OpenShift Dedicated 4

아키텍처

아키텍처 개요.

Last Updated: 2024-11-02

OpenShift Dedicated 4 아키텍처

아키텍처 개요.

법적 공지

초록

OpenShift Dedicated는 클라우드 기반 Kubernetes 컨테이너 플랫폼입니다. OpenShift Dedicated의 기반은 Kubernetes를 기반으로 하므로 동일한 기술을 공유합니다. OpenShift Dedicated 및 Kubernetes에 대한 자세한 내용은 제품 아키텍처를 참조하십시오.

차례

1장. 아키텍처 개요

OpenShift Dedicated는 클라우드 기반 Kubernetes 컨테이너 플랫폼입니다. OpenShift Dedicated의 기반은 Kubernetes를 기반으로 하므로 동일한 기술을 공유합니다. OpenShift Dedicated 및 Kubernetes에 대한 자세한 내용은 제품 아키텍처를 참조하십시오.

1.1. OPENSHIFT DEDICATED 아키텍처의 일반 용어집

이 용어집은 아키텍처 콘텐츠에 사용되는 일반적인 용어를 정의합니다.

액세스 정책

클러스터 내의 사용자, 애플리케이션 및 엔터티가 서로 상호 작용하는 방식을 지정하는 일련의 역할입니다. 액세스 정책은 클러스터 보안을 강화합니다.

승인 플러그인

승인 플러그인은 보안 정책, 리소스 제한 또는 구성 요구 사항을 적용합니다.

인증

OpenShift Dedicated 클러스터에 대한 액세스를 제어하기 위해 **dedicated-admin** 역할의 관리자는 승인된 사용자만 클러스터에 액세스할 수 있도록 사용자 인증을 구성할 수 있습니다. OpenShift Dedicated 클러스터와 상호 작용하려면 OpenShift Dedicated API로 인증해야 합니다. OpenShift Dedicated API에 대한 요청에 OAuth 액세스 토큰 또는 X.509 클라이언트 인증서를 제공하여 인증할 수 있습니다.

부트스트랩

최소 Kubernetes를 실행하고 OpenShift Dedicated 컨트롤 플레인을 배포하는 임시 머신입니다.

CSR(인증서 서명 요청)

리소스는 표시된 서명자가 인증서에 서명하도록 요청합니다. 이 요청은 승인되거나 거부될 수 있습니다.

Cluster Version Operator (CVO)

OpenShift Dedicated Update Service를 사용하여 현재 구성 요소 버전 및 그래프의 정보를 기반으로 유효한 업데이트 및 업데이트 경로를 확인하는 Operator입니다.

컴퓨팅 노드

클러스터 사용자에 대한 워크로드를 실행하는 노드입니다. 컴퓨팅 노드는 작업자 노드라고도 합니다.

구성 드리프트

노드의 구성이 머신 구성에서 지정하는 것과 일치하지 않는 경우입니다.

컨테이너

소프트웨어와 모든 종속 항목으로 구성된 경량 및 실행 가능한 이미지입니다. 컨테이너는 운영 체제를 가상화하므로 데이터 센터, 퍼블릭 또는 프라이빗 클라우드, 로컬 호스트와 같은 컨테이너를 어디에서나 실행할 수 있습니다.

컨테이너 오케스트레이션 엔진

컨테이너의 배포, 관리, 확장 및 네트워킹을 자동화하는 소프트웨어입니다.

컨테이너 워크로드

컨테이너에 패키지 및 배포되는 애플리케이션입니다.

컨트롤그룹(cgroups)

프로세스의 소비를 관리하고 제한하기 위해 프로세스의 파티션 세트를 그룹으로 분할합니다.

컨트롤 플레인

컨테이너의 라이프사이클을 정의, 배포, 관리하기 위해 API 및 인터페이스를 노출하는 컨테이너 오케스트레이션 계층입니다. 컨트롤 플레인은 컨트롤 플레인 시스템이라고도 합니다.

CRI-O

운영 체제와 통합되는 Kubernetes 네이티브 컨테이너 런타임 구현으로 효율적인 Kubernetes 환경을 제공합니다.

배포

애플리케이션의 라이프사이클을 유지 관리하는 Kubernetes 리소스 오브젝트입니다.

Dockerfile

터미널에서 이미지를 어셈블하기 위해 수행할 사용자 명령이 포함된 텍스트 파일입니다.

하이브리드 클라우드 배포

베어 메탈, 가상, 프라이빗 및 퍼블릭 클라우드 환경 전체에 일관된 플랫폼을 제공하는 배포. 이를 통해 속도, 민첩성 및 이식성을 제공합니다.

Ignition

초기 구성 중에 RHCOS가 디스크를 조작하는 데 사용하는 유틸리티입니다. 디스크 파티셔닝, 파티션 포맷, 파일 작성 및 사용자 구성을 포함한 일반적인 디스크 작업을 완료합니다.

설치 프로그램에서 제공하는 인프라

설치 프로그램은 클러스터가 실행되는 인프라를 배포하고 구성합니다.

kubelet

Pod에서 컨테이너가 실행 중인지 확인하기 위해 클러스터의 각 노드에서 실행되는 기본 노드 에이전트입니다.

Kubernetes 매니페스트

JSON 또는 YAML 형식의 Kubernetes API 오브젝트의 사양입니다. 구성 파일에는 배포, 구성 맵, 시크릿, 데몬 세트가 포함될 수 있습니다.

MCD(Machine Config Daemon)

노드에서 구성 드리프트가 있는지 정기적으로 확인하는 데몬입니다.

Machine Config Operator (MCO)

클러스터 머신에 새 구성을 적용하는 Operator입니다.

머신 구성 풀(MCP)

컨트롤 플레인 구성 요소 또는 사용자 워크로드와 같은 머신 그룹은 처리하는 리소스를 기반으로 합니다.

메타데이터

클러스터 배포 아티팩트에 대한 추가 정보입니다.

마이크로 서비스

소프트웨어 작성을 위한 접근 방식. 애플리케이션은 마이크로 서비스를 사용하여 서로 독립적으로 가장 작은 구성 요소로 분리할 수 있습니다.

미러 레지스트리

OpenShift Dedicated 이미지의 미러가 있는 레지스트리입니다.

모놀리식 애플리케이션

자체 포함, 빌드 및 패키지된 애플리케이션입니다.

네임스페이스

네임스페이스는 모든 프로세스에 표시되는 특정 시스템 리소스를 격리합니다. 네임스페이스 내에서 해당 네임스페이스의 멤버인 프로세스만 해당 리소스를 볼 수 있습니다.

네트워킹

OpenShift Dedicated 클러스터의 네트워크 정보

노드

OpenShift Dedicated 클러스터의 작업자 시스템입니다. 노드는 VM(가상 머신) 또는 물리적 머신입니다.

OpenShift CLI(oc)

터미널에서 OpenShift Dedicated 명령을 실행하는 명령줄 툴입니다.

OSUS(OpenShift Update Service)

인터넷에 액세스할 수 있는 클러스터의 경우 RHEL(Red Hat Enterprise Linux)은 공용 API 뒤에 있는 호스팅 서비스로 OpenShift 업데이트 서비스를 사용하여 무선 업데이트를 제공합니다.

OpenShift 이미지 레지스트리

이미지를 관리하기 위해 OpenShift Dedicated에서 제공하는 레지스트리입니다.

Operator

OpenShift Dedicated 클러스터에서 Kubernetes 애플리케이션을 패키징, 배포 및 관리하는 기본 방법입니다. Operator는 사람의 운영 지식을 패키지하고 고객과 공유하는 소프트웨어로 인코딩합니다.

OperatorHub

설치할 다양한 OpenShift Dedicated Operator가 포함된 플랫폼입니다.

OLM(Operator Lifecycle Manager)

OLM은 Kubernetes 네이티브 애플리케이션의 라이프사이클을 설치, 업데이트 및 관리할 수 있도록 지원합니다. OLM은 효과적이고 자동화되고 확장 가능한 방식으로 Operator를 관리하도록 설계된 오픈 소스 툴킷입니다.

ostree

전체 파일 시스템 트리 트리의 원자 업그레이드를 수행하는 Linux 기반 운영 체제의 업그레이드 시스템입니다. ostree는 주소 지정 가능 오브젝트 저장소를 사용하여 파일 시스템 트리에 대한 의미 있는 변경을 추적하고 기존 패키지 관리 시스템을 보완하도록 설계되었습니다.

OTA(Over-the-Air) 업데이트

OSUS(OpenShift Dedicated Update Service)는 RHCOS(Red Hat Enterprise Linux CoreOS)를 포함하여 OpenShift Dedicated에 무선 업데이트를 제공합니다.

Pod

OpenShift Dedicated 클러스터에서 실행되는 볼륨 및 IP 주소와 같은 공유 리소스가 있는 하나 이상의 컨테이너입니다. Pod는 정의, 배포 및 관리되는 최소 컴퓨팅 단위입니다.

프라이빗 레지스트리

OpenShift Dedicated는 컨테이너 이미지 레지스트리 API를 구현하는 모든 서버를 이미지 소스로 사용하여 개발자가 개인 컨테이너 이미지를 푸시하고 가져올 수 있습니다.

퍼블릭 레지스트리

OpenShift Dedicated는 컨테이너 이미지 레지스트리 API를 구현하는 모든 서버를 이미지 소스로 사용하여 개발자가 공개 컨테이너 이미지를 푸시하고 가져올 수 있습니다.

RHEL OpenShift Dedicated Cluster Manager

OpenShift Dedicated 클러스터를 설치, 수정, 운영 및 업그레이드할 수 있는 관리형 서비스입니다.

RHEL Quay 컨테이너 레지스트리

OpenShift Dedicated 클러스터에 대부분의 컨테이너 이미지 및 Operator를 제공하는 Quay.io 컨테이너 레지스트리입니다.

복제 컨트롤러

한 번에 실행하는 데 필요한 Pod 복제본 수를 나타내는 자산입니다.

RBAC(역할 기반 액세스 제어)

클러스터 사용자와 워크로드가 역할을 실행하는 데 필요한 리소스에만 액세스할 수 있도록 하는 주요 보안 제어입니다.

라우트

routes는 OpenShift Dedicated 인스턴스 외부의 사용자 및 애플리케이션에서 포드에 대한 네트워크 액세스를 허용하는 서비스를 노출합니다.

스케일링

리소스 용량이 증가하거나 감소합니다.

서비스

서비스는 Pod 집합에서 실행 중인 애플리케이션을 노출합니다.

S2I(Source-to-Image) 이미지

애플리케이션을 배포하기 위해 OpenShift Dedicated에서 애플리케이션 소스 코드의 프로그래밍 언어를 기반으로 생성된 이미지입니다.

storage

OpenShift Dedicated는 클라우드 공급자를 위한 다양한 유형의 스토리지를 지원합니다. OpenShift Dedicated 클러스터에서 영구 및 비영구 데이터에 대한 컨테이너 스토리지를 관리할 수 있습니다.

telemetry

OpenShift Dedicated의 크기, 상태 및 상태와 같은 정보를 수집하는 구성 요소입니다.

템플릿

템플릿은 OpenShift Dedicated에서 생성을 위한 오브젝트 목록을 생성하기 위해 매개 변수화 및 처리될 수 있는 오브젝트 세트를 설명합니다.

웹 콘솔

OpenShift Dedicated를 관리할 UI(사용자 인터페이스)입니다.

작업자 노드

클러스터 사용자에 대한 워크로드를 실행하는 노드입니다. 작업자 노드는 컴퓨팅 노드라고도 합니다.

추가 리소스

- 스토리지에 대한 자세한 내용은 OpenShift Dedicated 스토리지를 참조하십시오.
- 인증에 대한 자세한 내용은 OpenShift Dedicated 인증을 참조하십시오.
- OLM (Operator Lifecycle Manager)에 대한 자세한 내용은 OLM을 참조하십시오.

1.2. OPENSHIFT DEDICATED가 OPENSHIFT CONTAINER PLATFORM과 다른 방법 이해

OpenShift Dedicated는 OpenShift Container Platform과 동일한 코드 기반을 사용하지만, 성능, 확장성 및 보안에 맞게 최적화된 방식으로 설치됩니다. OpenShift Dedicated는 완전히 관리되는 서비스이므로 OpenShift Container Platform에서 수동으로 설정한 대부분의 OpenShift Dedicated 구성 요소 및 설정이 기본적으로 설정됩니다.

사용자 인프라에서 OpenShift Dedicated와 표준 OpenShift Container Platform 설치의 다음 차이점을 검토하십시오.

OpenShift Container Platform	OpenShift Dedicated
고객은 OpenShift Container Platform을 설치하고 구성합니다.	OpenShift Dedicated는 Red Hat OpenShift Cluster Manager를 통해 설치되며 성능, 확장성 및 보안에 최적화된 표준화된 방식으로 설치됩니다.

OpenShift Container Platform	OpenShift Dedicated
고객은 컴퓨팅 리소스를 선택할 수 있습니다.	OpenShift Dedicated는 Red Hat이 소유하거나 고객이 제공하는 퍼블릭 클라우드(Amazon Web Services 또는 Google Cloud Platform)에서 호스팅 및 관리됩니다.
고객은 인프라에 대한 최상위 수준의 관리 권한이 있습니다.	고객은 고객이 클라우드 계정을 제공할 때 최상위 관리 액세스 권한을 사용할 수 있지만 고객은 기본 관리자 그룹(**dedicated-admin**)을 사용할 수 있습니다.
고객은 OpenShift Container Platform에서 지원되는 모든 기능 및 구성 설정을 사용할 수 있습니다.	일부 OpenShift Container Platform 기능 및 구성 설정은 OpenShift Dedicated에서 사용할 수 없거나 변경되지 않을 수 있습니다.
컨트롤 역할을 가져오는 시스템에서 API 서버 및 etcd와 같은 **컨트롤** 플레인 구성 요소를 설정합니다. 컨트롤 플레인 구성 요소를 수정할 수 있지만 컨트롤 플레인 데이터를 고가용성으로 백업, 복원 및 만들어야 합니다.	Red Hat은 컨트롤 플레인을 설정하고 컨트롤 플레인 구성 요소를 관리합니다. 컨트롤 플레인은 고가용성입니다.
컨트롤 플레인 및 작업자 노드의 기본 인프라를 업데이트해야 합니다. OpenShift 웹 콘솔을 사용하여 OpenShift Container Platform 버전을 업데이트할 수 있습니다.	Red Hat은 업데이트가 제공될 때 고객에게 자동으로 알립니다. OpenShift Cluster Manager에서 수동으로 또는 자동으로 업데이트를 예약할 수 있습니다.
지원은 Red Hat 서브스크립션 또는 클라우드 공급자의 약관에 따라 제공됩니다.	99.95%의 가동 시간 SLA 및 24x7 서비스를 통해 Red Hat에서 설계, 운영 및 지원합니다. 자세한 내용은 Red Hat Enterprise Agreement 부록 4(Online Subscription Services) 를 참조하십시오.

1.3. 컨트롤 플레인 정보

컨트롤 플레인은 클러스터의 작업자 노드 및 Pod를 관리합니다. MCP(Machine config pool)를 사용하여 노드를 구성할 수 있습니다. MCPS는 컨트롤 플레인 구성 요소 또는 사용자 워크로드와 같은 머신 그룹으로, 처리하는 리소스를 기반으로 합니다. OpenShift Dedicated는 호스트에 다양한 역할을 할당합니다. 이러한 역할은 클러스터에서 머신의 기능을 정의합니다. 클러스터에는 표준 컨트롤 플레인 및 작업자 역할 유형에 대한 정의가 포함되어 있습니다.

Operator를 사용하여 컨트롤 플레인에서 서비스를 패키지, 배포, 관리할 수 있습니다. Operator는 다음 서비스를 제공하므로 OpenShift Dedicated에서 중요한 구성 요소입니다.

- 상태 점검 수행
- 애플리케이션 조사 방법 제공
- 무선(Over-the-Air) 업데이트 관리
- 애플리케이션이 지정된 상태로 유지됨

1.4. 개발자를 위한 컨테이너화된 애플리케이션 정보

개발자는 다양한 툴, 방법 및 형식을 사용하여 고유한 요구 사항에 따라 컨테이너화된 애플리케이션을 개발할 수 있습니다.

- 간단한 컨테이너 애플리케이션을 빌드하려면 다양한 빌드 툴, 기본 이미지 및 레지스트리 옵션을 사용합니다.
- OperatorHub 및 템플릿과 같은 지원 구성 요소를 사용하여 애플리케이션을 개발합니다.
- 애플리케이션을 Operator로 패키징하고 배포합니다.

Kubernetes 매니페스트를 생성하여 Git 리포지토리에 저장할 수도 있습니다. Kubernetes는 pod라는 기본 단위에서 작동합니다. Pod는 클러스터에서 실행 중인 프로세스의 단일 인스턴스입니다. Pod에는 하나 이상의 컨테이너가 포함될 수 있습니다. Pod 세트와 액세스 정책을 그룹화하여 서비스를 생성할 수 있습니다. 서비스는 다른 애플리케이션이 Pod를 생성하고 삭제할 때 사용할 영구 내부 IP 주소 및 호스트 이름을 제공합니다. Kubernetes는 애플리케이션 유형에 따라 워크로드를 정의합니다.

1.5. 승인 플러그인 정보

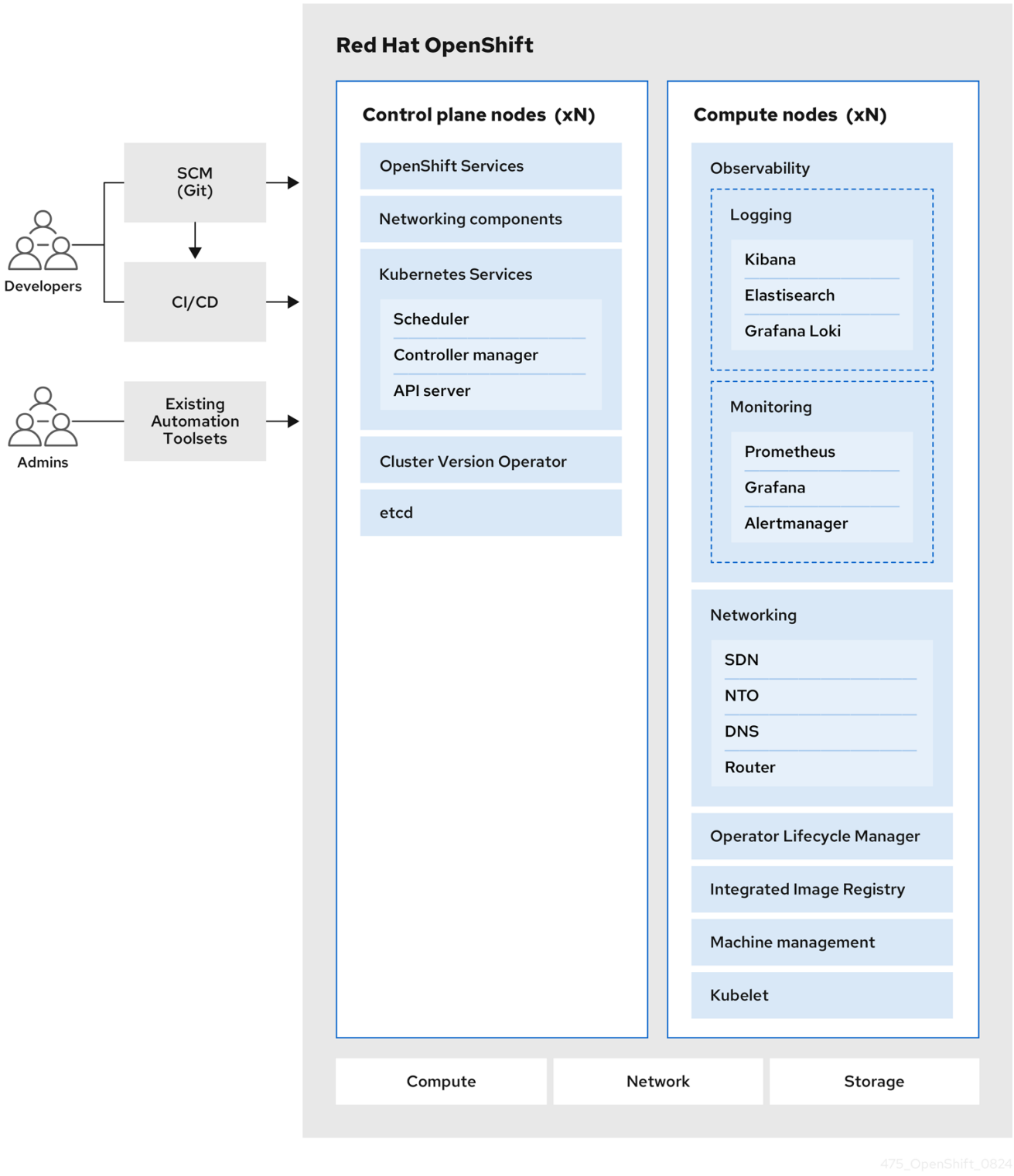
승인 플러그인 을 사용하여 OpenShift Dedicated의 기능을 규제할 수 있습니다. 리소스 요청이 인증되고 승인된 후 승인 플러그인은 마스터 API에 대한 리소스 요청을 가로채서 리소스 요청의 유효성을 검사하고 확장 정책이 준수되도록 합니다. 승인 플러그인은 보안 정책, 리소스 제한, 구성 요구 사항 및 기타 설정을 적용하는 데 사용됩니다.

2장. OPENSHIFT DEDICATED 아키텍처

2.1. OPENSHIFT DEDICATED 소개

OpenShift Dedicated는 컨테이너화된 애플리케이션을 개발하고 실행하기 위한 플랫폼입니다. 애플리케이션 및 애플리케이션을 지원하는 데이터센터를 몇 대의 머신 및 애플리케이션에서 수백만 클라이언트에 서비스를 제공하는 수천 대의 컴퓨터로 확장 가능하도록 설계되었습니다.

Kubernetes에 기반을 둔 OpenShift Dedicated는 대규모 통신, 스트리밍 비디오, 게임, 뱅킹 및 기타 애플리케이션의 엔진 역할을 하는 동일한 기술을 통합합니다. Red Hat의 오픈 기술로 구현하면 컨테이너화된 애플리케이션을 단일 클라우드에서 온프레미스 및 다중 클라우드 환경으로 확장할 수 있습니다.



2.1.1. Kubernetes 정보

컨테이너 이미지와 컨테이너 이미지에서 실행되는 컨테이너는 최신 애플리케이션 개발을 위한 기본 빌딩 블록이지만 규모에 맞게 실행하려면 안정적이고 유연한 배포 시스템이 필요합니다. Kubernetes는 컨테이너를 오케스트레이션하는 사실상의 표준입니다.

Kubernetes는 컨테이너화된 애플리케이션의 배포, 확장 및 관리를 자동화하기 위한 오픈 소스 컨테이너 오케스트레이션 엔진입니다. Kubernetes의 일반적인 개념은 다음과 같이 매우 간단합니다.

- 하나 이상의 작업자 노드에서 시작하여 컨테이너 워크로드를 실행합니다.

- 하나 이상의 컨트롤 플레인 노드에서 해당 워크로드의 배포를 관리합니다.
- 컨테이너를 포드라는 배포 단위로 래핑합니다. 포드를 사용하면 컨테이너에 추가 메타데이터가 제공되고 단일 배포 엔티티에서 여러 컨테이너를 그룹화할 수 있습니다.
- 특별한 종류의 자산을 생성합니다. 예를 들어 서비스는 포드 세트와 포드에 액세스하는 방법을 정의하는 정책으로 표시됩니다. 이 정책을 통해 컨테이너는 서비스에 대한 특정 IP 주소가 없어도 필요한 서비스에 연결할 수 있습니다. 복제 컨트롤러는 한 번에 실행하는 데 필요한 포드 복제본 수를 나타내는 또 다른 특수 자산입니다. 이 기능을 사용하여 현재 수요에 맞게 애플리케이션을 자동으로 확장할 수 있습니다.

불과 몇 년 만에 Kubernetes는 대규모 클라우드와 온프레미스에서 채택되었습니다. 오픈 소스 개발 모델을 사용하면 많은 사용자가 네트워킹, 스토리지 및 인증과 같은 구성 요소에 다른 기술을 구현하여 Kubernetes를 확장할 수 있습니다.

2.1.2. 컨테이너화된 애플리케이션의 이점

컨테이너화된 애플리케이션을 사용하면 기존 배포 방법을 사용하는 것보다 많은 이점이 있습니다. 애플리케이션이 모든 종속 항목을 포함하는 운영 체제에 한 번 설치될 것으로 예상되는 경우 컨테이너를 통해 애플리케이션이 해당 애플리케이션과 함께 종속 항목을 유지할 수 있습니다. 컨테이너화된 애플리케이션을 생성하면 많은 이점이 있습니다.

2.1.2.1. 운영 체제에서의 이점

컨테이너에서는 커널이 없는 소규모의 전용 Linux 운영 체제를 사용합니다. 파일 시스템, 네트워킹, cgroup, 프로세스 테이블 및 네임스페이스는 호스트 Linux 시스템과 별개이지만 필요한 경우 컨테이너를 호스트에 완벽하게 통합할 수 있습니다. 컨테이너는 Linux 기반이므로 빠르게 혁신되는 오픈 소스 개발 모델에 함께 제공되는 모든 이점을 사용할 수 있습니다.

각 컨테이너는 전용 운영 체제를 사용하므로 충돌하는 소프트웨어 종속 항목이 필요한 애플리케이션을 동일한 호스트에 배포할 수 있습니다. 각 컨테이너에서는 자체 종속 소프트웨어를 제공하고 네트워킹 및 파일 시스템과 같은 자체 인터페이스를 관리하므로 애플리케이션이 해당 자산과 경쟁할 필요가 없습니다.

2.1.2.2. 배포 및 스케일링에서의 이점

애플리케이션의 주요 릴리스 간에 롤링 업그레이드를 사용하는 경우 다운타임 없이 애플리케이션을 지속적으로 개선하고 현재 릴리스와의 호환성을 계속 유지할 수 있습니다.

기존 버전과 더불어 새 버전의 애플리케이션을 배포하고 테스트할 수도 있습니다. 컨테이너가 테스트를 통과하면 새 컨테이너를 추가로 배포하고 이전 컨테이너를 제거하면 됩니다.

애플리케이션의 모든 소프트웨어 종속 항목은 컨테이너 자체 내에서 해결되므로 데이터센터의 각 호스트에서 표준화된 운영 체제를 사용할 수 있습니다. 애플리케이션 호스트마다 특정 운영 체제를 구성할 필요가 없습니다. 데이터센터에 더 많은 용량이 필요하다면 또 다른 일반 호스트 시스템을 배포하면 됩니다.

컨테이너화된 애플리케이션도 이와 마찬가지로 간단하게 스케일링할 수 있습니다. OpenShift Dedicated는 컨테이너화된 서비스를 스케일링하는 간단한 표준 방법을 제공합니다. 예를 들어, 대규모 모놀리식 애플리케이션이 아닌 일련의 마이크로서비스로 애플리케이션을 빌드하는 경우 요구에 맞게 각 마이크로서비스를 개별적으로 스케일링할 수 있습니다. 이 기능을 사용하면 전체 애플리케이션이 아니라 필요한 서비스만 스케일링할 수 있으므로 최소한의 리소스만 사용하여 애플리케이션 요구사항을 충족할 수 있습니다.

2.1.3. OpenShift Dedicated 개요

OpenShift Dedicated는 다음과 같은 향상된 기능을 포함하여 Kubernetes에 엔터프라이즈급 개선 사항을 제공합니다.

- 통합된 Red Hat 기술. OpenShift Dedicated의 주요 구성 요소는 RHEL(Red Hat Enterprise Linux) 및 관련 Red Hat 기술을 기반으로 합니다. OpenShift Dedicated는 Red Hat의 엔터프라이즈급 소프트웨어에 대한 강력한 테스트 및 인증 이니셔티브의 이점을 제공합니다.

- 오픈 소스 개발 모델. 개발은 공개적으로 완료되었으며 소스 코드는 공개 소프트웨어 리포지토리에서 구할 수 있습니다. 이 오픈 협업을 통해 빠른 혁신과 개발을 촉진할 수 있습니다.

Kubernetes는 애플리케이션 관리에 뛰어나지만 플랫폼 수준 요구사항 또는 배포 프로세스를 지정하거나 관리하지는 않습니다. 강력하고 유연한 플랫폼 관리 도구 및 프로세스는 OpenShift Dedicated 4에서 제공하는 중요한 이점입니다. 다음 섹션에서는 OpenShift Dedicated의 몇 가지 고유한 기능과 이점에 대해 설명합니다.

2.1.3.1. 사용자 정의 운영 체제

OpenShift Dedicated는 모든 컨트롤 플레인 및 작업자 노드의 운영 체제로 RHCOS(Red Hat Enterprise Linux CoreOS)를 사용합니다.

RHCOS는 다음을 포함합니다.

- OpenShift Dedicated가 머신을 처음 시작하고 구성하기 위한 최초 부팅 시스템 구성으로 사용하는 Ignition

- 운영 체제와 밀접하게 통합되어 효율적이고 최적화된 Kubernetes 환경을 제공하는 Kubernetes 기본 컨테이너 런타임 구현인 CRI-O. CRI-O는 컨테이너 실행, 중지 및 다시 시작 기능을 제공합니다. OpenShift Dedicated 3에서 사용된 Docker Container Engine을 완전히 대체합니다.

- 컨테이너 시작 및 모니터링을 담당하는 Kubernetes의 기본 노드 에이전트인 Kubelet

2.1.3.2. 간소화된 업데이트 프로세스

OpenShift Dedicated 업데이트 또는 업그레이드는 고도로 자동화된 간단한 프로세스입니다. OpenShift Dedicated는 중앙 컨트롤 플레인에서 운영 체제 자체를 포함하여 각 머신에서 실행되는 시스템 및 서비스를 완전히 제어하므로 업그레이드가 자동 이벤트가 되도록 설계되었습니다.

2.1.3.3. 기타 주요 기능

Operator는 OpenShift Dedicated 4 코드 베이스의 기본 단위이자 애플리케이션 및 애플리케이션에서 사용할 소프트웨어 구성 요소를 편리하게 배포할 수 있는 방법입니다. OpenShift Dedicated에서 Operator는 플랫폼 기반 역할을 하며 운영 체제 및 컨트롤 플레인 애플리케이션을 수동으로 업그레이드할 필요가 없습니다. Cluster Version Operator 및 Machine Config Operator와 같은 OpenShift Dedicated Operator를 사용하면 중요한 구성 요소를 클러스터 전체로 관리할 수 있습니다.

OLM(Operator Lifecycle Manager)과 OperatorHub에서는 애플리케이션을 개발 및 배포하는 사용자에게 Operator를 저장하고 배포하는 기능을 제공합니다.

Red Hat Quay Container Registry는 대부분의 컨테이너 이미지와 Operator를 OpenShift Dedicated 클러스터에 제공하는 Quay.io 컨테이너 레지스트리입니다. Quay.io는 수백만 개의 이미지와 태그를 저장하는 Red Hat Quay의 공개 레지스트리 버전입니다.

OpenShift Dedicated의 Kubernetes의 기타 개선 사항에는 소프트웨어 정의 네트워킹(SDN), 인증, 로그 집계, 모니터링 및 라우팅 개선 사항이 포함됩니다. OpenShift Dedicated에서는 포괄적인 웹 콘솔 및 사용자 정의 OpenShift CLI(**oc**) 인터페이스도 제공합니다.

3장. 컨트롤 플레인 아키텍처

컨트롤 플레인 시스템으로 구성된 컨트롤 플레인 에서는 OpenShift Dedicated 클러스터를 관리합니다. 컨트롤 플레인 머신에서는 작업자 머신이라고도 하는 컴퓨팅 머신의 워크로드를 관리합니다. 클러스터 자체는 CVO(Cluster Version Operator) 작업과 개별 Operator 세트를 통해 머신의 모든 업그레이드를 관리합니다.

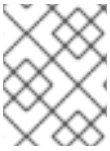
3.1. OPENSHIFT DEDICATED의 머신 역할

OpenShift Dedicated는 호스트에 다른 역할을 할당합니다. 이 역할은 클러스터 내에서 머신의 기능을 정의합니다. 클러스터에는 표준 **master** 및 **worker** 역할 유형에 대한 정의가 포함되어 있습니다.

3.1.1. 클러스터 작업자

Kubernetes 클러스터에서 작업자 노드는 Kubernetes 사용자가 요청한 실제 워크로드를 실행하고 관리합니다. 작업자 노드는 용량을 알리고 컨트롤 플레인 서비스인 스케줄러는 Pod 및 컨테이너를 시작할 노드를 결정합니다. 다음 중요한 서비스는 각 작업자 노드에서 실행됩니다.

- CRI-O - 컨테이너 엔진입니다.
- kubelet: 컨테이너 워크로드 실행 및 중지 요청을 수락하고 이행하는 서비스입니다.
- 작업자 간 Pod의 통신을 관리하는 서비스 프록시입니다.
- 컨테이너를 생성하고 실행하는 runC 또는 crun 하위 수준 컨테이너 런타임입니다.



참고

기본 runC 대신 crun을 활성화하는 방법에 대한 자세한 내용은 **ContainerRuntimeConfig** CR 생성 설명서를 참조하십시오.

OpenShift Dedicated에서 컴퓨팅 머신 세트는 **작업자** 머신 역할이 할당된 컴퓨팅 머신을 제어합니다. **worker** 역할 드라이브가 있는 머신은 자동 스케일링하는 특정 머신 풀이 관리하는 워크로드를 컴퓨팅합니다. OpenShift Dedicated에는 여러 머신 유형을 지원할 수 있는 용량이 있으므로 **worker** 역할의 머신은 *컴퓨팅* 머신으로 분류됩니다. 이 릴리스에서는 기본 유형의 컴퓨팅 머신이 작업자 머신이기 때문에 *작업자 머신*과 *컴퓨팅 머신*이라는 용어는 서로 바꿔 사용할 수 있습니다. 향후 OpenShift Dedicated 버전에서는 인프라 머신과 같은 다양한 유형의 컴퓨팅 머신이 기본적으로 사용될 수 있습니다.

참고

컴퓨팅 머신 세트는 **machine-api** 네임스페이스에서 컴퓨팅 머신 리소스의 그룹입니다. 컴퓨팅 머신 세트는 특정 클라우드 공급자에서 새 컴퓨팅 머신을 시작하도록 설계된 구성입니다. 반대로 MCP(Machine config pool)는 MCO(Machine Config Operator) 네임스페이스의 일부입니다. MCP는 MCO가 구성을 관리하고 업그레이드를 원활하게 수행할 수 있도록 머신을 그룹화하는데 사용됩니다.

3.1.2. 클러스터 컨트롤 플레인

Kubernetes 클러스터에서 *master* 노드는 Kubernetes 클러스터를 제어하는 데 필요한 서비스를 실행합니다. OpenShift Dedicated에서 컨트롤 플레인은 **마스터** 머신 역할이 있는 컨트롤 플레인 시스템으로 구성됩니다. OpenShift Dedicated 클러스터를 관리하기 위한 Kubernetes 서비스 이상의 것이 포함되어 있습니다.

대부분의 OpenShift Dedicated 클러스터의 경우 컨트롤 플레인 시스템은 일련의 독립 실행형 머신 API 리소스로 정의됩니다. 컨트롤 플레인은 컨트롤 플레인 머신 세트로 관리됩니다. 모든 컨트롤 플레인 시스템을 삭제하고 클러스터를 중단하지 못하도록 컨트롤 플레인 시스템에 추가 제어가 적용됩니다.

참고

단일 가용성 영역 클러스터 및 여러 가용성 영역 클러스터에는 최소 3개의 컨트롤 플레인 노드가 필요합니다.

컨트롤 플레인의 Kubernetes 카테고리에 속하는 서비스에는 Kubernetes API 서버, etcd, Kubernetes 컨트롤러 관리자 및 Kubernetes 스케줄러가 포함됩니다.

표 3.1. 컨트롤 플레인에서 실행되는 Kubernetes 서비스

구성 요소	설명
Kubernetes API 서버	Kubernetes API 서버는 포드, 서비스 및 복제 컨트롤러의 데이터를 검증하고 구성합니다. 클러스터의 공유 상태에 대한 초점도 제공합니다.
etcd	etcd는 영구 컨트롤 플레인 상태를 저장하고 다른 구성 요소는 etcd에서 변경 사항을 감시하여 지정된 상태로 전환합니다.
Kubernetes 컨트롤러 관리자	Kubernetes 컨트롤러 관리자는 etcd에서 복제, 네임스페이스 및 서비스 계정 컨트롤러 오브젝트와 같은 오브젝트의 변경사항을 감시한 다음 API를 사용하여 지정된 상태를 적용합니다. 이러한 여러 프로세스는 한 번에 하나의 활성 리더가 있는 클러스터를 생성합니다.
Kubernetes 스케줄러	Kubernetes 스케줄러는 할당된 노드가 없는 새로 생성된 Pod를 감시하고 Pod를 호스팅할 수 있는 최상의 노드를 선택합니다.

OpenShift API 서버, OpenShift 컨트롤러 관리자 및 OpenShift OAuth API 서버 및 OpenShift OAuth 서버를 포함하는 컨트롤 플레인에서 실행되는 OpenShift 서비스도 있습니다.

표 3.2. 컨트롤 플레인에서 실행되는 OpenShift 서비스

구성 요소	설명
OpenShift API 서버	OpenShift API 서버는 프로젝트, 경로 및 템플릿과 같은 OpenShift 리소스의 데이터 유효성을 검사하고 구성합니다. OpenShift API 서버는 OpenShift API Server Operator가 관리합니다.
OpenShift 컨트롤러 관리자	OpenShift 컨트롤러 관리자는 etcd에서 프로젝트, 경로 및 템플릿 컨트롤러 오브젝트와 같은 OpenShift 오브젝트의 변경사항을 감시한 다음 API를 사용하여 지정된 상태를 적용합니다. OpenShift 컨트롤러 관리자는 OpenShift Controller Manager Operator가 관리합니다.

구성 요소	설명
OpenShift OAuth API 서버	OpenShift OAuth API 서버는 사용자, 그룹, OAuth 토큰과 같은 OpenShift Dedicated에 인증할 데이터를 검증하고 구성합니다. OpenShift OAuth API 서버는 Cluster Authentication Operator가 관리합니다.
OpenShift OAuth 서버	사용자는 OpenShift OAuth 서버에서 토큰을 요청하여 API에 자신을 인증합니다. OpenShift OAuth 서버는 Cluster Authentication Operator가 관리합니다.

컨트롤 플레인 시스템에서 이러한 서비스 중 일부는 systemd 서비스로 실행되는 반면 다른 서비스는 정적 Pod로 실행됩니다.

시스템 서비스는 특정 시스템이 시작된 직후에 항상 필요한 서비스에 적합합니다. 컨트롤 플레인 시스템의 경우 원격 로그인을 허용하는 sshd가 포함됩니다. 다음과 같은 서비스도 포함됩니다.

- 컨테이너를 실행하고 관리하는 CRI-O 컨테이너 엔진(crio). OpenShift Dedicated 4는 Docker Container Engine 대신 CRI-O를 사용합니다.
- kubelet(kubelet): 컨트롤 플레인 서비스에서 머신의 컨테이너 관리 요청을 수락합니다.

CRI-O 및 Kubelet은 다른 컨테이너를 실행하기 전에 실행되어야 하기 때문에 systemd 서비스로 호스트에서 직접 실행해야 합니다.

installer-* 및 **revision-pruner-*** 컨트롤 플레인 Pod는 root 사용자가 소유한 **/etc/kubernetes** 디렉토리에 쓰기 때문에 root 권한으로 실행해야 합니다. 이러한 pod에는 다음과 같은 네임스페이스에 있습니다.

- **openshift-etcd**
- **openshift-kube-apiserver**
- **openshift-kube-controller-manager**
- **openshift-kube-scheduler**

3.2. OPENSHIFT DEDICATED의 OPERATOR

Operator는 OpenShift Dedicated의 가장 중요한 구성 요소 중 하나입니다. 컨트롤 플레인에서 서비스를 패키징, 배포 및 관리하는 데 선호되는 방법입니다. 또한 사용자가 실행하는 애플리케이션에 이점을 제공할 수 있습니다.

Operator는 Kubernetes API 및 **kubectl** 및 OpenShift CLI(**oc**)와 같은 CLI 툴과 통합됩니다. 애플리케이션을 모니터링하고, 상태 점검 수행, OTA(Over-the-Air) 업데이트를 관리하며 애플리케이션이 지정된 상태로 유지되도록 합니다.

Operator에서는 더 세분화된 구성 환경도 제공합니다. 글로벌 구성 파일을 수정하는 대신 Operator가 표시하는 API를 수정하여 각 구성 요소를 구성합니다.

CRI-O 및 Kubelet은 모든 노드에서 실행되므로 Operator를 사용하여 거의 모든 다른 클러스터 기능을 컨트롤 플레인에서 관리할 수 있습니다. Operator를 사용하여 컨트롤 플레인에 추가되는 구성 요소에는 중요한 네트워킹 및 인증 정보 서비스가 포함됩니다.

둘 다 유사한 Operator 개념과 목표를 따르지만 OpenShift Dedicated의 Operator는 목적에 따라 두 개의 다른 시스템으로 관리됩니다.

클러스터 Operator

CVO(Cluster Version Operator)에 의해 관리되고 클러스터 기능을 수행하기 위해 기본적으로 설치됩니다.

선택적 애드온 Operator

OLM(Operator Lifecycle Manager)에서 관리하며 사용자가 애플리케이션에서 실행할 수 있도록 할 수 있습니다. *OLM 기반 Operator라고도 합니다*.

3.2.1. 애드온 Operator

OLM(Operator Lifecycle Manager) 및 OperatorHub는 OpenShift Dedicated의 기본 구성 요소이며 Kubernetes 네이티브 애플리케이션을 Operator로 관리하는 데 도움이 됩니다. 함께 클러스터에서 사용 가능한 선택적 애드온 Operator를 검색, 설치 및 관리하기 위한 시스템을 제공합니다.

OpenShift Dedicated 웹 콘솔에서 OperatorHub를 사용하여 **dedicated-admin** 역할 및 권한이 있는 관리자는 Operator 카탈로그에서 설치할 Operator를 선택할 수 있습니다. OperatorHub에서 Operator를 설치한 후에는 전역 또는 특정 네임스페이스에서 사용자 애플리케이션에서 실행할 수 있도록 할 수 있습니다.

Red Hat Operator, 인증된 Operator, 커뮤니티 Operator가 포함된 기본 카탈로그 소스를 사용할 수 있습니다. **dedicated-admin** 역할의 관리자는 사용자 정의 Operator 세트를 포함할 수 있는 고유한 사용자 정의 카탈로그 소스도 추가할 수 있습니다.

참고

Operator Hub Marketplace에 나열된 모든 Operator를 설치할 수 있어야 합니다. 이러한 Operator는 고객 워크로드로 간주되며 Red Hat 사이트 안정성 엔지니어링(SRE)에서 모니터링하지 않습니다.

개발자는 Operator SDK를 사용하여 OLM 기능을 활용하는 사용자 정의 Operator를 작성할 수 있습니다. 그런 다음 Operator를 번들로 제공하고 사용자 정의 카탈로그 소스에 추가할 수 있습니다. 이 소스는 클러스터에 추가하고 사용자가 사용할 수 있도록 합니다.

참고

OLM은 OpenShift Dedicated 아키텍처를 구성하는 클러스터 Operator를 관리하지 않습니다.

추가 리소스

- OpenShift Dedicated에서 애드온 Operator를 실행하는 방법에 대한 자세한 내용은 OLM(*Operator* Lifecycle Manager) 및 OperatorHub 에 대한 Operator 가이드 섹션을 참조하십시오.
- Operator SDK에 대한 자세한 내용은 Operator 개발을 참조하십시오.

3.3. ETCD 개요

etcd는 메모리에 완전히 들어갈 수 있는 적은 양의 데이터를 보유하는 일관된 분산 키-값 저장소입니다. etcd는 많은 프로젝트의 핵심 구성 요소이지만 Kubernetes의 기본 데이터 저장소이며 이는 컨테이너 오케스트레이션의 표준 시스템입니다.

3.3.1. etcd를 사용하는 이점

etcd를 사용하면 다음과 같은 여러 가지 방법으로 이점을 누릴 수 있습니다.

- 클라우드 네이티브 애플리케이션의 일관된 가동 시간을 유지하고 개별 서버가 실패하더라도 계속 작동합니다.
- Kubernetes의 모든 클러스터 상태 저장 및 복제
- 구성 데이터를 배포하여 노드 구성에 중복 및 복원력 제공

3.3.2. etcd 작동 방식

클러스터 구성 및 관리에 대한 안정적인 접근 방식을 보장하기 위해 etcd는 etcd Operator를 사용합니다. Operator는 OpenShift Dedicated와 같은 Kubernetes 컨테이너 플랫폼에서 etcd 사용을 단순화합니다. etcd Operator를 사용하면 etcd 멤버를 만들거나 삭제하고, 클러스터의 크기를 조정하고, 백업을 수행하고, etcd를 업그레이드할 수 있습니다.

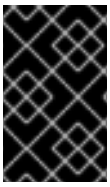
etcd Operator는 다음을 관찰, 분석 및 작동합니다.

1. Kubernetes API를 사용하여 클러스터 상태를 관찰합니다.
2. 현재 상태와 원하는 상태 간의 차이점을 분석합니다.
3. etcd 클러스터 관리 API, Kubernetes API 또는 둘 다를 통해 차이점을 수정합니다.

etcd에는 클러스터 상태가 있으며 이는 지속적으로 업데이트됩니다. 이 상태는 지속적으로 유지되므로 높은 빈도로 인해 작은 변화가 많이 발생합니다. 결과적으로 Red Hat SRE(사이트 안정성 엔지니어링)는 대기 시간이 짧은 I/O로 etcd 클러스터 멤버를 백업합니다.

3.4. 컨트롤 플레인 구성 자동 업데이트

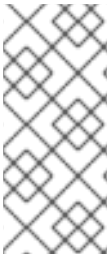
OpenShift Dedicated 클러스터에서 컨트롤 플레인 머신 세트는 컨트롤 플레인 구성에 변경 사항을 자동으로 전파합니다. 컨트롤 플레인 머신을 교체해야 하는 경우 컨트롤 플레인 머신 세트 Operator는 **ControlPlaneMachineSet** CR(사용자 정의 리소스)에 지정된 구성을 기반으로 교체 머신을 생성합니다. 새 컨트롤 플레인 머신이 준비되면 Operator는 클러스터 API 또는 워크로드 가용성에 미치는 영향을 완화하는 방식으로 이전 컨트롤 플레인 머신을 안전하게 드레이닝하고 종료합니다.



중요

유지 관리 기간 중에만 컨트롤 플레인 머신 교체를 요청할 수 없습니다. 컨트롤 플레인 머신 세트 Operator는 클러스터 안정성을 보장하기 위해 작동합니다. 유지 관리 기간 대기 중으로 인해 클러스터 안정성이 손상될 수 있습니다.

컨트롤 플레인 시스템은 언제든지 교체용으로 표시할 수 있습니다. 일반적으로 머신이 사양이 부족하거나 비정상 상태로 입력되었기 때문입니다. 이러한 교체는 클러스터 라이프사이클의 정상적인 부분이며 문제가 발생하지 않습니다. 컨트롤 플레인 노드 교체에 실패하는 경우 SRE가 자동으로 문제에 대해 경고합니다.



참고

OpenShift Dedicated 클러스터가 원래 생성된 시기에 따라 컨트롤 플레인 머신 세트의 도입은 다른 컨트롤 플레인 노드와 일치하지 않는 레이블 또는 머신 이름으로 하나 또는 두 개의 컨트롤 플레인 노드를 남겨 둘 수 있습니다. 예를 들어 **clustername-master-0**,**clustername-master-1** 및 **clustername-master-2-abcxyz**. 이러한 이름 불일치는 클러스터 작동에 영향을 미치지 않으며 문제가 되지 않습니다.

3.5. 실패한 컨트롤 플레인 시스템 복구

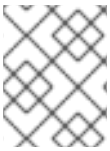
Control Plane Machine Set Operator는 컨트롤 플레인 머신의 복구를 자동화합니다. 컨트롤 플레인 머신이 삭제되면 Operator는 **ControlPlaneMachineSet** CR(사용자 정의 리소스)에 지정된 구성으로 교체를 생성합니다.

4장. NVIDIA GPU 아키텍처 개요

NVIDIA는 OpenShift Dedicated에서 GPU(그래픽 처리 장치) 리소스 사용을 지원합니다. OpenShift Dedicated는 대규모 Kubernetes 클러스터를 배포하고 관리하기 위해 Red Hat에서 개발하고 지원하는 보안 중심 및 강화된 Kubernetes 플랫폼입니다. OpenShift Dedicated에는 Kubernetes에 대한 개선 사항이 포함되어 있어 사용자가 워크로드를 가속화하기 위해 NVIDIA GPU 리소스를 쉽게 구성하고 사용할 수 있습니다.

NVIDIA GPU Operator는 OpenShift Dedicated 내에서 Operator 프레임워크를 활용하여 GPU 가속 워크로드를 실행하는 데 필요한 NVIDIA 소프트웨어 구성 요소의 전체 라이프사이클을 관리합니다.

이러한 구성 요소에는 NVIDIA 드라이버(CUDA 활성화), GPU용 Kubernetes 장치 플러그인, NVIDIA 컨테이너 툴킷, GPU 기능 검색을 사용한 자동 노드 태그 지정, DCGM 기반 모니터링 등이 포함됩니다.



참고

NVIDIA GPU Operator는 NVIDIA에서만 지원됩니다. NVIDIA에서 지원을 얻는 방법에 대한 자세한 내용은 NVIDIA에서 지원 받기를 참조하십시오.

4.1. NVIDIA GPU 사전 요구 사항

- GPU 작업자 노드가 하나 이상 있는 작동 중인 OpenShift 클러스터입니다.
- 필요한 단계를 수행하려면 **cluster-admin** 으로 OpenShift 클러스터에 액세스할 수 있습니다.
- OpenShift CLI(**oc**)가 설치되어 있어야 합니다.
- NFD(노드 기능 검색) Operator가 설치되고 **nodefeaturediscovery** 인스턴스가 생성됩니다.

4.2. GPU 및 OSD

NVIDIA GPU 인스턴스 유형에 OpenShift Dedicated를 배포할 수 있습니다.

이 컴퓨팅 인스턴스는 GPU 가속 컴퓨팅 인스턴스이고 GPU 유형이 NVIDIA AI Enterprise의 지원되는 GPU 목록과 일치해야 합니다. 예를 들어 T4, V100 및 A100은 이 목록의 일부입니다.

컨테이너화된 GPU에 액세스하기 위해 다음 방법 중 하나를 선택할 수 있습니다.

- VM(가상 머신) 내에서 GPU 하드웨어에 액세스하고 사용하기 위한 GPU 패스스루입니다.
- 전체 GPU가 필요하지 않은 경우 GPU(vGPU) 시간 분할입니다.

추가 리소스

- 클라우드의 Red Hat Openshift

4.3. GPU 공유 방법

Red Hat과 NVIDIA는 엔터프라이즈급 OpenShift Dedicated 클러스터에서 GPU 가속 컴퓨팅을 단순화하기 위해 GPU 동시성 및 공유 메커니즘을 개발했습니다.

일반적으로 애플리케이션에는 GPU를 사용하지 않는 다른 컴퓨팅 요구 사항이 있습니다. 각 워크로드에 적합한 양의 컴퓨팅 리소스를 제공하는 것은 배포 비용을 줄이고 GPU 사용률을 극대화하는 데 중요합니다.

가상화를 포함한 프로그래밍 모델 API에서 시스템 소프트웨어 및 하드웨어 파티셔닝에 이르기까지 다양한 GPU 사용률을 개선하기 위한 동시성 메커니즘이 존재합니다. 다음 목록은 GPU 동시성 메커니즘을 보여줍니다.

- Compute Unified Device Architecture (CUDA) 스트림
- 시간 분할
- CUDA Multi-Process Service (MPS)
- 다중 인스턴스 GPU(MIG)
- vGPU를 사용한 가상화

추가 리소스

- GPU 사용률 개선

4.3.1. CUDA 스트림

CUDA(Compute Unified Device Architecture)는 GPU의 일반 컴퓨팅을 위해 NVIDIA에서 개발한 병렬 컴퓨팅 플랫폼 및 프로그래밍 모델입니다.

스트림은 GPU에서 문제순으로 실행되는 일련의 작업입니다. CUDA 명령은 일반적으로 기본 스트림에서 순차적으로 실행되며 이전 작업이 완료될 때까지 작업이 시작되지 않습니다.

다양한 스트림의 작업을 비동기적으로 처리하면 병렬 작업을 실행할 수 있습니다. 한 스트림에서 발행된 작업은 다른 작업이 다른 스트림으로 발행되기 전, 중 또는 후에 실행됩니다. 이를 통해 GPU는 정해진 순서로 여러 작업을 동시에 실행하여 성능이 향상됩니다.

추가 리소스

- 비동기 동시 실행

4.3.2. 시간 분할

GPU 시간 분할은 여러 CUDA 애플리케이션을 실행할 때 과부하된 GPU에서 예약된 워크로드를 상호 저장합니다.

GPU의 복제본 세트를 정의하여 Kubernetes에서 GPU를 시간 분할할 수 있으며 각각 워크로드를 실행하기 위해 개별적으로 Pod에 배포할 수 있습니다. MIG(Multi-instance GPU)와 달리 복제본 간 메모리 또는 내결함성이 없지만 일부 워크로드의 경우 공유하지 않는 것보다 좋습니다. 내부적으로 GPU 시간 분할은 동일한 기본 GPU의 복제본에서 멀티플렉션에 사용됩니다.

시간 분할에 대한 클러스터 전체 기본 구성을 적용할 수 있습니다. 노드별 구성을 적용할 수도 있습니다. 예를 들어 Cryostat T4 GPU가 있는 노드에만 시간 분할 구성을 적용하고 다른 GPU 모델의 노드를 수정할 수 없습니다.

클러스터 전체 기본 구성을 적용한 다음 노드에 레이블을 지정하여 해당 노드에 노드별 구성을 지정하여 이러한 두 가지 접근 방식을 결합할 수 있습니다.

4.3.3. CUDA 다중 프로세스 서비스

CUDA Multi-Process Service(MPS)를 사용하면 단일 GPU가 여러 CUDA 프로세스를 사용할 수 있습니다. 프로세스는 GPU에서 병렬로 실행되며 GPU 컴퓨팅 리소스의 포화 상태를 제거합니다. 또한 MPS는 다양한 프로세스에서 커널 작업 및 메모리를 복사하여 사용률을 향상시킬 수 있습니다.

추가 리소스

- CUDA MPS

4.3.4. 다중 인스턴스 GPU

MG(Multi-instance GPU)를 사용하여 GPU 컴퓨팅 단위와 메모리를 여러MIG 인스턴스로 분할할 수 있습니다. 이러한 각 인스턴스는 시스템 관점에서 독립 실행형 GPU 장치를 나타내며 노드에서 실행되는 모든 애플리케이션, 컨테이너 또는 가상 머신에 연결할 수 있습니다. GPU를 사용하는 소프트웨어는 이러한 기타 각 인스턴스를 개별 GPU로 처리합니다.

MIG는 전체 GPU의 전체 성능이 필요하지 않은 애플리케이션이 있는 경우에 유용합니다. 새로운 NVIDIA Ampere 아키텍처의 MIG 기능을 사용하면 하드웨어 리소스를 여러 GPU 인스턴스로 분할할 수 있으며 각 GPU 인스턴스는 운영 체제에서 독립적인 CUDA 지원 GPU로 사용할 수 있습니다.

NVIDIA GPU Operator 버전 1.7.0 이상은 A100 및 A30 Ampere 카드에 대해MIG 지원을 제공합니다. 이러한 GPU 인스턴스는 전용 하드웨어 리소스와 완전히 격리된 최대 7개의 독립 CUDA 애플리케이션을 지원하도록 설계되었습니다.

추가 리소스

- NVIDIA Multi-Instance GPU 사용자 가이드

4.3.5. vGPU를 사용한 가상화

가상 머신(VM)은 NVIDIA vGPU를 사용하여 단일 물리적 GPU에 직접 액세스할 수 있습니다. 엔터프라이즈 전체의 VM에서 공유하여 다른 장치에서 액세스할 수 있는 가상 GPU를 생성할 수 있습니다.

이 기능은 GPU 성능의 기능과 vGPU가 제공하는 관리 및 보안 이점을 결합합니다. vGPU가 제공하는 추가 이점으로 VM 환경에 대한 사전 관리 및 모니터링, 혼합 VDI 및 컴퓨팅 워크로드를 위한 워크로드 분산, 여러 VM의 리소스 공유 등이 포함됩니다.

추가 리소스

- 가상 GPU

4.4. OPENSHIFT DEDICATED용 NVIDIA GPU 기능

NVIDIA 컨테이너 툴킷

NVIDIA 컨테이너 툴킷을 사용하면 GPU 가속 컨테이너를 생성하고 실행할 수 있습니다. 툴킷에는 NVIDIA GPU를 사용하도록 컨테이너를 자동으로 구성하는 컨테이너 런타임 라이브러리 및 유틸리티가 포함되어 있습니다.

NVIDIA AI Enterprise

NVIDIA AI Enterprise는 NVIDIA 인증 시스템에서 지원, 인증 및 지원되는 포괄적인 AI 및 데이터 분석 소프트웨어 제품군입니다.
NVIDIA AI Enterprise에는 Red Hat OpenShift Dedicated가 포함되어 있습니다. 지원되는 설치 방법은 다음과 같습니다.

- GPU Passthrough가 있는 베어 메탈 또는 VMware vSphere의 OpenShift Dedicated

- NVIDIA vGPU가 있는 VMware vSphere의 OpenShift Dedicated.

GPU 기능 검색

Kubernetes용 NVIDIA GPU 기능 검색은 노드에서 사용 가능한 GPU에 대한 레이블을 자동으로 생성할 수 있는 소프트웨어 구성 요소입니다. GPU 기능 검색에서는 NFD(노드 기능 검색)를 사용하여 이 레이블을 수행합니다.
NFD(Node Feature Discovery Operator)는 하드웨어 관련 정보로 노드에 레이블을 지정하여 OpenShift Container Platform 클러스터에서 하드웨어 기능 및 구성 검색을 관리합니다. NFD는 PCI 카드, 커널, OS 버전과 같은 노드별 속성을 사용하여 호스트에 레이블을 지정합니다.

"Node Feature Discovery"를 검색하여 Operator Hub에서 NFD Operator를 찾을 수 있습니다.

OpenShift Virtualization을 사용하는 NVIDIA GPU Operator

이 시점까지 GPU Operator는 GPU 가속 컨테이너를 실행하기 위해 작업자 노드만 프로비저닝했습니다. 이제 GPU Operator를 사용하여 GPU 가속 VM(가상 머신)을 실행하기 위해 작업자 노드를 프로비저닝할 수도 있습니다.
해당 노드에서 실행되도록 구성된 GPU 워크로드에 따라 작업자 노드에 다른 소프트웨어 구성 요소를 배포하도록 GPU Operator를 구성할 수 있습니다.

GPU 모니터링 대시보드

모니터링 대시보드를 설치하여 OpenShift Dedicated 웹 콘솔의 클러스터 모니터링 페이지에 **대한** GPU 사용 정보를 표시할 수 있습니다. GPU 사용률 정보에는 사용 가능한 GPU 수, 전력 소비(단위: 와트), 온도(단위: 섭씨), 사용률(%) 및 각 GPU에 대한 기타 메트릭이 포함됩니다.

추가 리소스

- NVIDIA 인증 시스템
- NVIDIA AI Enterprise
- NVIDIA 컨테이너 툴킷
- GPU 모니터링 대시보드 활성화
- OpenShift Container Platform에서 MIG 지원
- OpenShift에서 NVIDIA GPU 시간 분할
- 연결이 끊긴 또는 Airgapped 환경에 GPU Operator 배포

5장. OPENSHIFT DEDICATED 개발 이해

엔터프라이즈급 애플리케이션을 개발하고 실행할 때 컨테이너의 기능을 완전히 활용하려면 컨테이너가 다음을 수행할 수 있게 하는 도구를 통해 환경을 지원해야 합니다.

- 컨테이너화되었거나 그렇지 않은 다른 서비스에 연결할 수 있는 개별 마이크로서비스로 생성됩니다. 예를 들어, 애플리케이션을 데이터베이스와 결합하거나 모니터링 애플리케이션을 데이터베이스에 연결할 수 있습니다.
- 복원력이 뛰어나서 서버가 충돌하거나 유지보수 또는 서비스 해제를 위해 서버를 가동 중단해야 하는 경우 컨테이너가 또 다른 머신에서 시작될 수 있습니다.
- 코드 변경 사항을 자동으로 가져온 다음 새 버전을 자체 시작 및 배포하도록 자동화되었습니다.
- 수요가 증가하면 더 많은 인스턴스가 클라이언트에 서비스를 제공하고 수요가 감소하면 인스턴스 수가 감소하도록 확장 또는 복제됩니다.
- 애플리케이션 유형에 따라 다른 방식으로 실행할 수 있습니다. 예를 들어 한 애플리케이션이 한 달에 한 번 실행되어 보고서를 생성한 다음 종료될 수 있습니다. 또 다른 애플리케이션은 지속적으로 실행되어야 하며 클라이언트가 사용할 수 있는 가능성이 높아야 합니다.
- 애플리케이션 상태를 보고 문제가 발생했을 때 대응할 수 있도록 관리되었습니다.

컨테이너가 광범위하게 사용되고, 결과적으로 엔터프라이즈를 지원하기 위한 도구 및 방법이 필요하게 됨에 따라 많은 옵션이 제공되었습니다.

이 섹션의 나머지 부분에서는 OpenShift Dedicated에서 컨테이너화된 Kubernetes 애플리케이션을 빌드하고 배포할 때 생성할 수 있는 자산 옵션에 대해 설명합니다. 또한 다양한 종류의 애플리케이션 및 개발 요구사항에 사용할 수 있는 방법에 대해서도 설명합니다.

5.1. 컨테이너화된 애플리케이션 개발 정보

여러 가지 방법으로 컨테이너를 사용하여 애플리케이션 개발에 접근할 수 있으며 상황에 따라 다른 접근 방식이 더 적합할 수 있습니다. 이러한 다양성 중 일부를 설명하기 위해 제시된 일련의 접근 방식은 단일 컨테이너를 개발하는 것으로 시작하여 궁극적으로 해당 컨테이너를 대기업의 미션 크리티컬 애플리케이션으로 배포하는 것입니다. 이러한 접근 방식은 컨테이너화된 애플리케이션 개발에 사용할 수 있는 다양한 도구, 형식 및 방법을 보여줍니다. 이 주제에서는 다음을 설명합니다.

- 간단한 컨테이너를 빌드하여 레지스트리에 저장
- Kubernetes 매니페스트를 생성하여 Git 리포지토리에 저장
- Operator가 다른 사용자와 애플리케이션을 공유하게 만들기

5.2. 간단한 컨테이너 빌드

애플리케이션에 대한 아이디어가 있으며 컨테이너화하고 싶은 경우를 살펴보겠습니다.

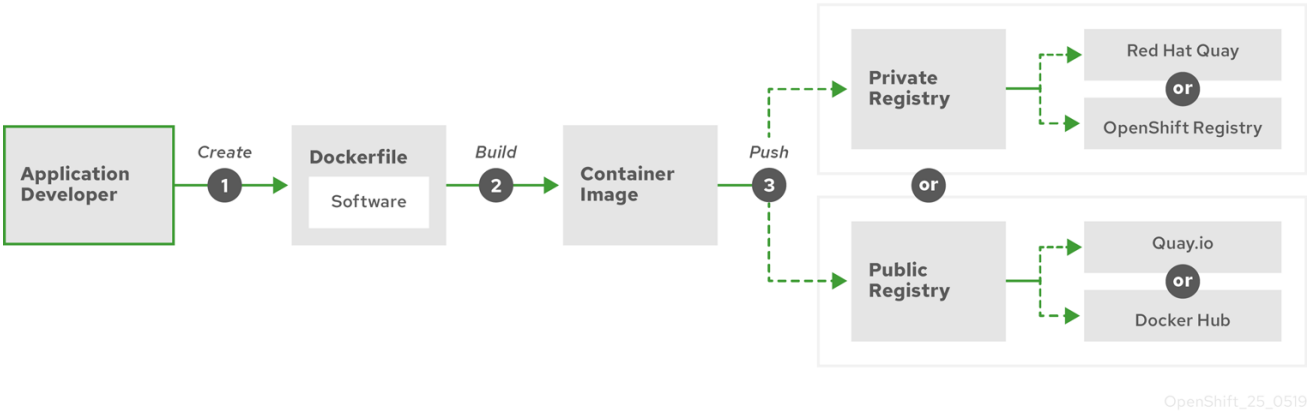
먼저 buildah 또는 docker와 같은 컨테이너를 빌드하는 도구와 컨테이너에 들어가는 내용을 설명하는 파일(일반적으로 Dockerfile)이 필요합니다.

다음으로 결과 컨테이너 이미지를 넣을 위치가 필요하므로 실행하려는 위치로 끌어올 수 있습니다. 이 위치는 컨테이너 레지스트리입니다.

이러한 각 구성 요소의 몇 가지 예는 사용자가 직접 제공하는 Dockerfile을 제외하고 대부분의 Linux 운영 체제에 기본적으로 설치됩니다.

다음 다이어그램은 이미지를 작성하고 푸시하는 프로세스를 보여줍니다.

그림 5.1. 컨테이너화된 간단한 애플리케이션을 생성하여 레지스트리로 푸시



RHEL(Red Hat Enterprise Linux)을 운영 체제로 실행하는 컴퓨터를 사용하는 경우 컨테이너화된 애플리케이션을 생성하는 프로세스에는 다음 단계가 필요합니다.

1. 컨테이너 빌드 도구 설치: RHEL에는 컨테이너를 빌드하고 관리하는 데 사용하는 podman, buildah 및 skopeo가 포함된 도구 세트가 포함되어 있습니다.

2. 기본 이미지와 소프트웨어를 결합하는 Dockerfile 생성: 컨테이너 빌드에 대한 정보는 **Dockerfile**이라는 파일에 입력됩니다. 이 파일에서 빌드한 기본 이미지, 설치한 소프트웨어 패키지 및 컨테이너에 복사한 소프트웨어를 식별합니다. 컨테이너 외부에 노출되는 네트워크 포트 및 컨테이너 내부에 마운트된 볼륨과 같은 매개변수 값도 식별합니다. RHEL 시스템의 디렉터리에 Dockerfile과 컨테이너화된 소프트웨어를 배치합니다.

3. buildah 또는 docker 빌드 실행: **buildah build-using-dockerfile** 또는 **docker build** 명령을 실행하여 선택한 기본 이미지를 가져와 로컬에 저장된 컨테이너 이미지를 생성합니다. buildah를 사용하여 Dockerfile 없이 컨테이너 이미지를 빌드할 수도 있습니다.

4. 태그 지정 및 레지스트리로 푸시: 컨테이너를 저장하고 공유할 레지스트리의 위치를 식별하는 새 컨테이너 이미지에 태그를 추가하십시오. 그런 다음 **podman push** 또는 **docker push** 명령을 실행하여 레지스트리에 이미지를 푸시합니다.

5. 이미지 가져오기 및 실행: podman 또는 docker와 같은 컨테이너 클라이언트 도구가 있는 시스템에서 새 이미지를 식별하는 명령을 실행하십시오. 예를 들어 **podman run <image_name>** 또는 **docker run <image_name>** 명령을 실행하십시오. 여기서 **<image_name>**은 새 컨테이너 이미지의 이름이며, **quay.io/myrepo/myapp:latest**와 비슷합니다. 이미지를 푸시하고 가져오려면 레지스트리에 자격 증명이 필요할 수 있습니다.

5.2.1. 컨테이너 빌드 도구 옵션

buildah, podman 및 skopeo를 사용하여 컨테이너를 빌드하고 관리하면 OpenShift Dedicated 또는 기타 Kubernetes 환경에서 컨테이너 배포를 위해 특별히 조정된 기능이 포함된 업계 표준 컨테이너 이미지가 생성됩니다. 이러한 툴은 데몬이 없으며 루트 권한 없이 실행할 수 있으므로 실행하기 위한 오버헤드가 줄어듭니다.

중요

컨테이너 런타임으로 Docker Container Engine에 대한 지원은 Kubernetes 1.20에서 더 이상 사용되지 않으며 향후 릴리스에서 제거됩니다. 그러나 Docker 생성 이미지는 CRI-O를 포함하여 모든 런타임에서 클러스터에서 계속 작동합니다. 자세한 내용은 Kubernetes 블로그 공지 를 참조하십시오.

궁극적으로 OpenShift Dedicated에서 컨테이너를 실행하면 CRI-O 컨테이너 엔진을 사용합니다. CRI-O는 OpenShift Dedicated 클러스터의 모든 작업자 및 컨트롤 플레인 머신에서 실행되지만 CRI-O는 아직 OpenShift Dedicated 외부의 독립 실행형 런타임으로 지원되지 않습니다.

5.2.2. 기본 이미지 옵션

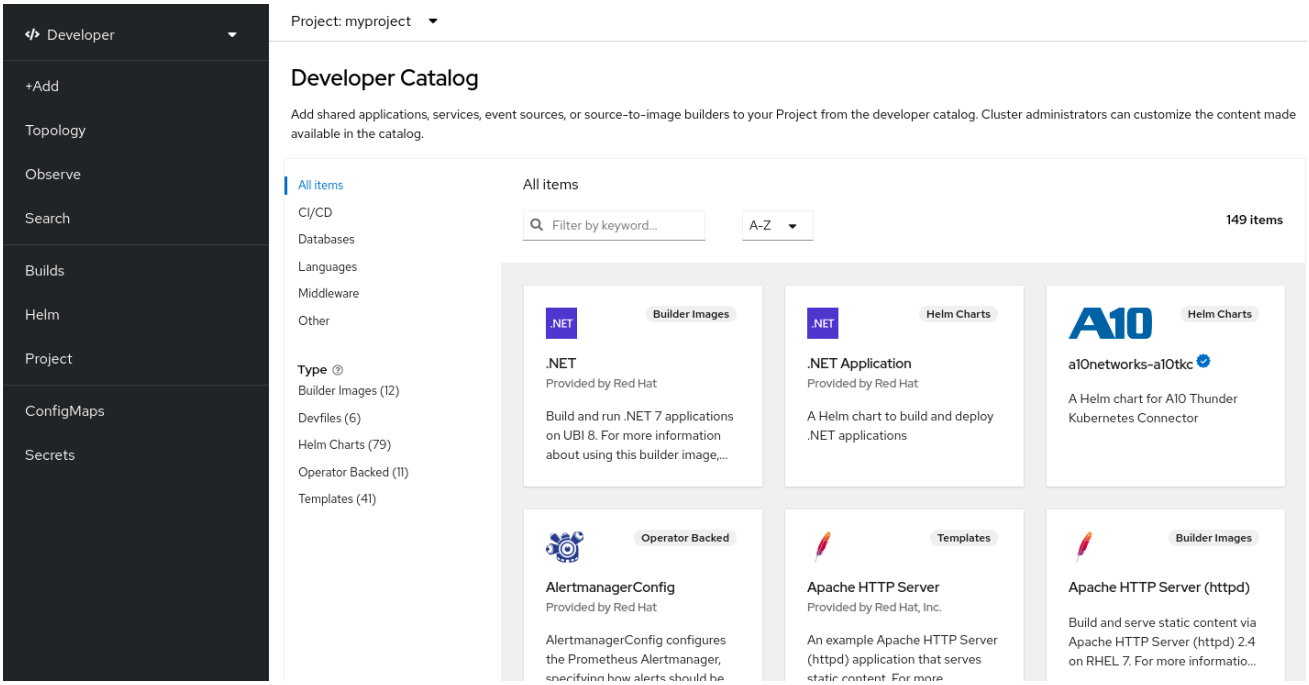
애플리케이션을 빌드하기 위해 선택한 기본 이미지에는 애플리케이션의 Linux 시스템과 유사한 소프트웨어 세트가 포함되어 있습니다. 고유 이미지를 빌드하면 소프트웨어가 해당 파일 시스템에 배치되고 해당 파일 시스템이 운영 체제를 보는 것처럼 인식합니다. 이 기본 이미지를 선택하면 향후 컨테이너의 안전성, 효율성 및 업그레이드 가능성에 큰 영향을 미칩니다.

Red Hat에서는 Red Hat Universal Base Images (UBI)라는 새로운 기본 이미지 세트를 제공합니다. 이 이미지는 Red Hat Enterprise Linux를 기반으로 하며 과거 Red Hat에서 제공한 기본 이미지와 유사하지만 한 가지 큰 차이점이 있습니다. 이는 Red Hat 서브스크립션이 없이 자유롭게 재배포할 수 있습니다. 결과적으로 공유 방법이나 환경에 따라 다른 이미지를 만드는 것에 대한 걱정 없이 UBI 이미지에서 애플리케이션을 빌드할 수 있습니다.

이 UBI 이미지에는 표준, 초기화 및 최소 버전이 있습니다. Node.js, Perl 또는 Python과 같은 특정 런타임 환경에 종속된 애플리케이션의 기초로 Red Hat Software Collections 이미지를 사용할 수도 있습니다. 이러한 런타임 기본 이미지 중 일부 특수 버전은 S2I(Source-to-Image) 이미지라고 합니다. S2I 이미지를 사용하면 해당 코드를 실행할 준비가 된 기본 이미지 환경에 코드를 삽입할 수 있습니다.

OpenShift Dedicated 웹 UI에서 S2I 이미지를 직접 사용할 수 있습니다. 개발자 화면에서 **+추가** 보기로 이동하고 **개발자 카탈로그 타일에서 개발자 카탈로그**에서 사용 가능한 모든 서비스를 확인합니다.

그림 5.2. 특정 런타임이 필요한 앱에 맞는 S2I 기본 이미지 선택



5.2.3. 레지스트리 옵션

컨테이너 레지스트리에 컨테이너 이미지를 저장하면 다른 사용자와 공유하고 궁극적으로 실행되는 플랫폼에서 사용할 수 있게 됩니다. 무료 계정을 제공하는 대규모 공용 컨테이너 레지스트리 또는 추가 스토리지와 특수 기능을 제공하는 프리미엄 버전을 선택할 수 있습니다. 조직에 독점적이거나 다른 사용자와 선택적으로 공유할 수 있는 고유한 레지스트리도 설치할 수 있습니다.

Red Hat 이미지 및 인증된 파트너 이미지를 얻으려면 Red Hat Registry에서 그릴 수 있습니다. Red Hat Registry는 인증되지 않고 더 이상 사용되지 않는 **registry.access.redhat.com**과 인증이 필요한 **registry.redhat.io**의 두 위치로 표시됩니다. Container images section of the Red Hat Ecosystem Catalog에서 Red Hat Registry의 Red Hat 및 파트너 이미지에 관해 알아볼 수 있습니다. Red Hat 컨테이너 이미지를 나열하는 것 외에도 적용된 보안 업데이트를 기반으로 한 상태 점수를 포함하여 해당 이미지의 내용과 품질에 대한 광범위한 정보를 보여줍니다.

대규모 공개 레지스트리에는 Docker Hub 및 Quay.io가 있습니다. Quay.io 레지스트리는 Red Hat이 소유하고 관리합니다. OpenShift Dedicated에 사용되는 많은 구성 요소는 컨테이너 이미지 및 OpenShift Dedicated 자체를 배포하는 데 사용되는 Operator를 포함하여 Quay.io에 저장됩니다. Quay.io는 Helm 차트를 포함하여 다른 유형의 콘텐츠를 저장하는 수단도 제공합니다.

자체 프라이빗 컨테이너 레지스트리를 원하는 경우 OpenShift Dedicated 자체에는 OpenShift Dedicated로 설치되고 해당 클러스터에서 실행되는 개인 컨테이너 레지스트리가 포함되어 있습니다. Red Hat은 Red Hat Quay라는 개인용 Quay.io 레지스트리 버전도 제공합니다. Red Hat Quay에는 지역 복제, Git 빌드 트리거, Clair 이미지 스캔 및 기타 여러 기능이 포함되어 있습니다.

여기에 언급된 모든 레지스트리는 해당 레지스트리에서 이미지를 다운로드하기 위해 자격 증명이 필요할 수 있습니다. 이러한 인증 정보 중 일부는 OpenShift Dedicated에서 클러스터 전체에서 제공되며 다른 인증 정보는 개인에게 할당할 수 있습니다.

5.3. OPENSHIFT DEDICATED용 KUBERNETES 매니페스트 생성

컨테이너 이미지는 컨테이너화된 애플리케이션의 기본 구성 블록이지만 OpenShift Dedicated와 같은 Kubernetes 환경에서 해당 애플리케이션을 관리하고 배포하려면 추가 정보가 필요합니다. 일반적으로 이미지를 생성한 후의 단계는 다음과 같습니다.

- Kubernetes 매니페스트에서 작업하는 데 사용할 다양한 리소스 이해
- 실행 중인 애플리케이션의 종류 결정
- 지원 구성 요소 수집
- Git 리포지토리에 매니페스트를 생성하고 저장하여 소스 버전 관리 시스템에 저장하고, 감사하며, 추적하고, 다음 환경으로 승격 및 배포하며, 필요한 경우 이전 버전으로 롤백하고 다른 사용자와 공유할 수 있습니다.

5.3.1. Kubernetes 포드 및 서비스 정보

컨테이너 이미지는 docker가 있는 기본 단위이지만 Kubernetes가 작동하는 기본 단위는 pod라고 합니다. 포드는 애플리케이션을 빌드하는 다음 단계를 나타냅니다. 포드는 하나 이상의 컨테이너를 포함할 수 있습니다. 핵심은 포드가 배포, 스케일링 및 관리하는 단일 단위라는 것입니다.

확장성과 네임스페이스는 포드에 들어갈 내용을 결정할 때 고려해야 할 주요 항목입니다. 쉽게 배포하기 위해 포드에 컨테이너를 배포하고 자체 로깅 및 모니터링 컨테이너를 포드에 포함시킬 수 있습니다. 나중에 포드를 실행하고 추가 인스턴스를 스케일링해야 하는 경우 다른 컨테이너도 함께 확장됩니다. 네임스페이스의 경우 포드의 컨테이너는 동일한 네트워크 인터페이스, 공유 스토리지 볼륨 및 리소스 제한(예: 메모리 및 CPU)을 공유하므로, 포드의 콘텐츠를 단일 단위로 관리하기가 쉬워집니다. 포드의 컨테이너는 System V 세마포어 또는 POSIX 공유 메모리와 같은 표준 프로세스 간 통신을 사용하여 서로 통신할 수도 있습니다.

개별 포드는 Kubernetes에서 확장 가능한 단위를 나타내지만 서비스는 포드 세트를 함께 그룹화하여 부하 분산과 같은 작업을 완료할 수 있는 완전하고 안정적인 애플리케이션을 생성하는 수단을 제공합니다. 또한 서비스는 삭제할 때까지 동일한 IP 주소에서 계속 사용할 수 있기 때문에 포드보다 영구적입니다. 서비스가 사용 중이면 이름으로 요청되고 OpenShift Dedicated 클러스터는 해당 이름을 서비스를 구성하는 포드에 도달할 수 있는 IP 주소 및 포트로 확인됩니다.

컨테이너화된 애플리케이션은 특성상 실행되는 운영 체제뿐 아니라 사용자와 분리됩니다. Kubernetes 매니페스트의 일부는 컨테이너화된 애플리케이션과의 통신을 세밀하게 제어할 수 있는 네트워크 정책을 정의하여 애플리케이션을 내부 및 외부 네트워크에 노출하는 방법을 설명합니다. HTTP, HTTPS 및 기타 서비스에 대한 수신 요청을 클러스터 외부에서 클러스터 내부 서비스로 연결하려면 **Ingress** 리소스를 사용할 수 있습니다.

컨테이너에 서비스를 통해 제공될 수 있는 데이터베이스 스토리지 대신 온-디스크 스토리지가 필요한 경우 매니페스트에 볼륨을 추가하여 해당 스토리지를 포드에서 사용하게 만들 수 있습니다. 영구 볼륨(PV)을 생성하거나 **포드** 정의에 추가되는 볼륨을 동적으로 생성하도록 매니페스트를 구성할 수 있습니다.

애플리케이션을 구성하는 Pod 그룹을 정의한 후 **Deployment** 및 **DeploymentConfig** 오브젝트에서 해당 Pod를 정의할 수 있습니다.

5.3.2. 애플리케이션 유형

다음으로 애플리케이션 유형이 애플리케이션 실행 방법에 어떤 영향을 미치는지 고려하십시오.

Kubernetes는 다양한 종류의 애플리케이션에 적합한 다양한 유형의 워크로드를 정의합니다. 애플리케이션에 적합한 워크로드를 판별하려면 애플리케이션이 다음과 같은지 고려하십시오.

- 완전히 실행되어 완료되어야 합니다. 예를 들어 보고서를 생성하기 시작하고 보고서가 완료되면 종료되는 애플리케이션이 있습니다. 그러면 애플리케이션이 한 달 동안 다시 실행되지 않을 수 있습니다. 이러한 유형의 애플리케이션에 적합한 OpenShift Dedicated 오브젝트에는 **Job** 및 **CronJob** 오브젝트가 포함됩니다.

- 지속적으로 실행될 것으로 예상됩니다. 장기 실행 애플리케이션의 경우 배포를 작성할 수 있습니다.

- 가용성이 높아야 합니다. 애플리케이션에 고가용성이 필요한 경우 둘 이상의 인스턴스가 있도록 배포 크기를 조정해야 합니다. **Deployment** 또는 **DeploymentConfig** 오브젝트는 해당 유형의 애플리케이션에 대한 복제본 세트를 통합할 수 있습니다. 복제본 세트를 사용하면 포드가 여러 노드에서 실행되어 작업자가 중단된 경우에도 애플리케이션을 항상 사용할 수 있습니다.

- 모든 노드에서 실행해야 합니다. 일부 유형의 Kubernetes 애플리케이션은 모든 마스터 또는 작업자 노드의 클러스터 자체에서 실행되도록 설계되었습니다. DNS 및 모니터링 애플리케이션은 모든 노드에서 지속적으로 실행해야 하는 애플리케이션의 예입니다. 이 유형의 애플리케이션을 데몬 세트로 실행할 수 있습니다. 노드 레이블을 기반으로 노드 서브 세트에서 데몬 세트를 실행할 수도 있습니다.

- 라이프사이클 관리가 필요합니다. 다른 사용자가 사용할 수 있도록 애플리케이션을 전달하려면 Operator 생성을 고려하십시오. Operator를 사용하면 지능적으로 빌드할 수 있으므로 백업 및 업그레이드와 같은 작업을 자동으로 처리할 수 있습니다. OLM(Operator Lifecycle Manager)과 함께 클러스터 관리자는 선택된 네임스페이스에 Operator를 노출시키므로 클러스터의 사용자가 이를 실행할 수 있습니다.

- ID 또는 번호 지정 요구사항이 있습니다. 애플리케이션에는 ID 요구사항이나 번호 지정 요구사항이 있을 수 있습니다. 예를 들어, 정확히 3개의 애플리케이션 인스턴스를 실행하고 인스턴스 이름을 **0**, **1** 및 **2**로 지정해야 할 수 있습니다. 이 애플리케이션에는 상태 저장 세트가 적합합니다. 상태 저장 세트는 데이터베이스 및 zookeeper 클러스터와 같은 독립 스토리지가 필요한 애플리케이션에 가장 유용합니다.

5.3.3. 사용 가능한 지원 구성 요소

작성하는 애플리케이션에는 데이터베이스 또는 로깅 구성 요소와 같은 지원 구성 요소가 필요할 수 있습니다. 이러한 요구를 충족하기 위해 OpenShift Dedicated 웹 콘솔에서 사용할 수 있는 다음 카탈로그에서 필요한 구성 요소를 가져올 수 있습니다.

- 각 OpenShift Dedicated 4 클러스터에서 사용할 수 있는 OperatorHub. OperatorHub를 통해 Red Hat, 인증된 Red Hat 파트너 및 커뮤니티 멤버가 클러스터 운영자까지 Operator를 사용할 수 있습니다. 클러스터 운영자는 해당 Operator를 클러스터의 모든 네임스페이스 또는 선택된 네임스페이스에서 사용 가능하게 할 수 있으므로, 개발자가 애플리케이션을 사용하여 이를 시작하고 구성할 수 있습니다.

- 구성 요소를 설치한 후 구성 요소의 라이프사이클이 중요하지 않은 일회용 애플리케이션 유형에 유용한 템플릿. 템플릿은 최소한의 오버헤드로 Kubernetes 애플리케이션 개발을 쉽게 시작할 수 있는 방법을 제공합니다. 템플릿은 **Deployment**, **Service**, **Route** 또는 기타 오브젝트가 될 수 있는 리소스 정의 목록일 수 있습니다. 이름이나 리소스를 변경하려면 템플릿에서 이러한 값을 매개변수로 설정할 수 있습니다.

개발 팀의 특정 요구에 맞게 지원 Operator 및 템플릿을 구성한 다음 개발자가 작업하는 네임스페이스에서 사용 가능하게 만들 수 있습니다. 많은 사용자가 다른 모든 네임스페이스에서 액세스할 수 있으므로 **openshift** 네임스페이스에 공유 템플릿을 추가합니다.

5.3.4. 매니페스트 적용

Kubernetes 매니페스트를 사용하면 Kubernetes 애플리케이션을 구성하는 구성 요소를 더 완벽하게 이해할 수 있습니다. 이러한 매니페스트를 YAML 파일로 작성하고 **oc apply** 명령을 실행하는 등의 작업을 통해 클러스터에 적용하여 배포합니다.

5.3.5. 다음 단계

이 시점에서 컨테이너 개발 프로세스를 자동화하는 방법을 고려하십시오. 이상적으로 이미지를 빌드하고 레지스트리로 푸시하는 일종의 CI 파이프라인이 있습니다. 특히, GitOps 파이프라인은 컨테이너 개발을 애플리케이션 빌드에 필요한 소프트웨어를 저장하는 데 사용하는 Git 리포지토리와 통합합니다.

이 시점의 워크플로우는 다음과 같습니다.

- 1일 차: YAML을 작성합니다. 그런 다음 **oc apply** 명령을 실행하여 해당 YAML을 클러스터에 적용하고 작동하는지 테스트합니다.

- 2일 차: YAML 컨테이너 구성 파일을 자체 Git 리포지토리에 넣습니다. 여기에서 해당 앱을 설치하거나 개선하는 데 도움이 되는 사용자는 YAML을 풀다운하고 클러스터에 적용하여 앱을 실행할 수 있습니다.

- 3일 차: 애플리케이션에 맞는 Operator 작성을 고려하십시오.

5.4. OPERATOR용 개발

다른 사용자가 애플리케이션을 실행할 수 있게 하려면 애플리케이션을 Operator로 패키징하고 배포하는 것이 좋습니다. 앞에서 언급했듯이 Operator는 애플리케이션을 설치하는 즉시 애플리케이션 실행 작업이 완료되지 않음을 확인하는 라이프사이클 구성 요소를 애플리케이션에 추가합니다.

Operator로 애플리케이션을 생성할 때 애플리케이션을 실행하고 유지보수하는 방법에 관한 고유 지식을 빌드할 수 있습니다. 애플리케이션 업그레이드, 백업, 스케일링 또는 지속적인 상태 추적을 위한 기능을 빌드할 수 있습니다. 애플리케이션을 올바르게 구성하면 Operator 업데이트와 같은 유지보수 작업이 Operator의 사용자에게 표시되지 않고 자동으로 발생할 수 있습니다.

유용한 Operator의 예는 특정 시간에 자동으로 데이터를 백업하도록 설정된 Operator입니다. Operator가 정해진 시간에 애플리케이션의 백업을 관리하게 하면 시스템 관리자가 이를 기억하는 데 쏟는 시간이 절약됩니다.

데이터 백업 또는 인증서 교체와 같이 일반적으로 수동으로 완료되는 모든 애플리케이션 유지보수는 Operator를 통해 자동으로 완료할 수 있습니다.

6장. 승인 플러그인

승인 플러그인은 OpenShift Dedicated의 기능을 규제하는 데 사용됩니다.

6.1. 승인 플러그인 정보

승인 플러그인은 마스터 API에 대한 요청을 가로채어 리소스 요청을 검증합니다. 요청이 인증 및 승인되면 승인 플러그인에서 관련 정책이 준수되도록 합니다. 예를 들어 보안 정책, 리소스 제한 또는 구성 요구사항을 적용하는 데 일반적으로 사용됩니다.

승인 플러그인은 순서대로 승인 체인으로 실행됩니다. 순서의 승인 플러그인이 요청을 거부하면 전체 체인이 중단되고 오류가 반환됩니다.

OpenShift Dedicated에는 각 리소스 유형에 대해 기본 승인 플러그인 세트가 활성화되어 있습니다. 클러스터가 올바르게 작동하는 데 필요합니다. 승인 플러그인은 책임이 없는 리소스는 무시합니다.

기본 설정 외에도 사용자 정의 웹 후크 서버를 호출하는 웹 후크 승인 플러그인을 통해 승인 체인을 동적으로 확장할 수 있습니다. 웹 후크 승인 플러그인에는 변경 승인 플러그인과 검증 승인 플러그인의 두 가지 유형이 있습니다. 변경 승인 플러그인이 먼저 실행되며 리소스를 수정하고 요청을 검증할 수 있습니다. 검증 승인 플러그인은 변경 승인 플러그인에서 트리거한 수정도 검증할 수 있도록 요청을 검증하고 변경 승인 플러그인 후에 실행됩니다.

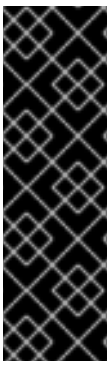
변경 승인 플러그인을 통해 웹 후크 서버를 호출하면 대상 오브젝트와 관련된 리소스에 부작용이 발생할 수 있습니다. 이러한 상황에서는 최종 결과가 예상한 대로인지 확인하는 단계를 수행해야 합니다.

주의

동적 승인은 클러스터 컨트롤 플레인 작업에 영향을 주기 때문에 주의해서 사용해야 합니다. OpenShift Dedicated 4에서 웹 후크 승인 플러그인을 통해 웹 후크 서버를 호출할 때 문서를 완전히 읽고 변경 부작용을 테스트했는지 확인하십시오. 요청이 전체 승인 체인을 통과하지 않는 경우 변경 이전에 리소스를 원래 상태로 복원하는 단계를 포함합니다.

6.2. 기본 승인 플러그인

OpenShift Dedicated 4에서는 기본 검증 및 승인 플러그인이 활성화됩니다. 이러한 기본 플러그인은 수신 정책, 클러스터 리소스 제한 덮어쓰기 및 할당량 정책과 같은 기본 컨트롤 플레인 기능에 기여합니다.



중요

기본 프로젝트에서 워크로드를 실행하거나 기본 프로젝트에 대한 액세스를 공유하지 마세요. 기본 프로젝트는 핵심 클러스터 구성 요소를 실행하기 위해 예약되어 있습니다.

다음 기본 프로젝트는 높은 권한이 있는 것으로 간주됩니다. **default**, **kube-public**, **kube-system**, **openshift**, **openshift-infra**, **openshift-node** 및 **openshift.io/run-level** 레이블이 **0** 또는 **1** 로 설정된 기타 시스템 생성 프로젝트입니다. Pod 보안 승인, 보안 컨텍스트 제약 조건, 클러스터 리소스 할당량 및 이미지 참조 확인과 같은 승인 플러그인에 의존하는 기능은 높은 권한 있는 프로젝트에서 작동하지 않습니다.

다음 목록에는 기본 승인 플러그인이 포함되어 있습니다.

예 6.1. 검증 승인 플러그인

- **LimitRanger**
- **ServiceAccount**
- **PodNodeSelector**
- **Priority**
- **PodTolerationRestriction**
- **OwnerReferencesPermissionEnforcement**
- **PersistentVolumeClaimResize**
- **RuntimeClass**
- **CertificateApproval**
- **CertificateSigning**
- **CertificateSubjectRestriction**
- **autoscaling.openshift.io/ManagementCPUsOverride**
- **authorization.openshift.io/RestrictSubjectBindings**
- **scheduling.openshift.io/OriginPodNodeEnvironment**
- **network.openshift.io/ExternalIPRanger**
- **network.openshift.io/RestrictedEndpointsAdmission**
- **image.openshift.io/ImagePolicy**
- **security.openshift.io/SecurityContextConstraint**
- **security.openshift.io/SCCExecRestrictions**
- **route.openshift.io/IngressAdmission**
- **config.openshift.io/ValidateAPIServer**
- **config.openshift.io/ValidateAuthentication**
- **config.openshift.io/ValidateFeatureGate**
- **config.openshift.io/ValidateConsole**
- **operator.openshift.io/ValidateDNS**
- **config.openshift.io/ValidateImage**
- **config.openshift.io/ValidateOAuth**

- **config.openshift.io/ValidateProject**
- **config.openshift.io/DenyDeleteClusterConfiguration**
- **config.openshift.io/ValidateScheduler**
- **quota.openshift.io/ValidateClusterResourceQuota**
- **security.openshift.io/ValidateSecurityContextConstraints**
- **authorization.openshift.io/ValidateRoleBindingRestriction**
- **config.openshift.io/ValidateNetwork**
- **operator.openshift.io/ValidateKubeControllerManager**
- **ValidatingAdmissionWebhook**
- **resourceQuota**
- **quota.openshift.io/ClusterResourceQuota**

예 6.2. 변경 승인 플러그인

- **NamespaceLifecycle**
- **LimitRanger**
- **ServiceAccount**
- **NodeRestriction**
- **TaintNodesByCondition**
- **PodNodeSelector**
- **Priority**
- **DefaultTolerationSeconds**
- **PodTolerationRestriction**
- **DefaultStorageClass**
- **StorageObjectInUseProtection**
- **RuntimeClass**
- **DefaultIngressClass**
- **autoscaling.openshift.io/ManagementCPUsOverride**
- **scheduling.openshift.io/OriginPodNodeEnvironment**
- **image.openshift.io/ImagePolicy**
- **security.openshift.io/SecurityContextConstraint**

- **security.openshift.io/DefaultSecurityContextConstraints**
- **MutatingAdmissionWebhook**

6.3. WEBHOOK 승인 플러그인

OpenShift Dedicated 기본 승인 플러그인 외에도 웹 후크 서버를 호출하는 웹 후크 승인 플러그인을 통해 동적 승인을 구현하여 승인 체인의 기능을 확장할 수 있습니다. 웹 후크 서버는 정의된 엔드포인트에서 HTTP를 통해 호출됩니다.

OpenShift Dedicated에는 두 가지 유형의 웹 후크 승인 플러그인이 있습니다.

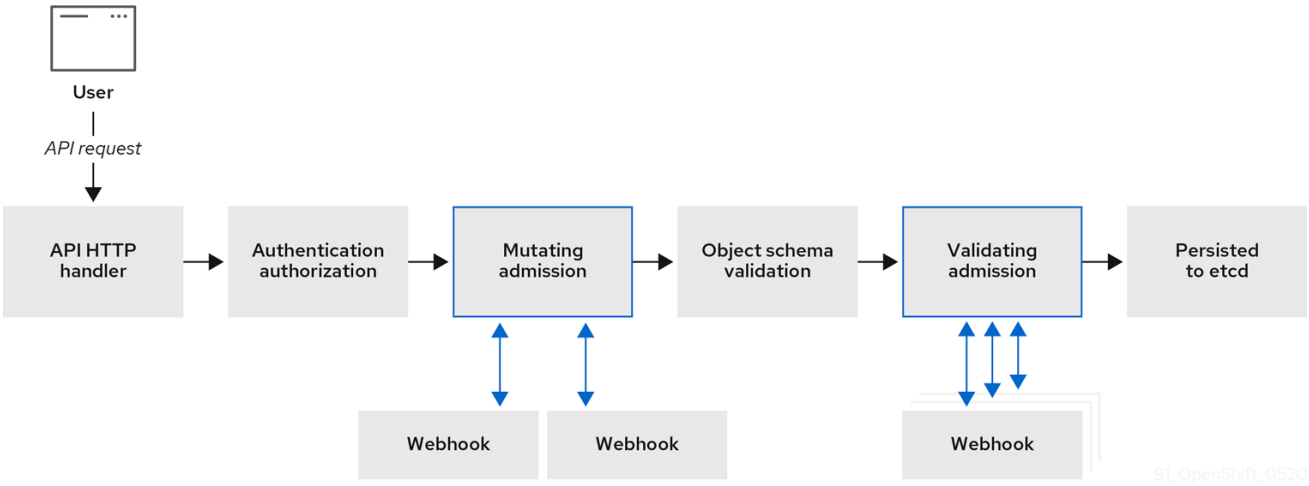
- 승인 프로세스 중에 *변경 승인 플러그인* 은 선호도 레이블 삽입과 같은 작업을 수행할 수 있습니다.
- 승인 프로세스가 끝나면 *검증 승인 플러그인을* 사용하여 오브젝트가 올바르게 구성되었는지 확인할 수 있습니다(예: 선호도 레이블이 예상대로 있는지 확인). 검증이 통과되면 OpenShift Dedicated는 구성된 대로 오브젝트를 예약합니다.

API 요청이 입력되면 변경 또는 검증 승인 플러그인이 구성의 외부 웹 후크 목록을 사용하여 병렬로 호출합니다.

- 모든 웹 후크가 요청을 승인하면 승인 체인이 계속됩니다.
- 웹 후크 중 하나라도 요청을 거부하면 승인 요청이 거부됩니다. 그 이유는 첫 번째 거부를 기반으로 합니다.
- 둘 이상의 웹 후크가 승인 요청을 거부하면 첫 번째 거부 이유만 사용자에게 반환됩니다.
- 웹 후크를 호출할 때 오류가 발생하면 오류 정책 세트에 따라 요청이 거부되거나 웹 후크가 무시됩니다. 오류 정책이 **Ignore**로 설정되면 실패 시 요청이 무조건 수락됩니다. 정책이 **Fail**로 설정되면 실패한 요청이 거부됩니다. **Ignore**를 사용하면 모든 클라이언트에 대해 예기치 않은 동작이 발생할 수 있습니다.

다음 다이어그램은 여러 웹 후크 서버가 호출되는 순차적 승인 체인 프로세스를 보여줍니다.

그림 6.1. 변경 및 검증 승인 플러그인을 사용하는 API 승인 체인



웹 후크 승인 플러그인 사용 사례의 예는 모든 Pod에 공통 레이블 세트가 있어야 하는 경우입니다. 이 예

에서 변경 승인 플러그인은 레이블을 삽입할 수 있으며 검증 승인 플러그인은 레이블이 예상대로 있는지 확인할 수 있습니다. OpenShift Dedicated는 나중에 필수 라벨이 포함된 Pod를 스케줄링하고 그렇지 않은 레이블을 거부합니다.

일반적인 웹 후크 승인 플러그인 사용 사례는 다음과 같습니다.

- 네임스페이스 예약.
- SR-IOV 네트워크 장치 플러그인에서 관리하는 사용자 정의 네트워크 리소스 제한.
- 포드 우선 순위 클래스 검증.

참고

OpenShift Dedicated의 최대 기본 Webhook 시간 초과 값은 13초이며 변경할 수 없습니다.

6.4. 웹 후크 승인 플러그인의 유형

클러스터 관리자는 API 서버 승인 체인의 변경 승인 플러그인 또는 검증 승인 플러그인을 통해 웹 후크 서버를 호출할 수 있습니다.

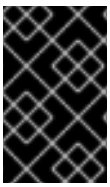
6.4.1. 변경 승인 플러그인

변경 승인 플러그인은 승인 프로세스의 변경 단계에서 호출되므로 리소스 콘텐츠가 지속되기 전에 수정할 수 있습니다. 변경 승인 플러그인을 통해 호출할 수 있는 웹 후크의 한 예는 네임스페이스의 주석을 사용하여 라벨 선택기를 찾고 Pod 사양에 추가하는 Pod 노드 선택기 기능입니다.

샘플 변경 승인 플러그인 구성

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: MutatingWebhookConfiguration (1)
metadata:
  name: <webhook_name> (2)
webhooks:
- name: <webhook_name> (3)
  clientConfig: (4)
    service:
      namespace: default (5)
      name: kubernetes (6)
      path: <webhook_url> (7)
    caBundle: <ca_signing_certificate> (8)
  rules: (9)
  - operations: (10)
    - <operation>
    apiGroups:
    - ""
    apiVersions:
    - "*"
    resources:
    - <resource>
  failurePolicy: <policy> (11)
  sideEffects: None
```

1. 변경 승인 플러그인 구성을 지정합니다.
2. **MutatingWebhookConfiguration** 오브젝트의 이름입니다. **<webhook_name>**을 적절한 값으로 바꿉니다.
3. 호출할 웹 후크의 이름입니다. **<webhook_name>**을 적절한 값으로 바꿉니다.
4. 웹 후크 서버에 연결하고 신뢰하고 데이터를 전송하는 방법에 대한 정보입니다.
5. 프런트 엔드 서비스가 생성되는 네임스페이스입니다.
6. 프론트 엔드 서비스의 이름입니다.
7. 승인 요청에 사용되는 웹 후크 URL입니다. **<webhook_url>**을 적절한 값으로 바꿉니다.
8. 웹 후크 서버가 사용하는 서버 인증서에 서명하는 PEM 인코딩된 CA 인증서입니다. **<ca_signing_certificate>**를 base64 형식의 적절한 인증서로 바꿉니다.
9. API 서버가 이 웹 후크 승인 플러그인을 사용해야 하는 시기를 정의하는 규칙입니다.
10. API 서버가 이 웹 후크 승인 플러그인을 호출하도록 트리거하는 하나 이상의 작업입니다. 가능한 값은 **create**, **update**, **delete** 또는 **connect**입니다. **<operation>** 및 **<resource>**를 적절한 값으로 바꿉니다.
11. 웹 후크 서버를 사용할 수 없는 경우 정책 진행 방법을 지정합니다. **<policy>**를 **Ignore**(실패한 경우 요청을 무조건 수락) 또는 **Fail**(실패한 요청 거부)로 바꿉니다. **Ignore**를 사용하면 모든 클라이언트에 대해 예기치 않은 동작이 발생할 수 있습니다.



중요

OpenShift Dedicated 4에서는 변경 승인 플러그인을 통해 사용자 또는 제어 루프가 생성한 오브젝트는 특히 초기 요청에 설정된 값을 덮어쓰는 경우 예기치 않은 결과를 반환할 수 있습니다. 이는 권장되지 않습니다.

6.4.2. 검증 승인 플러그인

승인 프로세스의 검증 단계에서 검증 승인 플러그인이 호출됩니다. 이 단계에서는 특정 API 리소스에 대한 변형을 적용하여 리소스가 다시 변경되지 않게 합니다. 포드 노드 선택기는 검증 승인 플러그인에 의해 호출되는 웹 후크의 예이며, 네임스페이스의 노드 선택기 제한에 따라 모든 **nodeSelector** 필드가 제한되도록 합니다.

샘플 검증 플러그인 구성

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingWebhookConfiguration (1)
metadata:
  name: <webhook_name> (2)
webhooks:
- name: <webhook_name> (3)
  clientConfig: (4)
    service:
      namespace: default (5)
      name: kubernetes (6)
      path: <webhook_url> (7)
```

```
  caBundle: <ca_signing_certificate> 8
rules: 9
- operations: 10
  - <operation>
  apiGroups:
  - ""
  apiVersions:
  - "*"
  resources:
  - <resource>
failurePolicy: <policy> 11
sideEffects: Unknown
```

1. 검증 승인 플러그인 구성을 지정합니다.
2. **ValidatingWebhookConfiguration** 오브젝트의 이름입니다. **<webhook_name>**을 적절한 값으로 바꿉니다.
3. 호출할 웹 후크의 이름입니다. **<webhook_name>**을 적절한 값으로 바꿉니다.
4. 웹 후크 서버에 연결하고 신뢰하고 데이터를 전송하는 방법에 대한 정보입니다.
5. 프런트 엔드 서비스가 생성되는 네임스페이스입니다.
6. 프론트 엔드 서비스의 이름입니다.
7. 승인 요청에 사용되는 웹 후크 URL입니다. **<webhook_url>**을 적절한 값으로 바꿉니다.
8. 웹 후크 서버가 사용하는 서버 인증서에 서명하는 PEM 인코딩된 CA 인증서입니다. **<ca_signing_certificate>**를 base64 형식의 적절한 인증서로 바꿉니다.
9. API 서버가 이 웹 후크 승인 플러그인을 사용해야 하는 시기를 정의하는 규칙입니다.
10. API 서버가 이 웹 후크 승인 플러그인을 호출하도록 트리거하는 하나 이상의 작업입니다. 가능한 값은 **create**, **update**, **delete** 또는 **connect**입니다. **<operation>** 및 **<resource>**를 적절한 값으로 바꿉니다.
11. 웹 후크 서버를 사용할 수 없는 경우 정책 진행 방법을 지정합니다. **<policy>**를 **Ignore**(실패한 경우 요청을 무조건 수락) 또는 **Fail**(실패한 요청 거부)로 바꿉니다. **Ignore**를 사용하면 모든 클라이언트에 대해 예기치 않은 동작이 발생할 수 있습니다.

6.5. 추가 리소스

- Pod 우선순위 이름