

Spring Boot Seminar - 4

Wafflestudio Rookies seminar 2024

Sangmin Kim, 2024

Seminar - 4 목차

- Assignment 3 리뷰
- Docker
- Kubernetes
- App Server 가 아니지만 필수적인 요소들

Disclaimer

- 컨테이너 기반 배포의 역사와 디테일
- K8s 의 정확한 동작과 디테일
- ..와 같이 아주 정확한 정보 전달 보다는, 서버 개발자에게 배포를 위해 필요한 기초 지식을 이해 하기 쉬운 수준으로만 전달합니다
- 더 알아보고 싶은 내용은 ChatGPT/구글링 해 보시는 걸 추천드립니다
 - <https://www.docker.com/101-tutorial/>
 - <https://kubernetes.io/docs/concepts/overview/>

Docker

Docker 가 등장한 배경

- 전통적인 서버 배포 방식의 문제점
- 복잡한 배포 프로세스
 - 배포 스크립트를 서버에 접속해서 돌리기
 - 배포 스크립트를 잘 관리해줘야 한다 (서버의 의존성을 깎다던지)
 - -> 새로운 서버를 증설할 때 배포 스크립트를 돌려야 한다

Docker

Docker 가 등장한 배경

- 전통적인 서버 배포 방식의 문제점
- 복잡한 배포 프로세스
 - 배포 스크립트를 서버에 접속해서 돌리기
 - 배포 스크립트를 잘 관리해줘야 한다 (서버의 의존성 설치 등)
 - -> 새로운 서버를 증설할 때 배포 스크립트를 돌려야 한다



Docker

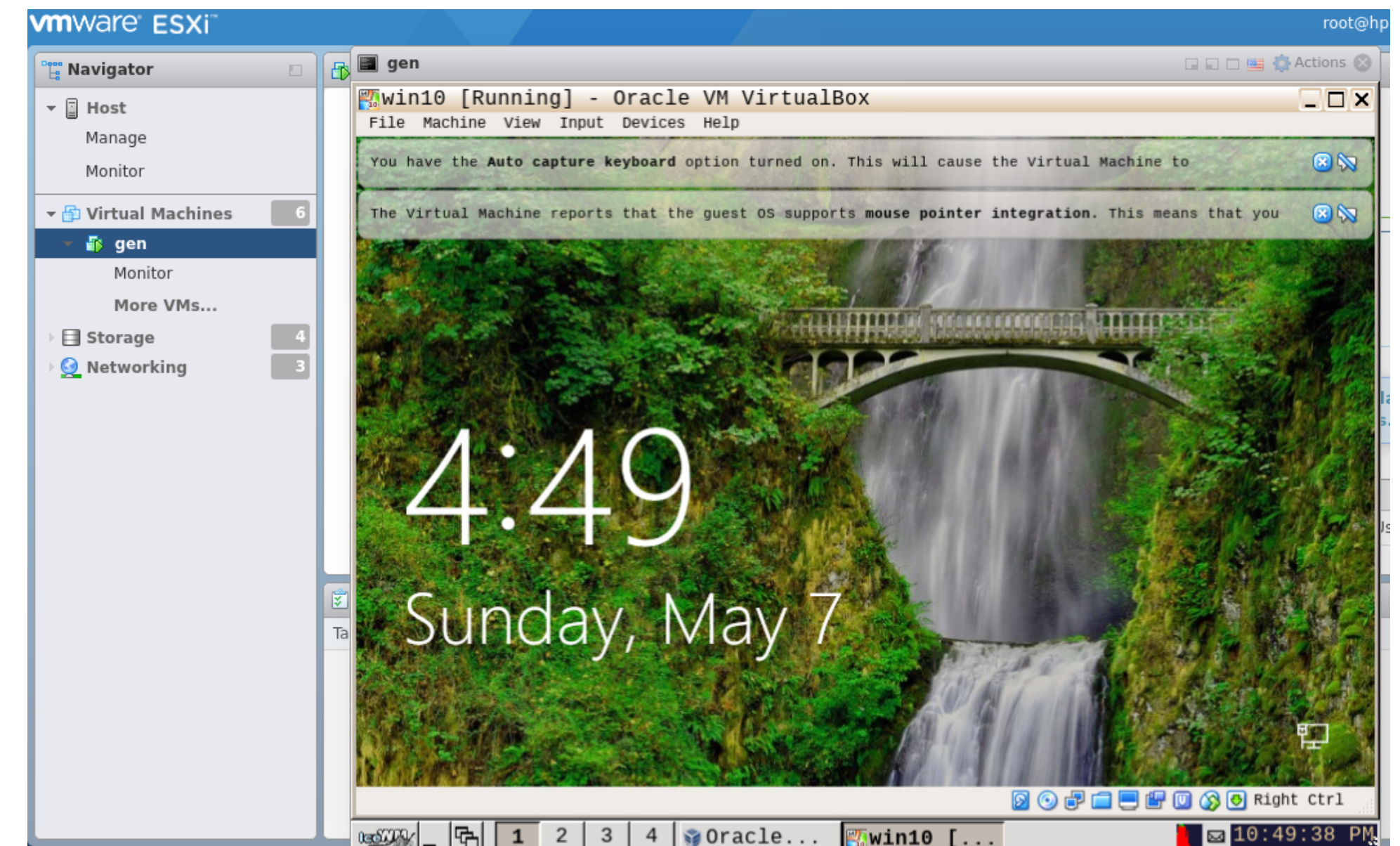
Docker 가 등장한 배경

- 전통적인 서버 배포 방식의 문제점
- 서버 자원의 격리
 - 하나의 컴퓨터 내에 여러 개의 서버를 띄운다면..?
 - 한 서버가 OOM 등으로 터지면서 다른 서버도 같이 터져버린다
 - (에러가 전파되는 문제)

Docker

Docker 가 등장한 배경

- 가상머신 (Virtual Machine)
 - OS 를 포함한 배포 환경을 완전히 Ctrl C + Ctrl V 한다면 문제가 해결되지 않을까?
 - 가상머신을 실제(물리적) 머신 위에서 따로 돌리는 식으로



Docker

Docker 가 등장한 배경

- 가상머신 (Virtual Machine)
 - 일관된 환경으로 서버를 실행할 수 있고
 - 독립적인 리소스를 할당받아 에러가 전파되지 않는다
 - 그런데 무겁다
- 컨테이너
 - VM 은 너무 무겁다(완전한 OS, 느린 부팅 등).. 가볍게 가보자

Docker

Docker 기반의 배포

- 컨테이너 기반의 배포
 - 한번 빌드하면 어떤 기기에서도 똑같은 환경에서 실행되는 것 처럼 돌아가길 원한다
 - 즉, 배포하고자 하는 서버를 Build 해서 저장소에 올려두면
 - 서버 컴퓨터에서는 저장소에서 그 서버 *Image 를 Pull 받아서 돌리면 된다

Docker

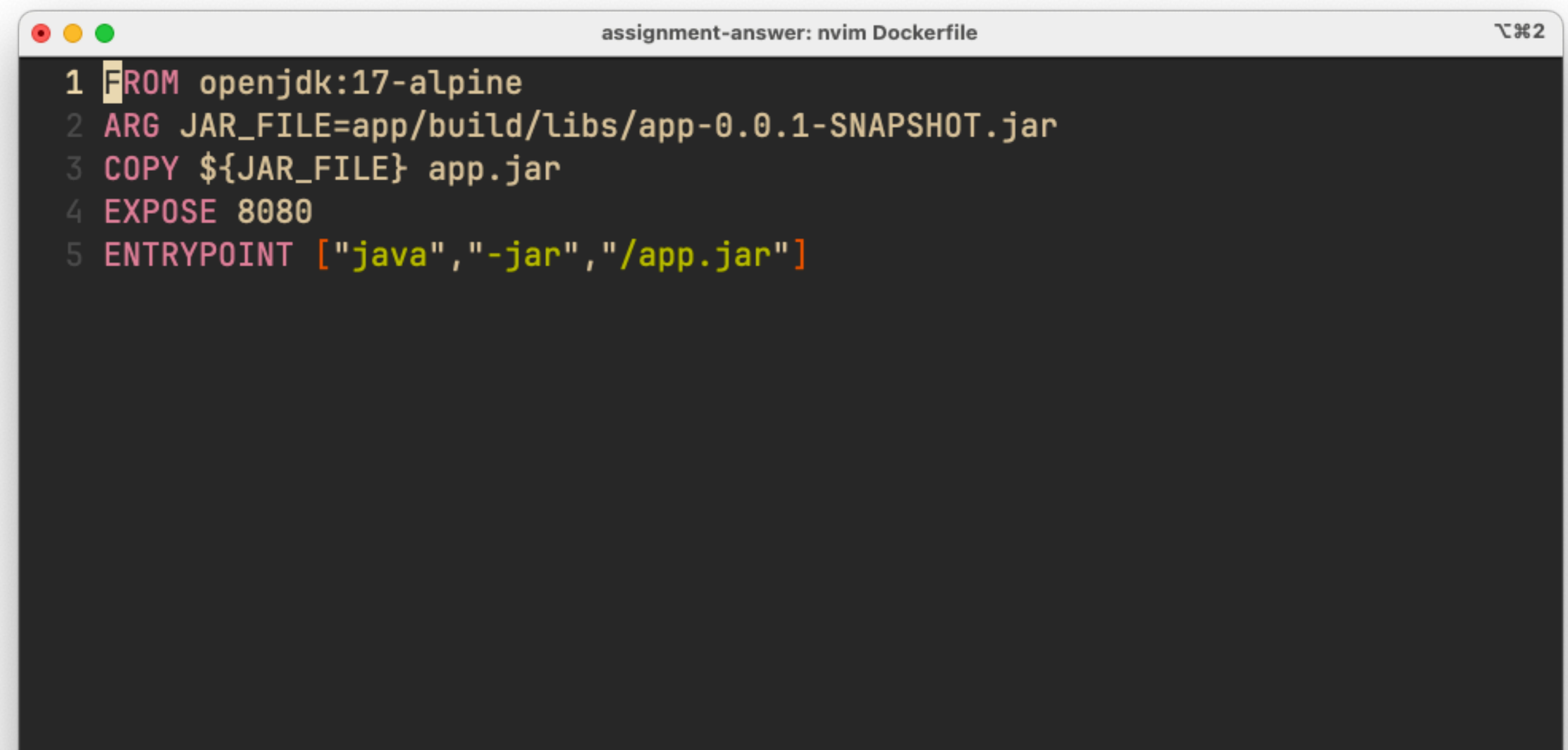
Docker 기반의 배포

- Container 와 Image
 - Container - Instance (빵)
 - Image - Class (빵틀)

Docker

Docker 기반의 배포

- Image 를 만드는 방법
- Base Image 위에 내가 새로운 Command 를 누적해서 만든다
 - Java 17 이 돌아가는 image 위에
 - 내가 build 한 jar 를 넣고
 - 실행하는 형태로 만든다

A screenshot of a code editor window titled "assignment-answer: nvim Dockerfile". The editor shows a Dockerfile with five lines of code, each numbered. The code is as follows:

```
1 FROM openjdk:17-alpine
2 ARG JAR_FILE=app/build/libs/app-0.0.1-SNAPSHOT.jar
3 COPY ${JAR_FILE} app.jar
4 EXPOSE 8080
5 ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Docker

Docker 기반의 배포

- Volume
 - Container 가 사용할 스토리지를 붙여준다 (Persist Storage 를 붙여주는 형태로 사용)
- Network
 - 여러 Container 간의 소통이 필요하면, 기본적으로 격리된 컨테이너 간의 소통이 가능하도록 해 줘야 한다

Docker

Container Orchestration

- 관리할 Container 들과, Container 간의 Network, 외부에서 유입될 유입 지점 (Ingress)
- 등등... Container 들 간의 관계를 정의하고, 재사용한다거나, 관리하고 싶다
 - -> Container Orchestration

Docker-Compose

Container Orchestration

- Docker 컨테이너의 스펙, 네트워크 스펙, 볼륨 스펙 등을 미리 정의해 두고 up - down 만 시키는 간단한 Orchestration 툴

```
assignment-answer: nvim docker-compose.yaml
1 version: '3.8'
2
3 services:
4   mysql:
5     image: mysql:8.4
6     container_name: mysql-db
7     ports:
8       - "3306:3306"
9     volumes:
10      - ./mysql/conf.d:/etc/mysql/conf.d
11     environment:
12       MYSQL_ROOT_PASSWORD: root_password # Change as needed
13       MYSQL_DATABASE: meal
14       MYSQL_USER: user
15       MYSQL_PASSWORD: somepassword
16     networks:
17       - test_network
18   app:
19     image: seminar/spring:latest
20     restart: on-failure
21     container_name: spring-app
22     ports:
23       - "8080:8080"
24     depends_on:
25       - mysql
26     environment:
27       SPRING_DATASOURCE_URL: "jdbc:mysql://localhost:3306/meal"
28       SPRING_DATASOURCE_USERNAME: "user"
29       SPRING_DATASOURCE_PASSWORD: "somepassword"
30     networks:
31       - test_network
32
33 networks:
34   test_network:
```

NORMAL docker-compose.yaml Top

Docker-Compose

Container Orchestration

- 지금까지 세미나에서 사용한 MySQL DB 도 이 Docker Container 로 띄운 것
- 물론 로컬에 설치해서 띄워도 되지만, 사람마다 다른 버전 / 다른 OS 를 깔면 문의가 아주 많았을 수 있음
 - Docker 로 모두에게 동일한 환경으로 DB 를 띄운 것

Kubernetes

Container Orchestration Tool

- Docker Compose 로 개발 시점에는 충분할 수도 있지만, 프로덕션은 훨씬 대규모 & 복잡한 방식으로 연결된 Container 들과 네트워크가 필요하다
- Kubernetes 가 필요한 이유 (사례 기반)
 - 무중단 배포를 하고 싶다
 - 트래픽 증가에 따라서 자동으로 인스턴스를 늘리고 싶다
 - 특정 시간대에 실행되어야 하는 Cronjob 들을 좋은 방법으로 관리하고 싶다
 - 리뉴얼된 서버의 안정성을 위해 1% 의 트래픽만 쫓아보고 싶다.

Kubernetes

Container Orchestration Tool

- Compose 와 비교되는 대략적인 동작 원리
 - Controller 를 통해 “내가 선언한 스펙 대로 인프라를 띄워 준다”
 - 내가 Pod 1개 필요해 라고 스펙을 변경하면 그에 맞춰 띄워준다
 - 내가 Pod 의 내부 컨테이너를 새로 배포한 Image 로 바꾸고 싶어라고 하면 바꿔준다
 - How? 무한루프 돌면서 상태를 검증하는 느낌

Kubernetes

Container Orchestration Tool

- Pod - Container 들의 집합 (보통 서버군과 같은 느낌)
- Deployment - N 개의 Pod 를 띄워 놓겠다 와 같은 스펙
 - Rolling Version Update 등의 무중단 배포를 가능하게 해준다
- Ingress
 - 외부에서 Pod 에 접근하는 네트워크를 정의한다
- 등등....

서버 인프라를 구성하는 다른 요소들

Storage Computing Networking

Relational
Database

File (Object)
Storage

Instance

Instance

Serverless
Function

Load
Balancer

CDN

서버 인프라를 구성하는 다른 요소들

- RDB
 - MySQL / PostgreSQL 같은 메인 데이터베이스
 - e.g. AWS RDS
- File Storage
 - 보통 Binary File 저장소 (이미지와 같은 데이터)
 - 내용물에 대한 인덱싱이 필요없는 데이터 위주
 - e.g. AWS S3

Storage

Relational
Database

File (Object)
Storage

서버 인프라를 구성하는 다른 요소들

- Computing Instance
 - 그냥 컴퓨터 1개와 같음
 - AWS EC2(노드 기반), AWS ECS(컨테이너 기반)
- Serverless Function
 - 한번만 실행하면 되는 코드들을 On-Demand 로 사용
 - 경제적 & 스케일링 용이
 - AWS Lambda

Computing

Instance

Instance

Serverless
Function

서버 인프라를 구성하는 다른 요소들

- Load Balancer
 - 외부에서 들어온 요청을 적절한 Computing Unit 으로 전달해줌 (IP 와 Domain 등으로 Routing)
 - e.g. AWS Elastic Load Balancer
- CDN
 - 정적인 파일(e.g. 이미지)의 경우 전달 레이턴시를 최적화 하기 위해 지역별 데이터 센터에 미리 캐싱해두는 서비스
 - e.g. AWS CloudFront

Networking



Load
Balancer

CDN

서버 인프라를 구성하는 다른 요소들

- Managed K8S Cluster
 - K8s 는 Computing Unit 과 Networking Unit 을 관리해야 한다
 - 그런데 이런 컴포넌트들을 Cloud Service (AWS) 에서 제공해주는 만큼, 이를 쉽게 할 수 있도록 k8s 자체를 클라우드에서 완전 관리형으로 지원해준다
 - e.g. Network, Container 등을 AWS Elastic LoadBalancer 와 AWS EC2? ECS? 등으로 바로 연결해서 만들어주는 AWS EKS

Computing Networking

