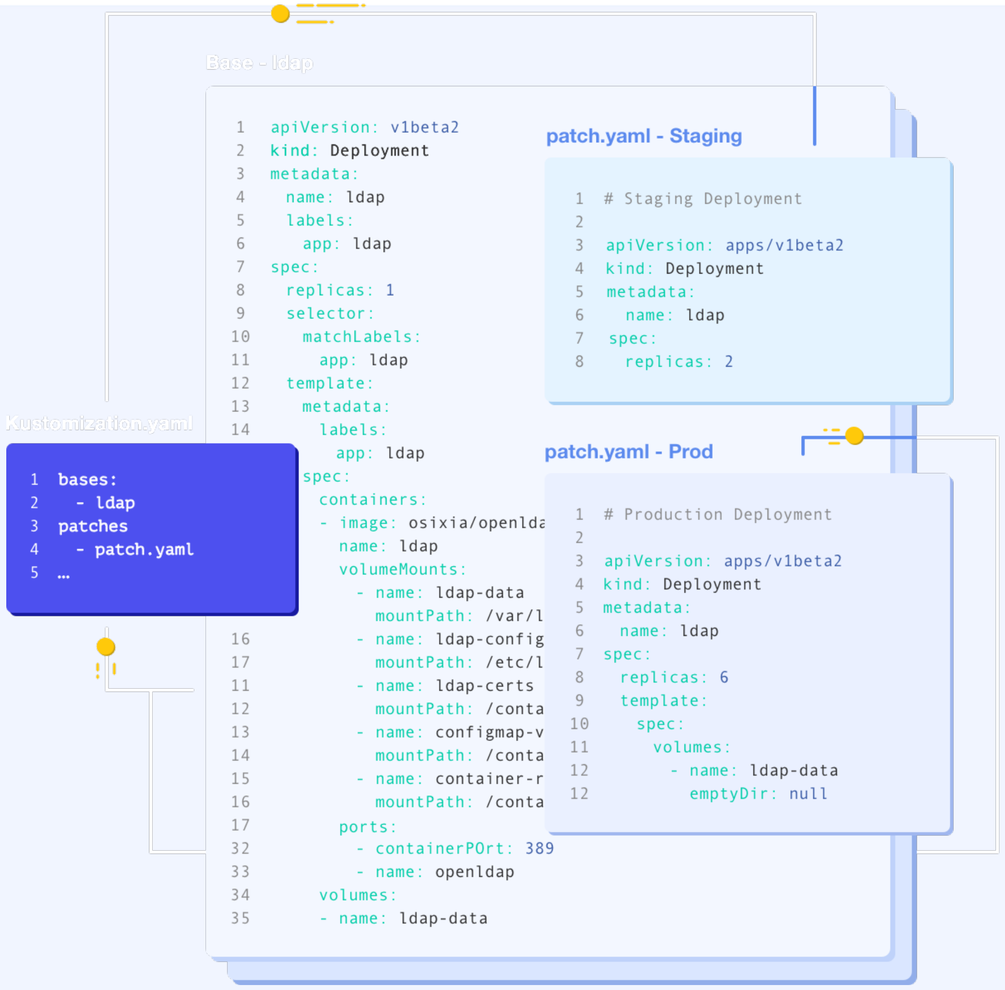
4 Kubernetes Manifest 작성을 위한 Helm 활용

02 Kustomize 소개

Kustomize란?



출처 : https://kustomize.io/

Kustomize 기본 명령어

kustomization 파일을 포함하는 디렉터리 내의 리소스 확인 방법

$ kubectl kustomize <kustomization 디렉토리>

Kustomize 리소스 적용

$ kubectl apply --kustomize <kustomization 디렉토리>

$ kubectl apply –k <kustomization 디렉토리>

기본 kustomization 파일 구성 #1

kustomization.yaml

```
resources:
- deployment.yaml
- service.yaml
```

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: my-nginx
  labels:
    run: my-nginx
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    run: my-nginx
```

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx
        ports:
        - containerPort: 80
```

기본 kustomization 파일 구성 #2

Kustomize 방식의 Manifest를 Kubernetes에 적용하는 방법

$ kubectl apply --kustomize <Kustomization 디렉토리>

Kustomization 디렉토리 구조

```
~/someApp
├── deployment.yaml
├── kustomization.yaml
└── service.yaml
```

커스텀 kustomization 파일 구성

patchesStrategicMerge	patchesJson6902
• 패치하려는 대상을 Kustomization에 병합하여 패치를 수행하는 방식(대상은 파편화, 조각화된 패치)	• kubernetes object의 변경을 JSONPatch 규약을 따름 • https://datatracker.ietf.org/doc/html/rfc6902
• 패치하려는 Patch Manifest 내부의 Object 네임은 반드시 대상으로 지정한 리소스 네임과 기본 템플릿과 일치해야함	• 패치하려는 대상의 정보는 Patch Manifest 내부의 Object path에 맞춰서 value값을 입력
• Manifest당 Object를 교체/병합/삭제하는 패치가 권장됨	• Json 패치의 정확한 리소스를 찾기 위해, 리소스의 group, version, kind, name을 kustomization.yaml 내에 명시

patchesStrategicMerge 방식 #1

설정 항목	설정 내용	설정 예
replace	replace를 포함하는 요소가 병합되는 대신 대체	containers: - name: nginx image: nginx-1.0 - $patch: replace
merge	merge를 포함하는 요소가 대체되는 대신 병합	containers: - name: nginx image: nginx-1.0 - name: log-tailer image: fluentd:latest
delete	delete를 포함하는 요소가 삭제	containers: - name: nginx image: nginx-1.0 - $patch: delete name: log-tailer

patchesStrategicMerge 방식 #2

kustomization.yaml

```
resources:
- deployment.yaml
patchesStrategicMerge:
- incr_replica.yaml
- set_memory.yaml
```

set_memory.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  template:
    spec:
      containers:
      - name: my-nginx
        resources:
          limits:
            memory: 512Mi
```

incr_replica.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  replicas: 3
```

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx
        ports:
        - containerPort: 80
```

patchesJson6902 방식 #1

설정 항목	설정 내용	설정 예
add	path에 명시된 value를 추가	- op: add path: /spec/replicas value: 2
replace	path에 명시된 value가 대체	- op: replace path: /spec/replicas value: 3
remove	path에 명시된 value가 삭제	- op: remove path: /spec/replicas

patchesJson6902 방식 #2

kustomization.yaml

```
resources:
- deployment.yaml

patchesJson6902:
- target:
    group: apps
    version: v1
    kind: Deployment
    name: my-nginx
  path: patch.yaml
```

patch.yaml

```
- op: replace
  path: /spec/replicas
  value: 3
```

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx
        ports:
        - containerPort: 80
```

Kustomize 사용방법 – Base와 Overlay 디렉토리 구성 #1

Base 및 Overlay를 사용한 수정된 Manifest를 적용하는 방법

$ kubectl apply --kustomize <Base or Overlay 디렉토리>

Overlay를 사용한 수정 파일 디렉토리 구조

```
~/someApp
├── base
│   ├── deployment.yaml
│   ├── kustomization.yaml
│   └── service.yaml
└── overlays
    ├── dev
    │   ├── cpu_count.yaml
    │   ├── kustomization.yaml
    │   └── replica_count.yaml
    └── prod
        ├── cpu_count.yaml
        ├── kustomization.yaml
        └── replica_count.yaml
```

Kustomize 사용방법 – Base와 Overlay 디렉토리 구성 #2 (Base)

kustomization.yaml

```
resources:
- deployment.yaml
- service.yaml
```

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: my-nginx
  labels:
    run: my-nginx
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    run: my-nginx
```

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx
```

Kustomize 사용방법 – Base와 Overlay 디렉토리 구성 #3 (Overlay)

dev/kustomization.yaml

```
bases:
- ../base
namePrefix: dev-
```

prod/kustomization.yaml

```
bases:
- ../base
namePrefix: prod-
```