OpenShift Container Platform 4.13

시작하기

OpenShift Container Platform 시작하기

Last Updated: 2024-07-25

OpenShift Container Platform 4.13 시작하기

OpenShift Container Platform 시작하기

법적 공지

초록

이 문서에서는 OpenShift Container Platform에서 시작하는 데 도움이 되는 정보를 제공합니다. 여기에는 Kubernetes 및 OpenShift Container Platform에 있는 일반 용어의 정의가 포함됩니다. 여기에는 OpenShift Container Platform 웹 콘솔 및 명령줄 인터페이스를 사용하여 애플리케이션을 생성하고 빌드하는 방법도 포함되어 있습니다.

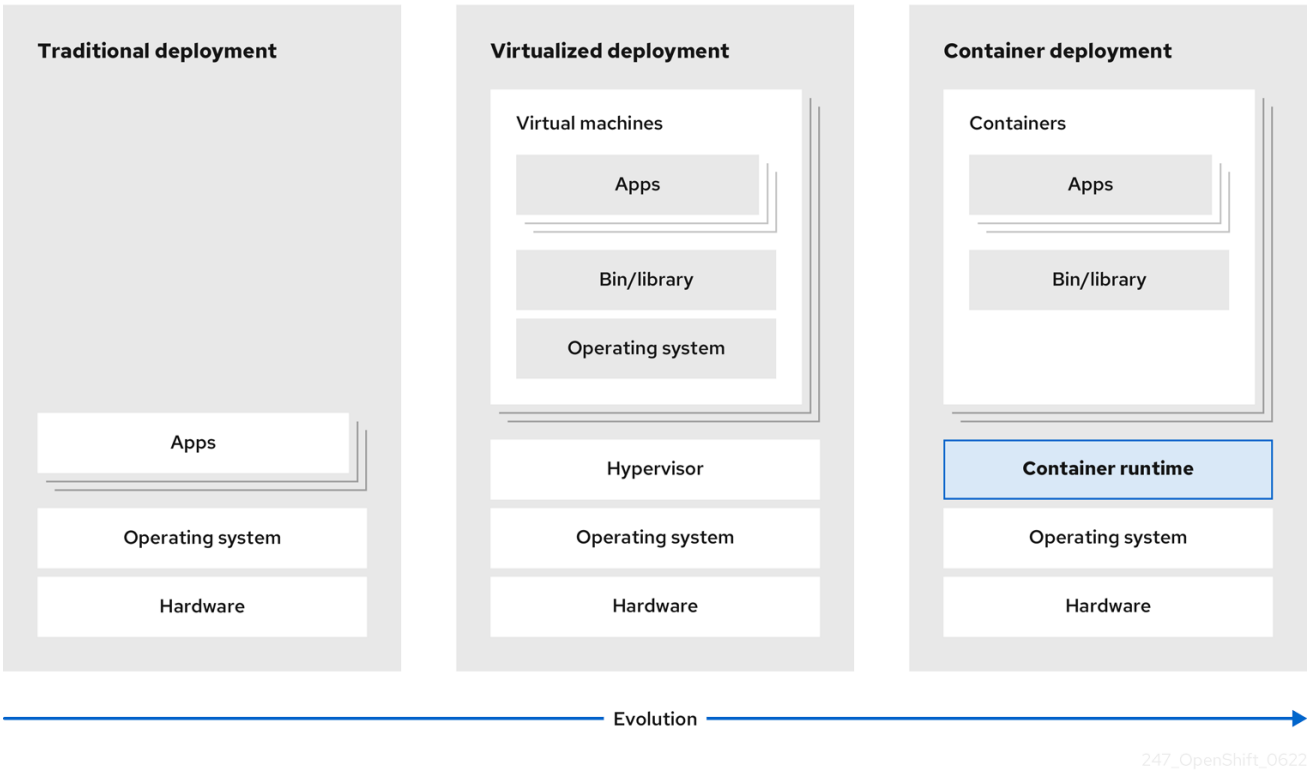
차례

1장. KUBERNETES 개요

Kubernetes는 Google에서 개발한 오픈 소스 컨테이너 오케스트레이션 툴입니다. Kubernetes를 사용하여 컨테이너 기반 워크로드를 실행하고 관리할 수 있습니다. 가장 일반적인 Kubernetes 사용 사례는 상호 연결된 마이크로 서비스 배열을 배포하여 클라우드 네이티브 방식으로 애플리케이션을 구축하는 것입니다. 온프레미스, 퍼블릭, 프라이빗 또는 하이브리드 클라우드에 걸쳐 호스트를 확장할 수 있는 Kubernetes 클러스터를 생성할 수 있습니다.

일반적으로 애플리케이션은 단일 운영 체제 위에 배포되었습니다. 가상화를 사용하면 물리적 호스트를 여러 가상 호스트로 분할할 수 있습니다. 공유 리소스의 가상 인스턴스에서 작업하는 것은 효율성 및 확장성에 적합하지 않습니다. VM(가상 머신)은 실제 머신만큼 많은 리소스를 사용하므로 CPU, RAM, 스토리지와 같은 VM에 리소스를 제공하면 비용이 많이 듭니다. 또한 공유 리소스에서 가상 인스턴스 사용량으로 인해 애플리케이션이 성능이 저하될 수 있습니다.

그림 1.1. 클래식 배포를 위한 컨테이너 기술의 진화



이 문제를 해결하려면 컨테이너화된 환경에서 애플리케이션을 분리하는 컨테이너화 기술을 사용할 수 있습니다. VM과 유사하게 컨테이너에는 자체 파일 시스템, vCPU, 메모리, 프로세스 공간, 종속성 등이 있습니다. 컨테이너는 기본 인프라와 분리되며 클라우드 및 OS 배포 전반에서 이식 가능합니다. 컨테이너는 완전한 기능을 갖춘 OS보다 훨씬 가벼우며 운영 체제 커널에서 실행되는 경량의 격리된 프로세스입니다. VM은 부팅 속도가 느리고 물리적 하드웨어에 대한 추상화입니다. 하이퍼바이저를 통해 단일 시스템에서 VM을 실행합니다.

Kubernetes를 사용하여 다음 작업을 수행할 수 있습니다.

- 리소스 공유
- 여러 호스트에서 컨테이너 오케스트레이션
- 새 하드웨어 구성 설치
- 상태 점검 실행 및 자동 복구 애플리케이션 실행

- 컨테이너화된 애플리케이션 스케일링

1.1. KUBERNETES 구성 요소

표 1.1. Kubernetes 구성 요소

구성 요소	목적
kube-proxy	클러스터의 모든 노드에서 실행되며 Kubernetes 리소스 간의 네트워크 트래픽을 유지 관리합니다.
kube-controller-manager	클러스터 상태를 관리합니다.
kube-scheduler	노드에 pod를 할당합니다.
etcd	클러스터 데이터를 저장합니다.
kube-apiserver	API 오브젝트의 데이터를 검증하고 구성합니다.
kubelet	노드에서 실행되며 컨테이너 매니페스트를 읽습니다. 정의된 컨테이너가 시작되어 실행 중인지 확인합니다.
kubectl	워크로드를 실행하려는 방법을 정의할 수 있습니다. **kubectl** 명령을 사용하여 **kube-apiserver**와 상호 작용합니다.
노드	노드는 Kubernetes 클러스터의 물리적 시스템 또는 VM입니다. 컨트롤 플레인은 모든 노드를 관리하고 Kubernetes 클러스터의 노드에서 Pod를 예약합니다.
컨테이너 런타임	컨테이너 런타임은 호스트 운영 체제에서 컨테이너를 실행합니다. Pod를 노드에서 실행할 수 있도록 각 노드에 컨테이너 런타임을 설치해야 합니다.
영구 스토리지	장치가 종료된 후에도 데이터를 저장합니다. Kubernetes는 영구 볼륨을 사용하여 애플리케이션 데이터를 저장합니다.
container-registry	컨테이너 이미지를 저장하고 액세스합니다.
Pod	Pod는 Kubernetes에서 가장 작은 논리 단위입니다. Pod에는 작업자 노드에서 실행할 하나 이상의 컨테이너가 포함되어 있습니다.

1.2. KUBERNETES 리소스

사용자 정의 리소스는 Kubernetes API의 확장입니다. 사용자 정의 리소스를 사용하여 Kubernetes 클러스터를 사용자 지정할 수 있습니다. Operator는 사용자 정의 리소스를 사용하여 애플리케이션 및 해당 구성 요소를 관리하는 소프트웨어 확장입니다. Kubernetes는 클러스터 리소스를 처리하는 동안 고정된 결과를 원하는 경우 선언적 모델을 사용합니다. Operator를 사용하여 Kubernetes는 선언적 방식으로 상태를 정의합니다. 필수 명령을 사용하여 Kubernetes 클러스터 리소스를 수정할 수 있습니다. Operator는 원하는 리소스 상태를 리소스의 실제 상태와 지속적으로 비교하고 원하는 상태와 일치하도록 현실을 구현하기 위한 조치를 취하는 제어 루프 역할을 합니다.

그림 1.2. Kubernetes 클러스터 개요

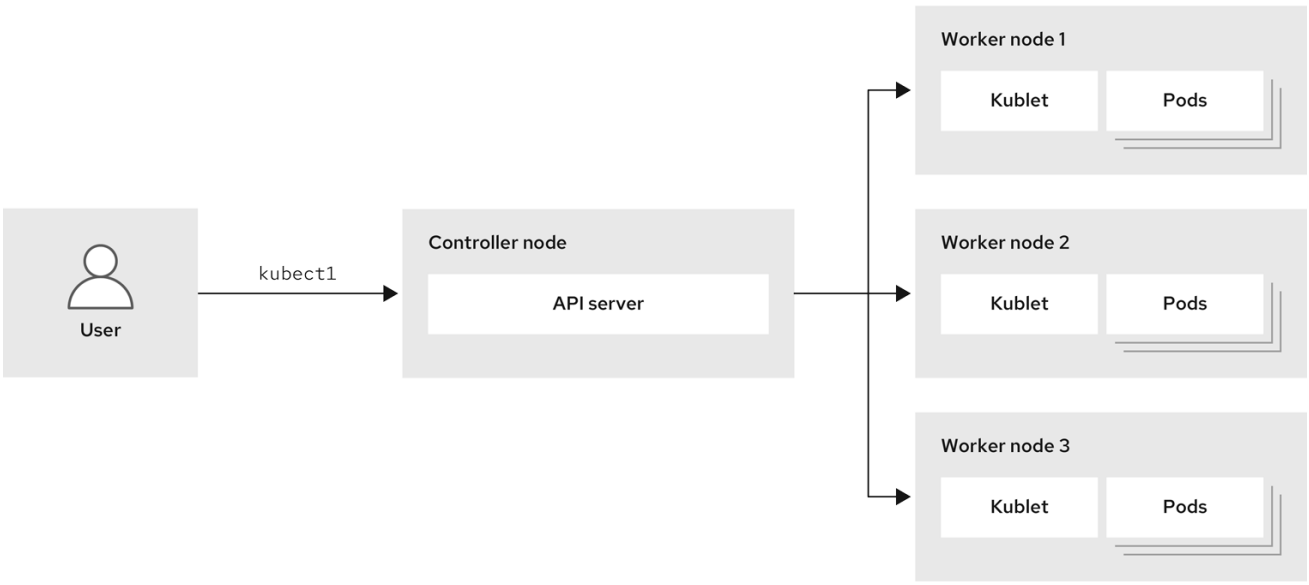
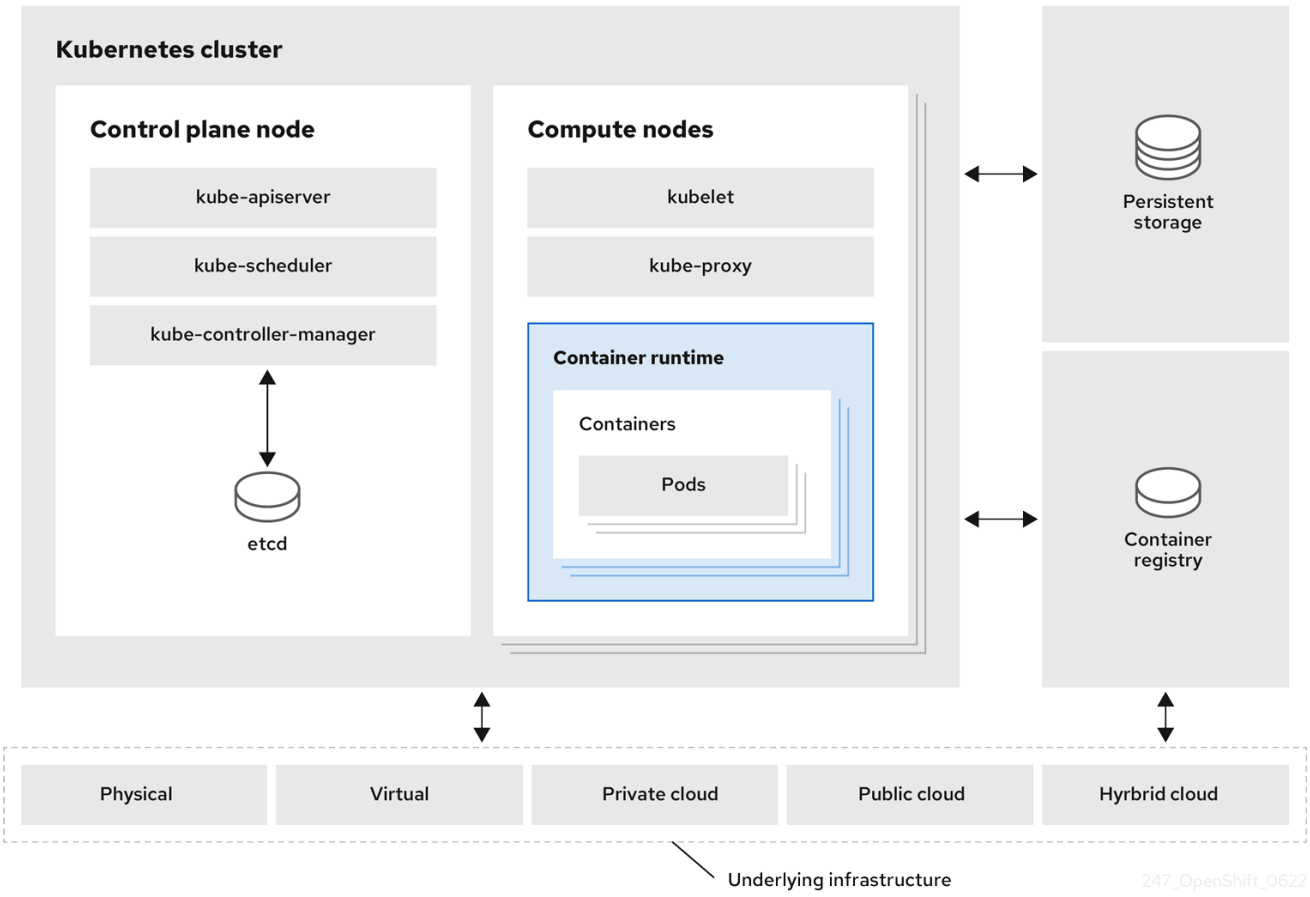


표 1.2. Kubernetes 리소스

리소스	목적
Service	Kubernetes는 서비스를 사용하여 pod 세트에 실행 중인 애플리케이션을 노출합니다.
ReplicaSets	Kubernetes는 **ReplicaSets**를 사용하여 일정한 pod 번호를 유지합니다.
Deployment	애플리케이션의 라이프사이클을 유지보수하는 리소스 오브젝트입니다.

Kubernetes는 OpenShift Container Platform의 핵심 구성 요소입니다. OpenShift Container Platform을 사용하여 컨테이너화된 애플리케이션을 개발하고 실행할 수 있습니다. Kubernetes에 기반을 둔 OpenShift Container Platform은 대규모 통신, 스트리밍 비디오, 게임, 뱅킹 및 기타 애플리케이션의 엔진 역할을 하는 동일한 기술을 통합합니다. OpenShift Container Platform을 사용하여 컨테이너화된 애플리케이션을 단일 클라우드 이상으로 온프레미스 및 다중 클라우드 환경으로 확장할 수 있습니다.

그림 1.3. Kubernetes 아키텍처



클러스터는 클라우드 환경에서 여러 노드로 구성된 단일 계산 단위입니다. Kubernetes 클러스터에는 컨트롤 플레인 및 작업자 노드가 포함됩니다. 다양한 머신 및 환경에서 Kubernetes 컨테이너를 실행할 수 있습니다. 컨트롤 플레인 노드는 클러스터 상태를 제어하고 유지보수합니다. 작업자 노드를 사용하여 Kubernetes 애플리케이션을 실행할 수 있습니다. Kubernetes 네임스페이스를 사용하여 클러스터의 클러스터 리소스를 구분할 수 있습니다. 네임스페이스 범위는 배포, 서비스 및 Pod와 같은 리소스 오브젝트에 적용할 수 있습니다. 스토리지 클래스, 노드, 영구 볼륨과 같은 클러스터 전체 리소스 오브젝트에는 네임스페이스를 사용할 수 없습니다.

1.3. KUBERNETES 개념 가이드라인

OpenShift Container Platform을 시작하기 전에 Kubernetes의 개념적 가이드라인을 고려하십시오.

- 하나 이상의 작업자 노드에서 시작하여 컨테이너 워크로드를 실행합니다.
- 하나 이상의 컨트롤 플레인 노드에서 해당 워크로드의 배포를 관리합니다.
- 컨테이너를 포드라는 배포 단위로 래핑합니다. Pod를 사용하면 컨테이너에 추가 메타데이터를 제공하고 단일 배포 엔티티에서 여러 컨테이너를 그룹화할 수 있습니다.
- 특별한 종류의 자산을 생성합니다. 예를 들어 서비스는 pod 세트와 Pod에 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.

OpenShift Container Platform 클러스터로의 API는 100% Kubernetes입니다. 다른 Kubernetes에서 실행되고 OpenShift Container Platform에서 실행되는 컨테이너 간에는 변경되지 않습니다. 애플리케이션에

대한 변경 사항이 없습니다. OpenShift Container Platform은 Kubernetes에 엔터프라이즈급 개선사항을 제공하기 위해 추가 값 기능을 제공합니다. OpenShift Container Platform CLI 툴(**oc**)은 **kubectl**과 호환됩니다. Kubernetes API는 OpenShift Container Platform 내에서 100% 액세스할 수 있지만 **kubectl** 명령줄에는 사용자 친화적인 기능이 많이 없습니다. OpenShift Container Platform에서는 **oc**와 같은 기능 및 명령줄 툴 세트를 제공합니다. Kubernetes는 애플리케이션 관리에 뛰어나지만 플랫폼 수준 요구사항 또는 배포 프로세스를 지정하거나 관리하지는 않습니다. 강력하고 유연한 플랫폼 관리 툴 및 프로세스는 OpenShift Container Platform에서 제공하는 중요한 이점입니다. 컨테이너화 플랫폼에 인증, 네트워킹, 보안, 모니터링 및 로그 관리를 추가해야 합니다.

2장. OPENSHIFT CONTAINER PLATFORM 개요

OpenShift Container Platform은 클라우드 기반 Kubernetes 컨테이너 플랫폼입니다. OpenShift Container Platform의 기초는 Kubernetes를 기반으로 하므로 동일한 기술을 공유합니다. 애플리케이션 및 애플리케이션을 지원하는 데이터센터를 몇 대의 머신 및 애플리케이션에서 수백만 클라이언트에 서비스를 제공하는 수천 대의 컴퓨터로 확장 가능하도록 설계되었습니다.

OpenShift Container Platform을 사용하면 다음을 수행할 수 있습니다.

- 개발자와 IT 조직에 안전하고 확장 가능한 리소스에 애플리케이션을 배포하는 데 사용할 수 있는 클라우드 애플리케이션 플랫폼을 제공합니다.
- 최소한의 구성 및 관리 오버헤드가 필요합니다.
- Kubernetes 플랫폼을 고객 데이터 센터 및 클라우드에 가져옵니다.
- 보안, 개인 정보 보호, 컴플라이언스 및 거버넌스 요구 사항을 충족합니다.

Kubernetes에 기반을 둔 OpenShift Container Platform은 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. OPENSHIFT CONTAINER PLATFORM의 일반 용어집

이 용어집은 일반적인 Kubernetes 및 OpenShift Container Platform 용어를 정의합니다.

Kubernetes

Kubernetes는 컨테이너화된 애플리케이션의 배포, 확장 및 관리를 자동화하기 위한 오픈 소스 컨테이너 오케스트레이션 엔진입니다.

컨테이너

컨테이너는 작업자 노드의 OCI 호환 컨테이너에서 실행되는 애플리케이션 인스턴스 및 구성 요소입니다. 컨테이너는 OCI(Open Container Initiative) 호환 이미지의 런타임입니다. 이미지는 바이너리 애플리케이션입니다. 작업자 노드는 여러 컨테이너를 실행할 수 있습니다. 노드 용량은 클라우드, 하드웨어 또는 가상화 여부에 관계없이 기본 리소스의 메모리 및 CPU 기능과 관련이 있습니다.

Pod

Pod는 하나의 호스트에 함께 배포되는 하나 이상의 컨테이너입니다. 볼륨 및 IP 주소와 같은 공유 리소스가 있는 배치된 컨테이너 그룹으로 구성됩니다. Pod는 정의, 배포 및 관리되는 최소 컴퓨팅 단위이기도 합니다.
OpenShift Container Platform에서 Pod는 개별 애플리케이션 컨테이너를 배포 가능한 가장 작은 단위로 교체합니다.

Pod는 OpenShift Container Platform의 오케스트레이션된 단위입니다. OpenShift Container Platform은 동일한 노드의 Pod에 있는 모든 컨테이너를 예약하고 실행합니다. 복잡한 애플리케이션은 각각 자체 컨테이너가 있는 여러 Pod로 구성됩니다. 외부적으로 상호 작용하고 OpenShift Container Platform 환경 내부와도 상호 작용합니다.

복제본 세트 및 복제 컨트롤러

Kubernetes 복제본 세트 및 OpenShift Container Platform 복제 컨트롤러를 모두 사용할 수 있습니다. 이 구성 요소의 작업은 지정된 수의 Pod 복제본이 항상 실행 중인지 확인하는 것입니다. Pod가 종료되거나 삭제되면 복제 세트 또는 복제 컨트롤러가 더 많이 시작됩니다. 필요한 것보다 많은 Pod가 실행 중인 경우 복제 세트는 지정된 복제 수와 일치하도록 필요한 수만큼 삭제됩니다.

배포 및 배포 구성

OpenShift Container Platform은 Kubernetes **Deployment** 오브젝트와 OpenShift Container Platform **DeploymentConfigs**오브젝트를 모두 구현합니다. 사용자는 둘 중 하나를 선택할 수 있습니다.
Deployment 오브젝트는 애플리케이션을 pod로 롤아웃하는 방법을 제어합니다. 레지스트리에서 가져올 컨테이너 이미지의 이름을 확인하고 노드에 pod로 배포됩니다. 배포할 pod 복제본 수를 설정하고 프로세스를 관리할 복제본 세트를 생성합니다. 표시된 레이블은 Pod를 배포할 노드를 스케줄러에 지시합니다. 레이블 세트는 복제 세트에서 인스턴스화하는 pod 정의에 포함되어 있습니다.

Deployment 오브젝트는 **Deployment** 오브젝트 버전 및 허용 가능한 애플리케이션 가용성 관리를 위한 다양한 롤아웃 전략을 기반으로 작업자 노드에 배포된 Pod를 업데이트할 수 있습니다. OpenShift Container Platform **DeploymentConfig** 오브젝트는 새 컨테이너 이미지 버전을 사용할 수 있는 경우 새로운 버전의 **Deployment** 오브젝트를 자동으로 생성할 수 있는 변경 트리거의 추가 기능을 추가합니다.

서비스

서비스는 pod의 논리 세트와 액세스 정책을 정의합니다. Pod가 생성되고 삭제될 때 사용할 다른 애플리케이션의 영구 내부 IP 주소 및 호스트 이름을 제공합니다.
서비스 계층은 애플리케이션 구성 요소를 연결합니다. 예를 들어 프런트 엔드 웹 서비스는 서비스와 통신하여 데이터베이스 인스턴스에 연결합니다. 서비스를 통해 애플리케이션 구성 요소 간에 간단한 내부 로드 밸런싱이 가능합니다. OpenShift Container Platform은 쉽게 검색할 수 있도록 실행 중인 컨테이너에 서비스 정보를 자동으로 삽입합니다.

경로

경로는 www.example.com과 같이 외부에 연결할 수 있는 호스트 이름을 지정하여 서비스를 노출하는 방법입니다. 각 경로는 경로 이름, 서비스 선택기 및 보안 구성(선택 사항)으로 구성됩니다. 라우터는 정의된 경로와 서비스에 의해 식별된 엔드포인트를 사용하여 외부 클라이언트가 애플리케이션에 도달할 수 있는 이름을 제공할 수 있습니다. 완전한 다중 계층 애플리케이션을 쉽게 배포할 수는 있지만 OpenShift Container Platform 환경 외부의 모든 위치에서의 트래픽은 라우팅 계층 없이 애플리케이션에 연결할 수 없습니다.

Build

빌드는 입력 매개변수를 결과 오브젝트로 변환하는 프로세스입니다. 대부분의 경우 프로세스는 입력 매개변수 또는 소스 코드를 실행 가능한 이미지로 변환하는 데 사용됩니다. **BuildConfig** 오브젝트는 전체 빌드 프로세스에 대한 정의입니다. OpenShift Container Platform은 빌드 이미지에서 컨테이너를 생성하고 이를 통합 레지스트리로 푸시함으로써 Kubernetes를 활용합니다.

프로젝트

OpenShift Container Platform은 프로젝트를 사용하여 사용자 또는 개발자 그룹이 함께 작업할 수 있도록 하여 격리 및 협업 단위 역할을 합니다. 리소스 범위를 정의하고, 프로젝트 관리자와 협업을 통해 리소스를 관리하고, 할당량 및 제한을 사용하여 사용자 리소스를 제한 및 추적할 수 있습니다.
프로젝트는 추가 주석이 있는 Kubernetes 네임스페이스입니다. 일반 사용자를 위한 리소스에 대한 액세스를 관리하는 핵심 수단입니다. 사용자 커뮤니티는 프로젝트를 통해 다른 커뮤니티와 별도로 콘텐츠를 구성하고 관리할 수 있습니다. 사용자는 관리자로부터 프로젝트에 대한 액세스 권한을 받아야 합니다. 그러나 클러스터 관리자는 개발자가 자신의 프로젝트를 만들 수 있도록 허용할 수 있으며, 이 경우 사용자는 자신의 프로젝트에 대한 액세스 권한이 자동으로 제공됩니다.

각 프로젝트에는 고유한 오브젝트, 정책, 제약 조건 및 서비스 계정이 있습니다.

프로젝트는 네임스페이스라고도 합니다.

Operator

Operator는 Kubernetes 네이티브 애플리케이션입니다. Operator의 목표는 운영 지식을 소프트웨어에 배치하는 것입니다. 이전에는 이러한 지식에는 관리자, 다양한 조합 또는 쉘 스크립트 또는 Ansible과 같은 자동화 소프트웨어만 있었습니다. Kubernetes 클러스터 외부에 있었고 통합하기 어려웠습니다. Operator를 사용하면 이러한 모든 사항이 변경됩니다.
Operator는 애플리케이션에 맞게 설계되었습니다. Kubernetes 개념 및 API와 기본적으로 통합되어

Kubernetes 클러스터 내부에서 실행되는 소프트웨어에서 설치 및 구성과 같은 일반적인 1일차(Day 1) 작업 및 2일차(Day 2) 작업(스케일업/다운, 재구성, 업데이트, 백업, 페일오버 및 복원)을 구현하고 자동화합니다. 이를 Kubernetes 네이티브 애플리케이션이라고 합니다.

Operator에서는 애플리케이션을 Pod, 배포, 서비스 또는 구성 맵과 같은 기본 구성 요소 컬렉션으로 취급해서는 안 됩니다. 대신 애플리케이션에 적합한 옵션을 노출하는 단일 오브젝트로 Operator를 처리해야 합니다.

2.2. OPENSHIFT CONTAINER PLATFORM 이해

OpenShift Container Platform은 베어 메탈, 가상화, 온프레미스, 클라우드와 같은 다양한 컴퓨팅 플랫폼에 대한 컨테이너 기반 애플리케이션의 라이프사이클과 종속 항목을 관리하는 Kubernetes 환경입니다. OpenShift Container Platform은 컨테이너를 배포, 구성 및 관리합니다. OpenShift Container Platform은 구성 요소의 사용성, 안정성 및 사용자 지정 기능을 제공합니다.

OpenShift Container Platform은 노드라고 하는 여러 컴퓨팅 리소스를 사용합니다. 노드에는 RHCOS(Red Hat Enterprise Linux CoreOS)라는 RHEL(Red Hat Enterprise Linux CoreOS)을 기반으로 하는 경량의 보안 운영 체제가 있습니다.

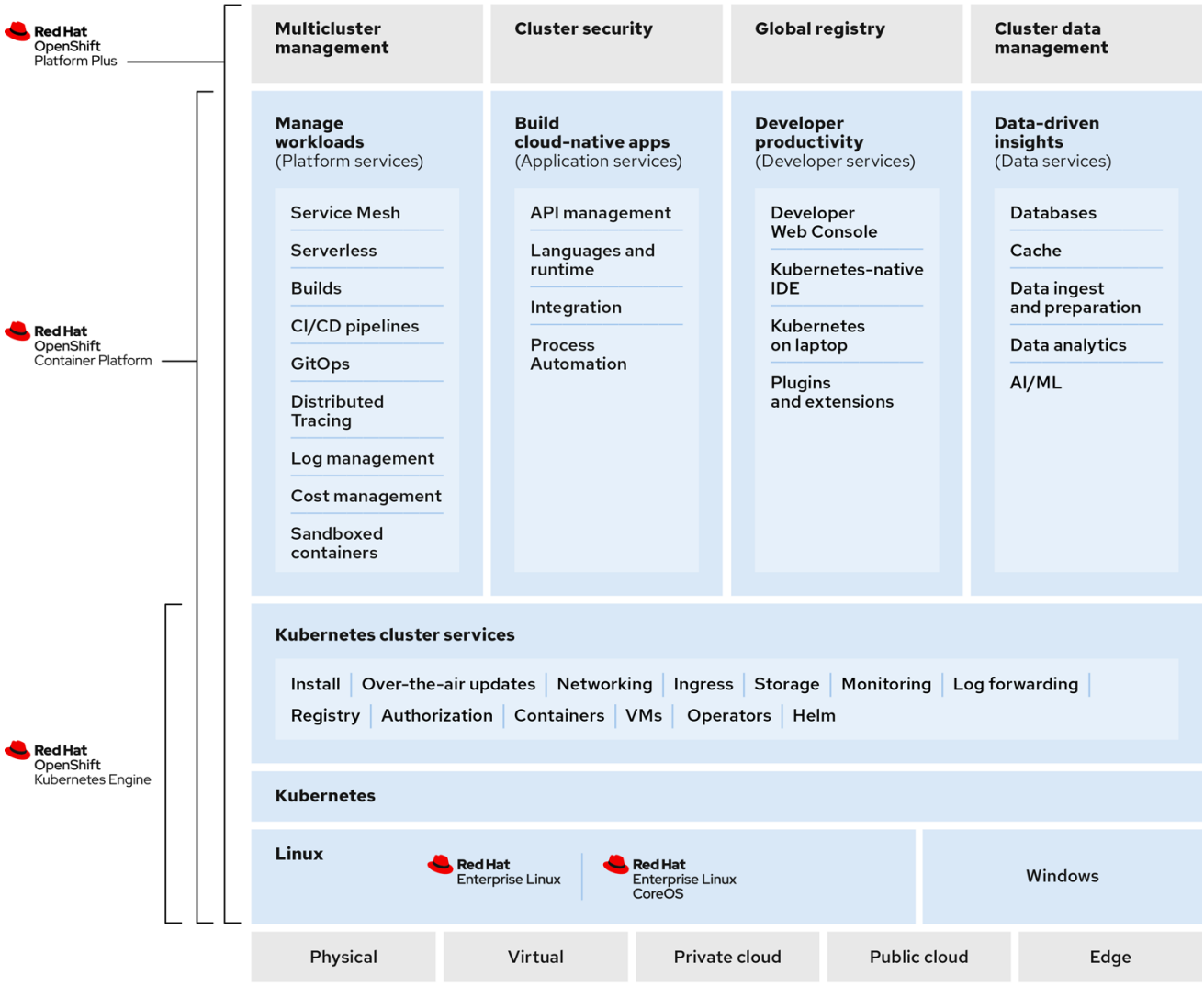
노드가 부팅 및 구성되면 예약된 컨테이너 워크로드의 이미지를 관리하고 실행하기 위해 CRI-O 또는 Docker와 같은 컨테이너 런타임을 가져옵니다. Kubernetes 에이전트 또는 kubelet은 노드에서 컨테이너 워크로드를 예약합니다. kubelet은 클러스터에 노드를 등록하고 컨테이너 워크로드의 세부 정보를 수신해야 합니다.

OpenShift Container Platform은 클러스터의 네트워킹, 로드 밸런싱 및 라우팅을 구성하고 관리합니다. OpenShift Container Platform은 클러스터 상태 및 성능, 로깅 및 업그레이드 관리를 위한 클러스터 서비스를 추가합니다.

컨테이너 이미지 레지스트리 및 OperatorHub에서는 클러스터 내에서 다양한 애플리케이션 서비스를 제공하기 위해 Red Hat 인증 제품 및 커뮤니티 빌드 소프트웨어를 제공합니다. 이러한 애플리케이션 및 서비스는 클러스터, 데이터베이스, 프런트 엔드 및 사용자 인터페이스, 애플리케이션 런타임 및 비즈니스 자동화, 컨테이너 애플리케이션 개발 및 테스트를 위한 개발자 서비스에 배포된 애플리케이션을 관리합니다.

사전 빌드된 이미지 또는 Operator라는 리소스를 통해 실행되는 컨테이너 배포를 구성하여 클러스터 내에서 애플리케이션을 수동으로 관리할 수 있습니다. 사전 빌드 이미지 및 소스 코드에서 사용자 정의 이미지를 빌드하고 이러한 사용자 지정 이미지를 내부, 개인 또는 퍼블릭 레지스트리에 로컬로 저장할 수 있습니다.

멀티클러스터 관리 계층은 단일 콘솔에서 워크로드 배포, 구성, 규정 준수 및 배포를 포함하여 여러 클러스터를 관리할 수 있습니다.



2.3. OPENSHIFT CONTAINER PLATFORM 설치

OpenShift Container Platform 설치 프로그램에서는 유연성을 제공합니다. 설치 프로그램을 사용하면 설치 프로그램이 프로비저닝하고 클러스터가 유지보수하는 인프라에 클러스터를 배포하거나 준비하고 유지보수하는 인프라에 클러스터를 배포할 수 있습니다.

설치 프로세스, 지원되는 플랫폼 및 클러스터 설치 및 준비 방법에 대한 자세한 내용은 다음을 참조하십시오.

- OpenShift Container Platform 설치 프로그램 개요
- 설치 프로세스
- OpenShift Container Platform 클러스터에서 지원되는 플랫폼
- 클러스터 설치 유형 선택

2.3.1. OpenShift 로컬 개요

OpenShift Local은 OpenShift Container Platform 클러스터 빌드를 시작할 수 있도록 빠른 애플리케이션 개발을 지원합니다. OpenShift Local은 로컬 컴퓨터에서 실행되어 설정 및 테스트를 단순화하고 컨테이너 기반 애플리케이션을 개발하는 데 필요한 모든 툴을 사용하여 클라우드 개발 환경을 로컬로 에뮬레이션하

도록 설계되었습니다.

사용하는 프로그래밍 언어와 관계없이 OpenShift Local은 애플리케이션을 호스팅하고 서버 기반 인프라 없이도 로컬 PC에 사전 구성된 최소 Red Hat OpenShift Container Platform 클러스터를 제공합니다.

호스팅된 환경에서 OpenShift Local은 Linux, macOS 또는 Windows 10 이상을 실행하는 랩탑 또는 데스크탑에서 직접 마이크로서비스를 생성하고 이미지로 변환한 컨테이너에서 직접 실행할 수 있습니다.

OpenShift Local에 대한 자세한 내용은 Red Hat OpenShift 로컬 개요 를 참조하십시오.

2.4. 다음 단계

2.4.1. 개발자의 경우

OpenShift Container Platform을 사용하여 컨테이너화된 애플리케이션을 개발하고 배포합니다. OpenShift Container Platform은 컨테이너화된 애플리케이션을 개발하고 배포하기 위한 플랫폼입니다. OpenShift Container Platform 설명서는 다음을 지원합니다.

- **OpenShift Container Platform 개발 이해**: 간단한 컨테이너에서 고급 Kubernetes 배포 및 Operator에 이르기까지 다양한 유형의 컨테이너화된 애플리케이션을 알아봅니다.
- **프로젝트 작업**: OpenShift Container Platform 웹 콘솔 또는 **oc**CLI(OpenShift CLI)에서 프로젝트를 생성하여 개발하는 소프트웨어를 구성하고 공유합니다.
- **애플리케이션 작업**:

OpenShift Container Platform 웹 콘솔 의 **개발자** 화면을 사용하여 애플리케이션을 생성하고 배포합니다.

토폴로지 보기를 사용하여 애플리케이션을 확인하고, 상태를 모니터링하고, 구성 요소를 연결 및 그룹화하며 코드 기반을 수정합니다.

- **odo(개발자 CLI 툴) 사용: odo** CLI 툴을 사용하면 개발자가 단일 또는 다중 구성 요소 애플리케이션을 생성하고 배포, 빌드 및 서비스 경로 구성을 자동화할 수 있습니다. 복잡한 Kubernetes 및 OpenShift Container Platform 개념을 추상화하여 애플리케이션 개발에 집중할 수 있습니다.
- **CI/CD Pipelines 생성**: Pipeline은 서버리스, 클라우드 네이티브, 연속 통합 및 격리된 컨테이너에서 실행되는 지속적인 배포 시스템입니다. 표준 Tekton 사용자 지정 리소스를 사용하여 배포를 자동화하고 마이크로서비스 기반 아키텍처에서 작업하는 분산된 팀을 위해 설계되었습니다.
- **Helm 차트 배포**:Helm 3 은 개발자가 Kubernetes에서 애플리케이션 패키지를 정의, 설치 및 업데이트하는 데 도움이 되는 패키지 관리자입니다. Helm 차트는 Helm CLI를 사용하여 배포할 수 있는 애플리케이션을 설명하는 패키징 형식입니다.
- **이미지 빌드 이해**: 다양한 종류의 소스 자료(Git 리포지토리, 로컬 바이너리 입력 및 외부 아티팩트)를 포함할 수 있는 다양한 빌드 전략(Docker, S2I, 사용자 정의 및 파이프라인)에서 선택합니다. 그런 다음 기본 빌드에서 고급 빌드로 빌드 유형 예제를 따릅니다.
- **컨테이너 이미지 생성**: 컨테이너 이미지는 OpenShift Container Platform (및 Kubernetes) 애플리케이션에서 가장 기본적인 빌딩 블록입니다. 이미지 스트림을 정의하면 개발을 계속할 때 이미지의 여러 버전을 한 곳에 수집할 수 있습니다. S2I 컨테이너를 사용하면 Ruby, Node.js 또는 Python과 같은 특정 유형의 코드를 실행하도록 설정된 기본 컨테이너에 소스 코드를 삽입할 수 있습니다.
- **배포 생성**: **Deployment** 및 **DeploymentConfig** 오브젝트를 사용하여 애플리케이션에 대한 세분화된 관리를 제공합니다. **워크로드** 페이지 또는 **oc**CLI(OpenShift CLI)를 사용하여 배포를 관리합니다. 롤링, 재현 및 사용자 정의 배포 전략에 대해 알아보십시오.

- **템플릿 생성**: 기존 템플릿을 사용하거나 애플리케이션 빌드 또는 배포 방법을 설명하는 자체 템플릿을 생성합니다. 템플릿은 이미지를 설명, 매개변수, 복제본, 노출된 포트 및 애플리케이션 실행 또는 구축 방법을 정의하는 기타 콘텐츠와 결합할 수 있습니다.

- **Operators 이해**: Operator는 OpenShift Container Platform 4.13용 클러스터 기반 애플리케이션을 생성하는 기본 방법입니다. Operator 프레임워크 및 설치된 Operator를 사용하여 프로젝트에 애플리케이션을 배포하는 방법을 알아봅니다.

- **Operators 개발**: Operator는 OpenShift Container Platform 4.13 용 클러스터 기반 애플리케이션을 생성하는 데 선호되는 방법입니다. Operator를 빌드, 테스트 및 배포하기 위한 워크플로를 알아봅니다. 그런 다음 Ansible 또는 Helm 을 기반으로 자체 Operator를 생성하거나 Operator SDK를 사용하여 기본 제공 Prometheus 모니터링 을 구성합니다.

- **REST API 참조**: OpenShift Container Platform 애플리케이션 프로그래밍 인터페이스 끝점에 대해 알아봅니다.

2.4.2. 관리자의 경우

- **OpenShift Container Platform 관리 이해**: OpenShift Container Platform 4.13 컨트롤 플레인의 구성 요소에 대해 알아봅니다. Machine API 및 Operator를 통해 OpenShift Container Platform 컨트롤 플레인 및 작업자 노드를 관리하고 업데이트하는 방법을 확인하십시오.

- **사용자 및 그룹 관리**: 클러스터를 사용하거나 수정할 수 있는 다양한 수준의 권한이 있는 사용자와 그룹을 추가합니다.

- **인증 관리**: OpenShift Container Platform에서 사용자, 그룹 및 API 인증이 작동하는 방식을 알아봅니다. OpenShift Container Platform은 여러 ID 공급자를 지원합니다.

- **Manage networking**: OpenShift Container Platform의 클러스터 네트워크는 CNO(Cluster Network Operator)에서 관리합니다. CNO는 kube-proxy 의 iptables 규칙을 사용하여 해당 노드에서 실행되는 노드와 Pod 간 트래픽을 전달합니다. Multus 컨테이너 네트워크 인터페이스는 여러 네트워크 인터페이스를 Pod에 연결하는 기능을 추가합니다. 네트워크 정책 기능을 사용하여 Pod를 분리하거나 선택한 트래픽을 허용할 수 있습니다.

- **스토리지 관리**: OpenShift Container Platform을 사용하면 클러스터 관리자가 영구 스토리지를 구성할 수 있습니다.

- **Operator 관리**: Red Hat, ISV 및 커뮤니티 Operator 목록은 클러스터 관리자가 검토하고 클러스터에 설치할 수 있습니다. 설치한 후 클러스터에서 , 업그레이드, 백업 또는 Operator를 관리할 수 있습니다.
https://docs.redhat.com/en/documentation/openshift_container_platform/4.13/html-single/operators/#olm-creating-apps-from-installed-operators

- **CRD(사용자 정의 리소스 정의)를 사용하여 클러스터를 수정합니다**: Operator와 함께 구현된 클러스터 기능은 CRD를 사용하여 수정할 수 있습니다. CRD를 생성하고 CRD 에서 리소스를 관리하는 방법을 알아봅니다.

- **리소스 할당량 설정**: CPU, 메모리 및 기타 시스템 리소스에서 선택하여 할당량을 설정합니다.

- **리소스 정리 및 회수**: 불필요한 Operator, 그룹, 배포, 빌드, 이미지, 레지스트리, cron 작업을 정리하여 공간을 회수합니다.

- **클러스터 스케일링 및 조정** : 클러스터 제한을 설정하고, 노드를 튜닝하고, 클러스터 모니터링을 스케일링하고, 환경에 대한 네트워킹, 스토리지 및 경로를 최적화합니다.

- **OpenShift Update Service 이해**: 연결이 끊긴 환경에서 OpenShift Container Platform 업데이트를 권장하기 위해 로컬 OpenShift Update Service 설치 및 관리에 대해 알아봅니다.

- **모니터 클러스터**: 모니터링 스택을 구성하는 방법을 알아봅니다. 모니터링을 구성한 후 웹 콘솔을 사용하여 모니터링 대시보드에 액세스합니다. 인프라 메트릭 외에도 자체 서비스에 대한 메트릭을 스크랩 및 볼 수 있습니다.

- **원격 상태 모니터링**: OpenShift Container Platform은 클러스터에 대한 익명화된 집계 정보를 수집합니다. Telemetry 및 Insights Operator를 사용하여 이 데이터는 Red Hat에서 수신하며 OpenShift Container Platform을 개선하는 데 사용됩니다. 원격 상태 모니터링에서 수집한 데이터를 볼 수 있습니다.

3장. 웹 콘솔을 사용하여 애플리케이션 생성 및 빌드

3.1. 사전 준비 사항

- 웹 콘솔 액세스를 검토합니다.
- 실행 중인 OpenShift Container Platform 인스턴스에 액세스할 수 있어야 합니다. 액세스 권한이 없는 경우 클러스터 관리자에게 문의하십시오.

3.2. 웹 콘솔에 로그인

OpenShift Container Platform 웹 콘솔에 로그인하여 클러스터에 액세스하여 관리할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.

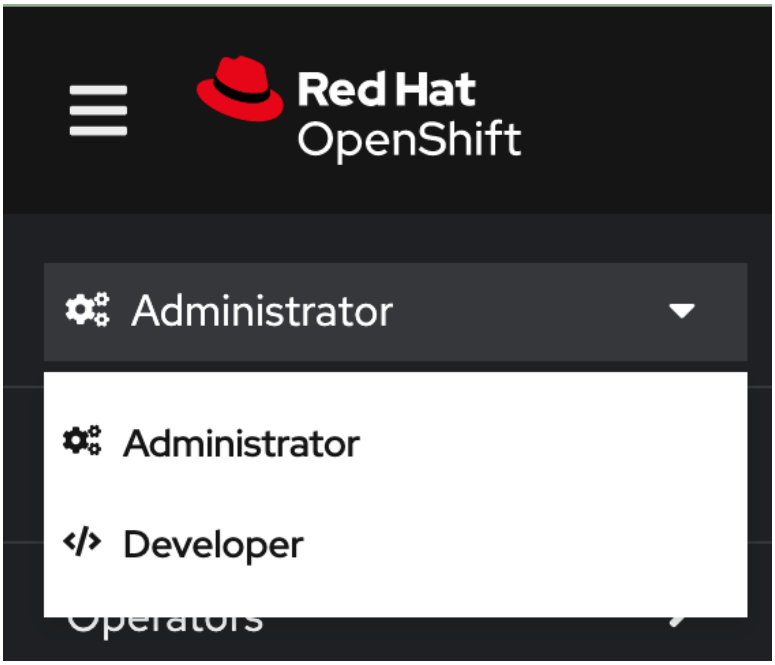
절차

- 로그인 인증 정보를 사용하여 OpenShift Container Platform 웹 콘솔에 로그인합니다.

프로젝트 페이지로 리디렉션됩니다. 관리자가 아닌 사용자의 경우 기본 보기는 **개발자** 화면입니다. 클러스터 관리자의 경우 기본 보기는 **관리자** 화면입니다. **cluster-admin** 권한이 없는 경우 웹 콘솔에 **관리자** 화면이 표시되지 않습니다.

웹 콘솔은 **관리자** 화면과 **개발자** 화면이라는 두 가지 화면을 제공합니다. **개발자 화면**은 개발자의 유스 케이스에 특정한 워크 플로우를 제공합니다.

그림 3.1. 화면 전환기



모드 전환 기능을 사용하여 **Developer** 모드로 전환하십시오. 애플리케이션을 생성할 수 있는 옵션이 포함된 **토폴로지** 보기가 표시됩니다.

3.3. 새 프로젝트 생성

사용자 커뮤니티는 프로젝트를 통해 별도로 콘텐츠를 구성하고 관리할 수 있습니다. 프로젝트는 Kubernetes 네임스페이스에 대한 OpenShift Container Platform 확장입니다. 프로젝트에는 사용자 셀프 프로비저닝을 활성화하는 추가 기능이 있습니다.

사용자는 관리자로부터 프로젝트에 대한 액세스 권한을 받아야 합니다. 클러스터 관리자는 개발자가 자신의 프로젝트를 만들 수 있도록 허용할 수 있습니다. 대부분의 경우 사용자는 자신의 프로젝트에 자동으로 액세스할 수 있습니다.

각 프로젝트에는 고유한 오브젝트, 정책, 제약 조건 및 서비스 계정이 있습니다.

사전 요구 사항

- OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.
- **개발자** 화면에 있습니다.
- 프로젝트에 OpenShift Container Platform에서 애플리케이션 및 기타 워크로드를 생성할 적절한 역할과 권한이 있습니다.

절차

1. **+추가** 보기에서 **프로젝트 → 프로젝트 생성**을 선택합니다.
2. **이름** 필드에 **user-getting-started**를 입력합니다.
3. 선택 사항: **표시 이름** 필드에 **OpenShift 시작하기**를 입력합니다.

참고

표시 이름 및 **설명** 필드는 선택 사항입니다.

4. **생성**을 클릭합니다.

OpenShift Container Platform에서 첫 번째 프로젝트를 생성했습니다.

추가 리소스

- 기본 클러스터 역할
- 웹 콘솔을 사용하여 프로젝트 보기
- 개발자 화면을 사용하여 프로젝트에 액세스 권한 제공
- 웹 콘솔을 사용하여 프로젝트 삭제

3.4. 보기 권한 부여

OpenShift Container Platform은 모든 프로젝트에서 자동으로 몇 가지 특수 서비스 계정을 생성합니다. 기본 서비스 계정은 Pod 실행을 담당합니다. OpenShift Container Platform은 이 서비스 계정을 사용하고 시작하는 모든 Pod에 삽입합니다.

다음 절차에서는 기본 **ServiceAccount** 오브젝트에 대한 **RoleBinding** 오브젝트를 생성합니다. 서비스 계정은 OpenShift Container Platform API와 통신하여 프로젝트 내의 Pod, 서비스 및 리소스에 대해 알아봅니다.

사전 요구 사항

- OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.
- 배포된 이미지가 있습니다.
- **관리자** 화면에 있습니다.

절차

1. **사용자 관리**로 이동한 다음 **바인딩 역할**을 클릭합니다.
2. **바인딩 생성**을 클릭합니다.
3. **Namespace role binding (RoleBinding)**을 선택합니다.
4. **이름** 필드에 **sa-user-account** 를 입력합니다.
5. **네임스페이스** 필드에서 **user-getting-started**를 검색하고 선택합니다.
6. **역할 이름** 필드에서 **view**를 검색하고 **view**를 선택합니다.
7. **제목** 필드에서 **ServiceAccount**를 선택합니다.
8. **제목 네임스페이스** 필드에서 **user-getting-started**를 검색하고 선택합니다.
9. **제목 이름** 필드에 **default**를 입력합니다.
10. **생성**을 클릭합니다.

추가 리소스

- 인증 이해
- RBAC 개요

3.5. 첫 번째 이미지 배포

OpenShift Container Platform에서 애플리케이션을 배포하는 가장 간단한 방법은 기존 컨테이너 이미지를 실행하는 것입니다. 다음 절차에서는 **national-parks-app**이라는 애플리케이션의 프런트 엔드 구성 요소를 배포합니다. 웹 애플리케이션에 대화식 맵이 표시됩니다. 이 지도는 전 세계 주요 국립공원의 위치를 표시합니다.

사전 요구 사항

- OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.
- **개발자** 화면에 있습니다.
- 프로젝트에 OpenShift Container Platform에서 애플리케이션 및 기타 워크로드를 생성할 적절한 역할과 권한이 있습니다.

절차

1. **개발자** 화면의 **+추가** 보기에서 **컨테이너 이미지**를 클릭하여 대화 상자를 엽니다.

2. **이미지 이름** 필드에 **quay.io/openshiftroadshow/parksmap:latest**를 입력합니다.
3. 다음 사항에 대한 현재 값이 있는지 확인합니다.
   a. 애플리케이션: **national-parks-app**
   b. 이름: **parksmap**
4. **배포**를 **리소스**로 선택합니다.
5. **애플리케이션에 대한 경로 생성**을 선택합니다.
6. **고급 옵션** 섹션에서 **레이블**을 클릭하고 레이블을 추가하면 나중에 이 배포를 더 잘 식별할 수 있습니다. 레이블은 웹 콘솔과 명령줄에서 구성 요소를 식별하고 필터링할 수 있습니다. 다음 레이블을 추가합니다.
   - **app=national-parks-app**
   - **component=parksmap**
   - **role=frontend**
7. **생성**을 클릭합니다.

national-parks-app 애플리케이션에서 **parksmap** 배포를 볼 수 있는 **토폴로지** 페이지로 리디렉션됩니다.

추가 리소스

- 개발자 화면을 사용하여 애플리케이션 생성
- 웹 콘솔을 사용하여 프로젝트 보기
- 애플리케이션의 토폴로지 보기
- 웹 콘솔을 사용하여 프로젝트 삭제

3.5.1. Pod 검사

OpenShift Container Platform은 하나의 호스트에 함께 배포되는 하나 이상의 컨테이너인 pod의 Kubernetes 개념과 정의, 배포 및 관리할 수 있는 최소 컴퓨팅 단위를 활용합니다. Pod는 컨테이너에 대한 머신 인스턴스, 물리 또는 가상과 대략적으로 동일합니다.

개요 패널을 사용하면 **parksmap** 배포의 많은 기능에 액세스할 수 있습니다. **세부 정보** 및 **리소스** 탭을 사용하면 애플리케이션 Pod를 스케일링하고 빌드 상태, 서비스 및 경로를 확인할 수 있습니다.

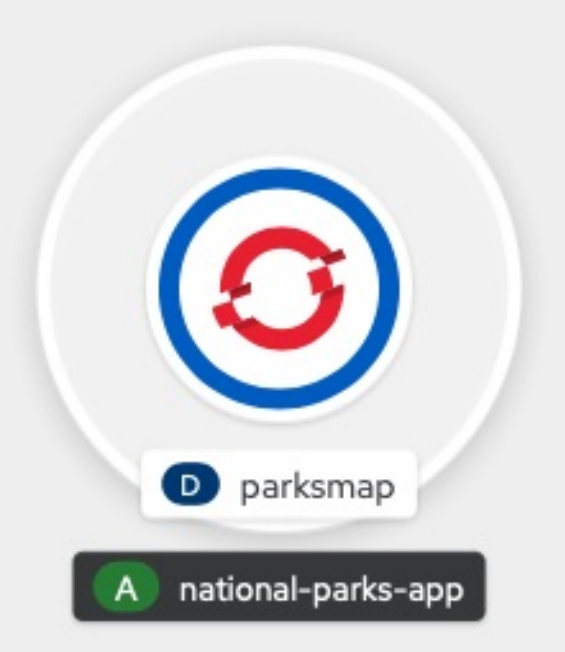
사전 요구 사항

- OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.
- **개발자** 화면에 있습니다.
- 배포된 이미지가 있습니다.

절차

- **토폴로지** 보기에서 **D parksmap**을 클릭하여 **개요** 패널을 엽니다.

그림 3.2. Parksmap 배포



개요 패널에는 **세부 정보**, **리소스** 및 **모니터링**에 대한 탭이 포함되어 있습니다. **세부 정보** 탭은 기본적으로 표시될 수 있습니다.

표 3.1. 개요 패널 탭 정의

탭	정의
세부 정보	애플리케이션을 확장하고 레이블, 주석, 애플리케이션 상태와 같은 Pod 구성을 볼 수 있습니다.
Resources	배포와 관련된 리소스를 표시합니다.
	Pod는 OpenShift Container Platform 애플리케이션의 기본 단위입니다. 사용 중인 Pod 수, Pod 상태 및 로그를 확인할 수 있습니다.
	Pod 및 할당된 포트용으로 생성된 **서비스**는 **서비스** 제목 아래에 나열됩니다.
	경로를 사용하면 pod에 대한 외부 액세스가 가능하며 URL을 사용하여 pod에 액세스할 수 있습니다.
모니터링	Pod와 관련된 다양한 **이벤트** 및 **메트릭** 정보를 확인합니다.

추가 리소스

- 애플리케이션 및 구성 요소와 상호 작용
- 애플리케이션 Pod 스케일링 및 빌드와 경로 확인
- 토폴로지 보기에 사용되는 라벨 및 주석

3.5.2. 애플리케이션 스케일링

Kubernetes에서 **Deployment** 오브젝트는 애플리케이션이 배포하는 방법을 정의합니다. 대부분의 경우 사용자는 **Pod**,**Service**,**ReplicaSets** 및 **Deployment** 리소스를 함께 사용합니다. 대부분의 경우 OpenShift Container Platform이 리소스를 생성합니다.

national-parks-app 이미지를 배포하면 배포 리소스가 생성됩니다. 예에서는 하나의 **Pod**만 배포됩니다.

다음 절차에서는 두 개의 인스턴스를 사용하도록 **national-parks-image** 크기를 조정합니다.

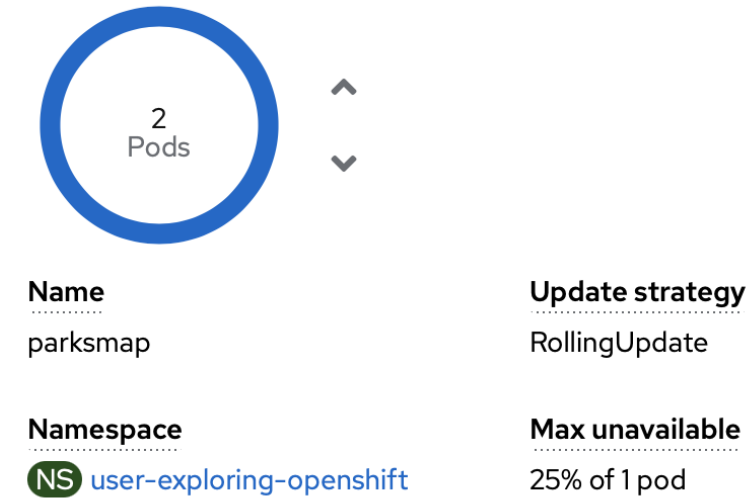
사전 요구 사항

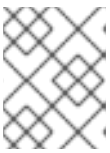
- OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.
- **개발자** 화면에 있습니다.
- 배포된 이미지가 있습니다.

절차

1. **토폴로지** 보기에서 **national-parks-app** 애플리케이션을 클릭합니다.
2. **세부 정보** 탭을 클릭합니다.
3. 위쪽 화살표를 사용하여 pod를 두 개의 인스턴스로 확장합니다.

   그림 3.3. 애플리케이션 확장





참고

OpenShift Container Platform이 기존 이미지의 새 인스턴스를 시작하므로 애플리케이션 확장이 빠르게 이루어질 수 있습니다.

4. 아래쪽 화살표를 사용하여 pod를 하나의 인스턴스로 축소합니다.

추가 리소스

- 클러스터 스케일링에 대한 권장 사례

- 수평 Pod 자동 스케일러 이해
- Vertical Pod Autoscaler Operator 정보

3.6. PYTHON 애플리케이션 배포

다음 절차에서는 **parksmap** 애플리케이션에 대한 백엔드 서비스를 배포합니다. Python 애플리케이션은 MongoDB 데이터베이스에 대해 2D 지리 공간 쿼리를 수행하여 전 세계 모든 국립공원의 지도 좌표를 찾고 반환합니다.

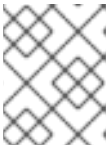
배포된 백엔드 서비스 (**nationalparks**)입니다.

사전 요구 사항

- OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.
- **개발자** 화면에 있습니다.
- 배포된 이미지가 있습니다.

절차

1. **개발자** 화면의 **+추가** 보기에서 **Git에서 가져오기**를 클릭하여 대화 상자를 엽니다.
2. Git Repo URL 필드에 다음 URL을 입력하십시오. **https://github.com/openshift-roadshow/nationalparks-py.git**
   빌더 이미지가 자동으로 감지됩니다.



참고

감지된 빌더 이미지가 Dockerfile인 경우 **가져오기 전략 편집**을 선택합니다. **빌더 이미지**를 선택한 다음 **Python**을 클릭합니다.

3. **일반** 섹션으로 스크롤합니다.
4. 다음 사항에 대한 현재 값이 있는지 확인합니다.
   a. 애플리케이션: **national-parks-app**
   b. 이름: **nationalparks**
5. **배포**를 **리소스**로 선택합니다.
6. **애플리케이션에 대한 경로 생성**을 선택합니다.
7. **고급 옵션** 섹션에서 **레이블**을 클릭하고 레이블을 추가하면 나중에 이 배포를 더 잘 식별할 수 있습니다. 레이블은 웹 콘솔과 명령줄에서 구성 요소를 식별하고 필터링할 수 있습니다. 다음 레이블을 추가합니다.
   a. **app=national-parks-app**
   b. **component=nationalparks**
   c. **role=backend**
   d. **type=parksmap-backend**

8. **생성**을 클릭합니다.
9. **토폴로지** 보기에서 **nationalparks** 애플리케이션을 선택합니다.

참고

Resources 탭을 클릭합니다. **Builds** (빌드) 섹션에서 빌드가 실행되는 것을 확인할 수 있습니다.

추가 리소스

- 애플리케이션에 서비스 추가
- Git에서 코드베이스를 가져와 애플리케이션 생성
- 애플리케이션의 토폴로지 보기
- 개발자 화면을 사용하여 프로젝트에 액세스 권한 제공
- 웹 콘솔을 사용하여 프로젝트 삭제

3.7. 데이터베이스에 연결

national-parks-app 애플리케이션에서 위치 정보를 저장하는 MongoDB 데이터베이스를 배포 및 연결합니다. **national-parks-app** 애플리케이션을 맵 시각화 도구의 백엔드로 표시하면 **parksmap** 배포에서는 OpenShift Container Platform 검색 메커니즘을 사용하여 지도를 자동으로 표시합니다.

사전 요구 사항

- OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.
- **개발자** 화면에 있습니다.
- 배포된 이미지가 있습니다.

절차

1. **개발자** 화면의 **+추가** 보기에서 **컨테이너 이미지**를 클릭하여 대화 상자를 엽니다.
2. **이미지 이름** 필드에 **quay.io/centos7/mongodb-36-centos7**을 입력합니다.
3. **런타임 아이콘** 필드에서 **mongodb**를 검색합니다.
4. **일반** 섹션까지 아래로 스크롤합니다.
5. 다음 사항에 대한 현재 값이 있는지 확인합니다.
   a. 애플리케이션: **national-parks-app**
   b. 이름: **mongodb-nationalparks**
6. **배포**를 **리소스**로 선택합니다.
7. **애플리케이션에 대한 경로 만들기** 옆에 있는 확인란을 선택 해제합니다.
8. **고급 옵션** 섹션에서 **배포**를 클릭하여 다음 환경 변수를 추가합니다.

표 3.2. 환경 변수 이름 및 값

이름	현재의
MONGODB_USER	**mongodb**
MONGODB_PASSWORD	**mongodb**
MONGODB_DATABASE	**mongodb**
MONGODB_ADMIN_PASSWORD	**mongodb**

9. **생성**을 클릭합니다.

추가 리소스

- 애플리케이션에 서비스 추가
- 웹 콘솔을 사용하여 프로젝트 보기
- 애플리케이션의 토폴로지 보기
- 개발자 화면을 사용하여 프로젝트에 액세스 권한 제공
- 웹 콘솔을 사용하여 프로젝트 삭제

3.7.1. 시크릿 생성

Secret 오브젝트는 암호, OpenShift Container Platform 클라이언트 구성 파일, 개인 소스 리포지토리 자격 증명 등과 같은 중요한 정보를 보유하는 메커니즘을 제공합니다. 보안은 Pod에서 중요한 콘텐츠를 분리합니다. 볼륨 플러그인을 사용하여 컨테이너에 보안을 마운트하거나 시스템에서 시크릿을 사용하여 Pod 대신 작업을 수행할 수 있습니다. 다음 절차에서는 시크릿 **nationalparks-mongodb-parameters**를 추가하고 이를 **nationalparks** 워크로드에 마운트합니다.

사전 요구 사항

- OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.
- **개발자** 화면에 있습니다.
- 배포된 이미지가 있습니다.

절차

1. **개발자** 화면에서 왼쪽 탐색의 **시크릿**으로 이동하여 **시크릿** 을 클릭합니다.
2. **생성 → 키/값 시크릿**을 클릭합니다.
   a. **시크릿 이름** 필드에 **nationalparks-mongodb-parameters**를 입력합니다.
   b. **키** 및 **값**에 대해 다음 값을 입력합니다.

표 3.3. 시크릿 키 및 값

키	현재의
MONGODB_USER	**mongodb**
DATABASE_SERVICE_NAME	**mongodb-nationalparks**
MONGODB_PASSWORD	**mongodb**
MONGODB_DATABASE	**mongodb**
MONGODB_ADMIN_PASSWORD	**mongodb**

c. **생성**을 클릭합니다.

3. **워크로드에 시크릿 추가**를 클릭합니다.

a. 드롭다운 메뉴에서 추가할 워크로드로 **nationalparks**를 선택합니다.

b. **저장**을 클릭합니다.

이러한 구성 변경으로 인해 환경 변수가 제대로 삽입된 **nationalparks** 배포의 새로운 롤아웃을 트리거합니다.

추가 리소스

- 보안 이해

3.7.2. 데이터 로드 및 국립 공원 지도 표시

parksmap 및 **nationalparks** 애플리케이션을 배포한 다음 **mongodb-nationalparks** 데이터베이스를 배포했습니다. 그러나 *데이터베이스에* 데이터가 로드되지 않았습니다. 데이터를 로드하기 전에 **mongodb-nationalparks** 및 **nationalparks** 배포에 적절한 레이블을 추가합니다.

사전 요구 사항

- OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.
- **개발자** 화면에 있습니다.
- 배포된 이미지가 있습니다.

절차

1. **토폴로지** 보기에서 **nationalparks** 배포로 이동하여 **리소스**를 클릭하고 경로 정보를 검색합니다.
2. URL을 복사하여 웹 브라우저에 붙여넣고 URL 끝에 다음을 추가합니다.

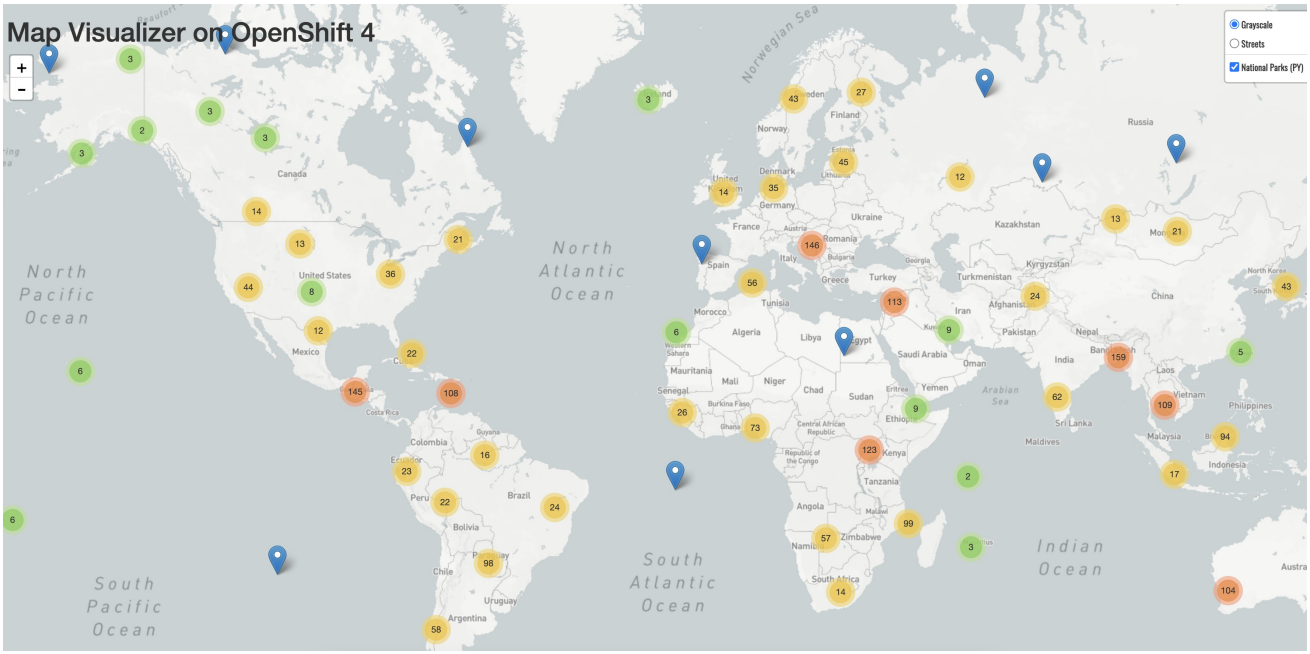
```
/ws/data/load
```

출력 예

```
Items inserted in database: 2893
```

3. **토폴로지** 보기에서 **parksmap** 배포로 이동하여 **리소스**를 클릭하고 경로 정보를 검색합니다.
4. URL을 복사하여 웹 브라우저에 붙여넣어 세계 지도에서 국립 공원을 볼 수 있습니다.

그림 3.4. 전 세계의 국립공원



추가 리소스

- 개발자 화면을 사용하여 프로젝트에 액세스 권한 제공
- 토폴로지 보기에 사용되는 라벨 및 주석

4장. CLI를 사용하여 애플리케이션 생성 및 빌드

4.1. 사전 준비 사항

- OpenShift CLI 정보를 검토합니다.
- 실행 중인 OpenShift Container Platform 인스턴스에 액세스할 수 있어야 합니다. 액세스 권한이 없는 경우 클러스터 관리자에게 문의하십시오.
- OpenShift CLI(**oc**)가 다운로드 및 설치되어 있어야 합니다.

4.2. CLI에 로그인

OpenShift CLI (**oc**) 에 로그인하면 클러스터에 액세스하여 관리할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있어야 합니다.

절차

- 사용자 이름과 암호, OAuth 토큰 또는 웹 브라우저를 사용하여 CLI에서 OpenShift Container Platform에 로그인합니다.
  - 사용자 이름 및 암호:

    ```
    $ oc login -u=<username> -p=<password> --server=<your-openshift-server> --insecure-skip-tls-verify
    ```

  - OAuth 토큰 사용:

    ```
    $ oc login <https://api.your-openshift-server.com> --token=<tokenID>
    ```

  - 웹 브라우저의 경우:

    ```
    $ oc login <cluster_url> --web
    ```

이제 클러스터를 관리하기 위한 프로젝트를 생성하거나 다른 명령을 실행할 수 있습니다.

추가 리소스

- oc login
- oc logout

4.3. 새 프로젝트 생성

사용자 커뮤니티는 프로젝트를 통해 별도로 콘텐츠를 구성하고 관리할 수 있습니다. 프로젝트는 Kubernetes 네임스페이스에 대한 OpenShift Container Platform 확장입니다. 프로젝트에는 사용자 셀프 프로비저닝을 활성화하는 추가 기능이 있습니다.

사용자는 관리자로부터 프로젝트에 대한 액세스 권한을 받아야 합니다. 클러스터 관리자는 개발자가 자신의 프로젝트를 만들 수 있도록 허용할 수 있습니다. 대부분의 경우 사용자는 자신의 프로젝트에 자동으로 액세스할 수 있습니다.

각 프로젝트에는 고유한 오브젝트, 정책, 제약 조건 및 서비스 계정이 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있어야 합니다.

절차

- 새 프로젝트를 생성하려면 다음 명령을 입력합니다.

```
$ oc new-project user-getting-started --display-name="Getting Started with OpenShift"
```

출력 예

```
Now using project "user-getting-started" on server "https://openshift.example.com:6443".
```

추가 리소스

- oc new-project

4.4. 보기 권한 부여

OpenShift Container Platform은 모든 프로젝트에서 자동으로 몇 가지 특수 서비스 계정을 생성합니다. 기본 서비스 계정은 Pod 실행을 담당합니다. OpenShift Container Platform은 이 서비스 계정을 사용하고 시작하는 모든 Pod에 삽입합니다.

다음 절차에서는 기본 **ServiceAccount** 오브젝트에 대한 **RoleBinding** 오브젝트를 생성합니다. 서비스 계정은 OpenShift Container Platform API와 통신하여 프로젝트 내의 Pod, 서비스 및 리소스에 대해 알아봅니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있어야 합니다.
- 배포된 이미지가 있습니다.
- **cluster-admin** 또는 **project-admin** 권한이 있어야 합니다.

절차

- **user-getting-started project**의 기본 서비스 계정에 보기 역할을 추가하려면 다음 명령을 입력합니다.

```
$ oc adm policy add-role-to-user view -z default -n user-getting-started
```

추가 리소스

- 인증 이해
- RBAC 개요
- oc policy add-role-to-user

4.5. 첫 번째 이미지 배포

OpenShift Container Platform에서 애플리케이션을 배포하는 가장 간단한 방법은 기존 컨테이너 이미지를 실행하는 것입니다. 다음 절차에서는 **national-parks-app**이라는 애플리케이션의 프런트 엔드 구성 요소를 배포합니다. 웹 애플리케이션에 대화식 맵이 표시됩니다. 이 지도는 전 세계 주요 국립공원의 위치를 표시합니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)를 설치합니다.

절차

- 애플리케이션을 배포하려면 다음 명령을 입력합니다.

```
$ oc new-app quay.io/openshiftroadshow/parksmap:latest --name=parksmap -l
'app=national-parks-app,component=parksmap,role=frontend,app.kubernetes.io/part-
of=national-parks-app'
```

출력 예

```
--> Found container image 0c2f55f (12 months old) from quay.io for
"quay.io/openshiftroadshow/parksmap:latest"

    * An image stream tag will be created as "parksmap:latest" that will track this image

--> Creating resources with label app=national-parks-app,app.kubernetes.io/part-of=national-
parks-app,component=parksmap,role=frontend ...
    imagestream.image.openshift.io "parksmap" created
    deployment.apps "parksmap" created
    service "parksmap" created
--> Success
```

추가 리소스

- oc new-app

4.5.1. 경로 생성

외부 클라이언트는 라우팅 계층과 *경로*인 데이터 오브젝트를 통해 OpenShift Container Platform에서 실행되는 애플리케이션에 액세스할 수 있습니다. 기본 OpenShift Container Platform 라우터(HAProxy)는 들어오는 요청의 HTTP 헤더를 사용하여 연결을 프록시할 위치를 결정합니다.

선택적으로 경로에 대해 TLS와 같은 보안을 정의할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있어야 합니다.
- 배포된 이미지가 있습니다.
- **cluster-admin** 또는 **project-admin** 권한이 있어야 합니다.

절차

1. 생성된 애플리케이션 서비스를 검색하려면 다음 명령을 입력합니다.

```
$ oc get service
```

출력 예

```
NAME      TYPE       CLUSTER-IP         EXTERNAL-IP    PORT(S)   AGE
parksmap  ClusterIP  <your-cluster-IP>  <123.456.789>  8080/TCP  8m29s
```

2. 경로를 생성하려면 다음 명령을 입력합니다.

```
$ oc create route edge parksmap --service=parksmap
```

출력 예

```
route.route.openshift.io/parksmap created
```

3. 생성된 애플리케이션 경로를 검색하려면 다음 명령을 입력합니다.

```
$ oc get route
```

출력 예

```
NAME       HOST/PORT                                          PATH   SERVICES   PORT
TERMINATION   WILDCARD
parksmap   parksmap-user-getting-started.apps.cluster.example.com           parksmap
8080-tcp   edge          None
```

추가 리소스

- oc create route edge
- oc get

4.5.2. Pod 검사

OpenShift Container Platform은 하나의 호스트에 함께 배포되는 하나 이상의 컨테이너인 pod의 Kubernetes 개념과 정의, 배포 및 관리할 수 있는 최소 컴퓨팅 단위를 활용합니다. Pod는 컨테이너에 대한 머신 인스턴스, 물리 또는 가상과 대략적으로 동일합니다.

클러스터의 Pod를 보고 해당 Pod 및 클러스터의 상태를 전체적으로 확인할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있어야 합니다.
- 배포된 이미지가 있습니다.

절차

1. 노드 이름이 있는 모든 Pod를 나열하려면 다음 명령을 입력합니다.

```
$ oc get pods
```

출력 예

```
NAME                  READY   STATUS    RESTARTS   AGE
parksmap-5f9579955-6sng8   1/1     Running   0          77s
```

2. 모든 Pod 세부 정보를 나열하려면 다음 명령을 입력합니다.

```
$ oc describe pods
```

출력 예

```
Name:         parksmap-848bd4954b-5pvcc
Namespace:    user-getting-started
Priority:     0
Node:         ci-ln-fr1rt92-72292-4fzf9-worker-a-g9g7c/10.0.128.4
Start Time:   Sun, 13 Feb 2022 14:14:14 -0500
Labels:       app=national-parks-app
              app.kubernetes.io/part-of=national-parks-app
              component=parksmap
              deployment=parksmap
              pod-template-hash=848bd4954b
              role=frontend
Annotations:  k8s.v1.cni.cncf.io/network-status:
                [{
                    "name": "openshift-sdn",
                    "interface": "eth0",
                    "ips": [
                        "10.131.0.14"
                    ],
                    "default": true,
                    "dns": {}
                }]
              k8s.v1.cni.cncf.io/network-status:
                [{
                    "name": "openshift-sdn",
                    "interface": "eth0",
                    "ips": [
                        "10.131.0.14"
                    ],
                    "default": true,
```

```
                "dns": {}
            }]
          openshift.io/generated-by: OpenShiftNewApp
          openshift.io/scc: restricted
Status:       Running
IP:           10.131.0.14
IPs:
  IP:           10.131.0.14
Controlled By:  ReplicaSet/parksmap-848bd4954b
Containers:
  parksmap:
    Container ID:   cri-
o://4b2625d4f61861e33cc95ad6d455915ea8ff6b75e17650538cc33c1e3e26aeb8
    Image:
quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51a
afbae73f2abd70a83d5fa173b
    Image ID:
quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51a
afbae73f2abd70a83d5fa173b
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 13 Feb 2022 14:14:25 -0500
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-6f844 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-6f844:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
    ConfigMapName:           openshift-service-ca.crt
    ConfigMapOptional:       <nil>
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason          Age   From               Message
  ----    ------          ----  ----               -------
  Normal  Scheduled       46s   default-scheduler  Successfully assigned user-getting-
started/parksmap-848bd4954b-5pvcc to ci-ln-fr1rt92-72292-4fzf9-worker-a-g9g7c
  Normal  AddedInterface  44s   multus             Add eth0 [10.131.0.14/23] from openshift-sdn
  Normal  Pulling         44s   kubelet            Pulling image
"quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51
aafbae73f2abd70a83d5fa173b"
```

```
  Normal  Pulled          35s   kubelet            Successfully pulled image
"quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51
aafbae73f2abd70a83d5fa173b" in 9.49243308s
  Normal  Created         35s   kubelet            Created container parksmap
  Normal  Started         35s   kubelet            Started container parksmap
```

추가 리소스

- oc describe
- oc get
- oc label
- Pod 보기
- Pod 로그 보기

4.5.3. 애플리케이션 스케일링

Kubernetes에서 **Deployment** 오브젝트는 애플리케이션이 배포하는 방법을 정의합니다. 대부분의 경우 사용자는 **Pod**,**Service**,**ReplicaSets** 및 **Deployment** 리소스를 함께 사용합니다. 대부분의 경우 OpenShift Container Platform이 리소스를 생성합니다.

national-parks-app 이미지를 배포하면 배포 리소스가 생성됩니다. 예에서는 하나의 **Pod**만 배포됩니다.

다음 절차에서는 두 개의 인스턴스를 사용하도록 **national-parks-image** 크기를 조정합니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있어야 합니다.
- 배포된 이미지가 있습니다.

절차

- 애플리케이션을 하나의 Pod 인스턴스에서 두 개의 Pod 인스턴스로 확장하려면 다음 명령을 입력합니다.

```
$ oc scale --current-replicas=1 --replicas=2 deployment/parksmap
```

출력 예

```
deployment.apps/parksmap scaled
```

검증

1. 애플리케이션이 올바르게 확장되도록 하려면 다음 명령을 입력합니다.

```
$ oc get pods
```

출력 예

```
NAME                  READY   STATUS    RESTARTS   AGE
parksmap-5f9579955-6sng8   1/1     Running   0          7m39s
parksmap-5f9579955-8tgft   1/1     Running   0          24s
```

2. 애플리케이션을 하나의 Pod 인스턴스로 다시 축소하려면 다음 명령을 입력합니다.

```
$ oc scale --current-replicas=2 --replicas=1 deployment/parksmap
```

추가 리소스

- oc scale

4.6. PYTHON 애플리케이션 배포

다음 절차에서는 **parksmap** 애플리케이션에 대한 백엔드 서비스를 배포합니다. Python 애플리케이션은 MongoDB 데이터베이스에 대해 2D 지리 공간 쿼리를 수행하여 전 세계 모든 국립공원의 지도 좌표를 찾고 반환합니다.

배포된 백엔드 서비스는 **nationalparks** 입니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있어야 합니다.
- 배포된 이미지가 있습니다.

절차

1. 새 Python 애플리케이션을 생성하려면 다음 명령을 입력합니다.

```
$ oc new-app python~https://github.com/openshift-roadshow/nationalparks-py.git --name
nationalparks -l 'app=national-parks-
app,component=nationalparks,role=backend,app.kubernetes.io/part-of=national-parks-
app,app.kubernetes.io/name=python' --allow-missing-images=true
```

출력 예

```
--> Found image 0406f6c (13 days old) in image stream "openshift/python" under tag "3.9-
ubi9" for "python"

    Python 3.9
    ----------
    Python 3.9 available as container is a base platform for building and running various
Python 3.9 applications and frameworks. Python is an easy to learn, powerful programming
language. It has efficient high-level data structures and a simple but effective approach to
object-oriented programming. Python's elegant syntax and dynamic typing, together with its
interpreted nature, make it an ideal language for scripting and rapid application development
in many areas on most platforms.

    Tags: builder, python, python39, python-39, rh-python39
```

```
    * A source build using source code from https://github.com/openshift-
roadshow/nationalparks-py.git will be created
      * The resulting image will be pushed to image stream tag "nationalparks:latest"
      * Use 'oc start-build' to trigger a new build

--> Creating resources with label app=national-parks-
app,app.kubernetes.io/name=python,app.kubernetes.io/part-of=national-parks-
app,component=nationalparks,role=backend ...
    imagestream.image.openshift.io "nationalparks" created
    buildconfig.build.openshift.io "nationalparks" created
    deployment.apps "nationalparks" created
    service "nationalparks" created
--> Success
```

2. **nationalparks** 애플리케이션을 공개할 경로를 생성하려면 다음 명령을 입력합니다.

```
$ oc create route edge nationalparks --service=nationalparks
```

출력 예

```
route.route.openshift.io/parksmap created
```

3. 생성된 애플리케이션 경로를 검색하려면 다음 명령을 입력합니다.

```
$ oc get route
```

출력 예

```
NAME            HOST/PORT                                                PATH   SERVICES
PORT       TERMINATION   WILDCARD
nationalparks    nationalparks-user-getting-started.apps.cluster.example.com
nationalparks   8080-tcp   edge          None
parksmap         parksmap-user-getting-started.apps.cluster.example.com
parksmap        8080-tcp   edge          None
```

추가 리소스

- oc new-app

4.7. 데이터베이스에 연결

national-parks-app 애플리케이션에서 위치 정보를 저장하는 MongoDB 데이터베이스를 배포 및 연결합니다. **national-parks-app** 애플리케이션을 맵 시각화 도구의 백엔드로 표시하면 **parksmap** 배포에서는 OpenShift Container Platform 검색 메커니즘을 사용하여 지도를 자동으로 표시합니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있어야 합니다.
- 배포된 이미지가 있습니다.

절차

- 데이터베이스에 연결하려면 다음 명령을 입력합니다.

```
$ oc new-app quay.io/centos7/mongodb-36-centos7 --name mongodb-nationalparks -e
MONGODB_USER=mongodb -e MONGODB_PASSWORD=mongodb -e
MONGODB_DATABASE=mongodb -e MONGODB_ADMIN_PASSWORD=mongodb -l
'app.kubernetes.io/part-of=national-parks-app,app.kubernetes.io/name=mongodb'
```

출력 예

```
--> Found container image dc18f52 (8 months old) from quay.io for
"quay.io/centos7/mongodb-36-centos7"

    MongoDB 3.6
    -----------
    MongoDB (from humongous) is a free and open-source cross-platform document-oriented
database program. Classified as a NoSQL database program, MongoDB uses JSON-like
documents with schemas. This container image contains programs to run mongod server.

    Tags: database, mongodb, rh-mongodb36

    * An image stream tag will be created as "mongodb-nationalparks:latest" that will track this
image

--> Creating resources with label app.kubernetes.io/name=mongodb,app.kubernetes.io/part-
of=national-parks-app ...
    imagestream.image.openshift.io "mongodb-nationalparks" created
    deployment.apps "mongodb-nationalparks" created
    service "mongodb-nationalparks" created
--> Success
```

추가 리소스

- oc new-project

4.7.1. 시크릿 생성

Secret 오브젝트는 암호, OpenShift Container Platform 클라이언트 구성 파일, 개인 소스 리포지토리 자격 증명 등과 같은 중요한 정보를 보유하는 메커니즘을 제공합니다. 보안은 Pod에서 중요한 콘텐츠를 분리합니다. 볼륨 플러그인을 사용하여 컨테이너에 보안을 마운트하거나 시스템에서 시크릿을 사용하여 Pod 대신 작업을 수행할 수 있습니다. 다음 절차에서는 시크릿 **nationalparks-mongodb-parameters**를 추가하고 이를 **nationalparks** 워크로드에 마운트합니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있어야 합니다.
- 배포된 이미지가 있습니다.

절차

1. 시크릿을 생성하려면 다음 명령을 입력합니다.

```
$ oc create secret generic nationalparks-mongodb-parameters --from-
literal=DATABASE_SERVICE_NAME=mongodb-nationalparks --from-
literal=MONGODB_USER=mongodb --from-literal=MONGODB_PASSWORD=mongodb --
from-literal=MONGODB_DATABASE=mongodb --from-
literal=MONGODB_ADMIN_PASSWORD=mongodb
```

출력 예

```
secret/nationalparks-mongodb-parameters created
```

2. mongodb 시크릿을 **nationalpartks** 워크로드에 연결하기 위해 환경 변수를 업데이트하려면 다음 명령을 입력합니다.

```
$ oc set env --from=secret/nationalparks-mongodb-parameters deploy/nationalparks
```

출력 예

```
deployment.apps/nationalparks updated
```

3. **nationalparks** 배포 상태를 표시하려면 다음 명령을 입력합니다.

```
$ oc rollout status deployment nationalparks
```

출력 예

```
deployment "nationalparks" successfully rolled out
```

4. **mongodb-nationalparks** 배포 상태를 표시하려면 다음 명령을 입력합니다.

```
$ oc rollout status deployment mongodb-nationalparks
```

출력 예

```
deployment "nationalparks" successfully rolled out
deployment "mongodb-nationalparks" successfully rolled out
```

추가 리소스

- oc create secret generic
- oc set env
- oc rollout status

4.7.2. 데이터 로드 및 국립 공원 지도 표시

parksmap 및 **nationalparks** 애플리케이션을 배포한 다음 **mongodb-nationalparks** 데이터베이스를 배포했습니다. 그러나 *데이터베이스에* 데이터가 로드되지 않았습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 대한 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있어야 합니다.
- 배포된 이미지가 있습니다.

절차

1. 국립 공원 데이터를 로드하려면 다음 명령을 입력합니다.

```
$ oc exec $(oc get pods -l component=nationalparks | tail -n 1 | awk '{print $1;}') -- curl -s
http://localhost:8080/ws/data/load
```

출력 예

```
"Items inserted in database: 2893"
```

2. 데이터가 올바르게 로드되었는지 확인하려면 다음 명령을 입력합니다.

```
$ oc exec $(oc get pods -l component=nationalparks | tail -n 1 | awk '{print $1;}') -- curl -s
http://localhost:8080/ws/data/all
```

출력 예(잘라냄)

```
, {"id": "Great Zimbabwe", "latitude": "-20.2674635", "longitude": "30.9337986", "name":
"Great Zimbabwe"}]
```

3. 경로에 레이블을 추가하려면 다음 명령을 입력합니다.

```
$ oc label route nationalparks type=parksmap-backend
```

출력 예

```
route.route.openshift.io/nationalparks labeled
```

4. 경로를 검색하여 지도를 보려면 다음 명령을 입력합니다.

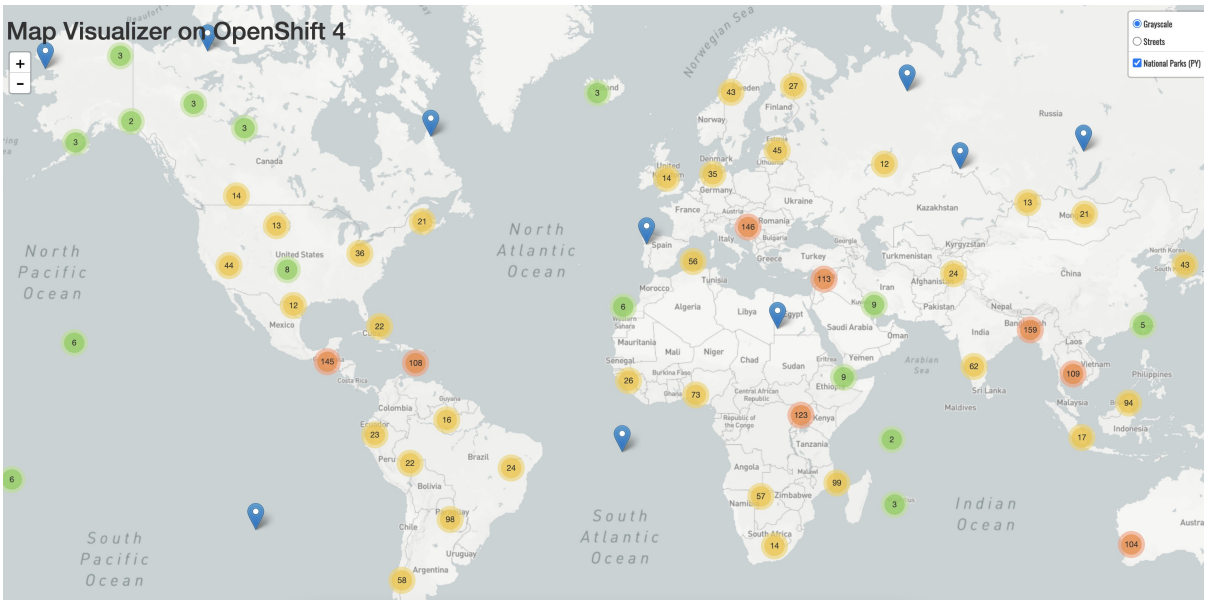
```
$ oc get routes
```

출력 예

```
NAME          HOST/PORT                                                PATH   SERVICES       PORT
TERMINATION   WILDCARD
nationalparks   nationalparks-user-getting-started.apps.cluster.example.com
nationalparks   8080-tcp   edge          None
parksmap        parksmap-user-getting-started.apps.cluster.example.com                parksmap
8080-tcp   edge          None
```

5. 위에서 검색한 **HOST/PORT** 경로를 복사하여 웹 브라우저에 붙여넣습니다. 사용 중인 브라우저에는 전 세계의 국립 공원 지도가 표시되어야 합니다.

그림 4.1. 전 세계의 국립공원



추가 리소스

- oc exec
- oc label
- oc get