웹 페이지 배포

쿠버네티스 클러스터는 이미 생성 되어 있는 관계로 바로 간단한 웹 페이지를 쿠버네티스 클러스터에 배포 및 테스트 해 보겠다.

제일 먼저 준비 해야 될 것은 물론 배포 해야 될 웹 페이지다. 일단 기존에 테스트 용으로 만든 장고 웹 사이트를 배포 해 보겠다.

먼저, 웹 애플리케이션을 K8s 클러스터에 배포하기 위해서는 애플리케이션의 컨테이너 이미지를 준비해야 한다. 일반적으로 Docker와 같은 컨테이너 플랫폼을 사용하여 애플리케이션을 컨테이너화

웹 사이트의 소스 코드를 받은 후 Dockerfile을 추가 하였다. Dockerfile은 도커 이미지를 생성하기 위한 설정 파일이다. 이 파일에는 도커 이미지를 구성하는데 필요한 모든 단계와 명

```
# 파이썬 이미지 받기
FROM python:3.11

# 환경 변수 지정
ENV PYTHONUNBUFFERED 1
ENV PYTHONDONTWRITEBYTECODE 1

# 워킹 디렉터리 지정
WORKDIR /app

# 의존성 설치
COPY requirements.txt .
RUN pip install -r requirements.txt

# 컨테이너 안에 소스 코드 복사
COPY . .

# 장고 사이트가 실행 되기 위해 포트 열기
EXPOSE 8000

# 웹 사이트가 실행 하기 위해 초기 명령어
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

마지막으로 이 Dockerfile을 빌드 및 Docker 레지스트리에 Push 해준다

```
# Dockerfile을 사용하여 커스텀 이미지 생성
docker build -t django:v2 ./
```

```
# 생성된 이미지 태그 달기
docker tag django:v1 waji97/django:v2

# 태그 된 이미지 레지스트리에 업로드
docker push waji97/django:v2
```

이제 배포를 위해 매니페스트 YAML 파일을 작성 해야 된다. 파일 안에 먼저 Deployment 리소스를 사용하여 애플리케이션의 Replica을 지정 하고 어떤 이미지를 사용 하여 파드를

```
# 매니페스트 파일 작성 및 적용을 위해 같은 경로 이동
mkdir myapp
cd myapp

# 매니페스트 파일 생성
vi my-app.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-app
          image: waji97/django:v2 □□□# 방금 업로드한 웹 사이트 이미지
          ports:
            - containerPort: 8000 □□□
---
apiVersion: v1
kind: Service
```

```
metadata:
  name: my-app-service
spec:
  type: NodePort
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000□□□□□□
```

이렇게 YAML파일 지정 후 이 파일을 적용 시킨다

```
kubectl apply -f my-app.yaml
deployment.apps/my-app created
service/my-app-service created
```

적용 후 생성 되는 Pod, Deployment 그리고 Service를 확인 할 수 있다

```
# Pod 확인
kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
my-app-74dd96c584-9plfb   1/1     Running   0          30s
my-app-74dd96c584-lf97s   1/1     Running   0          30s
my-app-74dd96c584-tvl2n   1/1     Running   0          30s

# Service 확인
kubectl get svc
NAME           TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)       AGE
kubernetes     ClusterIP   10.96.0.1       <none>       443/TCP       18d
my-app-service   NodePort    10.110.204.183   <none>       80:30001/TCP   50s

# Deployment 확인
kubectl get deployment
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
my-app   3/3     3            3           40s
```

이제 결과를 확인 하기 위해 노드 중 아무 IP 주소에 30001 포트로 이동

웹 사이트 접속이 문제 없이 된다는 걸 확인 할 수 있다.

□□ 만약에 웹 페이지를 수정 하거나, 새로운 웹 페이지의 이미지가 업로드 될 시, Deployment 리소스 덕분에 업데이트가 쉽게 가능하다. 웹 페이지를 배포한 매니페스트 파일 속에서 내가 [

간단한 테스트를 해 보았다

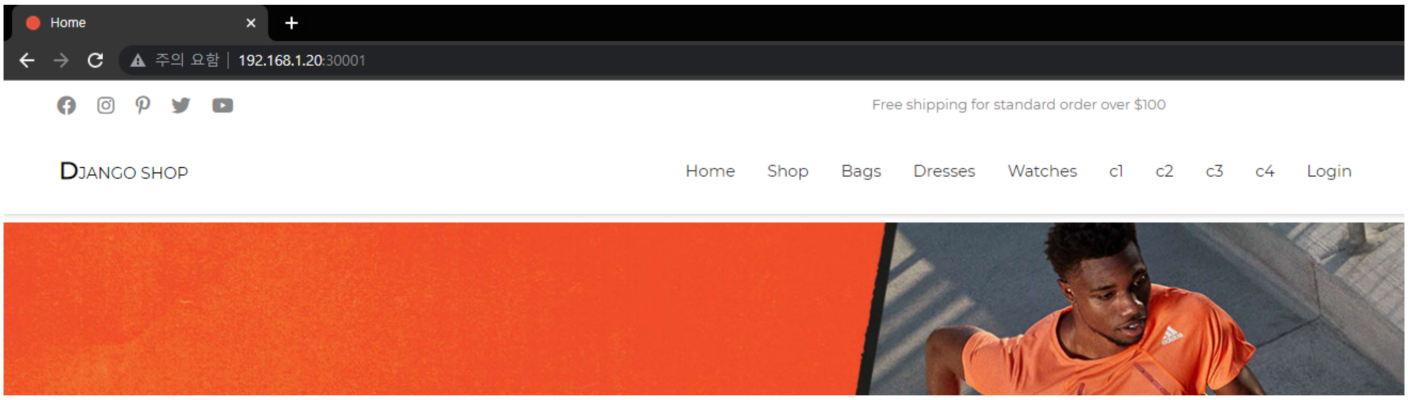
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-app
          #image: waji97/django:v2 □□□# 기존 업로드한 웹 사이트 이미지
          image: waji97/ecommerce:v2  □□# 새로 업로드한 웹 사이트 이미지
          ports:
            - containerPort: 8000 □□□
---
```

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
spec:
  type: NodePort
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000□□□□□□
```

다시 업데이트 된 YAML파일을 적용 시킨다

```
kubectl apply -f my-app.yaml
deployment.apps/my-app configured
service/my-app-service unchanged
```

적용 후 웹 사이트 방문 결과



Revision #2
Created 29 May 2023 13:29:58 by 와지
Updated 20 June 2023 12:51:32 by 와지