NetApp 사용한 Red Hat OpenShift

NetApp container solutions

NetApp
January 21, 2026

목차

NetApp 사용한 Red Hat OpenShift

NVA-1160: NetApp 사용한 Red Hat OpenShift

Alan Cowles와 Nikhil M Kulkarni, NetApp

이 참조 문서는 NetApp 에서 검증한 대로 여러 다른 데이터 센터 환경에서 Installer Provisioned Infrastructure(IPI)를 통해 배포된 Red Hat OpenShift 솔루션의 배포 검증을 제공합니다. 또한, Trident 스토리지 오케스트레이터를 사용하여 영구 스토리지를 관리함으로써 NetApp 스토리지 시스템과의 스토리지 통합에 대해서도 자세히 설명합니다. 마지막으로, 여러 솔루션 검증과 실제 사용 사례를 탐색하고 문서화합니다.

사용 사례

NetApp 솔루션이 포함된 Red Hat OpenShift는 다음과 같은 사용 사례를 가진 고객에게 탁월한 가치를 제공하도록 설계되었습니다.

- 베어 메탈, Red Hat OpenStack Platform, Red Hat Virtualization 및 VMware vSphere에서 IPI(Installer Provisioned Infrastructure)를 사용하여 배포된 Red Hat OpenShift는 쉽게 배포하고 관리할 수 있습니다.
- Red Hat OpenShift를 OSP, RHV 또는 vSphere에 가상으로 배포하거나 OpenShift 가상화를 통해 베어 메탈에 배포하여 엔터프라이즈 컨테이너와 가상화된 워크로드의 성능을 결합합니다.
- NetApp 스토리지와 Kubernetes를 위한 오픈소스 스토리지 오케스트레이터인 Trident 와 함께 사용할 때 Red Hat OpenShift의 기능을 강조하는 실제 구성 및 사용 사례입니다.

사업적 가치

기업들은 새로운 제품을 만들고, 출시 주기를 단축하고, 새로운 기능을 빠르게 추가하기 위해 DevOps 방식을 점점 더 많이 도입하고 있습니다. 컨테이너와 마이크로서비스는 본질적으로 민첩한 특성으로 인해 DevOps 실무를 지원하는 데 중요한 역할을 합니다. 그러나 기업 환경에서 프로덕션 규모로 DevOps를 실행하는 것은 고유한 과제를 제시하며, 다음과 같은 기본 인프라에 특정 요구 사항을 부과합니다.

- 스택의 모든 계층에서 높은 가용성
- 배포 절차의 용이성
- 중단 없는 운영 및 업그레이드
- 마이크로서비스 민첩성을 유지하기 위한 API 기반 및 프로그래밍 가능 인프라
- 성능 보장이 있는 멀티테넌시
- 가상화된 워크로드와 컨테이너화된 워크로드를 동시에 실행할 수 있는 기능
- 작업 부하 수요에 따라 인프라를 독립적으로 확장할 수 있는 기능

NetApp 이 탑재된 Red Hat OpenShift는 이러한 과제를 인식하고 고객이 선택한 데이터 센터 환경에서 RedHat OpenShift IPI의 완전 자동화된 배포를 구현하여 각 문제를 해결하는 솔루션을 제시합니다.

기술 개요

NetApp 솔루션이 포함된 Red Hat OpenShift는 다음과 같은 주요 구성 요소로 구성됩니다.

Red Hat OpenShift 컨테이너 플랫폼

Red Hat OpenShift Container Platform은 완벽하게 지원되는 엔터프라이즈 Kubernetes 플랫폼입니다. Red Hat은 컨테이너화된 애플리케이션을 빌드, 배포 및 관리하기 위한 모든 구성 요소가 완벽하게 통합된 애플리케이션 플랫폼을 제공하기 위해 오픈 소스 Kubernetes에 여러 가지 개선 사항을 적용했습니다.

자세한 내용은 OpenShift 웹사이트를 방문하세요. "여기" .

NetApp 스토리지 시스템

NetApp 기업 데이터 센터와 하이브리드 클라우드 구축에 적합한 여러 가지 스토리지 시스템을 보유하고 있습니다. NetApp 포트폴리오에는 NetApp ONTAP, NetApp Element, NetApp e-Series 스토리지 시스템이 포함되어 있으며, 이 모든 시스템은 컨테이너화된 애플리케이션에 대한 영구 스토리지를 제공할 수 있습니다.

자세한 내용은 NetApp 웹사이트를 방문하세요. "여기" .

NetApp 스토리지 통합

Trident Red Hat OpenShift를 포함한 컨테이너와 Kubernetes 배포판을 위한 오픈 소스이자 완벽하게 지원되는 스토리지 오케스트레이터입니다.

자세한 내용은 Trident 웹사이트를 방문하세요. "여기" .

고급 구성 옵션

이 섹션에서는 전용 개인 이미지 레지스트리를 만들거나 사용자 정의 로드 밸런서 인스턴스를 배포하는 등 실제 사용자가 이 솔루션을 프로덕션에 배포할 때 수행해야 할 가능성이 높은 사용자 정의에 대해 설명합니다.

검증된 릴리스에 대한 현재 지원 매트릭스

기술	목적	소프트웨어 버전
NetApp ONTAP	스토리지	9.8, 9.9.1, 9.12.1
NetApp Element	스토리지	12.3
NetApp Trident	스토리지 오케스트레이션	22.01.0, 23.04, 23.07, 23.10, 24.02
레드햇 오픈시프트	컨테이너 오케스트레이션	4.6 유럽, 4.7, 4.8, 4.10, 4.11, 4.12, 4.13, 4.14
VMware vSphere	데이터 센터 가상화	7.0, 8.0.2

레드햇 오픈시프트

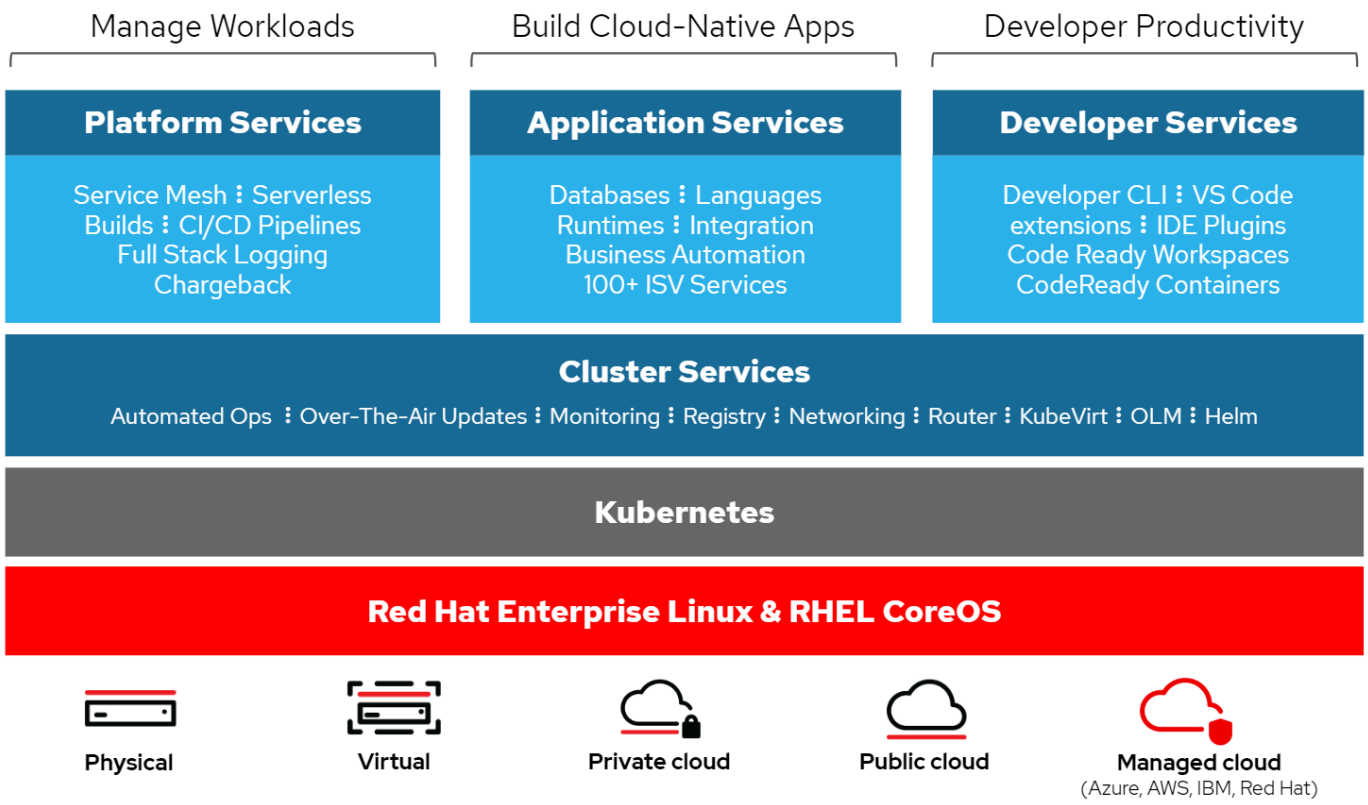
OpenShift 개요

Red Hat OpenShift Container Platform은 온프레미스 및 하이브리드 클라우드 인프라

전반에서 일관되게 애플리케이션을 빌드, 배포 및 관리할 수 있도록 개발 및 IT 운영을 단일 플랫폼에서 통합합니다. Red Hat OpenShift는 Kubernetes와 컨테이너 기반 워크로드에 맞춰 설계된 세계 최고의 엔터프라이즈 Linux 배포판인 Red Hat Enterprise Linux CoreOS를 포함한 오픈 소스 혁신과 업계 표준을 기반으로 구축되었습니다. OpenShift는 CNCF(Cloud Native Computing Foundation) 인증 Kubernetes 프로그램의 일부로, 컨테이너 워크로드의 이식성과 상호 운용성을 제공합니다.

Red Hat OpenShift는 다음과 같은 기능을 제공합니다**.**

- 셀프 서비스 프로비저닝 개발자는 가장 많이 사용하는 도구를 이용해 필요에 따라 빠르고 쉽게 애플리케이션을 만들 수 있으며, 운영팀은 전체 환경에 대한 완벽한 제어권을 유지합니다.
- 영구 저장소 OpenShift Container Platform은 영구 저장소에 대한 지원을 제공함으로써 상태 저장 애플리케이션과 클라우드 기반 상태 비저장 애플리케이션을 모두 실행할 수 있도록 합니다.
- 지속적인 통합 및 지속적인 개발**(CI/CD)** 이 소스 코드 플랫폼은 대규모 빌드 및 배포 이미지를 관리합니다.
- 오픈 소스 표준 이러한 표준은 컨테이너 오케스트레이션을 위해 다른 오픈 소스 기술 외에도 OCI(Open Container Initiative)와 Kubernetes를 통합합니다. 특정 공급업체의 기술이나 사업 로드맵에 구애받지 않습니다.
- **CI/CD** 파이프라인 OpenShift는 CI/CD 파이프라인에 대한 기본 제공 지원을 제공하므로 개발팀은 애플리케이션 제공 프로세스의 모든 단계를 자동화하고 애플리케이션의 코드나 구성에 대한 모든 변경 사항이 실행되도록 할 수 있습니다.
- 역할 기반 액세스 제어**(RBAC)** 이 기능은 팀 및 사용자 추적을 제공하여 대규모 개발자 그룹을 구성하는 데 도움이 됩니다.
- 자동화된 빌드 및 배포 OpenShift는 개발자에게 컨테이너화된 애플리케이션을 빌드하거나 플랫폼이 애플리케이션 소스 코드 또는 바이너리에서 컨테이너를 빌드하는 옵션을 제공합니다. 그러면 플랫폼은 애플리케이션에 대해 정의된 특성에 따라 인프라 전반에 걸쳐 이러한 애플리케이션을 자동으로 배포합니다. 예를 들어, 타사 라이선스를 준수하기 위해 할당해야 할 리소스 양과 인프라의 어느 위치에 배포해야 하는지입니다.
- 일관된 환경 OpenShift는 개발자에게 제공되는 환경과 애플리케이션의 수명 주기 전반에 걸쳐 일관성을 유지하도록 보장합니다. 이는 운영 체제, 라이브러리, 런타임 버전(예: Java 런타임), 심지어 사용 중인 애플리케이션 런타임(예: Tomcat)까지 일관성을 유지하여 일관되지 않은 환경에서 발생하는 위험을 제거합니다.
- 구성 관리 구성 및 민감한 데이터 관리 기능이 플랫폼에 내장되어 있어 애플리케이션을 구축하는 데 사용된 기술이나 배포 환경에 관계없이 일관되고 환경에 독립적인 애플리케이션 구성이 애플리케이션에 제공됩니다.
- 애플리케이션 로그 및 메트릭**.** 빠른 피드백은 애플리케이션 개발에 있어서 중요한 측면입니다. OpenShift의 통합 모니터링 및 로그 관리 기능은 개발자에게 즉각적인 측정 항목을 제공하여 개발자가 애플리케이션이 변경 사항에 따라 어떻게 동작하는지 연구하고 애플리케이션 수명 주기에서 가능한 한 빨리 문제를 해결할 수 있도록 지원합니다.
- 보안 및 컨테이너 카탈로그 OpenShift는 다중 테넌시를 제공하고 SELinux(Security-Enhanced Linux), CGroups, seccomp(Secure Computing Mode)를 사용하여 컨테이너를 격리하고 보호함으로써 사용자를 유해한 코드 실행으로부터 보호합니다. 또한 다양한 하위 시스템에 대한 TLS 인증서를 통한 암호화를 제공하고, 특히 보안에 중점을 두고 스캔 및 등급이 매겨진 Red Hat 인증 컨테이너(access.redhat.com/containers)에 대한 액세스를 제공하여 최종 사용자에게 인증되고 신뢰할 수 있으며 안전한 애플리케이션 컨테이너를 제공합니다.



Red Hat OpenShift 배포 방법

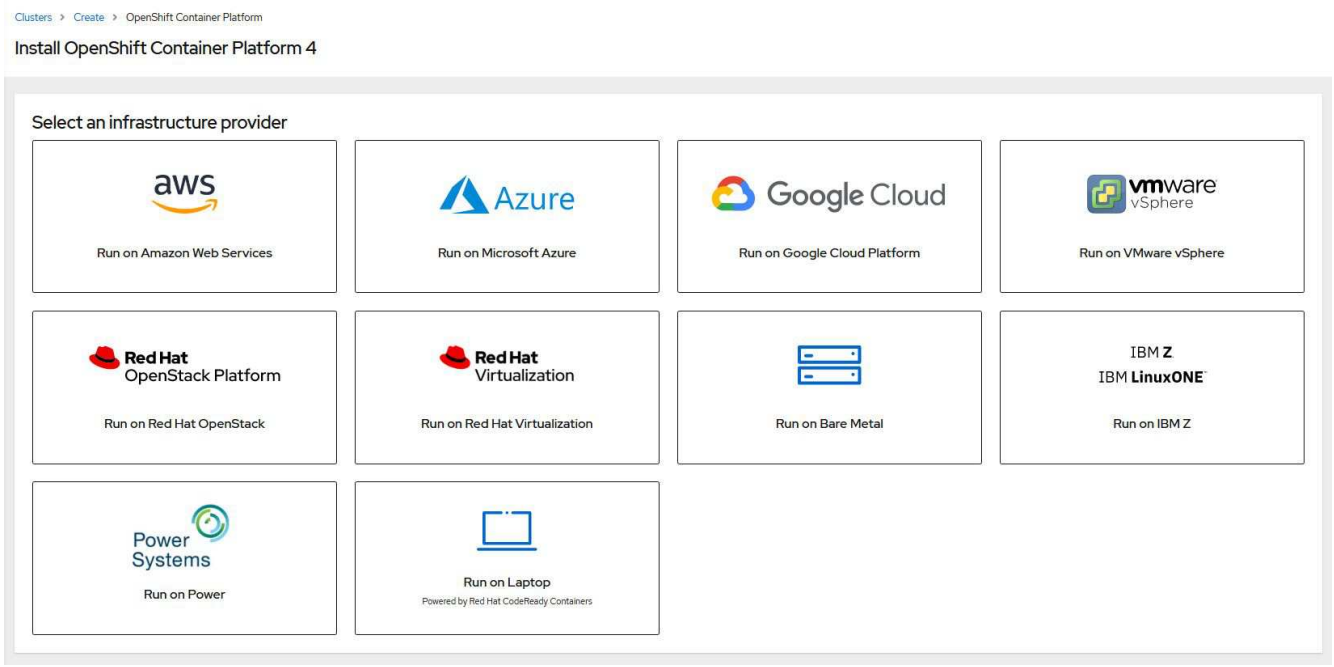
Red Hat OpenShift 4부터 OpenShift 배포 방법에는 고도로 사용자 정의된 배포를 위한 사용자 프로비저닝 인프라(UPI)를 사용한 수동 배포나 설치 프로그램 프로비저닝 인프라(IPI)를 사용한 완전 자동화된 배포가 포함됩니다.

대부분의 경우 IPI 설치 방법이 선호되는 방법입니다. 이 방법을 사용하면 개발, 테스트, 프로덕션 환경에 OpenShift 클러스터를 빠르게 배포할 수 있습니다.

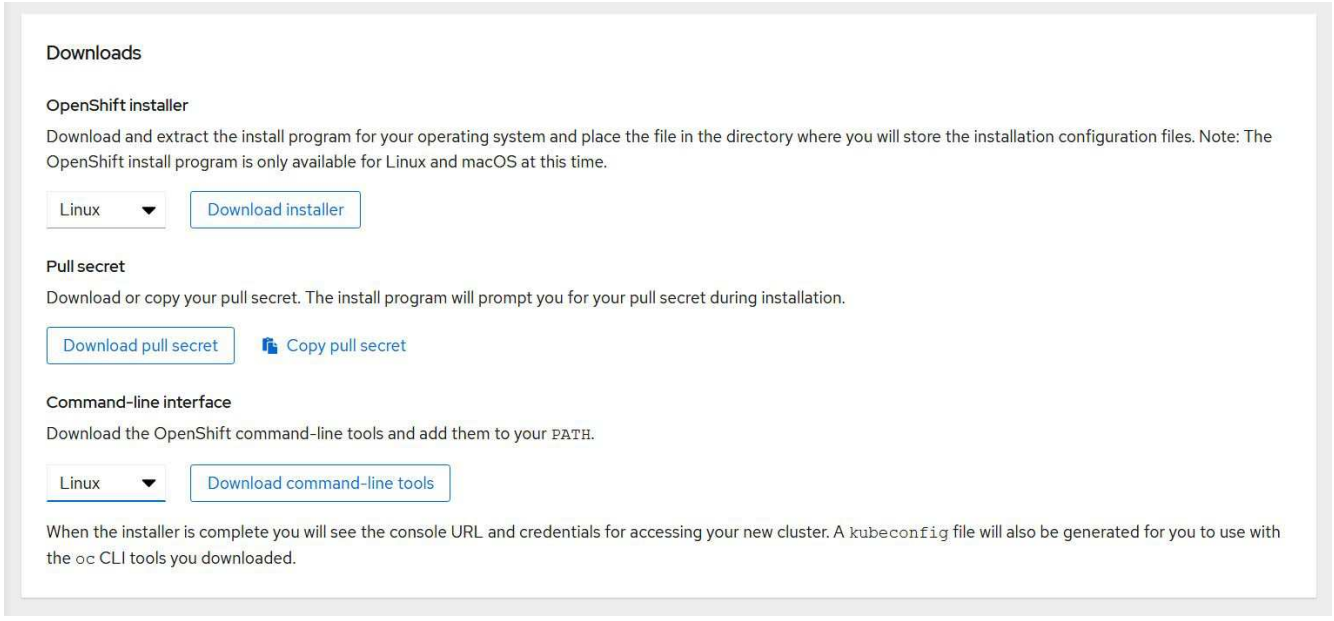
Red Hat OpenShift의 **IPI** 설치

OpenShift의 IPI(Installer Provisioned Infrastructure) 배포에는 다음과 같은 상위 단계가 포함됩니다.

1. Red Hat OpenShift를 방문하세요"웹사이트" SSO 자격 증명을 사용하여 로그인하세요.
2. Red Hat OpenShift를 배포할 환경을 선택하세요.



3. 다음 화면에서 설치 프로그램, 고유한 풀 시크릿, 관리용 CLI 도구를 다운로드하세요.



4. 를 따르세요"설치 지침" Red Hat에서 제공하여 원하는 환경에 배포할 수 있습니다.

NetApp OpenShift 배포를 검증했습니다.

NetApp 다음의 각 데이터 센터 환경에서 IPI(Installer Provisioned Infrastructure) 배포 방법을 사용하여 실험실에서 Red Hat OpenShift 배포를 테스트하고 검증했습니다.

- "베어 메탈의 OpenShift"
- "Red Hat OpenStack 플랫폼의 OpenShift"
- "Red Hat Virtualization의 OpenShift"
- "VMware vSphere에서의 OpenShift"

베어 메탈의 **OpenShift**

Bare Metal의 OpenShift는 상용 서버에 OpenShift 컨테이너 플랫폼을 자동으로 배포합니다.

Bare Metal의 OpenShift는 OpenShift의 가상 배포와 유사하며, 컨테이너화할 준비가 되지 않은 애플리케이션에 대한 가상화된 워크로드를 지원하는 동시에 OpenShift 클러스터의 배포 용이성, 빠른 프로비저닝 및 확장을 제공합니다. 베어 메탈에 배포하면 OpenShift 환경 외에 호스트 하이퍼바이저 환경을 관리하는 데 필요한 추가 오버헤드가 필요하지 않습니다. 베어 메탈 서버에 직접 배포하면 호스트와 OpenShift 환경 간에 리소스를 공유해야 하는 물리적 오버헤드 제한도 줄일 수 있습니다.

Bare Metal의 **OpenShift**는 다음과 같은 기능을 제공합니다**.**

- **IPI** 또는 지원 설치 프로그램 배포 베어 메탈 서버에 Installer Provisioned Infrastructure(IPI)를 통해 배포된 OpenShift 클러스터를 통해 고객은 하이퍼바이저 계층을 관리할 필요 없이 매우 다재다능하고 쉽게 확장 가능한 OpenShift 환경을 상용 서버에 직접 배포할 수 있습니다.
- 소형 클러스터 설계 하드웨어 요구 사항을 최소화하기 위해 베어 메탈 기반 OpenShift는 OpenShift 제어 평면 노드가 작업자 노드 및 호스트 컨테이너 역할도 수행할 수 있도록 하여 사용자가 3개의 노드만으로 구성된 클러스터를 배포할 수 있도록 합니다.
- **OpenShift** 가상화 OpenShift는 OpenShift 가상화를 사용하여 컨테이너 내에서 가상 머신을 실행할 수 있습니다. 이 컨테이너 기반 가상화는 컨테이너 내부에서 KVM 하이퍼바이저를 실행하고 VM 스토리지에 대한 영구 볼륨을 연결합니다.
- **AI/ML** 최적화 인프라 GPU 기반 워커 노드를 OpenShift 환경에 통합하고 OpenShift Advanced Scheduling을 활용하여 머신 러닝 애플리케이션을 위한 Kubeflow와 같은 애플리케이션을 배포합니다.

네트워크 디자인

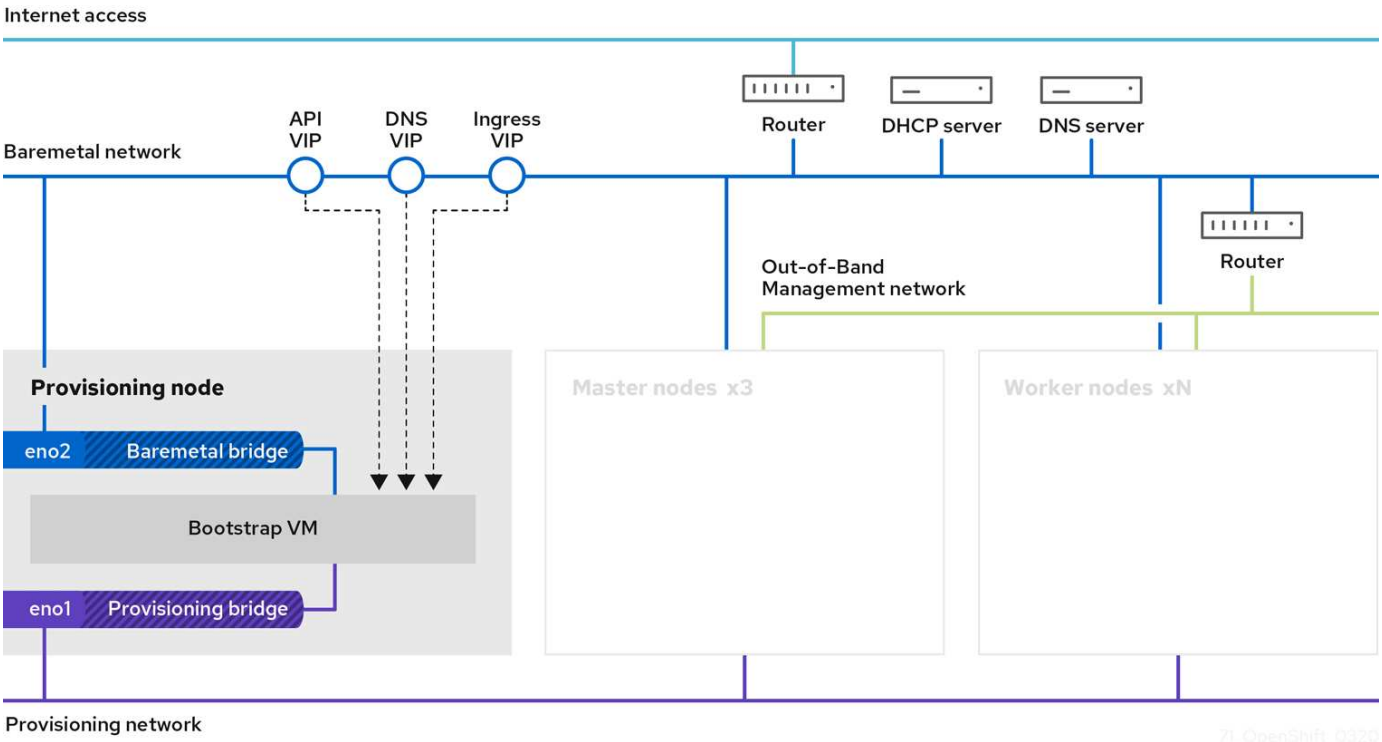
NetApp 솔루션의 Red Hat OpenShift는 두 개의 데이터 스위치를 사용하여 25Gbps의 기본 데이터 연결을 제공합니다. 또한 스토리지 노드의 대역 내 관리와 IPMI 기능의 대역 외 관리를 위해 1Gbps의 연결을 제공하는 두 개의 관리 스위치를 사용합니다.

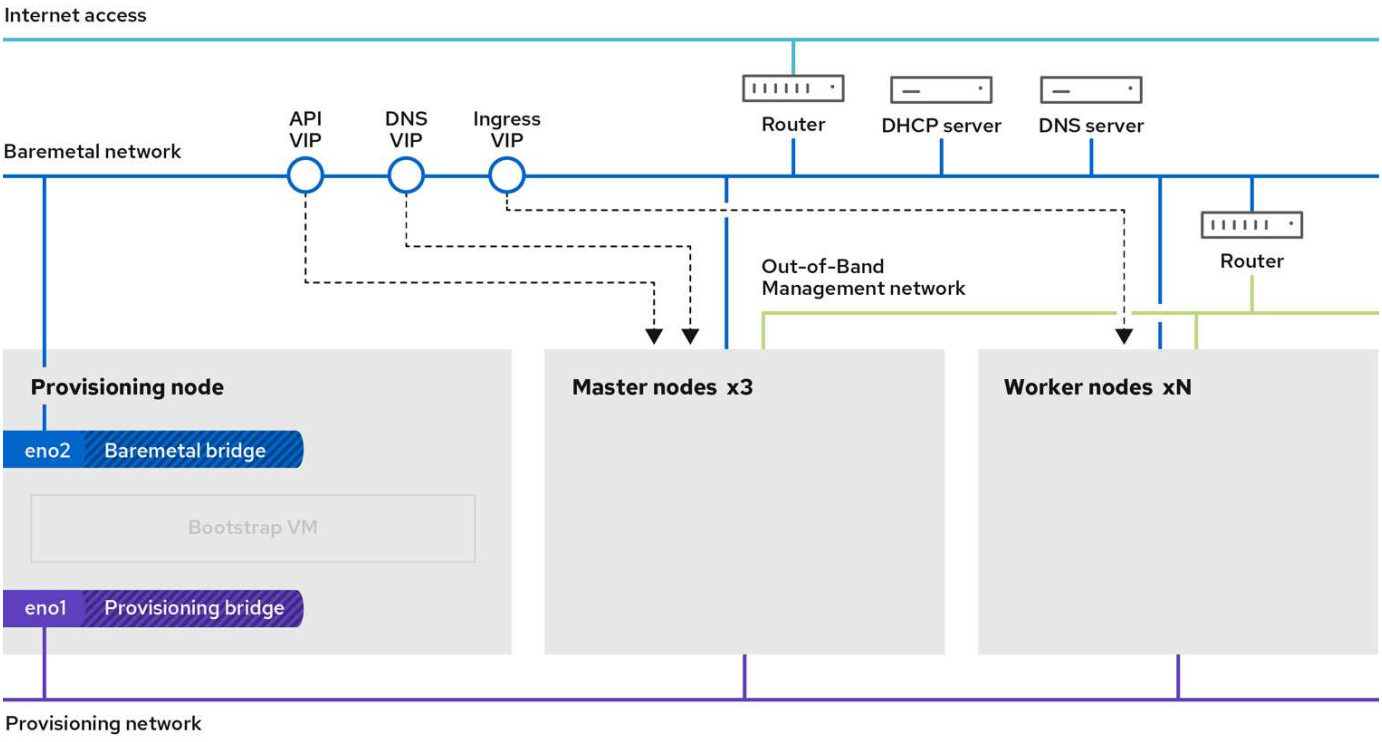
OpenShift 베어 메탈 IPI 배포의 경우 프로비저너 노드를 만들어야 합니다. 프로비저너 노드는 별도의 네트워크에 연결된 네트워크 인터페이스가 있는 Red Hat Enterprise Linux 8 머신입니다.

- 프로비저닝 네트워크 이 네트워크는 베어 메탈 노드를 부팅하고 OpenShift 클러스터를 배포하는 데 필요한 이미지와 패키지를 설치하는 데 사용됩니다.
- 베어메탈 네트워크 이 네트워크는 클러스터가 배포된 후 대중과 소통하는 데 사용됩니다.

프로비저너 노드를 설정하기 위해 고객은 트래픽이 노드 자체와 배포 목적으로 프로비저닝된 Bootstrap VM에서 적절하게 라우팅될 수 있도록 하는 브리지 인터페이스를 만듭니다. 클러스터가 배포된 후 API와 수신 VIP 주소가 부트스트랩 노드에서 새로 배포된 클러스터로 마이그레이션됩니다.

다음 이미지는 IPI 배포 중과 배포가 완료된 후의 환경을 보여줍니다.





VLAN 요구 사항

NetApp 솔루션이 포함된 Red Hat OpenShift는 가상 LAN(VLAN)을 사용하여 다양한 목적에 맞는 네트워크 트래픽을 논리적으로 분리하도록 설계되었습니다.

VLAN	목적	VLAN ID
대역 외 관리 네트워크	베어 메탈 노드 및 IPMI 관리	16
베어메탈 네트워크	클러스터가 사용 가능해지면 OpenShift 서비스를 위한 네트워크가 생성됩니다.	181
프로비저닝 네트워크	IPI를 통한 PXE 부팅 및 베어 메탈 노드 설치를 위한 네트워크	3485

이러한 각 네트워크는 VLAN으로 가상으로 분리되어 있지만, PXE 부팅 시퀀스 동안 VLAN 태그를 전달할 방법이 없기 때문에 각 물리적 포트는 기본 VLAN이 할당된 액세스 모드로 설정되어야 합니다.

네트워크 인프라 지원 리소스

OpenShift 컨테이너 플랫폼을 배포하기 전에 다음 인프라가 마련되어 있어야 합니다.

- 인밴드 관리 네트워크와 VM 네트워크에서 접근 가능한 전체 호스트 이름 확인을 제공하는 하나 이상의 DNS 서버.
- 인밴드 관리 네트워크와 VM 네트워크에서 접근할 수 있는 NTP 서버가 하나 이상 있어야 합니다.
- (선택 사항) 인밴드 관리 네트워크와 VM 네트워크 모두를 위한 아웃바운드 인터넷 연결.

Red Hat OpenStack 플랫폼의 OpenShift

Red Hat OpenStack Platform은 안전하고 안정적인 프라이빗 OpenStack 클라우드를 만들고, 배포하고, 확장할 수 있는 통합 기반을 제공합니다.

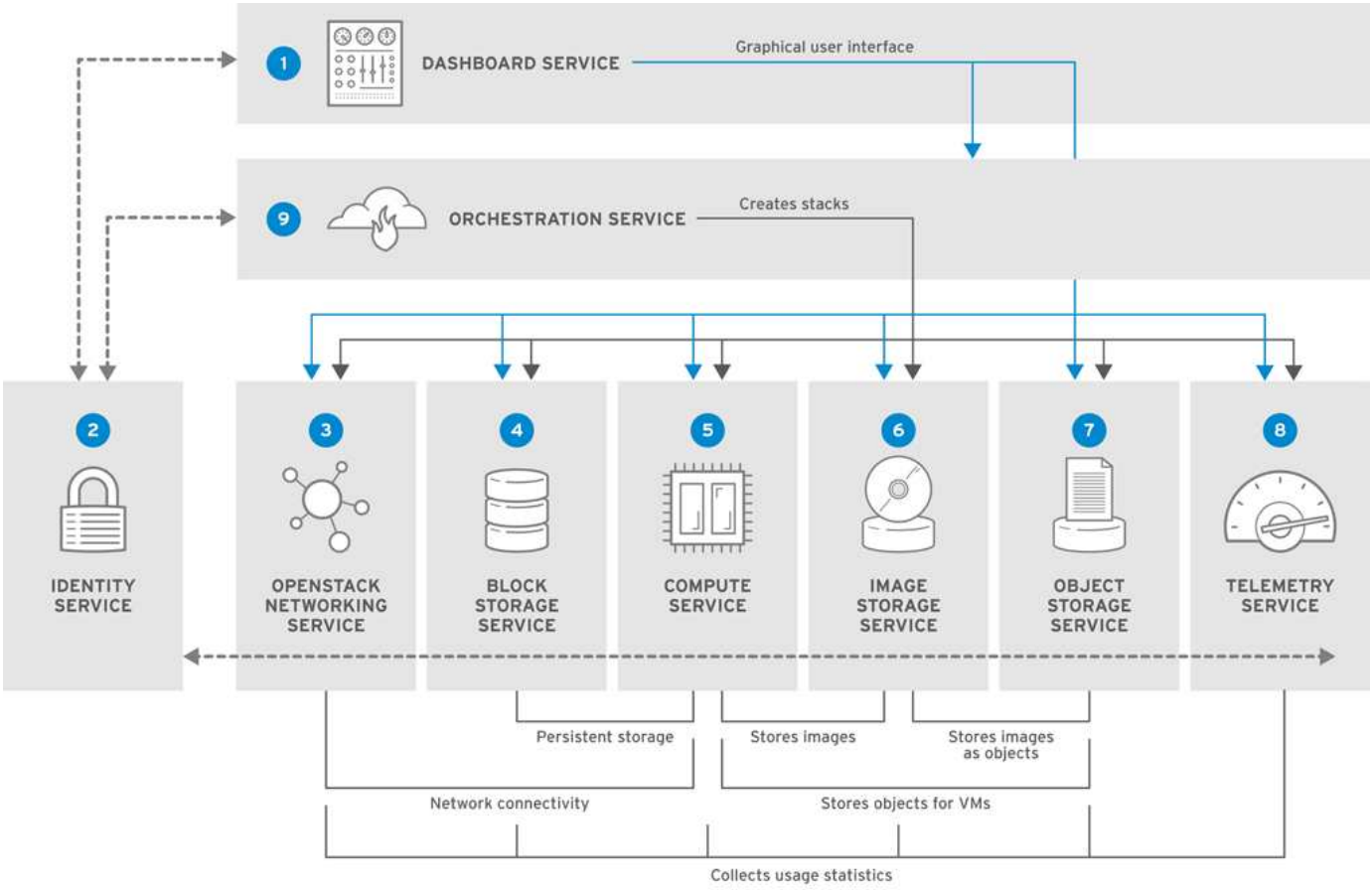
OSP는 컴퓨팅, 스토리지, 네트워킹 리소스를 관리하는 제어 서비스 모음으로 구현된 IaaS(Infrastructure-as-a-Service) 클라우드입니다. 환경은 관리자와 사용자가 OpenStack 리소스를 제어, 프로비저닝 및 자동화할 수 있는 웹 기반 인터페이스를 사용하여 관리됩니다. 또한, OpenStack 인프라는 광범위한 명령줄 인터페이스와 API를 통해 구축되어 관리자와 최종 사용자에게 완전한 자동화 기능을 제공합니다.

OpenStack 프로젝트는 6개월마다 업데이트된 릴리스를 제공하는 빠르게 개발되는 커뮤니티 프로젝트입니다. 초기에 Red Hat OpenStack Platform은 모든 업스트림 릴리스와 함께 새로운 릴리스를 게시하고 세 번째 릴리스마다 장기 지원을 제공함으로써 이 릴리스 주기를 따라잡았습니다. 최근 OSP 16.0 릴리스(OpenStack Train 기반)를 통해 Red Hat은 릴리스 번호 증가에 발맞추지 않고 대신 하위 릴리스에 새로운 기능을 백포트하기로 결정했습니다. 가장 최근에 출시된 버전은 Red Hat OpenStack Platform 16.1로, Ussuri 및 Victoria 릴리스 업스트림의 고급 기능이 백포트되어 포함되어 있습니다.

OSP에 대한 자세한 내용은 다음을 참조하세요."Red Hat OpenStack Platform 웹사이트" .

오픈스택 서비스

OpenStack Platform 서비스는 컨테이너로 배포되므로 서비스를 서로 분리하고 쉽게 업그레이드할 수 있습니다. OpenStack 플랫폼은 Kolla로 구축되고 관리되는 컨테이너 세트를 사용합니다. 서비스 배포는 Red Hat Custom Portal에서 컨테이너 이미지를 가져와서 수행됩니다. 이러한 서비스 컨테이너는 Podman 명령을 사용하여 관리되며 Red Hat OpenStack Director를 통해 배포, 구성 및 유지 관리됩니다.



서비스	프로젝트 이름	설명
계기반	수평선	OpenStack 서비스를 관리하는 데 사용하는 웹 브라우저 기반 대시보드입니다.
신원	Keystone	OpenStack 서비스의 인증 및 권한 부여와 사용자, 프로젝트, 역할 관리를 위한 중앙 집중식 서비스입니다.
오픈스택 네트워킹	중성자	OpenStack 서비스 인터페이스 간 연결을 제공합니다.
블록 스토리지	분석	가상 머신(VM)의 영구 블록 스토리지 볼륨을 관리합니다.
컴퓨팅	신성	컴퓨팅 노드에서 실행되는 VM을 관리하고 프로비저닝합니다.
영상	섬광	VM 이미지, 볼륨 스냅샷 등의 리소스를 저장하는 데 사용되는 레지스트리 서비스입니다.
객체 스토리지	스위프트	사용자가 파일과 임의의 데이터를 저장하고 검색할 수 있습니다.
원격 측정	운고계	클라우드 리소스 사용에 대한 측정을 제공합니다.
관현악법	열	리소스 스택의 자동 생성을 지원하는 템플릿 기반 오케스트레이션 엔진입니다.

네트워크 디자인

NetApp 솔루션이 포함된 Red Hat OpenShift는 두 개의 데이터 스위치를 사용하여 25Gbps의 기본 데이터 연결을 제공합니다. 또한 스토리지 노드의 대역 내 관리와 IPMI 기능의 대역 외 관리를 위해 1Gbps의 연결을 제공하는 두 개의 추가 관리 스위치를 사용합니다.

Red Hat OpenStack Director는 Ironic 베어 메탈 프로비저닝 서비스를 사용하여 Red Hat OpenStack Platform을 배포하는 데 IPMI 기능이 필요합니다.

VLAN 요구 사항

NetApp 탑재된 Red Hat OpenShift는 가상 LAN(VLAN)을 사용하여 다양한 목적에 맞는 네트워크 트래픽을 논리적으로 분리하도록 설계되었습니다. 이러한 구성은 고객 요구에 맞게 확장하거나 특정 네트워크 서비스에 대한 추가적인 격리를 제공하도록 확장할 수 있습니다. 다음 표는 NetApp 에서 솔루션을 검증하는 동안 솔루션을 구현하는 데 필요한 VLAN을 나열합니다.

VLAN	목적	VLAN ID
대역 외 관리 네트워크	Ironic의 물리적 노드와 IPMI 서비스를 관리하는 데 사용되는 네트워크입니다.	16
스토리지 인프라	Swift와 같은 인프라 서비스를 지원하기 위해 컨트롤러 노드가 볼륨을 직접 매핑하는 데 사용되는 네트워크입니다.	201
저장 신더	환경에 배포된 가상 인스턴스에 블록 볼륨을 직접 매핑하고 연결하는 데 사용되는 네트워크입니다.	202
내부 API	API 통신, RPC 메시지, 데이터베이스 통신을 사용하여 OpenStack 서비스 간 통신에 사용되는 네트워크입니다.	301
거주자	Neutron은 VXLAN을 통한 터널링을 통해 각 테넌트에게 자체 네트워크를 제공합니다. 네트워크 트래픽은 각 테넌트 네트워크 내에서 격리됩니다. 각 테넌트 네트워크에는 연결된 IP 서브넷이 있으며, 네트워크 네임스페이스는 여러 테넌트 네트워크가 충돌을 일으키지 않고 동일한 주소 범위를 사용할 수 있음을 의미합니다.	302
스토리지 관리	OpenStack Object Storage(Swift)는 이 네트워크를 사용하여 참여 복제 노드 간의 데이터 객체를 동기화합니다. 프록시 서비스는 사용자 요청과 기본 저장 계층 사이의 중개 인터페이스 역할을 합니다. 프록시는 들어오는 요청을 수신하고 요청된 데이터를 검색하는 데 필요한 복제본을 찾습니다.	303
PXE	OpenStack Director는 OSP Overcloud 설치를 조율하기 위해 Ironic 베어 메탈 프로비저닝 서비스의 일부로 PXE 부팅을 제공합니다.	3484
외부	그래픽 관리를 위한 OpenStack 대시보드(Horizon)를 호스팅하고 공개 API 호출을 통해 OpenStack 서비스를 관리할 수 있는 공개적으로 사용 가능한 네트워크입니다.	3485
인밴드 관리 네트워크	SSH 액세스, DNS 트래픽, NTP(네트워크 시간 프로토콜) 트래픽과 같은 시스템 관리 기능에 대한 액세스를 제공합니다. 이 네트워크는 컨트롤러가 아닌 노드에 대한 게이트웨이 역할도 합니다.	3486

네트워크 인프라 지원 리소스

OpenShift 컨테이너 플랫폼을 배포하기 전에 다음 인프라가 마련되어 있어야 합니다.

- 전체 호스트 이름 확인을 제공하는 하나 이상의 DNS 서버.
- 솔루션 내 서버의 시간을 동기화할 수 있는 NTP 서버가 최소 3개 있어야 합니다.
- (선택 사항) OpenShift 환경에 대한 아웃바운드 인터넷 연결.

프로덕션 배포를 위한 모범 사례

이 섹션에서는 조직이 이 솔루션을 프로덕션에 배포하기 전에 고려해야 할 몇 가지 모범 사례를 나열합니다.

최소 **3**개의 컴퓨팅 노드가 있는 **OSP** 프라이빗 클라우드에 **OpenShift**를 배포합니다**.**

이 문서에 설명된 검증된 아키텍처는 3개의 OSP 컨트롤러 노드와 2개의 OSP 컴퓨트 노드를 배포하여 HA 작업에 적합한 최소 하드웨어 배포를 제시합니다. 이 아키텍처는 두 컴퓨팅 노드 모두 가상 인스턴스를 시작하고 배포된 VM이 두 하이퍼바이저 간에 마이그레이션할 수 있는 장애 허용 구성을 보장합니다.

Red Hat OpenShift는 처음에 3개의 마스터 노드로 배포되므로 2노드 구성에서는 최소 2개의 마스터가 동일한 노드를 차지하게 될 수 있으며, 특정 노드를 사용할 수 없게 되면 OpenShift가 중단될 수 있습니다. 따라서 OpenShift 마스터를 균등하게 분산하고 솔루션의 내결함성을 높이기 위해 최소 3개의 OSP 컴퓨트 노드를 배포하는 것이 Red Hat의 모범 사례입니다.

가상 머신/호스트 친화성 구성

OpenShift 마스터를 여러 하이퍼바이저 노드에 분산하려면 VM/호스트 친화성을 활성화하면 됩니다.

친화성은 VM 및/또는 호스트 집합에 대한 규칙을 정의하는 방법으로, VM이 그룹 내의 동일한 호스트에서 함께 실행되는지 아니면 서로 다른 호스트에서 실행되는지 결정합니다. 동일한 매개변수와 조건 집합을 갖는 VM 및/또는 호스트로 구성된 친화성 그룹을 생성하여 VM에 적용됩니다. 친화성 그룹의 VM이 그룹 내의 동일한 호스트에서 실행되는지, 아니면 다른 호스트에서 별도로 실행되는지에 따라 친화성 그룹의 매개변수는 긍정적 친화성이나 부정적 친화성을 정의할 수 있습니다. Red Hat OpenStack Platform에서는 서버 그룹을 생성하고 필터를 구성하여 호스트 친화성 및 반친화성 규칙을 만들고 적용할 수 있습니다. 이를 통해 Nova가 서버 그룹에 배포한 인스턴스가 다른 컴퓨팅 노드에 배포됩니다.

서버 그룹은 기본적으로 최대 10개의 가상 인스턴스를 배치를 관리할 수 있습니다. 이는 Nova의 기본 할당량을 업데이트하여 수정할 수 있습니다.

OSP 서버 그룹에는 특정한 하드 친화성/반친화성 제한이 있습니다. 별도 노드에 배포할 리소스가 충분하지 않거나 노드 공유를 허용할 리소스가 충분하지 않으면 VM이 부팅되지 않습니다.

친화성 그룹을 구성하려면 다음을 참조하세요."OpenStack 인스턴스에 대해 Affinity와 Anti-Affinity를 어떻게 구성합니까?" .

OpenShift 배포를 위해 사용자 정의 설치 파일을 사용하세요

IPI는 이 문서의 앞부분에서 설명한 대화형 마법사를 통해 OpenShift 클러스터의 배포를 쉽게 만듭니다. 하지만 클러스터 배포의 일부로 일부 기본값을 변경해야 할 수도 있습니다.

이러한 경우, 클러스터를 즉시 배포하지 않고 마법사를 실행하여 작업을 지정할 수 있습니다. 대신 마법사는 나중에 클러스터를 배포할 수 있는 구성 파일을 생성합니다. 이는 IPI 기본 설정을 변경해야 하거나 멀티테넌시와 같은 다른 용도로 환경에 여러 개의 동일한 클러스터를 배포하려는 경우에 매우 유용합니다. OpenShift용 사용자 지정 설치 구성을 만드는 방법에 대한 자세한 내용은 다음을 참조하세요. "Red Hat OpenShift 사용자 정의를 사용하여 OpenStack에 클러스터 설치".

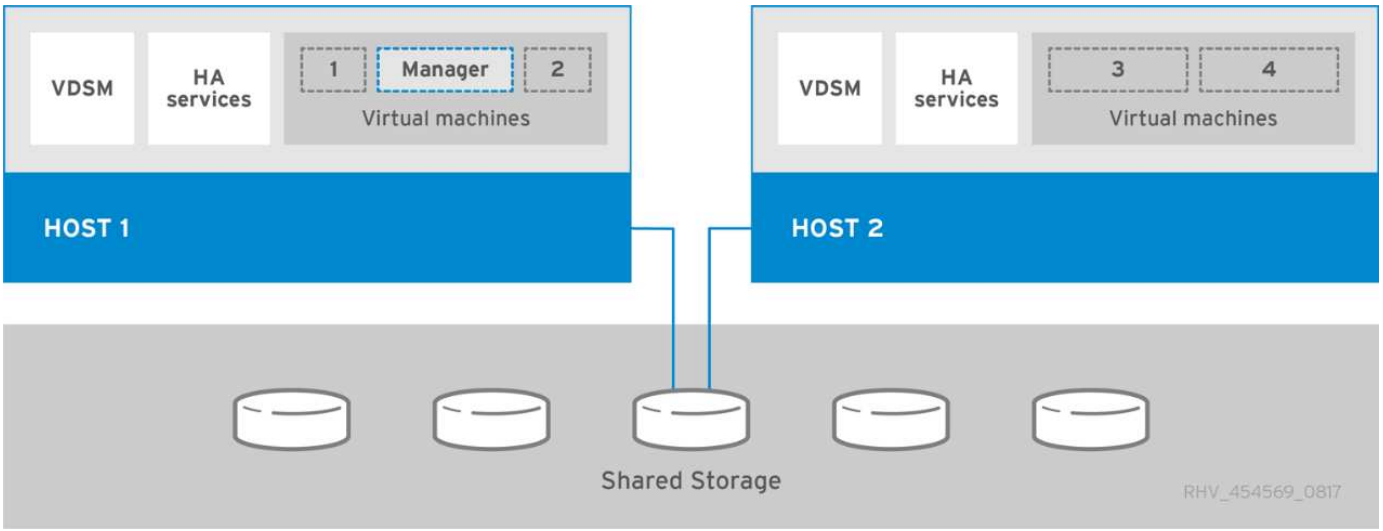
Red Hat Virtualization의 OpenShift

Red Hat Virtualization(RHV)은 Red Hat Enterprise Linux(RHEL)에서 실행되고 KVM 하이퍼바이저를 사용하는 엔터프라이즈 가상 데이터 센터 플랫폼입니다.

RHV에 대한 자세한 내용은 다음을 참조하세요."Red Hat Virtualization 웹사이트" .

RHV는 다음과 같은 기능을 제공합니다.

- **VM** 및 호스트의 중앙 집중식 관리 RHV 관리자는 배포 시 물리적 또는 가상 머신(VM)으로 실행되며 중앙 인터페이스에서 솔루션을 관리하기 위한 웹 기반 GUI를 제공합니다.
- 셀프 호스팅 엔진 하드웨어 요구 사항을 최소화하기 위해 RHV는 RHV Manager(RHV-M)를 게스트 VM을 실행하는 동일한 호스트에 VM으로 배포할 수 있도록 허용합니다.
- 고가용성 호스트 장애 발생 시 중단을 방지하기 위해 RHV에서는 VM을 고가용성으로 구성할 수 있습니다. 고가용성 VM은 복원력 정책을 사용하여 클러스터 수준에서 제어됩니다.
- 높은 확장성 단일 RHV 클러스터는 최대 200개의 하이퍼바이저 호스트를 가질 수 있으므로 리소스를 많이 필요로 하는 엔터프라이즈급 워크로드를 호스팅하는 대규모 VM의 요구 사항을 지원할 수 있습니다.
- 강화된 보안 RHV에서 상속받은 보안 가상화(sVirt) 및 보안 강화 Linux(SELinux) 기술은 RHV에서 호스트와 VM의 보안을 강화하고 강화하기 위해 사용됩니다. 이러한 기능의 주요 장점은 VM과 관련 리소스를 논리적으로 분리한다는 것입니다.



네트워크 디자인

NetApp 솔루션의 Red Hat OpenShift는 두 개의 데이터 스위치를 사용하여 25Gbps의 기본 데이터 연결을 제공합니다. 또한 스토리지 노드의 대역 내 관리와 IPMI 기능의 대역 외 관리를 위해 1Gbps의 연결을 제공하는 두 개의 추가 관리 스위치를 사용합니다. OCP는 클러스터 관리를 위해 RHV의 가상 머신 논리 네트워크를 사용합니다. 이 섹션에서는 솔루션에 사용된 각 가상 네트워크 세그먼트의 구성과 목적을 설명하고 솔루션을 배포하기 위한 전제 조건을 간략하게 설명합니다.

VLAN 요구 사항

RHV의 Red Hat OpenShift는 가상 LAN(VLAN)을 사용하여 다양한 목적에 맞는 네트워크 트래픽을 논리적으로 분리하도록 설계되었습니다. 이러한 구성은 고객 요구에 맞게 확장하거나 특정 네트워크 서비스에 대한 추가적인 격리를 제공하도록 확장할 수 있습니다. 다음 표는 NetApp 에서 솔루션을 검증하는 동안 솔루션을 구현하는 데 필요한 VLAN을 나열합니다.

VLAN	목적	VLAN ID
대역 외 관리 네트워크	물리적 노드 및 IPMI 관리	16

VLAN	목적	VLAN ID
VM 네트워크	가상 게스트 네트워크 액세스	1172
인밴드 관리 네트워크	RHV-H 노드, RHV-Manager 및 ovirtmgmt 네트워크 관리	3343
저장 네트워크	NetApp Element iSCSI용 스토리지 네트워크	3344
이주 네트워크	가상 게스트 마이그레이션을 위한 네트워크	3345

네트워크 인프라 지원 리소스

OpenShift 컨테이너 플랫폼을 배포하기 전에 다음 인프라가 마련되어 있어야 합니다.

- 인밴드 관리 네트워크와 VM 네트워크에서 접근 가능한 전체 호스트 이름 확인을 제공하는 하나 이상의 DNS 서버.
- 인밴드 관리 네트워크와 VM 네트워크에서 접근할 수 있는 NTP 서버가 하나 이상 있어야 합니다.
- (선택 사항) 인밴드 관리 네트워크와 VM 네트워크 모두를 위한 아웃바운드 인터넷 연결.

프로덕션 배포를 위한 모범 사례

이 섹션에서는 조직이 이 솔루션을 프로덕션에 배포하기 전에 고려해야 할 몇 가지 모범 사례를 나열합니다.

최소 **3**개 노드의 **RHV** 클러스터에 **OpenShift** 배포

이 문서에 설명된 검증된 아키텍처는 두 개의 RHV-H 하이퍼바이저 노드를 배포하고 두 호스트가 호스팅 엔진을 관리할 수 있고 배포된 VM이 두 하이퍼바이저 간에 마이그레이션할 수 있는 내결함성 구성을 보장함으로써 HA 작업에 적합한 최소 하드웨어 배포를 제시합니다.

Red Hat OpenShift는 처음에 3개의 마스터 노드로 배포되므로 2노드 구성에서는 최소 2개의 마스터가 동일한 노드를 차지하게 되며, 특정 노드를 사용할 수 없게 되면 OpenShift가 중단될 수 있습니다. 따라서 OpenShift 마스터를 균등하게 분산하고 솔루션의 내결함성을 높이기 위해 최소 3개의 RHV-H 하이퍼바이저 노드를 솔루션의 일부로 배포하는 것이 Red Hat의 모범 사례입니다.

가상 머신/호스트 친화성 구성

VM/호스트 친화성을 활성화하면 OpenShift 마스터를 여러 하이퍼바이저 노드에 분산할 수 있습니다.

친화성은 VM 및/또는 호스트 집합에 대한 규칙을 정의하는 방법으로, VM이 그룹 내의 동일한 호스트에서 함께 실행되는지 아니면 서로 다른 호스트에서 실행되는지 결정합니다. 동일한 매개변수와 조건 집합을 갖는 VM 및/또는 호스트로 구성된 친화성 그룹을 생성하여 VM에 적용됩니다. 친화성 그룹의 VM이 그룹 내의 동일한 호스트에서 실행되는지, 아니면 다른 호스트에서 별도로 실행되는지에 따라 친화성 그룹의 매개변수는 긍정적 친화성이나 부정적 친화성을 정의할 수 있습니다.

매개변수에 대해 정의된 조건은 하드 적용 또는 소프트 적용이 될 수 있습니다. 엄격한 시행은 친화도 그룹 내의 VM이 외부 조건에 관계없이 항상 긍정적 또는 부정적 친화도를 엄격하게 따르도록 보장합니다. 소프트 시행은 가능할 때마다 친화성 그룹 내 VM이 긍정적 또는 부정적 친화성을 따르도록 더 높은 기본 설정을 보장합니다. 이 문서에 설명된 2개 또는 3개의 하이퍼바이저 구성에서는 소프트 친화성이 권장되는 설정입니다. 대규모 클러스터에서는 하드 친화성을 통해 OpenShift 노드를 올바르게 분산할 수 있습니다.

친화성 그룹을 구성하려면 다음을 참조하세요."레드햇 6.11. Affinity Groups 문서" .

IPI는 이 문서의 앞부분에서 설명한 대화형 마법사를 통해 OpenShift 클러스터의 배포를 쉽게 만듭니다. 그러나 클러스터 배포의 일부로 변경해야 할 기본값이 있을 수 있습니다.

이런 경우, 클러스터를 즉시 배포하지 않고도 마법사를 실행하고 작업을 수행할 수 있습니다. 대신, 나중에 클러스터를 배포할 수 있는 구성 파일이 생성됩니다. 이 기능은 IPI 기본값을 변경하거나 다중 테넌시와 같은 다른 용도로 환경에 여러 개의 동일한 클러스터를 배포하려는 경우에 매우 유용합니다. OpenShift에 대한 사용자 정의 설치 구성을 만드는 방법에 대한 자세한 내용은 다음을 참조하세요."Red Hat OpenShift 사용자 지정을 사용하여 RHV에 클러스터 설치" .

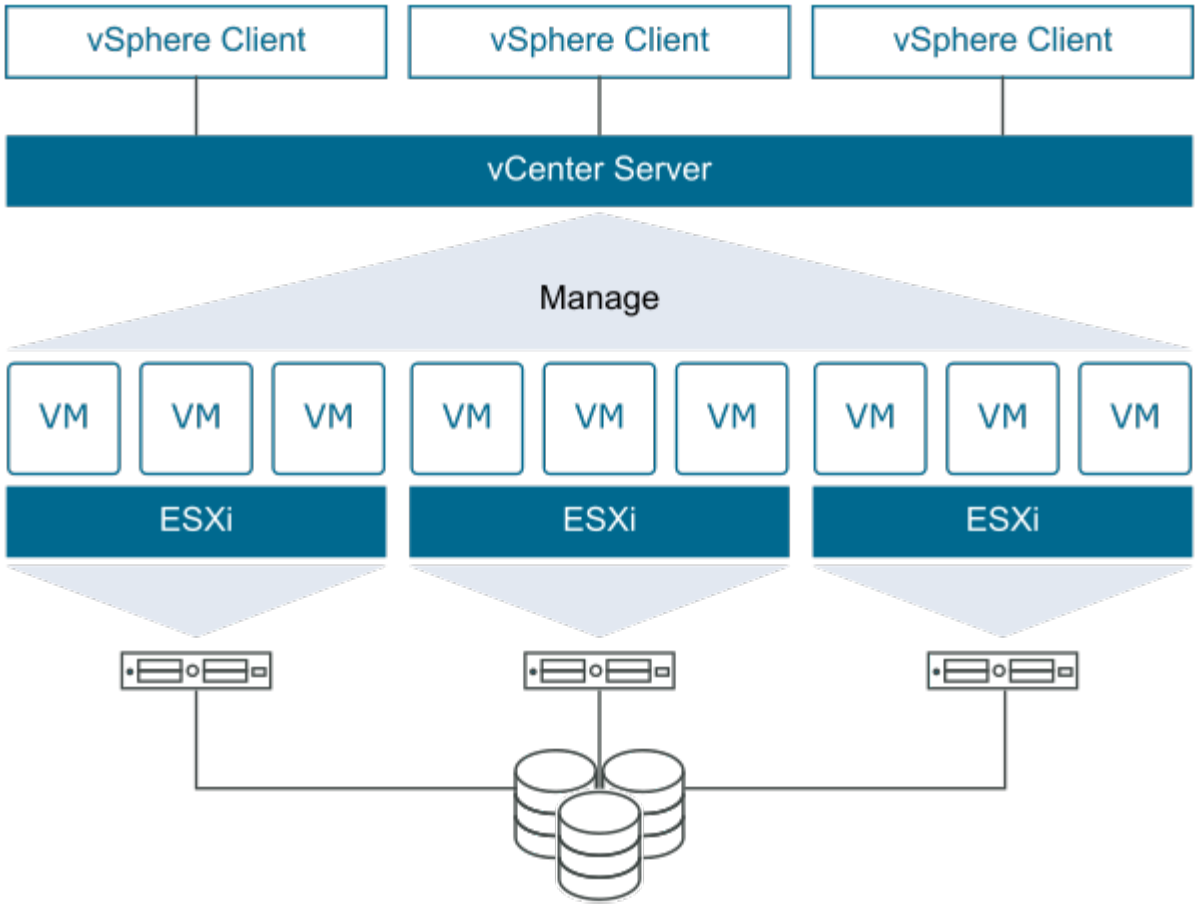
VMware vSphere에서의 OpenShift

VMware vSphere는 ESXi 하이퍼바이저에서 실행되는 수많은 가상화된 서버와 네트워크를 중앙에서 관리하기 위한 가상화 플랫폼입니다.

VMware vSphere에 대한 자세한 내용은 다음을 참조하세요."VMware vSphere 웹사이트" .

VMware vSphere는 다음과 같은 기능을 제공합니다.

- **VMware vCenter Server** VMware vCenter Server는 단일 콘솔에서 모든 호스트와 VM을 통합적으로 관리하고 클러스터, 호스트 및 VM의 성능 모니터링을 집계합니다.
- **VMware vSphere vMotion** VMware vCenter를 사용하면 중단 없이 요청에 따라 클러스터의 노드 간에 VM을 핫 마이그레이션할 수 있습니다.
- **vSphere** 고가용성 호스트 장애 발생 시 중단을 방지하기 위해 VMware vSphere를 사용하면 호스트를 클러스터링하고 고가용성을 위해 구성할 수 있습니다. 호스트 장애로 인해 중단된 VM은 클러스터의 다른 호스트에서 잠시 재부팅되어 서비스를 복구합니다.
- 분산 리소스 스케줄러**(DRS)** VMware vSphere 클러스터는 호스팅하는 VM의 리소스 요구 사항을 부하 분산하도록 구성할 수 있습니다. 리소스 경합이 발생하는 VM은 클러스터의 다른 노드로 핫 마이그레이션하여 충분한 리소스를 사용할 수 있는지 확인할 수 있습니다.



네트워크 디자인

NetApp 솔루션의 Red Hat OpenShift는 두 개의 데이터 스위치를 사용하여 25Gbps의 기본 데이터 연결을 제공합니다. 또한 스토리지 노드의 대역 내 관리와 IPMI 기능의 대역 외 관리를 위해 1Gbps의 연결을 제공하는 두 개의 추가 관리 스위치를 사용합니다. OCP는 클러스터 관리를 위해 VMware vSphere의 VM 논리 네트워크를 사용합니다. 이 섹션에서는 솔루션에 사용된 각 가상 네트워크 세그먼트의 구성과 목적을 설명하고 솔루션 배포에 필요한 전제 조건을 간략하게 설명합니다.

VLAN 요구 사항

VMware vSphere 기반 Red Hat OpenShift는 가상 LAN(Local Area Network)을 사용하여 다양한 목적에 맞는 네트워크 트래픽을 논리적으로 분리하도록 설계되었습니다. 이러한 구성은 고객 요구에 맞게 확장하거나 특정 네트워크 서비스에 대한 추가적인 격리를 제공하도록 확장할 수 있습니다. 다음 표는 NetApp 에서 솔루션을 검증하는 동안 솔루션을 구현하는 데 필요한 VLAN을 나열합니다.

VLAN	목적	VLAN ID
대역 외 관리 네트워크	물리적 노드 및 IPMI 관리	16
VM 네트워크	가상 게스트 네트워크 액세스	181
저장 네트워크	ONTAP NFS용 스토리지 네트워크	184
저장 네트워크	ONTAP iSCSI용 스토리지 네트워크	185
인밴드 관리 네트워크	ESXi 노드, VCenter Server, ONTAP Select 관리	3480

VLAN	목적	VLAN ID
저장 네트워크	NetApp Element iSCSI용 스토리지 네트워크	3481
이주 네트워크	가상 게스트 마이그레이션을 위한 네트워크	3482

네트워크 인프라 지원 리소스

OpenShift 컨테이너 플랫폼을 배포하기 전에 다음 인프라가 마련되어 있어야 합니다.

- 인밴드 관리 네트워크와 VM 네트워크에서 접근 가능한 전체 호스트 이름 확인을 제공하는 하나 이상의 DNS 서버.
- 인밴드 관리 네트워크와 VM 네트워크에서 접근할 수 있는 NTP 서버가 하나 이상 있어야 합니다.
- (선택 사항) 인밴드 관리 네트워크와 VM 네트워크 모두를 위한 아웃바운드 인터넷 연결.

프로덕션 배포를 위한 모범 사례

이 섹션에서는 조직이 이 솔루션을 프로덕션에 배포하기 전에 고려해야 할 몇 가지 모범 사례를 나열합니다.

최소 3개 노드의 **ESXi** 클러스터에 **OpenShift** 배포

이 문서에 설명된 검증된 아키텍처는 두 개의 ESXi 하이퍼바이저 노드를 배포하고 VMware vSphere HA 및 VMware vMotion을 활성화하여 장애 허용 구성을 보장함으로써 HA 작업에 적합한 최소 하드웨어 배포를 제시합니다. 이 구성을 사용하면 배포된 VM이 두 하이퍼바이저 간에 마이그레이션되고, 한 호스트를 사용할 수 없게 되면 재부팅할 수 있습니다.

Red Hat OpenShift는 처음에 3개의 마스터 노드로 배포되므로, 2노드 구성에서 최소 2개의 마스터가 특정 상황에서 동일한 노드를 점유할 수 있으며, 이로 인해 해당 노드를 사용할 수 없게 되면 OpenShift가 중단될 수 있습니다. 따라서 OpenShift 마스터를 균등하게 분산할 수 있도록 최소 3개의 ESXi 하이퍼바이저 노드를 배포하는 것이 Red Hat 모범 사례이며, 이를 통해 추가적인 수준의 내결함성을 제공합니다.

가상 머신 및 호스트 친화성 구성

VM 및 호스트 친화성을 활성화하면 OpenShift 마스터가 여러 하이퍼바이저 노드에 분산되도록 할 수 있습니다.

친화성 또는 반친화성은 VM 및/또는 호스트 집합에 대한 규칙을 정의하는 방법으로, VM이 그룹 내의 동일한 호스트 또는 호스트에서 함께 실행되는지 아니면 서로 다른 호스트에서 실행되는지 결정합니다. 동일한 매개변수와 조건 집합을 갖는 VM 및/또는 호스트로 구성된 친화성 그룹을 생성하여 VM에 적용됩니다. 친화성 그룹의 VM이 그룹 내의 동일한 호스트에서 실행되는지, 아니면 다른 호스트에서 별도로 실행되는지에 따라 친화성 그룹의 매개변수는 긍정적 친화성이나 부정적 친화성을 정의할 수 있습니다.

친화성 그룹을 구성하려면 다음을 참조하십시오. "vSphere 9.0 설명서: DRS 선호도 규칙 사용".

OpenShift 배포를 위해 사용자 정의 설치 파일을 사용하세요

IPI는 이 문서의 앞부분에서 설명한 대화형 마법사를 통해 OpenShift 클러스터의 배포를 쉽게 만듭니다. 하지만 클러스터 배포의 일부로 일부 기본값을 변경해야 할 수도 있습니다.

이런 경우 클러스터를 즉시 배포하지 않고도 마법사를 실행하고 작업을 지정할 수 있지만, 대신 마법사는 나중에 클러스터를 배포할 수 있는 구성 파일을 만듭니다. 이 기능은 IPI 기본값을 변경해야 하는 경우나 멀티테넌시와 같은 다른 용도로 환경에 여러 개의 동일한 클러스터를 배포하려는 경우에 매우 유용합니다. OpenShift에 대한 사용자 정의 설치 구성을 만드는 방법에 대한 자세한 내용은 다음을 참조하세요."Red Hat OpenShift 사용자 지정을 사용하여 vSphere에 클러스터 설치" .

AWS의 Red Hat OpenShift 서비스

AWS의 Red Hat OpenShift Service(ROSA)는 AWS에서 Red Hat OpenShift 엔터프라이즈 Kubernetes 플랫폼을 사용하여 컨테이너화된 애플리케이션을 빌드, 확장 및 배포하는 데 사용할 수 있는 관리형 서비스입니다. ROSA는 온프레미스 Red Hat OpenShift 워크로드를 AWS로 이전하는 과정을 간소화하고, 다른 AWS 서비스와의 긴밀한 통합을 제공합니다.

ROSA에 대한 자세한 내용은 여기의 설명서를 참조하세요."AWS의 Red Hat OpenShift 서비스(AWS 문서)" . "AWS의 Red Hat OpenShift 서비스(Red Hat 문서)" .

NetApp 스토리지 시스템

NetApp ONTAP

NetApp ONTAP 은 직관적인 GUI, 자동화 통합을 갖춘 REST API, AI 기반 예측 분석 및 시정 조치, 중단 없는 하드웨어 업그레이드, 스토리지 간 가져오기 등의 기능을 갖춘 강력한 스토리지 소프트웨어 도구입니다.

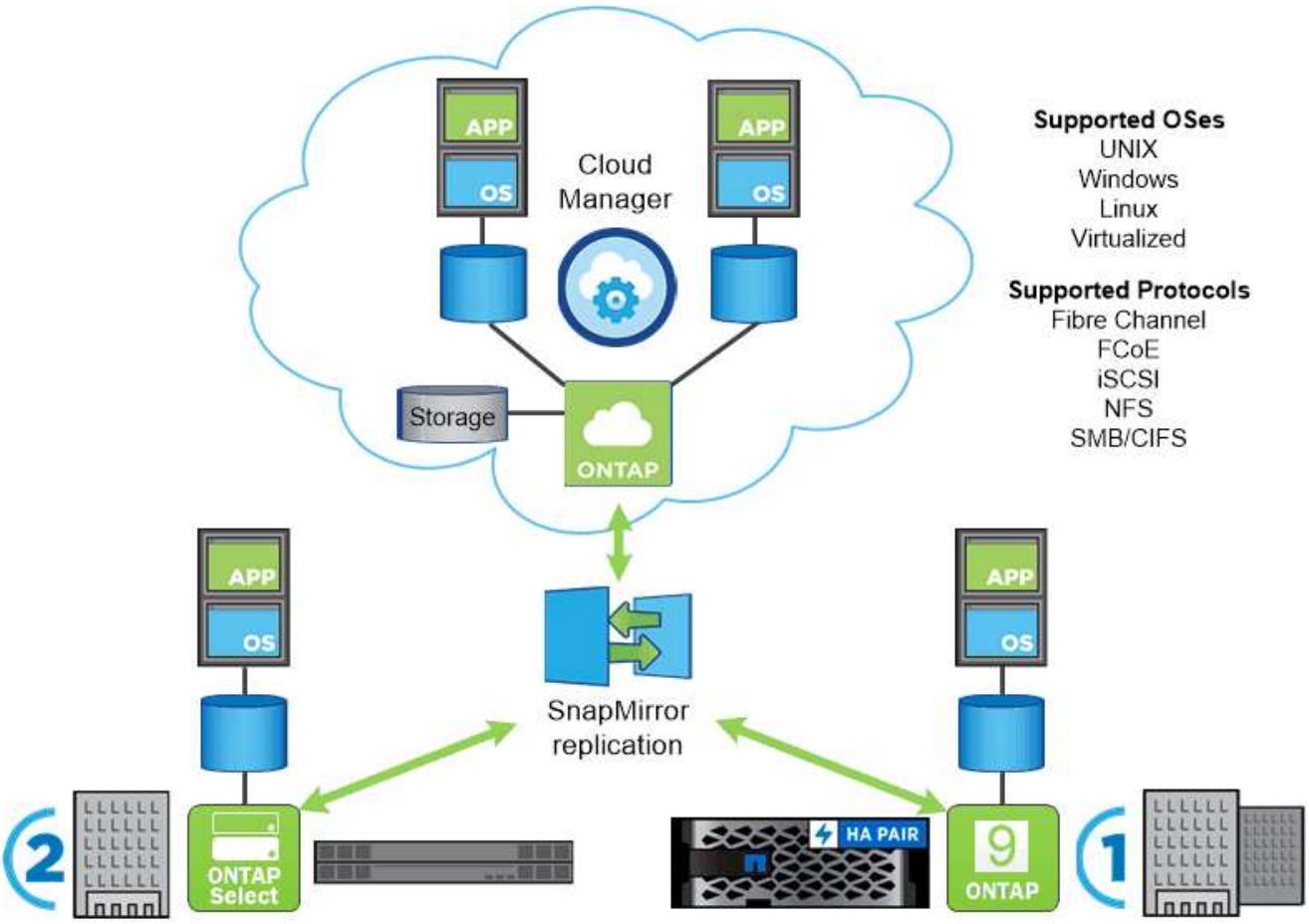
NetApp ONTAP 스토리지 시스템에 대한 자세한 내용은 다음을 방문하세요. "NetApp ONTAP 웹사이트" .

ONTAP 다음과 같은 기능을 제공합니다.

- NFS, CIFS, iSCSI, FC, FCoE, FC-NVMe 프로토콜에 대한 동시 데이터 액세스 및 관리 기능을 갖춘 통합 스토리지 시스템입니다.
- 다양한 배포 모델로는 온프레미스의 올플래시, 하이브리드, 올HDD 하드웨어 구성, ONTAP Select 와 같은 지원되는 하이퍼바이저의 VM 기반 스토리지 플랫폼, Cloud Volumes ONTAP 과 같은 클라우드가 있습니다.
- 자동 데이터 계층화, 인라인 데이터 압축, 중복 제거 및 압축을 지원하여 ONTAP 시스템의 데이터 저장 효율성이 향상되었습니다.
- 작업 부하 기반, QoS 제어 스토리지.
- 데이터의 계층화 및 보호를 위해 퍼블릭 클라우드와 완벽하게 통합됩니다. ONTAP 또한 어떤 환경에서도 차별화되는 강력한 데이터 보호 기능을 제공합니다.
    - * NetApp 스냅샷 복사본.* 추가적인 성능 오버헤드 없이 최소한의 디스크 공간을 사용하여 데이터를 빠르게 특정 시점에 백업합니다.
    - * NetApp SnapMirror.* 한 스토리지 시스템에서 다른 스토리지 시스템으로 데이터의 스냅샷 복사본을 미러링합니다. ONTAP 다른 물리적 플랫폼과 클라우드 기반 서비스에 대한 데이터 미러링을 지원합니다.
    - * NetApp SnapLock.* 지정된 기간 동안 덮어쓰거나 지울 수 없는 특수 볼륨에 데이터를 기록하여 다시 쓸 수 없는 데이터를 효율적으로 관리합니다.
    - * NetApp SnapVault.* 여러 저장 시스템의 데이터를 모든 지정된 시스템에 대한 백업 역할을 하는 중앙 스냅샷 사본으로 백업합니다.
    - * NetApp SyncMirror.* 동일한 컨트롤러에 물리적으로 연결된 두 개의 서로 다른 디스크 플렉스에 대한 데이터의 실시간 RAID 수준 미러링을 제공합니다.
    - * NetApp SnapRestore.* 스냅샷 복사본을 통해 필요에 따라 백업된 데이터를 빠르게 복원합니다.
    - * NetApp FlexClone.* 스냅샷 복사본을 기반으로 NetApp 볼륨의 완전히 읽고 쓸 수 있는 복사본을 즉시 프로비저닝합니다.

ONTAP 에 대한 자세한 내용은 다음을 참조하세요. "ONTAP 9 문서 센터" .

NetApp ONTAP 온프레미스, 가상화 또는 클라우드에서 사용할 수 있습니다.



NetApp 플랫폼

NetApp AFF/ FAS

NetApp 저지연 성능, 통합 데이터 보호, 다중 프로토콜 지원을 기반으로 맞춤 제작된 강력한 올플래시(AFF) 및 확장형 하이브리드(FAS) 스토리지 플랫폼을 제공합니다.

두 시스템 모두 NetApp ONTAP 데이터 관리 소프트웨어로 구동됩니다. 이 소프트웨어는 업계에서 가장 진보된 데이터 관리 소프트웨어로, 가용성이 높고 클라우드 통합이 가능하며 간소화된 스토리지 관리를 통해 데이터 패브릭에 필요한 엔터프라이즈급 속도, 효율성 및 보안을 제공합니다.

NETAPP AFF/ FAS 플랫폼에 대한 자세한 내용을 보려면 클릭하세요. "여기" .

ONTAP Select

ONTAP Select 사용자 환경의 하이퍼바이저에 배포할 수 있는 NetApp ONTAP 의 소프트웨어 정의 배포입니다. VMware vSphere 또는 KVM에 설치할 수 있으며 하드웨어 기반 ONTAP 시스템의 모든 기능과 경험을 제공합니다.

ONTAP Select 에 대한 자세한 내용을 보려면 클릭하세요. "여기" .

Cloud Volumes ONTAP

NetApp Cloud Volumes ONTAP Amazon AWS, Microsoft Azure, Google Cloud를 포함한 다양한 퍼블릭 클라우드에 배포할 수 있는 NetApp ONTAP 의 클라우드 배포 버전입니다.

Cloud Volumes ONTAP 에 대한 자세한 내용을 보려면 클릭하세요. "여기" .

Amazon FSx ONTAP

Amazon FSx ONTAP ONTAP 의 인기 있는 데이터 액세스 및 관리 기능을 갖춘 AWS 클라우드에서 완전 관리형 공유 스토리지를 제공합니다. Amazon FSx ONTAP 에 대한 자세한 내용을 보려면 클릭하세요. "여기" .

Azure NetApp Files

Azure NetApp Files 는 Azure 기반의 자체 개발 엔터프라이즈급 고성능 파일 스토리지 서비스입니다. NetApp 계정, 용량 풀, 볼륨을 생성할 수 있는 Volumes as a Service를 제공합니다. 서비스 및 성능 수준을 선택하고 데이터 보호를 관리할 수도 있습니다. 온프레미스에서 익숙하고 의존하는 것과 동일한 프로토콜과 도구를 사용하여 고성능, 고가용성, 확장 가능한 파일 공유를 만들고 관리할 수 있습니다. Azure NetApp Files 에 대한 자세한 내용을 보려면 클릭하세요. "여기" .

Google Cloud NetApp Volumes

Google Cloud NetApp Volumes 고급 데이터 관리 기능과 확장성이 뛰어난 성능을 제공하는 완전 관리형 클라우드 기반 데이터 스토리지 서비스입니다. 파일 기반 애플리케이션을 Google Cloud로 옮길 수 있습니다. 네트워크 파일 시스템(NFSv3 및 NFSv4.1)과 서버 메시지 블록(SMB) 프로토콜을 기본적으로 지원하므로 애플리케이션을 다시 설계할 필요가 없고 애플리케이션에 대한 영구 저장소를 계속 확보할 수 있습니다. Google Cloud NetApp VolumesP에 대한 자세한 내용을 보려면 클릭하세요. "여기" .

NetApp Element: NetApp 포함된 Red Hat OpenShift

NetApp Element 소프트웨어는 모듈식 확장 가능 성능을 제공하며, 각 스토리지 노드는 환경에 보장된 용량과 처리량을 제공합니다. NetApp Element 시스템은 단일 클러스터에서 4개에서 100개까지 노드를 확장할 수 있으며 다양한 고급 스토리지 관리 기능을 제공합니다.

NetApp Element 스토리지 시스템에 대한 자세한 내용은 다음을 방문하세요. "NetApp Solidfire 웹사이트" .

iSCSI 로그인 리디렉션 및 자체 복구 기능

NetApp Element 소프트웨어는 기존 TCP/IP 네트워크에서 SCSI 명령을 캡슐화하는 표준 방식인 iSCSI 스토리지 프로토콜을 활용합니다. SCSI 표준이 변경되거나 이더넷 네트워크의 성능이 향상되면 iSCSI 스토리지 프로토콜은 아무런 변경 없이도 이점을 얻습니다.

모든 스토리지 노드에는 관리 IP와 스토리지 IP가 있지만 NetApp Element 소프트웨어는 클러스터의 모든 스토리지

트래픽에 대해 단일 스토리지 가상 IP 주소(SVIP 주소)를 알립니다. iSCSI 로그인 프로세스의 일부로 스토리지는 대상 볼륨이 다른 주소로 이동되었으므로 협상 프로세스를 진행할 수 없다고 응답할 수 있습니다. 그런 다음 호스트는 호스트 측 재구성이 필요 없는 프로세스로 새 주소에 대한 로그인 요청을 다시 발행합니다. 이 프로세스를 iSCSI 로그인 리디렉션이라고 합니다.

iSCSI 로그인 리디렉션은 NetApp Element 소프트웨어 클러스터의 핵심 부분입니다. 호스트 로그인 요청이 수신되면 노드는 IOPS와 볼륨의 용량 요구 사항을 기준으로 클러스터의 어느 멤버가 트래픽을 처리해야 할지 결정합니다. 볼륨은 NetApp Element 소프트웨어 클러스터 전반에 분산되며, 단일 노드가 해당 볼륨에 대한 트래픽을 너무 많이 처리하거나 새 노드가 추가되는 경우 재분산됩니다. 주어진 볼륨의 여러 사본이 어레이 전체에 할당됩니다.

이런 방식으로 노드 장애가 발생한 후 볼륨이 재분배되는 경우 로그아웃하고 새 위치로 리디렉션하여 로그인하는 것 외에는 호스트 연결에 영향을 미치지 않습니다. iSCSI 로그인 리디렉션을 통해 NetApp Element 소프트웨어 클러스터는 중단 없는 업그레이드 및 운영이 가능한 자체 복구, 확장형 아키텍처입니다.

NetApp Element 소프트웨어 클러스터 QoS

NetApp Element 소프트웨어 클러스터를 사용하면 볼륨별로 QoS를 동적으로 구성할 수 있습니다. 정의한 SLA에 따라 볼륨별 QoS 설정을 사용하여 스토리지 성능을 제어할 수 있습니다. 다음 세 가지 구성 가능한 매개변수는 QoS를 정의합니다.

- 최소 **IOPS.** NetApp Element 소프트웨어 클러스터가 볼륨에 제공하는 지속형 IOPS의 최소 수입니다. 볼륨에 대해 구성된 최소 IOPS는 볼륨에 대해 보장되는 성능 수준입니다. 볼륨당 성능은 이 수준 이하로 떨어지지 않습니다.
- 최대 **IOPS.** NetApp Element 소프트웨어 클러스터가 특정 볼륨에 제공하는 지속형 IOPS의 최대 수입니다.
- 버스트 **IOPS.** 단기 버스트 시나리오에서 허용되는 최대 IOPS 수입니다. 버스트 지속 시간 설정은 구성 가능하며 기본값은 1분입니다. 볼륨이 최대 IOPS 수준보다 낮게 실행되면 버스트 크레딧이 누적됩니다. 성능 수준이 매우 높아지고 한계에 도달하면 볼륨에서 최대 IOPS를 넘는 짧은 IOPS 버스트가 허용됩니다.

멀티테넌시

안전한 멀티테넌시는 다음과 같은 기능을 통해 구현됩니다.

- 안전한 인증**.** CHAP(Challenge-Handshake 인증 프로토콜)는 안전한 볼륨 액세스에 사용됩니다. LDAP(Lightweight Directory Access Protocol)는 관리 및 보고를 위해 클러스터에 안전하게 액세스하는 데 사용됩니다.
- 볼륨 접근 그룹**(VAG).** 선택적으로 VAG를 인증 대신 사용하여 원하는 수의 iSCSI 이니시에이터별 iSCSI 정규 이름(IQN)을 하나 이상의 볼륨에 매핑할 수 있습니다. VAG의 볼륨에 액세스하려면 시작자의 IQN이 볼륨 그룹의 허용 IQN 목록에 있어야 합니다.
- 테넌트 가상 **LAN(VLAN).** 네트워크 수준에서 iSCSI 이니시에이터와 NetApp Element 소프트웨어 클러스터 간의 엔드투엔드 네트워크 보안은 VLAN을 사용하여 강화됩니다. 작업 부하나 테넌트를 격리하기 위해 생성된 모든 VLAN에 대해 NetApp Element Software는 해당 VLAN을 통해서만 액세스할 수 있는 별도의 iSCSI 대상 SVIP 주소를 생성합니다.
- **VRF** 지원 **VLAN.** 데이터 센터의 보안과 확장성을 더욱 지원하기 위해 NetApp Element 소프트웨어를 사용하면 모든 테넌트 VLAN에 VRF와 유사한 기능을 적용할 수 있습니다. 이 기능은 다음 두 가지 주요 기능을 추가합니다.
  - 테넌트 **SVIP** 주소로의 **L3** 라우팅**.** 이 기능을 사용하면 NetApp Element 소프트웨어 클러스터와 별도의 네트워크 또는 VLAN에 iSCSI 이니시에이터를 배치할 수 있습니다.
  - 중복되거나 중복된 **IP** 서브넷**.** 이 기능을 사용하면 테넌트 환경에 템플릿을 추가하여 각 테넌트 VLAN에 동일한 IP 서브넷의 IP 주소를 할당할 수 있습니다. 이 기능은 IP 공간의 규모와 보존이 중요한 서비스 제공자 환경에서 유용할 수 있습니다.

NetApp Element 소프트웨어 클러스터는 전반적인 스토리지 효율성과 성능을 향상시킵니다. 다음 기능은 인라인으로 수행되며 항상 켜져 있고 사용자가 수동으로 구성할 필요가 없습니다.

- 중복 제거. 이 시스템은 고유한 4K 블록만 저장합니다. 중복된 4K 블록은 자동으로 이미 저장된 데이터 버전에 연결됩니다. 데이터는 블록 드라이브에 저장되며 NetApp Element 소프트웨어 Helix 데이터 보호를 사용하여 미러링됩니다. 이 시스템은 시스템 내에서 용량 소모와 쓰기 작업을 크게 줄여줍니다.
- 압축. 압축은 데이터가 NVRAM 에 쓰여지기 전에 인라인으로 수행됩니다. 데이터는 압축되어 4K 블록으로 저장되며 시스템에 압축된 상태로 유지됩니다. 이러한 압축을 통해 클러스터 전체의 용량 소비, 쓰기 작업 및 대역폭 소비가 크게 줄어듭니다.
- 씬 프로비저닝. 이 기능은 필요할 때 적절한 양의 스토리지를 제공하여 과도하게 프로비저닝된 볼륨이나 활용도가 낮은 볼륨으로 인해 발생하는 용량 소모를 제거합니다.
- 나선. 개별 볼륨의 메타데이터는 메타데이터 드라이브에 저장되고 중복성을 위해 보조 메타데이터 드라이브에 복제됩니다.

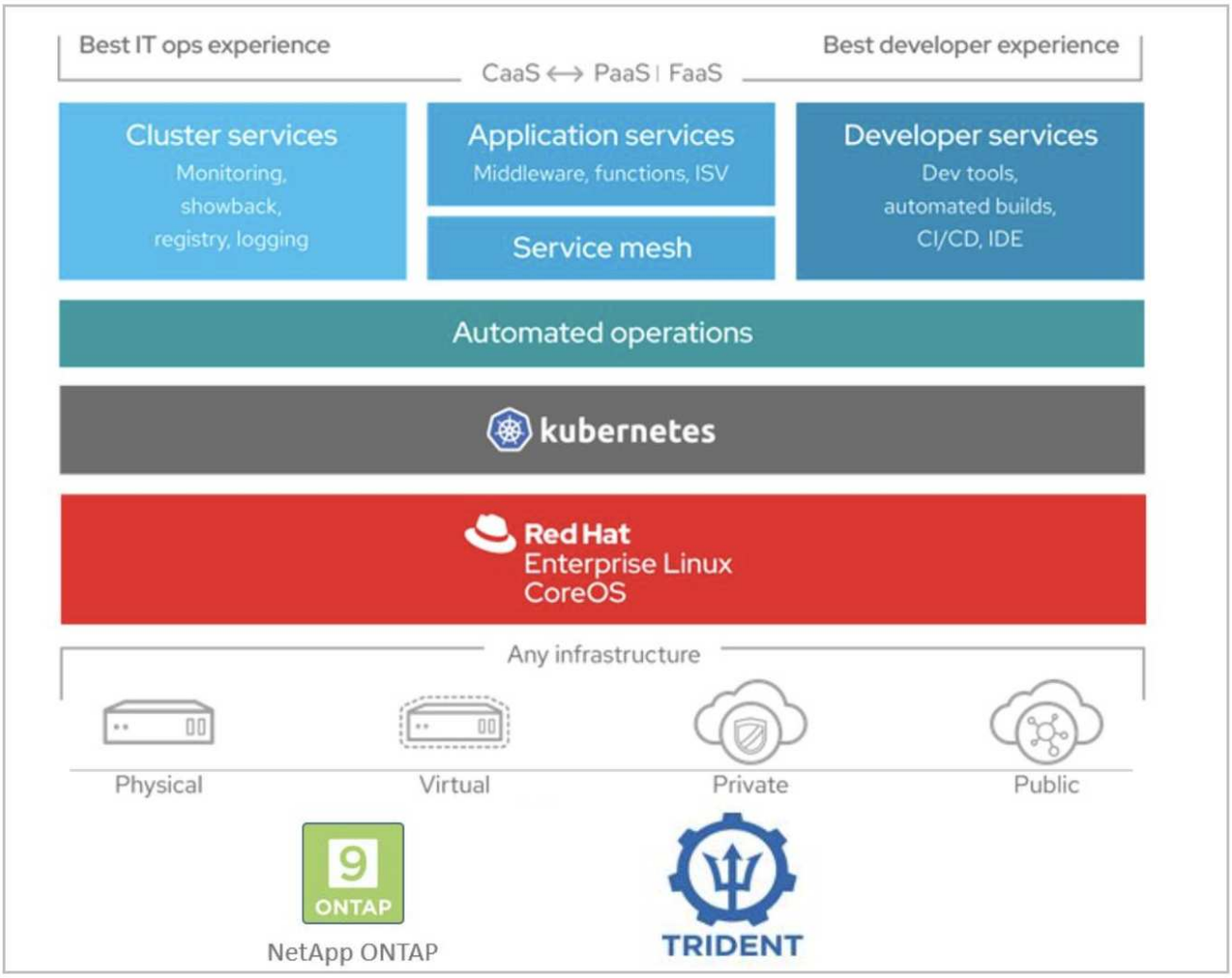
Element는 자동화를 위해 설계되었습니다. 모든 저장 기능은 API를 통해 사용할 수 있습니다. 이러한 API는 UI가 시스템을 제어하는 데 사용하는 유일한 방법입니다.

NetApp 스토리지 통합

Red Hat OpenShift와 NetApp Trident 통합에 대해 알아보세요

OpenShift Virtualization 솔루션을 위한 애플리케이션 및 영구 스토리지 관리에 대해 검증된 NetApp Trident Protect에 대해 알아보세요.

NetApp 과 NetApp Trident Protect가 유지 관리하는 오픈 소스 스토리지 프로비저닝 및 오케스트레이터 Trident 는 Red Hat OpenShift와 같은 컨테이너 기반 환경에서 영구 데이터를 오케스트레이션하고 관리하는 데 도움을 줍니다.



다음 페이지에는 NetApp 솔루션이 포함된 Red Hat OpenShift에서 애플리케이션 및 영구 스토리지 관리를 위해 검증된 NetApp 제품에 대한 추가 정보가 있습니다.

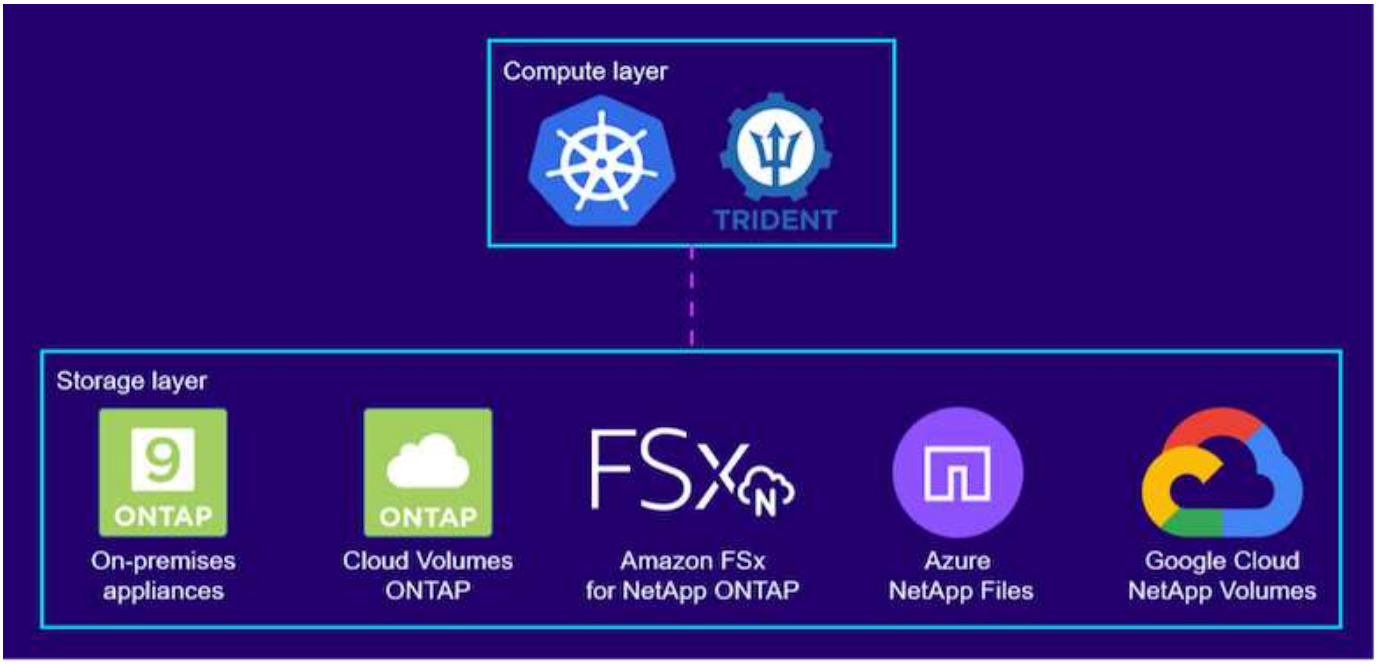
- "Trident 문서"
- "Trident 프로텍트 문서"

NetApp Trident

Trident 개요

Trident Red Hat OpenShift를 포함한 컨테이너와 Kubernetes 배포판을 위한 오픈 소스이자 완벽하게 지원되는 스토리지 오케스트레이터입니다. Trident NetApp ONTAP 및 Element 스토리지 시스템을 포함한 전체 NetApp 스토리지 포트폴리오와 호환되며 NFS 및 iSCSI 연결도 지원합니다. Trident 최종 사용자가 스토리지 관리자의 개입 없이 NetApp 스토리지 시스템에서 스토리지를 프로비저닝하고 관리할 수 있도록 하여 DevOps 워크플로를 가속화합니다.

관리자는 프로젝트 요구 사항과 압축, 특정 디스크 유형 또는 특정 수준의 성능을 보장하는 QoS 수준 등의 고급 스토리지 기능을 활성화하는 스토리지 시스템 모델을 기반으로 여러 스토리지 백엔드를 구성할 수 있습니다. 백엔드가 정의되면 개발자는 프로젝트에서 이러한 백엔드를 사용하여 영구 볼륨 클레임(PVC)을 생성하고 필요에 따라 영구 저장소를 컨테이너에 연결할 수 있습니다.



Trident 는 개발 주기가 빠르며, Kubernetes와 마찬가지로 1년에 4번 출시됩니다.

어떤 Kubernetes 배포판에서 어떤 버전의 Trident 테스트되었는지에 대한 지원 매트릭스를 찾을 수 있습니다. "여기" .

참고해주세요"Trident 제품 설명서" 설치 및 구성 세부 정보

Trident 다운로드

배포된 사용자 클러스터에 Trident 설치하고 영구 볼륨을 프로비저닝하려면 다음 단계를 완료하세요.

1. 설치 아카이브를 관리자 워크스테이션에 다운로드하고 내용을 추출합니다. Trident 의 현재 버전을 다운로드할 수 있습니다. "여기" .
2. 다운로드한 번들에서 Trident 설치를 추출합니다.

```
[netapp-user@rhel7 ~]$ tar -xzf trident-installer-22.01.0.tar.gz
[netapp-user@rhel7 ~]$ cd trident-installer/
[netapp-user@rhel7 trident-installer]$
```

Helm과 함께 Trident Operator 설치

1. 먼저 사용자 클러스터의 위치를 설정하세요. `kubeconfig` Trident 에는 이 파일을 전달하는 옵션이 없으므로, 이 파일을 참조할 필요가 없도록 환경 변수로 파일을 지정합니다.

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. 사용자 클러스터에 trident 네임스페이스를 생성하는 동안 helm 디렉토리의 tarball에서 Helm 명령을 실행하여 Trident 운영자를 설치합니다.

```
[netapp-user@rhel7 trident-installer]$ helm install trident
helm/trident-operator-22.01.0.tgz --create-namespace --namespace trident
NAME: trident
LAST DEPLOYED: Fri May  7 12:54:25 2021
NAMESPACE: trident
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing trident-operator, which will deploy and manage
NetApp's Trident CSI
storage provisioner for Kubernetes.

Your release is named 'trident' and is installed into the 'trident'
namespace.
Please note that there must be only one instance of Trident (and
trident-operator) in a Kubernetes cluster.

To configure Trident to manage storage resources, you will need a copy
of tridentctl, which is
available in pre-packaged Trident releases.  You may find all Trident
releases and source code
online at https://github.com/NetApp/trident.

To learn more about the release, try:

  $ helm status trident
  $ helm get all trident
```

3. 네임스페이스에서 실행 중인 포드를 확인하거나 tridentctl 바이너리를 사용하여 설치된 버전을 확인하여 Trident 성공적으로 설치되었는지 확인할 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-5z45l                   1/2     Running   2          30s
trident-csi-696b685cf8-htdb2        6/6     Running   0          30s
trident-csi-b74p2                   2/2     Running   0          30s
trident-csi-lrw4n                   2/2     Running   0          30s
trident-operator-7c748d957-gr2gw    1/1     Running   0          36s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0        | 22.01.0        |
+----------------+----------------+
```

어떤 경우에는 고객 환경에 따라 Trident 배포를 사용자 정의해야 할 수도 있습니다. 이러한 경우 Trident 운영자를 수동으로 설치하고 포함된 매니페스트를 업데이트하여 배포를 사용자 정의할 수도 있습니다.

Trident Operator를 수동으로 설치하세요

1. 먼저 사용자 클러스터의 위치를 설정합니다. `kubeconfig` Trident 에는 이 파일을 전달하는 옵션이 없으므로, 이 파일을 참조할 필요가 없도록 환경 변수로 파일을 지정합니다.

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. 그만큼 `trident-installer` 이 디렉토리에는 필요한 모든 리소스를 정의하는 매니페스트가 포함되어 있습니다. 적절한 매니페스트를 사용하여 다음을 생성합니다. `TridentOrchestrator` 사용자 정의 리소스 정의.

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
customresourcedefinition.apiextensions.k8s.io/tridentorchestrators.tride
nt.netapp.io created
```

3. 해당 네임스페이스가 없으면 제공된 매니페스트를 사용하여 클러스터에 Trident 네임스페이스를 만듭니다.

```
[netapp-user@rhel7 trident-installer]$ oc apply -f deploy/namespace.yaml
namespace/trident created
```

4. Trident 운영자 배포에 필요한 리소스(예: `ServiceAccount` 운영자에게는 `ClusterRole` 그리고 `ClusterRoleBinding` 에게 `ServiceAccount` , 헌신적인 `PodSecurityPolicy` , 또는 운영자 자체.

```
[netapp-user@rhel7 trident-installer]$ oc create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created
```

5. 다음 명령을 사용하면 배포된 후 운영자의 상태를 확인할 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc get deployment -n trident
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator   1/1     1            1           23s
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-operator-66f48895cc-lzczk   1/1     Running   0          41s
```

6. 운영자가 배포되었으므로 이제 이를 사용하여 Trident 설치할 수 있습니다. 이를 위해서는 다음을 생성해야 합니다. `TridentOrchestrator` .

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
[netapp-user@rhel7 trident-installer]$ oc describe torc trident
Name:         trident
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
Metadata:
  Creation Timestamp:  2021-05-07T17:00:28Z
  Generation:          1
  Managed Fields:
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:spec:
        .:
        f:debug:
        f:namespace:
    Manager:      kubectl-create
    Operation:    Update
    Time:         2021-05-07T17:00:28Z
```

```
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        .:
        f:currentInstallationParams:
          .:
          f:IPv6:
          f:autosupportHostname:
          f:autosupportimage:
          f:autosupportProxy:
          f:autosupportSerialNumber:
          f:debug:
          f:enableNodePrep:
          f:imagePullSecrets:
          f:imageRegistry:
          f:k8sTimeout:
          f:kubeletDir:
          f:logFormat:
          f:silenceAutosupport:
          f:tridentimage:
        f:message:
        f:namespace:
        f:status:
        f:version:
    Manager:         trident-operator
    Operation:       Update
    Time:            2021-05-07T17:00:28Z
  Resource Version:  931421
  Self Link:
/apis/trident.netapp.io/v1/tridentorchestrators/trident
  UID:               8a26a7a6-dde8-4d55-9b66-a7126754d81f
Spec:
  Debug:      true
  Namespace:  trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport image:         netapp/trident-autosupport:21.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    Image Pull Secrets:
    Image Registry:
```

```
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident image:        netapp/trident:22.01.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v22.01.0
Events:
  Type    Reason      Age   From                           Message
  ----    ------      ----  ----                           -------
  Normal  Installing  80s   trident-operator.netapp.io  Installing
Trident
  Normal  Installed   68s   trident-operator.netapp.io  Trident
installed
```

7. 네임스페이스에서 실행 중인 포드를 확인하거나 tridentctl 바이너리를 사용하여 설치된 버전을 확인하여 Trident 성공적으로 설치되었는지 확인할 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-bb64c6cb4-lmd6h         6/6     Running   0          82s
trident-csi-gn59q                   2/2     Running   0          82s
trident-csi-m4szj                   2/2     Running   0          82s
trident-csi-sb9k9                   2/2     Running   0          82s
trident-operator-66f48895cc-lzczk   1/1     Running   0          2m39s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0          | 22.01.0          |
+----------------+----------------+
```

저장을 위한 작업자 노드 준비

NFS

대부분의 Kubernetes 배포판에는 Red Hat OpenShift를 포함하여 기본적으로 설치된 NFS 백엔드를 마운트하기 위한 패키지와 유틸리티가 함께 제공됩니다.

하지만 NFSv3의 경우 클라이언트와 서버 간의 동시성을 협상하는 메커니즘이 없습니다. 따라서 NFS 연결에 대한 최상의 성능을 보장하려면 서버에서 지원하는 값과 클라이언트 측 sunrpc 슬롯 테이블 항목의 최대 개수를 수동으로 동기화해야 하며, 이를 위해 서버가 연결 창 크기를 줄여야 합니다.

ONTAP 의 경우 지원되는 최대 sunrpc 슬롯 테이블 항목 수는 128개입니다. 즉, ONTAP 한 번에 128개의 동시 NFS 요청을 처리할 수 있습니다. 그러나 기본적으로 Red Hat CoreOS/Red Hat Enterprise Linux는 연결당 최대 65,536개의 sunrpc 슬롯 테이블 항목을 갖습니다. 이 값을 128로 설정해야 하며, 이는 OpenShift의 Machine Config Operator(MCO)를 사용하여 수행할 수 있습니다.

OpenShift 워커 노드에서 최대 sunrpc 슬롯 테이블 항목을 수정하려면 다음 단계를 완료하세요.

1. OCP 웹 콘솔에 로그인하고 컴퓨팅 > 머신 구성으로 이동합니다. 머신 구성 만들기를 클릭합니다. YAML 파일을 복사하여 붙여넣고 만들기를 클릭합니다.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 98-worker-nfs-rpc-slot-tables
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - contents:
            source: data:text/plain;charset=utf-
8;base64,b3B0aW9ucyBzdW5ycGMgdGNwX21heF9zbG90X3RhYmxlX2VudHJpZXM9MTI4Cg=
=
          filesystem: root
          mode: 420
          path: /etc/modprobe.d/sunrpc.conf
```

2. MCO가 생성된 후에는 모든 워커 노드에 구성을 적용하고 하나씩 재부팅해야 합니다. 전체 과정은 약 20~30분 정도 걸립니다. 다음을 사용하여 머신 구성이 적용되는지 확인하세요. `oc get mcp` 그리고 작업자의 머신 구성 풀이 업데이트되었는지 확인하세요.

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                             UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False
```

iSCSI

iSCSI 프로토콜을 통해 블록 스토리지 볼륨을 매핑할 수 있도록 작업자 노드를 준비하려면 해당 기능을 지원하는 데 필요한 패키지를 설치해야 합니다.

Red Hat OpenShift에서는 배포 후 클러스터에 MCO(Machine Config Operator)를 적용하여 이를 처리합니다.

iSCSI 서비스를 실행하도록 작업자 노드를 구성하려면 다음 단계를 완료하세요.

1. OCP 웹 콘솔에 로그인하고 컴퓨팅 > 머신 구성으로 이동합니다. 머신 구성 만들기를 클릭합니다. YAML 파일을 복사하여 붙여넣고 만들기를 클릭합니다.

   멀티패스를 사용하지 않을 때:

   ```
   apiVersion: machineconfiguration.openshift.io/v1
   kind: MachineConfig
   metadata:
     labels:
       machineconfiguration.openshift.io/role: worker
     name: 99-worker-element-iscsi
   spec:
     config:
       ignition:
         version: 3.2.0
       systemd:
         units:
           - name: iscsid.service
             enabled: true
             state: started
     osImageURL: ""
   ```

   멀티패스를 사용하는 경우:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 99-worker-ontap-iscsi
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgbm8KICAgICA
gICBmaW5kX211bHRpcGF0aHMgbm8KfQoKYmxhY2tsaXN0X2V4Y2VwdGlvbnMgewogICAgICA
gIHByb3BlcnR5ICIoU0NTSV9JREVOVF98SURfV1dOKSIKfQoKYmxhY2tsaXN0IHsKfQoK
          verification: {}
        filesystem: root
        mode: 400
        path: /etc/multipath.conf
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
        - name: multipathd.service
          enabled: true
          state: started
  osImageURL: ""
```

2. 구성이 생성된 후, 워커 노드에 구성을 적용하고 다시 로드하는 데 약 20~30분이 걸립니다. 다음을 사용하여 머신 구성이 적용되는지 확인하세요. `oc get mcp` 그리고 작업자의 머신 구성 풀이 업데이트되었는지 확인하세요. 또한 작업자 노드에 로그인하여 iscsid 서비스가 실행 중인지 확인할 수 있습니다(멀티패스를 사용하는 경우 multipathd 서비스도 실행 중인지 확인할 수 있습니다).

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                             UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status iscsid
● iscsid.service - Open-iSCSI
   Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled;
vendor preset: disabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
     Docs: man:iscsid(8)
           man:iscsiadm(8)
 Main PID: 1242 (iscsid)
   Status: "Ready to process requests"
    Tasks: 1
   Memory: 4.9M
      CPU: 9ms
   CGroup: /system.slice/iscsid.service
           └─1242 /usr/sbin/iscsid -f

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status multipathd
 ● multipathd.service - Device-Mapper Multipath Device Controller
   Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled;
vendor preset: enabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
  Main PID: 918 (multipathd)
    Status: "up"
    Tasks: 7
    Memory: 13.7M
    CPU: 57ms
    CGroup: /system.slice/multipathd.service
            └─918 /sbin/multipathd -d -s
```

MachineConfig가 성공적으로 적용되었고 서비스가 예상대로 시작되었는지 확인하려면 다음을 실행하세요. `oc debug` 적절한 플래그를 사용하여 명령을 실행합니다.

스토리지 시스템 백엔드 생성

Trident Operator 설치를 완료한 후에는 사용 중인 특정 NetApp 스토리지 플랫폼에 대한 백엔드를 구성해야 합니다. Trident 의 설정 및 구성을 계속하려면 아래 링크를 따르세요.

- "NetApp ONTAP NFS"
- "NetApp ONTAP iSCSI"
- "NetApp Element iSCSI"

NetApp ONTAP NFS 구성

NetApp ONTAP 스토리지 시스템과 Trident 통합을 활성화하려면 스토리지 시스템과의 통신을 지원하는 백엔드를 만들어야 합니다.

1. 다운로드한 설치 아카이브에는 샘플 백엔드 파일이 있습니다. `sample-input` 폴더 계층구조. NFS를 제공하는 NetApp ONTAP 시스템의 경우 다음을 복사하십시오. `backend-ontap-nas.json` 작업 디렉토리에 파일을 복사하고 편집하세요.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-nas/backend-ontap-nas.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-nas.json
```

2. 이 파일에서 backendName, managementLIF, dataLIF, svm, username 및 password 값을 편집합니다.

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap-nas+10.61.181.221",
    "managementLIF": "172.21.224.201",
    "dataLIF": "10.61.181.221",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "password"
}
```

> ⓘ 쉽게 식별할 수 있도록 NFS를 제공하는 storageDriverName과 dataLIF의 조합으로 사용자 지정 backendName 값을 정의하는 것이 가장 좋습니다.

3. 백엔드 파일이 준비되면 다음 명령을 실행하여 첫 번째 백엔드를 만듭니다.

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-nas.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE   | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas+10.61.181.221 | ontap-nas      | be7a619d-c81d-445c-b80c-
5c87a73c5b1e | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. 백엔드를 만든 후에는 다음으로 스토리지 클래스를 만들어야 합니다. 백엔드와 마찬가지로, sample-inputs 폴더에서 해당 환경에 맞게 편집할 수 있는 샘플 스토리지 클래스 파일이 있습니다. 작업 디렉토리에 복사하고 생성된 백엔드를 반영하도록 필요한 편집을 합니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 이 파일에 대해 해야 할 유일한 편집은 다음을 정의하는 것입니다. `backendType` 새로 생성된 백엔드의 스토리지 드라이버 이름에 대한 값입니다. 또한, 이후 단계에서 참조해야 하는 이름 필드 값도 기록해 둡니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
```

라는 선택적 필드가 있습니다. `fsType` 이 파일에 정의되어 있습니다. 이 줄은 NFS 백엔드에서 삭제할 수 있습니다.

6. 실행하다 `oc` 저장 클래스를 생성하는 명령입니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. 스토리지 클래스가 생성되면 첫 번째 영구 볼륨 클레임(PVC)을 생성해야 합니다. 샘플이 있습니다 `pvc-`

`basic.yaml` 이 작업을 수행하는 데 사용할 수 있는 파일도 sample-inputs에 있습니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. 이 파일에 대해 수행해야 하는 유일한 편집은 다음을 보장하는 것입니다. `storageClassName` 필드가 방금 만든 필드와 일치합니다. PVC 정의는 프로비저닝할 작업 부하에 따라 추가로 사용자 정의할 수 있습니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. PVC를 생성하려면 다음을 실행하세요. `oc` 명령. 생성되는 백업 볼륨의 크기에 따라 생성에 시간이 걸릴 수 있으므로, 프로세스가 완료되는 모습을 지켜볼 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-b4370d37-0fa4-4c17-bd86-94f96c94b42d   1Gi
RWO            basic-csi      7s
```

NetApp ONTAP iSCSI 구성

NetApp ONTAP 스토리지 시스템과 Trident 통합을 활성화하려면 스토리지 시스템과의 통신을 지원하는 백엔드를 만들어야 합니다.

1. 다운로드한 설치 아카이브에는 샘플 백엔드 파일이 있습니다. `sample-input` 폴더 계층구조. iSCSI를 제공하는 NetApp ONTAP 시스템의 경우 다음을 복사하십시오. `backend-ontap-san.json` 작업 디렉토리에 파일을 복사하고 편집하세요.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-san/backend-ontap-san.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-san.json
```

2. 이 파일에서 managementLIF, dataLIF, svm, username 및 password 값을 편집합니다.

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "172.21.224.201",
  "dataLIF": "10.61.181.240",
  "svm": "trident_svm",
  "username": "admin",
  "password": "password"
}
```

3. 백엔드 파일이 준비되면 다음 명령을 실행하여 첫 번째 백엔드를 만듭니다.

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-san.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontapsan_10.61.181.241 | ontap-san      | 6788533c-7fea-4a35-b797-
fb9bb3322b91 | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. 백엔드를 만든 후에는 다음으로 스토리지 클래스를 만들어야 합니다. 백엔드와 마찬가지로, sample-inputs 폴더에서 해당 환경에 맞게 편집할 수 있는 샘플 스토리지 클래스 파일이 있습니다. 작업 디렉토리에 복사하고 생성된 백엔드를 반영하도록 필요한 편집을 합니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 이 파일에 대해 해야 할 유일한 편집은 다음을 정의하는 것입니다. `backendType` 새로 생성된 백엔드의 스토리지 드라이버 이름에 대한 값입니다. 또한, 이후 단계에서 참조해야 하는 이름 필드 값도 기록해 둡니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
```

라는 선택적 필드가 있습니다. `fsType` 이 파일에 정의되어 있습니다. iSCSI 백엔드에서 이 값은 특정 Linux 파일 시스템 유형(XFS, ext4 등)으로 설정하거나 삭제하여 OpenShift가 사용할 파일 시스템을 결정하도록 할 수 있습니다.

6. 실행하다 `oc` 저장 클래스를 생성하는 명령입니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. 스토리지 클래스가 생성되면 첫 번째 영구 볼륨 클레임(PVC)을 생성해야 합니다. 샘플이 있습니다 `pvc-basic.yaml` 이 작업을 수행하는 데 사용할 수 있는 파일도 sample-inputs에 있습니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. 이 파일에 대해 수행해야 하는 유일한 편집은 다음을 보장하는 것입니다. `storageClassName` 필드가 방금 만든 필드와 일치합니다. PVC 정의는 프로비저닝할 작업 부하에 따라 추가로 사용자 정의할 수 있습니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. PVC를 생성하려면 다음을 실행하세요. `oc` 명령. 생성되는 백업 볼륨의 크기에 따라 생성에 시간이 걸릴 수 있으므로, 프로세스가 완료되는 모습을 지켜볼 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-7ceac1ba-0189-43c7-8f98-094719f7956c   1Gi
RWO            basic-csi      3s
```

NetApp Element iSCSI 구성

NetApp Element 스토리지 시스템과 Trident 통합을 활성화하려면 iSCSI 프로토콜을 사용하여 스토리지 시스템과 통신할 수 있는 백엔드를 만들어야 합니다.

1. 다운로드한 설치 아카이브에는 샘플 백엔드 파일이 있습니다. `sample-input` 폴더 계층구조. iSCSI를 제공하는 NetApp Element 시스템의 경우 다음을 복사하세요. `backend-solidfire.json` 작업 디렉토리에 파일을 복사하고 편집하세요.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/solidfire/backend-solidfire.json ./
[netapp-user@rhel7 trident-installer]$ vi ./backend-solidfire.json
```

   a. 사용자, 비밀번호 및 MVIP 값을 편집합니다. `EndPoint` 선.

   b. 편집하다 `SVIP` 값.

```
 {
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://trident:password@172.21.224.150/json-
rpc/8.0",
    "SVIP": "10.61.180.200:3260",
    "TenantName": "trident",
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS":
2000, "burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS":
6000, "burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS":
8000, "burstIOPS": 10000}}]
}
```

2. 백엔드 파일이 준비되면 다음 명령을 실행하여 첫 번째 백엔드를 만듭니다.

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-solidfire.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| solidfire_10.61.180.200 | solidfire-san  | b90783ee-e0c9-49af-8d26-
3ea87ce2efdf | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

3. 백엔드를 만든 후에는 다음으로 스토리지 클래스를 만들어야 합니다. 백엔드와 마찬가지로, sample-inputs 폴더에서 해당 환경에 맞게 편집할 수 있는 샘플 스토리지 클래스 파일이 있습니다. 작업 디렉토리에 복사하고 생성된 백엔드를 반영하도록 필요한 편집을 합니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

4. 이 파일에 대해 해야 할 유일한 편집은 다음을 정의하는 것입니다. `backendType` 새로 생성된 백엔드의 스토리지 드라이버 이름에 대한 값입니다. 또한, 이후 단계에서 참조해야 하는 이름 필드 값도 기록해 둡니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "solidfire-san"
```

라는 선택적 필드가 있습니다. `fsType` 이 파일에 정의되어 있습니다. iSCSI 백엔드에서 이 값은 특정 Linux 파일 시스템 유형(XFS, ext4 등)으로 설정될 수 있으며, OpenShift가 사용할 파일 시스템을 결정하도록 이 값을 삭제할 수도 있습니다.

5. 실행하다 `oc` 저장 클래스를 생성하는 명령입니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

6. 스토리지 클래스가 생성되면 첫 번째 영구 볼륨 클레임(PVC)을 생성해야 합니다. 샘플이 있습니다 `pvc-basic.yaml` 이 작업을 수행하는 데 사용할 수 있는 파일도 sample-inputs에 있습니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

7. 이 파일에 대해 수행해야 하는 유일한 편집은 다음을 보장하는 것입니다. `storageClassName` 필드가 방금 만든 필드와 일치합니다. PVC 정의는 프로비저닝할 작업 부하에 따라 추가로 사용자 정의할 수 있습니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

8. PVC를 생성하려면 다음을 실행하세요. `oc` 명령. 생성되는 백업 볼륨의 크기에 따라 생성에 시간이 걸릴 수 있으므로, 프로세스가 완료되는 모습을 지켜볼 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-3445b5cc-df24-453d-a1e6-b484e874349d   1Gi
RWO            basic-csi      5s
```

고급 구성 옵션

로드 밸런서 옵션 살펴보기

로드 밸런서 옵션 살펴보기**: NetApp** 사용한 **Red Hat OpenShift**

대부분의 경우, Red Hat OpenShift는 경로를 통해 애플리케이션을 외부 세계에 제공합니다. 서비스는 외부에서 접근 가능한 호스트 이름을 제공하여 노출됩니다. 정의된 경로와 해당

서비스에서 식별된 엔드포인트는 OpenShift 라우터에서 사용되어 외부 클라이언트에 지정된 연결을 제공할 수 있습니다.

그러나 어떤 경우에는 애플리케이션에서 적절한 서비스를 노출하기 위해 사용자 정의 로드 밸런서를 배포하고 구성해야 합니다. 이에 대한 한 가지 예가 NetApp Trident Protect입니다. 이러한 요구 사항을 충족하기 위해 우리는 다양한 맞춤형 로드 밸런서 옵션을 평가했습니다. 이 섹션에서는 설치 및 구성에 대해 설명합니다.

다음 페이지에는 NetApp 솔루션이 포함된 Red Hat OpenShift에서 검증된 로드 밸런서 옵션에 대한 추가 정보가 있습니다.

- "메탈엘비"
- "F5 빅-IP"

MetalLB 로드 밸런서 설치**: NetApp** 사용한 **Red Hat OpenShift**

이 페이지에서는 MetalLB 로드 밸런서의 설치 및 구성 지침을 나열합니다.

MetalLB는 OpenShift 클러스터에 설치되는 셀프 호스팅 네트워크 로드 밸런서로, 클라우드 공급자에서 실행되지 않는 클러스터에서 로드 밸런서 유형의 OpenShift 서비스를 생성할 수 있도록 해줍니다. LoadBalancer 서비스를 지원하기 위해 함께 작동하는 MetalLB의 두 가지 주요 기능은 주소 할당과 외부 알림입니다.

MetalLB 구성 옵션

MetalLB가 OpenShift 클러스터 외부의 LoadBalancer 서비스에 할당된 IP 주소를 알리는 방식에 따라 두 가지 모드로 작동합니다.

- 레이어 **2** 모드**.** 이 모드에서는 OpenShift 클러스터의 한 노드가 서비스 소유권을 취득하고 해당 IP에 대한 ARP 요청에 응답하여 OpenShift 클러스터 외부에서도 접근 가능하도록 합니다. 노드만이 IP를 광고하므로 대역폭 병목 현상이 발생하고 장애 조치에 제한이 있습니다. 자세한 내용은 설명서를 참조하세요."여기" .
- **BGP** 모드**.** 이 모드에서는 OpenShift 클러스터의 모든 노드가 라우터와 BGP 피어링 세션을 설정하고 서비스 IP로 트래픽을 전달하는 경로를 광고합니다. 이를 위해서는 MetalLB를 해당 네트워크의 라우터와 통합해야 합니다. BGP의 해싱 메커니즘으로 인해 서비스의 IP-노드 매핑이 변경될 때 특정 제한이 발생합니다. 자세한 내용은 설명서를 참조하세요."여기" .

ⓘ 이 문서에서는 MetalLB를 2계층 모드로 구성합니다.

MetalLB 로드 밸런서 설치

1. MetalLB 리소스를 다운로드하세요.

```
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/name
space.yaml
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/meta
llb.yaml
```

2. 파일 편집 `metallb.yaml` 그리고 제거하다 `spec.template.spec.securityContext` 컨트롤러 배포와

스피커 DaemonSet에서.

삭제할 줄:

```
securityContext:
  runAsNonRoot: true
  runAsUser: 65534
```

3. 생성하다 `metallb-system` 네임스페이스.

```
[netapp-user@rhel7 ~]$ oc create -f namespace.yaml
namespace/metallb-system created
```

4. MetalLB CR을 만듭니다.

```
[netapp-user@rhel7 ~]$ oc create -f metallb.yaml
podsecuritypolicy.policy/controller created
podsecuritypolicy.policy/speaker created
serviceaccount/controller created
serviceaccount/speaker created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
role.rbac.authorization.k8s.io/config-watcher created
role.rbac.authorization.k8s.io/pod-lister created
role.rbac.authorization.k8s.io/controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller
created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker
created
rolebinding.rbac.authorization.k8s.io/config-watcher created
rolebinding.rbac.authorization.k8s.io/pod-lister created
rolebinding.rbac.authorization.k8s.io/controller created
daemonset.apps/speaker created
deployment.apps/controller created
```

5. MetalLB 스피커를 구성하기 전에 스피커 DaemonSet에 높은 권한을 부여하여 로드 밸런서가 작동하는 데 필요한 네트워킹 구성을 수행할 수 있도록 합니다.

```
[netapp-user@rhel7 ~]$ oc adm policy add-scc-to-user privileged -n
metallb-system -z speaker
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:privileged
added: "speaker"
```

6. MetalLB를 구성하려면 다음을 생성하세요. `ConfigMap` 에서 `metallb-system` 네임스페이스.

```
[netapp-user@rhel7 ~]$ vim metallb-config.yaml

apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 10.63.17.10-10.63.17.200

[netapp-user@rhel7 ~]$ oc create -f metallb-config.yaml
configmap/config created
```

7. 이제 로드 밸런서 서비스가 생성되면 MetalLB는 서비스에 외부 IP를 할당하고 ARP 요청에 응답하여 IP 주소를 알립니다.

BGP 모드에서 MetalLB를 구성하려면 위의 6단계를 건너뛰고 MetalLB 설명서의 절차를 따르세요 ."여기" .

F5 BIG-IP 로드 밸런서 설치

F5 BIG-IP는 L4-L7 부하 분산, SSL/TLS 오프로드, DNS, 방화벽 등을 비롯한 광범위한 고급 프로덕션 등급 트래픽 관리 및 보안 서비스를 제공하는 애플리케이션 전송 컨트롤러(ADC) 입니다. 이러한 서비스는 애플리케이션의 가용성, 보안 및 성능을 크게 향상시킵니다.

F5 BIG-IP는 전용 하드웨어, 클라우드 또는 온프레미스 가상 어플라이언스로 다양한 방식으로 배포 및 사용할 수 있습니다. 요구 사항에 따라 F5 BIG-IP를 탐색하고 배포하려면 여기의 설명서를 참조하세요.

F5 BIG-IP 서비스와 Red Hat OpenShift를 효율적으로 통합하기 위해 F5는 BIG-IP 컨테이너 인그레스 서비스(CIS)를 제공합니다. CIS는 특정 사용자 정의 리소스 정의(CRD)에 대한 OpenShift API를 감시하고 F5 BIG-IP 시스템 구성을 관리하는 컨트롤러 포드로 설치됩니다. F5 BIG-IP CIS는 OpenShift에서 LoadBalancer 및 Routes 서비스 유형을 제어하도록 구성할 수 있습니다.

또한, LoadBalancer 유형에 대한 서비스를 위해 자동으로 IP 주소를 할당하려면 F5 IPAM 컨트롤러를 활용할 수 있습니다. F5 IPAM 컨트롤러는 ipamLabel 주석을 사용하여 사전 구성된 풀에서 IP 주소를 할당하는 LoadBalancer 서비스의 OpenShift API를 감시하는 컨트롤러 포드로 설치됩니다.

이 페이지에서는 F5 BIG-IP CIS 및 IPAM 컨트롤러에 대한 설치 및 구성 지침을 나열합니다. 필수 조건으로 F5 BIG-IP 시스템을 배포하고 라이선스를 받아야 합니다. BIG-IP VE 기반 라이선스에 기본적으로 포함되는 SDN 서비스에 대한 라이선스도 필요합니다.

F5 BIG-IP는 독립형 또는 클러스터 모드로 구축할 수 있습니다. 이러한 검증을 위해 F5 BIG-IP는 독립 실행형 모드로 배포되었지만, 운영 목적으로는 단일 장애 지점을 방지하기 위해 BIG-IP 클러스터를 사용하는 것이 더 좋습니다.

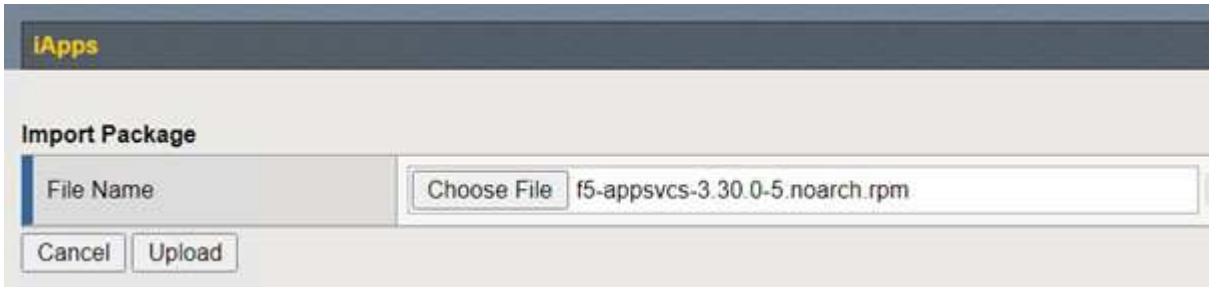
F5 BIG-IP 시스템은 전용 하드웨어, 클라우드 또는 온프레미스 가상 어플라이언스로 배포하여 F5 CIS와 통합할 수 있도록 12.x 이상 버전을 사용할 수 있습니다. 이 문서의 목적상 F5 BIG-IP 시스템은 BIG-IP VE 에디션 등을 사용하여 가상 어플라이언스로 검증되었습니다.

검증된 릴리스

기술	소프트웨어 버전
레드햇 오픈시프트	4.6 유로, 4.7
F5 BIG-IP VE 에디션	16.1.0
F5 컨테이너 유입 서비스	2.5.1
F5 IPAM 컨트롤러	0.1.4
F5 AS3	3.30.0

설치

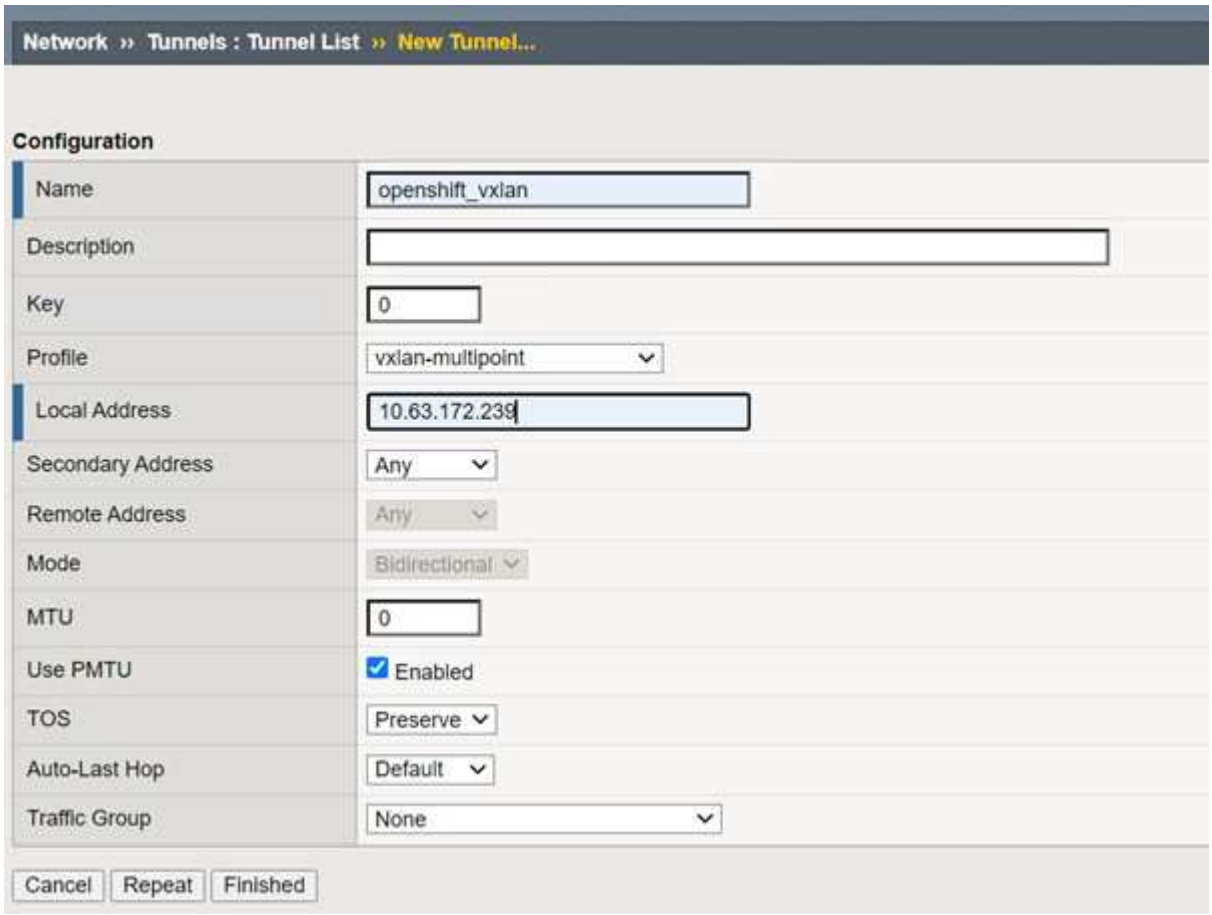
1. F5 Application Services 3 확장을 설치하면 BIG-IP 시스템이 명령형 명령 대신 JSON으로 구성을 허용할 수 있습니다. 로 가다 "F5 AS3 GitHub 저장소" , 최신 RPM 파일을 다운로드하세요.
2. F5 BIG-IP 시스템에 로그인하고 iApps > 패키지 관리 LX로 이동한 후 가져오기를 클릭합니다.
3. 파일 선택을 클릭하고 다운로드한 AS3 RPM 파일을 선택한 후 확인을 클릭하고 업로드를 클릭합니다.



4. AS3 확장 프로그램이 성공적으로 설치되었는지 확인하세요.

5. 다음으로 OpenShift와 BIG-IP 시스템 간 통신에 필요한 리소스를 구성합니다. 먼저 OpenShift SDN을 위한 BIG-IP 시스템에 VXLAN 터널 인터페이스를 생성하여 OpenShift와 BIG-IP 서버 간에 터널을 생성합니다. 네트워크 > 터널 > 프로필로 이동한 후 만들기를 클릭하고 부모 프로필을 vxlan으로, 플러딩 유형을 멀티캐스트로 설정합니다. 프로필 이름을 입력하고 '완료'를 클릭합니다.

6. 네트워크 > 터널 > 터널 목록으로 이동하여 만들기를 클릭하고 터널의 이름과 로컬 IP 주소를 입력합니다. 이전 단계에서 만든 터널 프로필을 선택하고 '마침'을 클릭합니다.



7. 클러스터 관리자 권한으로 Red Hat OpenShift 클러스터에 로그인합니다.
8. OpenShift에서 F5 BIG-IP 서버용 호스트 서브넷을 생성하여 OpenShift 클러스터의 서브넷을 F5 BIG-IP 서버로 확장합니다. 호스트 서브넷 YAML 정의를 다운로드합니다.

```
wget https://github.com/F5Networks/k8s-bigip-
ctlr/blob/master/docs/config_examples/openshift/f5-kctlr-openshift-
hostsubnet.yaml
```

9. 호스트 서브넷 파일을 편집하고 OpenShift SDN에 대한 BIG-IP VTEP(VXLAN 터널) IP를 추가합니다.

```
apiVersion: v1
kind: HostSubnet
metadata:
  name: f5-server
  annotations:
    pod.network.openshift.io/fixed-vnid-host: "0"
    pod.network.openshift.io/assign-subnet: "true"
# provide a name for the node that will serve as BIG-IP's entry into the
cluster
host: f5-server
# The hostIP address will be the BIG-IP interface address routable to
the
# OpenShift Origin nodes.
# This address is the BIG-IP VTEP in the SDN's VXLAN.
hostIP: 10.63.172.239
```

사용자 환경에 맞게 호스트 IP 및 기타 세부 정보를 변경하세요.

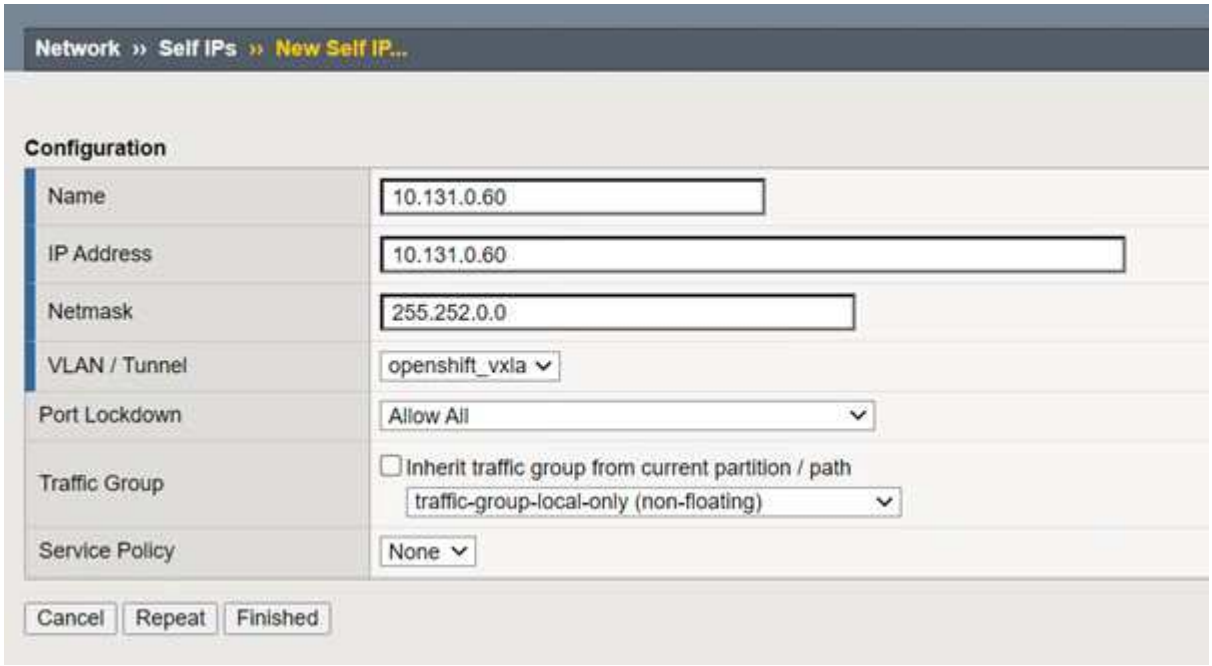
10. HostSubnet 리소스를 생성합니다.

```
[admin@rhel-7 ~]$ oc create -f f5-kctlr-openshift-hostsubnet.yaml

hostsubnet.network.openshift.io/f5-server created
```

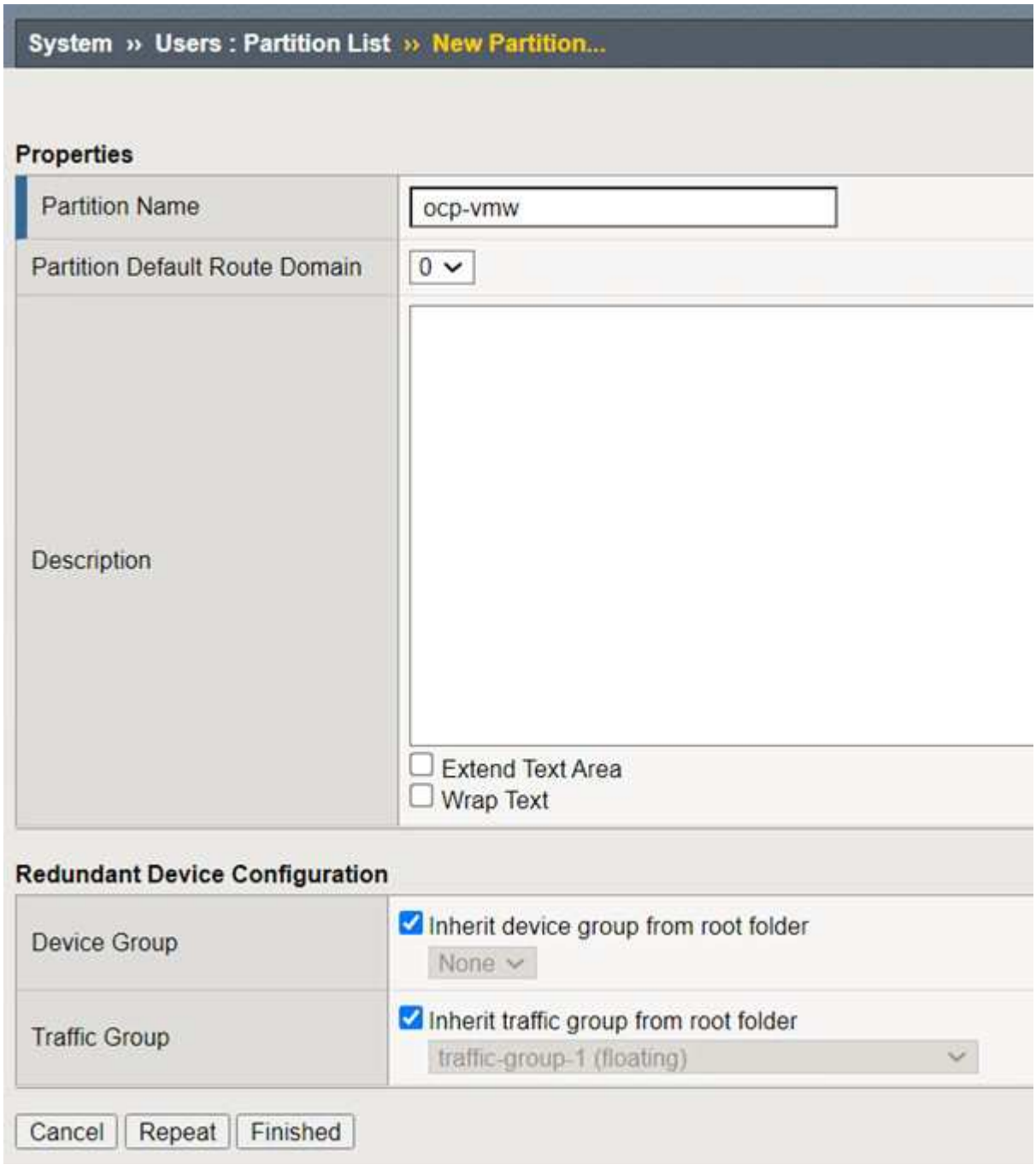
11. F5 BIG-IP 서버에 대해 생성된 호스트 서브넷에 대한 클러스터 IP 서브넷 범위를 가져옵니다.

```
[admin@rhel-7 ~]$ oc get hostsubnet

NAME                           HOST                           HOST IP
SUBNET          EGRESS CIDRS   EGRESS IPS
f5-server                      f5-server                      10.63.172.239
10.131.0.0/23
ocp-vmw-nszws-master-0         ocp-vmw-nszws-master-0         10.63.172.44
10.128.0.0/23
ocp-vmw-nszws-master-1         ocp-vmw-nszws-master-1         10.63.172.47
10.130.0.0/23
ocp-vmw-nszws-master-2         ocp-vmw-nszws-master-2         10.63.172.48
10.129.0.0/23
ocp-vmw-nszws-worker-r8fh4     ocp-vmw-nszws-worker-r8fh4     10.63.172.7
10.130.2.0/23
ocp-vmw-nszws-worker-tvr46     ocp-vmw-nszws-worker-tvr46     10.63.172.11
10.129.2.0/23
ocp-vmw-nszws-worker-wdxhg     ocp-vmw-nszws-worker-wdxhg     10.63.172.24
10.128.2.0/23
ocp-vmw-nszws-worker-wg8r4     ocp-vmw-nszws-worker-wg8r4     10.63.172.15
10.131.2.0/23
ocp-vmw-nszws-worker-wtgfw     ocp-vmw-nszws-worker-wtgfw     10.63.172.17
10.128.4.0/23
```

12. F5 BIG-IP 서버에 해당하는 OpenShift의 호스트 서브넷 범위에 있는 IP로 OpenShift VXLAN에 자체 IP를 생성합니다. F5 BIG-IP 시스템에 로그인하고 네트워크 > 자체 IP로 이동한 다음 생성을 클릭합니다. F5 BIG-IP 호스트 서브넷에 대해 생성된 클러스터 IP 서브넷의 IP를 입력하고, VXLAN 터널을 선택한 후, 다른 세부 정보를 입력합니다. 그런 다음 완료를 클릭합니다.



13. CIS와 함께 구성하고 사용할 F5 BIG-IP 시스템에 파티션을 만듭니다. 시스템 > 사용자 > 파티션 목록으로 이동하여 만들기를 클릭하고 세부 정보를 입력합니다. 그런 다음 완료를 클릭합니다.



F5는 CIS가 관리하는 파티션에 수동 구성을 수행하지 않을 것을 권장합니다.

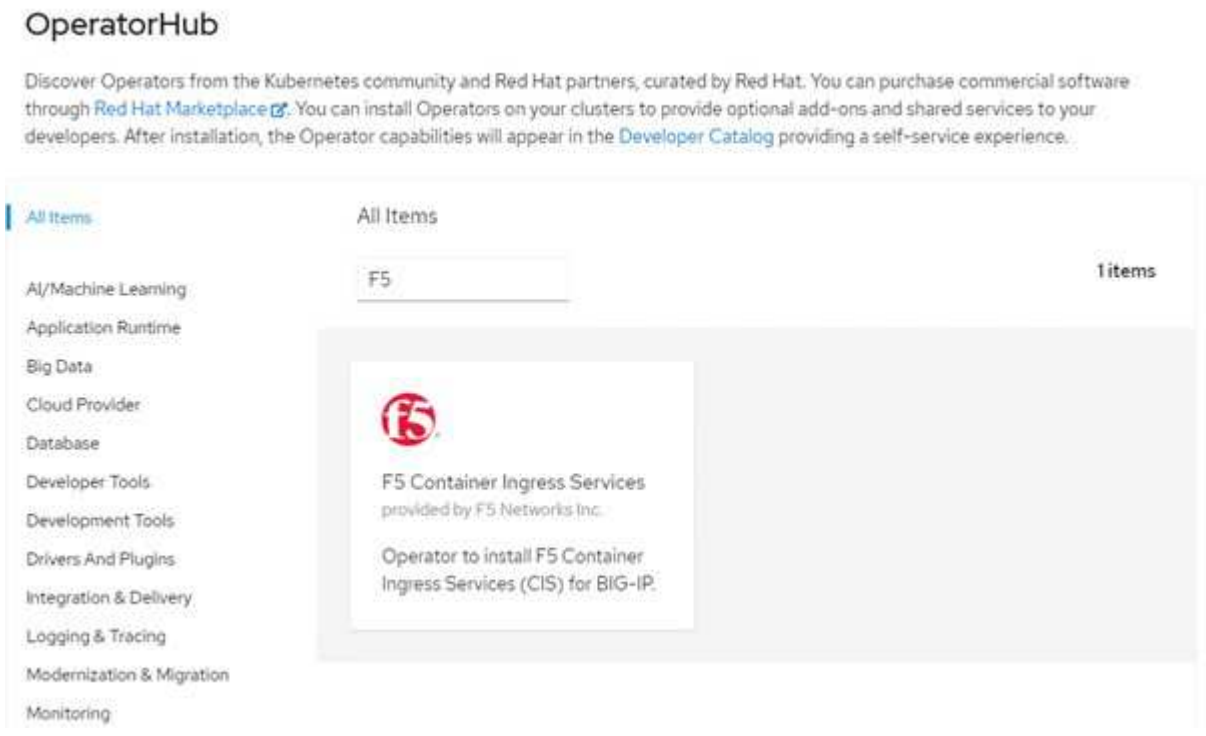
14. OperatorHub의 운영자를 사용하여 F5 BIG-IP CIS를 설치합니다. 클러스터 관리자 권한으로 Red Hat OpenShift 클러스터에 로그인하고 운영자의 전제 조건인 F5 BIG-IP 시스템 로그인 자격 증명으로 비밀을 생성합니다.

```
[admin@rhel-7 ~]$ oc create secret generic bigip-login -n kube-system
--from-literal=username=admin --from-literal=password=admin

secret/bigip-login created
```

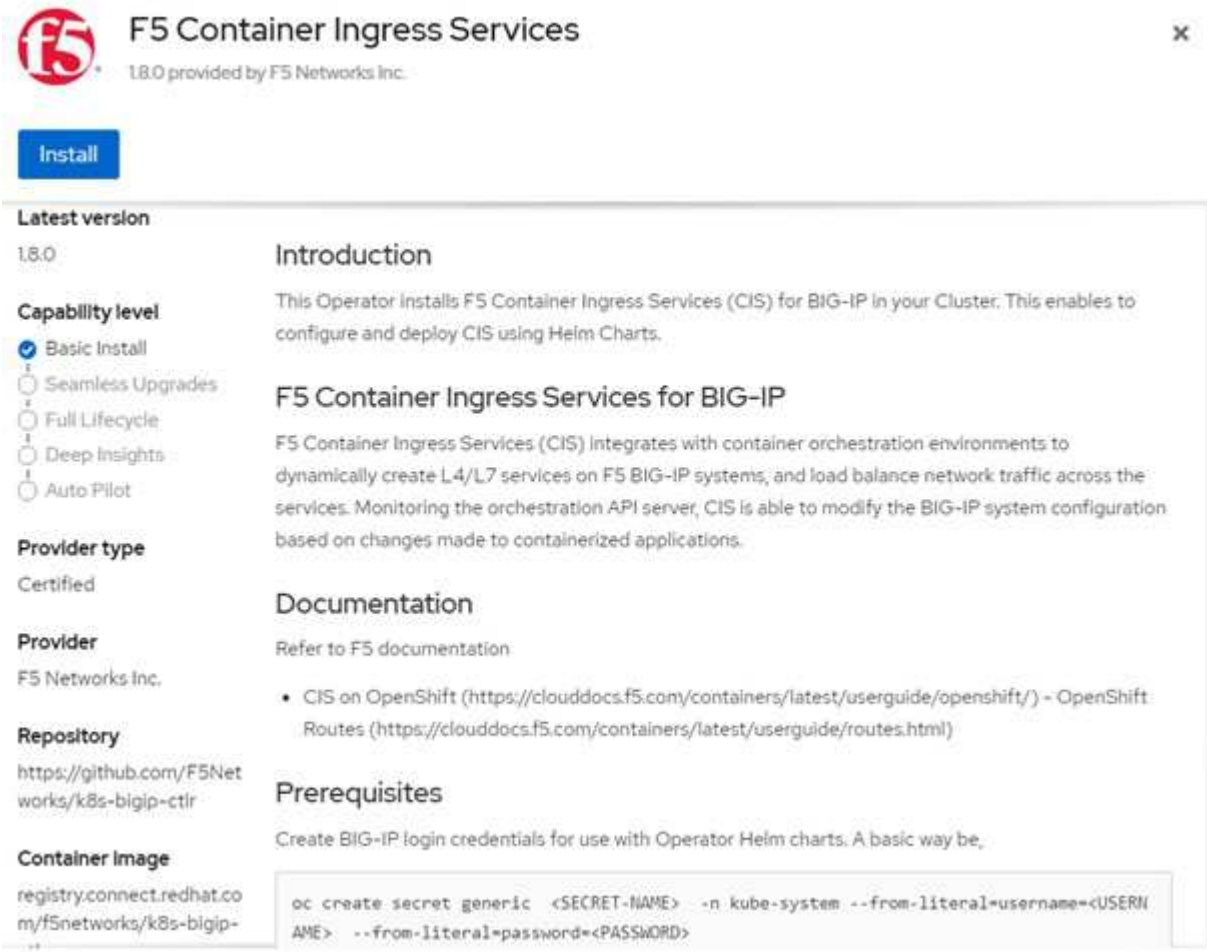
15. F5 CIS CRD를 설치합니다.

```
[admin@rhel-7 ~]$ oc apply -f
https://raw.githubusercontent.com/F5Networks/k8s-bigip-
ctlr/master/docs/config_examples/crd/Install/customresourcedefinitions.y
ml

customresourcedefinition.apiextensions.k8s.io/virtualservers.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/tlsprofiles.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/transportservers.cis.f5.co
m created
customresourcedefinition.apiextensions.k8s.io/externaldnss.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/ingresslinks.cis.f5.com
created
```

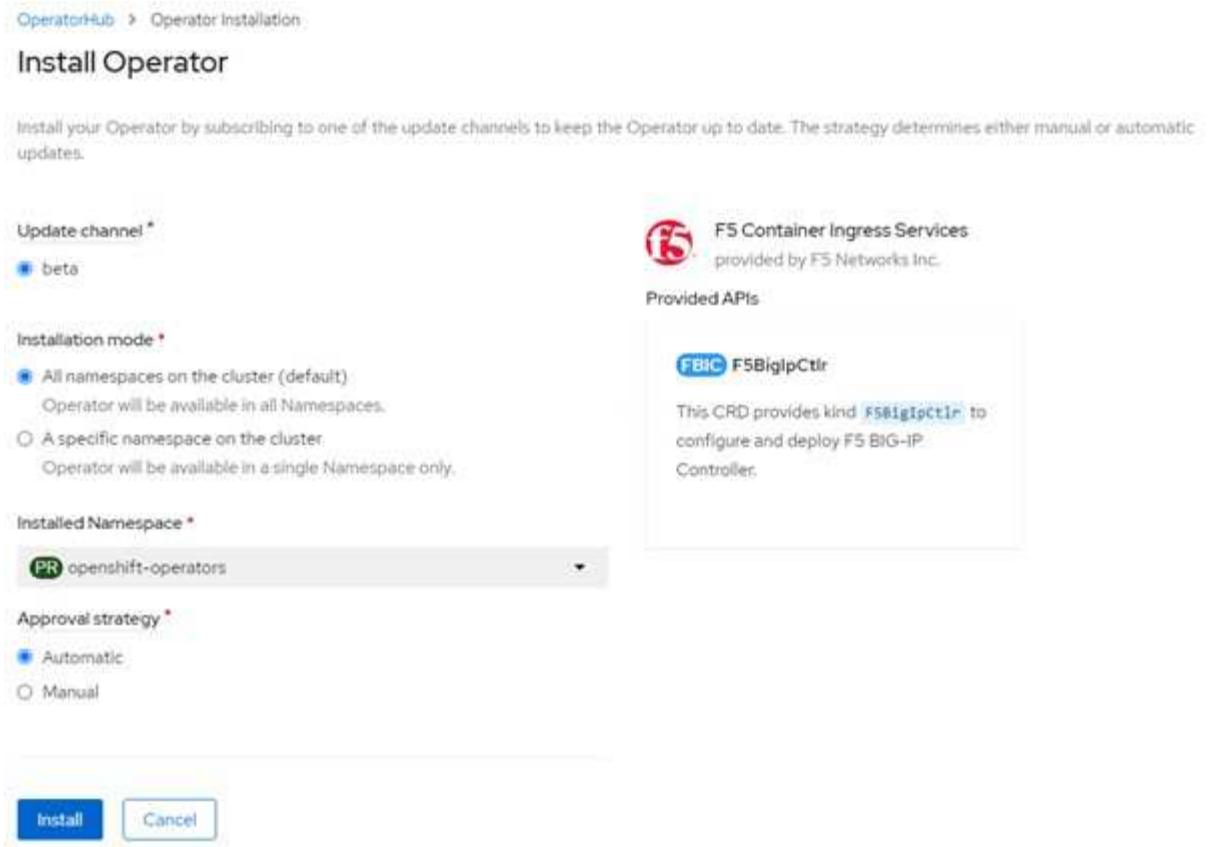
16. Operators > OperatorHub로 이동하여 키워드 F5를 검색하고 F5 Container Ingress Service 타일을 클릭합니다.



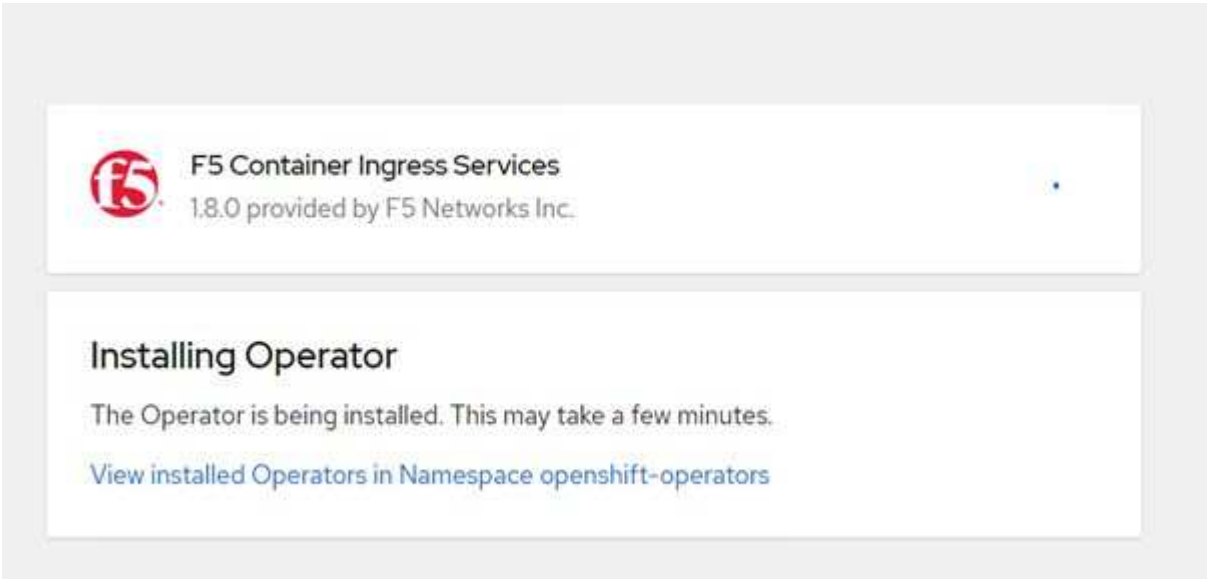
17. 운영자 정보를 읽고 설치를 클릭하세요.



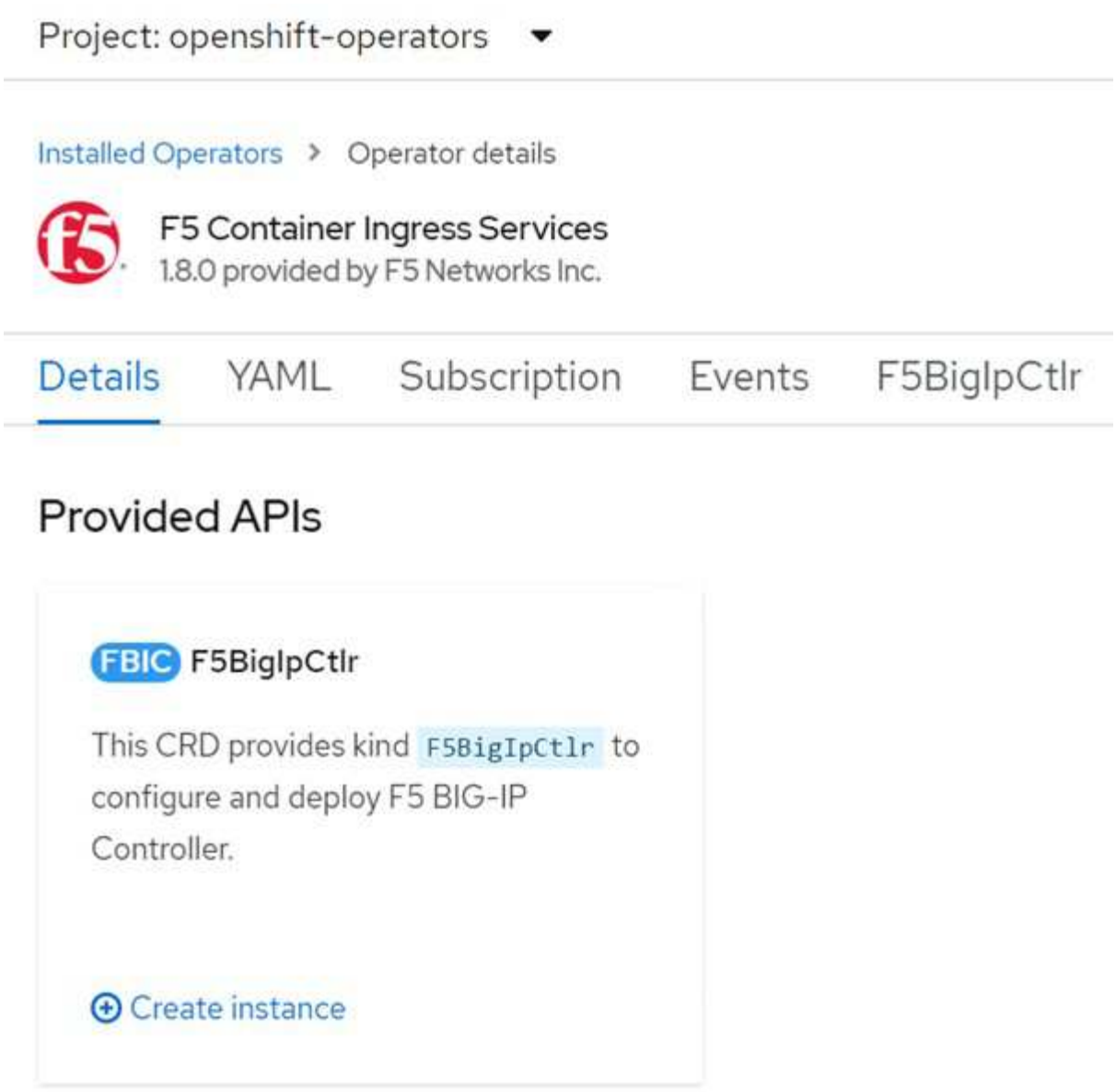
18. 설치 운영자 화면에서 모든 기본 매개변수를 그대로 두고 설치를 클릭합니다.



19. 운영자를 설치하는 데 시간이 걸립니다.



20. 운영자가 설치되면 설치 성공 메시지가 표시됩니다.
21. 운영자 > 설치된 운영자로 이동한 후 F5 컨테이너 수신 서비스를 클릭하고 F5BigIpCtlr 타일 아래에서 인스턴스 만들기를 클릭합니다.



22. YAML 보기를 클릭하고 필요한 매개변수를 업데이트한 후 다음 내용을 붙여넣습니다.

매개변수 업데이트 `bigip_partition` , ` openshift_sdn_name`, `bigip_url` 그리고 `bigip_login_secret` 아래는 콘텐츠를 복사하기 전에 설정에 필요한 값을 반영하기 위한 것입니다.

```
apiVersion: cis.f5.com/v1
kind: F5BigIpCtlr
metadata:
  name: f5-server
  namespace: openshift-operators
spec:
  args:
    log_as3_response: true
    agent: as3
    log_level: DEBUG
    bigip_partition: ocp-vmw
    openshift_sdn_name: /Common/openshift_vxlan
    bigip_url: 10.61.181.19
    insecure: true
    pool-member-type: cluster
    custom_resource_mode: true
    as3_validation: true
    ipam: true
    manage_configmaps: true
  bigip_login_secret: bigip-login
  image:
    pullPolicy: Always
    repo: f5networks/cntr-ingress-svcs
    user: registry.connect.redhat.com
  namespace: kube-system
  rbac:
    create: true
  resources: {}
  serviceAccount:
    create: true
  version: latest
```

23. 이 콘텐츠를 붙여넣은 후 만들기를 클릭합니다. 이렇게 하면 kube-system 네임스페이스에 CIS 포드가 설치됩니다.

Red Hat OpenShift는 기본적으로 L7 부하 분산을 위한 경로를 통해 서비스를 노출하는 방법을 제공합니다. 내장된 OpenShift 라우터는 이러한 경로에 대한 광고와 트래픽 처리를 담당합니다. 하지만 외부 F5 BIG-IP 시스템을 통해 경로를 지원하도록 F5 CIS를 구성할 수도 있습니다. 이 시스템은 보조 라우터로 실행되거나 자체 호스팅 OpenShift 라우터를 대체하는 장치로 실행될 수 있습니다. CIS는 OpenShift 경로에 대한 라우터 역할을 하는 가상 서버를 BIG-IP 시스템에 생성하고, BIG-IP는 광고 및 트래픽 라우팅을 처리합니다. 이 기능을 활성화하기 위한 매개변수에 대한 자세한 내용은 여기의 설명서를 참조하세요. 이러한 매개변수는 apps/v1 API의 OpenShift 배포 리소스에 대해 정의되어 있습니다. 따라서 이를 F5BigIpCtlr 리소스 cis.f5.com/v1 API와 함께 사용할 때 매개변수 이름의 하이픈(-)을 밑줄(_)로 바꾸세요.

24. CIS 리소스 생성에 전달되는 인수에는 다음이 포함됩니다. `ipam: true` 그리고 `custom_resource_mode: true` . 이러한 매개변수는 CIS와 IPAM 컨트롤러의 통합을 활성화하는 데 필요합니다. F5 IPAM 리소스를 생성하여 CIS가 IPAM 통합을 활성화했는지 확인합니다.

```
[admin@rhel-7 ~]$ oc get f5ipam -n kube-system

NAMESPACE    NAME                                AGE
kube-system    ipam.10.61.181.19.ocp-vmw       43s
```

25. F5 IPAM 컨트롤러에 필요한 서비스 계정, 역할 및 역할 바인딩을 생성합니다. YAML 파일을 만들고 다음 내용을 붙여넣습니다.

```
[admin@rhel-7 ~]$ vi f5-ipam-rbac.yaml

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole
rules:
  - apiGroups: ["fic.f5.com"]
    resources: ["ipams","ipams/status"]
    verbs: ["get", "list", "watch", "update", "patch"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole-binding
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ipam-ctlr-clusterrole
subjects:
  - apiGroup: ""
    kind: ServiceAccount
    name: ipam-ctlr
    namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ipam-ctlr
  namespace: kube-system
```

26. 리소스를 생성합니다.

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-rbac.yaml

clusterrole.rbac.authorization.k8s.io/ipam-ctlr-clusterrole created
clusterrolebinding.rbac.authorization.k8s.io/ipam-ctlr-clusterrole-
binding created
serviceaccount/ipam-ctlr created
```

27. YAML 파일을 만들고 아래에 제공된 F5 IPAM 배포 정의를 붙여넣습니다.

아래 spec.template.spec.containers[0].args의 ip-range 매개변수를 업데이트하여 설정에 해당하는 ipamLabels 및 IP 주소 범위를 반영합니다.

ipamLabels[`range1` 그리고 `range2` [아래 예] IPAM 컨트롤러가 정의된 범위에서 IP 주소를 감지하고 할당할 수 있도록 LoadBalancer 유형의 서비스에 대해 주석이 필요합니다.

```
[admin@rhel-7 ~]$ vi f5-ipam-deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    name: f5-ipam-controller
  name: f5-ipam-controller
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: f5-ipam-controller
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: f5-ipam-controller
    spec:
      containers:
      - args:
        - --orchestration=openshift
        - --ip-range='{"range1":"10.63.172.242-10.63.172.249",
"range2":"10.63.170.111-10.63.170.129"}'
        - --log-level=DEBUG
        command:
        - /app/bin/f5-ipam-controller
        image: registry.connect.redhat.com/f5networks/f5-ipam-
controller:latest
        imagePullPolicy: IfNotPresent
        name: f5-ipam-controller
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: ipam-ctlr
      serviceAccountName: ipam-ctlr
```

28. F5 IPAM 컨트롤러 배포를 생성합니다.

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-deployment.yaml

deployment/f5-ipam-controller created
```

29. F5 IPAM 컨트롤러 포드가 실행 중인지 확인하세요.

```
[admin@rhel-7 ~]$ oc get pods -n kube-system

NAME                                      READY   STATUS    RESTARTS
AGE
f5-ipam-controller-5986cff5bd-2bvn6       1/1     Running   0
30s
f5-server-f5-bigip-ctlr-5d7578667d-qxdgj  1/1     Running   0
14m
```

30. F5 IPAM 스키마를 생성합니다.

```
[admin@rhel-7 ~]$ oc create -f
https://raw.githubusercontent.com/F5Networks/f5-ipam-
controller/main/docs/_static/schemas/ipam_schema.yaml

customresourcedefinition.apiextensions.k8s.io/ipams.fic.f5.com
```

확인

1. LoadBalancer 유형의 서비스를 만듭니다.

```
[admin@rhel-7 ~]$ vi example_svc.yaml

apiVersion: v1
kind: Service
metadata:
  annotations:
    cis.f5.com/ipamLabel: range1
  labels:
    app: f5-demo-test
  name: f5-demo-test
  namespace: default
spec:
  ports:
  - name: f5-demo-test
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: f5-demo-test
  sessionAffinity: None
  type: LoadBalancer
```

```
[admin@rhel-7 ~]$ oc create -f example_svc.yaml

service/f5-demo-test created
```

2. IPAM 컨트롤러가 외부 IP를 할당하는지 확인하세요.

```
[admin@rhel-7 ~]$ oc get svc

NAME           TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
f5-demo-test   LoadBalancer   172.30.210.108   10.63.172.242
80:32605/TCP   27s
```

3. 배포를 생성하고 생성된 LoadBalancer 서비스를 사용합니다.

```
[admin@rhel-7 ~]$ vi example_deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: f5-demo-test
  name: f5-demo-test
spec:
  replicas: 2
  selector:
    matchLabels:
      app: f5-demo-test
  template:
    metadata:
      labels:
        app: f5-demo-test
    spec:
      containers:
      - env:
        - name: service_name
          value: f5-demo-test
        image: nginx
        imagePullPolicy: Always
        name: f5-demo-test
        ports:
        - containerPort: 80
          protocol: TCP
```

```
[admin@rhel-7 ~]$ oc create -f example_deployment.yaml

deployment/f5-demo-test created
```

4. 포드가 실행 중인지 확인하세요.

```
[admin@rhel-7 ~]$ oc get pods

NAME                           READY   STATUS    RESTARTS   AGE
f5-demo-test-57c46f6f98-47wwp   1/1     Running   0          27s
f5-demo-test-57c46f6f98-cl2m8   1/1     Running   0          27s
```

5. OpenShift의 LoadBalancer 유형 서비스에 대해 BIG-IP 시스템에 해당 가상 서버가 생성되었는지 확인하세요. 로컬 트래픽 > 가상 서버 > 가상 서버 목록으로 이동합니다.

개인 이미지 레지스트리 생성

Red Hat OpenShift의 대부분 배포의 경우 다음과 같은 공개 레지스트리를 사용합니다. "Quay.io" 또는 "도커허브" 대부분의 고객 요구 사항을 충족합니다. 하지만 고객이 자신만의 비공개 이미지나 맞춤형 이미지를 호스팅하고 싶어하는 경우도 있습니다.

이 절차에서는 Trident 와 NetApp ONTAP 에서 제공하는 영구 볼륨으로 지원되는 개인 이미지 레지스트리를 만드는 방법을 설명합니다.

> Trident Protect에는 Astra 컨테이너에 필요한 이미지를 호스팅하는 레지스트리가 필요합니다. 다음 섹션에서는 Red Hat OpenShift 클러스터에 개인 레지스트리를 설정하고 Trident Protect 설치를 지원하는 데 필요한 이미지를 푸시하는 단계를 설명합니다.

개인 이미지 레지스트리 만들기

1. 현재 기본 저장소 클래스에서 기본 주석을 제거하고 Trident 지원 저장소 클래스를 OpenShift 클러스터의 기본값으로 주석 처리합니다.

```
[netapp-user@rhel7 ~]$ oc patch storageclass thin -p '{"metadata":
{"annotations": {"storageclass.kubernetes.io/is-default-class":
"false"}}}'
storageclass.storage.k8s.io/thin patched

[netapp-user@rhel7 ~]$ oc patch storageclass ocp-trident -p
'{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-
class": "true"}}}'
storageclass.storage.k8s.io/ocp-trident patched
```

2. 다음 저장 매개변수를 입력하여 imageregistry 연산자를 편집합니다. `spec` 부분.

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

storage:
  pvc:
    claim:
```

3. 다음 매개변수를 입력하세요. `spec` 사용자 지정 호스트 이름으로 OpenShift 경로를 만드는 섹션입니다. 저장하고

종료합니다.

```
routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
```

위의 경로 구성은 경로에 사용자 정의 호스트 이름이 필요할 때 사용됩니다. OpenShift가 기본 호스트 이름으로 경로를 생성하도록 하려면 다음 매개변수를 추가할 수 있습니다. `spec` 부분: `defaultRoute: true` .

사용자 정의 **TLS** 인증서

경로에 사용자 지정 호스트 이름을 사용하는 경우 기본적으로 OpenShift Ingress 운영자의 기본 TLS 구성이 사용됩니다. 하지만 경로에 사용자 정의 TLS 구성을 추가할 수 있습니다. 그렇게 하려면 다음 단계를 완료하세요.

a. 경로의 TLS 인증서와 키를 사용하여 비밀을 생성합니다.

```
[netapp-user@rhel7 ~]$ oc create secret tls astra-route-tls -n
openshift-image-registry –cert/home/admin/netapp-astra/tls.crt
--key=/home/admin/netapp-astra/tls.key
```

b. imageregistry 연산자를 편집하고 다음 매개변수를 추가합니다. `spec` 부분.

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
    secretName: astra-route-tls
```

4. imageregistry 운영자를 다시 편집하고 운영자의 관리 상태를 다음으로 변경합니다. `Managed` 상태. 저장하고 종료합니다.

```
oc edit configs.imageregistry/cluster

managementState: Managed
```

5. 모든 전제 조건이 충족되면 개인 이미지 레지스트리에 대한 PVC, 포드 및 서비스가 생성됩니다. 몇 분 안에 레지스트리가 작동할 것입니다.

```
[netapp-user@rhel7 ~]$oc get all -n openshift-image-registry

NAME                                                  READY   STATUS
RESTARTS   AGE
pod/cluster-image-registry-operator-74f6d954b6-rb7zr   1/1     Running
3          90d
pod/image-pruner-1627257600-f5cpj                     0/1     Completed
0          2d9h
pod/image-pruner-1627344000-swqx9                     0/1     Completed
0          33h
pod/image-pruner-1627430400-rv5nt                     0/1     Completed
0          9h
pod/image-registry-6758b547f-6pnj8                    1/1     Running
0          76m
pod/node-ca-bwb5r                                     1/1     Running
0          90d
pod/node-ca-f8w54                                     1/1     Running
0          90d
pod/node-ca-gjx7h                                     1/1     Running
0          90d
pod/node-ca-lcx4k                                     1/1     Running
0          33d
pod/node-ca-v7zmx                                     1/1     Running
0          7d21h
pod/node-ca-xpppp                                     1/1     Running
0          89d

NAME                              TYPE        CLUSTER-IP       EXTERNAL-
IP   PORT(S)     AGE
service/image-registry            ClusterIP   172.30.196.167   <none>
5000/TCP    15h
service/image-registry-operator   ClusterIP   None             <none>
60000/TCP   90d

NAME                      DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/node-ca   6         6         6       6            6
kubernetes.io/os=linux   90d

NAME                                          READY   UP-TO-DATE
AVAILABLE   AGE
deployment.apps/cluster-image-registry-operator   1/1     1            1
90d
deployment.apps/image-registry                    1/1     1            1
15h
```

```
NAME                                                                DESIRED
CURRENT   READY   AGE
replicaset.apps/cluster-image-registry-operator-74f6d954b6   1         1
1       90d
replicaset.apps/image-registry-6758b547f                    1         1
1       76m
replicaset.apps/image-registry-78bfbd7f59                   0         0
0       15h
replicaset.apps/image-registry-7fcc8d6cc8                   0         0
0       80m
replicaset.apps/image-registry-864f88f5b                    0         0
0       15h
replicaset.apps/image-registry-cb47fffb                     0         0
0       10h

NAME                                 COMPLETIONS   DURATION   AGE
job.batch/image-pruner-1627257600   1/1           10s        2d9h
job.batch/image-pruner-1627344000   1/1           6s         33h
job.batch/image-pruner-1627430400   1/1           5s         9h

NAME                          SCHEDULE    SUSPEND   ACTIVE   LAST
SCHEDULE   AGE
cronjob.batch/image-pruner   0 0 * * *   False     0        9h
90d

NAME                                       HOST/PORT
PATH   SERVICES         PORT    TERMINATION   WILDCARD
route.route.openshift.io/public-routes   astra-registry.apps.ocp-
vmw.cie.netapp.com          image-registry   <all>   reencrypt     None
```

6. Ingress 운영자 OpenShift 레지스트리 경로에 기본 TLS 인증서를 사용하는 경우 다음 명령을 사용하여 TLS 인증서를 가져올 수 있습니다.

```
[netapp-user@rhel7 ~]$ oc extract secret/router-ca --keys=tls.crt -n
openshift-ingress-operator
```

7. OpenShift 노드가 레지스트리에 액세스하여 이미지를 가져올 수 있도록 하려면 OpenShift 노드의 Docker 클라이언트에 인증서를 추가합니다. configmap을 생성합니다. `openshift-config` TLS 인증서를 사용하여 네임스페이스를 만들고 클러스터 이미지 구성에 패치하여 인증서를 신뢰할 수 있도록 만듭니다.

```
[netapp-user@rhel7 ~]$ oc create configmap astra-ca -n openshift-config
--from-file=astra-registry.apps.ocp-vmw.cie.netapp.com=tls.crt

[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster
--patch '{"spec":{"additionalTrustedCA":{"name":"astra-ca"}}}'
--type=merge
```

8. OpenShift 내부 레지스트리는 인증을 통해 제어됩니다. 모든 OpenShift 사용자는 OpenShift 레지스트리에 액세스할 수 있지만, 로그인한 사용자가 수행할 수 있는 작업은 사용자 권한에 따라 달라집니다.
   a. 사용자 또는 사용자 그룹이 레지스트리에서 이미지를 가져올 수 있도록 하려면 해당 사용자에게 레지스트리 뷰어 역할이 할당되어야 합니다.

   ```
   [netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-viewer
   ocp-user

   [netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-viewer
   ocp-user-group
   ```

   b. 사용자 또는 사용자 그룹이 이미지를 작성하거나 푸시할 수 있도록 하려면 해당 사용자에게 registry-editor 역할이 할당되어야 합니다.

   ```
   [netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-editor
   ocp-user

   [netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-editor
   ocp-user-group
   ```

9. OpenShift 노드가 레지스트리에 액세스하여 이미지를 푸시하거나 풀하려면 풀 시크릿을 구성해야 합니다.

   ```
   [netapp-user@rhel7 ~]$ oc create secret docker-registry astra-registry-
   credentials --docker-server=astra-registry.apps.ocp-vmw.cie.netapp.com
   --docker-username=ocp-user --docker-password=password
   ```

10. 이 풀 시크릿은 서비스 계정에 패치되거나 해당 포드 정의에서 참조될 수 있습니다.
    a. 서비스 계정에 패치를 적용하려면 다음 명령을 실행하세요.

    ```
    [netapp-user@rhel7 ~]$ oc secrets link <service_account_name> astra-
    registry-credentials --for=pull
    ```

    b. 포드 정의에서 풀 시크릿을 참조하려면 다음 매개변수를 추가하세요. `spec` 부분.

```
imagePullSecrets:
  - name: astra-registry-credentials
```

11. OpenShift 노드와 다른 워크스테이션에서 이미지를 푸시하거나 풀하려면 다음 단계를 완료하세요.
    a. Docker 클라이언트에 TLS 인증서를 추가합니다.

```
[netapp-user@rhel7 ~]$ sudo mkdir /etc/docker/certs.d/astra-
registry.apps.ocp-vmw.cie.netapp.com

[netapp-user@rhel7 ~]$ sudo cp /path/to/tls.crt
/etc/docker/certs.d/astra-registry.apps.ocp-vmw.cie.netapp.com
```

    b. oc login 명령을 사용하여 OpenShift에 로그인합니다.

```
[netapp-user@rhel7 ~]$ oc login --token=sha256~D49SpB_lesSrJYwrM0LIO
-VRcjWHu0a27vKa0 --server=https://api.ocp-vmw.cie.netapp.com:6443
```

    c. podman/docker 명령을 사용하여 OpenShift 사용자 자격 증명을 사용하여 레지스트리에 로그인합니다.

포드맨

```
[netapp-user@rhel7 ~]$ podman login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t) --tls
-verify=false
```

+ 참고: 사용 중인 경우 `kubeadmin` 사용자가 개인 레지스트리에 로그인한 후 비밀번호 대신 토큰을 사용합니다.

도커

```
[netapp-user@rhel7 ~]$ docker login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t)
```

+ 참고: 사용 중인 경우 `kubeadmin` 사용자가 개인 레지스트리에 로그인한 후 비밀번호 대신 토큰을 사용합니다.

    d. 이미지를 밀거나 당깁니다.

포드맨

```
[netapp-user@rhel7 ~]$ podman push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ podman pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

도커

```
[netapp-user@rhel7 ~]$ docker push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ docker pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

솔루션 검증 및 사용 사례

솔루션 검증 및 사용 사례: **NetApp** 사용한 **Red Hat OpenShift**

이 페이지에 제공된 예는 NetApp 과 함께 사용되는 Red Hat OpenShift에 대한 솔루션 검증 및 사용 사례입니다.

- "영구 저장소를 사용하여 Jenkins CI/CD 파이프라인 배포"
- "NetApp 사용하여 Red Hat OpenShift에서 멀티테넌시 구성"
- "NetApp ONTAP 통한 Red Hat OpenShift 가상화"
- "NetApp 사용한 Red Hat OpenShift의 Kubernetes를 위한 고급 클러스터 관리"

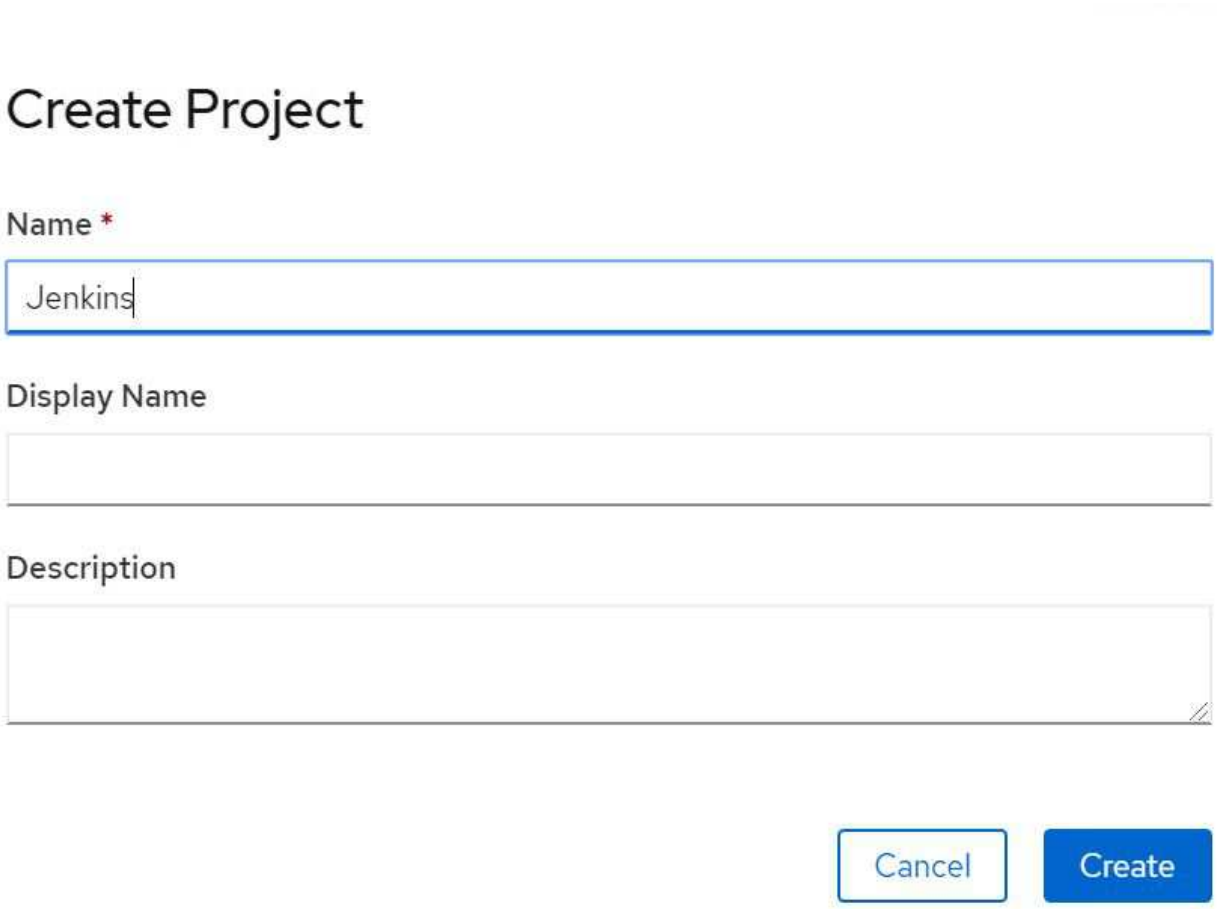
영구 스토리지를 사용한 **Jenkins CI/CD** 파이프라인 배포: **NetApp** 사용한 **Red Hat OpenShift**

이 섹션에서는 Jenkins를 사용하여 솔루션 운영을 검증하기 위해 CI/CD(지속적인 통합/지속적인 전달 또는 배포) 파이프라인을 배포하는 단계를 제공합니다.

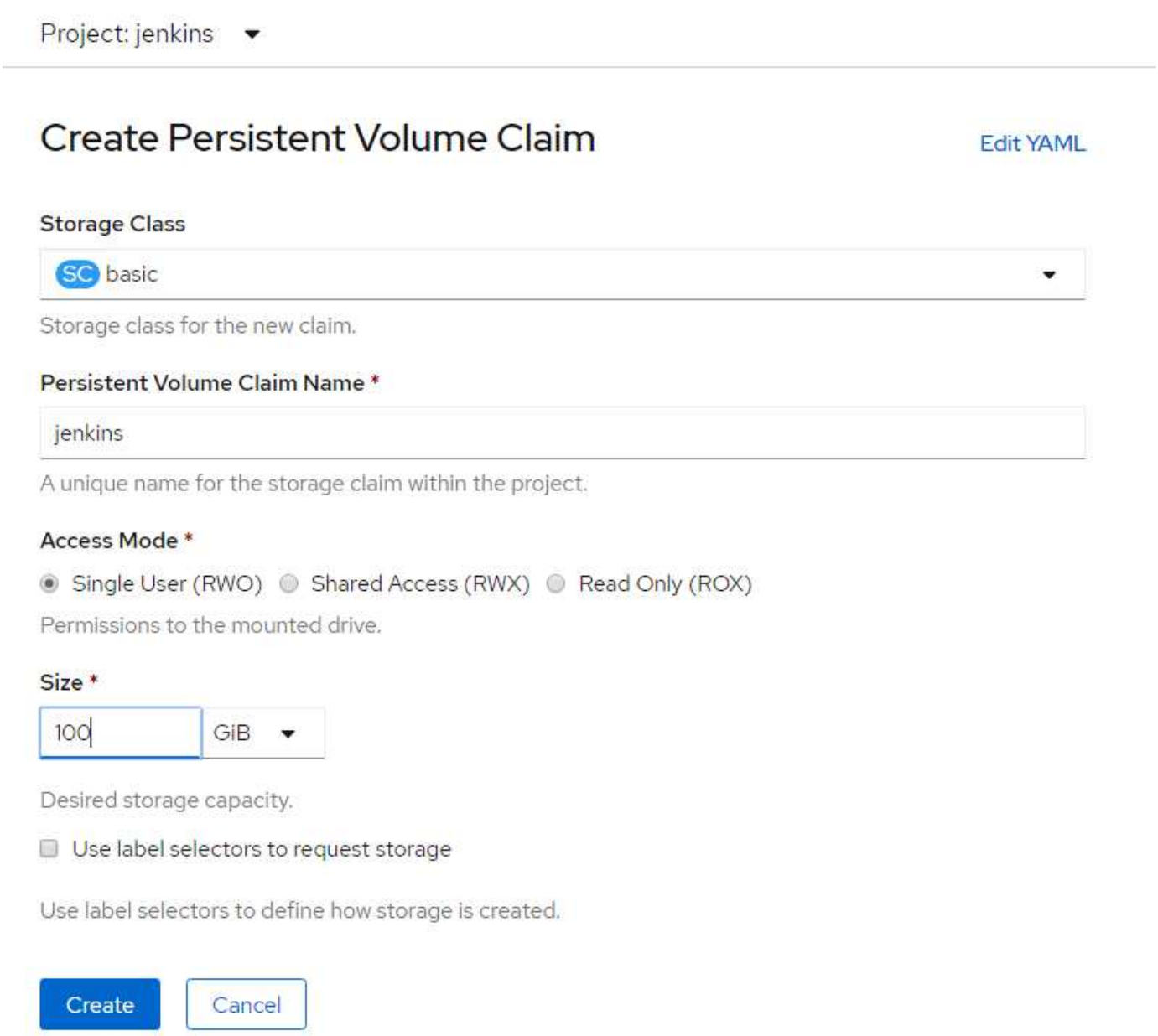
Jenkins 배포에 필요한 리소스를 생성합니다.

Jenkins 애플리케이션을 배포하는 데 필요한 리소스를 생성하려면 다음 단계를 완료하세요.

1. Jenkins라는 이름의 새로운 프로젝트를 만듭니다.



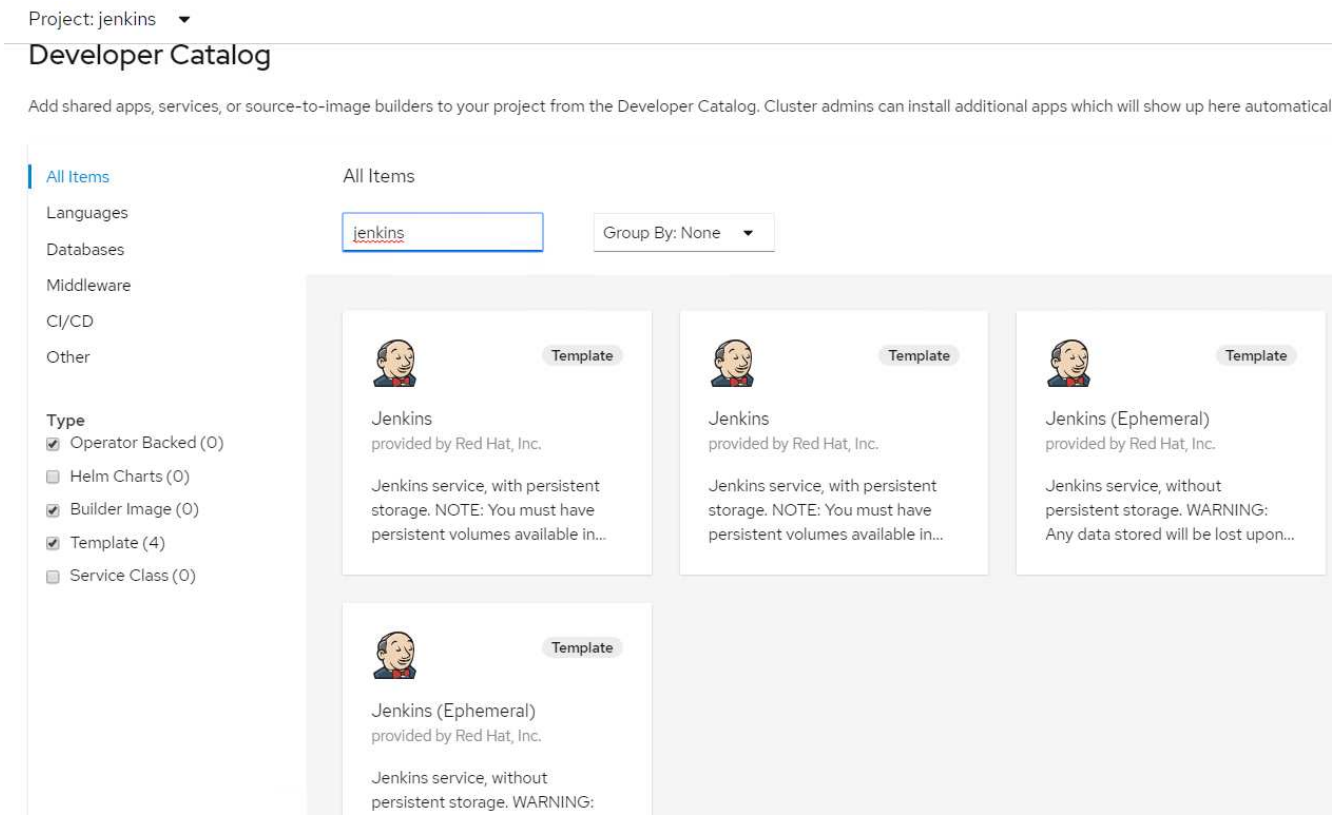
2. 이 예에서는 영구 저장소를 사용하여 Jenkins를 배포했습니다. Jenkins 빌드를 지원하려면 PVC를 만듭니다. 저장소 > 영구 볼륨 클레임으로 이동하여 영구 볼륨 클레임 만들기를 클릭합니다. 생성된 스토리지 클래스를 선택하고, 영구 볼륨 클레임 이름이 jenkins인지 확인하고, 적절한 크기와 액세스 모드를 선택한 다음, 생성을 클릭합니다.



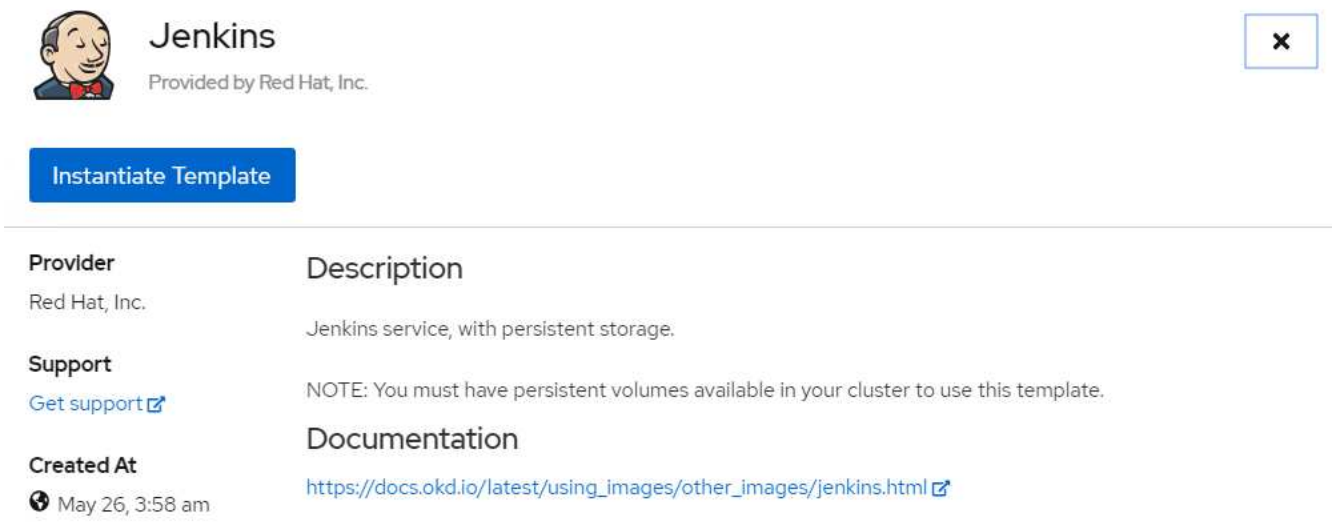
영구 저장소를 사용하여 **Jenkins** 배포

영구 저장소로 Jenkins를 배포하려면 다음 단계를 완료하세요.

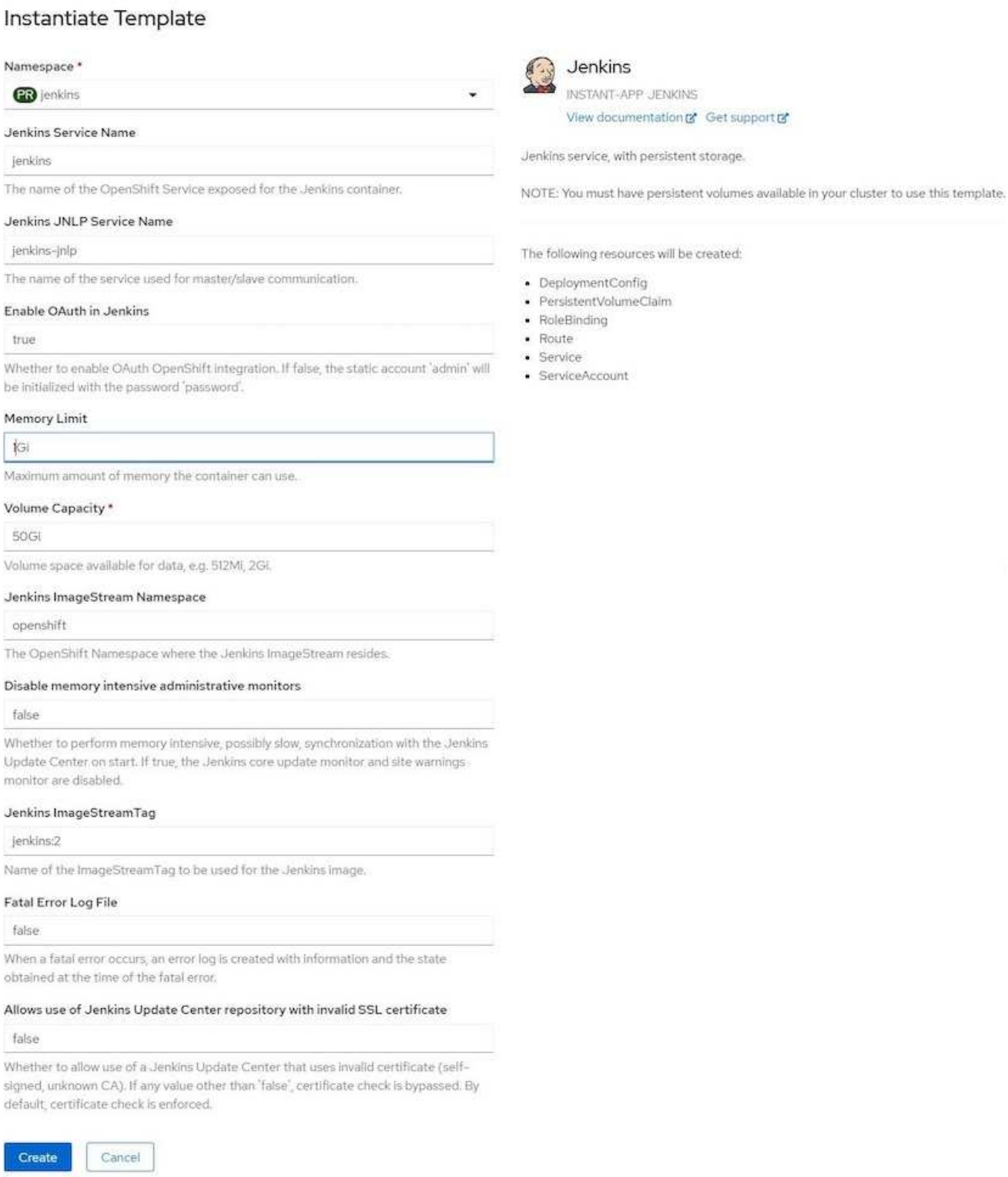
1. 왼쪽 상단에서 역할을 관리자에서 개발자로 변경합니다. +추가를 클릭하고 카탈로그에서를 선택합니다. 키워드로 필터링에서 jenkins를 검색하세요. 영구 저장소가 있는 Jenkins 서비스를 선택하세요.



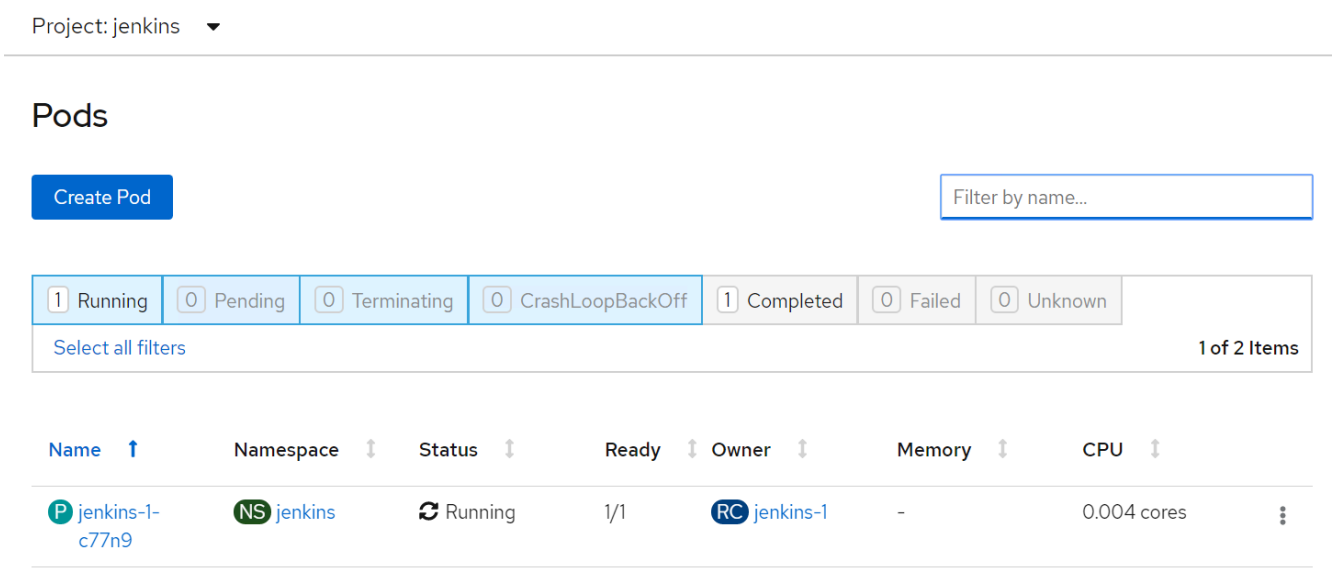
2. 딸깍 하는 소리 `Instantiate Template` .



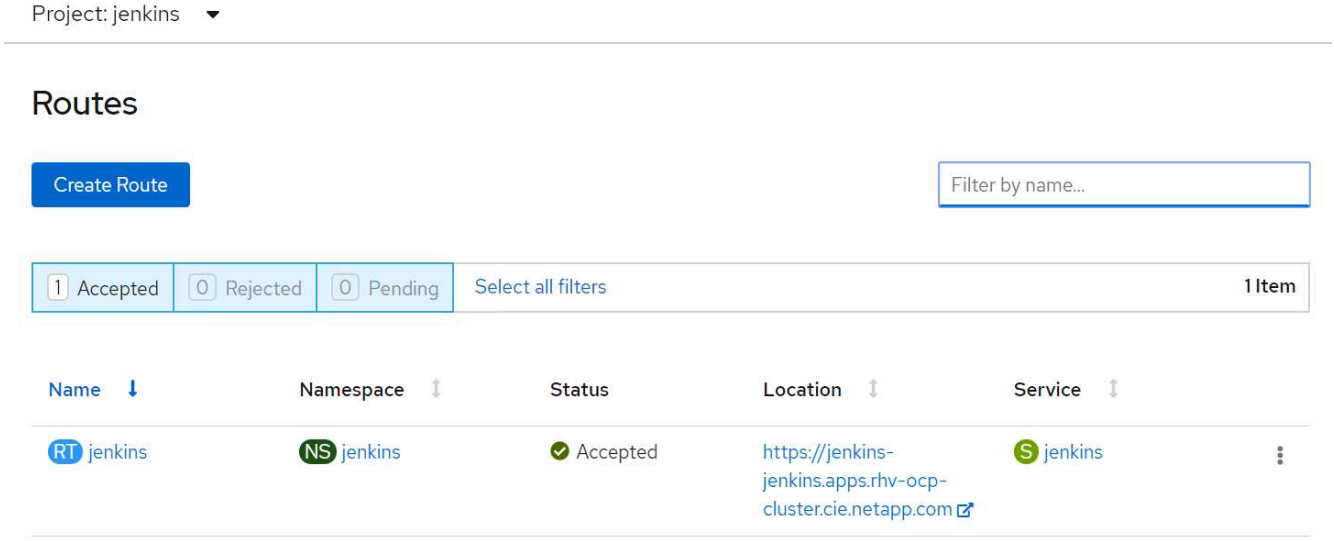
3. 기본적으로 Jenkins 애플리케이션의 세부 정보가 채워집니다. 요구 사항에 따라 매개변수를 수정하고 만들기를 클릭합니다. 이 프로세스는 OpenShift에서 Jenkins를 지원하는 데 필요한 모든 리소스를 생성합니다.



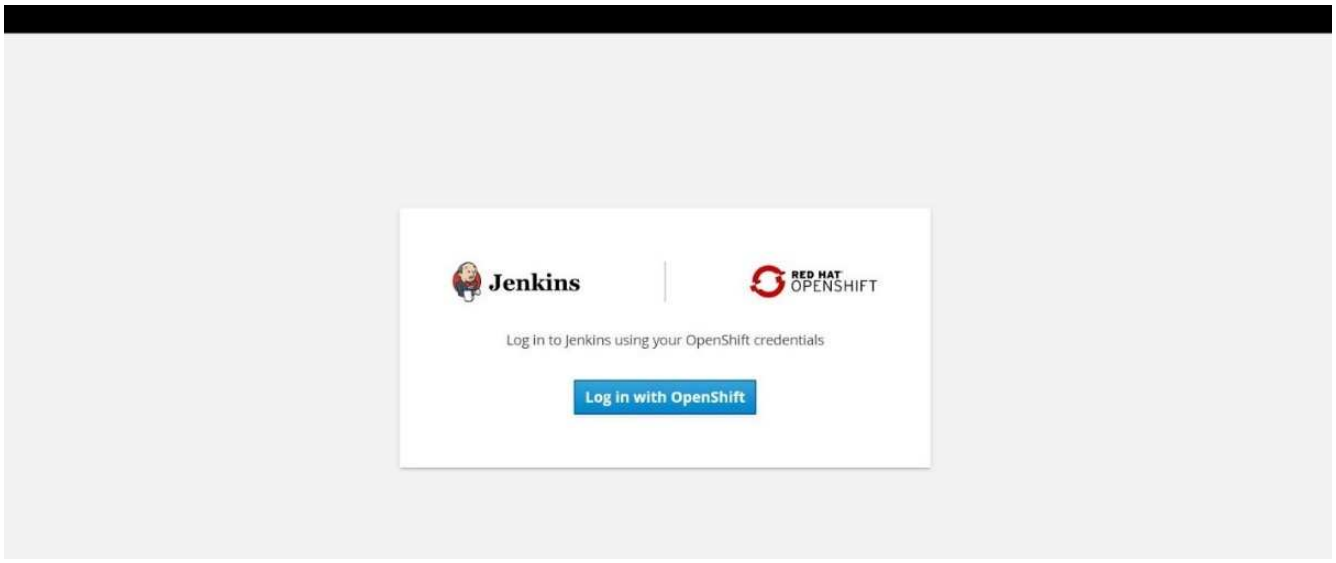
4. Jenkins 포드가 준비 상태에 진입하는 데 약 10~12분이 걸립니다.



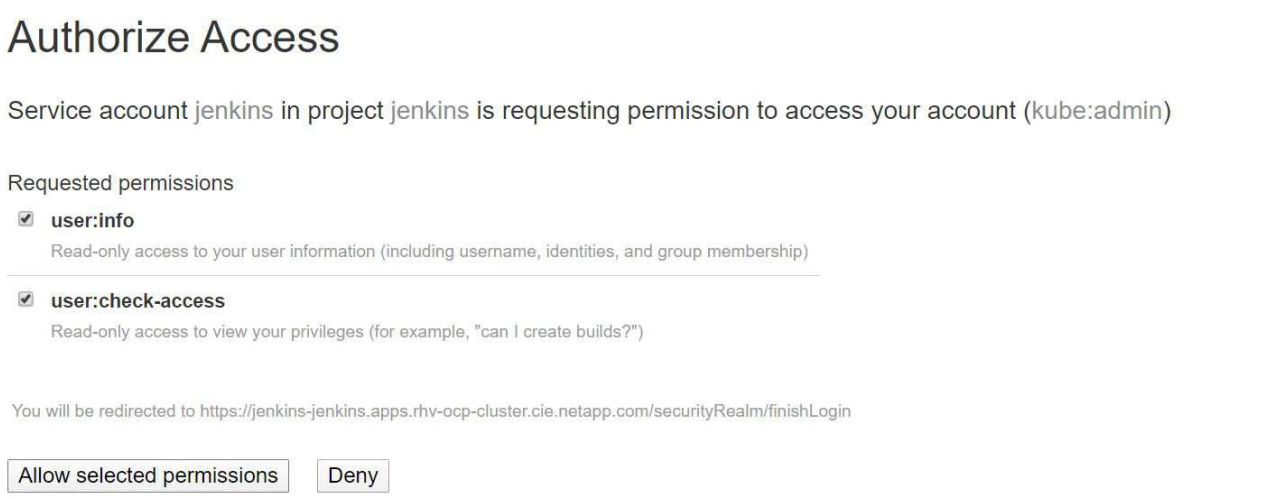
5. 포드가 인스턴스화된 후 네트워킹 > 경로로 이동합니다. Jenkins 웹페이지를 열려면 jenkins 경로에 제공된 URL을 클릭하세요.



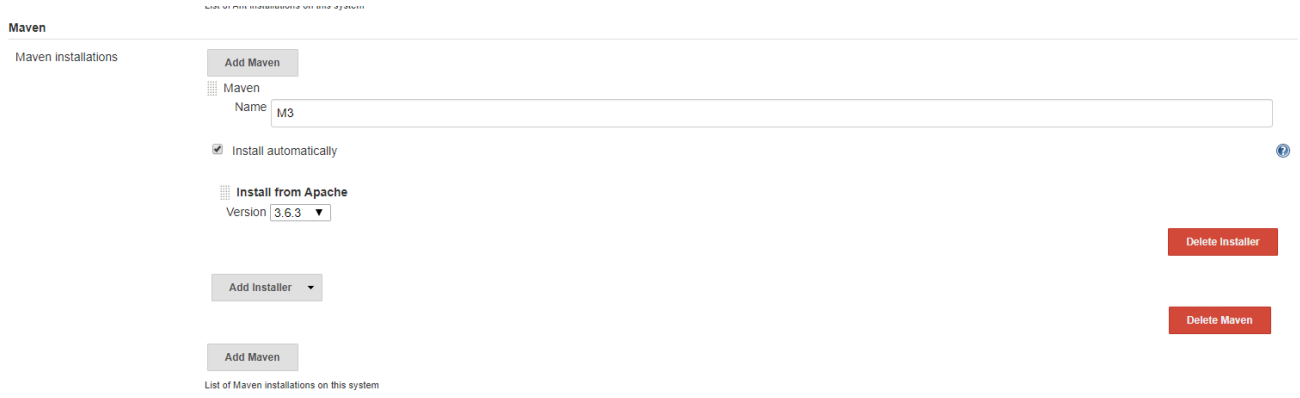
6. Jenkins 앱을 생성하는 동안 OpenShift OAuth가 사용되었으므로 OpenShift로 로그인을 클릭합니다.



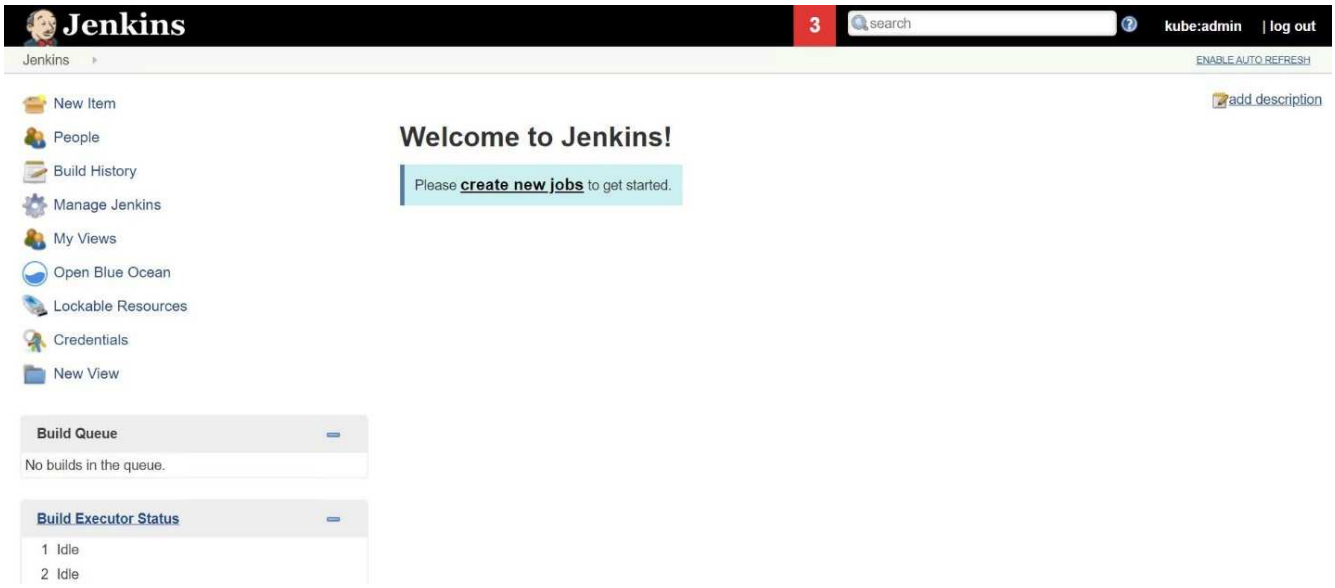
7. Jenkins 서비스 계정에 OpenShift 사용자에 대한 액세스를 승인합니다.



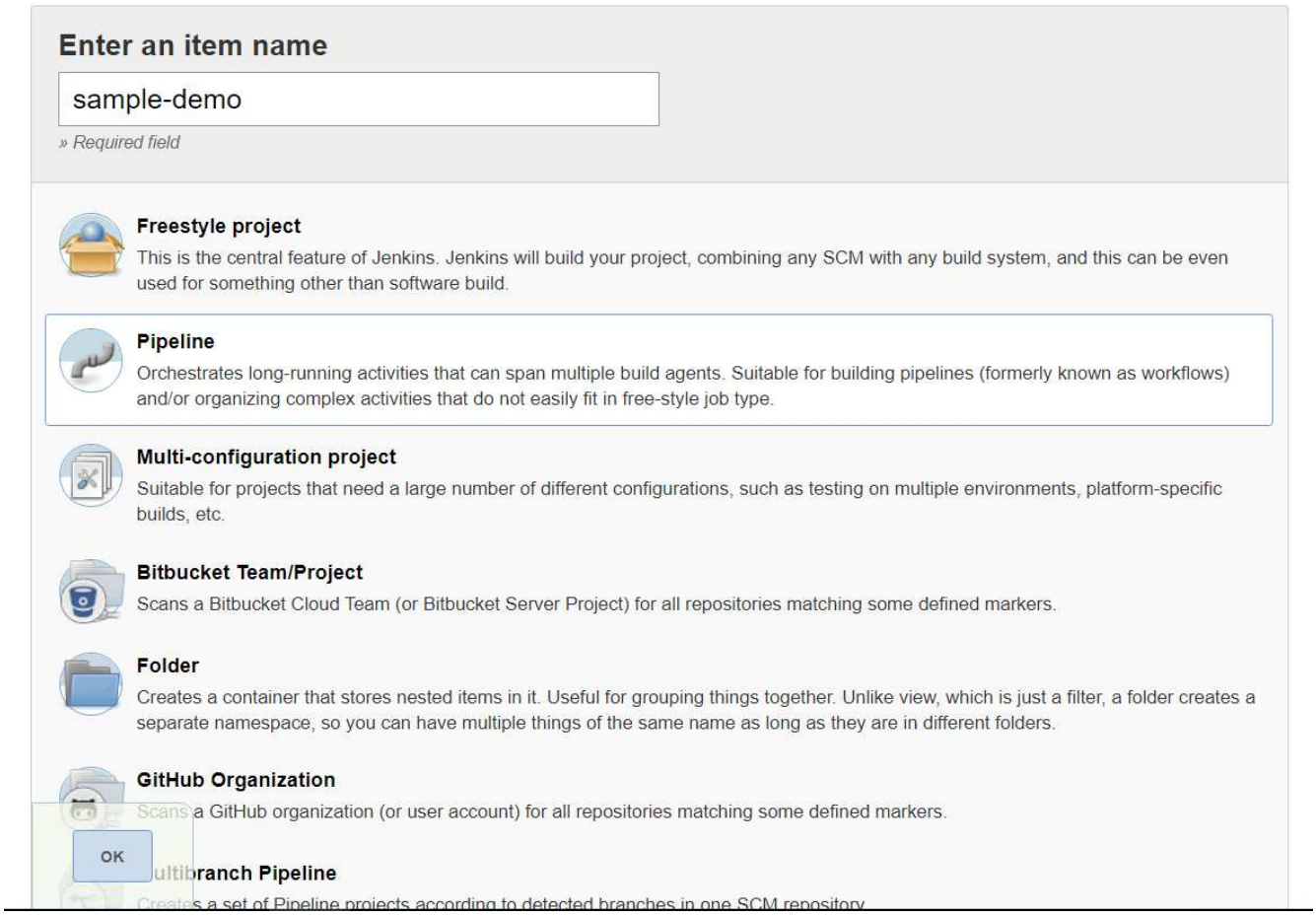
8. Jenkins 환영 페이지가 표시됩니다. Maven 빌드를 사용하고 있으므로 먼저 Maven 설치를 완료하세요. Jenkins 관리 > 글로벌 도구 구성으로 이동한 다음 Maven 하위 제목에서 Maven 추가를 클릭합니다. 원하는 이름을 입력하고 자동 설치 옵션이 선택되어 있는지 확인하세요. Save를 클릭합니다.



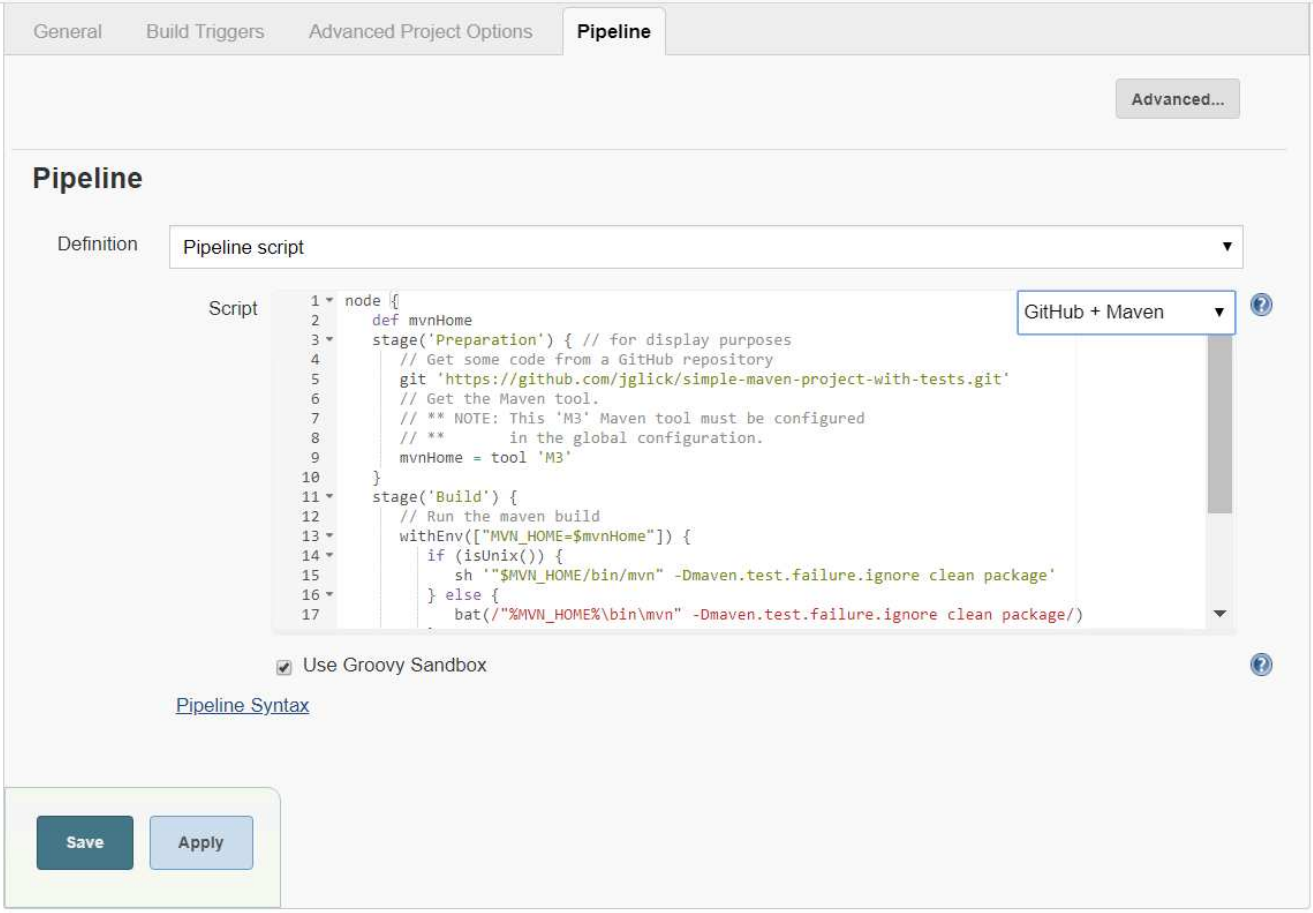
9. 이제 CI/CD 워크플로를 보여주는 파이프라인을 만들 수 있습니다. 홈페이지에서 왼쪽 메뉴에서 '새 작업 만들기' 또는 '새 항목 만들기'를 클릭합니다.



10. 항목 만들기 페이지에서 원하는 이름을 입력하고 파이프라인을 선택한 후 확인을 클릭합니다.



11. 파이프라인 탭을 선택합니다. 샘플 파이프라인 시도 드롭다운 메뉴에서 Github + Maven을 선택합니다. 코드는 자동으로 채워집니다. Save를 클릭합니다.



12. '지금 빌드'를 클릭하면 준비, 빌드, 테스트 단계의 개발을 시작할 수 있습니다. 전체 빌드 프로세스를 완료하고 빌드 결과를 표시하는 데 몇 분이 걸릴 수 있습니다.

13. 코드가 변경될 때마다 파이프라인을 다시 빌드하여 새로운 버전의 소프트웨어를 패치하여 지속적인 통합과 지속적인 배포가 가능해집니다. 이전 버전의 변경 사항을 추적하려면 최근 변경 사항을 클릭하세요.

멀티 테넌시 구성

NetApp 사용하여 **Red Hat OpenShift**에서 멀티테넌시 구성

컨테이너에서 여러 애플리케이션이나 워크로드를 실행하는 많은 조직은 애플리케이션이나 워크로드당 하나의 Red Hat OpenShift 클러스터를 배포하는 경향이 있습니다. 이를 통해 애플리케이션이나 워크로드에 대한 엄격한 격리를 구현하고, 성능을 최적화하고, 보안 취약성을 줄일 수 있습니다. 그러나 각 애플리케이션에 별도의 Red Hat OpenShift 클러스터를 배포하면 고유한 문제가 발생합니다. 각 클러스터를 개별적으로 모니터링하고 관리해야 하므로 운영 오버헤드가 증가하고, 다양한 애플리케이션에 전용 리소스를 할당해야 하므로 비용이 증가하고, 효율적인 확장성이 방해를 받습니다.

이러한 문제를 극복하려면 모든 애플리케이션이나 워크로드를 단일 Red Hat OpenShift 클러스터에서 실행하는 것을 고려할 수 있습니다. 하지만 이러한 아키텍처에서는 리소스 격리와 애플리케이션 보안 취약성이 주요 과제 중 하나입니다. 한 작업 부하에서 발생하는 보안 취약점은 자연스럽게 다른 작업 부하로 확산되어 영향 범위가 넓어질 수 있습니다. 또한, 기본적으로 리소스 할당 정책이 없기 때문에 한 애플리케이션에서 갑자기 통제되지 않은 리소스 사용이 발생하면 다른 애플리케이션의 성능에 영향을 미칠 수 있습니다.

따라서 조직에서는 두 가지 장점을 모두 취할 수 있는 솔루션을 찾습니다. 예를 들어, 모든 워크로드를 단일 클러스터에서 실행하면서도 각 워크로드에 대해 전용 클러스터의 이점을 제공하는 것입니다.

이러한 효과적인 솔루션 중 하나는 Red Hat OpenShift에서 멀티테넌시를 구성하는 것입니다. 멀티테넌시는 여러 테넌트가 리소스, 보안 등을 적절히 분리하여 동일한 클러스터에 공존할 수 있도록 하는 아키텍처입니다. 이러한 맥락에서 테넌트는 특정 사용자 그룹이 독점적인 목적으로 사용하도록 구성된 클러스터 리소스의 하위 집합으로 볼 수 있습니다. Red Hat OpenShift 클러스터에서 멀티테넌시를 구성하면 다음과 같은 이점이 있습니다.

- 클러스터 리소스를 공유함으로써 CapEx 및 OpEx 감소
- 운영 및 관리 비용 절감
- 보안 침해로 인한 교차 오염으로부터 작업 부하 보호
- 리소스 경합으로 인한 예상치 못한 성능 저하로부터 작업 부하 보호

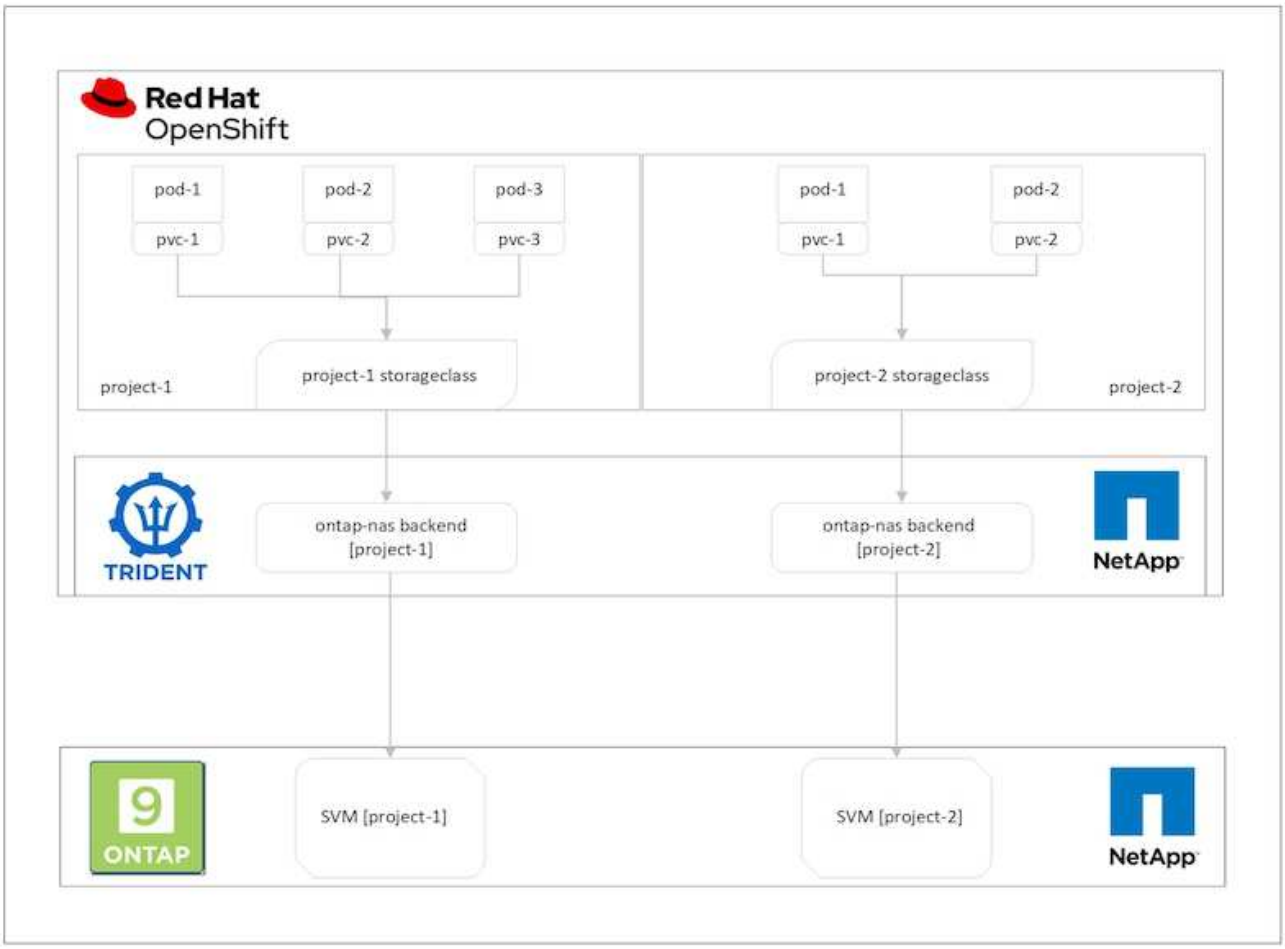
완전히 구현된 멀티 테넌트 OpenShift 클러스터의 경우 컴퓨팅, 스토리지, 네트워킹, 보안 등 다양한 리소스 버킷에 속하는 클러스터 리소스에 대한 할당량과 제한을 구성해야 합니다. 이 솔루션에서는 모든 리소스 버킷의 특정 측면을 다루지만, NetApp ONTAP에서 지원하는 Trident 에서 동적으로 할당된 스토리지 NetApp 에 다중 테넌시를 구성하여 동일한 Red Hat OpenShift 클러스터에서 여러 워크로드가 제공하거나 사용하는 데이터를 격리하고 보호하기 위한 모범 사례에 중점을 ONTAP.

아키텍처

NetApp ONTAP 이 지원하는 Red Hat OpenShift와 Trident 기본적으로 워크로드 간 격리를 제공하지 않지만, 멀티테넌시를 구성하는 데 사용할 수 있는 광범위한 기능을 제공합니다. NetApp ONTAP 지원하는 Trident 를 사용하여 Red Hat OpenShift 클러스터에서 멀티테넌트 솔루션을 설계하는 방법을 더 잘 이해하기 위해 요구 사항 집합이 있는 예를 살펴보고 이를 중심으로 구성을 간략하게 설명하겠습니다.

어떤 조직이 두 개의 프로젝트의 일환으로 두 개의 워크로드를 Red Hat OpenShift 클러스터에서 실행하고 있다고 가정해 보겠습니다. 두 개의 프로젝트는 서로 다른 두 팀이 작업하고 있습니다. 이러한 작업 부하에 대한 데이터는 NetApp ONTAP NAS 백엔드에서 Trident 동적으로 프로비저닝하는 PVC에 저장됩니다. 조직에서는 이 두 가지 작업 부하에 대한 멀티테넌트 솔루션을 설계하고 이러한 프로젝트에 사용되는 리소스를 분리하여 보안과 성능이 유지되도록 해야 하며, 주로 해당 애플리케이션에 서비스를 제공하는 데이터에 중점을 두어야 합니다.

다음 그림은 NetApp ONTAP 이 지원하는 Trident 탑재한 Red Hat OpenShift 클러스터의 멀티테넌트 솔루션을 보여줍니다.



기술 요구 사항

1. NetApp ONTAP 스토리지 클러스터
2. Red Hat OpenShift 클러스터
3. Trident

Red Hat OpenShift – 클러스터 리소스

Red Hat OpenShift 클러스터 관점에서 볼 때 시작할 최상위 리소스는 프로젝트입니다. OpenShift 프로젝트는 전체 OpenShift 클러스터를 여러 개의 가상 클러스터로 나누는 클러스터 리소스로 볼 수 있습니다. 따라서 프로젝트 수준의 격리는 멀티테넌시를 구성하기 위한 기반을 제공합니다.

다음으로는 클러스터에서 RBAC를 구성하는 것입니다. 가장 좋은 방법은 단일 프로젝트나 워크로드를 작업하는 모든 개발자를 ID 공급자(IdP)의 단일 사용자 그룹으로 구성하는 것입니다. Red Hat OpenShift는 IdP 통합과 사용자 그룹 동기화를 허용하므로 IdP의 사용자와 그룹을 클러스터로 가져올 수 있습니다. 이를 통해 클러스터 관리자는 프로젝트에 전용된 클러스터 리소스에 대한 액세스를 해당 프로젝트에서 작업하는 사용자 그룹으로 분리하여 클러스터 리소스에 대한 무단 액세스를 제한할 수 있습니다. Red Hat OpenShift와 IdP 통합에 대해 자세히 알아보려면 설명서를 참조하세요. "여기" .

NetApp ONTAP

각 프로젝트의 스토리지에 생성된 볼륨이 별도의 스토리지에 생성된 것처럼 호스트에 나타나도록 하려면 Red Hat OpenShift 클러스터의 영구 스토리지 공급자 역할을 하는 공유 스토리지를 격리하는 것이 중요합니다. 이렇게 하려면

NetApp ONTAP 에 프로젝트나 워크로드만큼 많은 SVM(스토리지 가상 머신)을 만들고 각 SVM을 워크로드에 전용으로 지정합니다.

Trident

NetApp ONTAP 에서 다양한 프로젝트에 대해 서로 다른 SVM을 생성한 후에는 각 SVM을 다른 Trident 백엔드에 매핑해야 합니다. Trident 의 백엔드 구성은 OpenShift 클러스터 리소스에 영구 저장소를 할당하며, 이를 위해서는 SVM의 세부 정보를 매핑해야 합니다. 이는 최소한 백엔드의 프로토콜 드라이버여야 합니다. 선택적으로, 볼륨이 저장소에 프로비저닝되는 방식을 정의하고 볼륨 크기나 집계 사용 등에 대한 제한을 설정할 수 있습니다. Trident 백엔드 정의에 대한 세부 사항은 다음에서 찾을 수 있습니다. "여기" .

Red Hat OpenShift – 스토리지 리소스

Trident 백엔드를 구성한 후 다음 단계는 StorageClass를 구성하는 것입니다. 백엔드 수만큼 스토리지 클래스를 구성하여 각 스토리지 클래스가 하나의 백엔드에서만 볼륨을 스핀업할 수 있도록 합니다. 스토리지 클래스를 정의하는 동안 storagePools 매개변수를 사용하여 StorageClass를 특정 Trident 백엔드에 매핑할 수 있습니다. 저장 클래스를 정의하는 세부 사항은 다음에서 찾을 수 있습니다. "여기" . 따라서 StorageClass에서 Trident 백엔드로의 일대일 매핑이 이루어져 하나의 SVM을 가리킵니다. 이를 통해 해당 프로젝트에 할당된 StorageClass를 통한 모든 스토리지 클레임이 해당 프로젝트에만 전담된 SVM을 통해 처리됩니다.

저장소 클래스는 네임스페이스 리소스가 아니므로 다른 네임스페이스 또는 프로젝트에 있는 포드가 한 프로젝트의 저장소 클래스에 대한 저장소 클레임을 거부하도록 하려면 어떻게 해야 할까요? 정답은 ResourceQuotas를 사용하는 것입니다. ResourceQuotas는 프로젝트당 리소스의 총 사용량을 제어하는 객체입니다. 이를 통해 프로젝트 내 객체가 소비할 수 있는 리소스의 수와 총량을 제한할 수 있습니다. ResourceQuotas를 사용하면 프로젝트의 거의 모든 리소스를 제한할 수 있으며, 이를 효율적으로 사용하면 조직에서 리소스의 과도한 프로비저닝이나 과소비로 인한 비용과 중단을 줄이는 데 도움이 될 수 있습니다. 문서를 참조하세요 "여기" 자세한 내용은.

이 사용 사례의 경우 특정 프로젝트의 포드가 해당 프로젝트에 전용되지 않은 스토리지 클래스의 스토리지를 청구하지 못하도록 제한해야 합니다. 이를 위해서는 다른 스토리지 클래스에 대한 영구 볼륨 클레임을 제한해야 합니다. `<storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims` 0으로. 또한, 클러스터 관리자는 프로젝트의 개발자가 ResourceQuotas를 수정할 수 있는 액세스 권한이 없도록 해야 합니다.

구성

모든 멀티테넌트 솔루션의 경우 사용자는 필요 이상으로 클러스터 리소스에 액세스할 수 없습니다. 따라서 멀티테넌시 구성의 일부로 구성될 리소스 전체 세트는 각 프로젝트에서 작업하는 클러스터 관리자, 스토리지 관리자 및 개발자 간에 나뉩니다.

다음 표는 다양한 사용자가 수행해야 하는 다양한 작업을 간략하게 설명합니다.

역할	작업
클러스터 관리자	다양한 애플리케이션이나 워크로드에 대한 프로젝트를 만듭니다.
	storage-admin에 대한 ClusterRoles 및 RoleBindings를 생성합니다.
	특정 프로젝트에 대한 액세스를 할당하는 개발자를 위한 역할 및 역할 바인딩을 만듭니다.
	[선택 사항] 특정 노드에서 Pod를 예약하도록 프로젝트 구성

역할	작업
저장소 관리자	NetApp ONTAP 에서 SVM 만들기
	Trident 백엔드 만들기
	스토리지 클래스 생성
	저장소 ResourceQuotas 생성
개발자	할당된 프로젝트에서 PVC 또는 Pod를 생성하거나 패치할 수 있는 액세스 권한을 검증합니다.
	다른 프로젝트에서 PVC 또는 Pod를 생성하거나 패치할 수 있는 액세스 권한 확인
	프로젝트, 리소스 할당량 및 스토리지 클래스를 보거나 편집할 수 있는 액세스 권한 확인

구성

NetApp 사용하여 Red Hat OpenShift에서 멀티테넌시를 구성하기 위한 전제 조건은 다음과 같습니다.

필수 조건

- NetApp ONTAP 클러스터
- Red Hat OpenShift 클러스터
- 클러스터에 Trident 설치
- tridentctl 및 oc 도구가 설치되고 $PATH에 추가된 관리 워크스테이션
- ONTAP 에 대한 관리자 액세스
- OpenShift 클러스터에 대한 클러스터 관리자 액세스
- 클러스터는 Identity Provider와 통합되어 있습니다.
- ID 공급자는 서로 다른 팀의 사용자를 효율적으로 구별하도록 구성됩니다.

구성**:** 클러스터 관리 작업

다음 작업은 Red Hat OpenShift 클러스터 관리자가 수행합니다.

1. cluster-admin으로 Red Hat OpenShift 클러스터에 로그인합니다.
2. 서로 다른 프로젝트에 해당하는 두 개의 프로젝트를 만듭니다.

```
oc create namespace project-1
oc create namespace project-2
```

3. project-1에 대한 개발자 역할을 만듭니다.

```
cat << EOF | oc create -f -
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: project-1
  name: developer-project-1
rules:
  - verbs:
      - '*'
    apiGroups:
      - apps
      - batch
      - autoscaling
      - extensions
      - networking.k8s.io
      - policy
      - apps.openshift.io
      - build.openshift.io
      - image.openshift.io
      - ingress.operator.openshift.io
      - route.openshift.io
      - snapshot.storage.k8s.io
      - template.openshift.io
    resources:
      - '*'
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - bindings
      - configmaps
      - endpoints
      - events
      - persistentvolumeclaims
      - pods
      - pods/log
      - pods/attach
      - podtemplates
      - replicationcontrollers
      - services
      - limitranges
      - namespaces
      - componentstatuses
      - nodes
  - verbs:
      - '*'
```

```
    apiGroups:
      - trident.netapp.io
    resources:
      - tridentsnapshots
EOF
```

이 섹션에 제공된 역할 정의는 단지 예시일 뿐입니다. 개발자 역할은 최종 사용자 요구 사항에 따라 정의되어야 합니다.

1. 마찬가지로, 프로젝트-2에 대한 개발자 역할을 만듭니다.
2. 모든 OpenShift 및 NetApp 스토리지 리소스는 일반적으로 스토리지 관리자가 관리합니다. 스토리지 관리자의 액세스는 Trident 설치될 때 생성되는 Trident 운영자 역할에 의해 제어됩니다. 이 외에도 스토리지 관리자는 ResourceQuotas에 액세스하여 스토리지 사용 방식을 제어해야 합니다.
3. 클러스터의 모든 프로젝트에서 ResourceQuotas를 관리하는 역할을 만들어 스토리지 관리자에 연결합니다.

```
cat << EOF | oc create -f -
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: resource-quotas-role
rules:
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - resourcequotas
  - verbs:
      - '*'
    apiGroups:
      - quota.openshift.io
    resources:
      - '*'
EOF
```

4. 클러스터가 조직의 ID 공급자와 통합되어 있는지, 사용자 그룹이 클러스터 그룹과 동기화되어 있는지 확인하세요. 다음 예에서는 ID 공급자가 클러스터와 통합되고 사용자 그룹과 동기화되었음을 보여줍니다.

```
$ oc get groups
NAME                          USERS
ocp-netapp-storage-admins     ocp-netapp-storage-admin
ocp-project-1                 ocp-project-1-user
ocp-project-2                 ocp-project-2-user
```

1. 스토리지 관리자를 위해 ClusterRoleBindings를 구성합니다.

```
cat << EOF | oc create -f -
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: netapp-storage-admin-trident-operator
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-netapp-storage-admins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-operator
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: netapp-storage-admin-resource-quotas-cr
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-netapp-storage-admins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: resource-quotas-role
EOF
```

스토리지 관리자의 경우 trident-operator와 resource-quotas라는 두 가지 역할을 바인딩해야 합니다.

1. project-1의 해당 그룹(ocp-project-1)에 developer-project-1 역할을 바인딩하는 개발자를 위한 RoleBindings를 생성합니다.

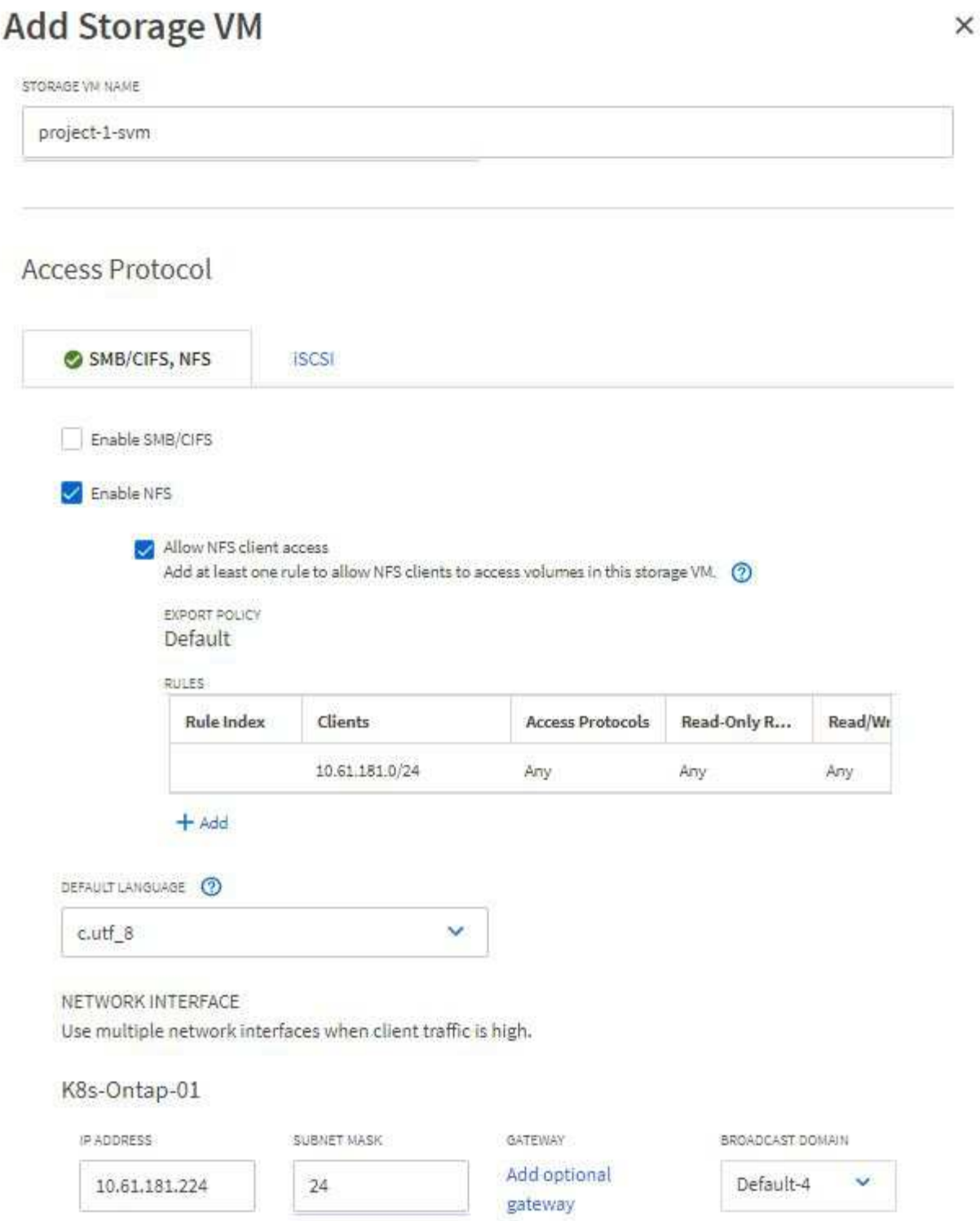
```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-1-developer
  namespace: project-1
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-project-1
EOF
```

2. 마찬가지로, 개발자 역할을 project-2의 해당 사용자 그룹에 바인딩하는 개발자용 RoleBindings를 생성합니다.

구성: 스토리지 관리 작업

다음 리소스는 스토리지 관리자가 구성해야 합니다.

1. NetApp ONTAP 클러스터에 관리자로 로그인합니다.
2. 저장소 > 저장소 VM으로 이동하여 추가를 클릭합니다. 필요한 세부 정보를 제공하여 프로젝트 1과 프로젝트 2에 대한 두 개의 SVM을 만듭니다. 또한 SVM과 해당 리소스를 관리하기 위해 vsadmin 계정을 만듭니다.



1. 스토리지 관리자로 Red Hat OpenShift 클러스터에 로그인합니다.
2. 프로젝트-1에 대한 백엔드를 만들고 이를 해당 프로젝트에 전용된 SVM에 매핑합니다. NetApp ONTAP 클러스터 관리자를 사용하는 대신 SVM의 vsadmin 계정을 사용하여 백엔드를 SVM에 연결할 것을 권장합니다.

```
cat << EOF | tridentctl -n trident create backend -f
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "nfs_project_1",
    "managementLIF": "172.21.224.210",
    "dataLIF": "10.61.181.224",
    "svm": "project-1-svm",
    "username": "vsadmin",
    "password": "NetApp123"
}
EOF
```

이 예제에서는 ontap-nas 드라이버를 사용하고 있습니다. 사용 사례에 따라 백엔드를 생성할 때 적절한 드라이버를 사용하세요.

트라이던트 프로젝트에는 Trident 설치되어 있다고 가정합니다.

1. 마찬가지로 프로젝트 2에 대한 Trident 백엔드를 만들고 프로젝트 2에 전용된 SVM에 매핑합니다.
2. 다음으로, 저장 클래스를 생성합니다. project-1에 대한 스토리지 클래스를 만들고 storagePools 매개변수를 설정하여 project-1에 전용된 백엔드의 스토리지 풀을 사용하도록 구성합니다.

```
cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: project-1-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
  storagePools: "nfs_project_1:.*"
EOF
```

3. 마찬가지로, project-2에 대한 스토리지 클래스를 만들고 project-2에 전용된 백엔드의 스토리지 풀을 사용하도록 구성합니다.
4. 다른 프로젝트에 전용된 스토리지 클래스에서 스토리지를 요청하는 project-1의 리소스를 제한하기 위해 ResourceQuota를 생성합니다.

```
cat << EOF | oc create -f -
kind: ResourceQuota
apiVersion: v1
metadata:
  name: project-1-sc-rq
  namespace: project-1
spec:
  hard:
    project-2-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
EOF
```

5. 마찬가지로, 다른 프로젝트에 전용된 스토리지 클래스에서 스토리지를 요청하는 project-2의 리소스를 제한하기 위해 ResourceQuota를 생성합니다.

확인

이전 단계에서 구성된 멀티테넌트 아키텍처를 검증하려면 다음 단계를 완료하세요.

할당된 프로젝트에서 **PVC** 또는 **Pod**를 생성하기 위한 액세스 권한 확인

1. ocp-project-1-user, project-1의 개발자로 로그인합니다.
2. 새로운 프로젝트를 생성하려면 액세스를 확인하세요.

```
oc create ns sub-project-1
```

3. project-1에 할당된 스토리지 클래스를 사용하여 project-1에 PVC를 생성합니다.

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-1
  namespace: project-1
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-1-sc
EOF
```

4. PVC와 연관된 PV를 확인하세요.

```
oc get pv
```

5. PV와 해당 볼륨이 NetApp ONTAP 의 project-1 전용 SVM에 생성되었는지 확인합니다.

```
volume show -vserver project-1-svm
```

6. project-1에 포드를 생성하고 이전 단계에서 생성한 PVC를 마운트합니다.

```
cat << EOF | oc create -f -
kind: Pod
apiVersion: v1
metadata:
  name: test-pvc-pod
  namespace: project-1
spec:
  volumes:
    - name: test-pvc-project-1
      persistentVolumeClaim:
       claimName: test-pvc-project-1
  containers:
    - name: test-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: test-pvc-project-1
EOF
```

7. 포드가 실행 중인지, 볼륨을 마운트했는지 확인하세요.

```
oc describe pods test-pvc-pod -n project-1
```

다른 프로젝트에서 **PVC** 또는 포드를 생성하거나 다른 프로젝트에 전용된 리소스를 사용할 수 있는 액세스 권한을 검증합니다**.**

1. ocp-project-1-user, project-1의 개발자로 로그인합니다.
2. 프로젝트-2에 할당된 스토리지 클래스를 사용하여 프로젝트-1에 PVC를 생성합니다.

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-1-sc-2
  namespace: project-1
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-2-sc
EOF
```

3. 프로젝트-2에서 PVC를 만듭니다.

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-2-sc-1
  namespace: project-2
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-1-sc
EOF
```

4. PVC를 확인하세요 `test-pvc-project-1-sc-2` 그리고 `test-pvc-project-2-sc-1` 생성되지 않았습니다.

```
oc get pvc -n project-1
oc get pvc -n project-2
```

5. project-2에 포드를 만듭니다.

```
cat << EOF | oc create -f -
kind: Pod
apiVersion: v1
metadata:
  name: test-pvc-pod
  namespace: project-1
spec:
  containers:
    - name: test-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
EOF
```

프로젝트, 리소스 할당량 및 스토리지 클래스를 보고 편집할 수 있는 액세스 권한을 확인합니다.

1. ocp-project-1-user, project-1의 개발자로 로그인합니다.
2. 새로운 프로젝트를 생성하려면 액세스를 확인하세요.

```
oc create ns sub-project-1
```

3. 프로젝트 보기에 대한 액세스를 검증합니다.

```
oc get ns
```

4. 사용자가 project-1에서 ResourceQuotas를 볼 수 있거나 편집할 수 있는지 확인하세요.

```
oc get resourcequotas -n project-1
oc edit resourcequotas project-1-sc-rq -n project-1
```

5. 사용자가 스토리지 클래스를 볼 수 있는 권한이 있는지 확인합니다.

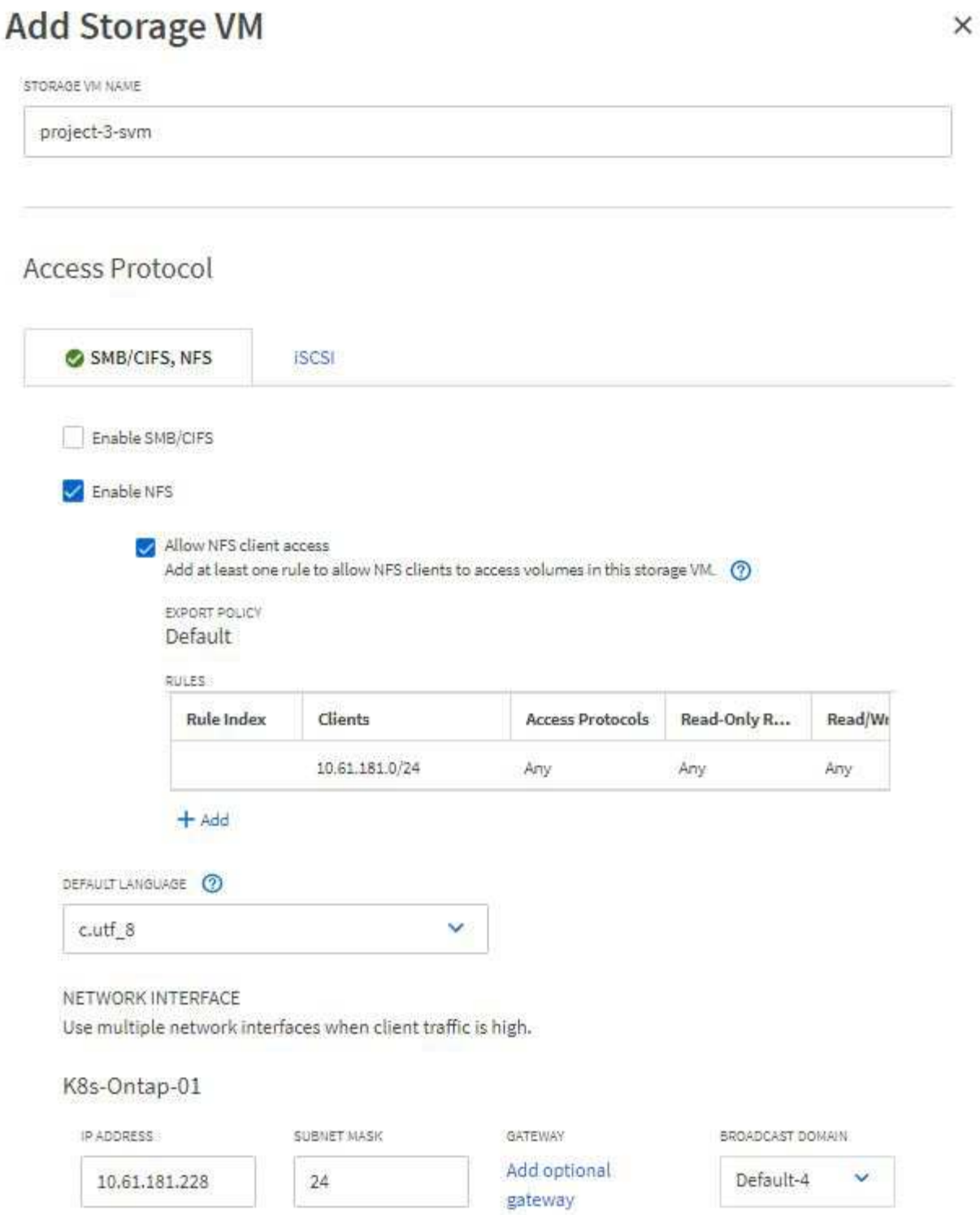
```
oc get sc
```

6. 스토리지 클래스를 설명하려면 액세스를 확인하세요.
7. 사용자가 스토리지 클래스를 편집할 수 있는 권한을 검증합니다.

```
oc edit sc project-1-sc
```

멀티테넌트 구성에서 스토리지 리소스가 있는 새 프로젝트를 추가하려면 멀티테넌시가 위반되지 않도록 추가 구성이 필요합니다. 멀티테넌트 클러스터에 더 많은 프로젝트를 추가하려면 다음 단계를 완료하세요.

1. 스토리지 관리자로 NetApp ONTAP 클러스터에 로그인합니다.
2. 로 이동 `Storage → Storage VMs` 그리고 클릭하세요 `Add` . 프로젝트 3에 전념하는 새로운 SVM을 만듭니다. 또한 SVM과 해당 리소스를 관리하기 위해 vsadmin 계정을 만듭니다.



1. 클러스터 관리자로 Red Hat OpenShift 클러스터에 로그인합니다.
2. 새로운 프로젝트를 만듭니다.

```
oc create ns project-3
```

3. project-3의 사용자 그룹이 IdP에 생성되었고 OpenShift 클러스터와 동기화되었는지 확인하세요.

```
oc get groups
```

4. 프로젝트-3에 대한 개발자 역할을 만듭니다.

```
cat << EOF | oc create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: project-3
  name: developer-project-3
rules:
  - verbs:
      - '*'
    apiGroups:
      - apps
      - batch
      - autoscaling
      - extensions
      - networking.k8s.io
      - policy
      - apps.openshift.io
      - build.openshift.io
      - image.openshift.io
      - ingress.operator.openshift.io
      - route.openshift.io
      - snapshot.storage.k8s.io
      - template.openshift.io
    resources:
      - '*'
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - bindings
      - configmaps
      - endpoints
      - events
      - persistentvolumeclaims
      - pods
      - pods/log
      - pods/attach
      - podtemplates
```

```
      - replicationcontrollers
      - services
      - limitranges
      - namespaces
      - componentstatuses
      - nodes
  - verbs:
      - '*'
    apiGroups:
      - trident.netapp.io
    resources:
      - tridentsnapshots
EOF
```

이 섹션에 제공된 역할 정의는 단지 예시일 뿐입니다. 개발자 역할은 최종 사용자 요구 사항에 따라 정의되어야 합니다.

1. 프로젝트 3의 개발자를 위한 RoleBinding을 생성하여 프로젝트 3의 해당 그룹(ocp-project-3)에 개발자-프로젝트-3 역할을 바인딩합니다.

```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-3-developer
  namespace: project-3
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-3
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-project-3
EOF
```

2. 스토리지 관리자로 Red Hat OpenShift 클러스터에 로그인합니다.
3. Trident 백엔드를 만들고 이를 project-3에 전용된 SVM에 매핑합니다. NetApp ONTAP 클러스터 관리자를 사용하는 대신 SVM의 vsadmin 계정을 사용하여 백엔드를 SVM에 연결할 것을 권장합니다.

```
cat << EOF | tridentctl -n trident create backend -f
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "nfs_project_3",
    "managementLIF": "172.21.224.210",
    "dataLIF": "10.61.181.228",
    "svm": "project-3-svm",
    "username": "vsadmin",
    "password": "NetApp!23"
}
EOF
```

이 예제에서는 ontap-nas 드라이버를 사용하고 있습니다. 사용 사례에 따라 백엔드를 생성하기 위해 적절한 드라이버를 사용합니다.

트라이던트 프로젝트에는 Trident 설치되어 있다고 가정합니다.

1. 프로젝트 3에 대한 스토리지 클래스를 만들고 프로젝트 3에 전용된 백엔드의 스토리지 풀을 사용하도록 구성합니다.

```
cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: project-3-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
  storagePools: "nfs_project_3:.*"
EOF
```

2. 다른 프로젝트에 전용된 스토리지 클래스에서 스토리지를 요청하는 project-3의 리소스를 제한하기 위해 ResourceQuota를 생성합니다.

```
cat << EOF | oc create -f -
kind: ResourceQuota
apiVersion: v1
metadata:
  name: project-3-sc-rq
  namespace: project-3
spec:
  hard:
    project-1-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
    project-2-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
EOF
```

3. 다른 프로젝트의 ResourceQuotas에 패치를 적용하여 해당 프로젝트의 리소스가 project-3에 전용된 스토리지 클래스의 스토리지에 액세스하는 것을 제한합니다.

```
oc patch resourcequotas project-1-sc-rq -n project-1 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
oc patch resourcequotas project-2-sc-rq -n project-2 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
```

쿠버네티스를 위한 고급 클러스터 관리

Kubernetes를 위한 고급 클러스터 관리**: NetApp** 사용한 **Red Hat OpenShift -** 개요

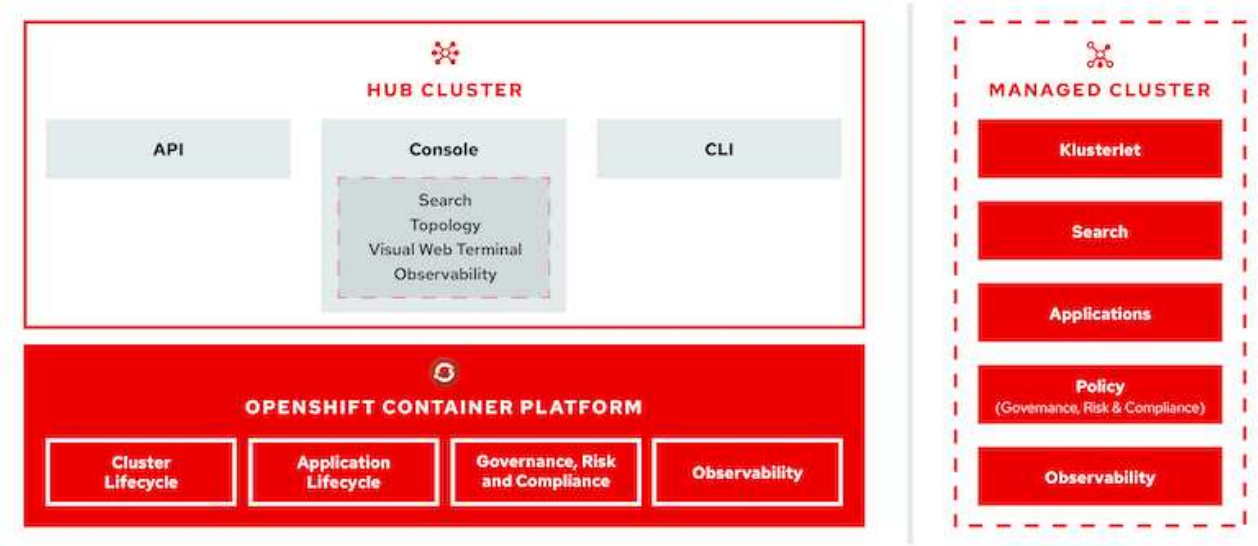
컨테이너화된 애플리케이션이 개발 단계에서 운영 단계로 전환됨에 따라 많은 조직에서는 해당 애플리케이션의 테스트와 배포를 지원하기 위해 여러 개의 Red Hat OpenShift 클러스터가 필요합니다. 이와 관련하여 조직은 일반적으로 OpenShift 클러스터에서 여러 애플리케이션이나 워크로드를 호스팅합니다. 따라서 각 조직은 일련의 클러스터를 관리하게 되고, OpenShift 관리자는 여러 온프레미스 데이터 센터와 퍼블릭 클라우드에 걸쳐 있는 다양한 환경에서 여러 클러스터를 관리하고 유지 관리해야 하는 추가적인 과제에 직면하게 됩니다. 이러한 과제를 해결하기 위해 Red Hat은 Kubernetes를 위한 고급 클러스터 관리를 도입했습니다.

Kubernetes용 Red Hat Advanced Cluster Management를 사용하면 다음 작업을 수행할 수 있습니다.

1. 데이터 센터와 퍼블릭 클라우드에서 여러 클러스터를 생성, 가져오기 및 관리합니다.
2. 단일 콘솔에서 여러 클러스터에 애플리케이션이나 워크로드를 배포하고 관리합니다.
3. 다양한 클러스터 리소스의 상태 및 상태를 모니터링하고 분석합니다.
4. 여러 클러스터에서 보안 규정 준수를 모니터링하고 시행합니다.

Kubernetes용 Red Hat Advanced Cluster Management는 Red Hat OpenShift 클러스터에 추가 기능으로 설치되며,

이 클러스터를 모든 작업의 중앙 컨트롤러로 사용합니다. 이 클러스터는 허브 클러스터로 알려져 있으며, 사용자가 Advanced Cluster Management에 연결할 수 있는 관리 평면을 제공합니다. 고급 클러스터 관리 콘솔을 통해 가져오거나 생성된 다른 모든 OpenShift 클러스터는 허브 클러스터에서 관리되며 관리형 클러스터라고 합니다. 클러스터 수명 주기 관리, 애플리케이션 수명 주기 관리, 관찰 가능성, 보안 규정 준수와 관련된 다양한 활동에 대한 요청을 처리하기 위해 관리되는 클러스터에 Klusterlet이라는 에이전트를 설치합니다.



자세한 내용은 설명서를 참조하세요. "여기" .

Kubernetes용 ACM 배포

Kubernetes를 위한 고급 클러스터 관리 배포

이 섹션에서는 NetApp 과 함께 Red Hat OpenShift에서 Kubernetes를 위한 고급 클러스터 관리에 대해 설명합니다.

필수 조건

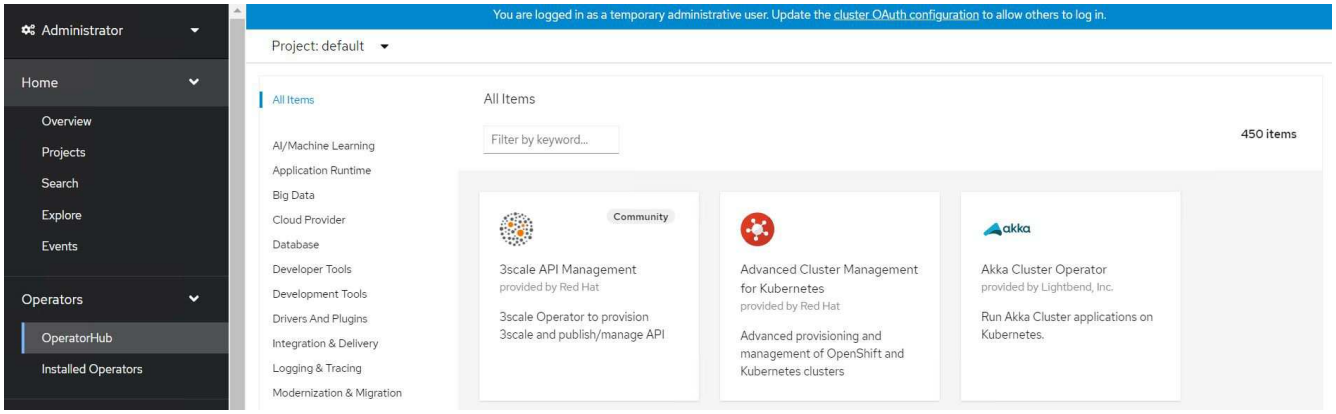
1. 허브 클러스터용 Red Hat OpenShift 클러스터(버전 4.5 이상)
2. 관리형 클러스터를 위한 Red Hat OpenShift 클러스터(버전 4.4.3 이상)
3. Red Hat OpenShift 클러스터에 대한 클러스터 관리자 액세스
4. Kubernetes를 위한 고급 클러스터 관리를 위한 Red Hat 구독

Advanced Cluster Management는 OpenShift 클러스터용 애드온이므로 허브와 관리형 클러스터에서 사용되는 기능에 따라 하드웨어 리소스에 대한 특정 요구 사항과 제한 사항이 있습니다. 클러스터 크기를 조정할 때 이러한 문제를 고려해야 합니다. 문서를 참조하세요 "여기" 자세한 내용은.

선택적으로, 허브 클러스터에 인프라 구성 요소를 호스팅하기 위한 전용 노드가 있고 해당 노드에만 고급 클러스터 관리 리소스를 설치하려는 경우, 해당 노드에 허용 및 선택기를 추가해야 합니다. 자세한 내용은 설명서를 참조하세요. "여기" .

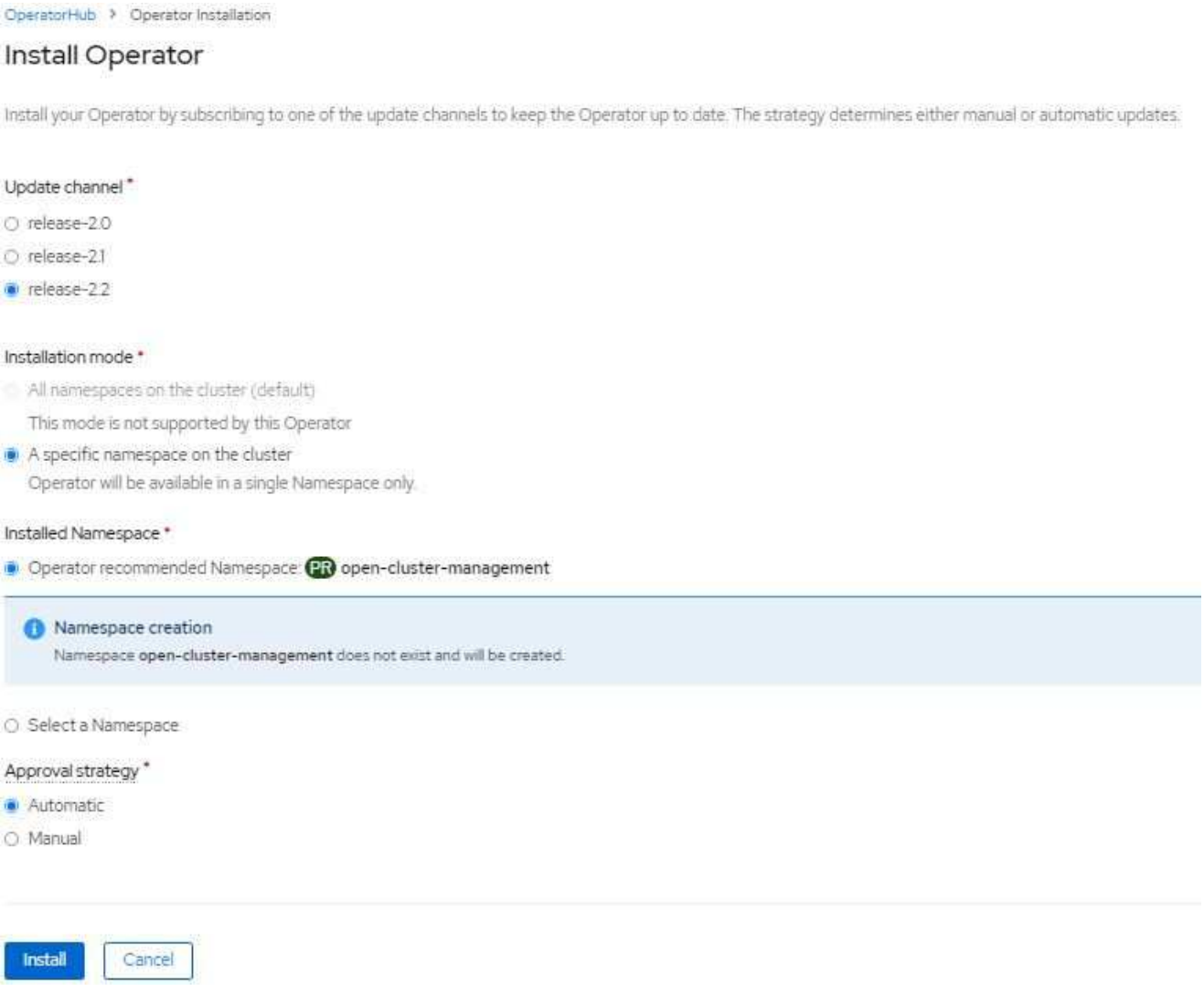
OpenShift 클러스터에 Kubernetes용 Advanced Cluster Management를 설치하려면 다음 단계를 완료하세요.

1. 허브 클러스터로 OpenShift 클러스터를 선택하고 클러스터 관리자 권한으로 로그인합니다.
2. 운영자 > 운영자 허브로 이동하여 Kubernetes용 고급 클러스터 관리를 검색합니다.

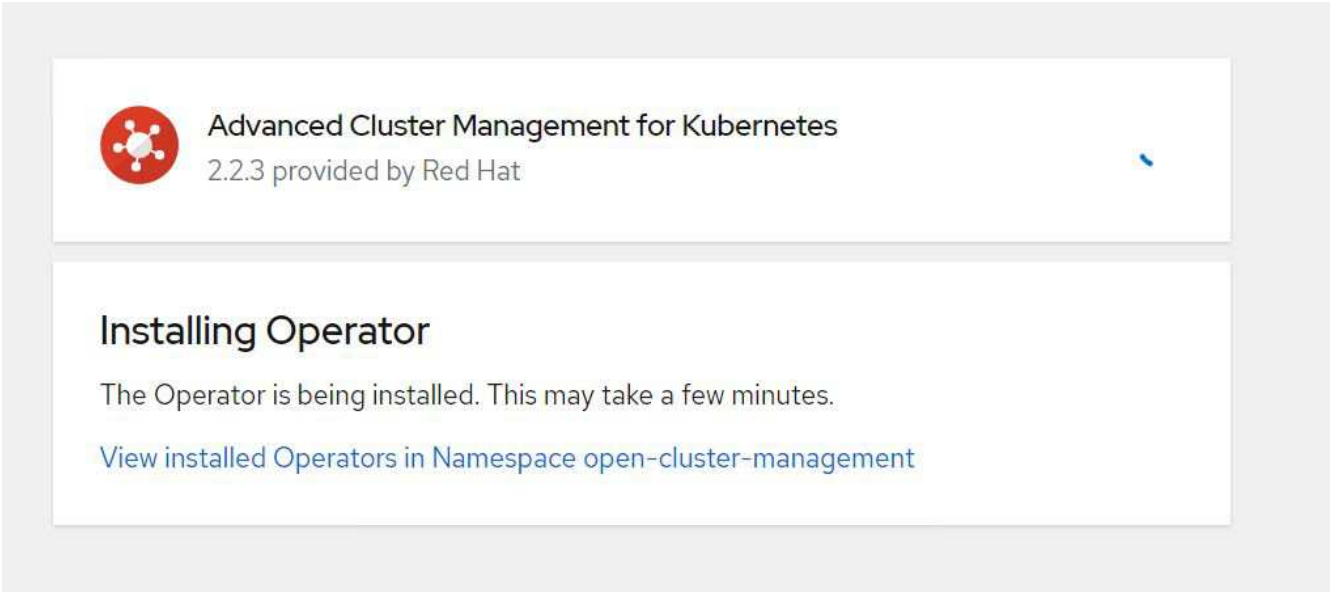


3. Kubernetes용 고급 클러스터 관리를 선택하고 설치를 클릭합니다.

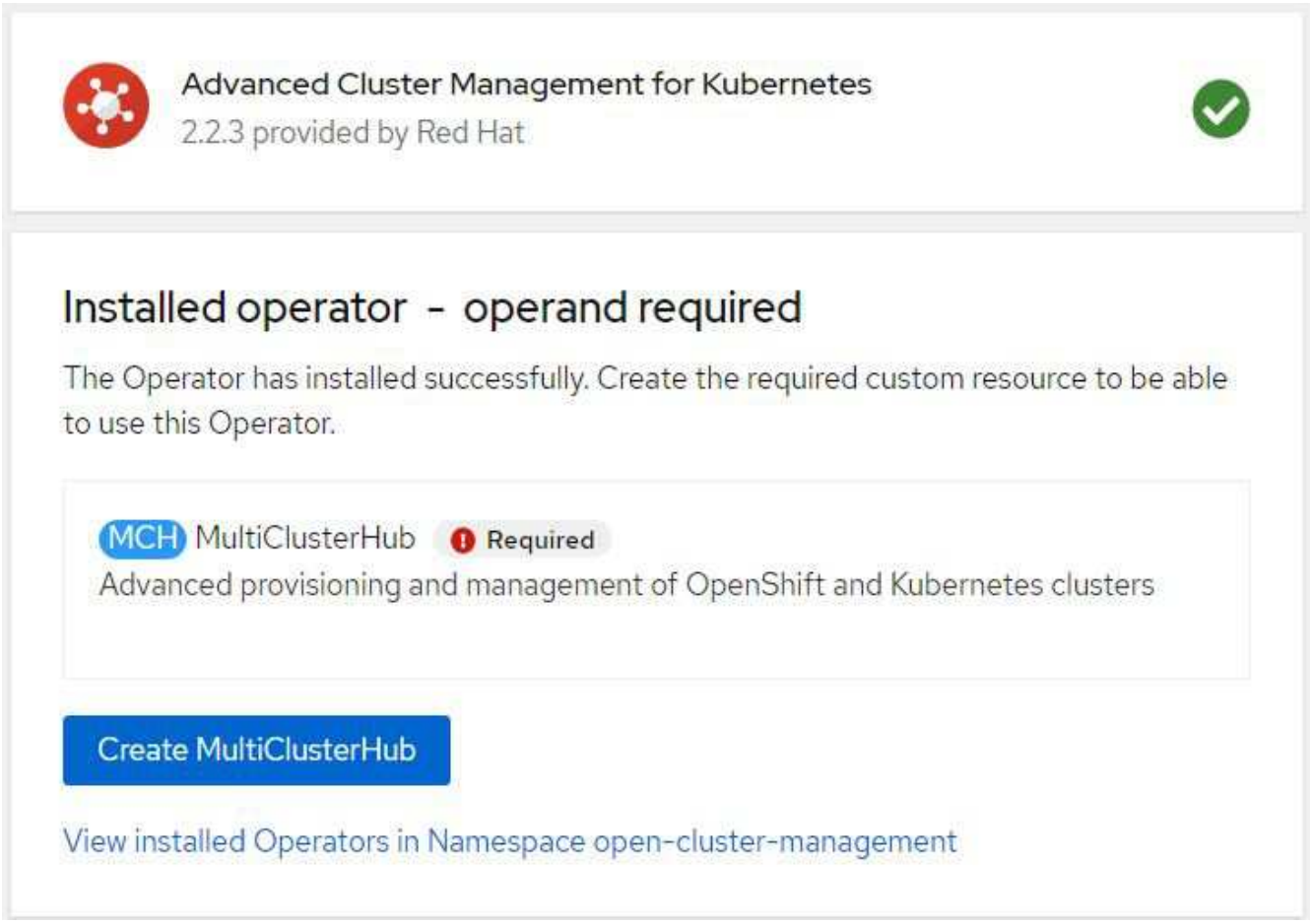
4. 설치 운영자 화면에서 필요한 세부 정보를 제공하고(NetApp 기본 매개변수를 유지할 것을 권장합니다) 설치를 클릭합니다.



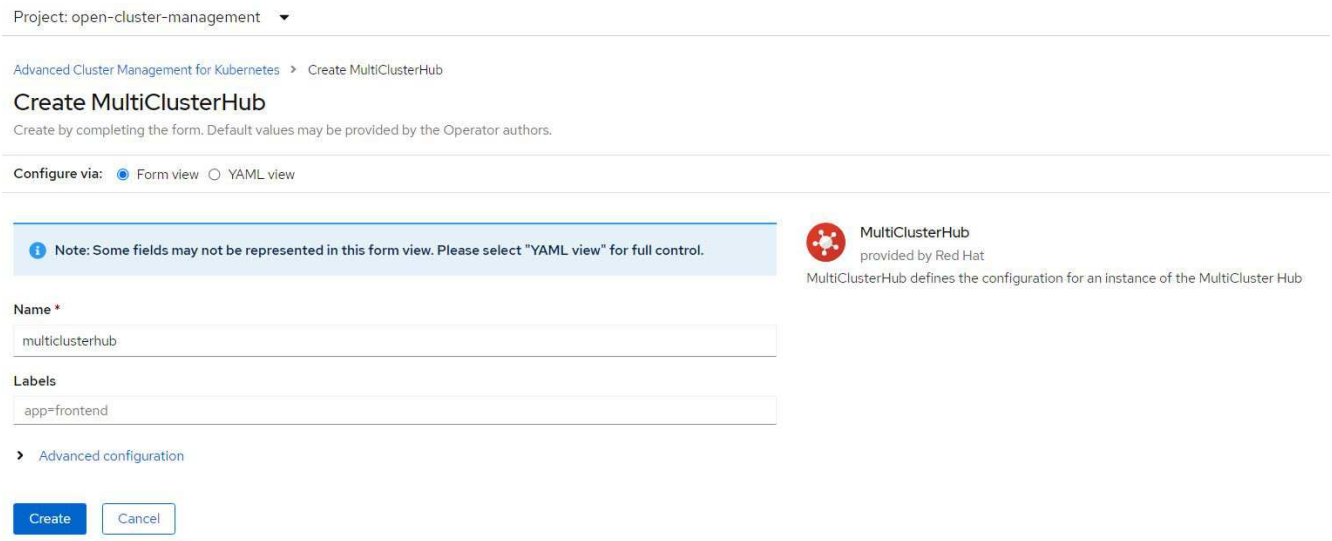
5. 운영자 설치가 완료될 때까지 기다리세요.



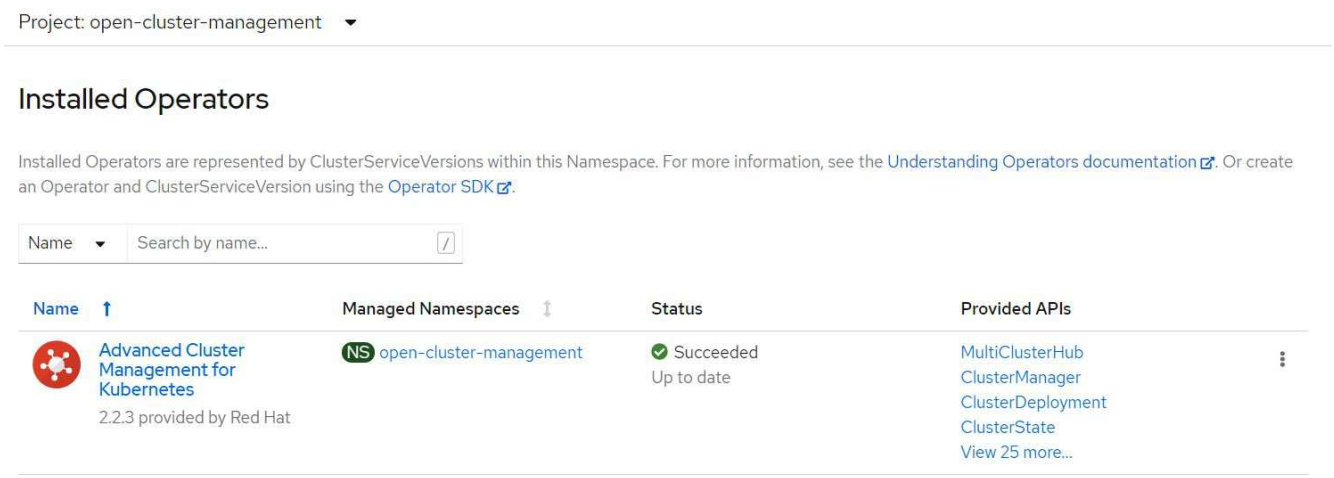
6. 운영자가 설치된 후 MultiClusterHub 만들기를 클릭합니다.



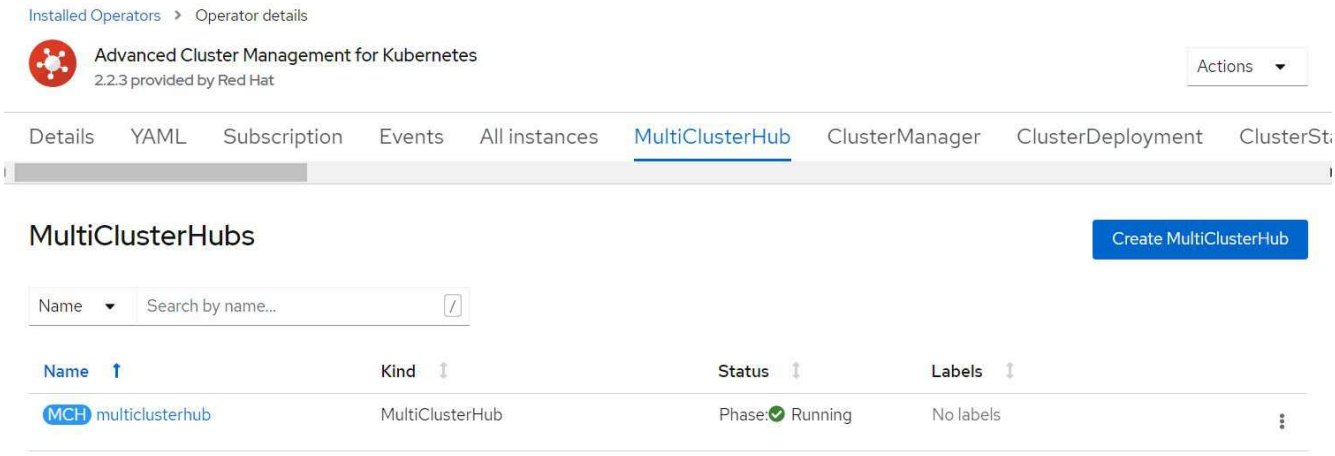
7. MultiClusterHub 생성 화면에서 세부 정보를 입력한 후 생성을 클릭합니다. 이렇게 하면 멀티 클러스터 허브 설치가 시작됩니다.



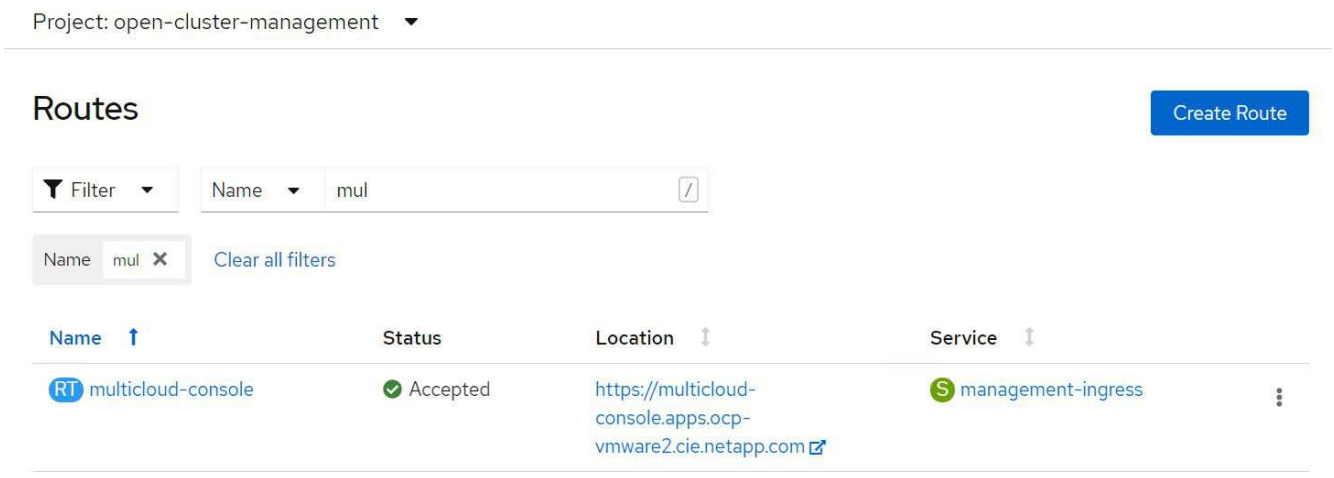
8. 모든 포드가 open-cluster-management 네임스페이스에서 Running 상태로 이동하고 운영자가 Succeeded 상태로 이동하면 Kubernetes용 Advanced Cluster Management가 설치됩니다.



9. 허브 설치를 완료하는 데는 시간이 다소 걸리며, 설치가 완료되면 MultiCluster 허브가 실행 상태로 전환됩니다.

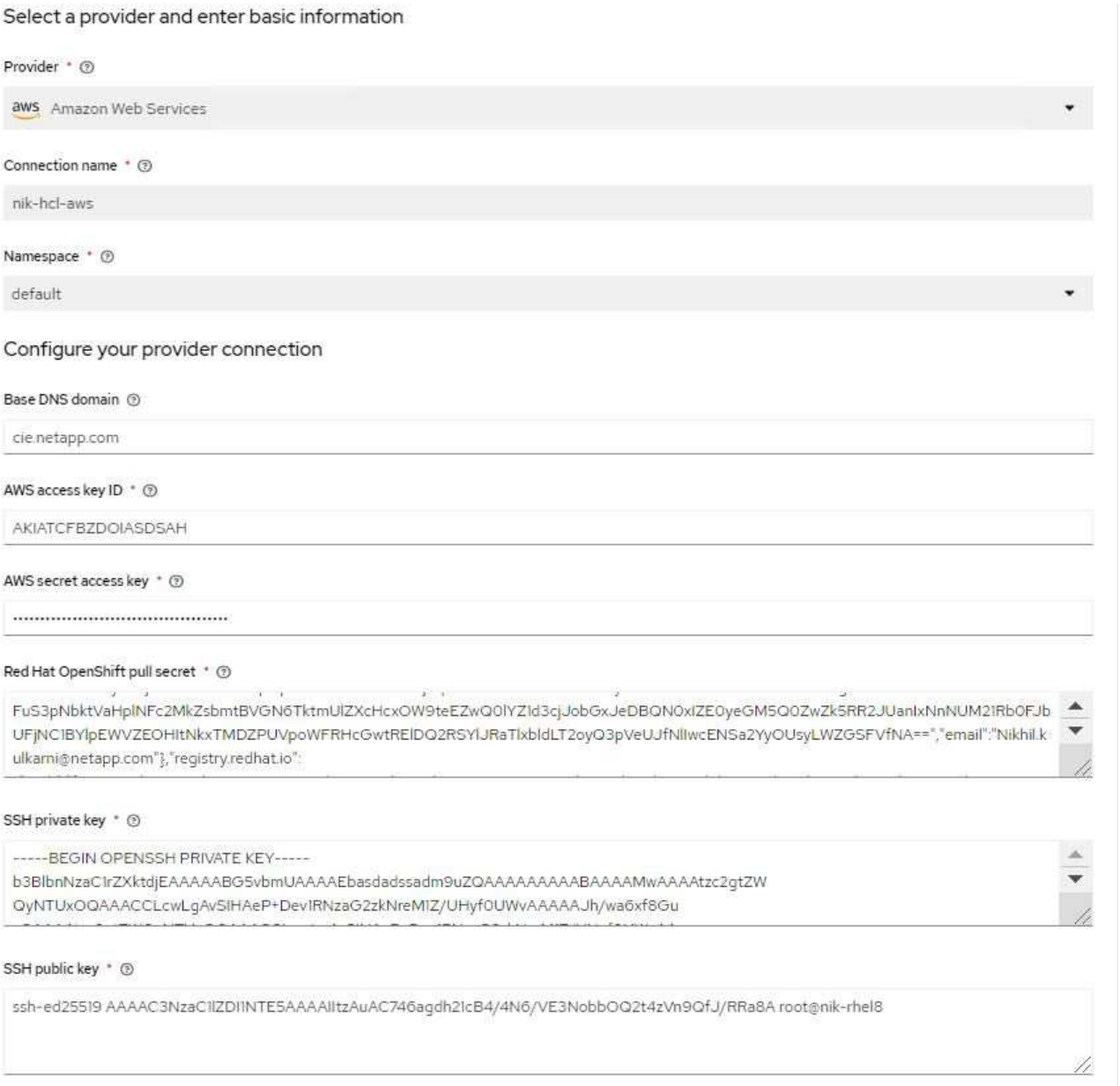


10. 이는 open-cluster-management 네임스페이스에 경로를 생성합니다. 경로의 URL에 연결하여 고급 클러스터 관리 콘솔에 액세스합니다.

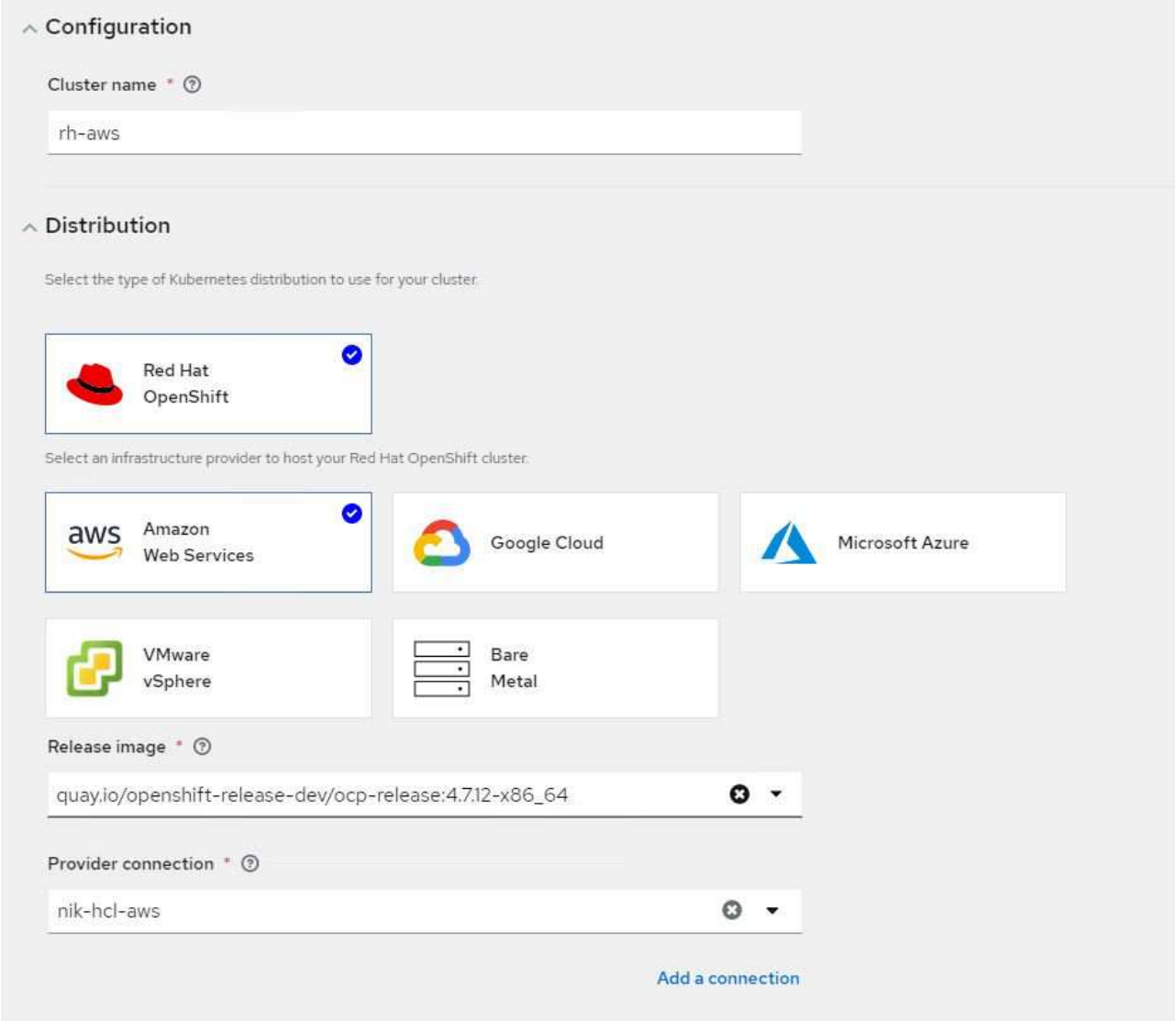


다양한 OpenShift 클러스터를 관리하려면 Advanced Cluster Management에서 클러스터를 만들거나 가져올 수 있습니다.

1. 먼저 Automate Infrastructures > Clusters로 이동합니다.
2. 새로운 OpenShift 클러스터를 생성하려면 다음 단계를 완료하세요.
   a. 공급자 연결 만들기: 공급자 연결로 이동하여 연결 추가를 클릭하고 선택한 공급자 유형에 해당하는 모든 세부 정보를 제공하고 추가를 클릭합니다.



   b. 새 클러스터를 만들려면 클러스터로 이동하여 클러스터 추가 > 클러스터 만들기를 클릭합니다. 클러스터와 해당 공급자에 대한 세부 정보를 제공하고 만들기를 클릭합니다.



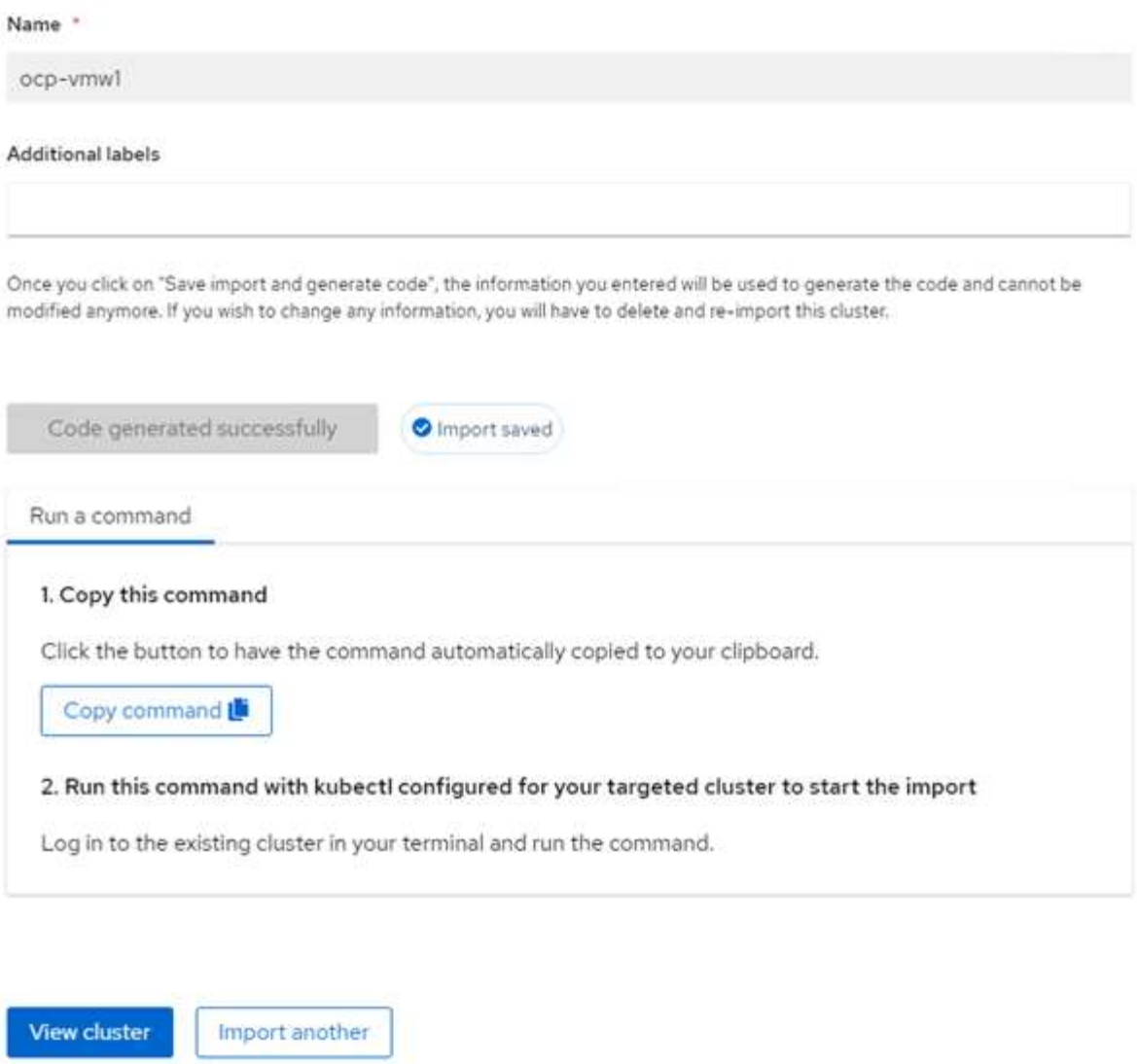
c. 클러스터가 생성되면 클러스터 목록에 준비 상태로 나타납니다.

3. 기존 클러스터를 가져오려면 다음 단계를 완료하세요.

a. 클러스터로 이동하여 클러스터 추가 > 기존 클러스터 가져오기를 클릭합니다.

b. 클러스터 이름을 입력하고 저장, 가져오기 및 코드 생성을 클릭합니다. 기존 클러스터를 추가하는 명령이 표시됩니다.

c. 명령 복사를 클릭하고 허브 클러스터에 추가할 클러스터에서 명령을 실행합니다. 이렇게 하면 클러스터에 필요한 에이전트가 설치되고, 이 프로세스가 완료되면 클러스터가 준비 상태로 클러스터 목록에 나타납니다.

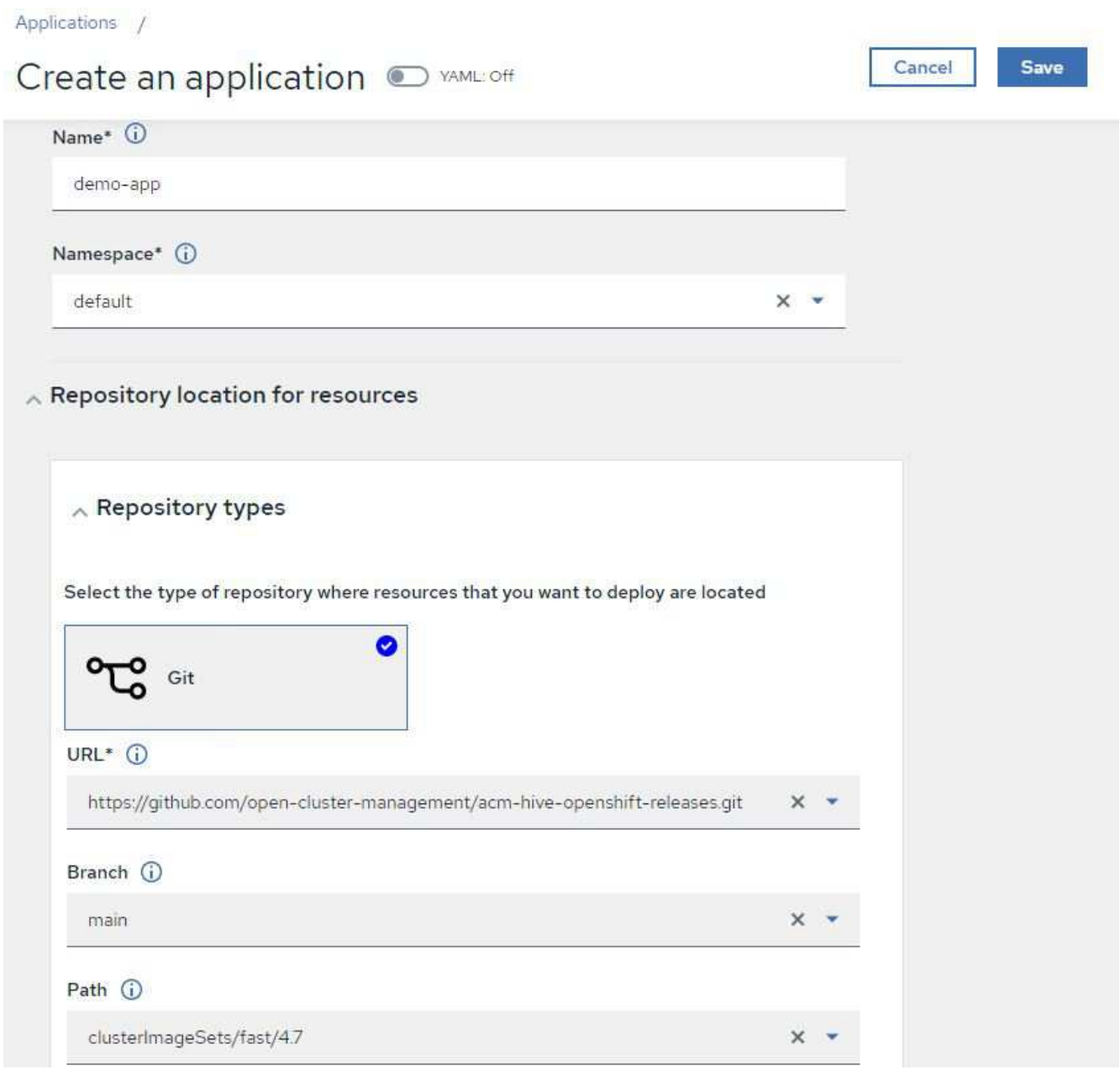


4. 여러 클러스터를 만들고 가져온 후에는 단일 콘솔에서 클러스터를 모니터링하고 관리할 수 있습니다.

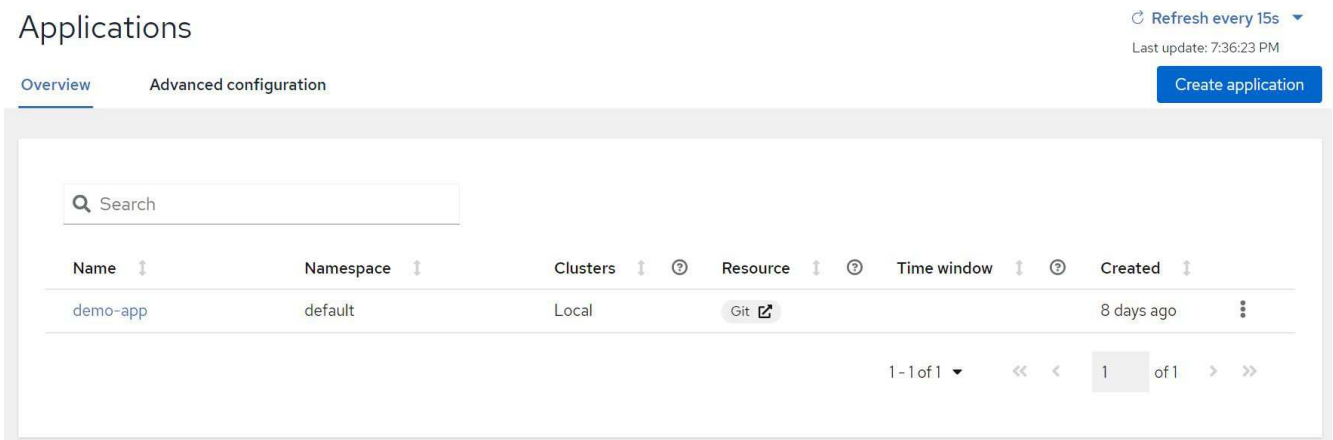
애플리케이션 수명 주기 관리

클러스터 집합에서 애플리케이션을 생성하고 관리하려면

1. 사이드바에서 애플리케이션 관리로 이동하여 애플리케이션 만들기를 클릭합니다. 만들고 싶은 애플리케이션의 세부 정보를 입력하고 '저장'을 클릭하세요.



2. 애플리케이션 구성 요소가 설치되면 해당 애플리케이션이 목록에 나타납니다.

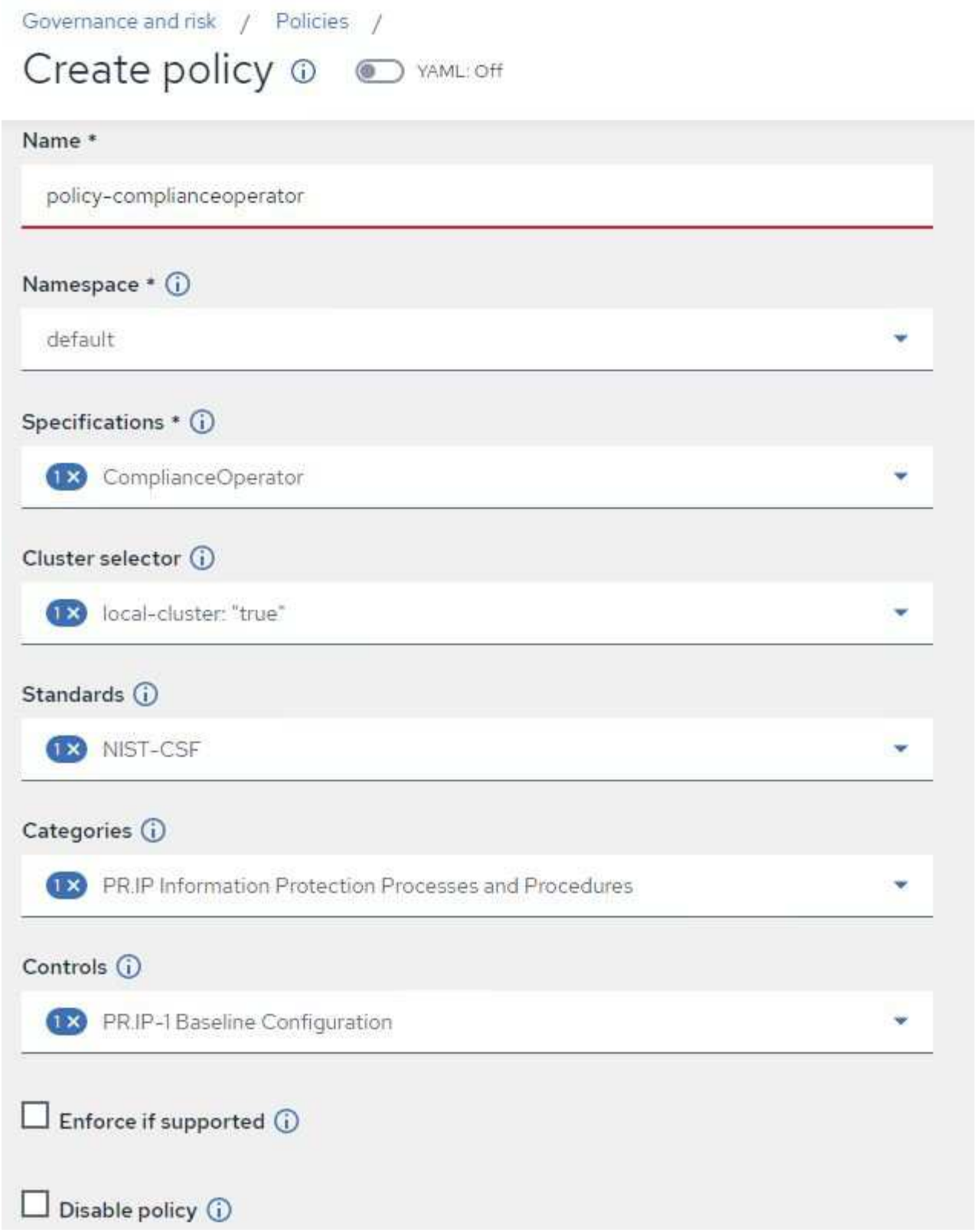


3. 이제 콘솔에서 애플리케이션을 모니터링하고 관리할 수 있습니다.

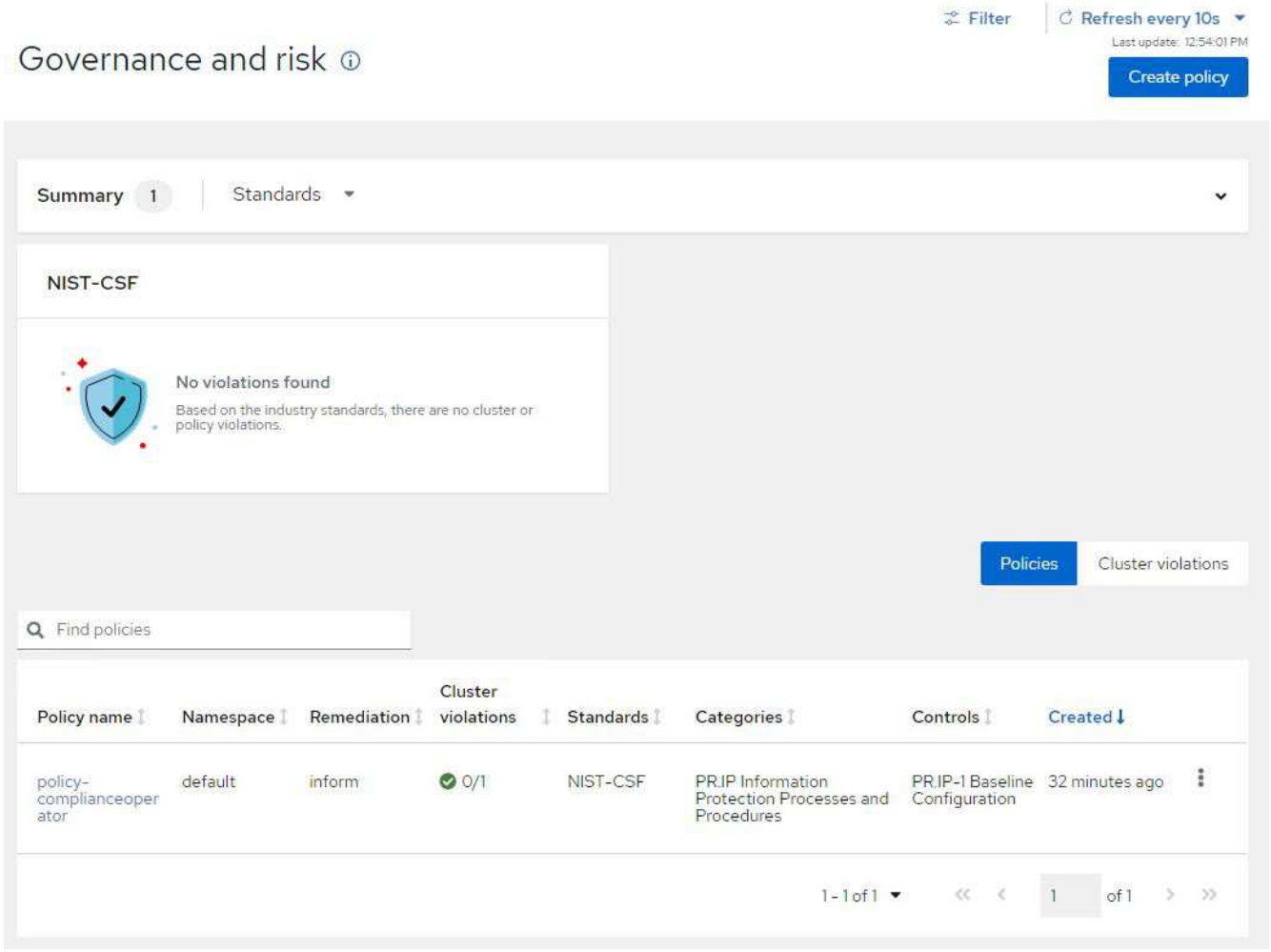
거버넌스와 위험

이 기능을 사용하면 다양한 클러스터에 대한 규정 준수 정책을 정의하고 클러스터가 해당 정책을 준수하는지 확인할 수 있습니다. 규칙을 위반하거나 벗어난 사항을 알리거나 수정하기 위해 정책을 구성할 수 있습니다.

1. 사이드바에서 거버넌스 및 위험으로 이동합니다.
2. 규정 준수 정책을 만들려면 정책 만들기를 클릭하고 정책 표준의 세부 정보를 입력한 다음 이 정책을 준수해야 하는 클러스터를 선택합니다. 이 정책 위반 사항을 자동으로 수정하려면 지원되는 경우 적용 확인란을 선택하고 만들기를 클릭합니다.



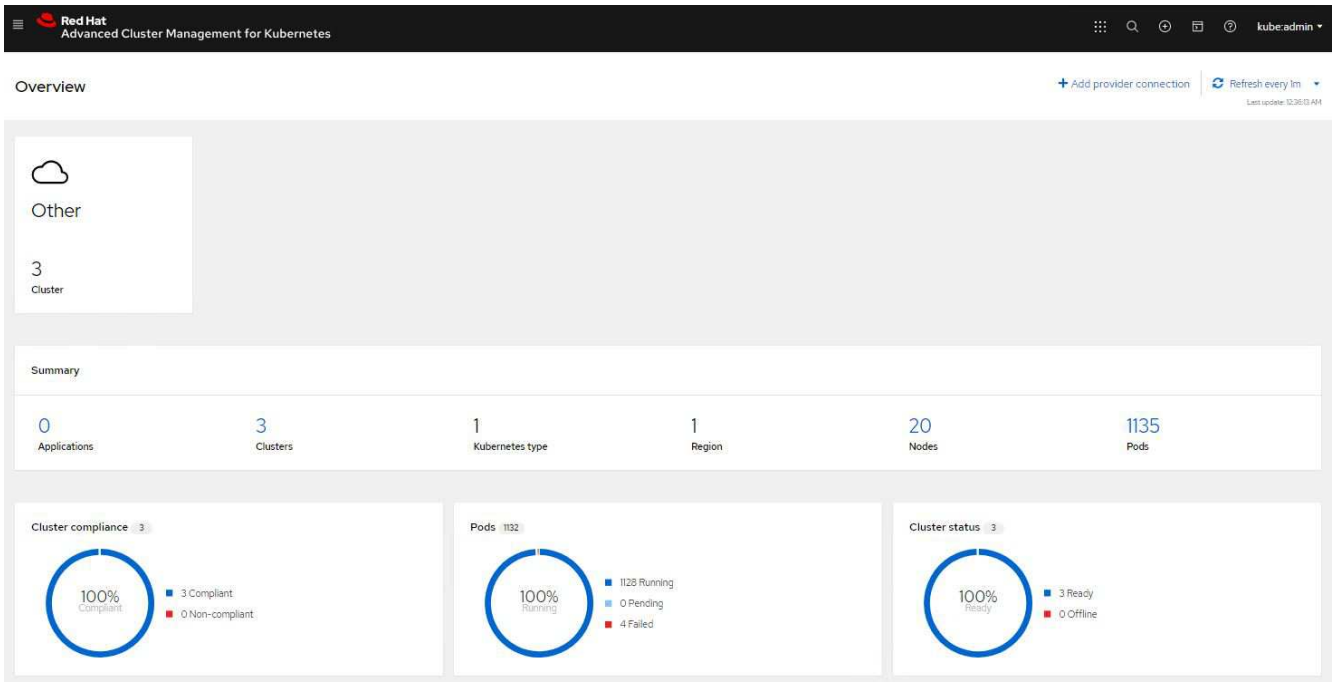
3. 모든 필수 정책이 구성된 후에는 Advanced Cluster Management에서 모든 정책 또는 클러스터 위반 사항을 모니터링하고 해결할 수 있습니다.



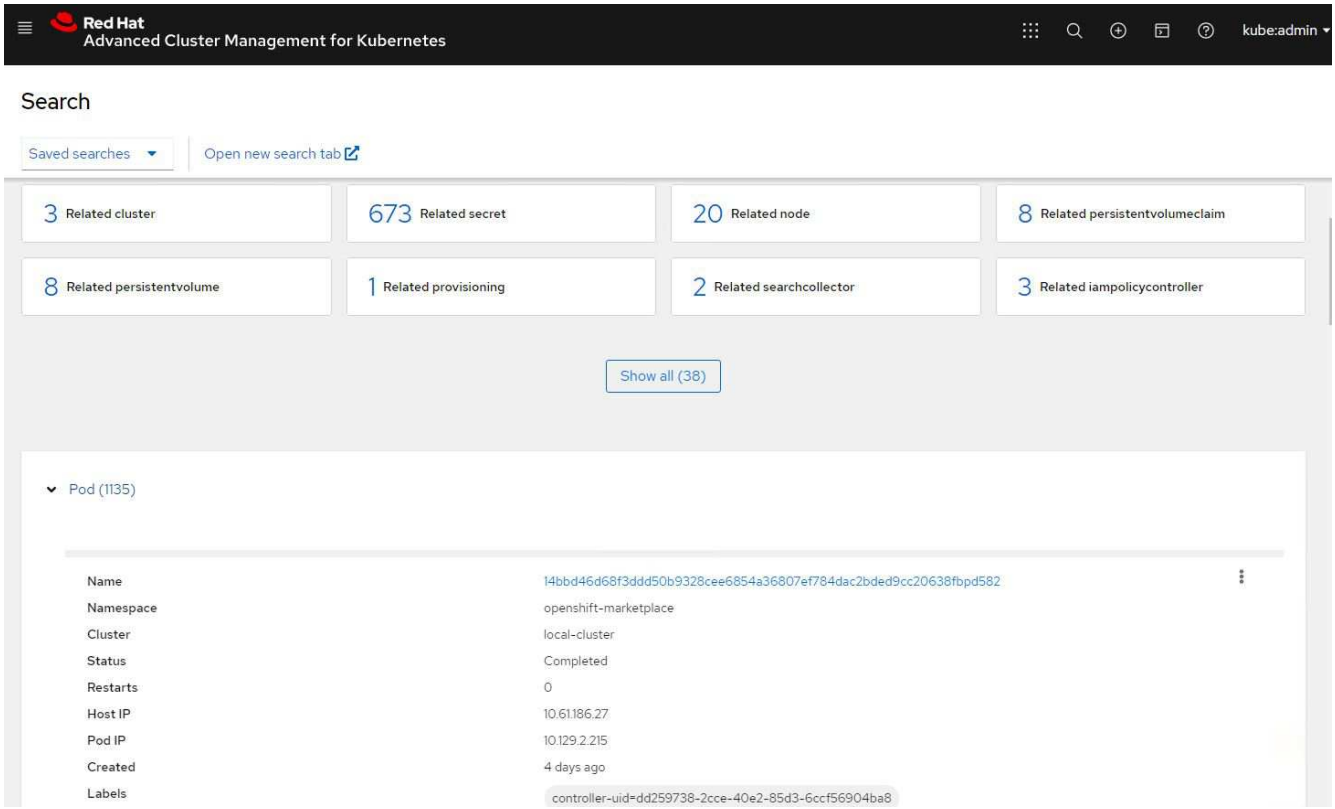
관찰 가능성

Kubernetes를 위한 고급 클러스터 관리 기능은 모든 클러스터에서 노드, 포드, 애플리케이션 및 워크로드를 모니터링하는 방법을 제공합니다.

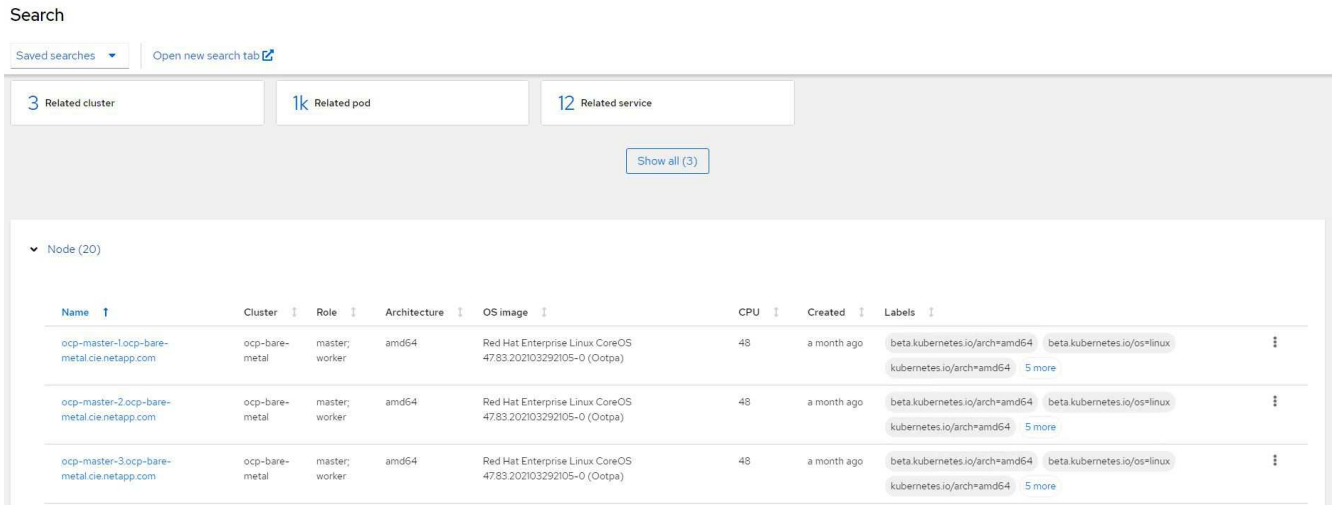
1. 환경 관찰 > 개요로 이동합니다.



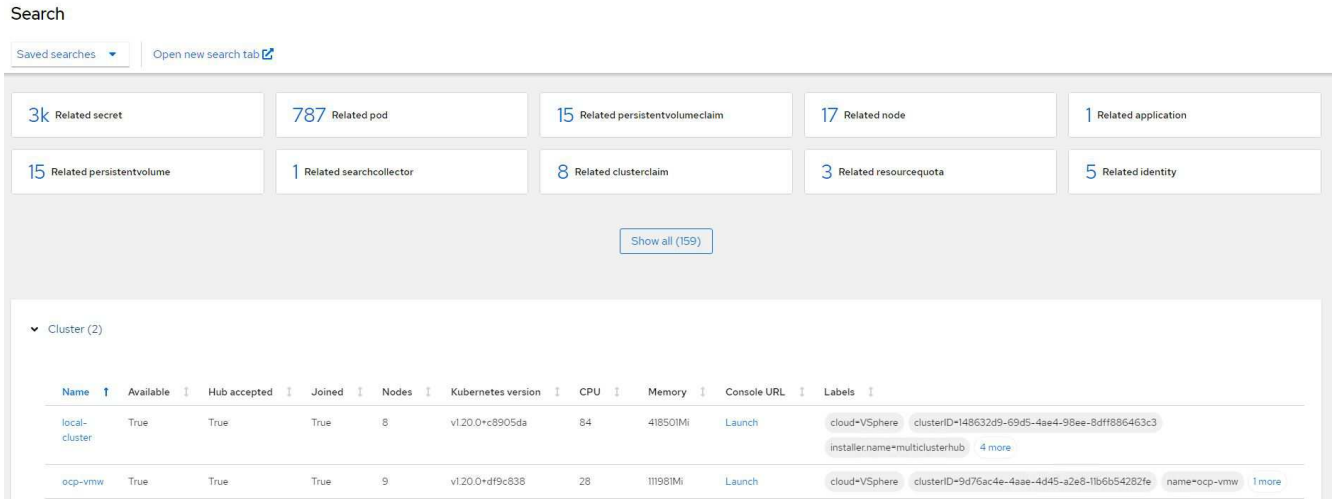
2. 모든 클러스터의 모든 포드와 워크로드는 다양한 필터를 기준으로 모니터링되고 정렬됩니다. 해당 데이터를 보려면 Pod를 클릭하세요.



3. 다양한 데이터 포인트를 기반으로 클러스터 전반의 모든 노드를 모니터링하고 분석합니다. 노드를 클릭하면 해당 세부 정보에 대한 자세한 정보를 얻을 수 있습니다.

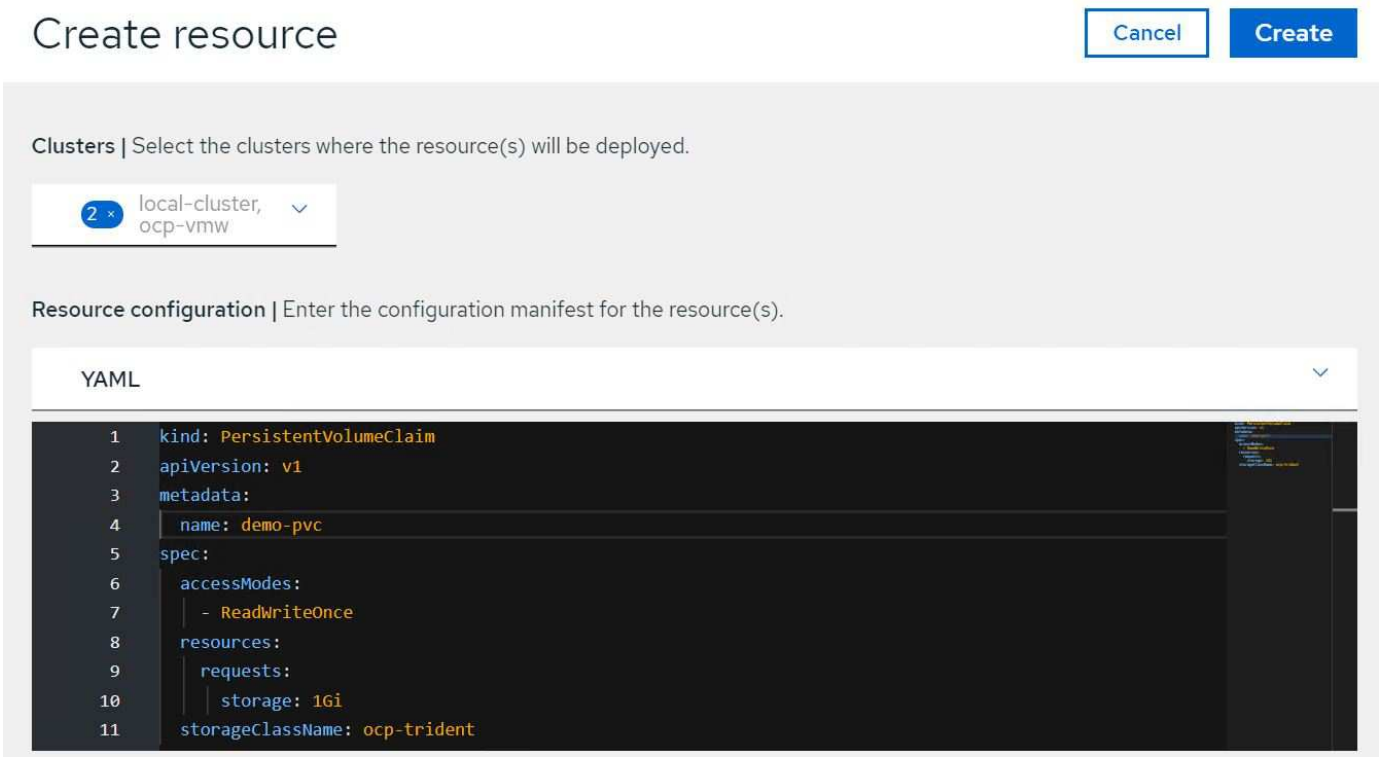


4. 모든 클러스터는 다양한 클러스터 리소스와 매개변수를 기반으로 모니터링되고 구성됩니다. 클러스터 세부 정보를 보려면 클러스터를 클릭하세요.



여러 클러스터에 리소스 생성

Kubernetes용 고급 클러스터 관리를 사용하면 사용자가 콘솔에서 하나 이상의 관리형 클러스터에 동시에 리소스를 생성할 수 있습니다. 예를 들어, 서로 다른 NetApp ONTAP 클러스터로 지원되는 여러 사이트에 OpenShift 클러스터가 있고 두 사이트 모두에 PVC를 프로비저닝하려면 상단 표시줄에 있는 (+) 기호를 클릭하면 됩니다. 그런 다음 PVC를 만들려는 클러스터를 선택하고 리소스 YAML을 붙여넣은 다음 만들기를 클릭합니다.



Trident Protect를 사용한 컨테이너 앱 및 VM에 대한 데이터 보호

이 솔루션은 Trident Protect를 사용하여 컨테이너 및 VM에 대한 데이터 보호 작업을 수행하는 방법을 보여줍니다.

1. OpenShift Container 플랫폼에서 컨테이너 애플리케이션에 대한 스냅샷 및 백업을 생성하고 이를 복원하는 방법에 대한 자세한 내용은 다음을 참조하세요."여기" .
2. OpenShift Container 플랫폼에 배포된 OpenShift Virtualization의 VM에 대한 백업을 생성하고 복원하는 방법에 대한 자세한 내용은 다음을 참조하세요."여기" .

타사 도구를 사용하여 컨테이너 앱 및 VM에 대한 데이터 보호

이 솔루션은 Red Hat OpenShift Container 플랫폼의 OADP 운영자와 통합된 Velero를 사용하여 컨테이너 및 VM에 대한 데이터 보호 작업을 수행하는 방법을 보여줍니다.

1. OpenShift Container 플랫폼에서 컨테이너 애플리케이션의 백업을 생성하고 복원하는 방법에 대한 자세한 내용은 다음을 참조하세요."여기" .
2. OpenShift Container 플랫폼에 배포된 OpenShift Virtualization의 VM에 대한 백업을 생성하고 복원하는 방법에 대한 자세한 내용은 다음을 참조하세요."여기" .

NetApp 스토리지와 Red Hat OpenShift Virtualization 통합에 대해 알아볼 수 있는 추가 리소스

다양한 플랫폼과 기술에서 ONTAP 사용하여 Red Hat OpenShift Virtualization을 배포, 관리 및 최적화하는 데 대한 자세한 정보를 제공하는 추가 리소스에 액세스하세요.

- NetApp 문서

  "https://docs.netapp.com/"

- Trident 문서

  "https://docs.netapp.com/us-en/trident/index.html"

- Red Hat OpenShift 문서

  "https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/"

- Red Hat OpenStack 플랫폼 문서

  "https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/"

- Red Hat Virtualization 문서

  "https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.4/"

- VMware vSphere 설명서

  "https://docs.vmware.com/"