

# 1. Helm 구성 및 사용

## 1.1 Helm 다운로드 및 설치

### 1.1.1 윈도우 설치

- Chocolatey 설치 : PowerShell 을 열어서 아래 명령을 수행 합니다. 이미 google cloud sdk 를 설치 하면서 설치 되었을 수 있습니다.

```
Set-ExecutionPolicy Bypass -Scope Process -Force;  
[System.Net.ServicePointManager]::SecurityProtocol =  
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object  
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

- Helm 설치

```
choco install kubernetes-helm
```

### 1.1.2 Mac 설치

- 수동 설치 방법

```
# helm 다운로드  
curl -fsSL -o get_helm.sh  
https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3  
  
# 실행권한 변경  
chmod 700 get_helm.sh  
  
# helm 설치  
./get_helm.sh  
  
# 버전 확인  
helm version  
  
# Helm Repository 추가  
helm repo add stable https://charts.helm.sh/stable  
  
# Repository 업데이트  
helm repo update
```

- Brew 설치

```
brew install helm
```

## 1.2 Mysql Helm 차트 다운로드 및 설치

### 1.3.1 mysql helm 검색

```
helm search repo stable/mysql
```

NAME	CHART VERSION	APP VERSION	DESCRIPTION
stable/mysql	1.6.3	5.7.28	Fast, reliable, scalable, and easy to use open-...
stable/mysqldump	2.6.0	2.4.1	A Helm chart to help backup MySQL databases usi...

### 1.3.2 패키지 메타 정보 보기

```
helm show chart stable/mysql
```

```
apiVersion: v1
appVersion: 5.7.28
description: Fast, reliable, scalable, and easy to use open-source relational database
  system.
home: https://www.mysql.com/
icon: https://www.mysql.com/common/logos/logo-mysql-170x115.png
keywords:
- mysql
- database
- sql
maintainers:
- email: o.with@sportradar.com
  name: olemarkus
- email: viglesias@google.com
  name: viglesiasce
name: mysql
sources:
- https://github.com/kubernetes/charts
- https://github.com/docker-library/mysql
version: 1.6.3
```

### 1.3.3 mysql helm 차트 설치 및 Deployment

```
helm install stable/mysql --generate-name
```

```
AME: mysql-1588321002
LAST DEPLOYED: Fri May 1 08:16:55 2020
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
```

MySQL can be accessed via port 3306 on the following DNS name from within your cluster:  
mysql-1588321002.default.svc.cluster.local

To `get` your root password run:

```
MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace default mysql-1588321701 -o  
jsonpath="{.data.mysql-root-password}" | base64 --decode; echo)
```

To connect to your database:

1. Run an Ubuntu pod that you can use as a client:

```
kubectl run -i --tty ubuntu --image=ubuntu:16.04 --restart=Never -- bash -il
```

2. Install the mysql client:

```
$ apt-get update && apt-get install mysql-client -y
```

3. Connect using the mysql cli, `then` provide your password:

```
$ mysql -h mysql-1588321701 -p
```

To connect to your database directly from outside the K8s cluster:

```
MYSQL_HOST=127.0.0.1
```

```
MYSQL_PORT=3306
```

```
# Execute the following command to route the connection:
```

```
kubectl port-forward svc/mysql-1588321002 3306
```

```
mysql -h ${MYSQL_HOST} -P${MYSQL_PORT} -u root -p${MYSQL_ROOT_PASSWORD}
```

```
helm ls
```

NAME	STATUS	NAMESPACE	REVISION	UPDATED
HART		C		
	APP VERSION			
mysql-1588321701	default	1	2020-05-01 17:28:25.322363879	
+0900 +09	deployed	m		
ysql-1.6.3	5.7.28			

## 1.3.4 helm 차트 uninstall

```
helm list
```

NAME	STATUS	NAMESPACE	REVISION	UPDATED
HART	C	APP VERSION		
mysql-1588321701	default	1	2020-05-01 17:28:25.322363879	
+0900 +09	deployed	m		
ysql-1.6.3	5.7.28			

```
helm uninstall mysql-1588321701
release "mysql-1588321701" uninstalled
```

## 1.3 Helm 차트 만들기

### 1.3.1 Helm 차트 생성

```
helm create nginxstd
```

### 1.3.2 Template 파일 수정

- Charts.yaml 파일 수정

```
apiVersion: v2
name: nginx-std
description: A Helm chart for Kubernetes
type: application
version: 0.1.0
appVersion: "1.16.0"
```

- Template/deployment.yaml 파일 생성

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Values.container.name }}
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: {{ .Values.container.name }}
  template:
    metadata:
      labels:
        app: {{ .Values.container.name }}
        environment: {{ .Values.environment }}
    spec:
```

```

containers:
  - name: {{ .Values.container.name }}
    image: {{ .Values.container.image }}:{{ .Values.container.tag }}
    ports:
      - containerPort: {{ .Values.container.port }}
    env:
      - name: environment
        value: {{ .Values.environment }}

```

- template/service.yaml 파일 생성

```

apiVersion: v1
kind: Service
metadata:
  name: {{ .Values.container.name }}-service
  labels:
    app: {{ .Values.container.name }}
spec:
  ports:
    - port: 80
      protocol: TCP
      targetPort: {{ .Values.container.port }}
  selector:
    app: {{ .Values.container.name }}
  type: LoadBalancer

```

- values.yaml 파일 생성

```

environment: development
container:
  name: nginx
  port: 80
  image: nginx
  tag: latest
replicas: 2

```

### 1.3.3 테스트 하기

- K8s 오브젝트 생성

```
helm install nginxstd ./nginxstd
```

- 삭제

```
# 확인
kubectl get all
helm list

# 삭제
helm uninstall nginxstd
```

## 1.4 패키지 및 리포지토리 생성

---

### 1.4.1 패키지 생성

```
helm package ./nginxstd

mkdir prod
mv ./nginx-std-0.1.0.tgz ./prod/
```

### 1.4.2 helm 리포지토리 파일 생성

```
# 리포지토리 파일 생성 (index.yaml)
helm repo index ./prod

# 파일 생성 확인
cat ./prod/index.yaml
```

## 1.5 Helm 패키지 및 Repository 구성하기

---

### 1.5.1 Github.com Repository 생성


- repository 생성

# Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

---


**Owner \*** **Repository name \***


 dangtong76 ▾ / **helm-prod**

Great repository names are short and memorable. Need inspiration? How about **didactic-guide?**

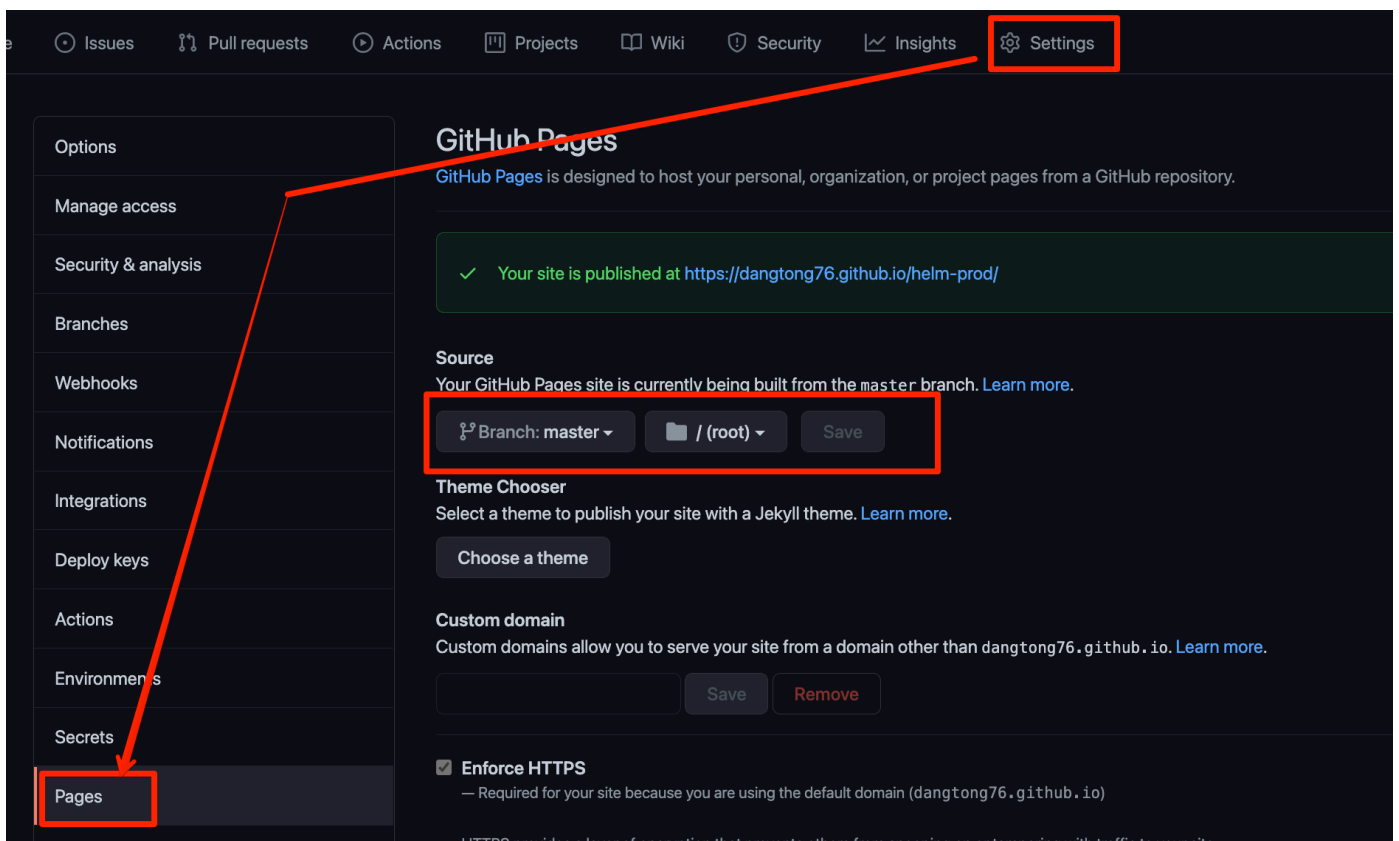
**Description (optional)**

---

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

- github page 설정



The screenshot shows the GitHub Settings page for a repository named 'helm-prod' owned by 'dangtong76'. The 'Settings' tab is selected in the top navigation bar. On the left sidebar, the 'Pages' option is highlighted with a red box. A red arrow points from this 'Pages' option to the 'Source' section of the GitHub Pages settings. In the 'Source' section, the 'Branch' dropdown is set to 'master' and the 'Folder' dropdown is set to '/ (root)'. Both dropdowns and the 'Save' button are enclosed in a red box. Other sections visible include 'Theme Chooser' (set to Jekyll), 'Custom domain', and 'Enforce HTTPS' (checked).

## 1.5.2 Git repository 생성 및 동기화

```
cd prod

git init

git add .

git commit -a -m "initial commit"

git remote add origin https://github.com/dangtong76/helm-prod.git

git push origin master
```

### 1.5.3 Helm 리포지토리 구성 및 추가

- Git page 로 서비스 되는 Git 리포지토리를 Helm 리포지토리에 추가

```
helm repo add helm-prod https://dangtong76.github.io/helm-prod
```

- 추가확인

```
helm repo list

helm search repo nginx
```

### 1.5.4 Helm 리포지토리에 redis 추가

- redis 안정버전 차트를 로컬 prod 디렉토리에 다운로드

```
helm search repo redis

helm fetch stable/redis -d ./prod
```

- index.yaml 갱신

```
helm repo index ./prod
```

- git 업데이트

```
git status

git add .

git commit -a -m "add redis"

git push origin master
```



- helm update 수행

```
helm repo update
```

```
helm search repo redis
```

업데이트 없이 "helm search repo redis" 를 검색하면 검색이 되지 않습니다.

## 1.6 Helm 차트 업그레이드

### 1.6.1 Repository 를 통한 Helm 인스톨

```
helm list
```

```
helm install nginxstd helm-prod/nginx-std
```

# 또는

```
helm install helm-prod/nginx-std --generate-name
```

#확인

```
helm status nginxstd
```

```
kubectl get all
```

### 1.6.2 helm 메니페스트를 통한 차트 변경 및 업데이트

- stage-values.yaml 파일 생성

```
environment: development
```

```
replicas: 4
```

- helm upgrade 로 차트 변경 적용

```
helm upgrade -f ./nginxstd/stage-values.yaml nginxstd helm-prod/nginx-std
```

- helm history 로 확인

```
helm history
```

- RollBack 수행

```
helm rollback nginxstd 1
```

- Rollback 확인

```
helm history nginxstd

helm helm status nginxstd

kubectl get po
```

## 1.6.4 Helm CLI 옵션을 통한 업그레이드

- 현재 차트의 value 를 확인

```
helm show values helm-prod/nginx-std

environment: development
container:
  name: nginx
  port: 80
  image: nginx:1.7.9
  tag: hello
replicas: 2
```

- CLI 옵션을 통한 업그레이드

```
helm upgrade --set replicas=4 --set environment=dev nginxstd helm-prod/nginx-std
```

- 확인

```
helm history

helm status nginxstd

kubectl get po
```

## 1.7 삭제

```
helm uninstall nginxstd
```

## 1.8 연습문제

1. helm-myrepo 라는 디렉토리를 만드세요
  1. helm create nginxstd
  2. helm package ./nginxstd
  3. helm repo index ./prod
  4. helm repo add helm-prod <https://dangtong76.github.io/helm-prod>

5. helm repo list
2. nodes-sts 라는 이름으로 helm 차트를 생성 하세요
3. 아래 아래 3개의 yaml 파일에 대해 helm 차트를 구성하고 려고 합니다.  
파일명 : sts-lb.yaml / sts-sc.yaml / sts.yaml
4. 변수화 할 항목은 아래와 같습니다.
  1. sts.yaml 에서 변수처리
    1. Spec.template.spec.containers.name
    2. Spec.template.spec.containers.image
    3. Spec.template.spec.containers.ports.containerPort
    4. Spec.template.spec.containers.volumeMounts.name
    5. Spec.template.spec.containers.volumeMounts.mountPath
    6. Spec.volumeClaimTemplates.metadata.name
    7. Spec.volumeClaimTemplates.spec.resources.requests.storage
    8. Spec.volumeClaimTemplates.spec.accessModes
    9. Spec.volumeClaimTemplates.spec.storageClassName
  2. sts-lb.yaml 에서 변수처리
    1. metadata.name
    2. Spec.ports.port
    3. Spec.ports.targetPort
  3. sts-sc.yaml 에서 변수처리
    1. reclaimPolicy
    2. Parameters.type
    3. Parameters.zone
5. Github.com 에 helm-myrepo 라는 이름으로 리포지토리를 생성하세요
6. nodes-sts 라는 helm 패키지를 생성하고 helm-myrepo 내에 패키지를 복사하세요
7. helm 리포지토리를 생성 하세요 (index.yaml 생성)
8. helm-myrepo 내에 git repository 를 초기화 하고 Commit 후에 GitHub.com 관 연동 하세요
9. github.com 의 helm-myrepo 리포지토리를 helm 리포지토리에 추가하세요
10. nodes-sts helm 설치 차트를 설치하세요.

## 1.8.1 3 3개 파일

- sts.yaml

```
apiVersion: apps/v1
kind: StatefulSet
```

```

metadata:
  name: nodejs-sfs
spec:
  selector:
    matchLabels:
      app: nodejs-sfs
  serviceName: nodejs-sfs
  replicas: 2
  template:
    metadata:
      labels:
        app: nodejs-sfs
    spec:
      containers:
        - name: nodejs
          image: dangtong76/nodejs
          ports:
            - name: http
              containerPort: 8080
          volumeMounts:
            - name: data
              mountPath: /var/data
  volumeClaimTemplates:
    - metadata:
        name: data
      spec:
        resources:
          requests:
            storage: 1Mi
        accessModes:
          - ReadWriteOnce
        storageClassName: sc-standard-retain

```

- sts-sc.yaml

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-standard-retain
provisioner: kubernetes.io/gce-pd
reclaimPolicy: Retain
parameters:
  type: pd-ssd
  zone: asia-northeast3-a

```

- sts-lb.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nodesjs-sfs-lb
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: nodejs-sfs
```

## 2. Istio 설치 및 활용

### 2.1 설치전 플랫폼별 설정

참고 URL : <https://istio.io/latest/docs/setup/platform-setup/>

위 링크에 들어가면 플랫폼별 Istio 사전 설치 셋업 관련 정보를 확인 할 수 있습니다.

#### 2.1.1 kubectl 클라이언트 Credential 설정

```
gcloud container clusters get-credentials $CLUSTER_NAME \
  --zone $ZONE \
  --project $PROJECT_ID
```

#### 2.1.2 클러스터 롤 설정

```
kubectl create clusterrolebinding cluster-admin-binding \
  --clusterrole=cluster-admin \
  --user=$(gcloud config get-value core/account)
```

#### 2.1.3 타사 토큰 지원 확인

```
# 리눅스 및 맥
brew install jq
kubectl get --raw /api/v1 | jq '.resources[] | select(.name |
index("serviceaccounts/token"))'
```

```
## 윈도우
choco install jq
kubectl get --raw /api/v1 | jq '.resources[] | select(.name |
index(\"serviceaccounts/token\"))'

# 아래와 같이 결과가 나와야함
{
  "name": "serviceaccounts/token",
  "singularName": "",
  "namespaced": true,
  "group": "authentication.k8s.io",
  "version": "v1",
  "kind": "TokenRequest",
  "verbs": [
    "create"
  ]
}
```

## 2.2 Istio 환경 변수 설정 및 다운로드

### 2.2.1 Istio 다운로드

- Mac and Linux

```
curl -L https://istio.io/downloadIstio | sh -
```

```
curl -L https://istio.io/downloadIstio | ISTIO_VERSION=1.6.8 TARGET_ARCH=x86_64 sh -
```

- Windows

<https://github.com/istio/istio/releases> 에서 다운로드 받아서 압축 해지

[istio-1.14.1-win.zip](#)

### 2.2.2 설치전 환경변수 설정

```
cd istio-1.10.3
export PATH=$PWD/bin:$PATH 또는
brew install istioctl
choco install istioctl
```

## 2.3 Istio 설치 with Helm

### 2.3.1 Helm Repo 추가 및 라벨추가

- helm repo 추가

```
helm repo add istio https://istio-release.storage.googleapis.com/charts
helm repo update
```

- Namespace 생성 및 istio-injection 라벨 추가

```
kubectl create namespace istio-system
kubectl label namespace default istio-injection=enabled
```

### 2.3.2 Base Chart 를 통한 Istio Control Plane 에 설치될 컨테이너 설치

```
helm install istio-base manifests/charts/base -n istio-system

helm install istio-base istio/base -n istio-system
```

### 2.3.3 Discovery Chart 를 통한 Istiod 설치

```
helm install istiod manifests/charts/istio-control/istio-discovery -n istio-system
```

### 2.3.4 Istio Ingress Gateway 설치(옵션)

```
helm install istio-ingress manifests/charts/gateways/istio-ingress -n istio-system
```

### 2.3.5 Istio Egress Gateway (옵션)

```
helm install istio-egress manifests/charts/gateways/istio-egress -n istio-system
```

### 2.3.6 설치 확인

```
kubectl get pods -n istio-system
```

## 2.4 Prometheus / Jeager / Grapana / Kiali 설치

## 2.4.1 Prometheus / Jeager / Grapna설치

```
cd <istion_install_dir>
cd samples/addones

kubectl apply -f samples/addons/prometheus.yaml
kubectl apply -f samples/addons/jaeger.yaml
kubectl apply -f samples/addons/grafana.yaml
```

## 2.4.2 Kiali 설치

```
kubectl apply -f samples/addons/kiali.yaml
```

## 2.4.2 서비스 설치 확인

```
kubectl get all -n istio-system
```

## 2.4.3 서비스 접속

```
kubectl port-forward service/kiali 20001:20001 -n istio-system
```

브라우저에 <http://localhost:20001> 로 접속

## 2.5 Sample Application 설치

### 2.5.1 설치

istion. 설치 바이너리 디렉토리에서 Samples 디렉토리 밑에 테스트 애플리케이션이 있다.

```
cd {istio_binary_dir}
kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml
kubectl apply -f samples/bookinfo/networking/bookinfo-gateway.yaml
```

### 2.5.2 서비스 확인

```
kubectl exec "$(kubectl get pod -l app=ratings -o
jsonpath='{.items[0].metadata.name}')" -c ratings -- curl -sS
productpage:9080/productpage | grep -o "<title>.*</title>"
<title>Simple Bookstore App</title>
```



## 2.6 Circuit Break 테스트

### 2.6.1 커넥션 개수에 따른 Circuit Break 테스트

- app.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: httpbin
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpbin
  template:
    metadata:
      labels:
        app: httpbin
    spec:
      containers:
        - name: httpbin
          image: dangtong/httpbin:latest
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  labels:
    app: httpbin
spec:
  selector:
    app: httpbin
  ports:
    - name: http
      port: 8000
      targetPort: 80
```

- fortio.yaml

```
apiVersion: v1
kind: Pod
metadata:
```

```

name: fortio
labels:
  app: fortio
spec:
  containers:
  - image: docker.io/fortio/fortio:latest_release
    imagePullPolicy: IfNotPresent
    name: fortio
    ports:
    - containerPort: 8080
      name: http-fortio
    - containerPort: 80

```

- fortio 클라이언트 테스트

```

kubectl exec -it fortio -c fortio -- /usr/bin/fortio load -curl
http://httpbin.default:8000/get

```

- 1개 커넥션 테스트

```

kubectl exec -it fortio -c fortio -- /usr/bin/fortio load -c 1 -qps 0 -n 10 -loglevel
Warning http://httpbin:8000/get

```

- Destination Rule 추가 하기

```

apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: dr-httpbin
spec:
  host: httpbin
  trafficPolicy:
    connectionPool:
      http:
        http1MaxPendingRequests: 1
        maxRequestsPerConnection: 1

```

- 커넥션 개수 늘려서 테스트 하기

```
kubectl exec -it fortio -c fortio -- /usr/bin/fortio load -c 10 -qps 0 -n 10 -loglevel
Warning http://httpbin:8000/get
```

## 2.6.2 서비스에 속한 POD 상태에 따른 Circuit Break 테스트

### 2.6.2.1 서비스 환경

- hello-server 는 환경변수 RANDOM\_ERROR 값의 확률로 랜덤하게 503 에러를 발생

### 2.6.2.2 서비스 생성

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-server-1
  labels:
    app: hello
spec:
  containers:
  - name: hello-server-1
    image: dangtong/hello-server:latest
    imagePullPolicy: IfNotPresent
    env:
      - name: VERSION
        value: "v1"
      - name: LOG
        value: "1"
---
apiVersion: v1
kind: Pod
metadata:
  name: hello-server-2
  labels:
    app: hello
spec:
  containers:
  - name: hello-server-2
    image: dangtong/hello-server:latest
    imagePullPolicy: IfNotPresent
    env:
      - name: VERSION
        value: "v2"
      - name: LOG
        value: "1"
      - name: RANDOM_ERROR
        value: "0.2"
---
```

```
apiVersion: v1
kind: Service
metadata:
  name: svc-hello
  labels:
    app: hello
spec:
  selector:
    app: hello
  ports:
  - name: http
    protocol: TCP
    port: 8080
```

### 2.6.2.3 httpbin POD 생성

```
apiVersion: v1
kind: Pod
metadata:
  name: httpbin
  labels:
    app: httpbin
spec:
  containers:
  - name: httpbin
    image: dangtong/httpbin:latest
    imagePullPolicy: IfNotPresent
```

### 2.6.2.4 테스트 1

- 총 20번의 요청을 2개의 POD에 10번씩 요청

```
for i in {1..10}; do kubectl exec -it httpbin -c httpbin -- curl http://svc-hello.default:8080; sleep 0.1; done
```

- 로그 확인

```
kubectl logs hello-server-2 -c hello-server-2
```

## 2.6.2.5 Destination Rule 적용후 테스트

- Destination Rule 적용

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: dr-hello
spec:
  host: svc-hello
  trafficPolicy:
    outlierDetection:
      interval: 1s # 1초마다 확인해서
      consecutive5xxErrors: 1 # 5XX 에러가 발생하면
      baseEjectionTime: 3m # 3분동안 로드밸런싱 풀에서 제외 함
      maxEjectionPercent: 100 # Default 10%, 최대 Eject 퍼센티지이며 100% 이면 서비스가 바라보는
      모든 로드밸런싱 풀이 제외 될 수 있음.
```

- 테스트

```
for i in {1..20}; do kubectl exec -it httpbin -c httpbin -- curl http://svc-hello.default:8080; sleep 0.1; done
```

- Kiali 대시보드 확인

## 2.6 Istio 및 서비스 삭제

### 2.6.1 서비스 삭제

```
kubectl delete -f samples/addons/kiali.yaml
kubectl delete -f samples/addons/grafana.yaml
kubectl delete -f samples/addons/jaeger.yaml
kubectl delete -f samples/addons/prometheus.yaml
```

### 2.6.2 Istio 삭제

```
helm delete istio-egress -n istio-system
helm delete istio-ingress -n istio-system

# delete Istio discovery chart
helm delete istiod -n istio-system

# delete Istion base chart
```

```
helm delete istio-base -n istio-system

# delete Istion istio-system namespace
kubectl delete namespace istio-system

# delete CustomResourceDefinitions(CRD)
kubectl get crd | grep --color=never 'istio.io' | awk '{print $1}' \
    | xargs -n1 kubectl delete crd

# delete label
kubectl label namespace default istio-injection-
```

## 3. Istio 연동

---

## 3. 로컬 k8s 클러스터 설치

---

### 3.1 Vagrant 설치

---

<https://www.vagrantup.com/downloads> 에서 운영체제 버전에 맞는 Vagrant 를 설치 합니다.

윈도우의 경우 hyper-v 활성화를 아래 명령을 통해 해야 합니다.

```
Disable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V-All
```

설치 사전요구사항으로 아래 소프트웨어 설치 되어야함

- Vagrant (<https://www.vagrantup.com/>)
- VirtualBox (<https://www.virtualbox.org/>)
- kubectl (<https://kubernetes.io/docs/tasks/tools/install-kubectl/>)
- git (<https://kubernetes.io/docs/tasks/tools/install-kubectl/>)