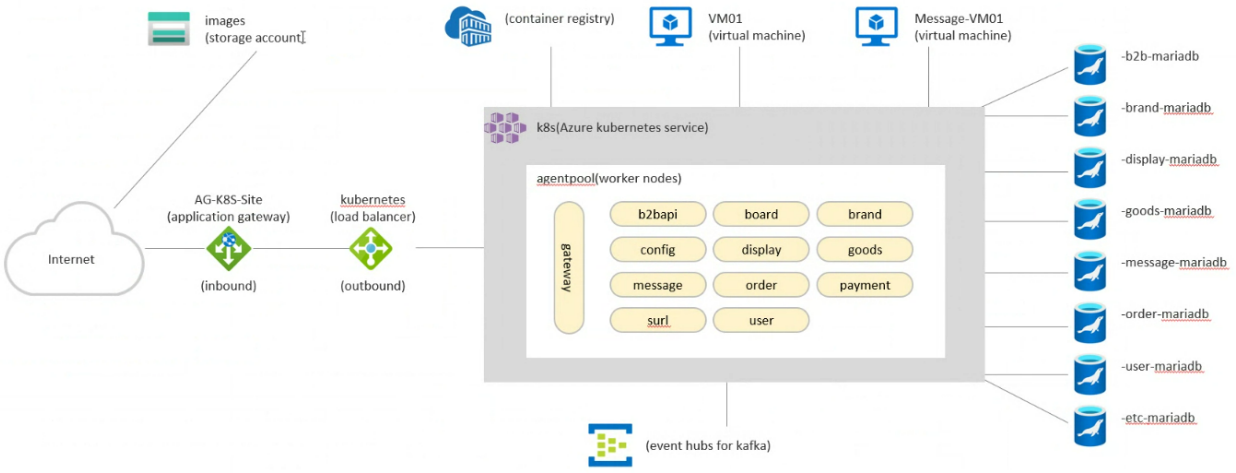
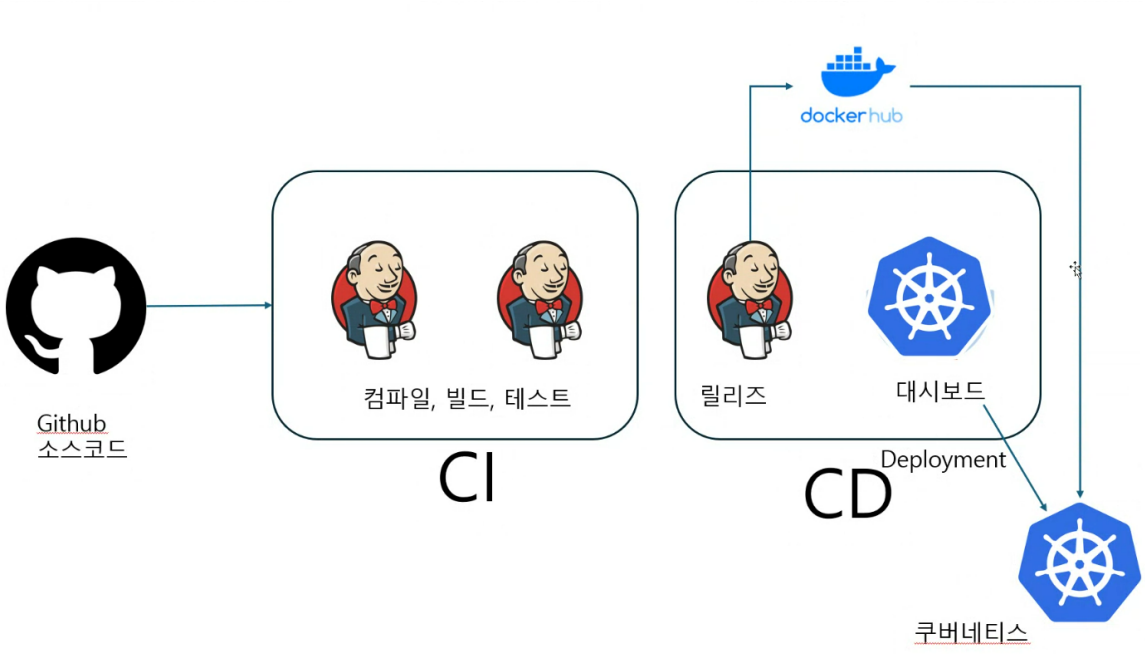
쿠버네티스 MSA 실습

Azure backend system architecture



쿠버네티스

- 쿠버네티스(Kubernetes, 약칭 k8s)는 컨테이너화된 애플리케이션을 자동으로 배포, 스케일링 및 관리하기 위한 오픈소스 플랫폼 (+ Service Discovery, ConfigMap, etc)
- 구글(Google)이 개발했으며, 현재는 Cloud Native Computing Foundation(CNCF)에서 관리
- 컨테이너(예: Docker)를 효율적으로 운영하기 위해 클러스터 환경에서 여러 서버를 하나의 시스템처럼 관리하며, 애플리케이션의 고가용성(HA), 확장성, 장애 복구 등을 지원



쿠버네티스 장점

- 자동화된 운영 관리
  - 애플리케이션 배포, 롤링 업데이트, 롤백 등을 자동화하여 운영 부담 감소.
  - Self-healing(자가 치유): 컨테이너 장애 시 자동 재시작 또는 교체.
- 효율적인 리소스 활용
  - 여러 서버(노드)를 클러스터로 묶어 CPU, 메모리 등을 최적화하여 사용.
  - 필요시 자동 스케일링(수평 확장) 지원
- 서비스 디스커버리 & 로드 밸런싱
  - 내장된 로드 밸런서로 트래픽을 자동 분산.
  - DNS 또는 IP를 통해 서비스 간 통신 관리.
- 선언적 구성 관리(Declarative Configuration)
  - YAML 파일로 인프라 상태를 정의하면, 쿠버네티스가 해당 상태를 유지하도록 조정.
- 커뮤니티 & 생태계
  - 대규모 오픈소스 생태계와 활발한 커뮤니티 지원.
  - 엔터프라이즈 솔루션(Red Hat OpenShift, Rancher 등)과의 호환성.

쿠버네티스 오브젝트 (대시보드의 워크로드, 서비스, 클러스터 등)

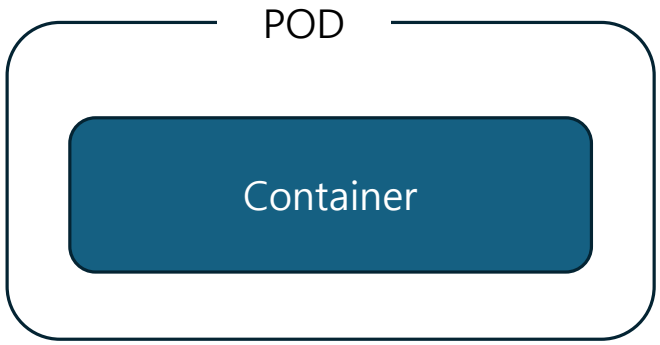
- 쿠버네티스 시스템에서 영속성을 가지는 객체로 쿠버네티스 클러스터의 상태 관리를 위한 객체
- 쿠버네티스 오브젝트의 가장 큰 특징은 생성 의도를 가지고 있다는 점임.
- 즉, Desired State가 기술된 오브젝트를 생성하면 쿠버네티스 시스템이 해당 오브젝트의 Desired State를 보장하는 방식으로 지속적으로 동작함. -> (선언적 구성 관리)
- 따라서 인프라 관리는 주로 yaml 파일을 이용한 선언형 프로그래밍 방법을 이용

- 쿠버네티스 오브젝트 주요 구성 요소
  - metadata
    - 오브젝트의 식별 정보 (이름, UID, 라벨 등)
    - 예) name: my-app, labels: { app: web }
  - spec (사양)
    - 사용자가 원하는 상태 (Desired State)
    - 예) replicas: 3, image: api-gateway:1.0.1
  - status (상태)
    - 쿠버네티스가 보고하는 실제 상태 (Observed State)
    - 예) availableReplicas: 3, phase: Running

name or label로 요청 가능

카나리 배포 : 1%의 사용자에게만 새로운 버전의 앱 제공
(label을 통해서 가능)
(시험적 배포 후 이상이 없으면 모든 사용자에게 적용하는 방식)

쿠버네티스 기본 오브젝트 - Pod

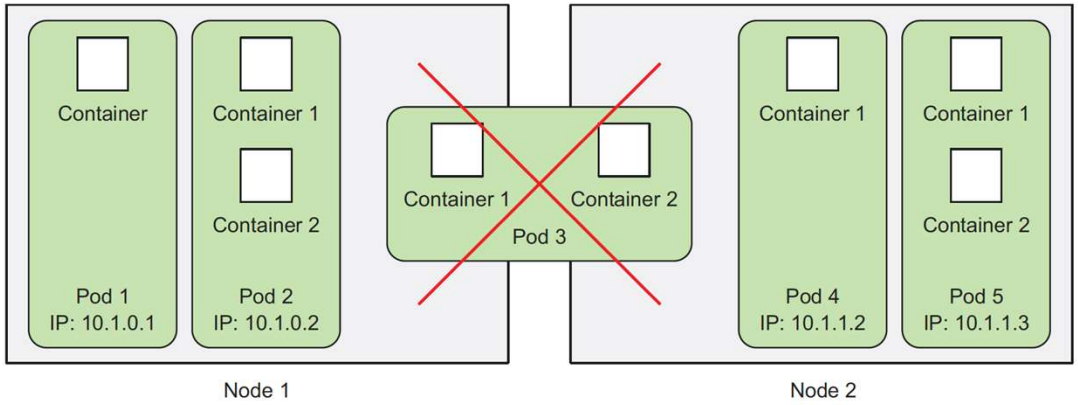


POD : 컴퓨터
Container : Spring Application

- Pod는 쿠버네티스에서 컨테이너의 기본 단위로, 가장 기본적인 배포 단위
- Pod는 1개 이상의 컨테이너로 구성된 컨테이너의 집합이며, 다음과 같은 특징을 지님 (대부분 한 개의 컨테이너만 사용)
  - 각 Pod에는 고유한 IP 주소가 할당
  - 같은 Pod내의 컨테이너는 IP와 Port 공간을 공유
  - 따라서 파드에 속한 컨테이너는 localhost를 사용하여 서로 통신할 수 있음
  - Pod내의 컨테이너간에는 디스크 볼륨을 공유할 수 있음 (볼륨을 통해 공유)

즉, Pod는 논리 호스트로서, 물리적 호스트 또는 가상 머신과 매우 흡사하게 동작

쿠버네티스 기본 오브젝트 – Pod (계속)



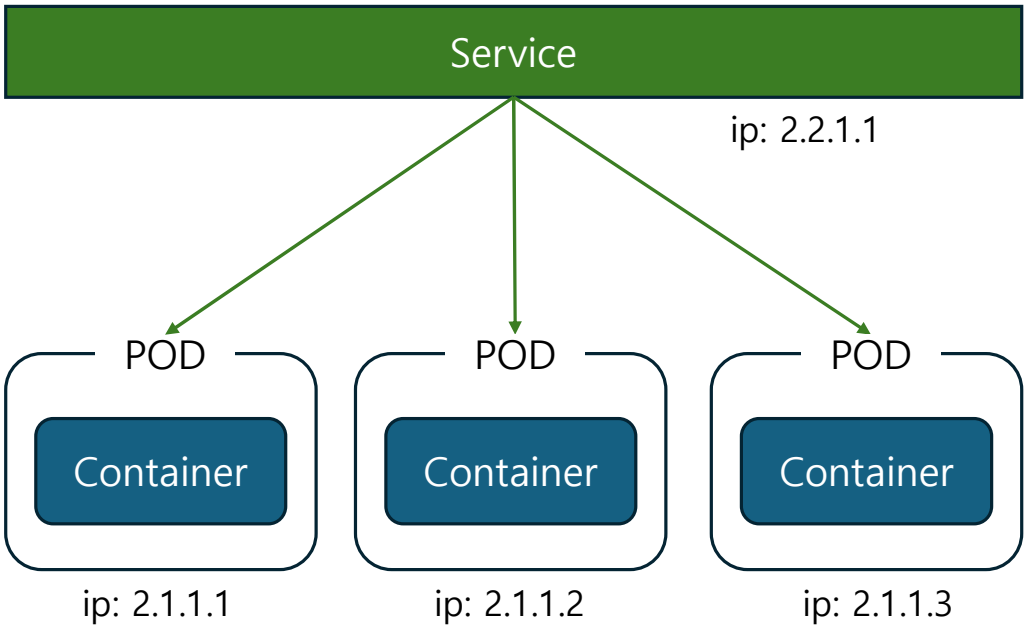
파드가 필요한 이유:

멀티 프로세스로 동작하는 하나의 서비스가 있다고 가정 (ex. Pod2)

(예, 하나의 프로세스는 인터넷에서 자료 수집을 담당하여 파일로 저장, (ex. Pod2 - Container1)
하나의 프로세스는 저장되는 파일을 읽어와서 자료 분석하는 역할을 담당) (ex. Pod2 - Container2)

- 이 두 프로세스는 동일한 Machine에서 동작해야 함.
- 컨테이너는 오직 하나의 프로세스만 실행하도록 설계되어 있음.
  (만일 두 개 이상의 프로세스를 하나의 컨테이너에서 동작시킨다면 일부 프로세스 장애 처리에 대한 별도 로직을 설계해야함.) -> 복잡성 ↑

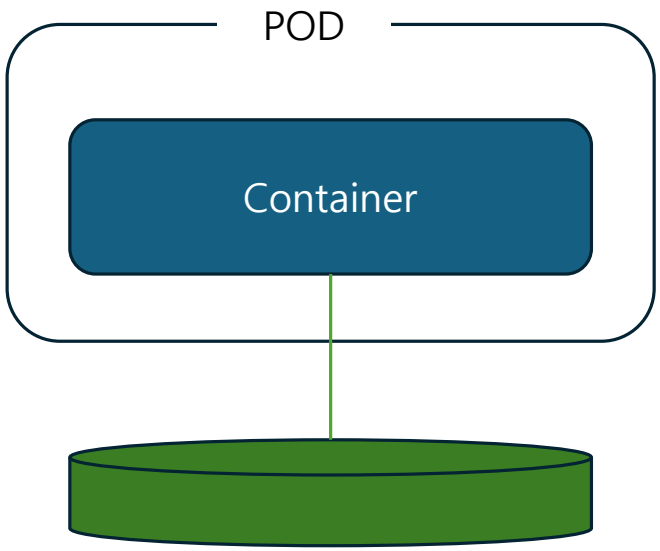
쿠버네티스 기본 오브젝트 – Service



- 서비스는 파드 집합에서 실행중인 애플리케이션을 네트워크 서비스로 노출하는 추상화 제공.
- 즉, 파드에게 부여되는 IP 주소들에 대해 단일 DNS 명을 부여하고, 파드들에 대해 로드밸런싱을 수행

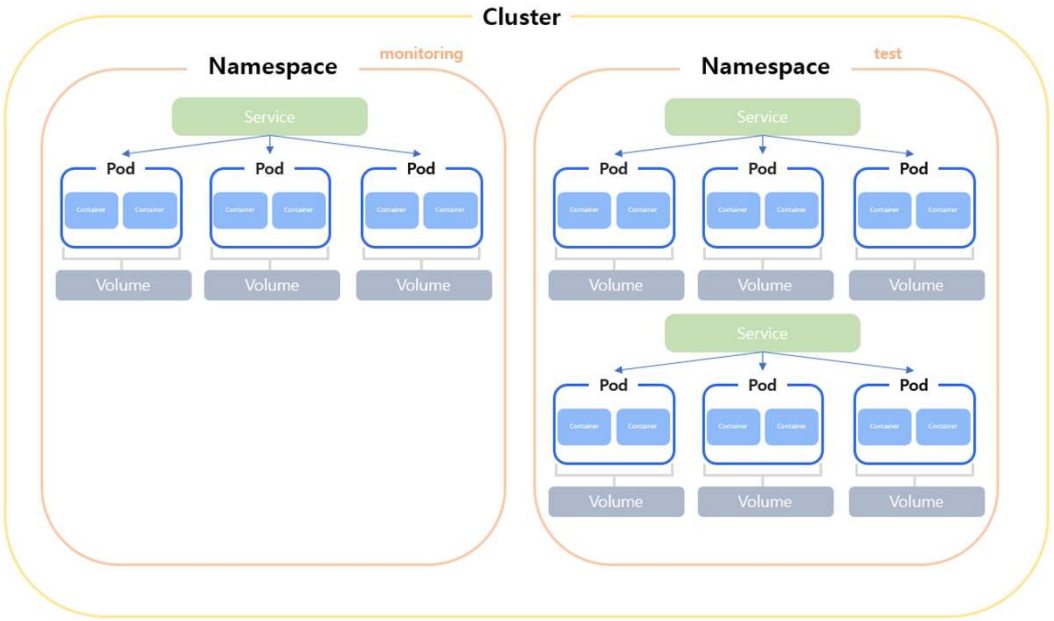
= (Discovery Server + Load Balancer)

쿠버네티스 기본 오브젝트 – Volume



- Pod가 생성될 때 컨테이너가 실행되면서 로컬 디스크를 생성하게 되는데, 이 로컬 디스크의 경우 컨테이너가 재실행되거나 종료되면 로컬 디스크 안에 있는 파일도 같이 손실된다는 단점이 있음.
- 따라서 파일을 영구적으로 저장해야할 필요가 스토리지가 필요하였고, 쿠버네티스는 이를 볼륨이라고 지칭함.
- 볼륨은 Pod가 실행될 때 컨테이너에 마운트되어 사용됨.
- 볼륨은 NFS, Cinder, CephFS 등과 같은 네트워크 스토리지부터 AWS EBS(Elastic Block Store), Google PD(Persistent Disk) , Azure Disk Storage 등과 같은 클라우드 스토리지까지 다양한 유형의 볼륨들을 지원

쿠버네티스 기본 오브젝트 – Namespace



- Namespace는 쿠버네티스 클러스터 내의 리소스들을 논리적으로 구분할 수 있게 해주는 하나의 단위

영속성을 가져야 하는 DB, Kafka 등은 Kubernetes 밖에서 실행
(Kubernetes에서는 자주 서비스가 죽었다가 살아나기 때문)