



초보를 위한 쿠버네티스 안내서

02 쿠버네티스 실습 준비

- YAML 문법
 - [기본문법](#)
 - [주의사항](#)
 - [참고](#)
- 쿠버네티스 설치 (windows)
- kubectl 설치 (windows)
- k3s 설치 (windows)

03 쿠버네티스 기본 실습

- 시작하기
 - [워드프레스 배포](#)
- 기본 명령어
 - [상태 설정하기 \(apply\)](#)
 - [리소스 목록보기 \(get\)](#)
 - [리소스 상세 상태보기 \(describe\)](#)
 - [리소스 제거 \(delete\)](#)
 - [컨테이너 로그 조회 \(logs\)](#)
 - [컨테이너 명령어 전달 \(exec\)](#)
 - [설정 관리 \(config\) & 기타](#)
- Pod
 - [빠르게 Pod 만들기](#)
 - [Pod 생성 분석](#)
 - [YAML로 설정파일Spec 작성하기](#)
 - [컨테이너 상태 모니터링](#)
 - [다중 컨테이너](#)
 -  [문제 오답노트](#)
- ReplicaSet
 - [ReplicaSet 만들기](#)
 - [스케일 아웃](#)
 -  [문제 오답노트](#)
- Deployment
 - [deployment 만들기](#)
 - [버전관리](#)
 - [배포 전략 설정](#)
- Service
 - [Service \(ClusterIP\) 만들기](#)
 - [Service 생성 흐름](#)
 - [Service \(NodePort\) 만들기](#)
 - [Service\(LoadBalancer\) 만들기](#)
 - [minikube에 가상 LoadBalancer 만들기](#)
- [중간평가] 웹 애플리케이션 배포
- Ingress
 - [Ingress 만들기](#)
 - [minikube에 Ingress 활성화하기](#)
 - [echo 웹 애플리케이션 배포](#)
 - [Ingress 생성 흐름](#)
- Volume (local)
 - [Volume 만들기](#)
- ConfigMap
 - [ConfigMap 만들기](#)
 - [env 파일로 만들기](#)
 - [YAML 선언하기](#)
 - [ConfigMap을 환경변수로 사용하기](#)
- Secret

02 쿠버네티스 실습 준비

•YAML 문법

쿠버네티스에게 우리가 “블로그를 만들 건데, APP서버 3개, DB 서버는 1대, ...” 요청을 한다.

⇒ 쿠버네티스가 못알아 듣는다.

YAML로 요청을 한다. ⇒ 쿠버네티스가 알아듣고 배포를 한다!

기본문법

들여쓰기 (indent)

들여쓰기는 기본적으로 2칸(추천) or 4칸 지원

데이터 정의 (map)

데이터는 key : value 형식으로 정의한다.

```
apiVersion: v1
kind: Pod
metadata:
  name: echo
  labels:
    type: app
```

배열 정의 (array)

배열은 -로 표시한다.

```
person:
  name: Chungsub Kim
  job: Developer
  skills:
    - docker
    - kubernetes
```

주석 (comment)

주석은 #으로 표시한다.

참/거짓, 숫자표현

- 참/거짓은 true, false 외에 yes, no를 지원한다.
- 숫자는 정수 또는 실수를 따옴표(") 없이 사용하면 숫자로 인식한다.

줄바꿈 (newline)

여러 줄을 표현하는 방법이다.

- “|” 지시어는 마지막 줄바꿈이 포함
- “-” 지시어는 마지막 줄바꿈을 제외
- “>” 지시어는 중간에 들어간 빈줄을 제외

주의사항

- 띄어쓰기 : key와 value 사이에는 반드시 빈칸이 필요함
- 문자열 따옴표 : 대부분의 문자열을 따옴표 없이 사용할 수 있지만 “.”가 들어간 경우는 반드시 따옴표가 필요함.

참고

- **json2yaml** : 변환 사이트
- **yamllint** : valid한 yaml인지 판단해주는 사이트

▪쿠버네티스 설치 (windows)

minikube-installer.exe (opens new window)파일을 다운로드

Windows 10 Home에서는 hyperv을 지원하지 않아 virtual box에서 진행

```
minikube start --driver=docker
minikube status
minikube ip
minikube delete
minikube start --kubernetes-version=v1.23.1
minikube status
```

▼ 기본 명령어

```
# 버전확인
minikube version

# 가상머신 시작 (반드시 관리자권한으로 실행)
minikube start --driver=hyperv
# driver 예러가 발생한다면 virtual box를 사용
minikube start --driver=virtualbox
# 특정 k8s 버전 실행
minikube start --kubernetes-version=v1.23.1

# 상태확인
minikube status

# 중지
minikube stop

# 삭제
minikube delete

# ssh 접속
minikube ssh

# ip 확인
minikube ip
```

▪kubectl 설치 (windows)

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.23.5/bin/windows/amd64/kubectl.exe
kubectl version
```

```
apiServer: Running
kubeconfig: Configured

[leeoo@DESKTOP-HCDBS1U ~]$ minikube ip
192.168.49.2

[leeoo@DESKTOP-HCDBS1U ~]$ minikube delete
* docker -- "minikube"를 삭제하는 중 ...
* Deleting container "minikube" ...
* C:\Users\leeoo\minikube\machines\minikube 제거 중 ...
* "minikube" 클러스터 관련 정보가 모두 삭제되었습니다
[leeoo@DESKTOP-HCDBS1U ~]$ minikube start --kubernetes-version=v1.23.1
* Microsoft Windows [Version 10.0.19045.3324] (c) 2024 Microsoft Corporation. All rights reserved.
* 자습서로 docker 드라이버가 선택되었습니다. 다른 드라이버 목록: virtualbox, ssh
* Using Docker Desktop driver with root privileges
* minikube 클러스터의 minikube 컨트롤러 노드를 시작하는 중
* 백업스 이미지를 다운로드 중 ...
* 주버전인 v1.23.1를 다운로드 중 ...
* > preloaded-images-k8s-v18-v1... 399.23 MiB / 399.23 MiB 100.00% 3.17 Mi
* Creating docker container (CPU=2, Memory=2200MB) ...
* 주버전인 v1.23.1을 Docker 24.0.4 컨테이너로 설치하는 중
* 캐시된 이미지를 로드할 수 없습니다: loading cached images: CreateFile C:\Users\leeoo\minikube\cache\images\amd64\registry.k8s.io\etcd\3.5.1-0: The system cannot find the path specified.
* 컨테이너 실행을 시작하는 중 ...
* RBAC 규칙을 구성하는 중 ...
* Kubernetes 구성 요소를 확인 ...
* Using image gcr.io/k8s-minikube/storage-provisioner:v5
* 하드웨어 설정: storage-provisioner, default-storageclass

[leeoo@DESKTOP-HCDBS1U ~]$ kubectl version
Client Version: v1.27.2
Server Version: v1.27.2

[leeoo@DESKTOP-HCDBS1U ~]$ kubectl get nodes
NAME                 STATUS   ROLES           AGE   VERSION
minikube              Ready    ControlPlane    1m    v1.27.2
```

•k3s 설치 (windows)

AWS → lightsail → Linux/Unix & Ubuntu → add Launch script

add Launch script

```
sed -i 's/PasswordAuthentication no/PasswordAuthentication yes/' /etc/ssh/sshd_config
echo "ubuntu:1q2w3e4r!!" | chpasswd
service sshd reload
curl -sL https://deb.nodesource.com/setup_14.x | bash -
apt-get -y update
DEBIAN_FRONTEND=noninteractive apt-get -y install nodejs build-essential
npm install -g wetty --unsafe
ln -s /usr/bin/wetty /usr/local/bin/wetty
curl https://gist.githubusercontent.com/subicura/9058671c16e2abd36533fea2798886b0/raw/e5d249612711b14c9c8f44798dea1368395e86a9/wetty.s
systemctl start wetty
systemctl enable wetty
```

IPv4 Firewall

Create rules to open ports to the internet, or to a specific IPv4 address or range.

[Learn more about firewall rules](#) 

 Add rule

Application	Protocol	Port or range / Code	Restricted to	
Custom	TCP	3000 → 65000	Any IPv4 address	 

PublicIP:4200 (아이디/패스워드 - ubuntu/1q2w3e4r!!)

설치

```
curl -sL https://get.k3s.io | sh -
sudo chown ubuntu:ubuntu /etc/rancher/k3s/k3s.yaml

# 확인
kubectl get nodes
```

```
# 설정 복사
cp /etc/rancher/k3s/k3s.yaml ~/.kube/config
```

03 쿠버네티스 기본 실습

■ 시작하기

워드프레스 배포

웹 브라우저 → 워드프레스 → MySQL

guide/index/docker-compose.yml

```
version: "3"

services:
  wordpress:
    image: wordpress:5.9.1-php8.1-apache
    environment:
      WORDPRESS_DB_HOST: mysql
      WORDPRESS_DB_NAME: wordpress
      WORDPRESS_DB_USER: root
      WORDPRESS_DB_PASSWORD: password
    ports:
      - "30000:80"

  mysql:
    image: mariadb:10.7
    environment:
      MYSQL_DATABASE: wordpress
      MYSQL_ROOT_PASSWORD: password
```

워드프레스 컨테이너 하나, MySQL 컨테이너 하나, 그리고 각각 포트와 환경변수 설정

```
docker-compose up -d
```

```
docker-compose down
```

이제 쿠버네티스로 배포를 해보자!

guide/index/wordpress-k8s.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: mysql
  template:
    metadata:
      labels:
        app: wordpress
        tier: mysql
    spec:
      containers:
        - image: mariadb:10.7
          name: mysql
          env:
            - name: MYSQL_DATABASE
              value: wordpress
            - name: MYSQL_ROOT_PASSWORD
              value: password
```

```

      ports:
        - containerPort: 3306
          name: mysql
    ---
  apiVersion: v1
  kind: Service
  metadata:
    name: wordpress-mysql
    labels:
      app: wordpress
  spec:
    ports:
      - port: 3306
    selector:
      app: wordpress
      tier: mysql
    ---
  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: wordpress
    labels:
      app: wordpress
  spec:
    selector:
      matchLabels:
        app: wordpress
        tier: frontend
    template:
      metadata:
        labels:
          app: wordpress
          tier: frontend
      spec:
        containers:
          - image: wordpress:5.9.1-php8.1-apache
            name: wordpress
            env:
              - name: WORDPRESS_DB_HOST
                value: wordpress-mysql
              - name: WORDPRESS_DB_NAME
                value: wordpress
              - name: WORDPRESS_DB_USER
                value: root
              - name: WORDPRESS_DB_PASSWORD
                value: password
            ports:
              - containerPort: 80
                name: wordpress
    ---
  apiVersion: v1
  kind: Service
  metadata:
    name: wordpress
    labels:
      app: wordpress
  spec:
    type: NodePort
    ports:
      - port: 80
    selector:
      app: wordpress
      tier: frontend

```

```
kubectl apply -f wordpress-k8s.yml
```

```
kubectl get all
```

```
kubectl delete -f wordpress-k8s.yml
```

아까 docker-compose랑 똑같아 보이는데 굳이 쿠버네티스 써야하는 이유?

컨테이너를 죽여도 다시 생성시켜준다. 그리고 수를 늘릴수도 있다. replicas이용. 하지만 도커로만은 이를 구현하기 쉽지 않음

기본 명령어

GUI 도구 > Lens

명령어	설명
<code>apply</code>	원하는 상태를 적용합니다. 보통 <code>-f</code> 옵션으로 파일과 함께 사용합니다.
<code>get</code>	리소스 목록을 보여줍니다.
<code>describe</code>	리소스의 상태를 자세하게 보여줍니다.
<code>delete</code>	리소스를 제거합니다.
<code>logs</code>	컨테이너의 로그를 봅니다.
<code>exec</code>	컨테이너에 명령어를 전달합니다. 컨테이너에 접근할 때 주로 사용합니다.
<code>config</code>	kubectl 설정을 관리합니다.

```
# alias 설정
alias k='kubectl'
```

상태 설정하기 (apply)

원하는 리소스의 상태를 yaml로 작성하고 `apply` 명령어로 선언

```
kubectl apply -f [파일명 또는 URL]
```

```
kubectl apply -f https://subicura.com/k8s/code/guide/index/wordpress-k8s.yml
```

리소스 목록보기 (get)

쿠버네티스에 선언된 리소스 확인

```
kubectl get [TYPE]
```

```
# Pod 조회
kubectl get pod

# 줄임말(Shortname)과 복수형 사용가능
kubectl get pods
kubectl get po

# 여러 TYPE 입력
kubectl get pod,service
kubectl get po,svc

# Pod, ReplicaSet, Deployment, Service, Job 조회 => all
kubectl get all

# 결과 포맷 변경
kubectl get pod -o wide
kubectl get pod -o yaml
kubectl get pod -o json

# Label 조회
kubectl get pod --show-labels
```

`kubectl api-resources` : 현재 쿠버네티스에서 사용할 수 있는 리소스 네임이 뜬다 (숫네임 확인 가능)

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
λ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/wordpress-74757b6ff-brp7v      1/1     Running   0           8m23s
pod/wordpress-mysql-5447bfc5b-gvlls 1/1     Running   0           8m23s

NAME                                TYPE               CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/kubernetes                 ClusterIP          10.96.0.1       <none>            443/TCP          85m
service/wordpress                  NodePort           10.100.155.15   <none>            80:32521/TCP     8m23s
service/wordpress-mysql            ClusterIP          10.110.20.14    <none>            3306/TCP          8m24s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/wordpress           1/1     1             1           8m24s
deployment.apps/wordpress-mysql     1/1     1             1           8m24s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/wordpress-74757b6ff 1         1         1       8m23s
replicaset.apps/wordpress-mysql-5447bfc5b 1         1         1       8m23s

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
λ kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
wordpress-74757b6ff-brp7v          1/1     Running   0           8m23s
wordpress-mysql-5447bfc5b-gvlls    1/1     Running   0           8m28s

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
λ kubectl get pod -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE             NOMINATED NODE   READINESS GATES
wordpress-74757b6ff-brp7v          1/1     Running   0           8m36s  172.17.0.4      minikube         <none>            <none>
wordpress-mysql-5447bfc5b-gvlls    1/1     Running   0           8m36s  172.17.0.3      minikube         <none>            <none>

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
λ kubectl get pod -o yaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Pod
  metadata:
    creationTimestamp: "2023-08-13T12:17:40Z"
    generateName: wordpress-74757b6ff-
    labels:
      app: wordpress
      pod-template-hash: 74757b6ff
      tiers: frontend
    name: wordpress-74757b6ff-brp7v
    namespace: default
    ownerReferences:
      - apiVersion: apps/v1
        blockOwnerDeletion: true
        controller: true
        kind: ReplicaSet
        name: wordpress-74757b6ff
        uid: d3baa948-7e02-432b-a2f7-3c464a7de718
    resourceVersion: "3755"
    uid: 96d767e9-b080-4f23-8319-6e2c5ead2016
  spec:
    containers:
      - env:
          - name: WORDPRESS_DB_HOST
            value: wordpress-mysql
          - name: WORDPRESS_DB_NAME
            value: wordpress
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
λ kubectl get pod --show-labels
NAME                                READY   STATUS    RESTARTS   AGE   LABELS
wordpress-74757b6ff-brp7v          1/1     Running   0           10m   app=wordpress,pod-template-hash=74757b6ff,tier=frontend
wordpress-mysql-5447bfc5b-gvlls    1/1     Running   0           10m   app=wordpress,pod-template-hash=5447bfc5b,tier=mysql

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
```

리소스 상세 상태보기 (describe)

쿠버네티스에 선언된 리소스의 상세한 상태를 확인

(특정 리소스의 상태가 궁금할 때, 생성 실패의 이유를 확인할 때)

```
kubectl describe [TYPE]/[NAME] 또는 [TYPE] [NAME]
```

```
# Pod 조회로 이름 검색
kubectl get pod

# 조회한 이름으로 상세 확인
kubectl describe pod/wordpress-5f59577d4d-8t2dg # 환경마다 이름이 다릅니다
```



```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
λ kubectl describe po/wordpress-74757b6ff-brp7v
Name: wordpress-74757b6ff-brp7v
Namespace: default
Priority: 0
Service Account: default
Node: minikube/192.168.49.2
Start Time: Sun, 13 Aug 2023 21:17:41 +0900
Labels: app=wordpress, pod-template-hash=74757b6ff
Annotations: <none>
Status: Running
IP: 172.17.0.4
IPs: IP: 172.17.0.4
Controlled By: ReplicaSet/wordpress-74757b6ff
Containers:
  wordpress:
    Container ID: docker://b92da5ecc7efc5ee7ac828abdd26e53da262597884b5cd40beda57c1176354e4
    Image: wordpress:5.9.1-php8.1-apache
    Image ID: docker-pullable://wordpress@sha256:54c1c0d69afb68aa972d2be131265200b799dbfabd9eda50eb0b3ff17023b1c4
    Port: 80/TCP
    Host Port: 0/TCP
    State: Running
      Started: Sun, 13 Aug 2023 21:17:47 +0900
    Ready: True
    Restart Count: 0
    Environment:
      WORDPRESS_DB_HOST: wordpress-mysql
      WORDPRESS_DB_NAME: wordpress
      WORDPRESS_DB_USER: root
      WORDPRESS_DB_PASSWORD: password
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-78hnh (ro)
Conditions:
  Type              Status
  Initialized        True
  Ready              True
  ContainersReady    True
  PodScheduled       True
Volumes:
  kube-api-access-78hnh:
    Type: Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName: kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI: true
QoS Class: BestEffort
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s
              node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type      Reason      Age   From          Message
  ----      -
  Normal    Scheduled   11m   default-scheduler   Successfully assigned default/wordpress-74757b6ff-brp7v to minikube
  Normal    Pulled      11m   kubelet         Container image "wordpress:5.9.1-php8.1-apache" already present on machine
  Normal    Created     11m   kubelet         Created container wordpress
  Normal    Started     11m   kubelet         Started container wordpress
```

Events를 많이 본다 (여러 원인도 여기서 보기 때문이다)

리소스 제거 (delete)

쿠버네티스에 선언된 리소스를 제거

```
kubectl delete [TYPE]/[NAME] 또는 [TYPE] [NAME]
```

```
# Pod 조회로 이름 검색
kubectl get pod

# 조회한 Pod 제거 (다시 살아난다)
kubectl delete pod/wordpress-5f59577d4d-8t2dg
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
λ kubectl delete po/wordpress-74757b6ff-brp7v
pod "wordpress-74757b6ff-brp7v" deleted
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
λ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
wordpress-74757b6ff-wvb5n          1/1     Running   0           14s
wordpress-mysql-5447bfc5b-gvlls    1/1     Running   0           12m
```

컨테이너 로그 조회 (logs)

컨테이너의 로그를 확인 (실시간 로그 -f, 하나의 Pod에 여러개의 컨테이너 있을 시 -c)

```
kubectl logs [POD_NAME]
```

```
# Pod 조회로 이름 검색
kubectl get pod

# 조회한 Pod 로그조회
kubectl logs wordpress-5f59577d4d-8t2dg

# 실시간 로그 보기
kubectl logs -f wordpress-5f59577d4d-8t2dg
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
λ kubectl logs wordpress-74757b6ff-wvb5n
WordPress not found in /var/www/html - copying now...
Complete! WordPress has been successfully copied to /var/www/html
No 'wp-config.php' found in /var/www/html, but 'WORDPRESS_...' variables supplied; copying 'wp-config-docker.php' (WORDPRESS_DB_HOST WORDPRESS_DB_NAME WORDPRESS_DB_PASSWORD WORDPRESS_DB_USER WORDPRESS_DB_CHARSET WORDPRESS_MYSQL_PORT_3306_TCP_ADDR WORDPRESS_MYSQL_PORT_3306_TCP_PORT WORDPRESS_MYSQL_PORT_3306_TCP_PROTO WORDPRESS_MYSQL_SERVICE_HOST WORDPRESS_MYSQL_SERVICE_PORT WORDPRESS_PORT WORDPRESS_PORT_80_TCP_PORT WORDPRESS_PORT_80_TCP_PROTO WORDPRESS_SERVICE_HOST WORDPRESS_SERVICE_PORT)
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.5. Set the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.5. Set the 'ServerName' directive globally to suppress this message
[Sun Aug 13 12:30:25.407076 2023] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.52 (Debian) PHP/8.1.3 configured -- resuming normal operations
[Sun Aug 13 12:30:25.414248 2023] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
λ kubectl logs -f wordpress-74757b6ff-wvb5n
WordPress not found in /var/www/html - copying now...
Complete! WordPress has been successfully copied to /var/www/html
No 'wp-config.php' found in /var/www/html, but 'WORDPRESS_...' variables supplied; copying 'wp-config-docker.php' (WORDPRESS_DB_HOST WORDPRESS_DB_NAME WORDPRESS_DB_PASSWORD WORDPRESS_DB_USER WORDPRESS_DB_CHARSET WORDPRESS_MYSQL_PORT_3306_TCP_ADDR WORDPRESS_MYSQL_PORT_3306_TCP_PORT WORDPRESS_MYSQL_PORT_3306_TCP_PROTO WORDPRESS_MYSQL_SERVICE_HOST WORDPRESS_MYSQL_SERVICE_PORT WORDPRESS_PORT WORDPRESS_PORT_80_TCP_PORT WORDPRESS_PORT_80_TCP_PROTO WORDPRESS_SERVICE_HOST WORDPRESS_SERVICE_PORT)
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.5. Set the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.5. Set the 'ServerName' directive globally to suppress this message
[Sun Aug 13 12:30:25.407076 2023] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.52 (Debian) PHP/8.1.3 configured -- resuming normal operations
```

컨테이너 명령어 전달 (exec)

컨테이너에 접속 (컨테이너 상태확인 -it, 여러 개의 컨테이너 존재 시 -c)

```
kubectl exec [-it] [POD_NAME] -- [COMMAND]
```

```
# Pod 조회로 이름 검색
kubectl get pod

# 조회한 Pod의 컨테이너에 접속
kubectl exec -it wordpress-5f59577d4d-8t2dg -- bash
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide
λ kubectl exec -it wordpress-74757b6ff-wvb5n -- bash
root@wordpress-74757b6ff-wvb5n:/var/www/html# ls -al
total 244
drwxr-xr-x 5 www-data www-data 4096 Aug 13 12:30 .
drwxr-xr-x 1 root root 4096 Mar 1 2022 ..
-rw-r--r-- 1 www-data www-data 261 Mar 11 2022 .htaccess
-rw-r--r-- 1 www-data www-data 405 Feb 6 2020 index.php
-rw-r--r-- 1 www-data www-data 19915 Jan 1 2022 license.txt
-rw-r--r-- 1 www-data www-data 7437 Dec 28 2021 readme.html
-rw-r--r-- 1 www-data www-data 7165 Jan 21 2021 wp-activate.php
drwxr-xr-x 9 www-data www-data 4096 Feb 22 2022 wp-admin
-rw-r--r-- 1 www-data www-data 351 Feb 6 2020 wp-blog-header.php
-rw-r--r-- 1 www-data www-data 2338 Nov 9 2021 wp-comments-post.php
-rw-rw-r-- 1 www-data www-data 5480 Mar 11 2022 wp-config-docker.php
-rw-r--r-- 1 www-data www-data 3001 Dec 14 2021 wp-config-sample.php
-rw-r--r-- 1 www-data www-data 5584 Aug 13 12:30 wp-config.php
drwxr-xr-x 5 www-data www-data 4096 Feb 22 2022 wp-content
-rw-r--r-- 1 www-data www-data 3939 Aug 3 2021 wp-cron.php
drwxr-xr-x 26 www-data www-data 16384 Feb 22 2022 wp-includes
-rw-r--r-- 1 www-data www-data 2496 Feb 6 2020 wp-links-opml.php
-rw-r--r-- 1 www-data www-data 3900 May 15 2021 wp-load.php
-rw-r--r-- 1 www-data www-data 47916 Jan 4 2022 wp-login.php
-rw-r--r-- 1 www-data www-data 8582 Sep 22 2021 wp-mail.php
-rw-r--r-- 1 www-data www-data 23025 Nov 30 2021 wp-settings.php
-rw-r--r-- 1 www-data www-data 31959 Oct 25 2021 wp-signup.php
-rw-r--r-- 1 www-data www-data 4747 Oct 8 2020 wp-trackback.php
-rw-r--r-- 1 www-data www-data 3236 Jun 8 2020 xmlrpc.php
root@wordpress-74757b6ff-wvb5n:/var/www/html#
```

도커 같은 경우에는 여러 서버에 컨테이너가 있으면 그 컨테이너가 실행중인 서버를 찾아서 먼저 그 서버에 접속을 하고 컨테이너에 접속을 해야하는데 kubectl을 사용하면 자동으로 어디에 있던 연결을 해줌. 중앙에서 제어 쉬움. 리스트를 꼭 보고 접속하고 싶은 pod에 이름으로 접속할 수 있음

설정 관리 (config) & 기타

```
# 현재 컨텍스트 확인
kubectl config current-context

# 컨텍스트 설정
kubectl config use-context minikube

# 전체 오브젝트 종류 확인
kubectl api-resources

# 특정 오브젝트 설명 보기
kubectl explain pod
```

```
kubectl delete -f https://subicura.com/k8s/code/guide/index/wordpress-k8s.yml
```

Pod

Pod : 쿠버네티스에서 관리하는 가장 작은 배포 단위

도커랑 비교해보면 도커는 보통 컨테이너를 만든다고 이야기를 한다. 쿠버네티스는 컨테이너를 만들긴 하지만 컨테이너 자체를 관리하는 게 아니라 pod를 통해서 컨테이너를 관리한다. 그러니깐 쿠버네티스에서는 pod를 배포한다고 하고 그 pod 안에 컨테이너가 존재하는 구조. 그리고 그 pod 안에는 보통 컨테이너를 하나만 쓰기는 하는데 하나 또는 여러개의 컨테이너를 포함할 수도 있다.

빠르게 Pod 만들기

```
kubectl run echo --image ghcr.io/subicura/echo:v1
kubectl get pod
kubectl describe pod/echo
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/pod
λ kubectl run echo --image ghcr.io/subicura/echo:v1
pod/echo created
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/pod
λ kubectl get pod
NAME READY STATUS RESTARTS AGE
echo 0/1 ContainerCreating 0 15s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/pod
λ kubectl get pod
NAME READY STATUS RESTARTS AGE
echo 0/1 ContainerCreating 0 23s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/pod
λ kubectl get pod
NAME READY STATUS RESTARTS AGE
echo 0/1 ContainerCreating 0 37s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/pod
λ kubectl get pod
NAME READY STATUS RESTARTS AGE
echo 1/1 Running 0 46s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/pod
λ kubectl describe pod/echo
error: unknown command "describe" for "kubectl"
```

설정 관리 (config) & 기타

- 현재 컨텍스트 확인
kubectl config current-context
- 컨텍스트 설정
kubectl config use-context minikube
- 전체 오브젝트 종류 확인
kubectl api-resources
- 특정 오브젝트 설명 보기
kubectl explain pod

Pod

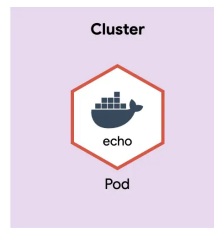
Pod : 쿠버네티스에서 관리하는 가장 작은 배포 단위

도커랑 비교해보면 도커는 보통 컨테이너를 만든다고 이야기를 한다. 쿠버네티스는 컨테이너를 만들긴 하지만 컨테이너 자체를 관리하는 게 아니라 pod를 통해서 컨테이너를 관리한다. 그러니깐 쿠버네티스에서는 pod를 배포한다고 하고 그 pod 안에 컨테이너가 존재하는 구조. 그리고 그 pod 안에는 보통 컨테이너를 하나만 쓰기는 하는데 하나 또는 여러개의 컨테이너를 포함할 수도 있다.

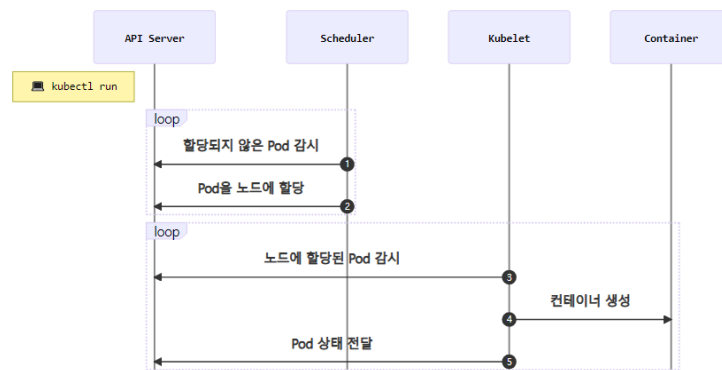
```
Did you mean this?
describe
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/pod
λ kubectl describe pod/echo
Name: echo
Namespace: default
Priority: 0
Service Account: default
Node: minikube/192.168.49.2
Start Time: Sun, 13 Aug 2023 21:45:46 +0900
Labels: run-echo
Annotations: <none>
Status: Running
IP: 172.17.0.3
IPs: 172.17.0.3
Containers:
  echo:
    Container ID: docker://5df5665801d4e0e278e97ca4ae72b585c3ff7fe87c176b51e91e41ea39bb62fc
    Image: ghcr.io/subicura/echo:v1
    Image ID: docker-pullable://ghcr.io/subicura/echo@sha256:486ea26869d277e4090812fe5ebb694eb754791bb5568f23983775948eee3c3
```

Pod 생성 분석

minikube 클러스터 안에 Pod, Pod 안에 컨테이너 존재.



Pod의 구성



Pod 생성 과정

컨테이너는 도커라고 볼 수도 있다.

```
kubectl delete pod/echo
```

YAML로 설정파일Spec 작성하기

스펙 작성시 필수 요소

정의	설명	예
<code>version</code>	오브젝트 버전	v1, app/v1, networking.k8s.io/v1, ...
<code>kind</code>	종류	Pod, ReplicaSet, Deployment, Service, ...
<code>metadata</code>	메타데이터	name과 label, annotation(주석)으로 구성
<code>spec</code>	상세명세	리소스 종류마다 다름

guide/pod/echo-pod.yml

```

apiVersion: v1
kind: Pod
metadata:
  name: echo
  labels:
    app: echo
spec:
  containers:
    - name: app
      image: ghcr.io/subicura/echo:v1

```

```

# Pod 생성
kubectl apply -f echo-pod.yml

```

```
# Pod 목록 조회
kubectl get pod

# Pod 로그 확인
kubectl logs echo
kubectl logs -f echo

# Pod 컨테이너 접속
kubectl exec -it echo -- sh
# ls
# ps
# exit

# Pod 제거
kubectl delete -f echo-pod.yml
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ ls
echo-pod.yml
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl apply -f echo-pod.yml
pod/echo created
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl get po
NAME      READY   STATUS    RESTARTS   AGE
echo      1/1     Running   0           2m1s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl logs echo
{"level":30,"time":1691931304001,"pid":7,"hostname":"echo","msg":"Server listening at http://0.0.0.0:3000"}
{"level":30,"time":1691931304006,"pid":7,"hostname":"echo","msg":"server listening on http://0.0.0.0:3000"}
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl exec -it echo -- sh
/app # ls
Dockerfile      README.md      app.js      node_modules  package-lock.json  package.json
/app # ps
PID   USER     TIME   COMMAND
1  root     0:00   /sbin/tini -- node app.js
7  root     0:01   node app.js
14  root     0:00   sh
21  root     0:00   ps
/app # exit
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl delete -f echo-pod.yml
pod "echo" deleted
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl get po
No resources found in default namespace.
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ
```

컨테이너 상태 모니터링

보통 서버를 실행하잖아. 자바로 만든 서버가 있는데 컨테이너로 실행하면 컨테이너는 실행이 됐는데 자바가 부팅을 하는데 시간이 짧으면 몇초일 수도 있지만 길면 몇분이 걸리는 부팅시간이 있고 그 부팅하는 동안 접속하면 당연히 에러가 난다. 따라서 pod에서는 서비스가 준 비되었다라는 것을 체크할 수 있는 기능을 제공한다.

LivenessProbe

컨테이너가 정상적으로 동작하는지 체크하고 정상적으로 동작하지 않는다면 **컨테이너를 재시작**하여 문제를 해결

guide/pod/echo-lp.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: echo-lp
  labels:
    app: echo
spec:
  containers:
    - name: app
      image: ghcr.io/subicura/echo:v1
      livenessProbe:
        httpGet:
          path: /not/exist
```

```
port: 8080
initialDelaySeconds: 5
timeoutSeconds: 2 # Default 1
periodSeconds: 5 # Defaults 10
failureThreshold: 1 # Defaults 3
```

livenessProbe라는 옵션이 있고, 이는 httpGet 명령어로 /not/exist라는 path에 8080 포트로 요청을 보낸다. 처음 5초 (initialDelaySeconds)를 시도를 하고 타임 아웃은 최대 2초(timeoutSeconds), 5초마다 한번씩 체크를 할 건데(periodSeconds), 만약 1 번이라도 실패를 하면(failureThreshold) 컨테이너를 재시작 한다.

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl apply -f echo-lp.yml
pod/echo-lp created
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl get po
NAME      READY   STATUS    RESTARTS   AGE
echo-lp   1/1     Running   1 (4s ago)  14s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl get po
NAME      READY   STATUS    RESTARTS   AGE
echo-lp   1/1     Running   2 (3s ago)  23s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl get po
NAME      READY   STATUS    RESTARTS   AGE
echo-lp   1/1     Running   3 (19s ago)  44s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
```

readinessProbe

컨테이너가 준비되었는지 체크하고 정상적으로 준비되지 않았다면 **Pod으로 들어오는 요청을 제외**합니다. 즉, 컨테이너를 재시작 하지는 않고 단지 들어오는 요청을 막는다. 외부에서 접속할때 pod이 10개가 떠있고 5개는 서버가 났고 5개는 안났을 때 성공적으로 떠있는 서버 5개에만 요청을 보내게 된다. 이 **readinessProbe**를 따로 설정하지 않게 되면 서버가 뜨기전에 모든 pod으로 요청을 보내게 된다.

guide/pod/echo-rp.yml

```
guide/pod/echo-lp.yml에서
livenessProbe:
↓
readinessProbe:
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl apply -f echo-rp.yml
pod/echo-rp created
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl get po
NAME      READY   STATUS             RESTARTS   AGE
echo-lp   0/1     CrashLoopBackOff   6 (99s ago)  4m54s
echo-rp   0/1     Running            0           9s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ
```

livenessProbe + readinessProbe

guide/pod/echo-health.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: echo-health
  labels:
    app: echo
spec:
  containers:
    - name: app
      image: ghcr.io/subicura/echo:v1
      livenessProbe:
```

```
httpGet:
  path: /
  port: 3000
readinessProbe:
  httpGet:
    path: /
    port: 3000
```

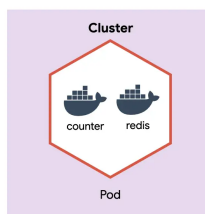
```
lneon@DESKTOP-HCDESIU ~/Desktop/k8s/guide/pod
$ kubectl apply -f echo-health.yml
pod/echo-health created
lneon@DESKTOP-HCDESIU ~/Desktop/k8s/guide/pod
$ kubectl get po
NAME          READY   STATUS    RESTARTS   AGE
echo-health   1/1     Running   0           9s
echo-ip       0/1     CrashLoopBackOff   7 (82s ago)   7s2s
echo-rp       0/1     Running   0           2m47s
lneon@DESKTOP-HCDESIU ~/Desktop/k8s/guide/pod
$ kubectl logs -f echo
Error from server (NotFound): pods "echo" not found
lneon@DESKTOP-HCDESIU ~/Desktop/k8s/guide/pod
$ kubectl logs -f echo-health
{"level":30,"time":"1691932411381","pid":7,"hostname":"echo-health","msg":"Server listening at http://0.0.0.0:3000"}
{"level":30,"time":"1691932411383","pid":7,"hostname":"echo-health","msg":"server listening on http://0.0.0.0:3000"}
{"level":30,"time":"1691932412144","pid":7,"hostname":"echo-health","reqId":"req-1","req":{"method":"GET","url":"/","hostname":"172.17.0.5:3000","remoteAddress":"172.17.0.1","remotePort":57294},"msg":"incoming request"}
{"level":30,"time":"1691932412200","pid":7,"hostname":"echo-health","reqId":"req-1","res":{"statusCode":200},"responseTime":12.567013001806943,"msg":"request completed"}
{"level":30,"time":"1691932417689","pid":7,"hostname":"echo-health","reqId":"req-2","req":{"method":"GET","url":"/","hostname":"172.17.0.5:3000","remoteAddress":"172.17.0.1","remotePort":41048},"msg":"incoming request"}
{"level":30,"time":"1691932417692","pid":7,"hostname":"echo-health","reqId":"req-2","res":{"statusCode":200},"responseTime":2.7526430003345813,"msg":"request completed"}
{"level":30,"time":"1691932417693","pid":7,"hostname":"echo-health","reqId":"req-3","req":{"method":"GET","url":"/","hostname":"172.17.0.5:3000","remoteAddress":"172.17.0.1","remotePort":41046},"msg":"incoming request"}
{"level":30,"time":"1691932417796","pid":7,"hostname":"echo-health","reqId":"req-3","res":{"statusCode":200},"responseTime":12.386742999777198,"msg":"request completed"}
{"level":30,"time":"1691932427691","pid":7,"hostname":"echo-health","reqId":"req-4","req":{"method":"GET","url":"/","hostname":"172.17.0.5:3000","remoteAddress":"172.17.0.1","remotePort":52568},"msg":"incoming request"}
{"level":30,"time":"1691932427693","pid":7,"hostname":"echo-health","reqId":"req-4","res":{"statusCode":200},"responseTime":1.4915469996631145,"msg":"request completed"}
{"level":30,"time":"1691932427694","pid":7,"hostname":"echo-health","reqId":"req-5","req":{"method":"GET","url":"/","hostname":"172.17.0.5:3000","remoteAddress":"172.17.0.1","remotePort":52562},"msg":"incoming request"}
{"level":30,"time":"1691932427696","pid":7,"hostname":"echo-health","reqId":"req-5","res":{"statusCode":200},"responseTime":1.5957459995158566,"msg":"request completed"}
{"level":30,"time":"1691932427688","pid":7,"hostname":"echo-health","reqId":"req-6","req":{"method":"GET","url":"/","hostname":"172.17.0.5:3000","remoteAddress":"172.17.0.1","remotePort":37116},"msg":"incoming request"}
{"level":30,"time":"1691932427696","pid":7,"hostname":"echo-health","reqId":"req-6","res":{"statusCode":200},"responseTime":1.1012489998476595,"msg":"request completed"}
{"level":30,"time":"1691932427698","pid":7,"hostname":"echo-health","reqId":"req-7","req":{"method":"GET","url":"/","hostname":"172.17.0.5:3000","remoteAddress":"172.17.0.1","remotePort":37114},"msg":"incoming request"}
{"level":30,"time":"1691932427692","pid":7,"hostname":"echo-health","reqId":"req-7","res":{"statusCode":200},"responseTime":1.2258419985582958,"msg":"request completed"}
{"level":30,"time":"1691932427699","pid":7,"hostname":"echo-health","reqId":"req-8","req":{"method":"GET","url":"/","hostname":"172.17.0.5:3000","remoteAddress":"172.17.0.1","remotePort":47194},"msg":"incoming request"}
{"level":30,"time":"1691932447691","pid":7,"hostname":"echo-health","reqId":"req-8","res":{"statusCode":200},"responseTime":1.8509670003908873,"msg":"request completed"}
{"level":30,"time":"1691932447692","pid":7,"hostname":"echo-health","reqId":"req-9","req":{"method":"GET","url":"/","hostname":"172.17.0.5:3000","remoteAddress":"172.17.0.1","remotePort":47196},"msg":"incoming request"}
{"level":30,"time":"1691932447694","pid":7,"hostname":"echo-health","reqId":"req-9","res":{"statusCode":200},"responseTime":1.8100850004702886,"msg":"request completed"}
```

다중 컨테이너

guide/pod/counter-pod-redis.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: counter
  labels:
    app: counter
spec:
  containers:
    - name: app
      image: ghcr.io/subicura/counter:latest
      env:
        - name: REDIS_HOST
          value: "localhost"
    - name: db
      image: redis
```

counter라는 앱을 만들고 redis를 백엔드로 한다. counter앱은 redis에 접속을 해서 count + 1을 해주는 단순한 애플리케이션. counter가 redis에 접속을 하기 위해서 localhost로 접속할 것이다. 그러니깐 하나의 Pod에(kind: Pod) 두개의 컨테이너(containers: - name: app - name: db)를 실행할건데 이 앱이 localhost로 접근을 할수있는거임. 이거는 쿠버네티스가 그렇게 구성을 해주는거다.



이 둘은 localhost로 통신을 할 수 있다.

```
# Pod 생성
kubectl apply -f counter-pod-redis.yml

# Pod 목록 조회
```

```
kubectl get pod

# Pod 로그 확인
kubectl logs counter # 오류 발생 (컨테이너 지정 필요)
kubectl logs counter app
kubectl logs counter db

# Pod의 app컨테이너 접속
kubectl exec -it counter -c app -- sh
# curl localhost:3000
# curl localhost:3000
# telnet localhost 6379
dbsize
KEYS *
GET count
quit

# Pod 제거
kubectl delete -f counter-pod-redis.yml
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl apply -f counter-pod-redis.yml
pod/counter created
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl get po
NAME          READY   STATUS             RESTARTS   AGE
counter       0/2     ContainerCreating   0           11s
echo-health   1/1     Running             0           8m57s
echo-ip       0/1     CrashLoopBackOff    9 (4m55s ago)  16m
echo-rp       0/1     Running             0           11m
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl logs counter
Defaulted container 'app' out of: app, db
{"level":30,"time":1691932955703,"pid":8,"hostname":"counter","msg":"Server listening at http://0.0.0.0:3000"}
Redis Client Error Error: connect ECONNREFUSED 127.0.0.1:6379
    at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1157:16) {
  errno: -111,
  code: 'ECONNREFUSED',
  syscall: 'connect',
  address: '127.0.0.1',
  port: 6379
}
Redis Client Error Error: connect ECONNREFUSED 127.0.0.1:6379
    at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1157:16) {
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl exec -it count -c app -- sh
Error from server (NotFound): pods "count" not found
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/pod
λ kubectl exec -it counter -c app -- sh
/app # ls
Dockerfile  README.md  app.js  node_modules  package-lock.json  package.json
/app # apk add curl busybox-extras
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/community/x86_64/APKINDEX.tar.gz
OK: 10 MiB in 23 packages
/app # curl localhost:3000
1
/app # curl localhost:3000
2
/app # telnet localhost 6379
Connected to localhost
dbsize
21
KEYS *
*1
$5
count
GET count
$1
2
quit
+OK
Connection closed by foreign host
/app # quit
sh: quit: not found
/app # exit
command terminated with exit code 127
```

```
kubectl delete -f ./
```

문제 오답노트

```
containers:
- name: ~~
```

하나의 요소라도 `~` 를 붙여줘야한다.

```
env:
- name: MYSQL_ROOT_PASSWORD
  value: "123456"
```


환경변수 설정시에는 name, value로 명시한다.

▪ReplicaSet

Pod을 단독으로 만들면 Pod에 어떤 문제(서버가 죽어서 Pod이 사라졌다면)가 생겼을 때 자동으로 복구되지 않습니다. 이러한 Pod을 정해진 수만큼 복제하고 관리하는 것이 ReplicaSet입니다.

ReplicaSet 만들기

guide/replicaset/echo-rs.yml

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: echo-rs
spec:
  replicas: 1
  selector:
    matchLabels:
      app: echo
      tier: app
  template:
    metadata:
      labels:
        app: echo
        tier: app
    spec:
      containers:
        - name: echo
          image: ghcr.io/subicura/echo:v1
```

replicas를 몇개를 사용할 거냐를 1로 설정했음. 그래서 Pod을 하나가 뜬거. selector가 뭐냐면 어떤 label을 찾을거냐(matchLabels)라고 하는데 보니까 app은 echo면서 tier는 app인 pod을 체크하겠단. 그런데 그런 pod이 없다고 하면 템플릿에 있는 내용(template:~~)을 보고 ~~한 형태의 pod을 만든다. template 아래의 내용은 앞서서 pod에서 작성했던 내용과 동일하다. 즉, ReplicaSet은 위의 조건을 찾는 데 이 조건에 맞는 pod없으면 template에 있는 pod을 만드는 것이다.

```
# ReplicaSet 생성
kubectl apply -f echo-rs.yml

# 리소스 확인
kubectl get po,rs
```

```
leeoo@DESKTOP-HCDB5IU ~/Desktop/k8s/guide/replicaset
λ kubectl apply -f echo-rs.yml
replicaset.apps/echo-rs created
leeoo@DESKTOP-HCDB5IU ~/Desktop/k8s/guide/replicaset
λ kubectl get po,rs
error: arguments in resource/name form must have a single resource and name
leeoo@DESKTOP-HCDB5IU ~/Desktop/k8s/guide/replicaset
λ kubectl get po,rs
NAME                                READY   STATUS    RESTARTS   AGE
pod/echo-rs-tck99                  1/1     Running   0           23s
```

```
NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/echo-rs             1         1         1       23s
```

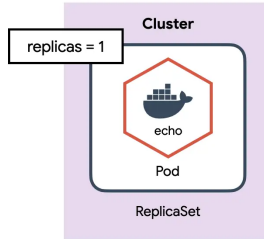
복구되지 않습니다. 이러한 Pod을 정해진 수만큼 복제하고 관리하는 것이 Rep

ReplicaSet 만들기

guide/replicaset/echo-rs.yml

apiVersion: apps/v1

kind: ReplicaSet



클러스터 안에 ReplicaSet이 이렇게 하나 있는데 ReplicaSet의 조건이 1개. 그래서 Pod하나를 ReplicaSet안에 만든거다. ReplicaSet이 Pod을 관리하는 모습.

label을 보고 관리하는게 맞는지 체크해보자!

```
# 생성된 Pod의 label 확인
kubectl get pod --show-labels

# app- 를 지정하면 app label을 제거
kubectl label pod/echo-rs-tcdwj app-

# 다시 Pod 확인
kubectl get pod --show-labels

# app=echo로 다시 label 붙이기
kubectl label pod/echo-rs-tcdwj app=echo

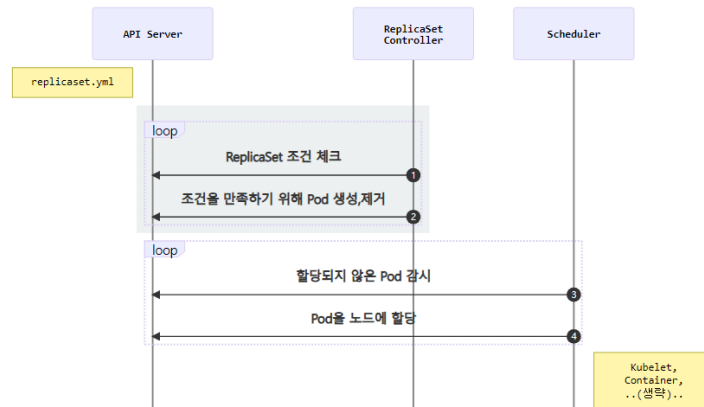
# 다시 Pod 확인
kubectl get pod --show-labels
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/replicaset
λ kubectl get po --show-labels
NAME          READY   STATUS    RESTARTS   AGE   LABELS
echo-rs-tck99 1/1     Running   0          6m20s app=echo,tier=app
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/replicaset
λ kubectl label pod/echo-rs-tck99 app-
pod/echo-rs-tck99 unlabeled
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/replicaset
λ kubectl get po --show-labels
NAME          READY   STATUS    RESTARTS   AGE   LABELS
echo-rs-g6fbk 1/1     Running   0          9s    app=echo,tier=app
echo-rs-tck99 1/1     Running   0          7m4s  tier=app
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/replicaset
λ kubectl label pod/echo-rs-tck99 app=echo
pod/echo-rs-tck99 labeled
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/replicaset
λ kubectl get po --show-labels
NAME          READY   STATUS    RESTARTS   AGE   LABELS
echo-rs-tck99 1/1     Running   0          7m41s app=echo,tier=app
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/replicaset
λ
```



label을 제거하면 그 label을 가진 다른 pod를 알아서 생성하고 이전에 지웠던 라벨을 다시 추가하면 철저하게 다시 1개가 되도록 다른 pod를 삭제시킨다.

ReplicaSet 동작 방식



스케일 아웃

guide/replicaset/echo-rs.yml

```
guide/replicaset/echo-rs.yml 에서
replicas: 1
↓
replicas: 4
```

```

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/replicaset
λ vi echo-rs.yml
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/replicaset
λ kubectl apply -f echo-rs.yml
replicaset.apps/echo-rs configured
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/replicaset
λ kubectl get po
NAME          READY   STATUS    RESTARTS   AGE
echo-rs-k9c8l 1/1     Running   0           14s
echo-rs-tck99 1/1     Running   0           13m
echo-rs-tgds1 1/1     Running   0           14s
echo-rs-v92p8 1/1     Running   0           14s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/replicaset
λ
  
```

문제 오답노트

```
apiVersion: v1 이 아닌
apiVersion: apps/v1
```

Deployment

Deployment 는 가장 흔하게 사용하는 배포방식이다. 이외에 StatefulSet, DaemonSet, CronJob, Job 등이 있지만 사용법은 크게 다르지 않다.

deployment 만들기

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: echo-deploy
spec:
  replicas: 4
  selector:
    matchLabels:
      app: echo
      tier: app
  
```

```

template:
  metadata:
    labels:
      app: echo
      tier: app
  spec:
    containers:
      - name: echo
        image: ghcr.io/subicura/echo:v1

```

```

# Deployment 생성
kubectl apply -f echo-deployment.yml

# 리소스 확인
kubectl get po,rs,deploy

```

```

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/deployment
λ kubectl apply -f echo-deployment.yml
deployment.apps/echo-deploy created
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/deployment
λ get all
C:\Users\leeoo\Desktop\cmder\vendor\conemu-maximus5\..\git-for-windows\usr\bin\bash: get: command not found
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/deployment
λ kubectl get all
NAME                                READY    STATUS    RESTARTS   AGE
pod/echo-deploy-77dbb94b69-bb9rx    1/1      Running   0           35s
pod/echo-deploy-77dbb94b69-fcmqj    1/1      Running   0           35s
pod/echo-deploy-77dbb94b69-g9pwq    1/1      Running   0           35s
pod/echo-deploy-77dbb94b69-v4qq1    1/1      Running   0           35s
pod/echo-rs-k9c81                   1/1      Running   3 (3m44s ago)  11d
pod/echo-rs-tck99                   1/1      Running   3 (3m44s ago)  11d
pod/echo-rs-tgds1                   1/1      Running   3 (3m44s ago)  11d
pod/echo-rs-v92p8                   1/1      Running   3 (3m44s ago)  11d

NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes  ClusterIP   10.96.0.1    <none>        443/TCP   11d

NAME      READY    UP-TO-DATE   AVAILABLE   AGE
deployment.apps/echo-deploy  4/4        4             4           35s

NAME                DESIRED   CURRENT   READY   AGE
replicaset.apps/echo-deploy-77dbb94b69  4         4         4       35s
replicaset.apps/echo-rs                  4         4         4       11d

```

v1 → v2로 변경 후 리소스 확인

```

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/deployment
λ kubectl get all
NAME                                READY    STATUS    RESTARTS   AGE
pod/echo-deploy-665857f7dd-lrzkg    0/1      ContainerCreating   0           16s
pod/echo-deploy-665857f7dd-pdzbd    0/1      ContainerCreating   0           15s
pod/echo-deploy-77dbb94b69-fcmqj    1/1      Running             0           119s
pod/echo-deploy-77dbb94b69-g9pwq    1/1      Running             0           119s
pod/echo-deploy-77dbb94b69-v4qq1    1/1      Running             0           119s
pod/echo-rs-k9c81                   1/1      Running             3 (5m8s ago)  11d
pod/echo-rs-tck99                   1/1      Running             3 (5m8s ago)  11d
pod/echo-rs-tgds1                   1/1      Running             3 (5m8s ago)  11d
pod/echo-rs-v92p8                   1/1      Running             3 (5m8s ago)  11d

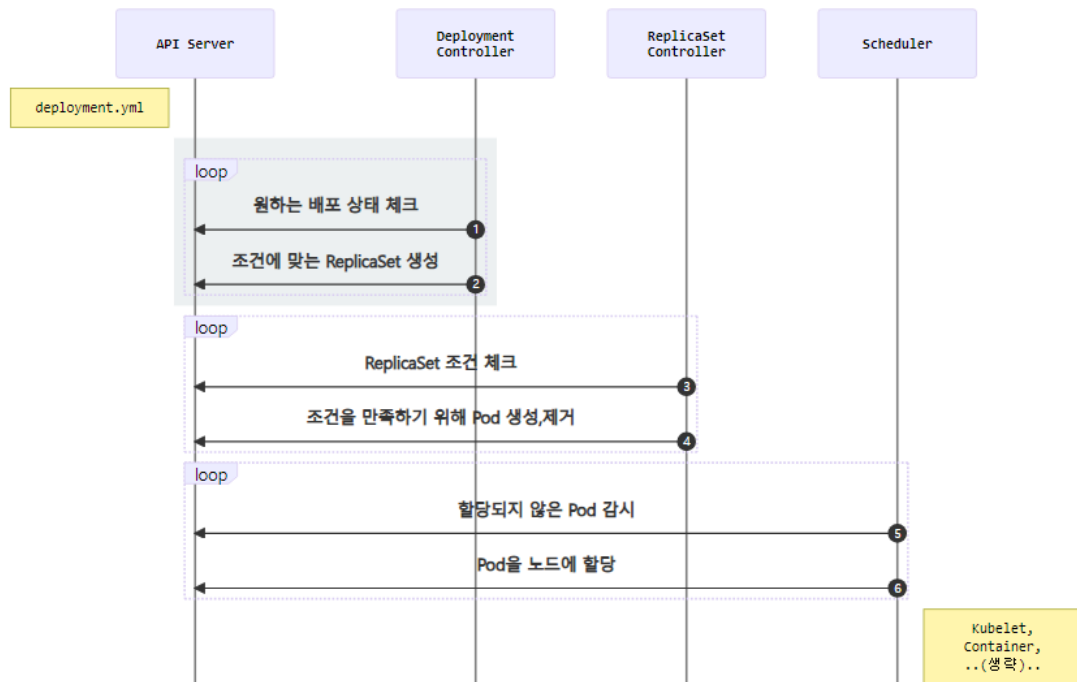
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes  ClusterIP   10.96.0.1    <none>        443/TCP   11d

NAME      READY    UP-TO-DATE   AVAILABLE   AGE
deployment.apps/echo-deploy  3/4        2             3           119s

NAME                DESIRED   CURRENT   READY   AGE
replicaset.apps/echo-deploy-665857f7dd  2         2         0       16s
replicaset.apps/echo-deploy-77dbb94b69  3         3         3       119s
replicaset.apps/echo-rs                  4         4         4       11d

```

Deployment는 새로운 이미지로 업데이트하기 위해 ReplicaSet을 이용합니다. 버전을 업데이트하면 새로운 ReplicaSet을 생성하고 해당 ReplicaSet이 새로운 버전의 Pod을 생성합니다.



버전관리

```

# 히스토리 확인
kubectl rollout history deploy/echo-deploy

# revision 1 히스토리 상세 확인
kubectl rollout history deploy/echo-deploy --revision=1

# 바로 전으로 롤백
kubectl rollout undo deploy/echo-deploy

# 특정 버전으로 롤백
kubectl rollout undo deploy/echo-deploy --to-revision=2
  
```

```

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/deployment
λ kubectl rollout history deploy/echo-deploy
deployment.apps/echo-deploy
REVISION  CHANGE-CAUSE
1          <none>
2          <none>

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/deployment
λ kubectl rollout history deploy/echo-deploy --revision=1
deployment.apps/echo-deploy with revision #1
Pod Template:
  Labels:      app=echo
              pod-template-hash=77dbb94b69
              tier=app
  Containers:
    echo:
      Image:    ghcr.io/subicura/echo:v1
      Port:     <none>
      Host Port: <none>
      Environment:
      Mounts:    <none>
      Volumes:   <none>

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/deployment
λ kubectl rollout undo deploy/echo-deploy
deployment.apps/echo-deploy rolled back
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/deployment
λ kubectl rollout undo deploy/echo-deploy --to-revision=2
deployment.apps/echo-deploy rolled back
  
```

배포 전략 설정

롤링업데이트RollingUpdate 방식을 사용할 때 동시에 업데이트하는 Pod의 개수를 변경하는 방법

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: echo-deploy-st
spec:
  replicas: 4
  selector:
    matchLabels:
      app: echo
      tier: app
  minReadySeconds: 5
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 3
      maxUnavailable: 3
  template:
    metadata:
      labels:
        app: echo
        tier: app
    spec:
      containers:
        - name: echo
          image: ghcr.io/subicura/echo:v1
          livenessProbe:
            httpGet:
              path: /
              port: 3000
```

Service

Service(서비스)를 이용하여 Pod을 노출하고 클러스터 외부에서 접근할 수 있는 방법을 알아보자!

Service (ClusterIP) 만들기

- ClusterIP는 클러스터 내부에 새로운 IP를 할당하고 여러 개의 Pod을 바라보는 로드밸런서 기능을 제공.
- 내부 도메인 서버에 등록하여 Pod 간에 서비스 이름으로 통신 가능

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
spec:
  selector:
    matchLabels:
      app: counter
      tier: db
  template:
    metadata:
      labels:
        app: counter
        tier: db
    spec:
      containers:
        - name: redis
          image: redis
          ports:
            - containerPort: 6379
              protocol: TCP

---
apiVersion: v1
kind: Service
metadata:
  name: redis
spec:
  ports:
```

```
- port: 6379
  protocol: TCP
selector:
  app: counter
  tier: db
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/service
λ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/redis-574497df9d-nz8gm          1/1     Running   0           3m3s

NAME                                TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/kubernetes                  ClusterIP     10.96.0.1    <none>         443/TCP    11d
service/redis                       ClusterIP     10.103.215.93 <none>        6379/TCP   3m3s

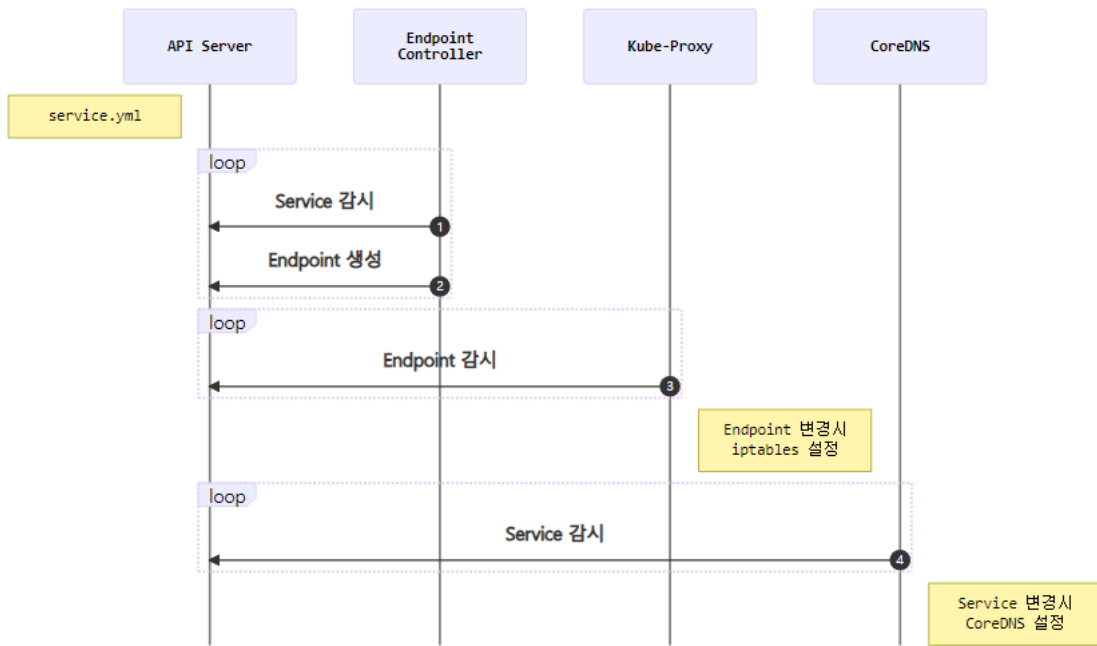
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/redis               1/1     1            1           3m3s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/redis-574497df9d    1         1         1       3m3s
```

counter app Pod에서 redis Pod으로 접근이 되는지 테스트

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/service
λ kubectl exec -it counter-5ddb989cb6-r479n -- sh
/app # curl localhost:3000
1
/app # curl localhost:3000
2
/app # telnet redis 6379
Connected to redis
dbsize
:1
KEYS *
*1
$5
count 0
GET count
$1
2
quit
+OK
Connection closed by foreign host.
/app # exit
command terminated with exit code 1
```

Service 생성 흐름



!! endpoints : 서비스 접속 정보를 가지고 있음

```

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/service
λ kubectl get endpoints
NAME            ENDPOINTS            AGE
kubernetes      192.168.49.2:8443    11d
redis           172.17.0.7:6379      8m5s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/service
λ kubectl describe ep/redis
Name:           redis
Namespace:      default
Labels:         <none>
Annotations:    endpoints.kubernetes.io/last-change-trigger-time: 2023-08-25T04:51:38Z
Subsets:
  Addresses:    172.17.0.7
  NotReadyAddresses: <none>
  Ports:
    Name      Port      Protocol
    ----      -
    <unset>    6379     TCP
Events: <none>
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/service
λ get po -o wide
C:\Users\leeoo\Desktop\cmdr\vendor\conemu-maximus5\..\git-for-windows\usr\bin\bash: get: command not found
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/service
λ kubectl get po -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP            NODE   NOMINATED NODE   READINESS GATES
counter-5ddb989cb6-r479n    1/1     Running   0           3m33s  172.17.0.2    minikube   <none>           <none>
redis-574497df9d-nz8gn     1/1     Running   0           8m54s  172.17.0.7    minikube   <none>           <none>
  
```

Service (NodePort) 만들기

CluterIP는 클러스터 내부에서만 접근할 수 있으므로 클러스터 외부(노드)에서 접근할 수 있도록 NodePort 서비스를 만들기

```

apiVersion: v1
kind: Service
metadata:
  name: counter-np
spec:
  type: NodePort
  ports:
    - port: 3000
      protocol: TCP
      nodePort: 31000
  selector:
    app: counter
    tier: app
  
```


Service(LoadBalancer) 만들기

NodePort의 단점은 노드가 사라졌을 때 자동으로 다른 노드를 통해 접근이 불가능한 것.

자동으로 살아 있는 노드에 접근하기 위해 모든 노드를 바라보는 **Load Balancer** 가 필요

```
apiVersion: v1
kind: Service
metadata:
  name: counter-lb
spec:
  type: LoadBalancer
  ports:
    - port: 30000
      targetPort: 3000
      protocol: TCP
  selector:
    app: counter
    tier: app
```

```
service/counter-lb created
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/service
λ kubectl get svc
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
counter-lb    LoadBalancer 10.108.89.183  <pending>      30000:30910/TCP  13s
counter-np    NodePort      10.99.109.148  <none>         3000:31000/TCP   43m
kubernetes    ClusterIP     10.96.0.1      <none>         443/TCP          11d
redis         ClusterIP     10.103.215.93  <none>         6379/TCP         57m
```

minikube에 가상 LoadBalancer 만들기

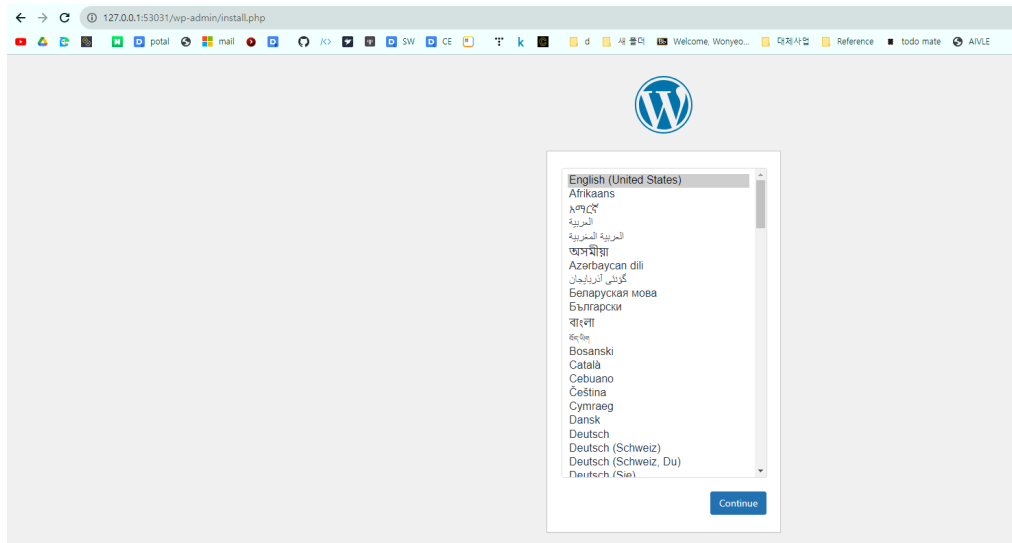
```
minikube addons enable metallb
kubectl apply -f metallb-cm.yml
kubectl get svc
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/service
λ minikube addons enable metallb
 metallb is a 3rd party addon and is not maintained or verified by minikube maintainers, enable at your own risk.
 metallb does not currently have an associated maintainer.
   ▪ Using image quay.io/metallb/controller:v0.9.6
   ▪ Using image quay.io/metallb/speaker:v0.9.6
◆ 'metallb' 애드온이 활성화되었습니다
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/service
λ kubectl apply -f metallb-cm.yml
configmap/config configured
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/service
λ kubectl get svc
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
counter-lb    LoadBalancer 10.108.89.183  <pending>      30000:30910/TCP  70s
counter-np    NodePort      10.99.109.148  <none>         3000:31000/TCP   43m
kubernetes    ClusterIP     10.96.0.1      <none>         443/TCP          11d
redis         ClusterIP     10.103.215.93  <none>         6379/TCP         58m
```

-[중간평가] 웹 애플리케이션 배포

minikube ip로 접근이 되지 않을 때

minikube service --url <service-name> 명령어를 사용하면 된다.



web → vote:80 → redis:6379 → worker → db:5432 → result:80 ← web

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/sample 9-10월
λ kubectl apply -f wordpress.yml
deployment.apps/wordpress-mysql unchanged
service/wordpress-mysql unchanged
deployment.apps/wordpress unchanged
service/wordpress unchanged
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/sample
λ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/wordpress-74757b6ff-xts7c       1/1     Running   0           2m23s
pod/wordpress-mysql-5447bfc5b-d4pdc 1/1     Running   0           2m23s

NAME                                TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/kubernetes                  ClusterIP     10.96.0.1    <none>         443/TCP          11d
service/wordpress                  NodePort      10.99.103.148 <none>         80:30080/TCP     2m22s
service/wordpress-mysql            ClusterIP     10.102.55.173 <none>         3306/TCP         2m23s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/wordpress           1/1     1             1           2m23s
deployment.apps/wordpress-mysql     1/1     1             1           2m23s

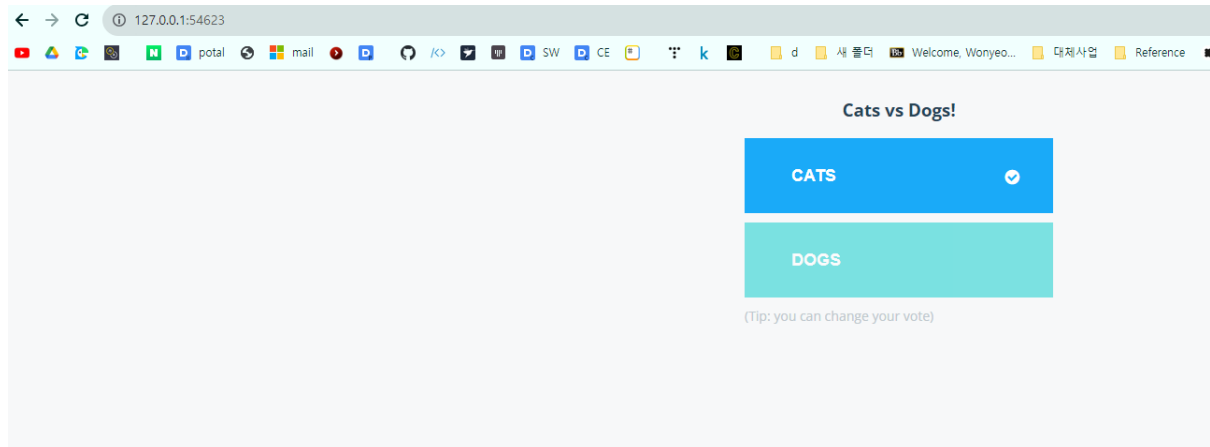
NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/wordpress-74757b6ff 1         1         1       2m23s
replicaset.apps/wordpress-mysql-5447bfc5b 1         1         1       2m23s
```

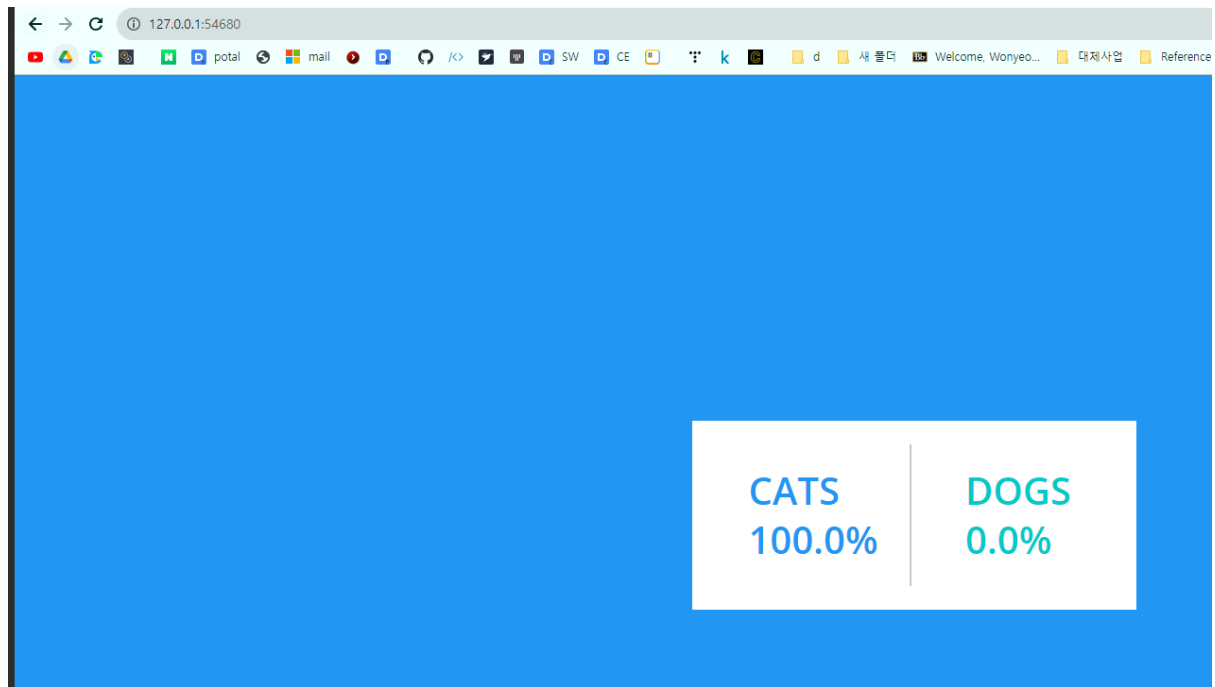
```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/sample$ kubectl apply -f vote.yml
deployment.apps/vote created
service/vote created
deployment.apps/result created
service/result created
deployment.apps/worker created
deployment.apps/redis created
service/redis created
deployment.apps/db created
service/db created
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/sample$ kubectl get all
C:\Users\leeoo\Desktop\cmdr\vendor\conemu-maximus5\..\git-for-windows\usr\bin\bash: kubectl: command not found
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/sample$ kubectl get all
NAME                                READY    STATUS              RESTARTS   AGE
pod/db-747c768488-f514b             0/1     ContainerCreating   0           18s
pod/redis-d78dc4dd7-tmrng           0/1     ContainerCreating   0           19s
pod/result-7f5fb447d8-vshjg         0/1     ContainerCreating   0           20s
pod/vote-76765b4599-9snz2           0/1     ContainerCreating   0           20s
pod/wordpress-74757b6ff-xts7c       1/1     Running             0           12m
pod/wordpress-mysql-5447bfc5b-d4pdc 1/1     Running             0           12m
pod/worker-67978cd8f7-4jljh         0/1     ContainerCreating   0           19s

NAME                                TYPE               CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/db                        ClusterIP          10.108.88.12    <none>            5432/TCP         18s
service/kubernetes                ClusterIP          10.96.0.1       <none>            443/TCP          11d
service/redis                     ClusterIP          10.97.135.166   <none>            6379/TCP         19s
service/result                    NodePort           10.99.133.60    <none>            80:31001/TCP     20s
service/vote                      NodePort           10.96.237.59    <none>            80:31000/TCP     20s
service/wordpress                 NodePort           10.99.103.148   <none>            80:30000/TCP     12m
service/wordpress-mysql           ClusterIP          10.102.55.173   <none>            3306/TCP         12m

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/db                  0/1     1              0            18s
deployment.apps/redis               0/1     1              0            19s
deployment.apps/result              0/1     1              0            20s
deployment.apps/vote                0/1     1              0            20s
deployment.apps/wordpress           1/1     1              1            12m
deployment.apps/wordpress-mysql     1/1     1              1            12m
deployment.apps/worker              0/1     1              0            19s

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/db-747c768488       1          1          0        18s
replicaset.apps/redis-d78dc4dd7     1          1          0        19s
replicaset.apps/result-7f5fb447d8   1          1          0        20s
replicaset.apps/vote-76765b4599     1          1          0        20s
replicaset.apps/wordpress-74757b6ff 1          1          1        12m
replicaset.apps/wordpress-mysql-5447bfc5b 1          1          1        12m
replicaset.apps/worker-67978cd8f7   1          1          0        19s
```





•Ingress

서로 다른 서비스에 접근하도록 포트에 여러 개의 서비스를 연결할 때 사용한다.

Ingress 만들기

- v1.echo.<IP>.sslip.io
- v2.echo.<IP>.sslip.io

minikube에 Ingress 활성화하기

```
minikube addons enable ingress

# ingress 컨트롤러 확인
kubectl -n ingress-nginx get pod

# 확인
curl -I http://192.168.64.5/healthz # minikube ip를 입력
```

echo 웹 애플리케이션 배포

v1, v2

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: echo-v1
spec:
  rules:
    - host: v1.echo.192.168.64.5.sslip.io
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
```

```

      name: echo-v1
      port:
        number: 3000
    ---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: echo-v1
spec:
  replicas: 3
  selector:
    matchLabels:
      app: echo
      tier: app
      version: v1
  template:
    metadata:
      labels:
        app: echo
        tier: app
        version: v1
    spec:
      containers:
        - name: echo
          image: ghcr.io/subicura/echo:v1
          livenessProbe:
            httpGet:
              path: /
              port: 3000
    ---
apiVersion: v1
kind: Service
metadata:
  name: echo-v1
spec:
  ports:
    - port: 3000
      protocol: TCP
  selector:
    app: echo
    tier: app
    version: v1

```

```

leeeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/ingress
λ kubectl apply -f echo-v1.yml
ingress.networking.k8s.io/echo-v1 created
deployment.apps/echo-v1 created
service/echo-v1 created
leeeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/ingress
λ kubectl get all
NAME                                READY   STATUS              RESTARTS   AGE
pod/echo-v1-7495d856f9-5hr5g        0/1     ContainerCreating   0           7s
pod/echo-v1-7495d856f9-hpft8        0/1     ContainerCreating   0           7s
pod/echo-v1-7495d856f9-pglns        0/1     ContainerCreating   0           7s

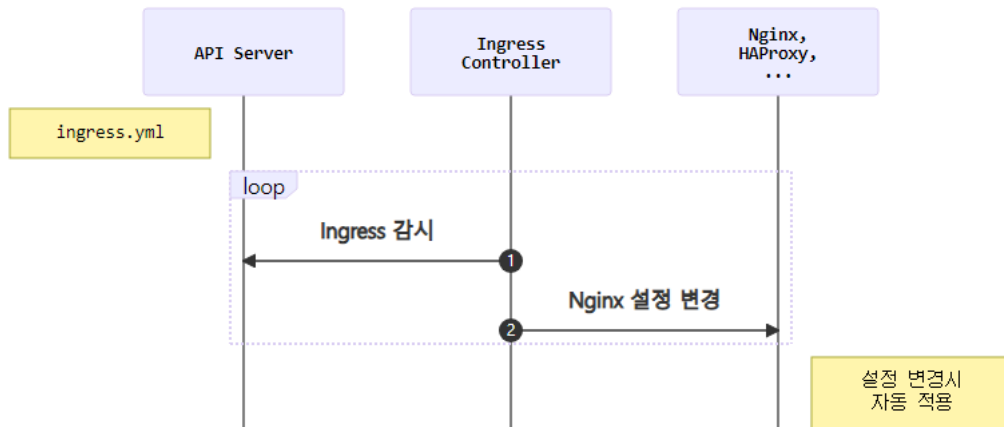
NAME                                TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/echo-v1                     ClusterIP      10.97.42.247    <none>            3000/TCP         8s
service/kubernetes                   ClusterIP      10.96.0.1       <none>            443/TCP          11d

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/echo-v1              0/3     3             0           8s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/echo-v1-7495d856f9  3         3         0       8s
leeeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/ingress
λ kubectl get ingress
NAME      CLASS    HOSTS
echo-v1   nginx    v1.echo.192.168.49.2.sslip.io

```

Ingress 생성 흐름



Volume (local)

Pod 안의 컨테이너 간 디렉토리를 공유하는 방법과 컨테이너의 특정 디렉토리를 호스트 디렉토리와 연결하는 방법

Volume 만들기

empty-dir

```

apiVersion: v1
kind: Pod
metadata:
  name: sidecar
spec:
  containers:
    - name: app
      image: busybox
      args:
        - /bin/sh
        - -c
        - >
          while true;
          do
            echo "$(date)\n" >> /var/log/example.log;
            sleep 1;
          done
      volumeMounts:
        - name: varlog
          mountPath: /var/log
    - name: sidecar
      image: busybox
      args: [/bin/sh, -c, "tail -f /var/log/example.log"]
      volumeMounts:
        - name: varlog
          mountPath: /var/log
  volumes:
    - name: varlog
      emptyDir: {}

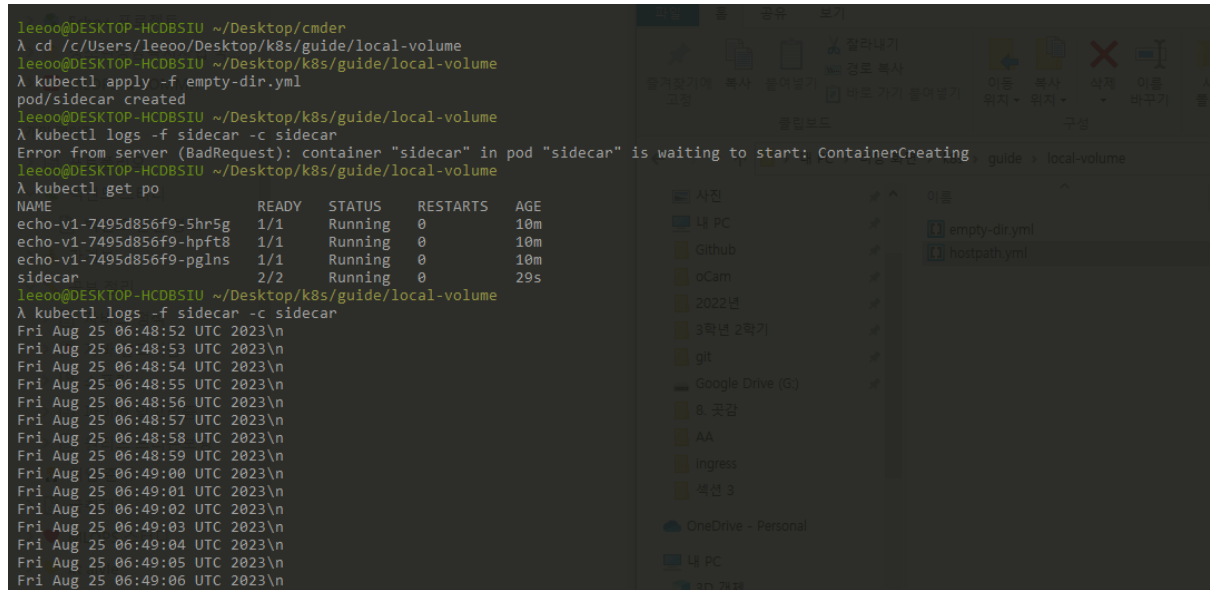
```

```

kubectl apply -f empty-dir.yml

# sidecar 로그 확인
kubectl logs -f sidecar -c sidecar

```



hostpath

```

apiVersion: v1
kind: Pod
metadata:
  name: host-log
spec:
  containers:
    - name: log
      image: busybox
      args: ["/bin/sh", "-c", "sleep infinity"]
      volumeMounts:
        - name: varlog
          mountPath: /host/var/log
  volumes:
    - name: varlog
      hostPath:
        path: /var/log

```

```
kubectl apply -f hostpath.yml
```

```

# 컨테이너 접속 후 /host/var/log 디렉토리를 확인
kubectl exec -it host-log -- sh
ls -al /host/var/log

```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/local-volume
λ kubectl apply -f hostpath.yml
pod/host-log created
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/local-volume
λ kubectl get po
NAME                                READY    STATUS    RESTARTS   AGE
echo-v1-7495d856f9-5hr5g           1/1     Running   0           11m
echo-v1-7495d856f9-hpft8           1/1     Running   0           11m
echo-v1-7495d856f9-pg1ns           1/1     Running   0           11m
host-log                            1/1     Running   0           21s
sidecar                             2/2     Running   0           108s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/local-volume
λ kubectl exec -it host-log -- sh
/ #
/ # ls -al /host/var/log
total 20
drwxr-xr-x  4 root    root          4096 Aug 13 11:00 .
drwxr-xr-x  3 root    root          4096 Aug 25 06:50 ..
-rw-r--r--  1 root    root          775 Aug 25 04:23 alternatives.log
drwxr-xr-x  2 root    root          4096 Aug 25 06:50 containers
drwxr-xr-x 21 root    root          4096 Aug 25 06:50 pods
/ # ls -al
total 52
drwxr-xr-x  1 root    root          4096 Aug 25 06:50 .
drwxr-xr-x  1 root    root          4096 Aug 25 06:50 ..
-rwxr-xr-x  1 root    root           0 Aug 25 06:50 .dockerenv
drwxr-xr-x  2 root    root        12288 Jul 17 18:30 bin
drwxr-xr-x  5 root    root          360 Aug 25 06:50 dev
drwxr-xr-x  1 root    root          4096 Aug 25 06:50 etc
drwxr-xr-x  2 nobody nobody        4096 Jul 17 18:30 home
drwxr-xr-x  3 root    root          4096 Aug 25 06:50 host
drwxr-xr-x  2 root    root          4096 Jul 17 18:30 lib
lrwxrwxrwx  1 root    root           3 Jul 17 18:30 lib64 -> lib
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/local-volume
λ kubectl apply -f hostpath.yml
# 컨테이너 접속 후 /host/var/log 디렉토리
kubectl exec -it host-log -- sh
ls -al /host/var/log
```

•ConfigMap

ConfigMap은 쿠버네티스에서 각종 설정을 관리하는 가장 좋은 방법. 실제 운영에서 자주 접하게 된다.

ConfigMap 만들기

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: prometheus
    metrics_path: /prometheus/metrics
    static_configs:
      - targets:
        - localhost:9090
```

```
# ConfigMap 생성 configmap -> cm
kubectl create cm my-config --from-file=config-file.yml

# ConfigMap 조회
kubectl get cm

# ConfigMap 내용 상세 조회
kubectl describe cm/my-config
```



```

pod --kubeconfig=/etc/kubernetes/admin.conf --context=cluster1 deleted
leeoo@DESKTOP-HCDBSTU ~/Desktop/k8s/guide/configmap
λ kubectl create cm my-config --from-file=config-file.yml
configmap/my-config created
leeoo@DESKTOP-HCDBSTU ~/Desktop/k8s/guide/configmap
λ kubectl get cm
NAME          DATA   AGE
kube-root-ca.crt  1      11d
my-config      1       4s
leeoo@DESKTOP-HCDBSTU ~/Desktop/k8s/guide/configmap
λ kubectl describe cm/my-config
Name:         my-config
Namespace:    default
Labels:       <none>
Annotations:  <none>
Data
====
config-file.yml:
---
global:
  scrape_interval: 15s
  # scrape_timeout: 10s
scrape_configs:
- job_name: prometheus
  metrics_path: /prometheus/metrics
  static_configs:
  - targets:
    - localhost:9090
BinaryData
====
Events: <none>

```

```

image: alpine
command: ["sleep"]
args: ["100000"]
env:
- name: hello
  valueFrom:
    configMapKeyRef:
      name: my-config
      key: hello

```

```

kubectl apply -f alpine-env.yml
# env 확인
kubectl exec -it alpine-env -- env

```

생성한 ConfigMap을 `/etc/config` 디렉토리에 연결

```

apiVersion: v1
kind: Pod
metadata:
  name: alpine
spec:
  containers:
    - name: alpine
      image: alpine
      command: ["sleep"]
      args: ["100000"]
      volumeMounts:
        - name: config-vol
          mountPath: /etc/config
  volumes:
    - name: config-vol
      configMap:
        name: my-config

```

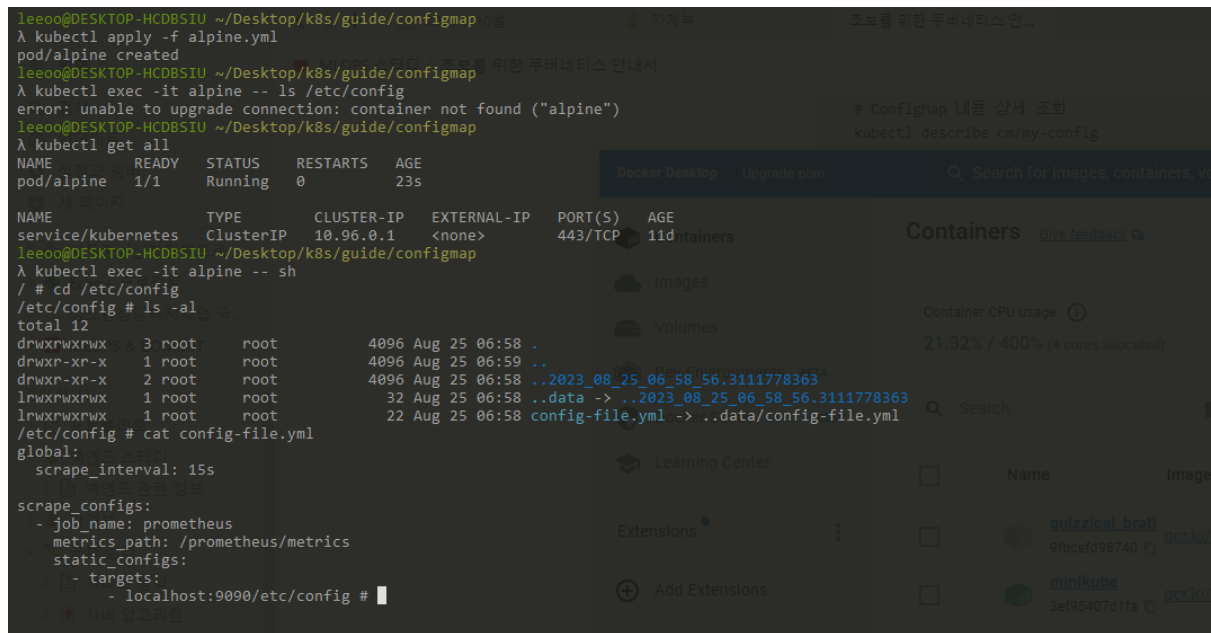
volume을 연결하여 배포하고 확인

```

kubectl apply -f alpine.yml

# 접속 후 설정 확인
kubectl exec -it alpine -- ls /etc/config
kubectl exec -it alpine -- cat /etc/config/config-file.yml

```



env 파일로 만들기

```

hello=world
haha=hoho

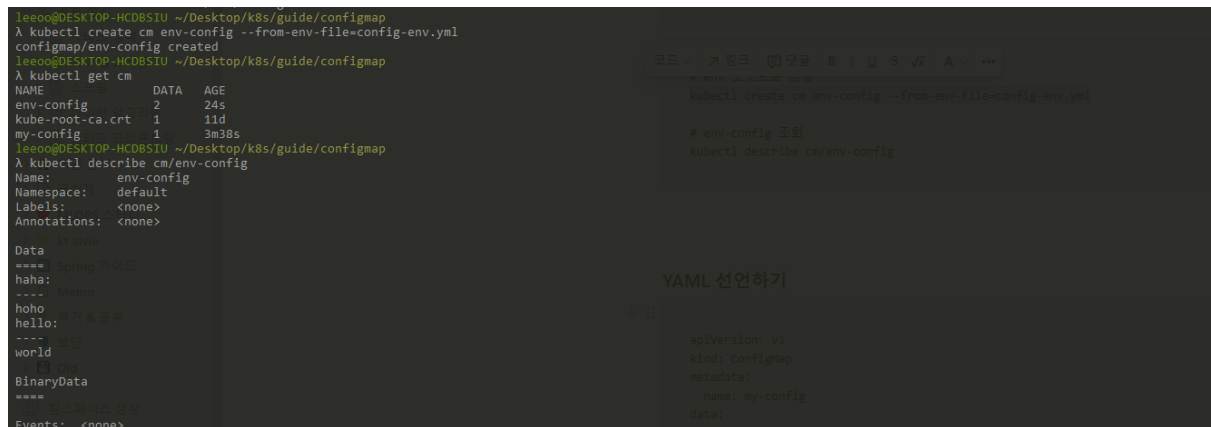
```

```

# env 포맷으로 생성
kubectl create cm env-config --from-env-file=config-env.yml

# env-config 조회
kubectl describe cm/env-config

```



YAML 선언하기

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
data:
  hello: world
  kuber: netes
  multiline: |-

```

```
first
second
third
```

`config-map.yml` 적용 후 마운트 된 내용을 확인합니다.

```
# 기존 configmap 삭제
kubectl delete cm/my-config

# configmap 생성
kubectl apply -f config-map.yml

# alpine 적용
kubectl apply -f alpine.yml

# 적용내용 확인
kubectl exec -it alpine -- cat /etc/config/multiline
```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/configmap : ~
λ kubectl delete cm/my-config
configmap "my-config" deleted
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/configmap : ~
λ kubectl apply -f config-map.yml
configmap/my-config created
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/configmap : ~
λ kubectl apply -f alpine.yml
pod/alpine configured
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/configmap : ~
λ kubectl describe cm/my-config
Name:         my-config
Namespace:    default
Labels:       <none>
Annotations:  <none>
Data
====
hello: MLOPS & COMMIT
----
world
kuber: 2021
----
netes 코도틀리요
multiline:는 스택인
----
first : 백엔드 관련 정보
second
third
BinaryData
====
Events: <none>
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/configmap : ~
```

```
kind: ConfigMap
metadata:
  name: my-config
data:
  hello: world
  kuber: netes
  multiline: |-
    first
    second
    third
```

`config-map.yml` 적용 후 마운트 된 내용을 확인합니다.

```
# 기존 configmap 삭제
kubectl delete cm/my-config

# configmap 생성
kubectl apply -f config-map.yml
```

ConfigMap을 환경변수로 사용하기

```
apiVersion: v1
kind: Pod
metadata:
  name: alpine-env
spec:
  containers:
    - name: alpine
      image: alpine
      command: ["sleep"]
      args: ["100000"]
      env:
        - name: hello
          valueFrom:
            configMapKeyRef:
              name: my-config
              key: hello
```

```
kubectl apply -f alpine-env.yml

# env 확인
kubectl exec -it alpine-env -- env
```

```

leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/configmap
λ kubectl get po
NAME          READY   STATUS    RESTARTS   AGE
alpine        1/1     Running   0           4m58s
alpine-env    1/1     Running   0           16s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/configmap
λ kubectl exec -it alpine-env -- sh
/ # env
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT=tcp://10.96.0.1:443
HOSTNAME=alpine-env
SHLVL=1
HOME=/root
hello=world
TERM=xterm
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_SERVICE_HOST=10.96.0.1
PWD=/

```

■ Secret

이름

- alpine-env.yml
- password.txt
- username.txt

```

# secret 생성
kubectl create secret generic db-user-pass --from-file=./username.txt --from-file=./password.txt

# secret 상세 조회
kubectl describe secret/db-user-pass

# -o yaml로 상세 조회
kubectl get secret/db-user-pass -o yaml

# 저장된 데이터 base64 decode
echo 'MXEydzNlNHl=' | base64 --decode

```

```

apiVersion: v1
kind: Pod
metadata:
  name: alpine-env
spec:
  containers:
    - name: alpine
      image: alpine
      command: ["sleep"]
      args: ["1000000"]
      env:
        - name: DB_USERNAME
          valueFrom:
            secretKeyRef:
              name: db-user-pass
              key: username.txt
        - name: DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: db-user-pass
              key: password.txt

```

```

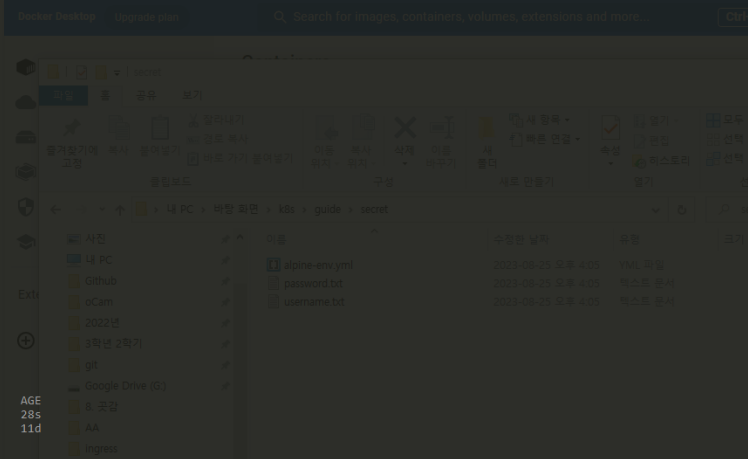
kubectl apply -f alpine-env.yml

# env 확인
kubectl exec -it alpine-env -- env

```

```
leeoo@DESKTOP-HCDBSIU ~/Desktop/cmdr
λ cd /c/Users/leeoo/Desktop/k8s/guide/secret
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/secret
λ kubectl create secret generic db-user-pass --from-file=./username.txt --from-file=./password.txt
secret/db-user-pass created
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/secret
λ kubectl describe secret/db-user-pass
Name:          db-user-pass
Namespace:     default
Labels:        <none>
Annotations:   <none>

Type: Opaque
Data:
=====
password.txt: 8 bytes
username.txt: 5 bytes
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/secret
λ kubectl get secret/db-user-pass -o yaml
apiVersion: v1
data:
  password.txt: MXYydzNlNHl=
  username.txt: YWRtaW4=
kind: Secret
metadata:
  creationTimestamp: "2023-08-25T07:08:45Z"
  name: db-user-pass
  namespace: default
  resourceVersion: "16587"
  uid: 72896a31-3e68-442d-8522-83acf70013e2
type: Opaque
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/secret
λ echo 'MXYydzNlNHl=' | base64 --decode
1q2w3e4r
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/secret
λ kubectl get secret
NAME          TYPE          DATA      AGE
db-user-pass  Opaque        2          28s
default-token-m5d5z  kubernetes.io/service-account-token  3          11d
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/secret
λ
```



```
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/secret
λ kubectl get po
NAME          READY   STATUS    RESTARTS   AGE
alpine        1/1     Running   0           11m
alpine-env    1/1     Running   0           6m54s
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/secret
λ kubectl exec -it alpine-env -- sh
/ # evn
sh: evn: not found
/ # env
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT=tcp://10.96.0.1:443
HOSTNAME=alpine-env
SHLV1=1
HOME=/root
hello=world
TERM=xterm
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_SERVICE_HOST=10.96.0.1
PWD=/
leeoo@DESKTOP-HCDBSIU ~/Desktop/k8s/guide/secret
λ kubectl apply -f alpine-env.yml
# env 확인
kubectl exec -it alpine-env -- env
```