정회원 스터디 Docker/k8s

김낙현 김민수 양민석 임경섭

목차

- 이전까지 했던 것들
  - Docker 이미지 빌드 (with Dockerfile)
  - Docker compose
- 이번 스터디 진행 상황
  - Kubernetes (k8s)
  - Load balancing
  - Kubernetes 작동 방식
  - 실습

Docker 이미지 빌드 (with Dockerfile)

```
FROM python:3.6-slim
MAINTAINER Mark Gituma <mark.gituma@gmail.com>

ENV PROJECT_ROOT /app
WORKDIR $PROJECT_ROOT

COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
COPY . .
CMD python manage.py runserver 0.0.0.0:8000
```

- Dockerfile은 Docker 이미지 설정 파일
- Dockerfile에 설정된 내용대로 이미지를 생성

Docker compose

- Docker Compose는 container 실행에 사용되는 옵션과 container 간의 dependency들을 모두 `docker-compose.yml` 파일에 적어두고, 명령어를 통해 container를 실행/관리

```
version: '2.1'

services:
  db:
    image: postgres:9.6.1
    volumes:
      - ./docker/data:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=sampledb
      - POSTGRES_USER=sampleuser
      - POSTGRES_PASSWORD=samplesecret
      - POSTGRES_INITDB_ARGS=--encoding=UTF-8
    healthcheck:
      test: "pg_isready -h localhost -p 5432 -q -U postgr
      interval: 3s
      timeout: 1s
      retries: 10
```

Docker Compose(Cont')

```
  django:
    build:
      context: .
      dockerfile: ./compose/django/Dockerfile-dev
    environment:
      - DJANGO_DEBUG=True
      - DJANGO_DB_HOST=db
      - DJANGO_DB_PORT=5432
      - DJANGO_DB_NAME=sampledb
      - DJANGO_DB_USERNAME=sampleuser
      - DJANGO_DB_PASSWORD=samplesecret
      - DJANGO_SECRET_KEY=dev_secret_key
    ports:
      - "8000:8000"
    depends_on:
      db:
        condition: service_healthy
    links:
      - db
    command: /start-dev.sh
    volumes:
      - ./:/app/
```

Kubernetes (k8s)

- Kubernetes는 container를 스케줄링하고 실행 하는 container orchestration tool
- Kubernetes의 장점
  - 스케줄링을 알아서 해줌
  - Resource limitation
  - Load balancing

Kubernetes 작동 방식

- Pod
  - 가장 기본적인 단위로 pod을 사용
  - pod은 container의 집합
  - 각 pod은 IP주소, 포트 등을 가짐
- Node
  - node는 pod들을 배치할 수 있는 환경을 제공하는 가상머신
  - 또한 pod들의 관리/통신을 제공하는 리소스들을 포함
  - 내부적으로 pod들을 생성, 파괴, 교체, 재배치 등등.. 할 수 있음
  - node들 스스로도 생성되고 파괴되고 재배치 될 수 있음
  - 위 작업들은 모두 controller라는 내부 프로세스에 의해 실행

실습 (on mac OS)

- Requirements
  - Minikube
  - Virtualbox
  - Docker
  - kubectl

실습 (on mac OS)

- Minikube
    - Minikube는 node를 실행하는 가상머신
    - 가상머신이므로, virtualbox를 설치해서 사용
- Minikube 설치

```
$ brew cask install minikube
```

실습 (on mac OS)

- Kubectl
  - Kubernetes command line tool
  - Kubernetes에서 node를 배치하고 관리하는데에 사용
- Kubectl 설치

```
$ brew install kubectl
```

실습 (on mac OS)

- 실습 with Django

```
$ git clone https://github.com/gitumarkk/kubernetes
            _django.git
```

실습 (on mac OS)

- Minikube 실행
  - 하나의 node를 실행하는 가상머신을 만드는 과정

```
$ minikube start
```

- 실행이 잘 되었으면..

```
$ minikube status
host: Running
kubelet: Running
apiserver: Running
kubectl: Correctly Configured: pointing to minikube-vm at
192.168.99.103
```

실습 (on mac OS)

- 아래 명령어를 통해 현재 context를 확인 가능

```
$ kubectl config current-context
minikube
```

- minikube 내의 Docker 활성화

```
$ eval $(minikube docker-env)
```

- host의 Docker로 되돌아가기

```
$ eval $(minikube docker-env -u)
```

실습 (on mac OS)

- Node 관리 방법
  - 주로 kubectl의 명령어를 통해 관리함. but,
  - minikube에서 dashboard를 제공
  - 이 dashboard를 사용해서 관리하면 됨
- dashboard 실행

```
$ minikube dashboard
```

실습 (on mac OS)

- dashboard

실습 (on mac OS)

- Container 생성
  - Dockerfile을 통한 build로 이미지 생성 후 container 생성
- Dockerfile

```
FROM python:3.6-slim
MAINTAINER Mark Gituma <mark.gituma@gmail.com>

ENV PROJECT_ROOT /app
WORKDIR $PROJECT_ROOT

COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
COPY . .
CMD python manage.py runserver 0.0.0.0:8000
```

실습 (on mac OS)

- Build

```
$ docker build -t <IMAGE_NAME>:<TAG> .
```

```
$ docker build -t django:latest .
```

실습 (on mac OS)

- Deployment 생성
  - 위에서 생성한 pod를 관리하기 위함

```
$ kubectl run <deployment-name> --image=<IMAGE-NAME> \
                                --port=8000
```

- 아래 명령어를 통해, 8000번 포트에 deploy-django라는 이름의 deployment를 생성

```
$ kubectl run deploy-django --image=django --port=8000
deployment.apps "deploy-django" created
```

실습 (on mac OS)

- 생성된 deployment 확인

```
$ kubectl get deployment
NAME            DESIRED   CURRENT   UP-TO-DATE
AVAILABLE       AGE
deploy-django   1         1         1
0               12s
```

- Deployment 삭제

```
$ kubectl delete deployment/deploy-django
deployment.extensions "deploy-django" deleted
```

실습 (on mac OS)

- 아쉽게도 성공한 사람이 없습니다..

끝! 감사합니다

References

- Kubernetes 개념
  - https://rancher.com/load-balancing-in-kubernetes/
- Kubernetes tutorial
  - https://medium.com/@markgituma/kubernetes-local-to-production-with-django-1-introduction-d73adc9ce4b4