[Hands-on] 08. Kubernetes Workload(1) - Pod

이번에는 Pod를 관리하는 명령어들을 알아보겠습니다.

먼저 Pod를 생성하는 세 가지 방법을 알아보겠습니다.

첫 번째는 명령형 커맨드(Imperative commands) 입니다.
Pod를 직접 동작시키는 방법입니다.

```
ubuntu@ip-10-0-1-14:~$ kubectl run my-nginx1 --image=nginx
pod/my-nginx1 created
```

> **명령어** : `kubectl run my-nginx1 --image=nginx:1.19.3`

생성된 Pod를 볼까요?

```
ubuntu@ip-10-0-1-14:~$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
my-nginx1   1/1     Running   0          40s   172.17.0.5   minikube   <none>           <none>
```

> **명령어** : `kubectl get pods -o wide`

nginx:1.19.3 이미지를 이용해서 my-nginx1 pod를 생성했습니다.

두 번째는, 명령형 오브젝트 구성 (Imperative object configuration) 입니다.
미리 정의된 yaml파일을 이용해서 생성(creat) 합니다.

먼저 아래와 같은 파일을 작성합니다.

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    tier: frontend
  name: my-nginx2
spec:
  containers:
  - image: nginx:1.19.3
    name: my-nginx2
    ports:
    - containerPort: 80
```

> 파일명은 nginx2-pod.yaml로 합니다.

그리고, 아래와 같이 Pod를 생성합니다.

```
ubuntu@ip-10-0-1-14:~$ kubectl create -f nginx2-pod.yaml
pod/my-nginx2 created
```

> **명령어** : `kubectl create -f nginx2-pod.yaml`

생성된 Pod를 볼까요?

```
ubuntu@ip-10-0-1-14:~$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE     IP           NODE       NOMINATED NODE   READINESS GATES
my-nginx1   1/1     Running   0          5m39s   172.17.0.5   minikube   <none>           <none>
my-nginx2   1/1     Running   0          57s     172.17.0.6   minikube   <none>           <none>
```

> **명령어** : `kubectl get pods -o wide`

두 번째 Nginx Pod가 생성된 걸 볼 수 있습니다.
첫 번째와 다른 명령어를 사용하였지만, 결과는 동일한 걸 알 수 있습니다.

세 번째는, 선언형 오브젝트 구성 (Declarative object configuration) 입니다.
어떤 작업을 할지(**create**, **update**) 명시하지 않고 단순히 `apply`라는 키워드를 씁니다.
무엇을 할지는 쿠버네티스가 알아서 해줍니다. (ง˙▽˙)ง

먼저 아래와 같은 파일을 작성합니다.

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    tier: frontend
  name: my-nginx3
spec:
  containers:
  - image: nginx:1.19.3
    name: my-nginx3
    ports:
    - containerPort: 80
```

> 파일명은 nginx3-pod.yaml로 합니다.

nginx2-pod.yaml과 동일하고 name만 다르게 작성했습니다.

그리고, 아래 명령어로 Pod를 생성해보겠습니다.

```
ubuntu@ip-10-0-1-14:~$ kubectl apply -f nginx3-pod.yaml
pod/my-nginx3 created
```

> **명령어** : `kubectl apply -f nginx3-pod.yaml`

생성된 Pod를 볼까요?

```
ubuntu@ip-10-0-1-14:~$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE     IP           NODE       NOMINATED NODE   READINESS GATES
my-nginx1   1/1     Running   0          12m     172.17.0.5   minikube   <none>           <none>
my-nginx2   1/1     Running   0          7m48s   172.17.0.6   minikube   <none>           <none>
my-nginx3   1/1     Running   0          48s     172.17.0.7   minikube   <none>           <none>
```

> **명령어** : `kubectl get pods -o wide`

역시 동일한 결과를 얻을 수 있습니다.

선언형(Declarative) 이라는 말을 잘 기억해두세요.
그리고 `kubectl apply ~` 명령어도 많이 쓰이니, 잘 기억해 두시구요.

이제 **Container probe**를 알아보겠습니다.
여러 종류가 있지만, 그 중 한 가지만 실습을 통해서 알아보겠습니다.

먼저 아래와 같은 파일을 작성합니다. (httpGet을 이용하는 livenessProbe 입니다.)

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: k8s.gcr.io/liveness
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8080
        httpHeaders:
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
```

> 파일명은 livenessProbe_httpGet.yaml로 합니다.
> `livenessProbe` 부분이 설정 부분입니다.

그리고, 아래 명령어를 이용해서 Pod를 생성합니다.

```
ubuntu@ip-10-0-1-14:~$ kubectl apply -f livenessProbe_httpGet.yaml
pod/liveness-http created
```

> **명령어** : `kubectl apply -f livenessProbe_httpGet.yaml`

어느정도(10초이상) 시간이 지난 후 조회를 해보면 아래와 같이 보일거예요.

```
ubuntu@ip-10-0-1-14:~$ kubectl get pods -o wide
NAME            READY   STATUS    RESTARTS      AGE   IP            NODE       NOMINATED NODE   READINESS GATES
liveness-http   1/1     Running   1 (12s ago)   33s   172.17.0.8    minikube   <none>           <none>
my-nginx1       1/1     Running   0             68s   172.17.0.5    minikube   <none>           <none>
my-nginx2       1/1     Running   0             59s   172.17.0.6    minikube   <none>           <none>
my-nginx3       1/1     Running   0             51s   172.17.0.7    minikube   <none>           <none>
```

> **명령어** : `kubectl get pods -o wide`

테스트에 사용된 컨테이너는 libenessProbe 테스트를 위해서 생성 후 10초가 지난 뒤부터는 httpGet 요청에 대해서 500 Error를 발생하도록 되어있습니다.
(소스코드 참고)

livenessProbe는 계속해서 Pod의 상태를 살피고, 문제가 발생하면(500 error) 컨테이너를 재시작(RESTARTS) 합니다.
`kubectl get pods` 명령어의 결과에서 **RESTARTS** 의 숫자가 바로 이 재시작 횟수입니다.

이번 실습은 여기까지 입니다.

다음 실습을 위해서 생성한 모든 Pod를 삭제할게요.

```
ubuntu@ip-10-0-1-14:~$ kubectl delete pod --all
pod "my-nginx1" deleted
pod "my-nginx2" deleted
pod "my-nginx3" deleted
```

> **명령어** : `kubectl delete pod --all`