

# Contents

- ConfigMap
- Secret



### ConfigMap

ConfigMap은 Key-Value 쌍으로 기밀이 아닌 데이터를 저장하는데 사용하는 API Object입니다. Pod는 볼륨에서 환경변수, 명령줄 인수 또는 구성(Config.) 파일로 ConfigMap을 사용할 수 있습니다.

ConfigMap을 사용하면 컨테이너 이미지에서 환경별 구성(Config.)을 분리하여, 애플리케이션을 쉽게 이식할 수 있습니다.

ConfigMap은 많은 양의 데이터를 보유하도록 설계되지 않았습니다. ConfigMap에 저장된 데이터는 1MiB를 초과할 수 없습니다. 이 제한보다 큰 설정을 저장해야 하는 경우, Volume을 마운트하는 것을 고려하거나 별도의 데이터베이스 또는 파일 서비스를 사용할 수 있습니다.

ConfigMap은 다른 Kubernetes object들과 달리 `.spec` 대신 `data` 와 `binaryData` 를 가지고 있습니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: game-demo
data:
  # 속성과 비슷한 키; 각 키는 간단한 값으로 맵핑됨
  player_initial_lives: "3"
  ui_properties_file_name: "user-interface.properties"

  # 파일과 비슷한 키
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
  user-interface.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
```

### ConfigMap

위와같이 생성된 Configmap을 사용할 때는 다음과 같이 할 수 있습니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-demo-pod
spec:
  containers:
    - name: demo
      image: alpine
      command: ["sleep", "3600"]
      env:
        # 환경 변수 정의
        - name: PLAYER_INITIAL_LIVES # 참고로 여기서는 컨피그맵의 키 이름과 대소문자가 다르다.
          valueFrom:
            configMapKeyRef:
              name: game-demo           # 이 값의 컨피그맵.
              key: player_initial_lives # 가져올 키.
        - name: UI_PROPERTIES_FILE_NAME
          valueFrom:
            configMapKeyRef:
              name: game-demo
              key: ui_properties_file_name
  ...

```

환경변수(`PLAYER_INITIAL_LIVES`, `UI_PROPERTIES_FILE_NAME`)의 값으로 Configmap에 정의된 값(`3`, `user-interface.properties`)을 사용함.

### ConfigMap

앞의 파일에서 이어진 파일입니다.

```
...
volumeMounts:
- name: config
  mountPath: "/config"
  readOnly: true
volumes:
# 파드 레벨에서 볼륨을 설정한 다음, 해당 파드 내의 컨테이너에 마운트한다.
- name: config
  configMap:
    # 마운트하려는 컨피그맵의 이름을 제공한다.
    name: game-demo
    # 컨피그맵에서 파일로 생성할 키 배열
    items:
      - key: "game.properties"
        path: "game.properties"
      - key: "user-interface.properties"
        path: "user-interface.properties"
```

Volume을 정의하고 mountPath(/config)에 두 개의 파일( `game.properties` , `user-interface.properties` )을 마운트

### Secret

Secret은 암호, 토큰 또는 키와 같은 소량의 중요한 데이터를 포함하는 Object입니다. 이를 사용하지 않으면 중요한 정보가 파드 명세나 컨테이너 이미지에 포함될 수 있다. Secret을 사용한다는 것은 사용자의 기밀 데이터를 애플리케이션 코드에 넣을 필요가 없음을 뜻합니다.

Secret은 Secret을 사용하는 Pod와 독립적으로 생성될 수 있기 때문에, 파드를 생성하고, 확인하고, 수정하는 워크플로우 동안 Secret이 노출되는 것에 대한 위험을 경감시킬 수 있습니다.

Secret은 Configmap과 유사하지만 특별히 기밀 데이터를 보관하기 위한 것입니다.

간단한 사용예를 보겠습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  USER_NAME: YWRtaW4=
  PASSWORD: MWYyZDFlMmU2N2Rm
```

USER\_NAME 과 PASSWORD 를 가진 secret

### Secret

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-test-pod
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "/bin/sh", "-c", "env" ]
      envFrom:
        - secretRef:
            name: mysecret
  restartPolicy: Never
```

환경변수로 앞에서 생성된 Secret의 정보를 사용

▶ Hands-on : 12\_Kubernetes\_ConfigMaps & Secrets

### Summary

- ConfigMap
- Secret