쿠버네티스 실습 3 - POD

1. Pod 이란?
2. Pod 생성
3. Pod 조회
4. Pod 로그
5. Pod 내부 접속
6. Pod 삭제

---

1. Pod 이란?

- Pod(파드)는 쿠버네티스에서 생성하고 관리할 수 있는 배포 가능한 **가장 작은 컴퓨팅 단위**입니다.
  - https://kubernetes.io/ko/docs/concepts/workloads/pods/
- 쿠버네티스는 Pod 단위로 스케줄링, 로드밸런싱, 스케일링 등의 관리 작업을 수행합니다.
  - 쿠버네티스에 어떤 애플리케이션을 배포하고 싶다면 최소 Pod 으로 구성해야 한다는 의미입니다.
- 조금 어렵다면 Pod 은 Container 를 감싼 개념이라고 생각할 수 있습니다.
  - 하나의 Pod 은 한 개의 Container 혹은 여러 개의 Container 로 이루어져있을 수 있습니다.
  - Pod 내부의 여러 Container 는 자원을 공유합니다.
- Pod 의 자세한 구조는 생략하겠습니다.
  - 다만 Pod 은 Stateless 한 특징을 지니고 있으며, 언제든지 삭제될 수 있는 자원이라는 점을 꼭 기억해주시기 바랍니다.

---

2. Pod 생성

- 간단한 Pod 의 예시입니다.

```
apiVersion: v1 # kubernetes resource 의 API Version
kind: Pod # kubernetes resource name
metadata: # 메타데이터 : name, namespace, labels, annotations 등을 포함
  name: counter
```

```
spec: # 메인 파트 : resource 의 desired state 를 명시
  containers:
  - name: count # container 의 이름
    image: busybox # container 의 image
    args: [/bin/sh, -c, 'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 1;
 done'] # 해당 image 의 entrypoint 의 args 로 입력하고 싶은 부분
```

- 위의 스펙대로 Pod 을 하나 생성해보겠습니다.

```
vi pod.yaml
# 위의 내용을 복사 후 붙여넣습니다.

kubectl apply -f pod.yaml
```

- `kubectl apply -f <yaml-file-path>` 를 수행하면, `<yaml-file-path>` 에 해당하는 kubernetes resource 를 생성 또는 변경 할 수 있습니다.
  - kubernetes resource 의 desired state 를 기록해놓기 위해 항상 YAML 파일을 저장하고, 버전 관리하는 것을 권장합니다.
  - `kubectl run` 명령어로 YAML 파일 생성 없이 pod 을 생성할 수도 있지만, 이는 kubernetes 에서 권장하는 방식이 아니므로 생략하겠습니다.
- 생성한 Pod 의 상태를 확인합니다.

```
kubectl get pod
# ContainerCreating

kubectl get pod
# 시간이 지난 후 Running 으로 변하는 것을 확인할 수 있습니다.
```

---

3. Pod 조회

- 방금 current namespace 의 Pod 목록을 조회하는 명령을 수행하였습니다.
  - 조회 결과는 Desired state 가 아닌, **Current State** 를 출력합니다.

```
kubectl get pod
```

- **namespace** 란 ?
  - namespace 는 kubernetes 에서 리소스를 격리하는 가상의(논리적인) 단위

    - `kubectl config view --minify | grep namespace:` 로 current namespace 가 어떤 namespace 로 설정되었는지 확인할 수 있습니다.
        - 따로 설정하지 않았다면 `default` namespace 가 기본으로 설정되어 있을 것입니다.

- 특정 namespace 혹은 모든 namespace 의 pod 을 조회할 수도 있습니다.

```
kubectl get pod -n kube-system
# kube-system namespace 의 pod 을 조회합니다.

kubectl get pod -A
# 모든 namespace 의 pod 을 조회합니다.
```

- pod 하나를 조회하는 명령어는 다음과 같습니다.
    - `<pod-name>` 에 해당하는 pod 을 조회합니다.

```
kubectl get pod <pod-name>
```

- pod 하나를 조금 더 자세히 조회하는 명령어는 다음과 같습니다.
    - `<pod-name>` 에 해당하는 pod 을 자세히 조회합니다.

```
kubectl describe pod <pod-name>
```

- 기타 유용한 명령을 소개드리겠습니다.

```
kubectl get pod -o wide
# pod 목록을 보다 자세히 출력합니다.

kubectl get pod <pod-name> -o yaml
# <pod-name> 을 yaml 형식으로 출력합니다.

kubectl get pod -w
# kubectl get pod 의 결과를 계속 보여주며, 변화가 있을 때만 업데이트됩니다.
```

---

4. Pod 로그

- pod 의 로그를 확인하는 명령어는 다음과 같습니다.

```
kubectl logs <pod-name>

kubectl logs <pod-name> -f
# <pod-name> 의 로그를 계속 보여줍니다.
```

- pod 안에 여러 개의 container 가 있는 경우에는 다음과 같습니다.

```
kubectl logs <pod-name> -c <container-name>

kubectl logs <pod-name> -c <container-name> -f
```

---

5. Pod 내부 접속

- pod 내부에 접속하는 명령어는 다음과 같습니다.

```
kubectl exec -it <pod-name> -- <명령어>
```

- pod 안에 여러 개의 container 가 있는 경우에는 다음과 같습니다.

```
kubectl exec -it <pod-name> -c <container-name> -- <명령어>
```

- docker exec 과 비슷한 명령임을 확인할 수 있습니다.

---

6. Pod 삭제

- pod 을 삭제하는 명령어는 다음과 같습니다.

```
kubectl delete pod <pod-name>
```

- 혹은 다음과 같이 리소스를 생성할 때, 사용한 YAML 파일을 사용해서 삭제할 수도 있습니다.

```
kubectl delete -f <YAML-파일-경로>
```

- 위 명령어는 꼭 pod 이 아니더라도 모든 kubernetes resource 에 적용할 수 있습니다.