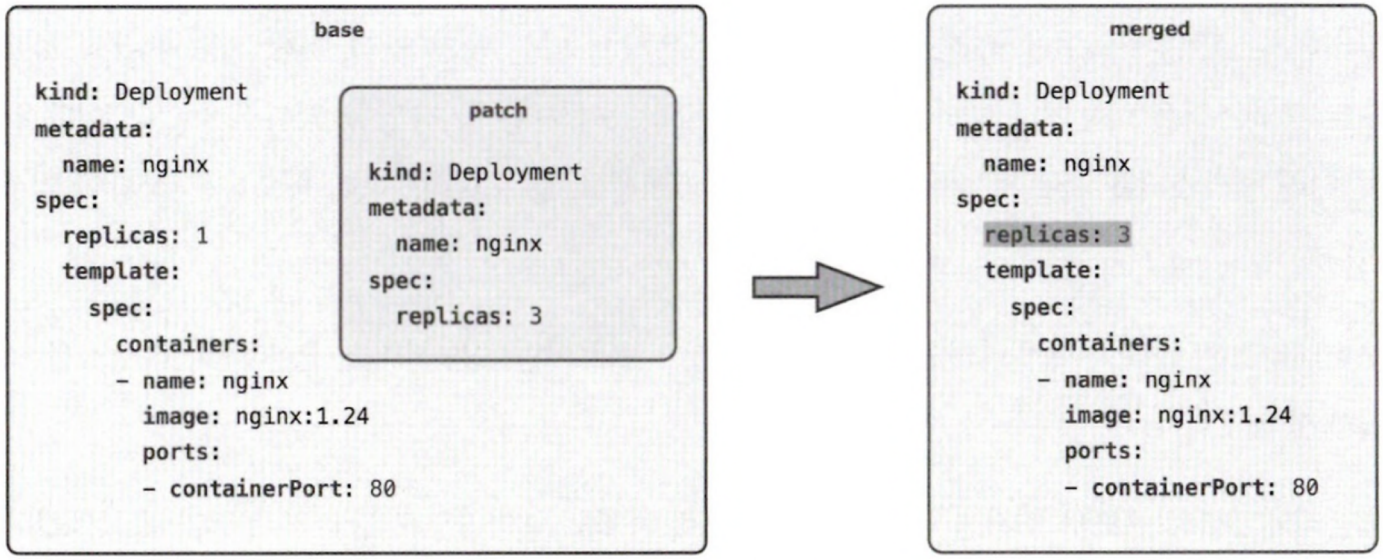
Packaging

Kustomize

❖ 개요

- ✓ Kustomize는 쿠버네티스 매니페스트 관리 도구로 매니페스트의 구성 관리를 구현
- ✓ Kustomize의 기본인 매니페스트 생성 기능은 kubectl도 제공
- ✓ base 매니페스트의 변경 위치를 정의한 패치 매니페스트를 준비하고 병합된 매니페스트를 출력



Kustomize

- ❖ 개요
  - ✓ Kustomize가 제공하는 다른 기능
    - 여러 매니페스트 파일을 하나의 매니페스트 파일로 병합
    - 레이블 등 공통 부분을 설정하여 매니페스트 파일의 중복 감소
    - 오버레이의 재사용성을 통해 복수의 환경 구성 관리가 용이
    - 시크릿 생성

Kustomize

- ❖ 개요
  - ✓ 설치: kubectl에도 포함되어 있지만 다양한 기능을 사용하고자 하는 경우 설치
    - asdf
      - ◆ 여러 언어와 도구의 버전을 한 곳에서 관리할 수 있는 통합 버전 관리자
      - ◆ 프로그래밍 언어마다 별도의 버전 관리 도구(Node.js의 nvm, Python의 pyenv, Ruby의 rbenv 등)가 존재하는데 asdf는 이러한 도구들을 하나로 통합하여 하나의 설정 파일(.tool-versions)로 모든 도구의 버전을 제어
      - ◆ asdf 설치
        - sudo apt update
        - sudo apt install curl git
        - git clone https://github.com/asdf-vm/asdf.git ~/.asdf --branch v0.14.0
        - echo '. "$HOME/.asdf/asdf.sh"' >> ~/.bashrc
        - source ~/.bashrc
        - asdf version
        - asdf reshim
      - ◆ asdf를 이용한 kustomize 설치
        - asdf plugin add kustomize
        - asdf install kustomize 5.3.0
        - asdf global kustomize 5.3.0

Kustomize

- 기본적인 사용
  - ✓ 작업용 디렉토리 생성: mkdir -p echo/base/
  - ✓ 작업 디렉토리 이동: cd echo/base/

Kustomize

- 기본적인 사용
  - ✓ Deployment Manifest: deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: echo
  labels:
    app.kubernetes.io/name: echo
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: echo
  template:
    metadata:
      labels:
        app.kubernetes.io/name: echo
```

Kustomize

- 기본적인 사용
  - ✓ Deployment Manifest: deployment.yaml

```
spec:
  containers:
  - name: nginx
    image: ghcr.io/jpubdocker/simple-nginx-proxy:v0.1.0
    env:
    - name: NGINX_PORT
      value: "80"
    - name: SERVER_NAME
      value: "localhost"
    - name: BACKEND_HOST
      value: "localhost:8080"
    - name: BACKEND_MAX_FAILS
      value: "3"
    - name: BACKEND_FAIL_TIMEOUT
      value: "10s"
    ports:
    - name: http
      containerPort: 80
  - name: echo
    image: ghcr.io/jpubdocker/echo:v0.1.0
```

Kustomize

- 기본적인 사용
  - ✓ Service Manifest: service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: echo
  labels:
    app.kubernetes.io/name: echo
spec:
  selector:
    app.kubernetes.io/name: echo
  ports:
  - name: echo
    port: 80
    targetPort: http
    protocol: TCP
```

Kustomize

- 기본적인 사용
  - ✓ Ingress Manifest: ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: echo
  labels:
    app.kubernetes.io/name: echo
spec:
  ingressClassName: nginx
  rules:
  - host: echo.jpub.local
    http:
      paths:
      - pathType: Prefix
        path: /
        backend:
          service:
            name: echo
            port:
              number: 80
```

Kustomize

- 기본적인 사용
  - ✓ 매니페스트 정리
    - 매니페스트 파일을 정리하기 위해서는 kustomization.yaml 파일을 생성해야 하는데 이렇게 매니페스트 파일 결합이 Kustomize의 기본 기능
    - kustomize create --autodetect 명령을 사용하면 현재 디렉토리 내의 매니페스트 파일을 정리하기 위한 kustomization.yaml 파일을 생성할 수 있음
    - kustomize create --autodect 명령으로 생성된 kustomization.yaml 파일의 내용

      apiVersion: kustomize.config.k8s.io/v1beta1

      kind: Kustomization

      resources:

      - deployment.yaml

      - ingress.yaml

      - service.yaml

      - ◆ resources에는 구성 관리의 대상이 되는 매니페스트 파일이 현재 디렉토리의 상대 경로로 나열됨

Kustomize

- 기본적인 사용
  - ✓ 매니페스트 정리
    - 정리된 매니페스트 파일의 출력은 kustomize build .

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/name: echo
  name: echo
spec:
  ports:
  - name: echo
    port: 80
    protocol: TCP
    targetPort: http
  selector:
    app.kubernetes.io/name: echo
---
apiVersion: apps/v1
kind: Deployment
```

Kustomize

- ❖ 기본적인 사용
  - ✓ 매니페스트 정리
    - 파이프를 이용해서 배포: kustomize build . | kubectl apply -f -

```
service/echo created
deployment.apps/echo created
ingress.networking.k8s.io/echo created
```

    - 확인: kubectl get pod,svc,ingress

```
NAME                        READY   STATUS    RESTARTS   AGE
pod/echo-748556c654-k74v5   2/2     Running   0          3m1s

NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/echo         ClusterIP   10.107.76.138   <none>        80/TCP    3m1s
service/kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP   50m

NAME                                CLASS   HOSTS             ADDRESS          PORTS
AGE
ingress.networking.k8s.io/echo   nginx   echo.jpub.local   10.100.254.224   80
3m1s
```

Kustomize

- ❖ 기본적인 사용
  - ✓ 매니페스트 정리
    - 삭제: kustomize build . | kubectl delete -f -

      service "echo" deleted

      deployment.apps "echo" deleted

      ingress.networking.k8s.io "echo" deleted

Kustomize

- 기본적인 사용
  - ✓ 공통 부분 설정
    - 쿠버네티스에서 다양한 매니페스트를 생성하면 레이블(.metadata.labels)과 같이 매니페스트에 나타나는 몇 가지 메타 데이터가 존재하는 것을 알 수 있는데 Kustomize는 이와 같은 공통 부분도 간단하게 관리 가능
    - echo 애플리케이션의 매니페스트는 app. kubernetes. io/name:echo 레이블을 매니페스트 여러 곳에 사용
    - deployment.yaml 파일에서 공통 부분 주석 처리

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: echo
  # labels:
    # app.kubernetes.io/name: echo
spec:
  replicas: 1
  # selector:
    # matchLabels:
      # app.kubernetes.io/name: echo
  template:
    metadata:
      # labels:
        # app.kubernetes.io/name: echo
```

Kustomize

- ❖ 기본적인 사용
  - ✓ 공통 부분 설정
    - service.yaml 파일에서 공통 부분 주석 처리

```
apiVersion: v1
kind: Service
metadata:
  name: echo
  # labels:
    # app.kubernetes.io/name: echo
spec:
  # selector:
    # app.kubernetes.io/name: echo
```

    - ingress.yaml 파일에서 공통 부분 주석 처리

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: echo
  # labels:
    # app.kubernetes.io/name: echo
```

Kustomize

- ❖ 기본적인 사용
  - ✓ 공통 부분 설정
    - 레이블 설정: kustomize edit set label "app.kubernetes.io/name:echo"
    - kustomization.yaml 파일 확인

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
- deployment.yaml
- ingress.yaml
- service.yaml
commonLabels:
  app.kubernetes.io/name: echo
```

Kustomize

- 기본적인 사용
  - ✓ 공통 부분 설정
    - 빌드 결과 출력: kustomize build .

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/name: echo
  name: echo
spec:
  ports:
  - name: echo
    port: 80
    protocol: TCP
    targetPort: http
  selector:
    app.kubernetes.io/name: echo
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/name: echo
  name: echo
```

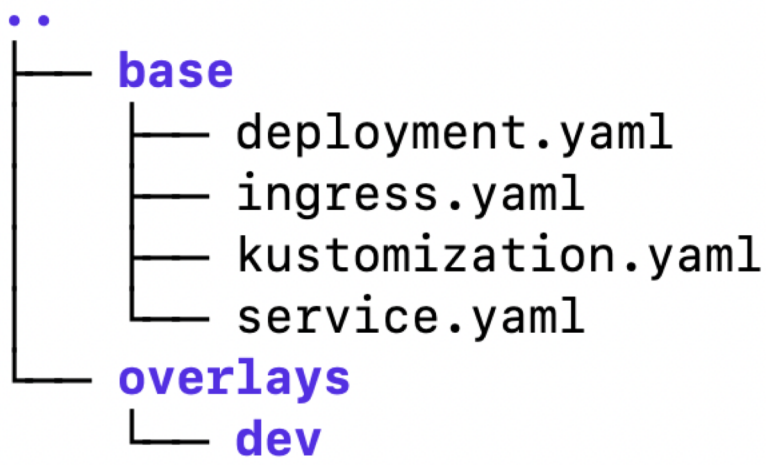
Kustomize

- ❖ 기본적인 사용
  - ✓ 공통 부분 설정
    - ● 빌드 결과 출력: kustomize build .
      - ◆ commonLabels는 .metadata.labels뿐만 아니라 디플로이먼트의 .spec.selector.matchLabels와 서비스의 .spec.selector 같은 레이블을 사용하는 항목에도 적용됨
      - ◆ 디플로이먼트의 .spec.selector.matchLabels와 서비스의 .spec.selector 같은 레이블을 사용하는 항목에도 적용됨
      - ◆ # Warning: 'commonLabels' is deprecated. Please use 'labels' instead. Run 'kustomize edit fix' to update your Kustomization automatically. 라는 경고가 발생하는데 kustomize edit fix 명령을 수행한 후 kustomization.yaml 파일이 수정됨

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
- deployment.yaml
- ingress.yaml
- service.yaml
labels:
- includeSelectors: true
  pairs:
    app.kubernetes.io/name: echo
```

Kustomize

- 재사용과 부분 오버레이
  - ✓ dev 환경을 가정하고 이 환경에 적합한 매니페스트의 구성 관리를 진행
  - ✓ Kustomize는 설정 파일의 위치를 overlays/[환경] 경로에 두는 것이 일반적이므로 echo 디렉토리에 overlays/dev 디렉토리를 생성: mkdir -p ../overlays/dev
    - 현재 디렉토리 구조



  - ✓ 여러 환경에 배포하면 각 환경마다 다른 설정이 필요한데 다른 설정의 예로는 데이터베이스, API, 파드 수, 도메인 등이 있음
  - ✓ base 디렉토리에 위치한 매니페스트를 재사용하여 각 환경의 필요에 따라 설정을 오버레이 할 수 있음

Kustomize

- 재사용과 부분 오버레이
  - ✓ 다른 디렉토리의 매니페스트 파일을 이용해서 Kustomize
    - 다른 디렉토리의 리소스 매니페스트 파일을 이용해서 생성할 때는 kustomize create --resources 매니페스트파일디렉토리경로
    - dev 디렉토리에서 kustomization.yaml을 생성하는데 base 디렉토리의 매니페스트를 사용할 수 있도록 설정: kustomize create --resources ../../base
    - kustomization.yaml

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
- ../../base
```

Kustomize

- ❖ 재사용과 부분 오버레이
  - ✓ 디플로이먼트 패치 파일
    - Kustomize에는 매니페스트 파일을 그대로 패치 파일로 다루고 쿠버네티스에 최적의 형태로 병합해주는 Strategic Merge Patch 구조가 존재
    - patch-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: echo
  spec:
    replicas: 2 # 오버레이하고 싶은 데이터
  template:
    spec:
      containers:
      - name: nginx
        env:
          - name: BACKEND.MAX.FAILS
            value: "5" # 오버레이하고 싶은 데이터
```

Kustomize

- 재사용과 부분 오버레이
  - ✓ 인그레스 패치 파일
    - Kustomize에서 인그레스를 변경할 때는 주로 호스트(.spec.rules[0].host)를 업데이트하지만 Strategic Merge Patch 방식으로는 구현이 어려움
    - Strategic Merge Patch 방식은 업데이트 위치를 식별하는 키를 사용하므로 패치 파일로 이 값을 변경하면 오버레이할 수 없음
    - Strategic Merge Patch 방식이 맞지 않는 경우

```
# ...
spec:
  ingressClassName: nginx
  rules:
  - host: echo.jpub.local # 이 부분을 오버레이하고 싶지만 업데이트 위치를 식별하는 값으로 사용
    http:
```

Kustomize

- 재사용과 부분 오버레이
  - ✓ 인그레스 패치 파일
    - 이와 같은 케이스를 해결하기 위해 Kustomize는 JSON Patch 방식을 패치에도 사용
    - JSON Patch 방식은 JSON으로 변환 가능한 YAML 파일로 패치를 작성할 수 있는데 Strategic Merge Patch보다 작성하는 양은 증가하지만 Strategic Merge Patch로는 할 수 없는 패치 방법을 적용할 수 있음
    - patch-ingress.yaml

      - op: replace # 1 동작

      path: "/spec/rules/0/host" # 2 대상 위치

      value: "dev-echo.jpub.local" # 3 변경하는 데이터

Kustomize

- 재사용과 부분 오버레이
  - ✓ 패치 파일 추가
    - Strategic Merge Patch 방식인 디플로이먼트는 --patch 옵션으로 패치 파일을 지정하는 것으로 충분하며 JSON Patch 방식인 인그레스는 패치 파일만으로는 전달할 수 있는 정보가 적으므로 --kind와 --name 옵션을 사용해 대상 리소스를 특정할 수 있도록 적용
    - 패치 파일 추가 명령
      - kustomize edit add patch --path patch-deployment.yaml
      - kustomize edit add patch --kind Ingress --path patch-ingress.yaml --name echo
    - kustomization.yaml

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
- ../../base
patches:
- path: patch-deployment.yaml
- path: patch-ingress.yaml
  target:
    kind: Ingress
    name: echo
```

Kustomize

- 재사용과 부분 오버레이
  - ✓ Kustomize는 베이스가 되는 매니페스트를 재사용하여 각 환경에 적용할 수 있음
  - ✓ 이 구조를 통해 매니페스트의 중복을 줄이고 효율적인 구성 관리를 할 수 있음
  - ✓ 매니페스트의 VCS 버전 관리는 kustomize build 커맨드로 출력하는 완성형 매니페스트를 관리하는 것이 아니라 base나 overlays/dev와 같은 디렉토리를 그대로 리포지터리에서 관리

Kustomize

- Secret 사용
  - ✓ Kustomize는 미리 생성한 시크릿 매니페스트를 관리 대상으로 하는 것이 아니라 완성된 매니페스트를 출력할 때 시크릿의 매니페스트를 생성
  - ✓ 시크릿의 매니페스트는 버전 관리 대상에서 제외할 수 있음
  - ✓ statefulset.yaml 매니페스트 파일을 생성하는데 레이블은 kustomization.yaml에서 설정하므로 이 파일에서는 설정하지 않음
  - ✓ 작업 디렉토리 생성: mysql

Kustomize

- Secret 사용
  - ✓ Statefulset 매니페스트: statefulset.yaml

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
spec:
  serviceName: "mysql"
  replicas: 1
  template:
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: mysql
        image: ghcr.io/jpubdocker/taskapp-mysql:v1.0.0
        env:
          - name: MYSQL_ROOT_PASSWORD_FILE
            value: /var/run/secrets/mysql/root_password
          - name: MYSQL_DATABASE
            value: taskapp
          - name: MYSQL_JSER
            value: taskapp_user
          - name: MYSQL_PASSWORD_FILE
            value: /var/run/secrets/mysql/user_password
```

Kustomize

- Secret 사용
  - ✓ Statefulset 매니페스트: statefulset.yaml

```
        ports:
          - containerPort: 3306
            name: mysql
        volumeMounts:
          - name: mysql-persistent-storage
            mountPath: /var/lib/mysql
          - name: mysql-secret
            mountPath: "/var/run/secrets/mysql"
            readOnly: true
      volumes:
        - name: mysql-secret
          secret:
            secretName: mysql
  volumeClaimTemplates:
    - metadata:
        name: mysql-persistent-storage
      spec:
        accessModes: [ "ReadWriteOnce" ]
        resources:
          requests:
            storage: 1Gi
```

Kustomize

- Secret 사용
  - ✓ 매니페스트 파일 정리: kustomize create --autodetect
  - ✓ 생성된 kustomization.yaml

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
- statefulset.yaml
```

Kustomize

- ❖ Secret 사용
  - ✓ secretGenerator
    - 쿠버네티스의 시크릿에 등록된 데이터는 Base64로 인코딩된 값이다.
    - 수동으로 시크릿 매니페스트 파일을 생성하면 해당 파일을 버전 관리 대상에서 제외해야 하는 번거로움이 발생
    - Kustomize는 시크릿을 생성하는 secretGenerator라는 구조를 제공
    - 비밀번호 파일 생성
      - ◆ echo -n "wnddkd" > mysql_root_password
      - ◆ echo -n "adam" > mysql_user_password
    - secretGenerator 설정 추가
      - ◆ kustomize edit add secret mysql --from-file=root_password=./mysql_root_password
      - ◆ kustomize edit add secret mysql --from-file=user_password=./mysql_user_p
    - 레이블을 설정
      - ◆ kustomize edit set label "app.kubernetes.io/component:mysql"

Kustomize

- ❖ Secret 사용
  - ✓ secretGenerator
    - mysql의 Service 매니페스트 파일: service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  ports:
  - protocol: TCP
    port: 3306
    targetPort: 3306
  clusterIP: None
```

    - mysql의 Service 매니페스트 파일을 Kustomize 관리 대상으로 추가: kustomize edit add resource service.yaml

Kustomize

- ❖ Secret 사용
  - ✓ secretGenerator
    - ● 완성된 kustomization.yaml

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
- statefulset.yaml
- service.yaml
secretGenerator:
- files:
  - root_password=./mysql_root_password
  - user_password=./mysql_user_password
  name: mysql
  type: Opaque
commonLabels:
  app.kubernetes.io/component: mysql
```

Kustomize

- Secret 사용
  - ✓ secretGenerator
    - 시크릿을 포함하는 완성된 매니페스트: kustomize build .

```
lly.
apiVersion: v1
data:
  root_password: d25kZGtk
  user_password: YWRhbQ==
kind: Secret
metadata:
  labels:
    app.kubernetes.io/component: mysql
  name: mysql-624tf72mk4
type: Opaque
```

Kustomize

- Secret 사용
  - ✓ Kustomize의 secretGenerator를 사용하면 시크릿의 매니페스트 파일을 직접 다루지 않아도 되므로 비교적 안전하게 보안 정보를 다룰 수 있음
  - ✓ 이 방법으로 Kustomize를 사용해 mysql을 구축할 수 있으며 다른 컴포넌트도 동일한 방식으로 매니페스트를 구성하여 생성할 수 있음

Kustomize

- 네트워크를 경유해서 매니페스트 생성
  - ✓ 매니페스트 파일과 같이 특정 리소스에 대한 kustomization.yaml이 공개되어 있을 때는 이 파일만 있으면 바로 배포할 수 있음
  - ✓ Ingress NGINX Controller의 kustomization.yaml은 간단하게 deploy.yaml만 관리
    - curl https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.8.1/deploy/static/provider/cloud/kustomization.yaml

```
# NOTE: kustomize is not supported. This file exists only to be able to reference it from bases.
# https://kubectl.docs.kubernetes.io/references/kustomize/bases/
#
# ```
# namespace: ingress-nginx
# bases:
#   - github.com/kubernetes/ingress-nginx/tree/main/deploy/static/provider/cloud
# ```

resources:
  - deploy.yaml
```

Kustomize

- 네트워크를 경유해서 매니페스트 생성
  - ✓ kustomization.yaml에서 GitHub 파일은 https://github.com/[유저]/[리포지터리].git//[kustomization.yaml 디렉토리]/?ref=[태그명] 형식으로 참조할 수 있음
  - ✓ Ingress NGINX Controller는 https://github.com/kubemetes/ingressnginx.git//deploy/static/provider/cloud/?ref=controller-v1.8.1 로 참조할 수 있음
  - ✓ Ingress-nginx를 관리 대상으로 추가한 kustomization.yaml 생성
    - mkdir ingress-nginx
    - cd ingress-nginx
    - kustomize create --resources "https://github.com/kubernetes/ingress-nginx.git//deploy/static/provider/cloud?ref=controller-v1.8.1"
  - ✓ kustomization.yaml 파일 확인
    - apiVersion: kustomize.config.k8s.io/v1beta1
    - kind: Kustomization
    - resources:
    - - https://github.com/kubernetes/ingress-nginx.git//deploy/static/provider/cloud?ref=controller-v1.8.1

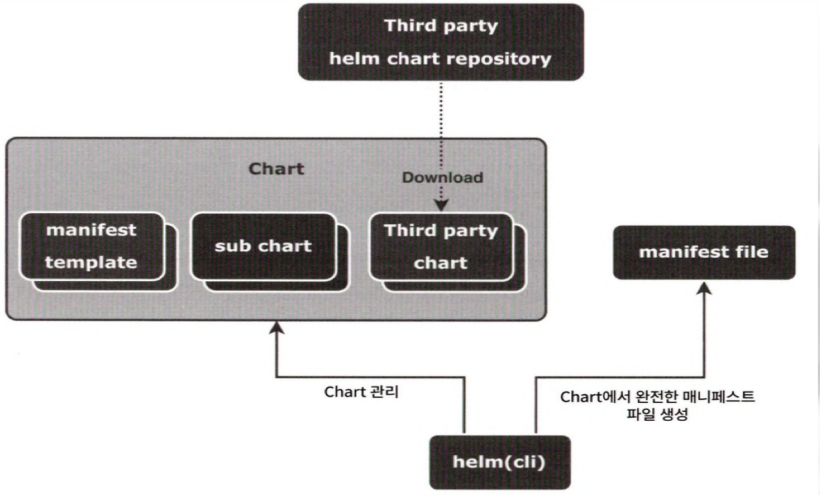
Kustomize

- 네트워크를 경유해서 매니페스트 생성
  - ✓ Ingress-nginx 완성된 매니페스트: kustomize build .

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/name: ingress-nginx
  name: ingress-nginx
---
apiVersion: v1
automountServiceAccountToken: true
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
    app.kubernetes.io/version: 1.8.1
  name: ingress-nginx
  namespace: ingress-nginx
```

Helm

- 개요
  - ✓ 헬름(Helm)은 CNCF가 관리하는 쿠버네티스의 패키지 도구: Helm is a tool for managing Charts. Charts are packages of pre-configured Kubernetes resources.
  - ✓ 쿠버네티스와 헬름의 관계는 레드헷 계열 리눅스에서의 Yum이나 데비안 계열 리눅스에서의 APT에 해당하는 관계라고 할 수 있음
  - ✓ 헬름은 오픈 소스 소프트웨어로 공개되어 있으며 이미 많은 패키지(Chart)가 있는데 예를 들어 레디스 클러스터나 워드프레스 환경 등의 소프트웨어를 하나의 명령어로 쿠버네티스 클러스터에 배포할 수 있으며 롤링 업데이트 등에도 지원하는 것들이 많아 쿠버네티스에서 최적화된 설정으로 사용할 수 있는 장점도 있음
  - ✓ Helm, Chart, Manifest 관계



Helm

- ❖ 개요
  - ✓ Helm, Chart, Manifest 관계
    - Chart는 여러 매니페스트 파일의 템플릿을 정리할 수 있는데 대규모 애플리케이션이라면 매니페스트 템플릿도 많아지고 관리도 복잡해지므로 이를 하위 Chart로 분할하여 관리할 수 있음
    - Chart는 다른 Chart와 의존 관계를 가질 수도 있음
    - 서드 파티의 Helm 차트 리포지터리에서 Chart를 다운로드하고 사용할 수 있음
    - Helm으로 Chart를 관리하여 번거로운 매니페스트 파일을 쉽게 관리하는 것이 목적
    - Helm을 통해 여러 클러스터의 배포도 간결하게 관리할 수 있음
    - Helm은 강력한 패키지 관리자이며 쿠버네티스 애플리케이션의 종합적인 관리 도구로 사용
    - 실제 개발에서 로컬 환경과 운영 환경의 클러스터에 상관없이 여러 환경에 배포하기 위한 애플리케이션이라면 Chart로 패키징을 실시하고, kubectl이 아닌 Helm을 통해 배포와 업데이트를 완료할 수 있음
    - kubectl은 배포된 리소스 수정 시 사용

Helm

- Helm 설치
  - ✓ Ubuntu에서 설치
    - sudo apt-get install curl gpg apt-transport-https --yes
    - curl -fsSL https://packages.buildkite.com/helm-linux/helm-debian/gpgkey | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null
    - echo "deb [signed-by=/usr/share/keyrings/helm.gpg] https://packages.buildkite.com/helm-linux/helm-debian/any/ any main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list
    - sudo apt-get update
    - sudo apt-get install helm
  - ✓ Fedora에서 설치: sudo dnf install helm
  - ✓ asdf로 설치
    - asdf plugin add helm
    - asdf install helm 3.13.3
    - asdf global helm 3.13.3
  - ✓ 설치 확인: helm version

    version.BuildInfo{Version:"v3.13.3", GitCommit:"c8b948945e52abba22ff885446a1486cb5fd3474", GitTreeState:"clean", GoVersion:"go1.20.11"}

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart Repository
    - ● 자주 사용되는 Repositoy
      - ◆ stable: Helm 공식 stable 버전 Chart Repository로 현재 개발은 멈춘 상태
      - ◆ incubator: Helm 공식 stable 이전 버전 Chart Repository로 현재 개발은 멈춘 상태
      - ◆ bitnami: bitnami가 운영하는 서드 파티 Repository로 stable 보다 개발과 유지가 활발하며 공식 버전 보다 더 인기 있음
    - ● 원래 Helm의 공식 리포지터리인 stable과 incubator가 널리 사용되고 있었지만 현재는 bitnami의 리포지터리가 활발하게 사용되고 있으며 bitnami는 품질이 높은 Chart를 제공
    - ● 현재 Helm의 공식 문서의 퀵 스타트 가이드에서 stable 리포지터리를 사용하는 예를 확인할 수 있지만 많은 Chart가 deprecated 상태이므로 사용하지 않는 것이 좋음
    - ● bitnami Repository 추가: helm repo add bitnami https://charts.bitnami.com/bitnami
    - ● 참조할 수 있는 Chart Repository 리스트 확인: helm repo list

```
NAME        URL
bitnami     https://charts.bitnami.com/bitnami
```

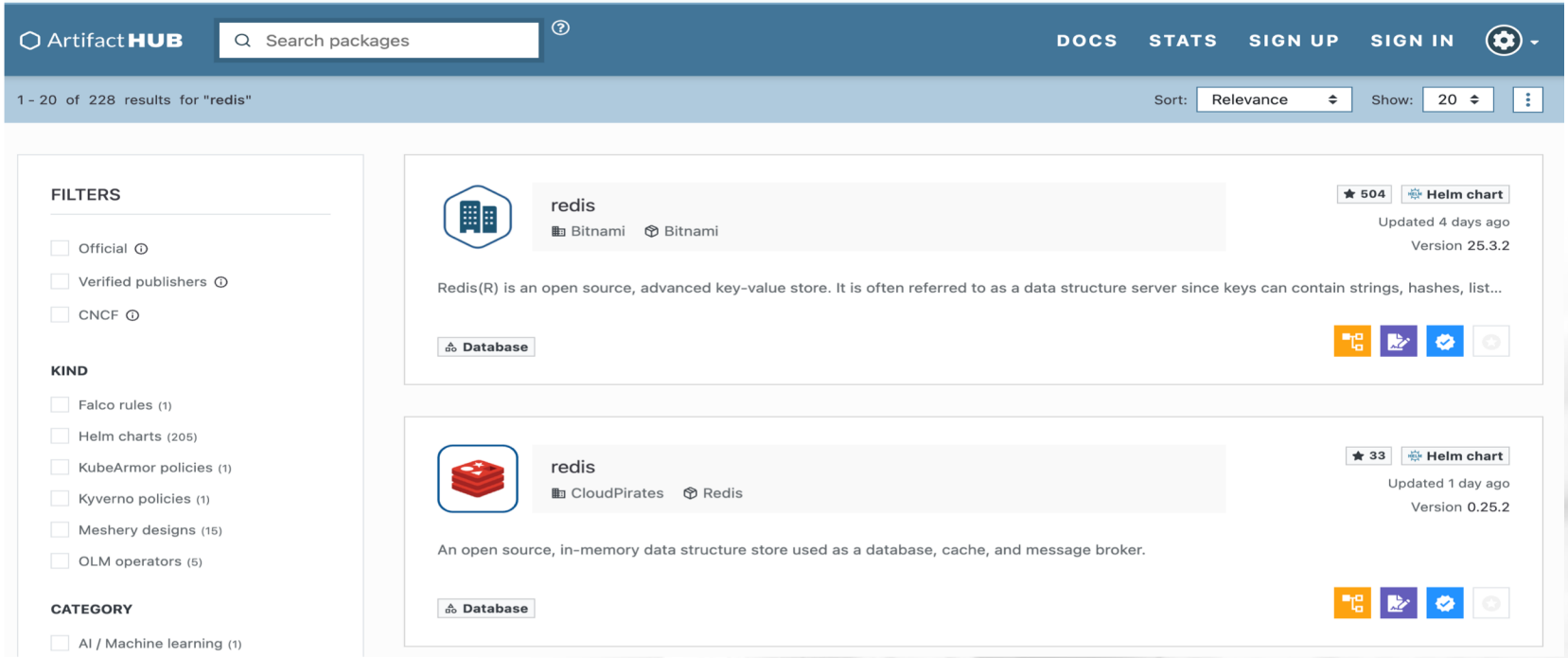
Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart Repository
    - ● 사용 가능한 Chart 검색: helm search repo redis

NAME	CHART VERSION	APP VERSION	DESCRIPTION
bitnami/redis	25.3.2	8.6.1	Redis(R) is an open source, advanced key-value ...
bitnami/redis-cluster	13.0.4	8.2.1	Redis(R) is an open source, scalable, distribut...
helm/redis	25.3.2	8.6.1	Redis(R) is an open source, advanced key-value ...
helm/redis-cluster	13.0.4	8.2.1	Redis(R) is an open source, scalable, distribut...
bitnami/keydb	0.5.22	6.3.4	KeyDB is a high performance fork of Redis with ...
helm/keydb	0.5.22	6.3.4	KeyDB is a high performance fork of Redis with ...

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart Repository
    - 헬름 허브에서 검색: helm search hub redis
    - 헬름 허브에서 검색: https://artifacthub.io/



Helm

- Helm Chart 와 Repository
  - ✓ Chart 구성

```
.
└── chart_name
    ├── Chart.yaml # Chart 정보 정의 파일
    ├── charts # 의존 관계의 Chart 디렉터리
    ├── templates # 매니페스트 파일 등 템플릿 디렉터리
    │   ├── xxxx.yaml # 각종 쿠버네티스 리소스의 매니페스트 템플릿
    │   ├── NOTES.txt # Chart 이용 방법 등 문서 템플릿
    │   └── _helpers.tpl # 매니페스트 생성에 사용되는 템플릿 헬퍼
    └── values.yaml # 템플릿 변수에 해당하는 기본값을 정의하는 파일
```

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● helm install 커맨드

      helm install [릴리스명] [Chart리포지터리명] [Chart명]

      --namespace [설치하는 위치의 네임스페이스]

      --create-namespace

      --version [Chart 버전]

      --values [커스텀 values 파일]

      - ◆ 릴리스명은 Helm에서 사용되는 이름이며 임의로 지정할 수 있는데 이는 쿠버네티스의 네임스페이스에서 고유성을 갖으며 릴리스명은 Chart 업데이트와 삭제 시 필요
      - ◆ --namespace 옵션은 Chart를 설치하는 쿠버네티스의 네임스페이스를 지정하는데 지정하지 않으면 default 네임스페이스를 사용
      - ◆ 기존의 네임스페이스를 오염시키지 않기 위해 전용 네임스페이스를 사용하는 경우가 많은데 대상 네임스페이스가 존재하지 않을 때는 --create-namespace 옵션을 부여하면 --namespace로 지정한 네임스페이스 생성
      - ◆ --version 옵션은 선택사항이지만 명시적으로 특정 버전을 지정
      - ◆ --values 옵션은 커스텀 values 파일의 경로를 지정

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● nginx 설치
      - ◆ 레포지토리 업데이트: helm repo update
      - ◆ Chart 검색: helm search repo bitnami/nginx -l

```
NAME                    CHART VERSION   APP VERSION     DESCRIPTION
bitnami/nginx           22.5.4          1.29.5          NGINX Open Source is a web server that can be a...
bitnami/nginx           22.5.3          1.29.5          NGINX Open Source is a web server that can be a...
bitnami/nginx           22.5.2          1.29.5          NGINX Open Source is a web server that can be a...
bitnami/nginx           22.5.1          1.29.5          NGINX Open Source is a web server that can be a...
bitnami/nginx           22.5.0          1.29.5          NGINX Open Source is a web server that can be a...
bitnami/nginx           22.4.7          1.29.5          NGINX Open Source is a web server tha
```

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● nginx 설치
      - ◆ 설정 가능한 파라미터 확인: helm show values bitnami/nginx

```
# @section Global parameters

## Global Docker image parameters

## Please, note that this will override the image parameters, including
   dependencies, configured to use the global value

## Current available global Docker image parameters: imageRegistry,
   imagePullSecrets and storageClass


## @param global.imageRegistry Global Docker image registry

## @param global.imagePullSecrets Global Docker registry secret names as
   an array

## @param global.defaultFips Default value for the FIPS configuration
   (allowed values: '', restricted, relaxed, off). Can be overriden by the 'fips'
   object

##
```

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - nginx 설치
      - ◆ 설치: helm install my-nginx bitnami/nginx
      - ◆ 설치 확인: helm ls

```
NAME      NAMESPACE          REVISION  UPDATED
          STATUS    CHART                APP VERSION
my-nginx  default    1         2026-03-03 07:11:43.922051827 +0000
    UTC   deployed  nginx-22.5.4         1.29.5
```

      - ◆ 배포 확인: kubectl get pods

```
NAME                        READY  STATUS   RESTARTS  AGE
my-nginx-6f758c6b75-vl87h   1/1    Running  0         62s
```

      - ◆ 서비스 확인: kubectl get services

```
NAME       TYPE          CLUSTER-IP     EXTERNAL-IP  PORT(S)
    AGE
my-nginx   LoadBalancer  10.97.140.39   10.0.2.50
    80:31366/TCP,443:30400/TCP   4m24s
```

      - ◆ 삭제: helm uninstall my-nginx

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ MariaDB Chart 확인: helm search repo bitnami/mariadb -l

```
NAME                    CHART VERSION   APP VERSION     DESCRIPTION
bitnami/mariadb         25.0.1          12.2.2          MariaDB is an open source, community-developed ...
bitnami/mariadb         25.0.0          12.2.2          MariaDB is an open source, community-developed ...
bitnami/mariadb         24.2.0          12.1.2          MariaDB is an open source, community-developed ...
```

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ 파라미터 확인: helm show values bitnami/mariadb

```
auth:
  ## @param auth.rootPassword Password for the `root` user. Ignored if
     existing secret is provided.
  ## ref:
     https://github.com/bitnami/containers/tree/main/bitnami/mariadb#settin
     g-the-root-password-on-first-run
  ##
  rootPassword: ""
  ## @param auth.database Name for a custom database to create
  ## ref:
     https://github.com/bitnami/containers/blob/main/bitnami/mariadb/READ
     ME.md#creating-a-database-on-first-run
  ##
  database: my_database
  ## @param auth.username Name for a custom user to create
  ## ref:
     https://github.com/bitnami/containers/blob/main/bitnami/mariadb/READ
     ME.md#creating-a-database-user-on-first-run
```

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ 기본 설치: helm install my-mariadb bitnami/mariadb
        - ▪ 사용법이 결과로 출력됨

NAME: my-mariadb
LAST DEPLOYED: Thu Mar 5 04:52:58 2026
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: mariadb
CHART VERSION: 25.0.1
APP VERSION: 12.2.2

WARNING: Since August 28th, 2025, only a limited subset of images/charts are available for free.
Subscribe to bitnami Secure Images to receive continued support and security updates.
More info at https://bitnami.com and https://github.com/bitnami/containers/issues/83267

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ 기본 설치: helm install my-mariadb bitnami/mariadb

```
** Please be patient while the chart is being deployed **

Tip:

  Watch the deployment status using the command: kubectl get pods -w --
    namespace default -l app.kubernetes.io/instance=my-mariadb

Services:

  echo Primary: my-mariadb.default.svc.cluster.local:3306

Administrator credentials:

  Username: root
  Password : $(kubectl get secret --namespace default my-mariadb -o
    jsonpath="{.data.mariadb-root-password}" | base64 -d)
```

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ 기본 설치: helm install my-mariadb bitnami/mariadb

To connect to your database, create a client pod:

```
kubectl run my-mariadb-client --rm --tty -i --restart='Never' --image
registry-1.docker.io/bitnami/mariadb:latest --namespace default --
command -- bash
```

To connect to primary service (read/write):

```
mariadb -h my-mariadb.default.svc.cluster.local -uroot -p my_database
```

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ 헬름 설치 확인: helm ls

```
NAME        NAMESPACE           REVISION  UPDATED
            STATUS    CHART                 APP VERSION
my-mariadb  default   1         2026-03-05 04:52:58.42842883
    +0000 UTC
            deployed  mariadb-25.0.1        12.2.2
```

      - ◆ 파드 확인: kubectl get pods

```
NAME           READY   STATUS    RESTARTS   AGE
my-mariadb-0   0/1     Pending   0          8m2s
```

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ 파드 자세히 확인: kubectl describe pods my-mariadb-0
        - ▪ 데이터베이스는 StatefulSet으로 생성되는데 PV 와 PVC 설정이 되지 않아서 파드가 만들어지지 않음

```
Events:
  Type     Reason            Age                From               Message
  ----     ------            ----               ----               -------
  Warning  FailedScheduling  26m                default-scheduler  0/4 nodes are available: pod has unbound immediate PersistentVolumeClaims. preemption: 0/4 nodes are available: 4 Preemption is not helpful for scheduling.
  Warning  FailedScheduling  11m (x3 over 21m)  default-scheduler  0/4 nodes are available: pod has unbound immediate PersistentVolumeClaims. preemption: 0/4 nodes are available: 4 Preemption is not helpful for scheduling.
```

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ 삭제: helm uninstall my-mariadb
      - ◆ 파라미터를 이용해서 스토리지를 사용하지 않는 형태로 생성: helm install my-mariadb bitnami/mariadb --set primary.persistence.enabled=false --set auth.rootPassword=wnddkd
      - ◆ 파드 확인: kubectl get pods

        NAME READY STATUS RESTARTS AGE

        my-mariadb-0 1/1 Running 0 65s
      - ◆ 컨테이너에 접속
        - $ kubectl exec -it my-mariadb-0 -- bash
        - I have no name! [/]$ mariadb -u root –p
      - ◆ 삭제: helm uninstall my-mariadb

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ PV 와 PVC 생성
        - ▪ 디렉토리 생성 및 권한 설정
          - sudo mkdir -p /mnt/data/mariadb
          - sudo chmod 777 /mnt/data/mariadb

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ PV 와 PVC 생성
        - ▪ 스토리지 설정 매니페스트: pv-pvc.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mariadb-pv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 3Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data/mariadb" # 실제 호스트의 경로 (폴더가 미리 생성되어 있어야 함)
```

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ PV 와 PVC 생성
        - ▪ 스토리지 설정 매니페스트: pv-pvc.yaml

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mariadb-pvc
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

        - ▪ 스토리지 적용: kubectl apply -f pv-pvc.yaml

Helm

- ❖ Helm Chart 와 Repository
  - ✓ Chart 설치
    - ● MariaDB 설치
      - ◆ 파라미터를 설정할 values.yaml 파일

```
auth:
  rootPassword: "your-root-password"
  username: "my-user"
  password: "my-password"
  database: "my-app-db"
primary:
  persistence:
    existingClaim: mariadb-pvc
```

      - ◆ 설치: helm install my-mariadb bitnami/mariadb -f values.yaml

Helm

- 자체 Chart 생성
  - ✓ Chart 모형 생성: helm create echo
  - ✓ Chart 디렉토리 확인: ls echo

```
echo
├── charts
├── Chart.yaml
├── templates
│   ├── deployment.yaml
│   ├── _helpers.tpl
│   ├── hpa.yaml
│   ├── ingress.yaml
│   ├── NOTES.txt
│   ├── serviceaccount.yaml
│   ├── service.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml
```

Helm

- ❖ 자체 Chart 생성
  - ✓ 불필요한 템플릿 삭제
    - 디렉토리 이동: cd echo/templates
    - 삭제: rm -rf hpa.yaml serviceaccount.yaml tests

Helm

- 자체 Chart 생성
  - ✓ 애플리케이션 배포를 위한 deployment.yaml 파일의 containers 부분 수정

```
containers:
- name: nginx
  image: "{{ .Values.nginx.image.repository }}:{{ .Values.nginx.image.tag }}"
  imagePullPolicy: {{ .Values.nginx.image.pullPolicy }}
  env:
  - name: NGINX_PORT
    value: "{{ .Values.nginx.port }}"
  - name: SERVER_NAME
    value: "{{ .Values.nginx.serverName }}"
  - name: BACKEND_HOST
    value: "{{ .Values.nginx.backendHost }}"
  - name: BACKEND_MAX_FAILS
    value: "{{ .Values.nginx.maxFails }}"
  - name: BACKEND_FAIL_TIMEOUT
    value: "{{ .Values.nginx.failTimeout }}"
  ports:
    - name: http
      containerPort: {{ .Values.nginx.port }}
- name: echo
  image: "{{ .Values.echo.image.repository }}:{{ .Values.echo.image.tag }}"
  imagePullPolicy: "{{ .Values.echo.image.pullPolicy }}"
```

Helm

- 자체 Chart 생성
  - ✓ deployment에 해당하는 values.yaml 수정

```
nginx:
  image:
    repository: ghcr.io/jpubdocker/simple-nginx-proxy
    tag: v0.1.0
    pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVersion.
  port: 80
  serverName: localhost
  backendHost: localhost:8080
  maxFails: 3
  failTimeout: 10s

echo:
  image:
    repository: ghcr.io/jpubdocker/echo
    tag: v0.1.0
    pullPolicy: IfNotPresent
  port: 8080
```

Helm

- 자체 Chart 생성
  - ✓ ingress에 해당하는 values.yaml 수정

```
ingress:
  enabled: true
  className: "nginx"
  annotations: {}
    # kubernetes.io/ingress.class: nginx
    # kubernetes.io/tls-acme: "true"
  hosts:
    - host: chart-example.local
      paths:
        - path: /
          pathType: Prefix
  tls: []
  #   - secretName: chart-example-tls
  #     hosts:
  #       - chart-example.local
```

Helm

- 자체 Chart 생성
  - ✓ serviceAccount 부분에 해당하는 values.yaml 수정

```
serviceAccount:
  # Specifies whether a service account should be created
  create: false
  # Automatically mount a ServiceAccount's API credentials?
  automount: true
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name: ""
```

Helm

- 자체 Chart 생성
  - ✓ helm template 커맨드로 출력한 매니페스트: helm$ helm template echo

```
---
# Source: echo/templates/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: release-name-echo
  labels:
    helm.sh/chart: echo-0.1.0
    app.kubernetes.io/name: echo
    app.kubernetes.io/instance: release-name
    app.kubernetes.io/version: "1.16.0"
    app.kubernetes.io/managed-by: Helm
spec:
  type: ClusterIP
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app.kubernetes.io/name: echo
    app.kubernetes.io/instance: release-name
```

Helm

- 자체 Chart 생성
  - ✓ Chart의 설정 파일 수정(echo/Chart.yaml)

```
apiVersion: v2
name: echo
description: A Helm chart for Kubernetes
type: application
version: 0.0.0
appVersion: "0.0.0"
```

Helm

- 자체 Chart 생성
  - ✓ Chart 패키지 생성: helm package echo

    Successfully packaged chart and saved it to: /home/adam/kubernetes/kubernetes-perfect-guide/samples/chapter14/helm/echo-0.0.0.tgz
  - ✓ Custom Values 파일 생성: helm/local-echo.yaml

```
ingress:
  hosts:
  - host: echo.jpub.local
    paths:
      - path: /
        pathType: Prefix
```

  - ✓ Chart 설치: helm install echo ./echo-0.0.0.tgz -f local-echo.yaml

```
NAME: echo
LAST DEPLOYED: Thu Mar  5 22:50:05 2026
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
1. Get the application URL by running these commands:
  http://echo.jpub.local/
```

Helm

- 자체 Chart 생성
  - ✓ 리소스 확인: kubectl get deployment,service,ingress -l app.kubernetes.io/name=echo

    Successfully packaged chart and saved it to: /home/adam/kubernetes/kubernetes-perfect-guide/samples/chapter14/helm/echo-0.0.0.tgz
  - ✓ EndPoint 확인: kubectl get endpoints

```
NAME    ENDPOINTS        AGE
echo    10.244.3.42:80   2m13s
```