필터링 필드를 사용하여 **OADP Operator** 를 찾습니다.
3. **OADP Operator** 를 선택하고 설치를 클릭합니다.
4. **openshift-adp** 프로젝트에서 **Operator**를 설치하려면 설치를 클릭합니다.
5. **Operators** → 설치된 **Operators** 를 클릭하여 설치를 확인합니다.

15.1.2. 백업 및 스냅샷 위치 및 보안 정보

DataProtectionApplication CR(사용자 정의 리소스**)**에서 백업 및 스냅샷 위치 및 해당 시크릿을 지정합니다**.**

백업 위치

AWS S3 호환 오브젝트 스토리지를 **Multicloud Object Gateway, Ceph RADOS Gateway** 또는 **MinIO**와 같은 백업 위치로 지정합니다**.**

Velero는 **OpenShift Container Platform** 리소스**, Kubernetes** 오브젝트 및 내부 이미지를 오브젝트 스토리지의 아카이브 파일로 백업합니다**.**

스냅샷 위치

영구 볼륨을 백업하기 위해 클라우드 공급자의 네이티브 스냅샷 **API**를 사용하는 경우 클라우드 공급자를 스냅샷 위치로 지정해야 합니다**.**

CSI(Container Storage Interface) 스냅샷을 사용하는 경우 **CSI** 드라이버를 등록하기 위해 **VolumeSnapshotClass CR**을 생성하기 때문에 스냅샷 위치를 지정할 필요가 없습니다**.**

Restic을 사용하는 경우 **Restic**이 개체 스토리지의 파일 시스템을 백업하므로 스냅샷 위치를 지정할 필요가 없습니다**.**

보안

백업 및 스냅샷 위치가 동일한 자격 증명을 사용하거나 스냅샷 위치가 필요하지 않은 경우 기본 시크릿을 생성합니다**.**

백업 및 스냅샷 위치가 다른 인증 정보를 사용하는 경우 두 개의 시크릿 오브젝트를 생성합니다**.**

- **DataProtectionApplication CR**에서 지정하는 백업 위치에 대한 사용자 정의 시크릿 입니다**.**
- **DataProtectionApplication CR**에서 참조되지 않는 스냅샷 위치에 대한 기본 시크릿 입니다**.**

중요

데이터 보호 애플리케이션에는 기본 보안이 필요합니다. 그렇지 않으면 설치에 실패합니다.

설치 중에 백업 또는 스냅샷 위치를 지정하지 않으려면 빈 **credentials-velero** 파일을 사용하여 기본 보안을 생성할 수 있습니다.

15.1.2.1. 기본 보안 생성

백업 및 스냅샷 위치가 동일한 자격 증명을 사용하거나 스냅샷 위치가 필요하지 않은 경우 기본 보안을 생성합니다.

참고

DataProtectionApplication CR(사용자 정의 리소스**)**에는 기본 시크릿 이 필요합니다. 그렇지 않으면 설치에 실패합니다. 백업 위치 **Secret** 의 이름이 지정되지 않은 경우 기본 이름이 사용됩니다.

설치 중에 백업 위치 자격 증명을 사용하지 않으려면 빈 **credentials-velero** 파일을 사용하여 기본 이름으로 **Secret** 을 생성할 수 있습니다.

사전 요구 사항

- 오브젝트 스토리지 및 클라우드 스토리지는 동일한 인증 정보를 사용해야 합니다.
- **Velero**에 대한 오브젝트 스토리지를 구성해야 합니다.
- 오브젝트 스토리지의 **credentials-velero** 파일을 적절한 형식으로 만들어야 합니다.

절차

- 기본 이름으로 보안을 생성합니다.

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file
cloud=credentials-velero
```

Secret 은 데이터 보호 애플리케이션을 설치할 때 **DataProtectionApplication CR**의 **spec.backupLocations.credential** 블록에서 참조합니다.

15.1.3. 데이터 보호 애플리케이션 구성

Velero 리소스 할당을 설정하거나 자체 서명된 **CA** 인증서를 활성화하여 데이터 보호 애플리케이션을 구성할 수 있습니다.

15.1.3.1. Velero CPU 및 메모리 리소스 할당 설정

DataProtectionApplication CR(사용자 정의 리소스) 매니페스트를 편집하여 **Velero Pod**의 **CPU** 및 메모리 리소스 할당을 설정합니다.

사전 요구 사항

- **OADP(OpenShift API for Data Protection) Operator**가 설치되어 있어야 합니다.

절차

- 다음 예와 같이 **DataProtectionApplication CR** 매니페스트의 **spec.configuration.velero.podConfig.ResourceAllocations** 블록에서 값을 편집합니다.

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  configuration:
    velero:
      podConfig:
        nodeSelector: <node selector> (1)
        resourceAllocations: (2)
          limits:
            cpu: "1"
            memory: 1024Mi
          requests:
            cpu: 200m
            memory: 256Mi
```

(1) **Velero podSpec**에 제공할 노드 선택기를 지정합니다.

2 나열된 **resourceAllocations** 는 평균 사용량입니다.

15.1.3.2. 자체 서명 CA 인증서 활성화

알 수 없는 기관 오류로 서명된 인증서를 방지하려면 **DataProtectionApplication CR**(사용자 정의 리소스) 매니페스트를 편집하여 개체 스토리지에 대해 자체 서명된 **CA** 인증서를 활성화해야 합니다.

사전 요구 사항

- **OADP(OpenShift API for Data Protection) Operator**가 설치되어 있어야 합니다.

절차

- **DataProtectionApplication CR** 매니페스트의 **spec.backupLocations.velero.objectStorage.caCert** 매개변수 및 **spec.backupLocations.velero.config** 매개변수를 편집합니다.

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket>
          prefix: <prefix>
          caCert: <base64_encoded_cert_string> (1)
        config:
          insecureSkipTLSVerify: "false" (2)
...
```

1 **Base64** 인코딩 **CA** 인증서 문자열을 지정합니다.

2 **insecureSkipTLSVerify** 구성은 **"true"** 또는 **"false"** 로 설정할 수 있습니다. **"true"** 로 설정하면 **SSL/TLS** 보안이 비활성화됩니다. **"false"** 로 설정하면 **SSL/TLS** 보안이 활성화

됩니다.

15.1.4. 데이터 보호 애플리케이션 설치

Data ProtectionApplication API 인스턴스를 만들어 **DPA(Data Protection Application)**를 설치합니다.

사전 요구 사항

- **OADP Operator**를 설치해야 합니다.
- 오브젝트 스토리지를 백업 위치로 구성해야 합니다.
- 스냅샷을 사용하여 **PV**를 백업하는 경우 클라우드 공급자는 기본 스냅샷 **API** 또는 **CSI(Container Storage Interface)** 스냅샷을 지원해야 합니다.
- 백업 및 스냅샷 위치가 동일한 자격 증명을 사용하는 경우 기본 이름 **cloud-credentials** 를 사용하여 **Secret** 을 생성해야 합니다.
- 백업 및 스냅샷 위치가 다른 인증 정보를 사용하는 경우 두 개의 시크릿 을 생성해야 합니다.
  - 백업 위치에 대한 사용자 지정 이름이 있는 시크릿 입니다. 이 보안을 **DataProtectionApplication CR**에 추가합니다.
  - 스냅샷 위치에 대한 기본 이름 **cloud-credentials** 가 있는 시크릿 입니다. 이 보안 은 **DataProtectionApplication CR**에서 참조되지 않습니다.

참고

설치 중에 백업 또는 스냅샷 위치를 지정하지 않으려면 빈 **credentials-velero** 파일을 사용하여 기본 보안을 생성할 수 있습니다. 기본 보안이 없으면 설치에 실패합니다.

참고

Velero는 기본 백업 리포지토리 암호가 포함된 **OADP** 네임스페이스에 **velero-repo-credentials** 라는 시크릿을 생성합니다. 백업 리포지토리를 대상으로 하는 첫 번째 백업을 실행하기 전에 **base64**로 인코딩된 고유한 암호로 시크릿을 업데이트할 수 있습니다. 업데이트할 키의 값은 **Data[repository-password]** 입니다.

DPA를 생성한 후 백업 리포지토리를 대상으로 하는 백업을 처음 실행하면 **Velero**에서 시크릿이 기본 암호 또는 교체한 암호가 포함된 **velero-repo-credentials** 인 백업 리포지토리를 생성합니다. 첫 번째 백업 후 시크릿 암호를 업데이트하면 새 암호가 **velero-repo-credentials** 의 암호와 일치하지 않으므로 **Velero**는 이전 백업과 연결할 수 없습니다.

절차

1. **Operators** → 설치된 **Operator** 를 클릭하고 **OADP Operator**를 선택합니다.

2. 제공된 **API** 아래의 **DataProtectionApplication** 상자에서 인스턴스 생성을 클릭합니다.

3. **YAML** 보기를 클릭하고 **DataProtectionApplication** 매니페스트의 매개변수를 업데이트합니다.

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - kubevirt (1)
        - gcp (2)
        - csi (3)
        - openshift (4)
      resourceTimeout: 10m (5)
    restic:
      enable: true (6)
      podConfig:
        nodeSelector: <node_selector> (7)
  backupLocations:
    - velero:
        provider: gcp (8)
        default: true
```

```
    credential:
      key: cloud
      name: <default_secret> (9)
    objectStorage:
      bucket: <bucket_name> (10)
      prefix: <prefix> (11)
```

1. **OpenShift Virtualization**에는 **kubevirt** 플러그인이 필요합니다.

2. 백업 공급자의 플러그인을 지정합니다(예: 존재하는 경우 **gcp**).

3. **CSI** 스냅샷을 사용하여 **PV**를 백업하려면 **csi** 플러그인이 필요합니다. **csi** 플러그인은 **Velero CSI** 베타 스냅샷 **API** 를 사용합니다. 스냅샷 위치를 구성할 필요가 없습니다.

4. **openshift** 플러그인은 필수입니다.

5. **Velero CRD** 가용성, **volumeSnapshot** 삭제, 백업 리포지토리 가용성과 같이 시간 초과가 발생하기 전에 여러 **Velero** 리소스를 대기하는 분을 지정합니다. 기본값은 **10m**입니다.

6. **Restic** 설치를 비활성화하려면 이 값을 **false** 로 설정합니다. **Restic**은 데몬 세트를 배포합니다. 즉, **Restic pod**는 각 작업 노드에서 실행됩니다. **OADP** 버전 **1.2** 이상에서는 **Backup CR**에 **spec.defaultVolumesToFsBackup: true** 를 추가하여 백업에 대해 **Restic** 을 구성할 수 있습니다. **OADP** 버전 **1.1**에서 **Backup CR**에 **spec.defaultVolumesToRestic: true** 를 추가합니다.

7. **Restic**을 사용할 수 있는 노드를 지정합니다. 기본적으로 **Restic**은 모든 노드에서 실행됩니다.

8. 백업 공급자를 지정합니다.

9.

10

버킷을 백업 스토리지 위치로 지정합니다. 버킷이 **Velero** 백업용 전용 버킷이 아닌 경우 접두사를 지정해야 합니다.

11

버킷이 여러 용도로 사용되는 경우 **Velero** 백업의 접두사(예: **velero**)를 지정합니다.

4. 생성을 클릭합니다.

5. **OADP** 리소스를 확인하여 설치를 확인합니다.

```
$ oc get all -n openshift-adp
```

출력 예

```
NAME                                                    READY   STATUS    RESTARTS   AGE
pod/oadp-operator-controller-manager-67d9494d47-6l8z8   2/2     Running   0          2m8s
pod/restic-9cq4q                                        1/1     Running   0          94s
pod/restic-m4lts                                        1/1     Running   0          94s
pod/restic-pv4kr                                        1/1     Running   0          95s
pod/velero-588db7f655-n842v                             1/1     Running   0          95s

NAME                                                    TYPE        CLUSTER-IP      EXTERNAL-IP
PORT(S)    AGE
service/oadp-operator-controller-manager-metrics-service   ClusterIP   172.30.70.140
<none>        8443/TCP   2m8s

NAME                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/restic   3         3         3       3            3           <none>          96s

NAME                                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/oadp-operator-controller-manager   1/1     1            1           2m9s
deployment.apps/velero                             1/1     1            1           96s

NAME                                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/oadp-operator-controller-manager-67d9494d47   1         1         1       2m9s
replicaset.apps/velero-588db7f655                             1         1         1       96s
```

15.1.4.1. DataProtectionApplication CR에서 CSI 활성화

CSI 스냅샷으로 영구 볼륨을 백업하기 위해 **DataProtectionApplication CR(**사용자 정의 리소스**)**에서 **CSI(Container Storage Interface)**를 활성화합니다**.**

사전 요구 사항

- 클라우드 공급자는 **CSI** 스냅샷을 지원해야 합니다**.**

절차

- 다음 예와 같이 **DataProtectionApplication CR**을 편집합니다**.**

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - csi (1)
```

(1) **csi** 기본 플러그인을 추가합니다**.**

15.1.5. OADP 설치 제거

OADP Operator를 삭제하여 **OADP(Data Protection)**용 **OpenShift API**를 설치 제거합니다**.** 자세한 내용은 클러스터에서 **Operator** 삭제를 참조하십시오**.**

15.2. 가상 머신 백업 및 복원

중요

OpenShift Virtualization의 **OADP**는 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.

OADP(OpenShift API for Data Protection) 를 사용하여 가상 머신을 백업하고 복원합니다.

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** 플러그인 을 사용하여 데이터 보호 애플리케이션을 설치합니다.
3. **Backup CR(**사용자 정의 리소스**)** 을 생성하여 가상 머신을 백업합니다.
4. **Restore CR** 을 생성하여 **Backup CR**을복원합니다.

15.2.1. 추가 리소스

- **OADP** 기능 및 플러그인
- 문제 해결

15.3. 가상 머신 백업

중요

OpenShift Virtualization의 **OADP**는 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.

OADP(OpenShift API for Data Protection) Backup CR(사용자 정의 리소스**)**을 생성하여 **VM(**가상 머신**)** 을 백업합니다.

Backup CR은 다음 작업을 수행합니다.

- **Multicloud Object Gateway, Noobaa** 또는 **Minio**와 같은 **S3** 호환 오브젝트 스토리지에 아카이브 파일을 생성하여 **OpenShift Virtualization** 리소스를 백업합니다.
- 다음 옵션 중 하나를 사용하여 **VM** 디스크를 백업합니다.
  - **CSI(Container Storage Interface)** 스냅샷 **(**예**: Ceph RBD** 또는 **Ceph FS)**
  - 파일 시스템 백업을 사용하여 애플리케이션 백업**:** 오브젝트 스토리지에서 **Kopia** 또는 **Restic.**

참고

OADP는 백업 작업 전에 **VM** 파일 시스템을 정지하고 백업이 완료되면 이를 해제 해제하는 백업 후크를 제공합니다.

kubevirt-controller 는 백업 작업 전후에 **Velero**가 **virt-freezer** 바이너리를 실행할 수 있는 주석을 사용하여 **virt-launcher Pod**를 생성합니다.

freeze 및 **unfreeze API**는 **VM** 스냅샷 **API**의 하위 리소스입니다. 자세한 내용은 가상 머신 스냅샷 정보를 참조하십시오.

Backup CR에 후크를 추가하여 백업 작업 전이나 후에 특정 **VM**에서 명령을 실행할 수 있습니다.

Backup CR 대신 **Schedule CR**을 생성하여 백업을 예약합니다.

15.3.1. Backup CR 생성

Kubernetes 리소스, 내부 이미지 및 **PV(**영구 볼륨**)**를 백업하려면 **Backup** 사용자 정의 리소스**(CR)**를 생성합니다.

사전 요구 사항

- **OADP(OpenShift API for Data Protection) Operator**를 설치해야 합니다.
- **DataProtectionApplication CR**은 **Ready** 상태여야 합니다.
- 백업 위치 사전 요구 사항:
  - **Velero**용으로 **S3** 오브젝트 스토리지가 구성되어 있어야 합니다.
  - **DataProtectionApplication CR**에 백업 위치가 구성되어 있어야 합니다.

- 스냅샷 위치 사전 요구 사항:
  - 클라우드 공급자에는 기본 스냅샷 **API**가 있거나 **CSI(Container Storage Interface)** 스냅샷을 지원해야 합니다.
  - **CSI** 스냅샷의 경우 **CSI** 드라이버를 등록하려면 **VolumeSnapshotClass CR**을 생성해야 합니다.
  - **DataProtectionApplication CR**에 구성된 볼륨 위치가 있어야 합니다.

절차

1. 다음 명령을 입력하여 **backupStorageLocations CR**을 검색합니다.

```
$ oc get backupStorageLocations -n openshift-adp
```

출력 예

```
NAMESPACE       NAME              PHASE       LAST VALIDATED   AGE   DEFAULT
openshift-adp   velero-sample-1   Available   11s              31m
```

2. 다음 예와 같이 **Backup CR**을 생성합니다.

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: <backup>
  labels:
    velero.io/storage-location: default
  namespace: openshift-adp
spec:
  hooks: {}
  includedNamespaces:
  - <namespace> (1)
  includedResources: [] (2)
  excludedResources: [] (3)
```

```
storageLocation: <velero-sample-1> (4)
ttl: 720h0m0s
labelSelector: (5)
  matchLabels:
    app=<label_1>
    app=<label_2>
    app=<label_3>
orLabelSelectors: (6)
- matchLabels:
    app=<label_1>
    app=<label_2>
    app=<label_3>
```

1. 백업할 네임스페이스의 배열을 지정합니다.

2. 선택 사항: 백업에 포함할 리소스 배열을 지정합니다. 리소스는 바로 가기(예: 'pods'의 경우 'po')이거나 정규화될 수 있습니다. 지정하지 않으면 모든 리소스가 포함됩니다.

3. 선택 사항: 백업에서 제외할 리소스 배열을 지정합니다. 리소스는 바로 가기(예: 'pods'의 경우 'po')이거나 정규화될 수 있습니다.

4. backupStorageLocations CR의 이름을 지정합니다.

5. 지정된 라벨이 모두 있는 백업 리소스의 {key,value} 쌍의 맵입니다.

6. 지정된 라벨이 하나 이상 있는 백업 리소스의 {key,value} 쌍의 맵입니다.

3. Backup CR의 상태가 Completed 인지 확인합니다.

```
$ oc get backup -n openshift-adp <backup> -o jsonpath='{.status.phase}'
```

15.3.1.1. CSI 스냅샷을 사용하여 영구 볼륨 백업

Backup CR을 생성하기 전에 클라우드 스토리지의 **VolumeSnapshotClass CR(**사용자 정의 리소스**)**을 편집하여 **CSI(Container Storage Interface)** 스냅샷을 사용하여 영구 볼륨을 백업 합니다**.**

사전 요구 사항

- 클라우드 공급자는 **CSI** 스냅샷을 지원해야 합니다**.**
- **DataProtectionApplication CR**에서 **CSI**를 활성화해야 합니다**.**

절차

- **metadata.labels.velero.io/csi-volumesnapshot-class: "true"** 키**-**값 쌍을 **VolumeSnapshotClass CR**에 추가합니다**.**

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: <volume_snapshot_class_name>
  labels:
    velero.io/csi-volumesnapshot-class: "true"
driver: <csi_driver>
deletionPolicy: Retain
```

이제 **Backup CR**을 생성할 수 있습니다**.**

15.3.1.2. Restic을 사용하여 애플리케이션 백업

Restic을 사용하여 **Kubernetes** 리소스**,** 내부 이미지 및 영구 볼륨을 백업하여 **Backup CR(**사용자 정의 리소스**)**을 편집합니다**.**

DataProtectionApplication CR에서 스냅샷 위치를 지정할 필요가 없습니다**.**

중요

Restic은 **hostPath** 볼륨 백업을 지원하지 않습니다**.** 자세한 내용은 추가 **Restic** 제한 사항을 참조하십시오**.**

사전 요구 사항

- **OADP(OpenShift API for Data Protection) Operator**를 설치해야 합니다.
- **DataProtectionApplication CR**에서 **spec.configuration.restic.enable** 을 **false** 로 설정하여 기본 **Restic** 설치를 비활성화할 수 없습니다.
- **DataProtectionApplication CR**은 **Ready** 상태여야 합니다.

절차

- 다음 예와 같이 **Backup CR**을 편집합니다.

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: <backup>
  labels:
    velero.io/storage-location: default
  namespace: openshift-adp
spec:
  defaultVolumesToFsBackup: true (1)
...
```

(1) **OADP** 버전 **1.2** 이상에서 **spec** 블록 내에 **defaultVolumesToFsBackup: true** 설정을 추가합니다. **OADP** 버전 **1.1**에서 **defaultVolumesToRestic: true** 를 추가합니다.

15.3.1.3. 백업 후크 생성

Backup CR(사용자 정의 리소스)을 편집하여 **Pod**의 컨테이너에서 명령을 실행하는 백업 후크를 생성합니다.

포드를 백업하기 전에 ***사전*** 후크가 실행됩니다. 백업 후 후크가 실행됩니다.

절차

- *다음 예와 같이 **Backup CR**의 **spec.hooks** 블록에 후크를 추가합니다.*

```
apiVersion: velero.io/v1
```

```
kind: Backup
metadata:
  name: <backup>
  namespace: openshift-adp
spec:
  hooks:
    resources:
      - name: <hook_name>
        includedNamespaces:
        - <namespace> 1
        excludedNamespaces: 2
        - <namespace>
        includedResources: []
        - pods 3
        excludedResources: [] 4
        labelSelector: 5
          matchLabels:
            app: velero
            component: server
        pre: 6
          - exec:
              container: <container> 7
              command:
              - /bin/uname 8
              - -a
              onError: Fail 9
              timeout: 30s 10
        post: 11
...
```

1. 선택 사항: 후크가 적용되는 네임스페이스를 지정할 수 있습니다. 이 값을 지정하지 않으면 후크가 모든 네임스페이스에 적용됩니다.

2. 선택 사항: 후크가 적용되지 않는 네임스페이스를 지정할 수 있습니다.

3. 현재 **Pod**는 후크를 적용할 수 있는 유일한 지원 리소스입니다.

4. 선택 사항: 후크가 적용되지 않는 리소스를 지정할 수 있습니다.

5. 선택 사항: 이 후크는 레이블과 일치하는 오브젝트에만 적용됩니다. 이 값을 지정하지 않으면 후크가 모든 네임스페이스에 적용됩니다.

6. 백업 전에 실행할 후크 배열입니다.

7. 선택 사항: 컨테이너를 지정하지 않으면 **Pod**의 첫 번째 컨테이너에서 명령이 실행됩니다.

8. 이는 추가되는 **init** 컨테이너의 진입점입니다.

9. 오류 처리에 허용되는 값은 **Fail** 및 **Continue** 입니다. 기본값은 **Fail** 입니다.

10. 선택 사항: 명령이 실행될 때까지 대기하는 시간입니다. 기본값은 **30s** 입니다.

11. 이 블록은 백업 후 실행할 후크 배열과 **pre-backup** 후크와 동일한 매개변수를 정의합니다.

15.3.2. 추가 리소스

- CSI 볼륨 스냅샷 개요

15.4. 가상 머신 복원

Restore CR 을 생성하여 **OADP(OpenShift API for Data Protection) Backup CR**(사용자 정의 리소스)을복원합니다.

애플리케이션 컨테이너가 시작되기 전에 또는 애플리케이션 컨테이너 자체에서 **init** 컨테이너에서 명령을 실행하려면 **Restore CR**에 후크 를 추가할 수 있습니다.

15.4.1. Restore CR 생성

Restore CR을 생성하여 Backup CR(사용자 정의 리소스)을 복원합니다.

사전 요구 사항

- OADP(OpenShift API for Data Protection) Operator를 설치해야 합니다.
- DataProtectionApplication CR은 Ready 상태여야 합니다.
- Velero Backup CR이 있어야 합니다.
- PV(영구 볼륨) 용량이 백업 시 요청된 크기와 일치해야 합니다. 필요한 경우 요청된 크기를 조정합니다.

절차

1. 다음 예와 같이 Restore CR을 생성합니다.

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: <restore>
  namespace: openshift-adp
spec:
  backupName: <backup> (1)
  includedResources: [] (2)
  excludedResources:
  - nodes
  - events
  - events.events.k8s.io
  - backups.velero.io
  - restores.velero.io
  - resticrepositories.velero.io
  restorePVs: true (3)
```

(1) Backup CR의 이름입니다.

(2) 선택 사항: 복원 프로세스에 포함할 리소스 배열을 지정합니다. 리소스는 바로 가기 (예: Pod의 경우) 또는 정규화된일 수 있습니다. 지정하지 않으면 모든 리소스가 포함됩니

다.

3. 선택 사항: restorePVs 매개변수를 false 로 설정하여 VolumeSnapshot of Container Storage Interface(CSI) 스냅샷에서 PersistentVolumes 복원을 끄거나 VolumeSnapshotLocation 이 구성된 경우 기본 스냅샷에서 설정할 수 있습니다.

2. 다음 명령을 입력하여 Restore CR의 상태가 Completed 인지 확인합니다.

```
$ oc get restore -n openshift-adp <restore> -o jsonpath='{.status.phase}'
```

3. 다음 명령을 입력하여 백업 리소스가 복원되었는지 확인합니다.

```
$ oc get all -n <namespace> (1)
```

1. 백업한 네임스페이스입니다.

4. Restic을 사용하여 DeploymentConfig 오브젝트를 복원하거나 post-restore 후크를 사용하는 경우 다음 명령을 입력하여 dc-restic-post-restore.sh cleanup 스크립트를 실행합니다.

```
$ bash dc-restic-post-restore.sh <restore-name>
```

참고

복원 프로세스 중에 OADP Velero 플러그인은 DeploymentConfig 오브젝트를 축소하고 Pod를 독립 실행형 포드로 복원합니다. 이 작업은 클러스터가 복원 시 복원된 DeploymentConfig Pod를 즉시 삭제하고 Restic 및 복원 후 후크를 허용하여 복원된 Pod에서 작업을 완료하지 못하도록 하기 위해 수행됩니다. 아래에 표시된 정리 스크립트는 이러한 연결이 끊긴 Pod를 제거하고 적절한 복제본 수로 DeploymentConfig 오브젝트를 다시 확장합니다.

예 15.1. dc-restic-post-restore.sh cleanup script

```
#!/bin/bash
set -e

# if sha256sum exists, use it to check the integrity of the file
if command -v sha256sum >/dev/null 2>&1; then
```

```
  CHECKSUM_CMD="sha256sum"
else
  CHECKSUM_CMD="shasum -a 256"
fi

label_name () {
   if [ "${#1}" -le "63" ]; then
 echo $1
 return
   fi
   sha=$(echo -n $1|$CHECKSUM_CMD)
   echo "${1:0:57}${sha:0:6}"
}

OADP_NAMESPACE=${OADP_NAMESPACE:=openshift-adp}

if [[ $# -ne 1 ]]; then
   echo "usage: ${BASH_SOURCE} restore-name"
   exit 1
fi

echo using OADP Namespace $OADP_NAMESPACE
echo restore: $1

label=$(label_name $1)
echo label: $label

echo Deleting disconnected restore pods
oc delete pods -l oadp.openshift.io/disconnected-from-dc=$label

for dc in $(oc get dc --all-namespaces -l oadp.openshift.io/replicas-modified=$label
-o jsonpath='{range .items[*]}{.metadata.namespace}{","}{.metadata.name}{","}
{.metadata.annotations.oadp\.openshift\.io/original-replicas}{","}
{.metadata.annotations.oadp\.openshift\.io/original-paused}{"\n"}')
do
   IFS=',' read -ra dc_arr <<< "$dc"
   if [ ${#dc_arr[0]} -gt 0 ]; then
 echo Found deployment ${dc_arr[0]}/${dc_arr[1]}, setting replicas: ${dc_arr[2]},
paused: ${dc_arr[3]}
 cat <<EOF | oc patch dc  -n ${dc_arr[0]} ${dc_arr[1]} --patch-file /dev/stdin
spec:
  replicas: ${dc_arr[2]}
  paused: ${dc_arr[3]}
EOF
   fi
done
```

15.4.1.1. 복원 후크 생성

Restore CR(사용자 정의 리소스)을 편집하여 **Pod**의 컨테이너에서 명령을 실행할 복원 후크를 생성합니다.

두 가지 유형의 복원 후크를 생성할 수 있습니다.

- init 후크는 애플리케이션 컨테이너가 시작되기 전에 설정 작업을 수행하기 위해 Pod에 init 컨테이너를 추가합니다.

  Restic 백업을 복원하는 경우 복원 후크 init 컨테이너 앞에 restic-wait init 컨테이너가 추가됩니다.

- exec 후크는 복원된 Pod의 컨테이너에서 명령 또는 스크립트를 실행합니다.

절차

- 다음 예와 같이 Restore CR의 spec.hooks 블록에 후크를 추가합니다.

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: <restore>
  namespace: openshift-adp
spec:
  hooks:
    resources:
      - name: <hook_name>
        includedNamespaces:
        - <namespace> 1
        excludedNamespaces:
        - <namespace>
        includedResources:
        - pods 2
        excludedResources: []
        labelSelector: 3
          matchLabels:
            app: velero
            component: server
        postHooks:
        - init:
            initContainers:
            - name: restore-hook-init
              image: alpine:latest
              volumeMounts:
              - mountPath: /restores/pvc1-vm
                name: pvc1-vm
              command:
              - /bin/ash
              - -c
            timeout: 4
```

```
    - exec:
      container: <container> (5)
      command:
      - /bin/bash (6)
      - -c
      - "psql < /backup/backup.sql"
      waitTimeout: 5m (7)
      execTimeout: 1m (8)
      onError: Continue (9)
```

1. *선택 사항: 후크가 적용되는 네임스페이스 배열입니다. 이 값을 지정하지 않으면 후크가 모든 네임스페이스에 적용됩니다.*

2. *현재 **Pod**는 후크를 적용할 수 있는 유일한 지원 리소스입니다.*

3. *선택 사항: 이 후크는 라벨 선택기와 일치하는 오브젝트에만 적용됩니다.*

4. *선택 사항: 시간 초과는 **Velero**가 **initContainers** 가 완료될 때까지 대기하는 최대 시간을 지정합니다.*

5. *선택 사항: 컨테이너를 지정하지 않으면 **Pod**의 첫 번째 컨테이너에서 명령이 실행됩니다.*

6. *이는 추가되는 **init** 컨테이너의 진입점입니다.*

7. *선택 사항: 컨테이너가 준비될 때까지 대기하는 시간입니다. 컨테이너가 시작되고 동일한 컨테이너의 이전 후크가 완료될 때까지 충분히 길어야 합니다. 설정되지 않은 경우 복원 프로세스는 무기한 대기합니다.*

8. *선택 사항: 명령이 실행될 때까지 대기하는 시간입니다. 기본값은 **30s** 입니다.*

9.

- ***continue:*** *명령 실패만 기록됩니다.*

- ***fail: Pod****의 컨테이너에서 더 이상 복원 후크가 실행되지 않습니다.* ***Restore CR*** *의 상태는* ***PartiallyFailed*** *가 됩니다.*