ch08_쿠버네티스를_익히자

태영

쿠버네티스란?

- 쿠버네티스(Kubernetes)는 컨테이너 오케스트레이션 도구
  - 컨테이너 오케스트레이션이란, 시스템 전체를 통과하고 복수의 컨테이너를 관리하는 일
- 쿠버네티스를 일반 프로그레머가 관리하는 일은 드물다
  - 여러개의 서버(=컨테이너), 즉 대규모 시스템을 관리 할 일이 많지 않다.
  - 그러나, 쿠버네티스로 어떤 일을 할 수 있는가에 대한 지식은 개발에 유용하다

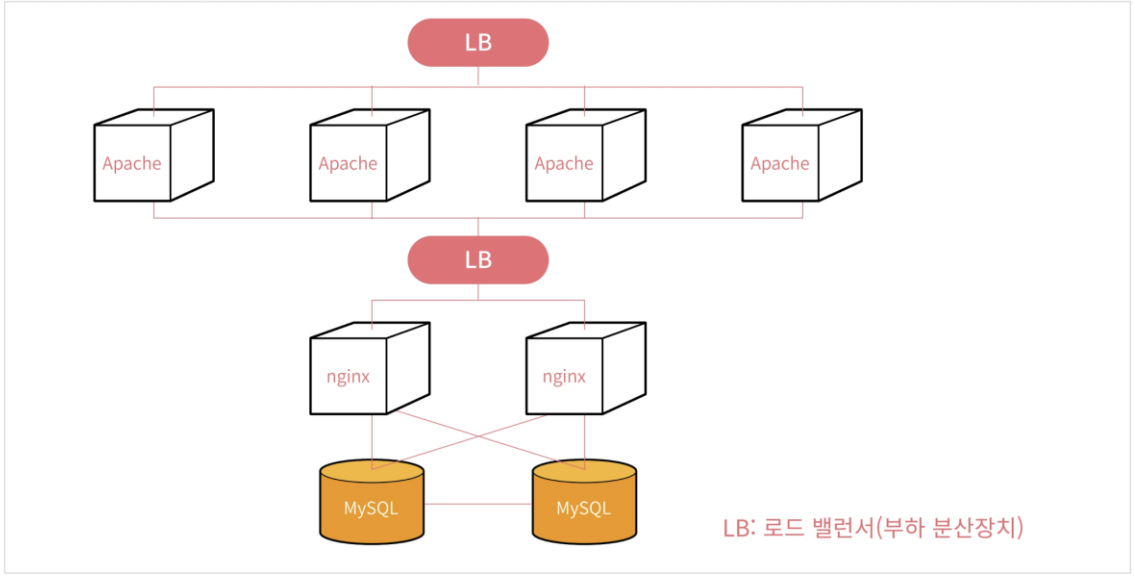


그림 8-1-2 쿠버네티스는 대규모 시스템에 적용되는 경우가 많다.

쿠버네티스는 여러 대의 컨테이너가 여러 대의 물리적 서버에 걸쳐 실행되는 것을 전제로 한다.

- 도커는 한 대의 물리적 서버에서 실행 되는 경우가 많지만, 쿠버네티스는 여러대의 물리적 서버의 존재를 전제로 한다
- 20대의 컨테이너를 만들려면 docker run 명령어를 20번 실행 해야 한다.
- 에러 발생 시, 물리적 서버를 하나 하나 모니터링 해 관리 하는 일은 매우 번거롭다.
- 쿠버네티스는 번거로운 컨테이너 생성이나 관리 수고를 덜어 주는 도구.
- 정의 파일(매니페스트 파일)만 작성하면, 모든 물리적 서버에 컨테이너 생성하고, 관리 해 준다

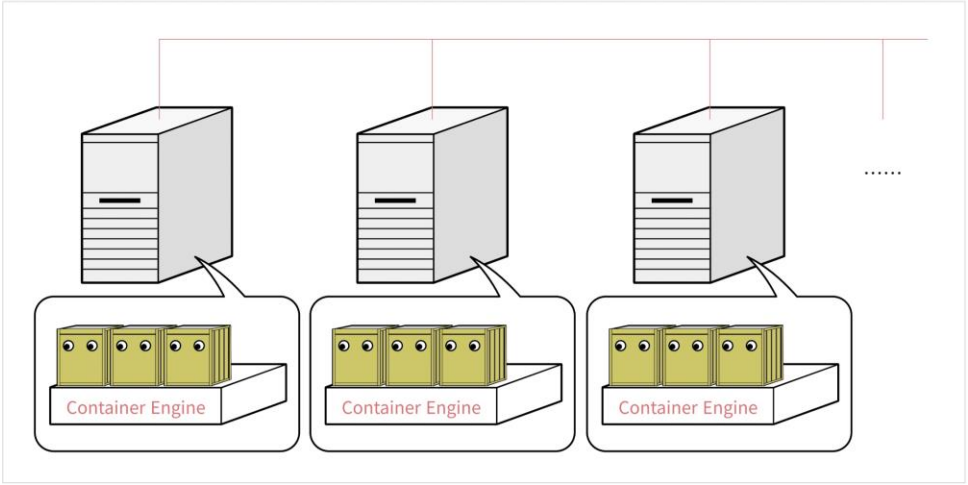


그림 8-1-3 쿠버네티스는 여러 대의 물리적 서버가 존재한다는 것을 전제로 한다.

마스터 노드와 워커 노드

- 클러스터의 구성
  - 마스터 노드 : 전체적인 제어를 담당, 마스터 노드에서는 컨테이너를 실행하지 않고, 워커 노드에서 실행되는 컨테이너를 관리, 컨테이너 엔진도 설치되지 않는다.
  - 워커 노드 : 실제 동작을 담당, 컨테이너가 실제 동작하는 서버, 컨테이너 엔진 설치 됨
  - 클러스터 : 마스터 노드와 워커 노드로 구성된 일군의 쿠버네티스 시스템, 사람 개입 없이 마스터 노드에 설정된 내용에 따라 워커 노드 관리

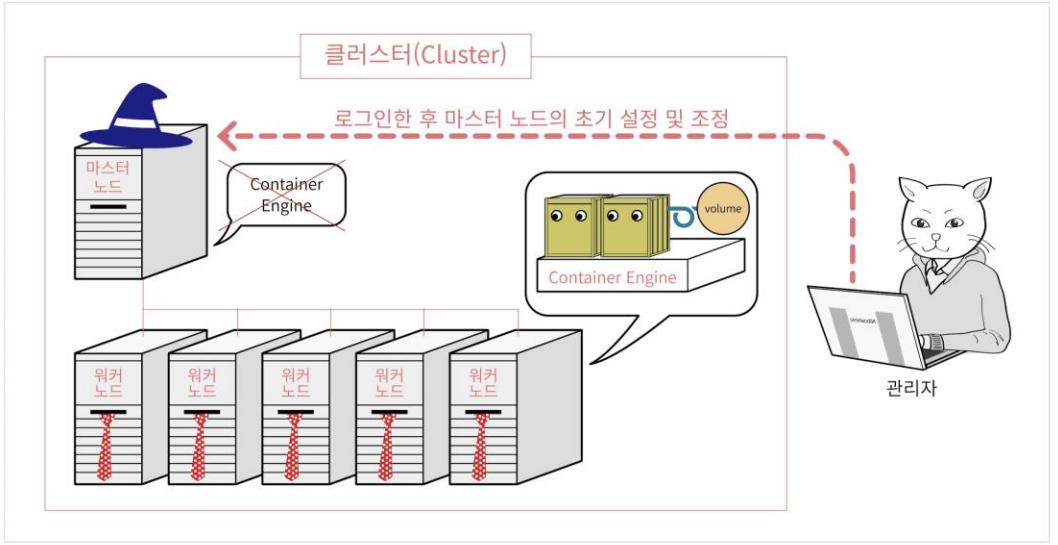


그림 8-2-1 클러스터는 설정 파일의 내용에 따라 자율적으로 동작한다.

쿠버네티스를 사용하려면 먼저 설치가 필요하다

- 쿠버네티스는 도커 엔진 등의 컨테이너 엔진과는 별로의 소프트웨어
- 따라서, 쿠버네티스 소프트웨어와 CNI(가상 네트워크 드라이버)를 설치 해야
  - CNI에는 플란넬, 칼리코, AWS VPC CNI 등
- 마스터 노드에 컨테이너 상태 관리를 위한 etcd 라는 데이터 베이스 설치
- 워커 노드에는 도커 엔진과 같은 컨테이너 엔진 필요
- 또, 마스터 노드를 설정하는 관리자의 컴퓨터에 kubectl 설치, 마스터 노드에 로그인해 초기 설정 및 추구 조정

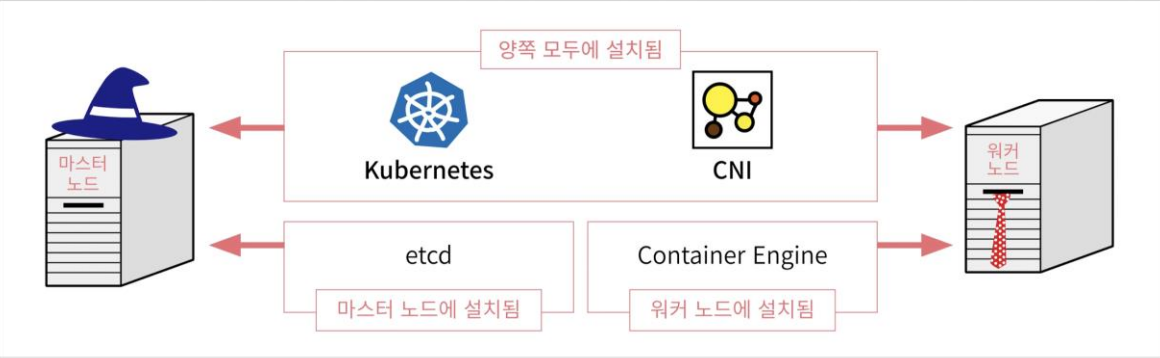


그림 8-2-2 마스터 노드, 워커 노드에는 각기 다른 소프트웨어가 설치된다.

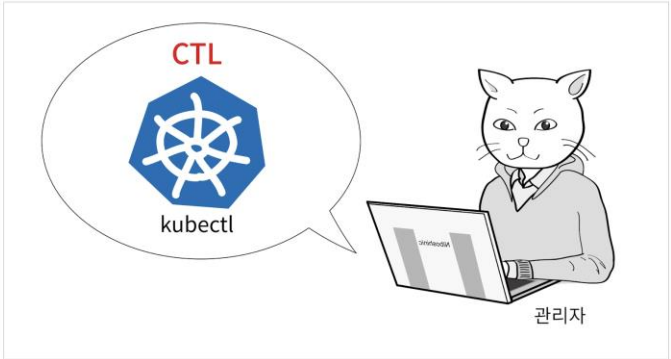


그림 8-2-3 관리자의 컴퓨터에는 kubectl을 설치해야 한다.

컨트롤 플레인(제어판)과 kube-let

- 마느터 노드는 컨트롤 플레인을 통해 워커 노드를 관리

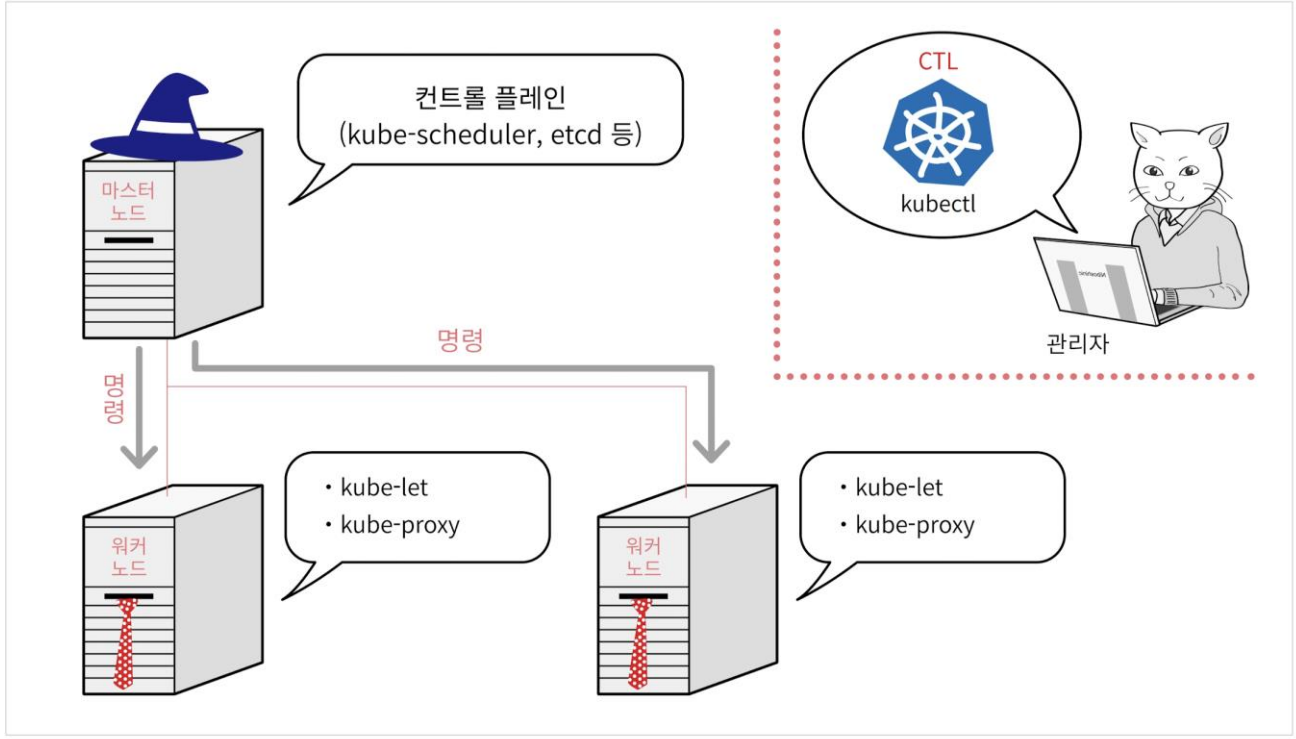


그림 8-2-4 쿠버네티스의 동작 구조

쿠버네티스는 항상 '바람직한' 상태를 유지 한다

- 쿠버네티스는, '컨테이너 OO개, 볼륨은 XX개로 구성하라'와 같이 어떤 '바람직한' 상태를 YAML 파일에 정의, 자동으로 컨테이너를 생성/삭제 하면서 관리 하며 '바람직한 ' 상태를 유지 하는 기능이 있다.
- 도커 컴포즈는, 컨테이너 수를 바꿀 수는 있어도 모니터링 기능이 없어 관리하지는 않는다.

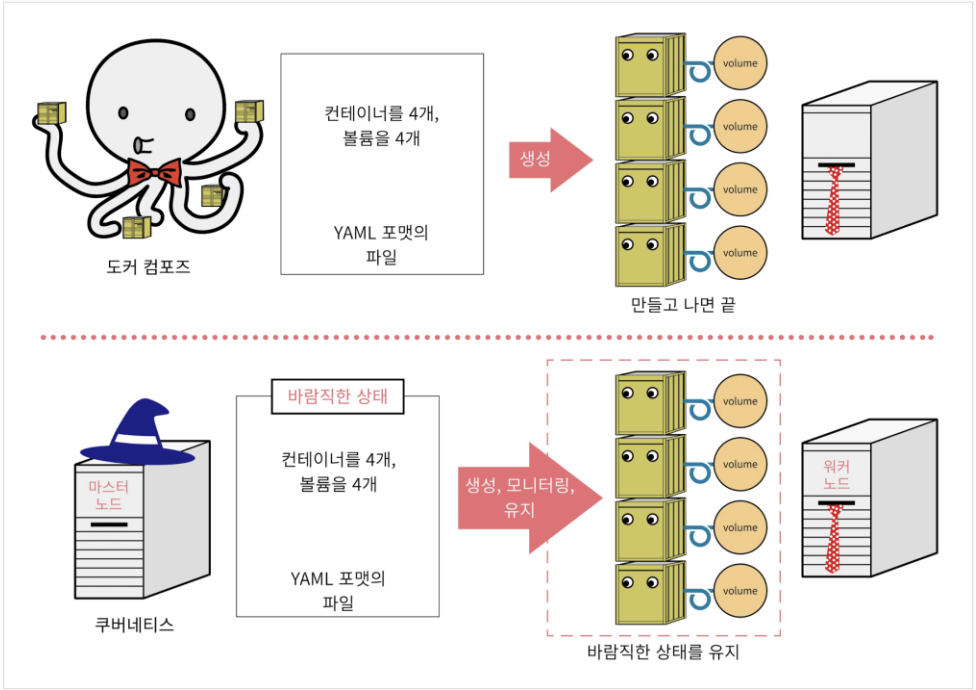


그림 8-2-5 도커 컴포즈와 쿠버네티스의 차이점

쿠버네티스는 항상 '바람직한' 상태를 유지 한다

- 컨테이너가 망가졌다면, 쿠버네티스가 알아서 망가진 컨테이너를 삭제하고 새 컨테이너로 대체한다.

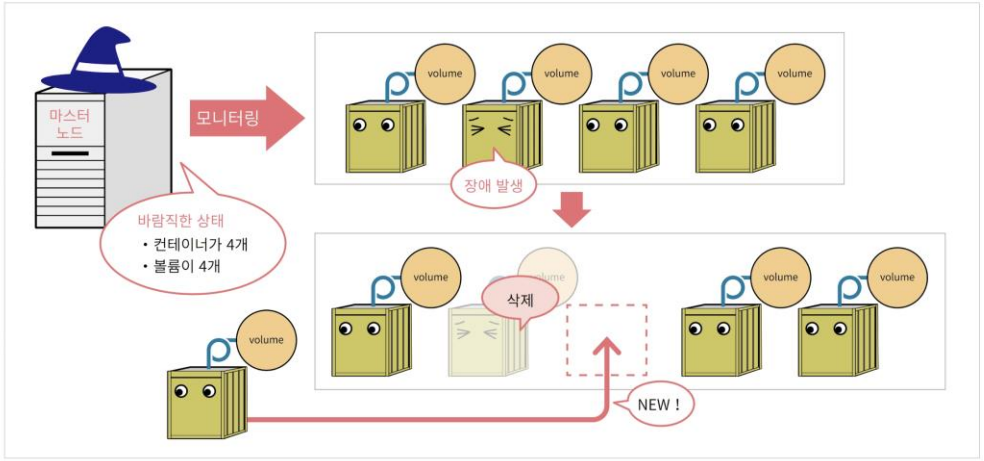


그림 8-2-6 컨테이너가 하나 망가지면 해당 컨테이너를 자동 삭제하고 새 컨테이너로 대체한다

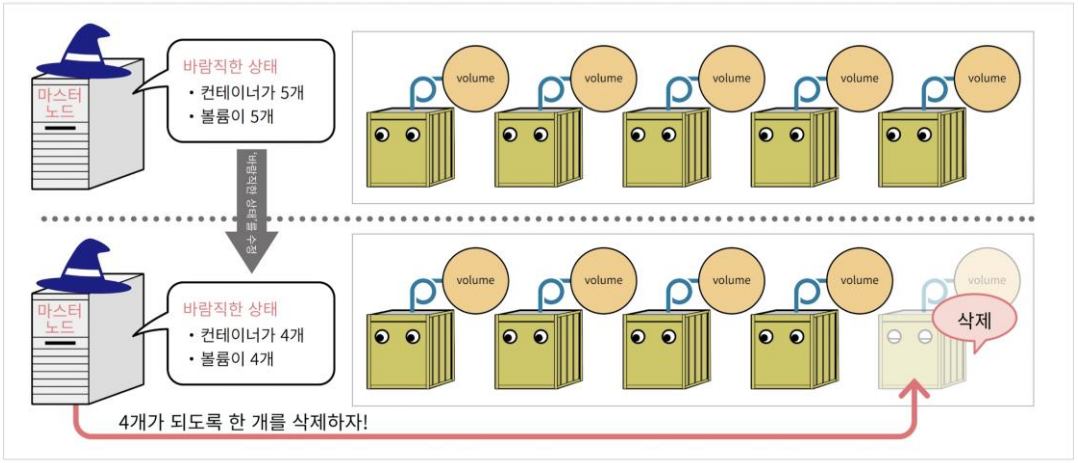


그림 8-2-7 바람직한 상태가 바뀌어도 삭제 또는 생성을 통해 정해진 상태를 유지한다.

쿠버네티스를 사용하는 시스템에서 컨테이너 삭제

- 컨테이너를 삭제 하고 싶다면, 직접 삭제 하는 것이 아니라, '바람 직한 상태'를 수정해야 한다.
- 쿠버네티스의 목표는 '바람 직한 상태'를 유지 하는 것으로, 사람이 개입해 컨테이너를 삭제 해서는 안된다.

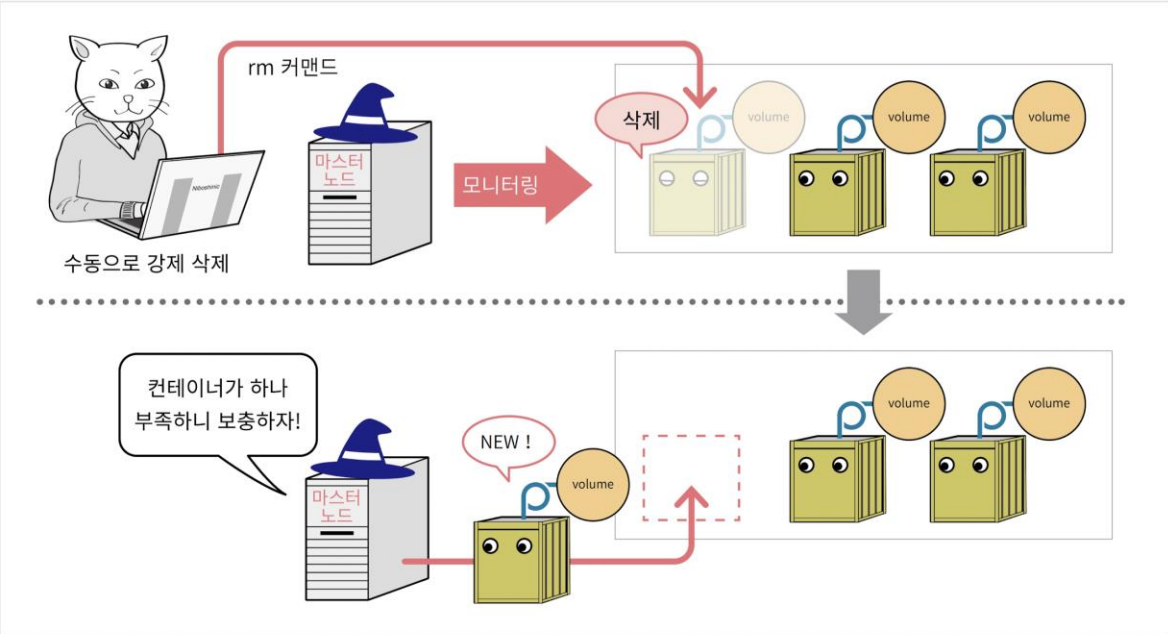
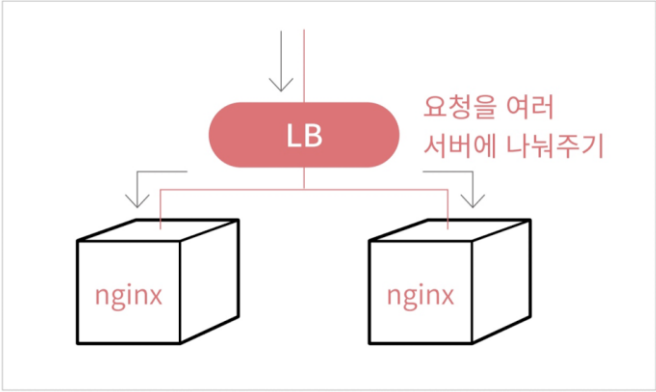


그림 8-2-8 관리자가 직접 컨테이너를 삭제해도 쿠버네티스가 대체 컨테이너를 생성한다.

로드 밸런서와 클라우드 컴퓨팅

- **로드 밸런서의 역할**
  - 서버에 대한 요청을 여러 서버에 분산하여 과부하를 방지
- **서버의 유연한 운영**
  - 요청량에 따라 서버 수를 늘리거나 줄임
  - 클라우드 컴퓨팅으로 자동 대응 가능
- **클라우드 컴퓨팅의 이점**
  - AWS, GCP 등은 서버를 자동으로 확장/축소
  - 대규모 시스템 운영 효율성 향상
- **쿠버네티스의 중요성**
  - 클라우드 환경의 서버와 컨테이너를 효율적으로 관리 및 확장
  - 자원 절약과 서버 자동 관리 가능



etcd의 역할

- **쿠버네티스와 도커 컴포즈의 차이점**:
  - 쿠버네티스의 매니페스트 파일의 정의에 따른 상태가 데이터베이스인 etcd에 저장되며, 도커 컴포즈와 차별화되는 중요한 특징입니다. (매니페스트 파일 자체가 저장되는 것은 아님)
- **파드의 관리**:
  - 쿠버네티스는 etcd에 저장된 정보를 바탕으로 파드를 관리하며, 이를 통해 커맨드를 통해 정의 파일을 수정할 수 있습니다.
  - 도커 컴포즈와 달리, 쿠버네티스에서는 정의 파일이 데이터베이스에 저장되기 때문에 컨테이너에 직접 손을 대면 문제가 발생할 수 있습니다.
- **운영 규칙의 중요성**:
  - 쿠버네티스에서 정의 파일과 etcd에 저장된 정보가 일치하도록 유지하는 것이 중요하며, 이를 위해 운영 규칙을 철저히 관리해야 합니다.

쿠버네티스의 구성과 관련 용어

- 파드는 컨테이너와 볼륨을 함께 묶은 것이다
  - 쿠버네티스에서 컨테이너는 파드(pod)라는 단위로 관리
  - 파드는 컨테이너와 볼륨을 묶은 것
  - 파드 하나에 컨테이너 하나 일 수도, 여러 개 일 수도, 없을 수도 있다.

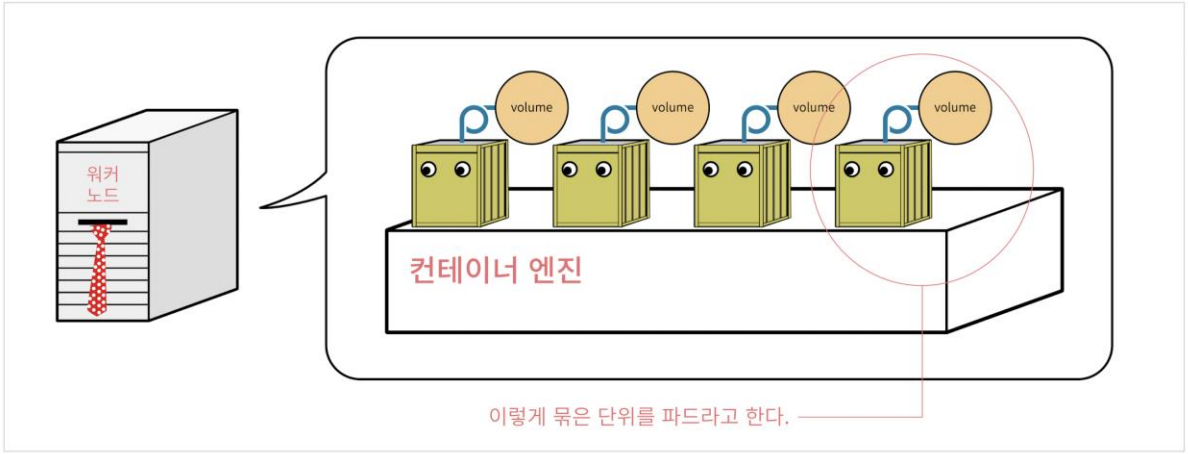


그림 8-3-1 파드는 컨테이너와 볼륨을 묶은 것이다.

파드가 모여 구성하는 서비스

- 서비스 : 여러 개의 파드를 이끄는 반장
  - 파드가 여러개의 워커 노드(물리적 서버)에 걸쳐 동작 하더라도, 서비스가 이들 모두를 관리 한다.
- 서비스의 역할은 로드 밸런서로, 각 서비스는 자동적으로 고정된 IP 주소(cluster IP)를 부여 받고, 이 주소로 들어오는 통신을 처리 한다.
  - 예, 워드프레스 파드를 관리하는 서비스는, 특정 파드에 요청이 몰리지 않게, 관리한다.
- 실제로는 로드 밸런서 또는 인그레이스가 여러 워커 노드 간의 분배 담당

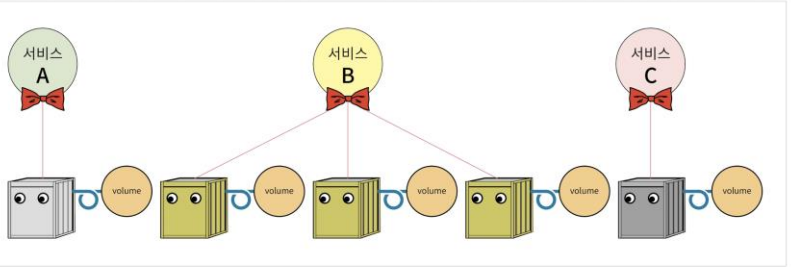


그림 8-3-2 같은 종류의 파드를 하나의 서비스가 관리한다.

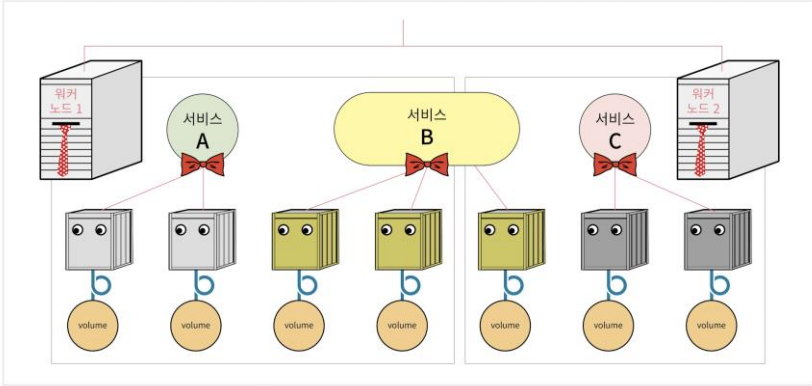


그림 8-3-3 여러 개의 워커 노드에 걸쳐 실행되더라도 동일한 구성의 파드는 하나의 서비스가 관리한다.

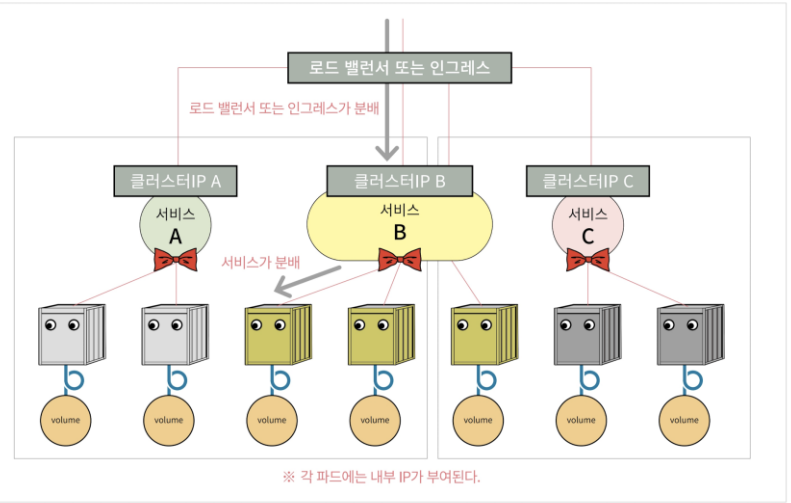


그림 8-3-4 서비스는 각 파드에 요청을 배분한다.

디플로이먼트와 레플리카세트

- 레플리카세트(ReplicaSet)은 파드의 수를 관리한다.
- 레플리카세트가 관리하는 동일한 구성의 파드를 레플리카 라고 한다.
  - 따라서, 파드의 수를 조정 하는 것을 '레플리카의 수를 조정' 한다 라고 한다.
- 레플리카세트는 단독으로 쓰이는 경우가 드물고 디플로이먼트(deployment)와 함께 쓰일 때가 많다.
- 디플로이먼트란, 파드의 디플로이(배포)를 관리하는 요소로, 파드가 사용하는 이미지 등 파드에 대한 정보를 가지고 있다.
  - 디플로이먼트 내에서 레플리카세트가 자동으로 생성된다.

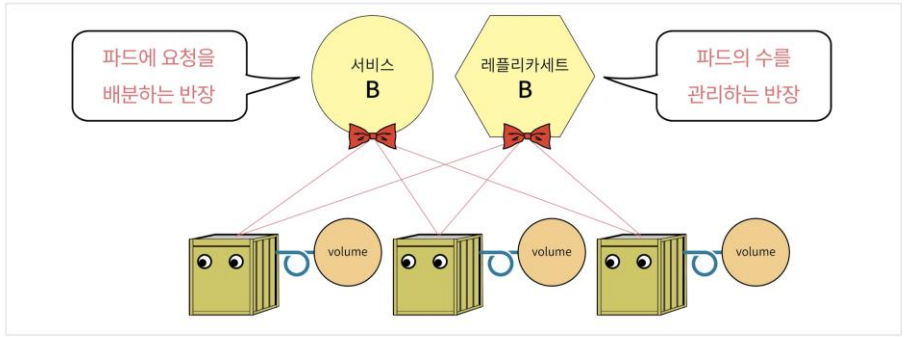


그림 8-3-5 레플리카세트는 파드의 수를 관리한다.

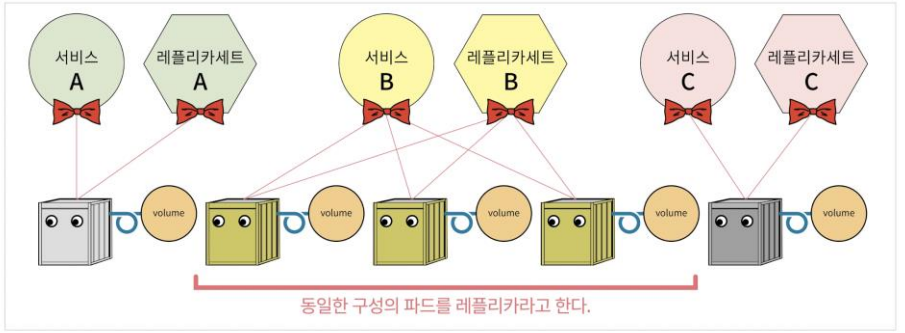


그림 8-3-6 레플리카세트가 관리하는 파드를 레플리카라고 한다.

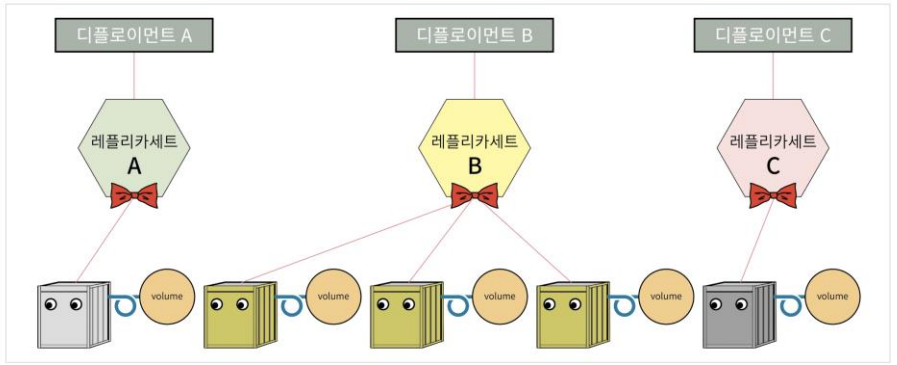


그림 8-3-7 디플로이먼트는 파드의 배포를 관리한다.

그 밖의 쿠버네티스 리소스

- 파드, 서비스, 디플로이먼트, 레플리카세트 등을 '리소스' 라고 한다.
- 리소스는 다음과 같이 50여종 있다.

주요 쿠버네티스 리소스

리소스 이름	내용
파드(pods)	파드. 컨테이너와 볼륨을 합친 것.
파드템플릿(podtemplates)	배포 시 파드의 형틀 역할
레플리케이션컨트롤러(replicationcontrollers)	레플리케이션을 제어
리소스쿼터(resourcequotas)	쿠버네티스 리소스의 사용량 제한을 설정
비밀값(secrets)	키 정보를 관리
서비스어카운트(serviceaccounts)	리소스를 다루는 사용자를 관리
서비스(services)	파드에 요청을 배분
데몬세트(daemonsets)	워커 노드마다 하나의 파드를 생성
디플로이먼트(deployments)	파드의 배포를 관리
레플리카세트(replicasets)	파드의 수를 관리
스테이트풀세트(statefulsets)	파드의 배포를 상태를 유지하며 관리
크론잡(cronjobs)	지정된 스케줄대로 파드를 실행
잡(jobs)	파드를 한번 실행

쿠버네티스 용어 정리

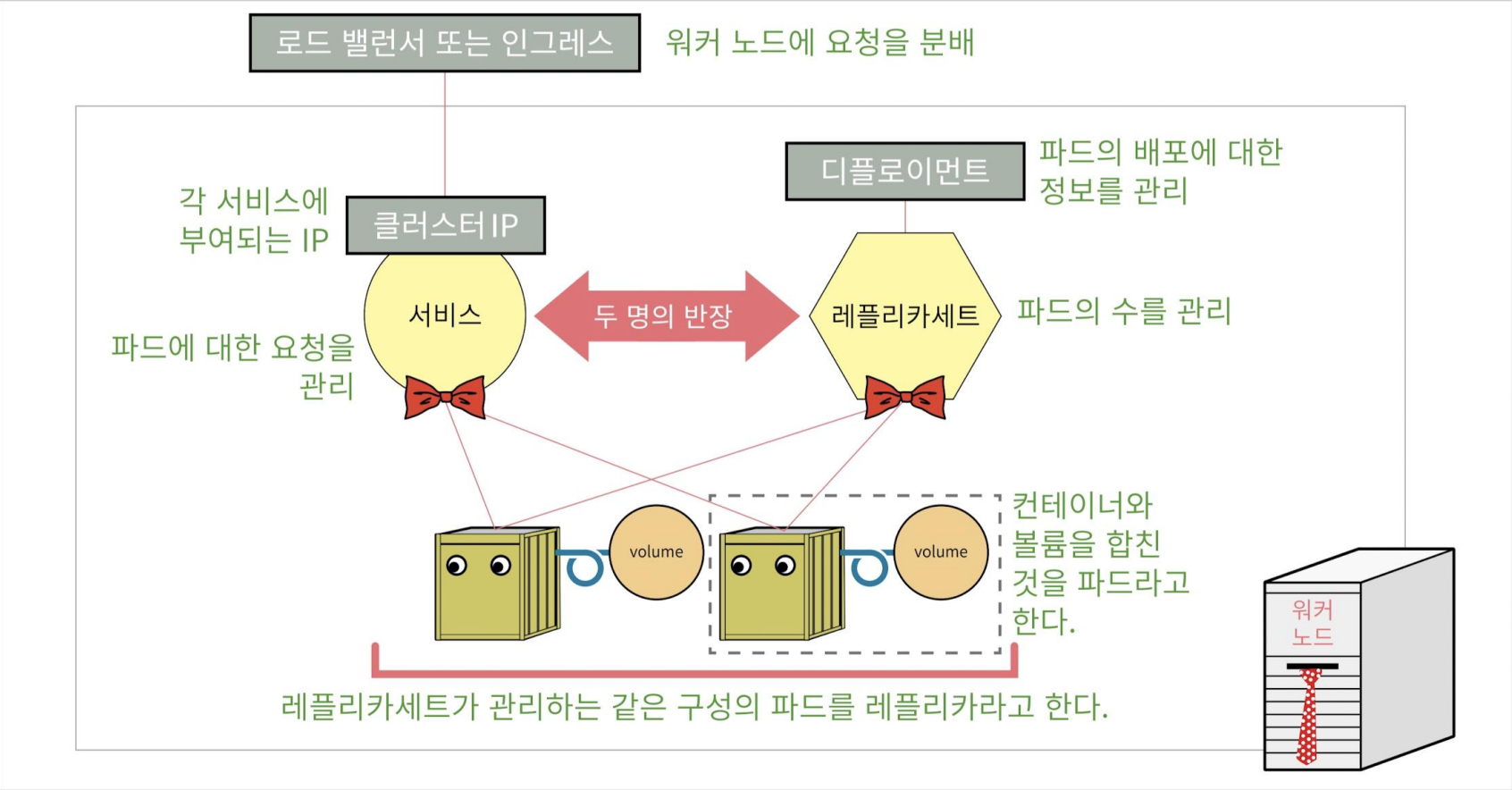
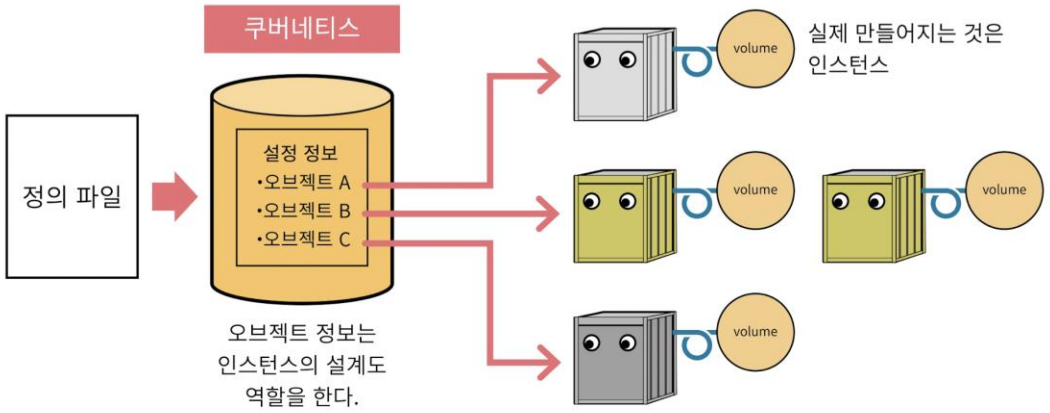


그림 8-3-8 용어 정리

오브젝트와 인스턴스

1. **오브젝트란?**
    1. 파드, 서비스, 레플리카, 디플로이먼트 등은 각각 "파드 오브젝트", "서비스 오브젝트"와 같이 "OO 오브젝트"라고 불립니다.
    2. 이러한 오브젝트는 쿠버네티스의 정의 파일에 기록되고, etcd 데이터베이스에 등록되어 관리됩니다.
2. **인스턴스란?**
    1. 쿠버네티스는 정의 파일에 따라 etcd에 등록된 오브젝트 정보를 기반으로 실제 파드나 서비스를 생성합니다. 이 실제로 생성된 것을 "인스턴스"라고 부릅니다.
    2. 데이터베이스에 저장된 오브젝트 정보는 인스턴스의 설계도와 같은 역할을 합니다.
3. **전체 과정 요약**
    1. 정의 파일에 설정된 오브젝트 정보는 etcd에 저장됩니다.
    2. 쿠버네티스는 이 정보를 바탕으로 인스턴스를 실제로 생성합니다.



서버 한 대로도 쿠버네티스를 사용할 수 있을까?

1. **쿠버네티스의 필요성**:
    1. 쿠버네티스는 대규모 서비스에서 여러 대의 서버를 관리하고 자동화된 관리를 가능하게 해주기 때문에 주로 사용됩니다.
    2. 하지만 소규모 서비스에서도 쿠버네티스는 유용할 수 있습니다. 장애가 발생해도 자동으로 파드를 생성해 대체하기 때문에 관리가 편리해집니다.
2. **소규모 서비스에서의 장점**:
    1. 설정만 잘 되어 있으면, 소규모 서비스에서도 쿠버네티스는 유용하며, 관리가 극대화될 수 있습니다. 특히 클라우드 환경에서는 이러한 자동화 기능이 매우 편리합니다.
3. **표준화된 환경 제공**:
    1. 쿠버네티스는 표준화된 컨테이너 실행 환경을 제공하기 때문에, 시스템의 일관성을 유지하면서 쉽게 설정 정보를 배포할 수 있습니다.
    2. 설치 후 설정이 간단하여, 복잡한 시스템에서도 유용합니다.

쿠버네티스 설치 및 사용법

- 쿠버네티스의 종류
  - 쿠버네티스는 클라우드 네이티브 컴퓨팅 재단(Cloud Native Computing Foundation, CNCF)에서 제정한 표준
  - 본래 구글에서 개발, CNCF를 조직, 여기에 쿠버네티스 기부, 오픈 소스로 개발, 급속하게 보급
  - AWS, Azure, GCP 같은 클라우드 서비스에서 자사 서비스에 맞는 커스텀된 쿠버네티스 제공

원조 쿠버네티스와 클라우드 버전

- 원조 쿠버네티스를 채택하는 것 자체는 꽤 흔하지만, 이를 '직접 구축' 하는 것은 별개의 문제, 대개는 외주로 구축
- 서비스마다 부여되는 클러스터IP에 로드 밸런싱을 적용 하려면 이를 지원하는 전용 하드웨어 하고, 드라이버에도 상성 문제가 있어, '인프라 전문 업체'에 맡길 수 밖에 없고 유지 보수도 힘들다.
- 관련 정보도 모으기 힘들고, 인하우스 커스터마이닝이 흔한 회사가 아닌 이상, 불안정한 내부 서비스를 쓰느니 인프라 전문 업체에 비용을 태워 맡기는 것이 낫다.
- 일반적으로 AWS 같은 클라우드 컴퓨팅 서비스를 사용 구축 한다.
  - AWS의 EC2나 Fargate를 워커 노드로 사용하고 EKS로 관리

험난한 길의 구세주! 도커 데스크톱과 Minikube

- 윈도우와 맥에서는 도커 데스크탑, 리눅스에서는 Minikube 사용
- 서버 한대에 마스터와 워커 노드 모두 구축
- 그러나 이 것은 모두 학습 용, 안정성과 대규모, 대용량 시스템에 사용하는 쿠버네티스와 명령어를 안다고 동일한 것은 아니다.
- 실제 쿠버네티스를 배우는 것은 험난 하다. 우선 명령어, 정의 파일(매니페스트 파일)을 익히며 한발짝 내딛자.

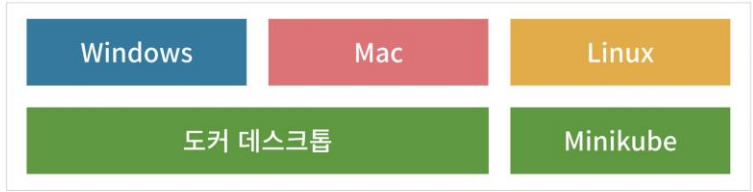


그림 8-4-3 학습 목적에 적합한 쿠버네티스

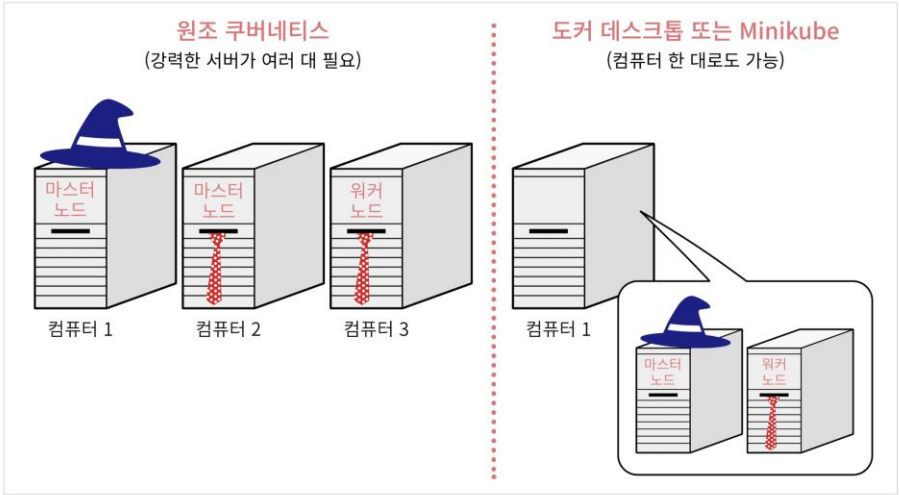


그림 8-4-4 도커 데스크톱이나 Minikube를 사용하면 컴퓨터 한 대로도 쿠버네티스를 배울 수 있다.

Kubeadm: 복수의 물리 서버에 쿠버네티스 설치

- Kubeadm이란?
  - **Kubeadm**은 여러 대의 물리 서버에 **쿠버네티스 클러스터**를 손쉽게 구축하고 관리할 수 있는 도구
  - 마스터 노드와 워커 노드를 설정하고 클러스터를 초기화할 수 있도록 돕는다
- Kubeadm 설치 과정
  - 환경 준비
    - 마스터 및 워커 노드 준비 (물리 서버 또는 가상 머신)
    - 각 서버에 리눅스 기반 OS와 필수 구성 요소(Kubernetes, CNI, etcd) 설치
  - 마스터 노드 초기화
    - kubeadm init 명령어로 마스터 노드에서 클러스터 초기화
  - 워커 노드 연결
    - kubeadm join 명령어로 워커 노드를 마스터 노드에 연결하여 클러스터 확장

매니페스트 파일(정의 파일)

- 매니페스트 파일이란?
  - 쿠버네티스에서 파드를 생성하고 서버 환경을 유지하기 위해 작성된 정의 파일.
  - 매니페스트 파일에 기록된 내용은 etcd에 저장되고, 서버 환경을 원하는 상태로 유지.

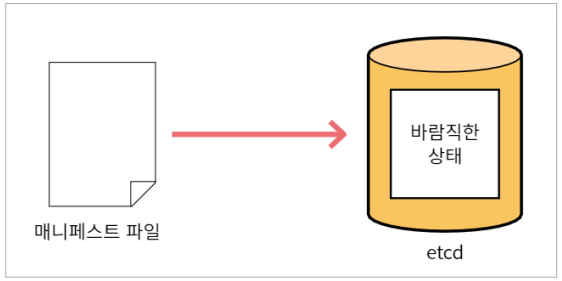


그림 8-5-1 매니페스트 파일을 작성해 etcd에 등록한다.

- 매니페스트 파일 형식
  - YAML 또는 JSON 형식으로 작성 가능.
  - 사람이 읽기 쉽게 YAML 형식을 주로 사용.
- 파일 이름 작성
  - 매니페스트 파일의 이름은 특정 규칙 없이 지정 가능하나, 프로젝트와 관련된 의미 있는 이름 사용 권장.
  - 주의: 'Kubernetes', 'test', 'Docker'와 같은 일반적 이름을 피하고, 특정 조직명이나 시스템명을 붙이지 않도록 유의.
- 쿠버네티스 정의 파일 예시
  - 파일 형식: YAML(.yml)
  - 파일 이름: 의미 있는 이름 사용

매니페스트 파일은 리소스 단위로 작성

- 주요 리소스
  - 쿠버네티스 매니페스트 파일은 파드, 서비스, 디플로이먼트, 레플리카세트 등 리소스를 단위로 작성.
  - 초보자 수준에서는 주로 서비스와 디플로이먼트를 다룸.
- 파드 항목 사용
  - 파드 항목은 따로 작성하지 않으며, 실제로는 디플로이먼트나 레플리카세트에서 파드를 생성하고 관리.
  - 파드 자체에 대한 개수 유지 기능은 없으며, 디플로이먼트나 레플리카세트가 이를 담당.
- 예를 들어, 아파치 서버를 배포할 경우, 아파치 디플로이먼트와 아파치 서비스 리소스를 작성하면 됨.

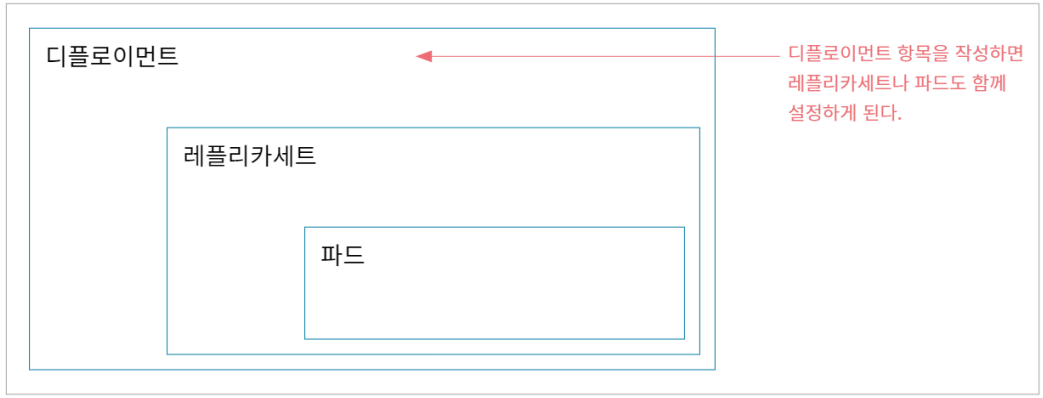


그림 8-5-3 '레플리카세트'와 '파드'는 '디플로이먼트'에서 설정할 수 있다.

매니페스트 파일은 여러 파일로 분할 가능

- 파일 분할 및 합쳐서 작성
  - 매니페스트 파일은 리소스 단위로 분할하거나, 한 파일에 합쳐서 작성할 수 있음.
  - 분할 작성 시, 각 리소스는 별도의 파일로 작성되고, 합쳐서 작성 시, 리소스를 ---로 구분하여 하나의 파일에 작성.
- 예시
  - 분할 작성: 디플로이먼트와 서비스 각각 따로 파일 작성 (예: apa000dep.yml, apa000ser.yml)
  - 합쳐서 작성: 디플로이먼트와 서비스를 한 파일에 작성하며 ---로 구분.

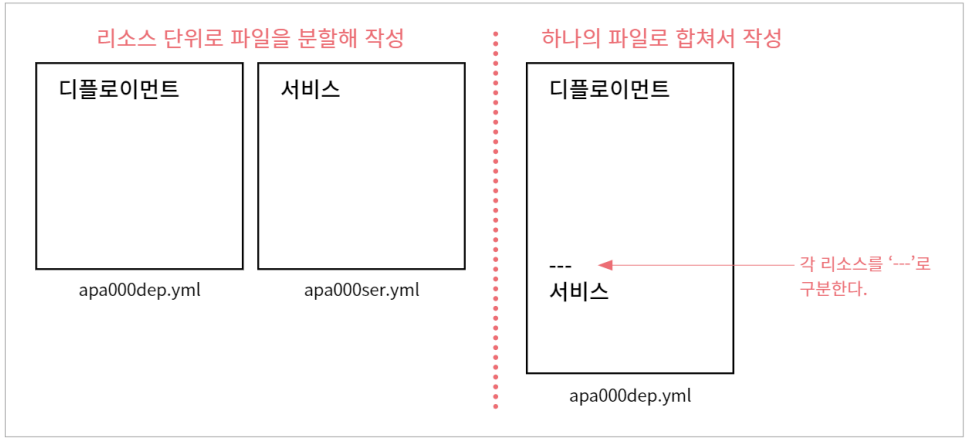


그림 8-5-4 매니페스트 파일은 리소스 단위로 분할하거나 한 파일로 작성할 수 있다.

매니페스트 파일로 작성할 내용

- 주요 항목
  - apiVersion: API 그룹 및 버전
  - kind: 리소스 유형
  - metadata: 메타데이터 작성 (리소스 이름, 레이블 등)
  - spec: 리소스의 구체적 설정 정보

자주 사용되는 리소스의 API 그룹 및 리소스 유형

리소스	API 그룹 / 버전	리소스 유형
파드	core/v1(v1으로 축약 가능)	Pod
서비스	core/v1(v1으로 축약 가능)	Service
디플로이먼트	apps/v1	Deployment
레플리카세트	apps/v1	ReplicaSet

주요 메타데이터

항목	내용
name	리소스의 이름. 문자열로 된 유일 식별자.
namespace	리소스를 세분화한 DNS 호환 레이블
uid	유일 식별자
resourceVersion	리소스 버전
generation	생성 순서를 나타내는 번호
creationTimestamp	생성 일시
deletionTimestamp	삭제 일시
labels	임의의 레이블
anotation	리소스에 설정할 값. 선택 대상은 되지 못한다.

메타데이터와 스펙 작성 (1) - 파드

- 파드의 메타데이터 및 스펙 주요 항목
  - metadata:
    - name: 파드 이름
    - labels: 파드 레이블
  - spec:
    - containers: 컨테이너 구성
      - name: 컨테이너 이름
      - image: 컨테이너 이미지
      - ports: 포트 설정

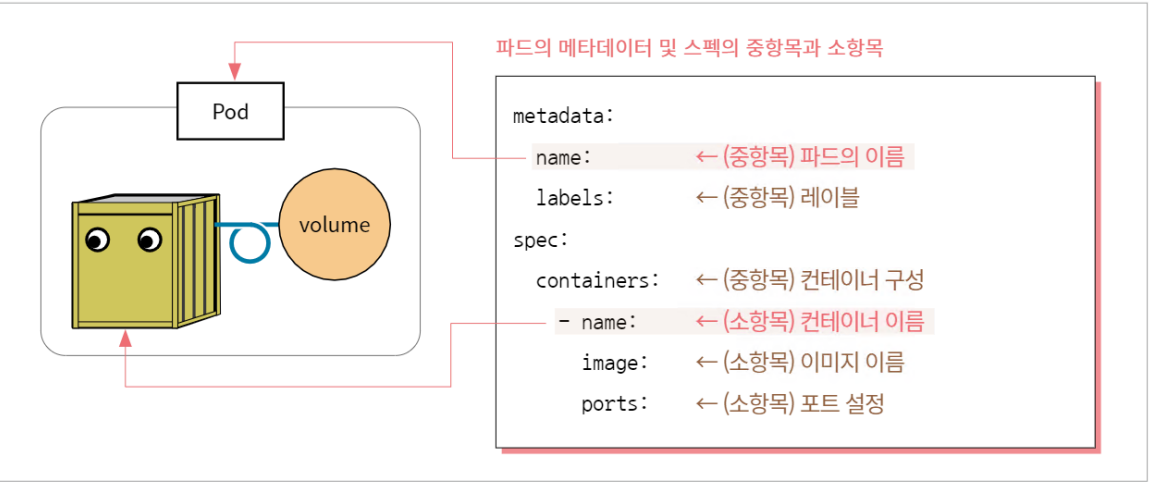
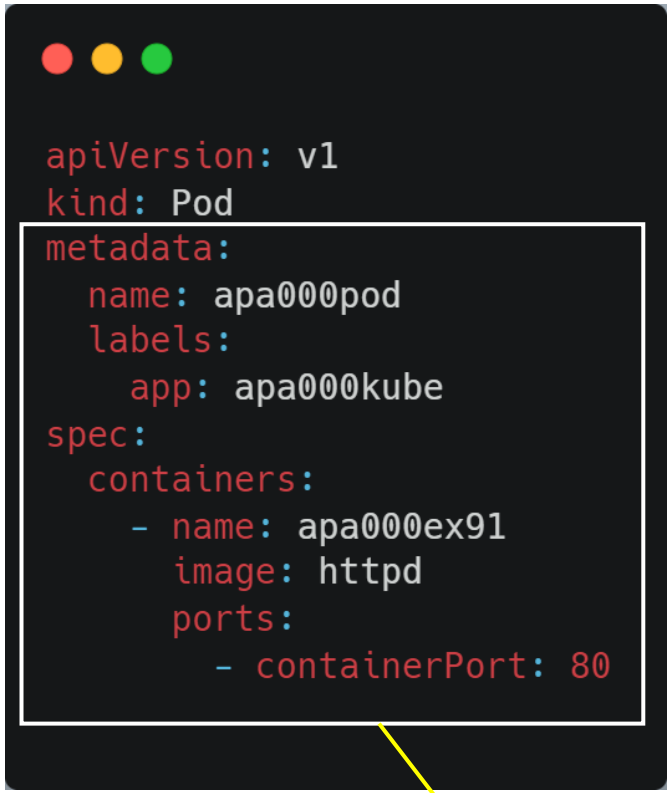


그림 8-5-7 metadata의 name 항목과 containers의 name 항목은 다른 것이다.

[실습] 매니페스트 파일 작성 (1) - 파드

- 단계 1: 주요 항목 정의
  - apiVersion, kind, metadata, spec 필드 구성
  - apiVersion: API 버전 (v1)
  - kind: 리소스 유형 (Pod)
- 단계 2: 메타데이터 설정
  - 파드 이름 설정: apa000pod
  - 레이블 설정: app: apa000kube
- 단계 3: 스펙 정의
  - 컨테이너 정보:
    - 이름: apa000ex91
    - 이미지: httpd
    - 포트: 80
- 단계 4: 확인 및 저장
  - 파일을 확인하고 저장



뒤에 사용 되는 부분

메타데이터와 스펙 작성(2) - 디플로이먼트

- Selector 설정
  - 디플로이먼트가 특정 레이블이 부여된 파드를 관리할 수 있도록 설정.
  - matchLabels: 파드를 선택할 때 사용하는 레이블, 템플릿 아래 metadata에서 기재.
- Replicas 설정
  - 파드의 복제 개수를 설정하는 부분.
  - 파드가 몇 개로 유지될지 결정 (0으로 설정 시, 파드가 사라짐).
- Template 작성
  - 생성할 파드의 정보를 기재.
  - metadata 및 spec은 거의 동일하게 기재하며, 레이블 관리를 위한 설정도 포함.

디플로이먼트의 항목

```
apiVersion:
kind:
metadata:
  name:              ← (중항목) 디플로이먼트 이름
spec:
  selector:          ← (중항목) 셀렉터 설정
    matchLabels:     ← (소항목) 셀렉터가 선택할 관리 대상 레이블
  replicas:          ← (중항목) 레플리카 설정
  template:          ← (중항목) 템플릿 (파드의 정보)
    metadata:        ← (소항목) 파드의 메타데이터를 기재
    spec:            ← (소항목) 파드의 스펙을 기재
```

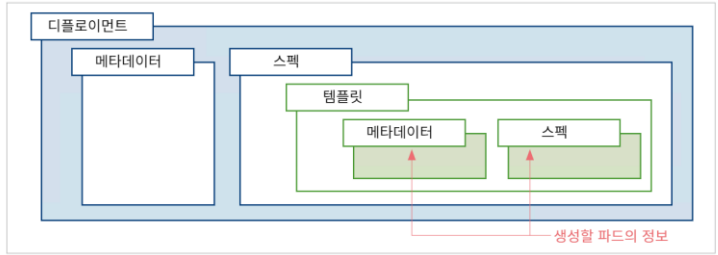


그림 8-5-10 디플로이먼트 정의 파일의 열개

[실습] 매니페스트 파일 작성 (2) - 디플로이먼트

- 주 항목 기재
  - 필요한 주요 항목(apiVersion, kind, metadata, spec) 기재.
- apiVersion 및 kind 설정
  - apiVersion: apps/v1
  - kind: Deployment
- metadata 설정
  - 디플로이먼트 이름: apa000dep
- Selector와 레플리카 수 설정
  - selector 항목에서 관리할 파드를 지정하는 레이블을 설정: app: apa000kube
  - replicas 항목에서 컨테이너 복제 수 설정: replicas: 3
- Template 항목에 파드 매니페스트 파일 내용 복사
  - 템플릿 안에 파드 매니페스트 정보를 그대로 복사해서 붙여 넣기

파드의 매니페스트 일부

메타데이터와 스펙 작성 (3) - 서비스

- 유형(type) 설정
  - 서비스의 종류를 정의하며, 외부에서 서비스에 어떻게 접속할 수 있을지 설정하는 항목.

	유형 이름	내용
①	ClusterIP	클러스터IP를 통해 서비스에 접근하도록 함(외부에서는 접근 불가)
②	NodePort	워커 노드의 IP를 통해 서비스에 접근하도록 함
③	LoadBalancer	로드밸런서의 IP를 통해 서비스에 접근하도록 함
④	ExternalName	파드에서 서비스를 통해 외부로 나가기 위한 설정

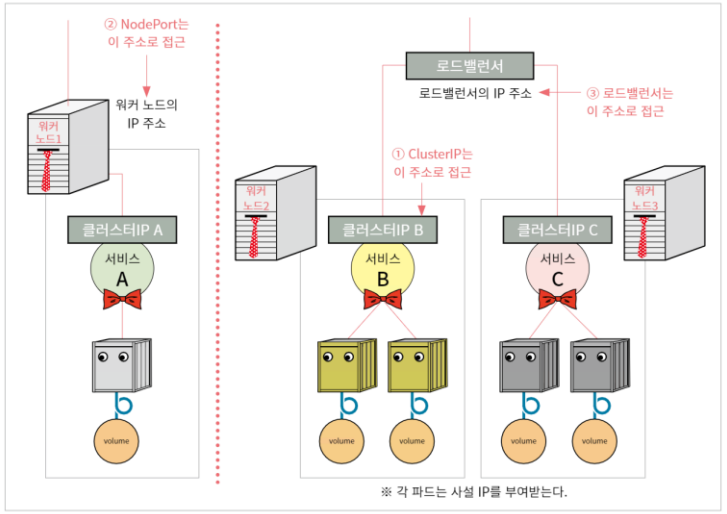


그림 8-5-13 유형 설정 항목과 의미

- 포트 설정
  - 서비스와 컨테이너 간의 통신을 위해 각각의 포트를 지정하여 외부 또는 내부에서 접근할 수 있도록 구성하는 작업
- 셀렉터 설정
  - 셀렉터는 서비스가 특정 레이블이 부여된 파드를 관리하도록 설정.
  - 디플로이먼트에서 설정된 레이블을 사용하여, 서비스에서 파드를 선택.
  - 디플로이먼트에서 matchLabels:가 필수 항목인 반면, 서비스에서는 선택 사항.

	항목	내용
①	port	서비스의 포트
②	nodePort	워커 노드의 포트
③	targetPort	컨테이너 포트

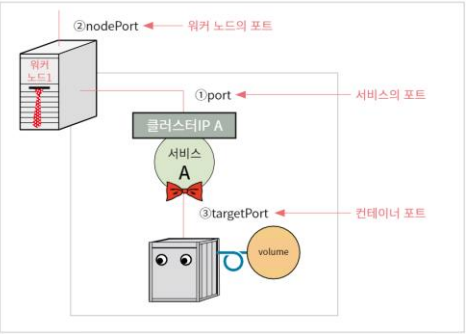


그림 8-5-14 포트 설정 항목의 의미

[실습] 메타데이터와 스펙 작성 (3) - 서비스

- 주항목 기재
  - 필수 항목: apiVersion, kind, metadata, spec 설정.
- apiVersion 및 kind 설정
  - apiVersion 값: 'v1'
  - kind 값: 'Service'
- metadata 설정
  - 서비스 이름 설정: name: apa000ser
- spec 설정
  - 외부 요청을 특정 포트를 통해 내부 파드로 전달하는 서비스 구성을 정의

```yaml
apiVersion: v1
kind: Service
metadata:
  name: apa000ser
spec:
  type: NodePort
  ports:
    - port: 8099
      targetPort: 80
      protocol: TCP
      nodePort: 30080
  selector:
    app: apa000kube
```

쿠버네티스 명령어

- 매니페스트 파일을 작성한 후, 쿠버네티스에 적용하려면 kubectl 명령어를 사용.

kubectl 명령어의 형식

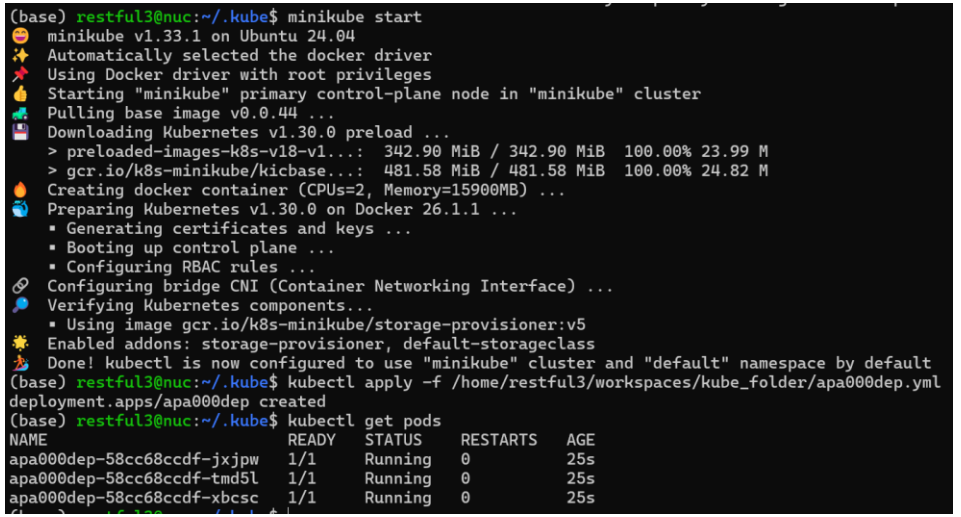
```
kubectl 커맨드 옵션
```

- kubectl은 쿠버네티스 클러스터에 명령을 내리는 방식으로, 도커 엔진에 명령을 내리는 것과 유사.
- 초보자는 apply와 delete 명령어를 중점적으로 익히는 것이 좋으며, 나머지 명령어는 상황에 맞게 사용할 수 있다.
- 쿠버네티스는 도커와 달리 매니페스트 파일을 사용하여 모든 리소스를 한 번에 생성하고, 시스템 상태를 자동으로 유지 관리
  - 사용자가 수작업으로 명령을 자주 입력할 필요가 없다.

커맨드	내용
create	리소스 생성
edit	리소스 편집
delete	**리소스 삭제**
get	리소스 상태 출력
set	리소스 값 설정
apply	**리소스 변경 사항 반영**
describe	상세 정보 확인
diff	리소스 상태 비교
expose	파드에 새로운 서비스 생성
scale	레플리카 수 변경
autoscale	자동 스케일링 적용
rollout	롤아웃 수행
exec	컨테이너 내에서 명령 실행
run	컨테이너 내에서 명령을 한 번 실행
attach	컨테이너 접속
cp	컨테이너 내 파일 복사
logs	컨테이너 로그 출력
cluster-info	클러스터 상세 정보 출력
top	시스템 자원(CPU, 메모리 등) 확인

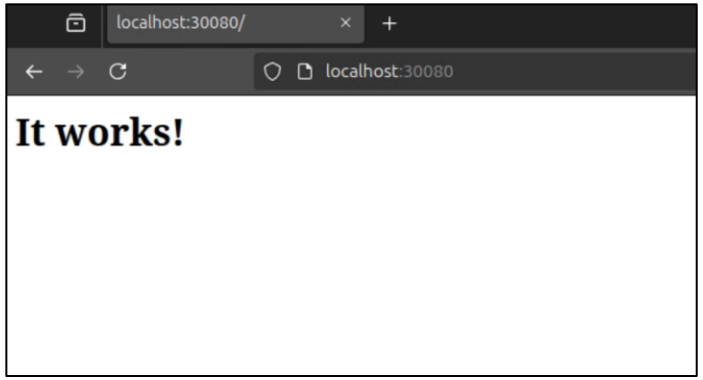
[실습] 매니페스트 파일로 파드 생성 (1) - 디플로이먼트

- 매니페스트 파일을 사용하여 디플로이먼트를 통해 파드를 생성.
- kubectl apply 명령어로 매니페스트 파일의 내용을 실제 리소스에 반영.
- 디플로이먼트로 생성된 파드는 직접 접근이 불가능하므로, 파드의 목록을 통해 생성 여부를 확인.
- 1단계: 디플로이먼트 매니페스트 파일 읽어들이기
  - 매니페스트 파일을 쿠버네티스에 적용하여 리소스를 생성하기 위해 kubectl apply -f <파일경로> 명령어를 사용.
  - 실행 결과: deployment.apps/apa000dep created (디플로이먼트가 성공적으로 생성됨).
- 2단계: 파드 생성 확인
  - kubectl get pods 명령어로 파드의 상태를 확인.
  - 파드 목록에 세 개의 파드가 생성되었으나, ImagePullBackOff 상태로 이미지 다운로드에 실패한 것을 확인 가능.



[실습] 매니페스트 파일로 파드 생성(2) - 서비스

- 서비스의 매니페스트 파일을 읽어들이기
  - 매니페스트 파일(apa000ser.yml)을 쿠버네티스에 적용하여 그 내용을 리소스에 반영합니다.
- 서비스가 잘 생성됐는지 확인
  - 서비스의 목록을 확인하여 생성된 서비스가 정상적으로 반영되었는지 확인합니다.




```
(base) restful3@nuc:~/workspaces/kube_folder$ kubectl apply -f /home/restful3/workspaces/kube_folder/apa000ser.yml
service/apa000ser created
(base) restful3@nuc:~/workspaces/kube_folder$ kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
apa000ser    NodePort    10.104.99.100   <none>        8099:30080/TCP   6s
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP          42m
```

[실습] 매니페스트 파일로 파드의 개수 늘리기

- 디플로이먼트 매니페스트 파일 수정
  - 매니페스트 파일(apa000dep.yml)에서 레플리카 수를 3에서 5로 수정합니다.
- 매니페스트 파일 읽어 들이기
  - 수정된 매니페스트 파일을 쿠버네티스에 반영합니다.
- 파드 수 증가 확인
  - 파드의 목록을 확인하여 새로 생성된 파드가 있는지 확인합니다.

```
(base) restful3@nuc:~/workspaces/kube_folder$ vi apa000dep.yml
(base) restful3@nuc:~/workspaces/kube_folder$ kubectl apply -f ./apa000dep.yml
deployment.apps/apa000dep configured
(base) restful3@nuc:~/workspaces/kube_folder$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
apa000dep-58cc68ccdf-jxjpw    1/1     Running   0          39m
apa000dep-58cc68ccdf-q6wm7    1/1     Running   0          14s
apa000dep-58cc68ccdf-r8gg6    1/1     Running   0          14s
apa000dep-58cc68ccdf-tmd5l    1/1     Running   0          39m
apa000dep-58cc68ccdf-xbcsc    1/1     Running   0          39m
```

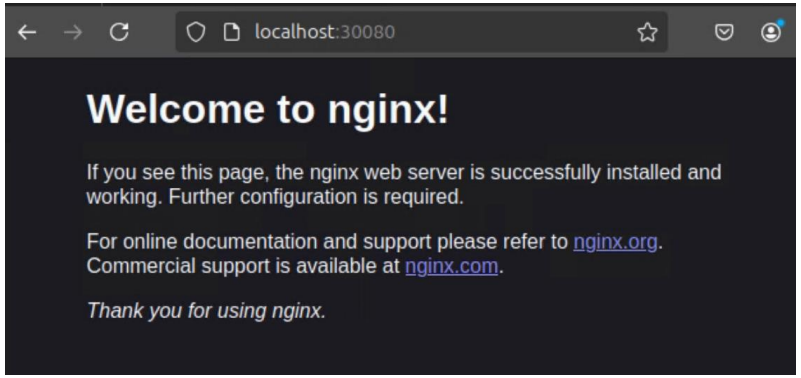
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apa000dep
spec:
  selector:
    matchLabels:
      app: apa000kube
  replicas: 5
  template:
    metadata:
      labels:
        app: apa000kube
    spec:
      containers:
        - name: apa000ex91
          image: httpd
          ports:
            - containerPort: 80
```

[실습] 매니페스트 파일로 아파치를 nginx로 바꾸기

- 디플로이먼트 매니페스트 파일 수정
  - 매니페스트 파일에서 이미지 항목을 **httpd**에서 **nginx**로 수정
- 매니페스트 파일 다시 읽어들이기
  - 수정된 매니페스트 파일을 쿠버네티스에 읽어 들여 리소스에 반영

```
(base) restful3@nuc:~/workspaces/kube_folder$ vi apa000dep.yml
(base) restful3@nuc:~/workspaces/kube_folder$ kubectl apply -f ./apa000dep.yml
deployment.apps/apa000dep configured
```

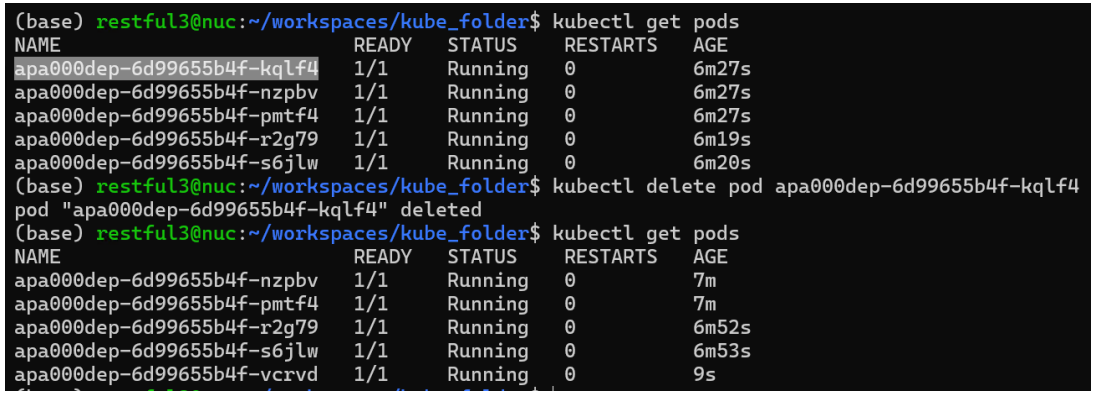
- 동작 확인
  - 웹 브라우저에서 http://localhost:30080에 접속하여 nginx 초기 화면이 나타나는지 확인




```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apa000dep
spec:
  selector:
    matchLabels:
      app: apa000kube
  replicas: 5
  template:
    metadata:
      labels:
        app: apa000kube
    spec:
      containers:
        - name: apa000ex91
          image: nginx
          ports:
            - containerPort: 80
```

수동으로 파드를 삭제한 후 자동복구 되는지 확인

- 쿠버네티스는 '바람직한 상태'를 유지하려고 합니다. 이번 실습에서는 수동으로 파드를 삭제한 후, 쿠버네티스가 자동으로 파드를 다시 생성하는지 확인합니다.
- get 커맨드로 파드의 목록 확인
  - 파드의 목록을 출력하고 삭제할 파드의 ID를 선택합니다.
- 수동으로 파드를 삭제
  - 특정 파드를 삭제하기 위해 delete 명령어를 사용합니다.
- 삭제된 파드가 다시 보충되는지 확인
  - 삭제된 파드의 ID와 동일한 파드가 사라지고, 새로운 파드가 자동으로 생성되었는지 확인합니다.



생성했던 디플로이먼트와 서비스 삭제

- 디플로이먼트 삭제
  - delete 명령어를 통해 디플로이먼트 리소스를 삭제
- 디플로이먼트 삭제 확인
  - 삭제 후 디플로이먼트 목록에서 해당 리소스가 제거되었는지 확인
- 서비스 삭제
  - 디플로이먼트 삭제 후 관련 서비스를 삭제
- 서비스 삭제 확인
  - 서비스 목록을 확인하여 서비스가 성공적으로 삭제되었는지 확인

```
(base) restful3@nuc:~/workspaces/kube_folder$ kubectl delete -f ./apa000dep.yml
deployment.apps "apa000dep" deleted
(base) restful3@nuc:~/workspaces/kube_folder$ kubectl get deployment
No resources found in default namespace.
(base) restful3@nuc:~/workspaces/kube_folder$ kubectl delete -f ./apa000ser.yml
service "apa000ser" deleted
(base) restful3@nuc:~/workspaces/kube_folder$ kubectl get service
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   89m
```