Kubernetes Manifest 기초 문법

- Kubernetes Manifest는 우리가 원하는 상태가 무엇인지를 YAML 형식으로 정의한 설정 파일입니다.

kubectl을 통해 Kubernetes 클러스터에 전달하면, Kubernetes는 이 파일의 내용대로 리소스를 생성하고 관리해 줍니다.

> 4가지 필수 필드

1. apiVersion

> "이 YAML 파일은 어떤 버전의 API 규칙을 사용해서 작성되었으니, 이 버전에 맞게 해석해"

- 만들려는 리소스의 종류(kind)에 따라 사용할 수 있는 API 버전이 정해져 있습니다.

v1

Pod, Service, ConfigMap 등 kubernetes의 핵심 오브젝트에 사용됩니다.

apps/v1

Deployment, ReplicaSet, StatefulSet 등 애플리케이션 배포와 관련된 오브젝트에 사용됩니다.

batch/v1

Job, CronJob 등 배치 작업에 사용됩니다.

2. kind (리소스 종류)

> 이 Manifest 파일이 무엇을 만들고 싶은지를 명시하는 필드입니다.

웹 서버 컨테이너를 실행하고 싶다면 Pod 또는 Deployment를, 이 컨테이너를 외부에 노출시키고 싶다면 Service를 지정합니다.

> 종류

Pod

- 컨테이너를 실행하는 가장 작은 기본 단위입니다.

Deployment

- Pod의 개수를 원하는 대로 유지하고, 업데이트를 관리하는 역할을 수행합니다.

Service

- 여러 Pod에 대한 고정된 접속 주소를 제공합니다.

ConfigMap / Secret

- 설정값이나 비밀번호 등의 민감한 정보를 코드와 분리해서 관리합니다.

Namespace

- 하나의 클러스터를 여러 개의 가상 클러스터처럼 나누어 사용할 때 쓰는 논리적인 격리 공간입니다.

3. metadata (리소스의 정보)

> 생성될 리소스를 식별하기 위한 이름, 소속(네임스페이스), 별명(라벨) 등 부가 정보를 담는 곳입니다.

> 주요 metadata 필드

name (필수)

- 이 리소스의 고유한 이름. 같은 namespace 안에서는 이름이 중복될 수 없습니다.

namespace

- 이 리소스가 속할 네임스페이스를 지정합니다. 지정하지 않으면 default 네임스페이스에 생성됩니다.

labels

- 리소스를 구분하고 그룹화하기 위한 key: value 형태의 별명입니다.
- Service가 특정 Pod들을 찾거나, 여러 리소스를 한 번에 관리할 때 아주 유용하게 사용됩니다.

4. spec

- (중요) kind에서 정의한 리소스를 어떤 상태로 만들고 운영할 것인지에 대한 상세한 명세를 작성하는 공간입니다.
- kind가 무엇이냐에 따라 완전히 달라집니다.

> Deployment의 spec 예시

```
spec:
  replicas: 3 # 똑같은 Pod 3개 유지
  selector: # 이 Deployment는 아래 label을 가진 Pod들을 관리
    matchLabels:
      app: my-nginx
  template: # 이 template을 보고 Pod를 만들어라
    metadata: # Pod에 붙일 metadata
      labels:
        app: my-nginx # 반드시 selector의 matchLabels와 일치해야 한다
    spec: # Pod의 상세 명세
      containers:
      - name: nginx-container # 컨테이너 이름
        image: nginx:1.21 # 사용할 도커 이미지
        ports:
        - containerPort: 80 # 이 컨테이너는 80번 포트를 사용한다.
```

Manifest 예시 파일 이해해보기

```yaml
# 앱 배포와 관련된 apps/v1 사용
apiVersion: apps/v1

# 리소스 종류: Pod를 관리하는 Deployment를 생성
kind: Deployment

# 리소스 정보: 이름과 라벨을 지정한다
metadata:
  name: my-nginx-deployment
  labels:
    app: my-nginx

# 상세 명세: 어떻게 동작할지 정의
spec:
  # Pod를 3개 복제해서 생성
  replicas: 3
  # 'app: my-nginx' 라벨이 붙은 Pod를 찾아 관리
  selector:
    matchLabels:
      app: my-nginx

  # template: 위 selector 조건에 맞는 Pod를 생성하기 위한 설계도
  template:
    metadata:
      # Pod에 'app: my-nginx' 라벨 붙여줌
      labels:
        app: my-nginx
    spec:
    # Pod 안에서 실행될 컨테이너 목록
      containers:
      - name: nginx-container # 컨테이너 이름
        image: nginx:latest # 사용할 도커 이미지
        ports:
        - containerPort: 80 # 컨테이너가 사용할 포트
```