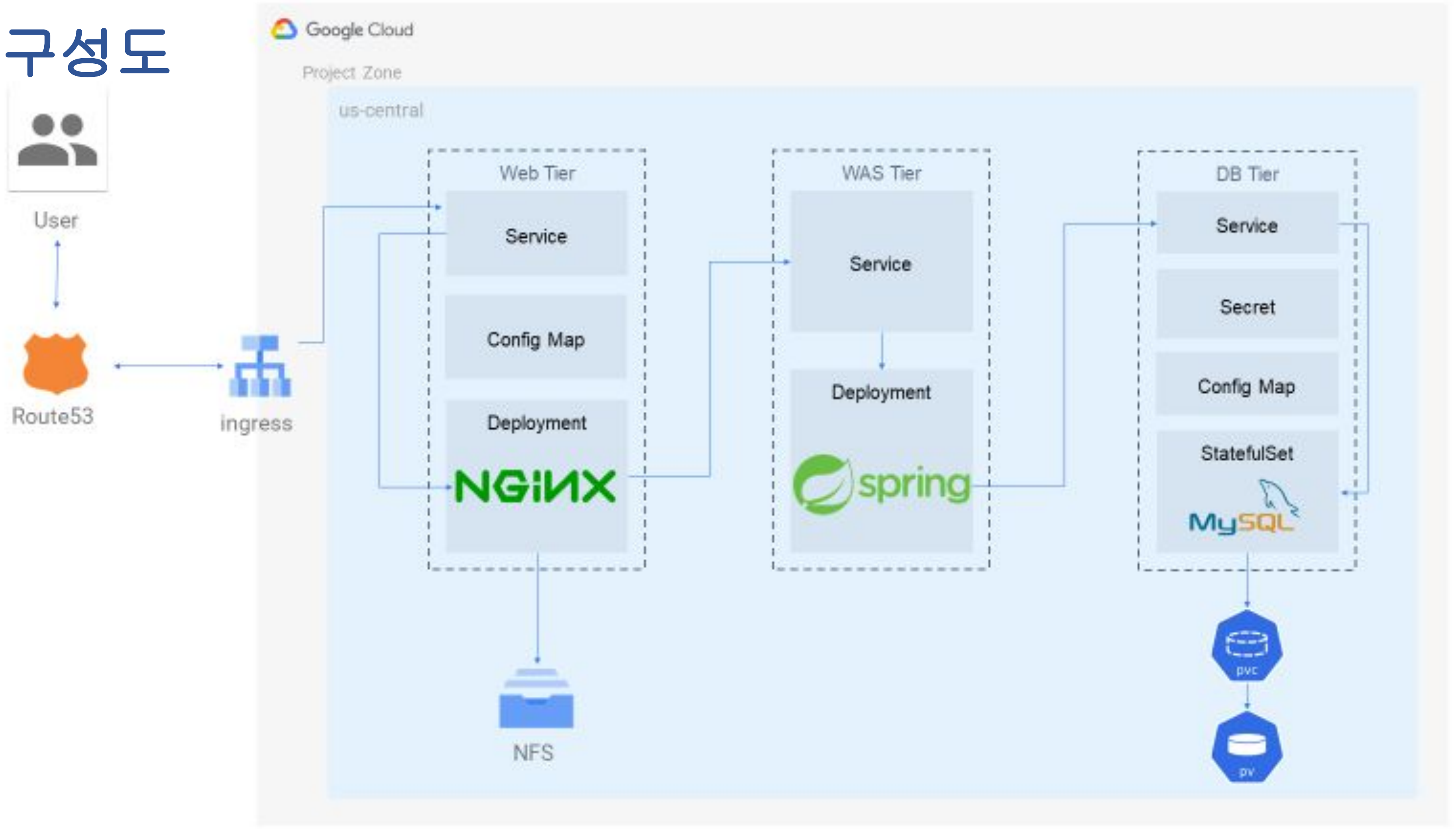
GKE 3-Tier 구성

3조
김경민 김도원 박상철
이병헌 임지원 윤상민

목차

구성도


Google Cloud
Project Zone
us-central
User
Route53
ingress
Web Tier
Service
Config Map
Deployment
NGINX
NFS
WAS Tier
Service
Deployment
spring
DB Tier
Service
Secret
Config Map
StatefulSet
MySQL
pvc
pv

선행 작업

1. VPC / VM 생성

※아래의 Spec 을 참고하여 생성하고 ssh로 접속한다.

VPC Spec

	VPC	Subnet1	Subnet2
Name	group3	nfs	bastion
Region		us-central1	us-central1
IP range		10.1.100.0/24	10.2.100.0/24
Private Google Access		on	on
Firewall rules	All check		
Dynamic routing mode	Global		

VM Spec

	NFS	Bastion
Name	nfs-server	bastion
Region	us-central1(Iowa)	us-central1(Iowa)
Machine configuration	Series: E2 Type: e2-midium	Series: E2 Type: e2-midium
Boot Disk	centOS	centOS
Access Scopes	Allow full acess to all Cloud APIs	Allow full access to all Cloud APIs
Network interfaces	VPC: group3 Subnet: nfs	VPC: group3 Subnet: bastion
External IP	None	Ephemeral

선행 작업

2. NFS 설정

0. 관리자 권한 실행

```
sudo su -
```

1. NFS 서버 설치

```
setenforce 0
sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
systemctl stop firewalld && systemctl disable firewalld
yum -y install nfs-utils
systemctl start nfs-server
systemctl start rpcbind
systemctl enable nfs-server
systemctl enable rpcbind
```

2. Directory 생성

```
mkdir /nginx
chmod -R 777 /nginx
```

3. Exports 추가

```
vi /etc/exports
 └/nginx 10.0.0.0/8(rw,sync,no_subtree_check,no_root_squash)
exportfs -a
```

4. NFS 재시작

```
systemctl restart nfs
```

선행 작업

3. Springboot 기반 WAS 이미지 building

1. Dockerfile 작성

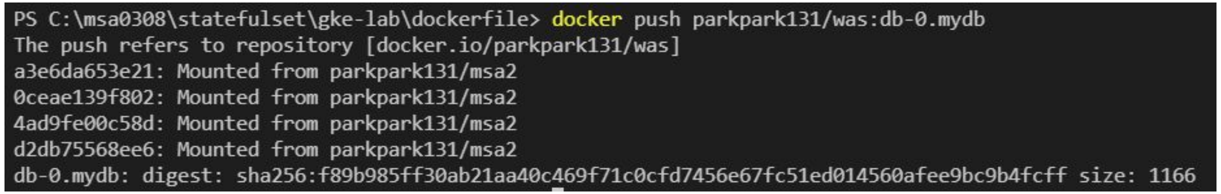
```
FROM openjdk
VOLUME /tmp
ARG JAR_FILE=*.jar
COPY ${JAR_FILE} myspringboot-0.0.1-SNAPSHOT.jar
ENTRYPOINT ["java","-jar","/myspringboot-0.0.1-SNAPSHOT.jar","--DB_IP=db-0.mydb","spring.profiles.active=prod"]
```

2. Dockerfile / myspringboot-0.0.1-SNAPSHOT.jar 이미지 빌드

```
docker build -t hub계정/was:db-0.mydb .
```

3. Image PUSH

```
docker push hub계정/was:db-0.mydb
```

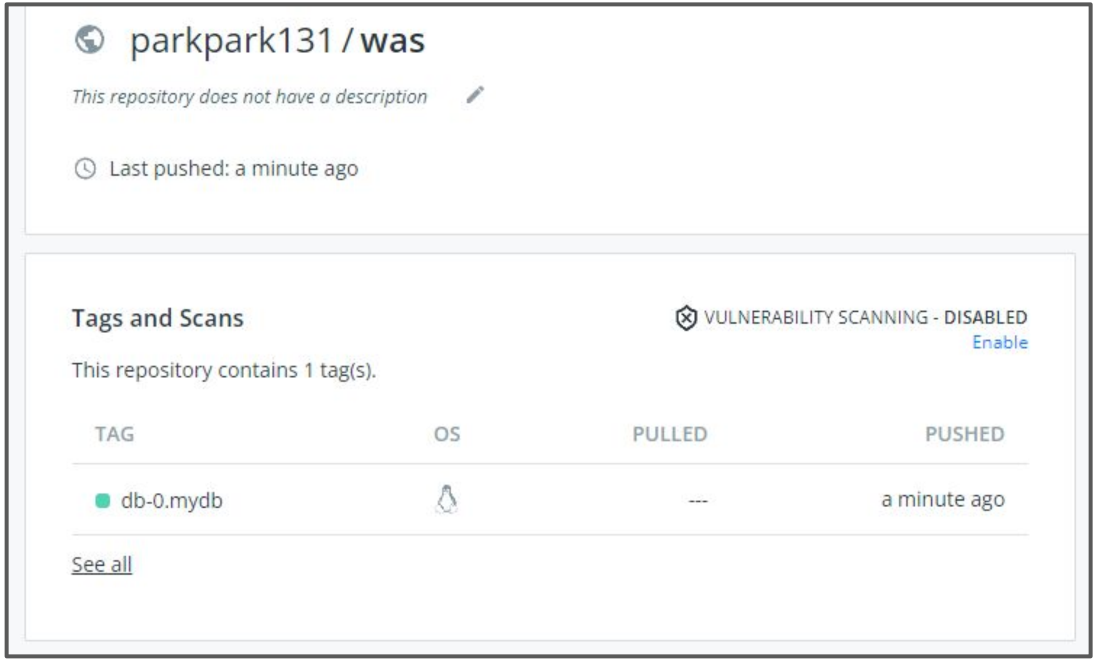


```
PS C:\msa0308\statefulset\gke-lab\dockerfile> docker push parkpark131/was:db-0.mydb
The push refers to repository [docker.io/parkpark131/was]
a3e6da653e21: Mounted from parkpark131/msa2
0ceae139f802: Mounted from parkpark131/msa2
4ad9fe00c58d: Mounted from parkpark131/msa2
d2db75568ee6: Mounted from parkpark131/msa2
db-0.mydb: digest: sha256:f89b985ff30ab21aa40c469f71c0cfd7456e67fc51ed014560afee9bc9b4fcff size: 1166
```

Image Build 완료

선행 작업

3-1. Build/Push 완료화면



parkpark131 / was

This repository does not have a description

Last pushed: a minute ago

Tags and Scans

VULNERABILITY SCANNING - DISABLED
Enable

This repository contains 1 tag(s).

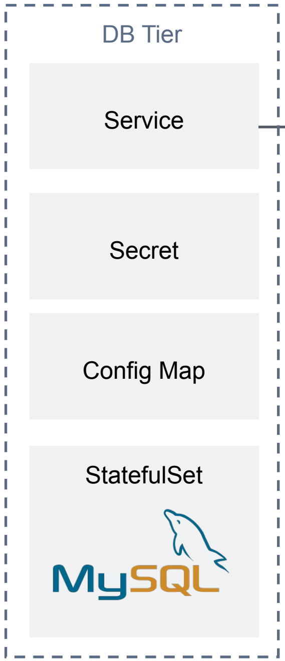
TAG	OS	PULLED	PUSHED
db-0.mydb		---	a minute ago

See all

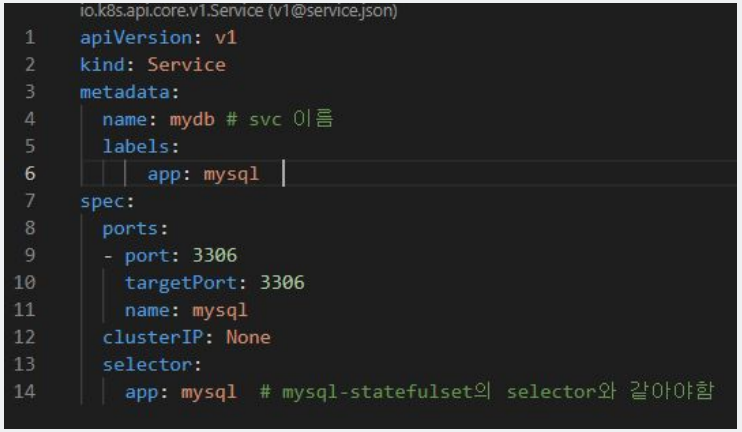
Docker Hub에 PUSH 완료

DB tier 구성하기

Service Resource



mysql-svc.yaml

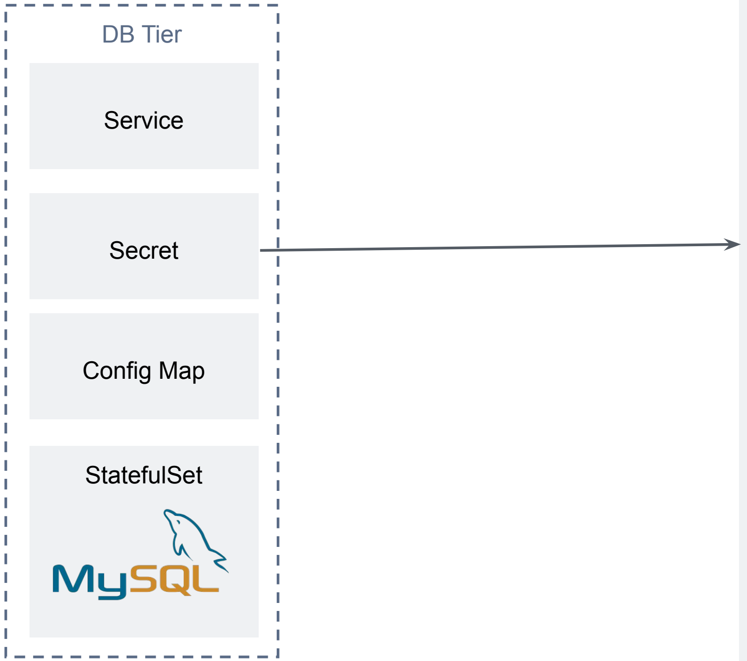


```
io.k8s.api.core.v1.Service (v1@service.json)
apiVersion: v1
kind: Service
metadata:
  name: mydb # svc 이름
  labels:
    app: mysql
spec:
  ports:
  - port: 3306
    targetPort: 3306
    name: mysql
  clusterIP: None
  selector:
    app: mysql  # mysql-statefulset의 selector와 같아야함
```

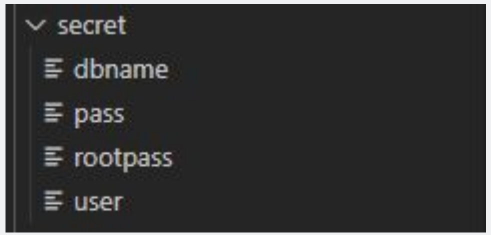
실행 명령어 :
kubectl apply -f mysql-svc.yaml

DB tier 구성하기

Secret Resource



mysql-env.yaml

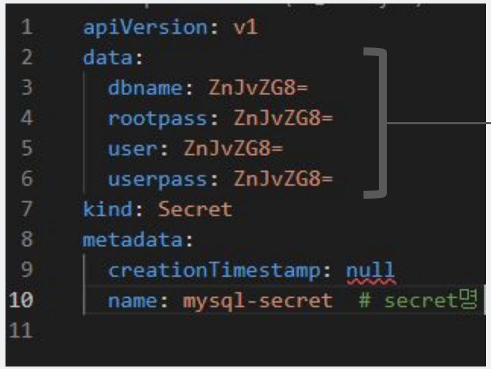


```
secret
  dbname
  pass
  rootpass
  user
```

모든 secretfile value
frodo

실행 명령어 :

kubectl.exe create secret generic **mysql-secret** --from-file ./secretfile -o yaml --dry-run=client > mysql-env.yaml



```
apiVersion: v1
data:
  dbname: ZnJvZG8=
  rootpass: ZnJvZG8=
  user: ZnJvZG8=
  userpass: ZnJvZG8=
kind: Secret
metadata:
  creationTimestamp: null
  name: mysql-secret  # secret명
```

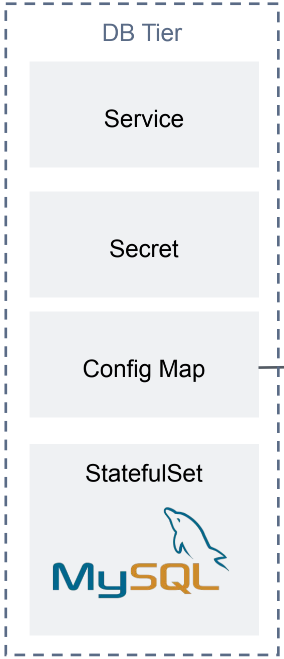
Base64
인코딩된 값

실행 명령어 :

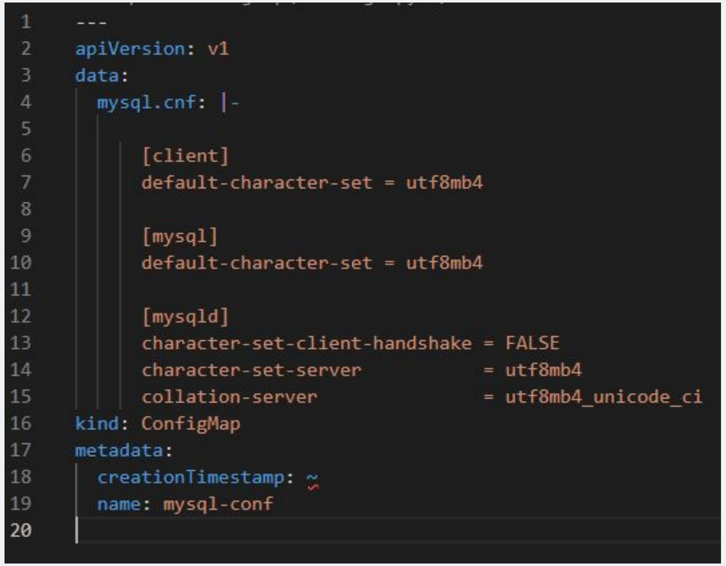
kubectl apply -f mysql-env.yaml
kubectl get secret

DB tier 구성하기

ConfigMap Resource



mysql-conf.yaml



```yaml
---
apiVersion: v1
data:
  mysql.cnf: |-

    [client]
    default-character-set = utf8mb4

    [mysql]
    default-character-set = utf8mb4

    [mysqld]
    character-set-client-handshake = FALSE
    character-set-server           = utf8mb4
    collation-server               = utf8mb4_unicode_ci
kind: ConfigMap
metadata:
  creationTimestamp: ~
  name: mysql-conf

```

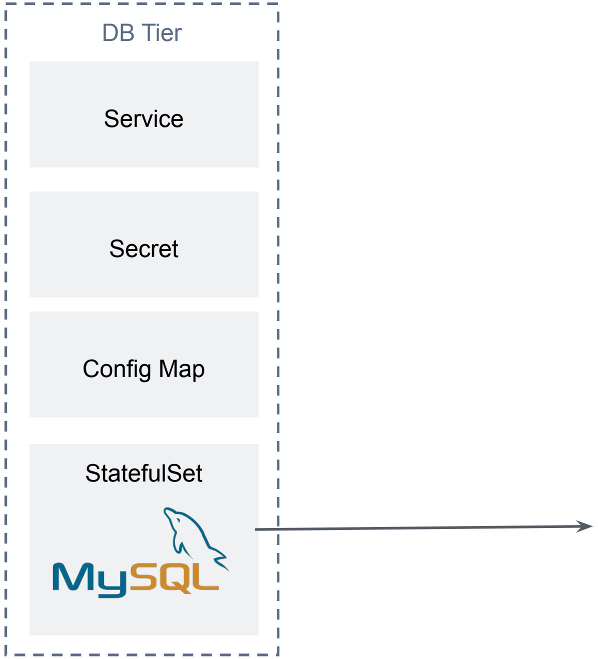
db는 한글이 지원되도록 하기 위해 mysql.cnf 파일을 만든 후 해당 cnf 파일은 config map으로 처리한다.

mysql-conf 생성 명령어 :
kubectl create configmap **mysql-conf** --from-file=**mysql.cnf** -o yaml --dry-run > mysql-conf.yaml

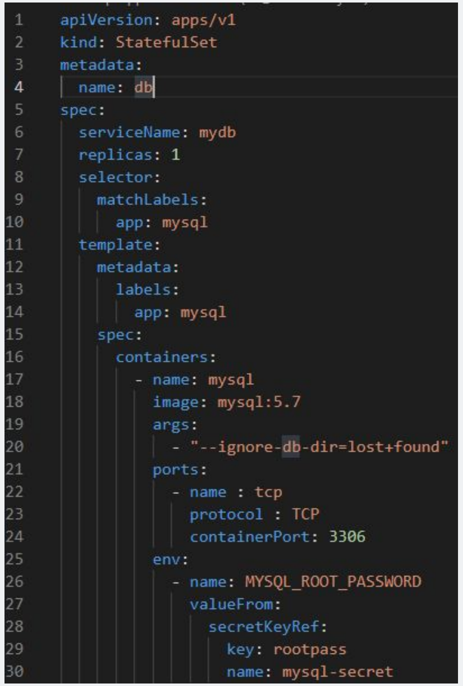
실행 명령어 : kubectl apply -f mysql-conf.yaml
kubectl get cm

DB tier 구성하기

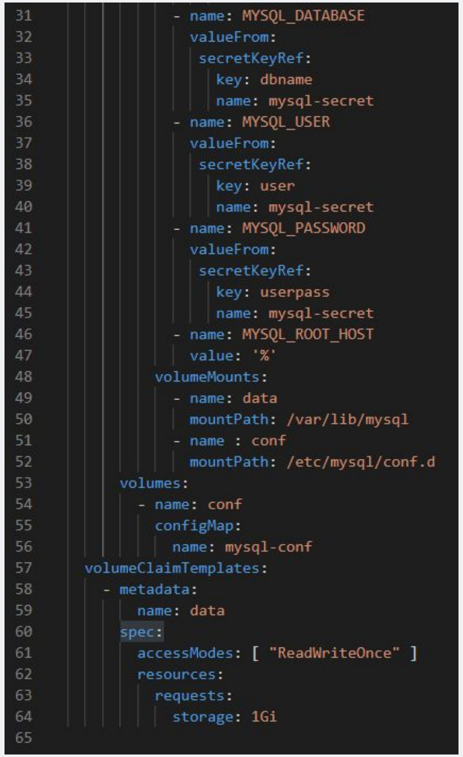
StatefulSet Resource



mysql-statefulset.yaml



```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: db
spec:
  serviceName: mydb
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: mysql:5.7
          args:
            - "--ignore-db-dir=lost+found"
          ports:
            - name : tcp
              protocol : TCP
              containerPort: 3306
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  key: rootpass
                  name: mysql-secret
```



```yaml
            - name: MYSQL_DATABASE
              valueFrom:
                secretKeyRef:
                  key: dbname
                  name: mysql-secret
            - name: MYSQL_USER
              valueFrom:
                secretKeyRef:
                  key: user
                  name: mysql-secret
            - name: MYSQL_PASSWORD
              valueFrom:
                secretKeyRef:
                  key: userpass
                  name: mysql-secret
            - name: MYSQL_ROOT_HOST
              value: '%'
          volumeMounts:
            - name: data
              mountPath: /var/lib/mysql
            - name : conf
              mountPath: /etc/mysql/conf.d
      volumes:
        - name: conf
          configMap:
            name: mysql-conf
  volumeClaimTemplates:
    - metadata:
        name: data
      spec:
        accessModes: [ "ReadWriteOnce" ]
        resources:
          requests:
            storage: 1Gi
```

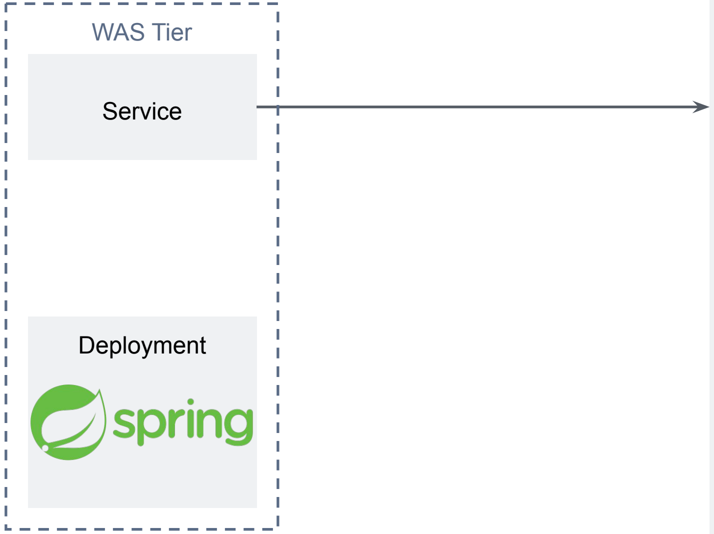
실행 명령어 : kubectl apply -f mysql-statefulset.yaml

MYSQL_ROOT_PASSWORD, MYSQL_DATABASE, MYSQL_PASSWORD, MYSQL_USER의 value는 frodo로 해당 내용은 모두 secret으로 처리한다.

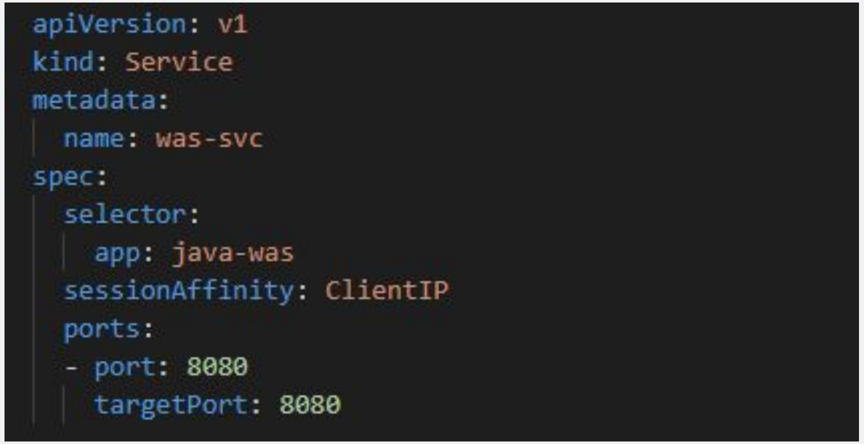
외부에서도 접속 가능하도록 MYSQL_ROOT_HOST value는 ‘%’로 처리한다.

WAS tier 구성하기

Service Resource



was-svc.yaml

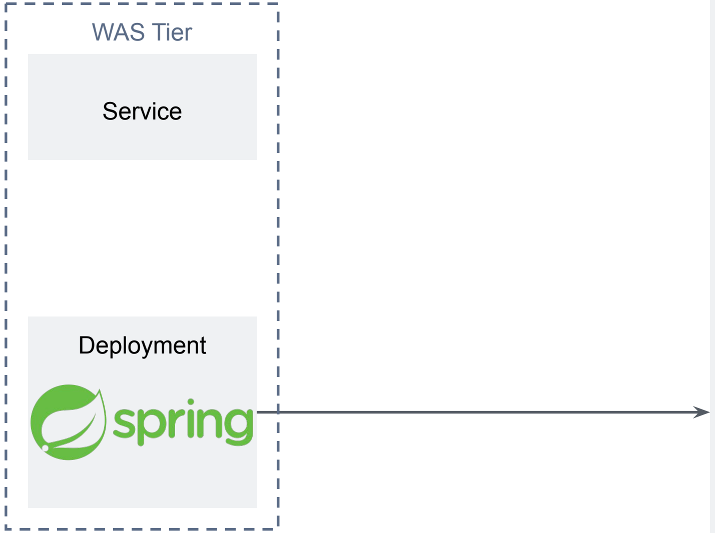


```yaml
apiVersion: v1
kind: Service
metadata:
  name: was-svc
spec:
  selector:
    app: java-was
  sessionAffinity: ClientIP
  ports:
  - port: 8080
    targetPort: 8080
```

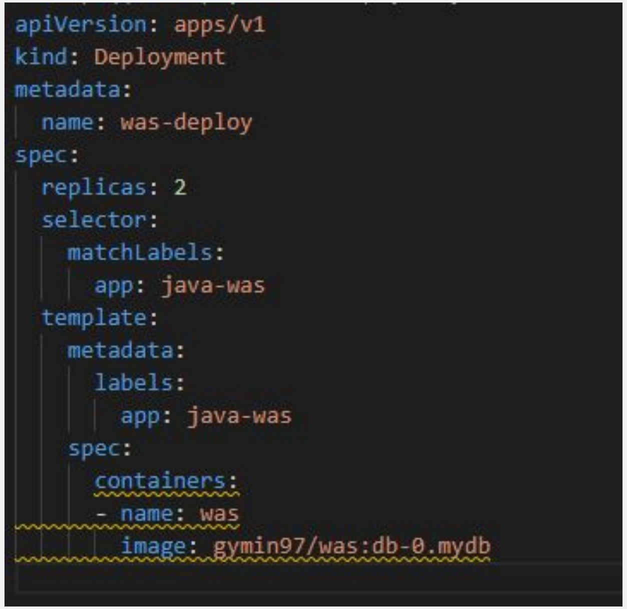
실행 명령어 :
kubectl apply -f was-svc.yaml

WAS tier 구성하기

Deployment Resource



was-deploy.yaml



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: was-deploy
spec:
  replicas: 2
  selector:
    matchLabels:
      app: java-was
  template:
    metadata:
      labels:
        app: java-was
    spec:
      containers:
      - name: was
        image: gymin97/was:db-0.mydb
```

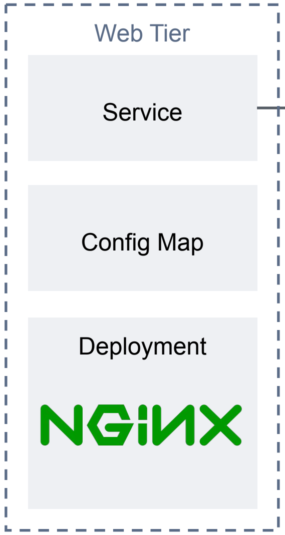
was server image → 빌드한 이미지 사용
image : ***hub계정/was:db-0.mydb***

실행 명령어 :

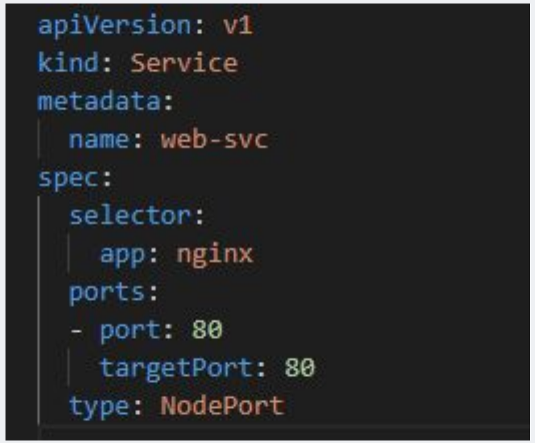
kubectl apply -f was-deploy.yaml

Web tier 구성하기

Service Resource



nginx-svc.yaml



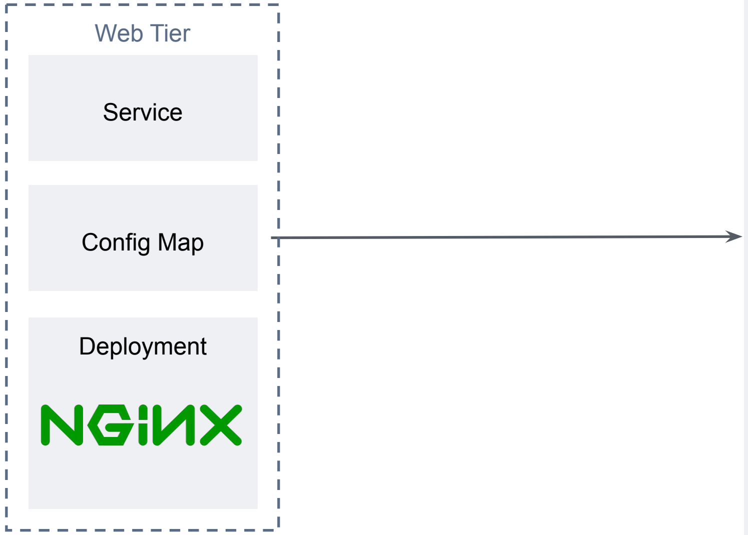
```yaml
apiVersion: v1
kind: Service
metadata:
  name: web-svc
spec:
  selector:
    app: nginx
  ports:
  - port: 80
    targetPort: 80
  type: NodePort
```

Ingress를 하기 위해 type: NodePort로 한다

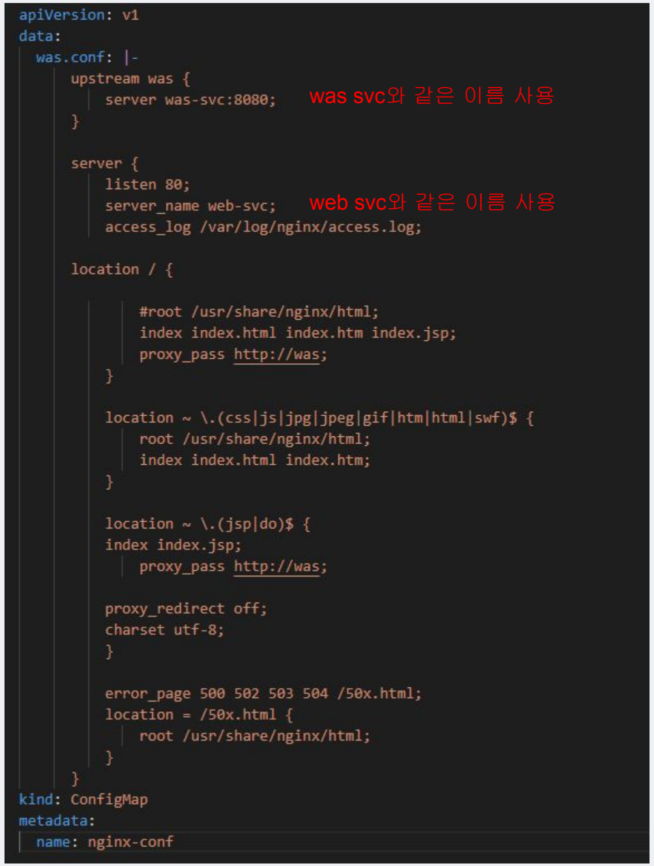
실행 명령어 :
kubectl apply -f nginx-svc.yaml

Web tier 구성하기

ConfigMap Resource



nginx-conf.yaml



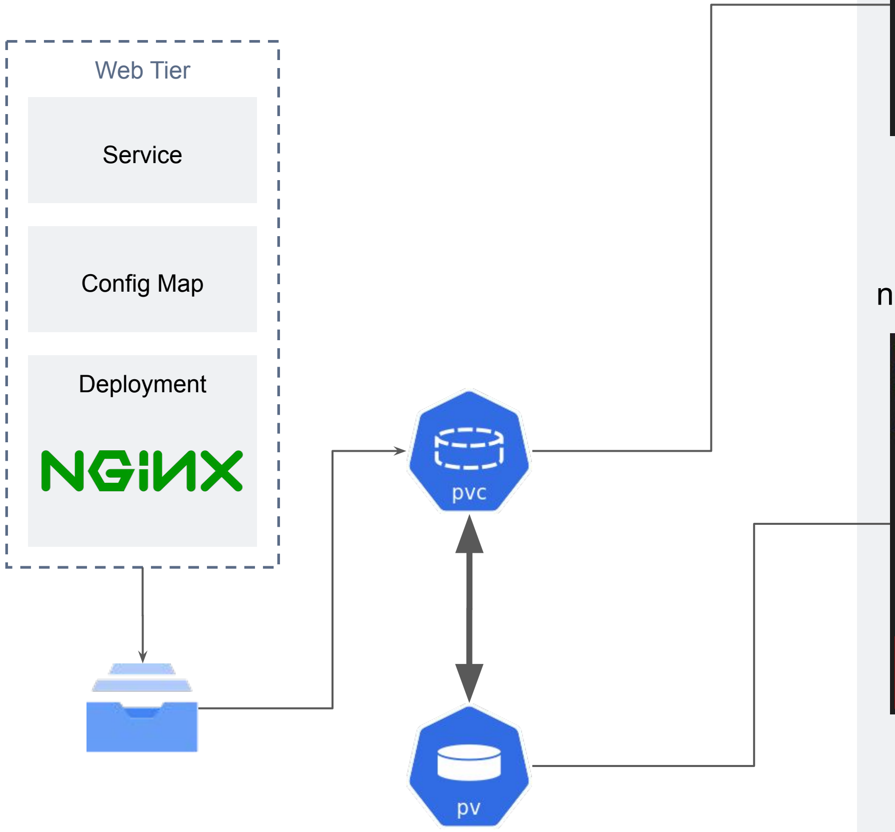
```yaml
apiVersion: v1
data:
  was.conf: |-
    upstream was {
        server was-svc:8080;
    }

    server {
        listen 80;
        server_name web-svc;
        access_log /var/log/nginx/access.log;

    location / {

            #root /usr/share/nginx/html;
            index index.html index.htm index.jsp;
            proxy_pass http://was;
        }

        location ~ \.(css|js|jpg|jpeg|gif|htm|html|swf)$ {
            root /usr/share/nginx/html;
            index index.html index.htm;
        }

        location ~ \.(jsp|do)$ {
        index index.jsp;
            proxy_pass http://was;

        proxy_redirect off;
        charset utf-8;
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
            root /usr/share/nginx/html;
        }
    }
kind: ConfigMap
metadata:
  name: nginx-conf
```

실행 명령어 :

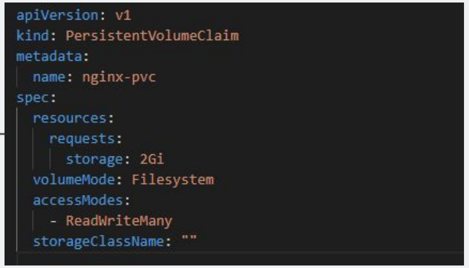
kubectl apply -f nginx-conf.yaml

Web tier 구성하기

nginx-pvc.yaml



nginx-pvc.yaml

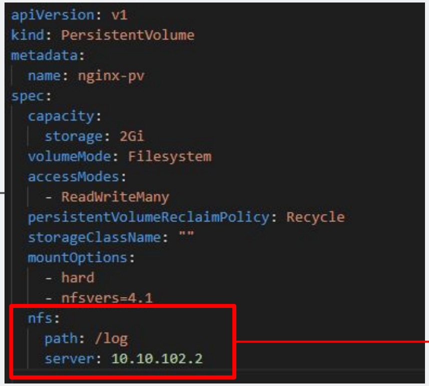


```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nginx-pvc
spec:
  resources:
    requests:
      storage: 2Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  storageClassName: ""
```

실행 명령어 :

kubectl apply -f nginx-pvc.yaml

nginx-pv.yaml



```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nginx-pv
spec:
  capacity:
    storage: 2Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: ""
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /log
    server: 10.10.102.2
```

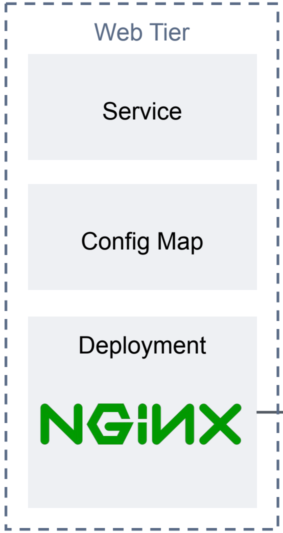
nfs 서버 정보 입력
  path: 마운트할 위치
  server: nfs 서버 ip

실행 명령어 :

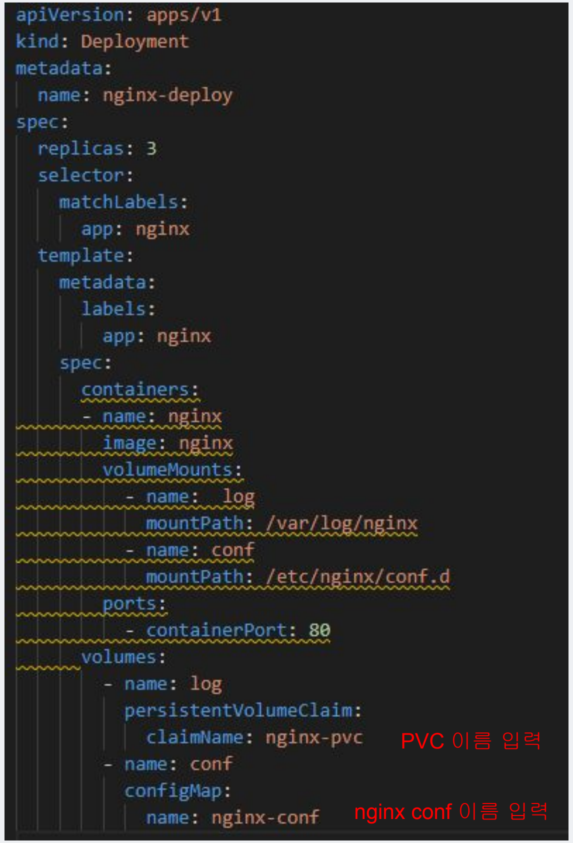
kubectl apply -f nginx-pv.yaml

Web tier 구성하기

Deployment Resource



nginx-deploy.yaml



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        volumeMounts:
          - name:  log
            mountPath: /var/log/nginx
          - name: conf
            mountPath: /etc/nginx/conf.d
        ports:
          - containerPort: 80
      volumes:
        - name: log
          persistentVolumeClaim:
            claimName: nginx-pvc
        - name: conf
          configMap:
            name: nginx-conf
```

PVC 이름 입력

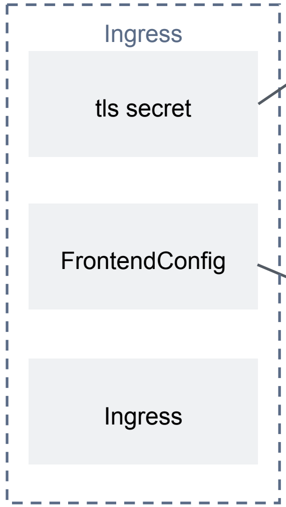
nginx conf 이름 입력

실행 명령어 :

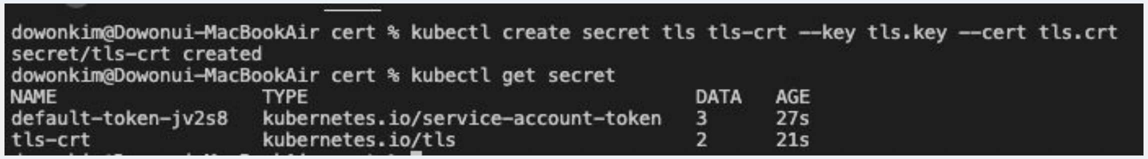
kubectl apply -f nginx-deploy.yaml

Ingress

Secret Resource



Create tls-secret



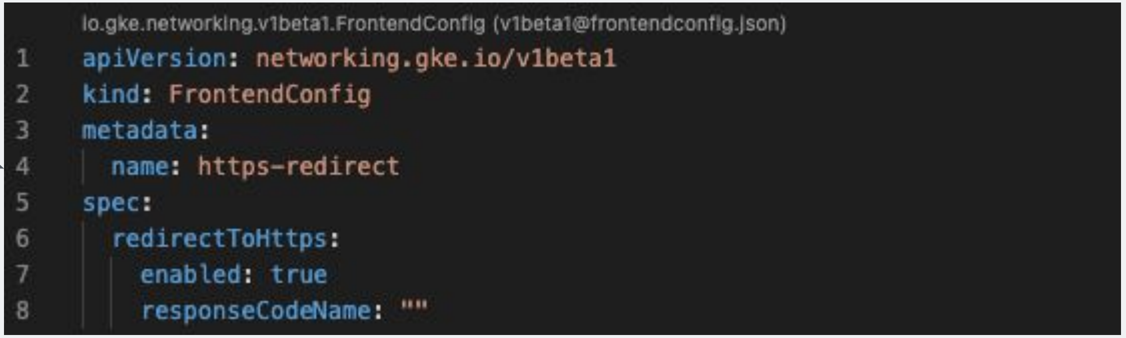
```
dowonkim@Dowonui-MacBookAir cert % kubectl create secret tls tls-crt --key tls.key --cert tls.crt
secret/tls-crt created
dowonkim@Dowonui-MacBookAir cert % kubectl get secret
NAME                  TYPE                                  DATA   AGE
default-token-jv2s8   kubernetes.io/service-account-token   3      27s
tls-crt               kubernetes.io/tls                     2      21s
```

cert가 있는 위치로 이동하여 다음 명령어 입력

실행 명령어 :

kubectl create secret tls tls-crt --key tls.key --cert tls.crt

frontendConfig.yaml



io.gke.networking.v1beta1.FrontendConfig (v1beta1@frontendconfig.json)

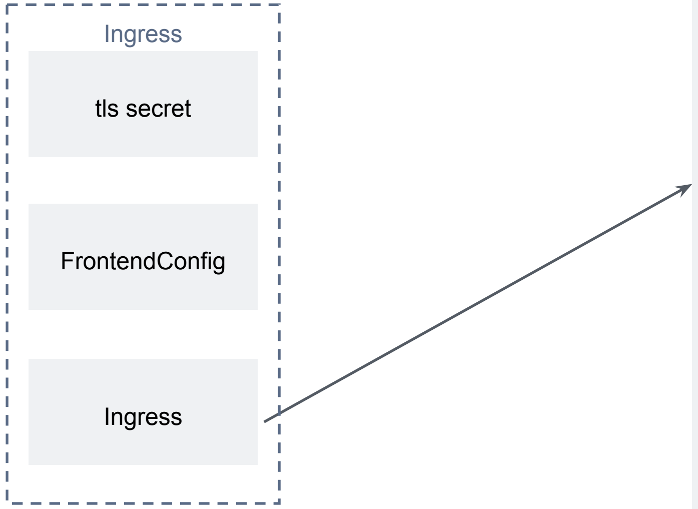
```yaml
apiVersion: networking.gke.io/v1beta1
kind: FrontendConfig
metadata:
  name: https-redirect
spec:
  redirectToHttps:
    enabled: true
    responseCodeName: ""
```

실행 명령어 :

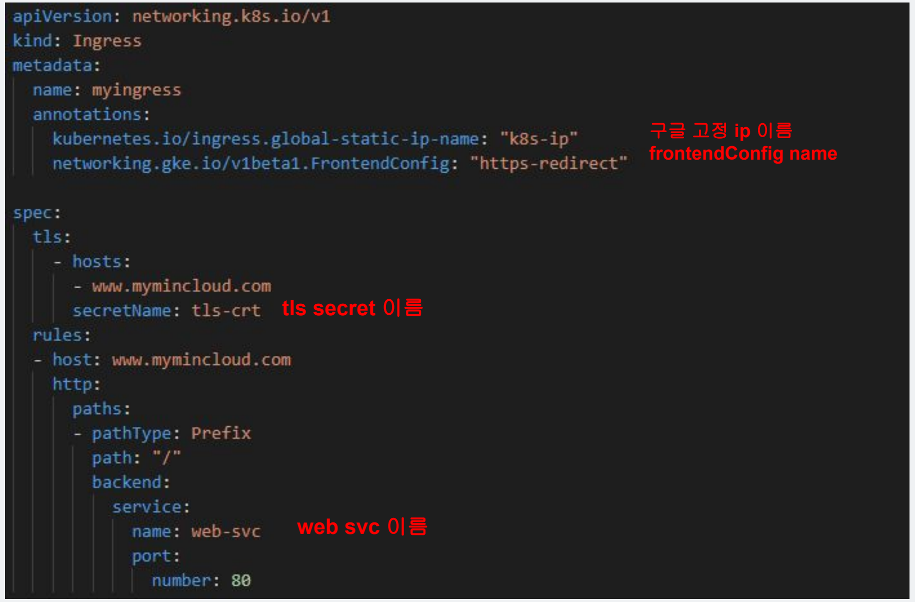
kubectl apply -f frontendConfig.yaml

Ingress

Secret Resource



ingress.yaml



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: myingress
  annotations:
    kubernetes.io/ingress.global-static-ip-name: "k8s-ip"
    networking.gke.io/v1beta1.FrontendConfig: "https-redirect"

spec:
  tls:
  - hosts:
    - www.mymincloud.com
    secretName: tls-crt
  rules:
  - host: www.mymincloud.com
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: web-svc
            port:
              number: 80
```

구글 고정 ip 이름

frontendConfig name

tls secret 이름

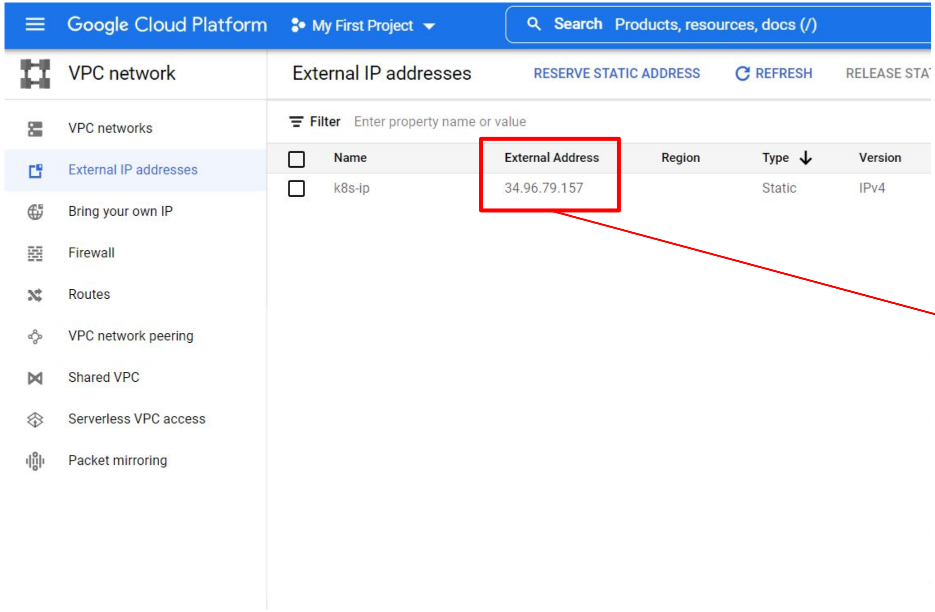
web svc 이름

실행 명령어 :

```
kubectl apply -f ingress.yaml
```

Ingress

DNS 서버 연결



Google Cloud Platform · My First Project · Search Products, resources, docs (/)

VPC network

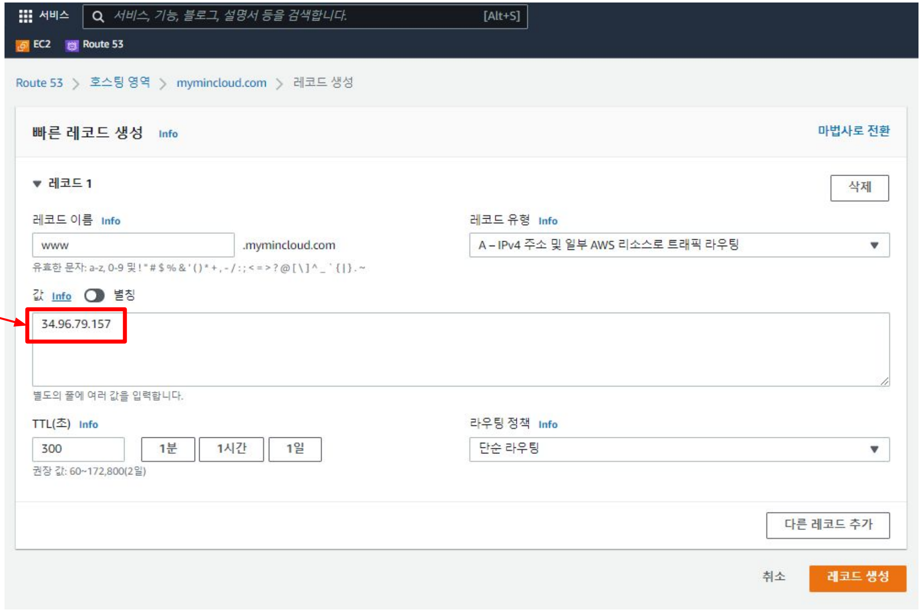
- VPC networks
- External IP addresses
- Bring your own IP
- Firewall
- Routes
- VPC network peering
- Shared VPC
- Serverless VPC access
- Packet mirroring

External IP addresses · RESERVE STATIC ADDRESS · REFRESH · RELEASE STA

Filter Enter property name or value

Name	External Address	Region	Type	Version
k8s-ip	34.96.79.157		Static	IPv4

GCP 콘솔 → VPC network → External IP 확인



서비스 · 서비스, 기능, 블로그, 설명서 등을 검색합니다. [Alt+S] · EC2 · Route 53

Route 53 > 호스팅 영역 > mymincloud.com > 레코드 생성

빠른 레코드 생성 Info · 마법사로 전환

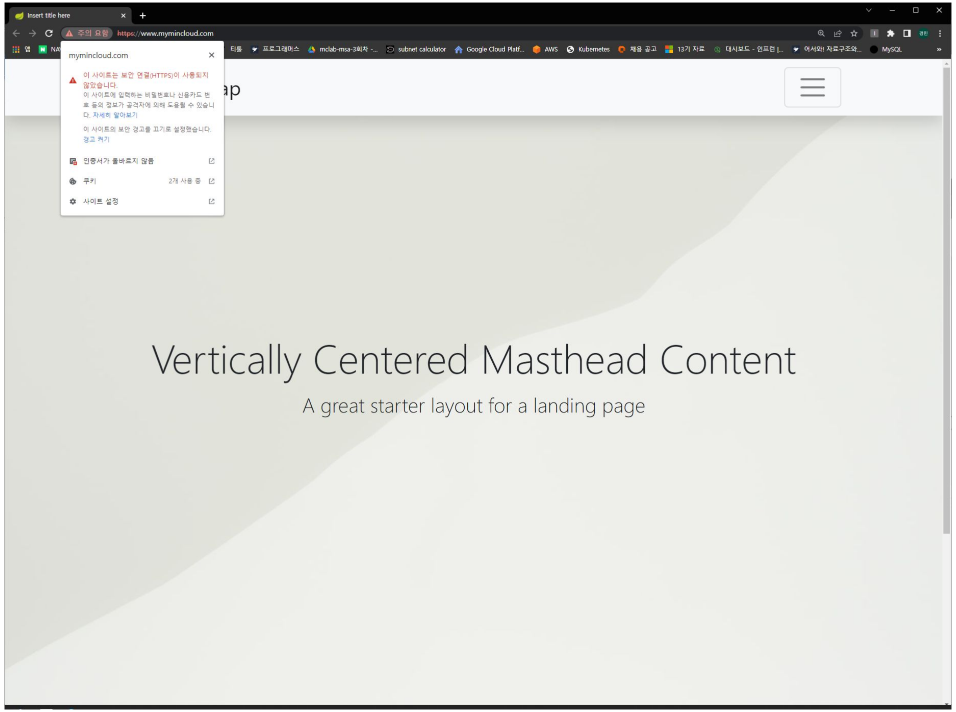
▼ 레코드 1 · 삭제

레코드 이름	www .mymincloud.com
	유효한 문자: a-z, 0-9 및 ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [\ ] ^ _ ` { | } . ~
레코드 유형	A – IPv4 주소 및 일부 AWS 리소스로 트래픽 라우팅
값 (별칭 off)	34.96.79.157
	별도의 줄에 여러 값을 입력합니다.
TTL(초)	300 (1분 / 1시간 / 1일)
	권장 값: 60~172,800(2일)
라우팅 정책	단순 라우팅

다른 레코드 추가 · 취소 · 레코드 생성

AWS 콘솔 → Route53 → 레코드 생성

접속확인



https://www.mymincloud.com

mymincloud.com

이 사이트는 보안 연결(HTTPS)이 사용되지 않았습니다.
이 사이트에 입력하는 비밀번호나 신용카드 번호 등의 정보가 공격자에 의해 도용될 수 있습니다. 자세히 알아보기

이 사이트의 보안 경고를 끄기로 설정했습니다.
경고 켜기

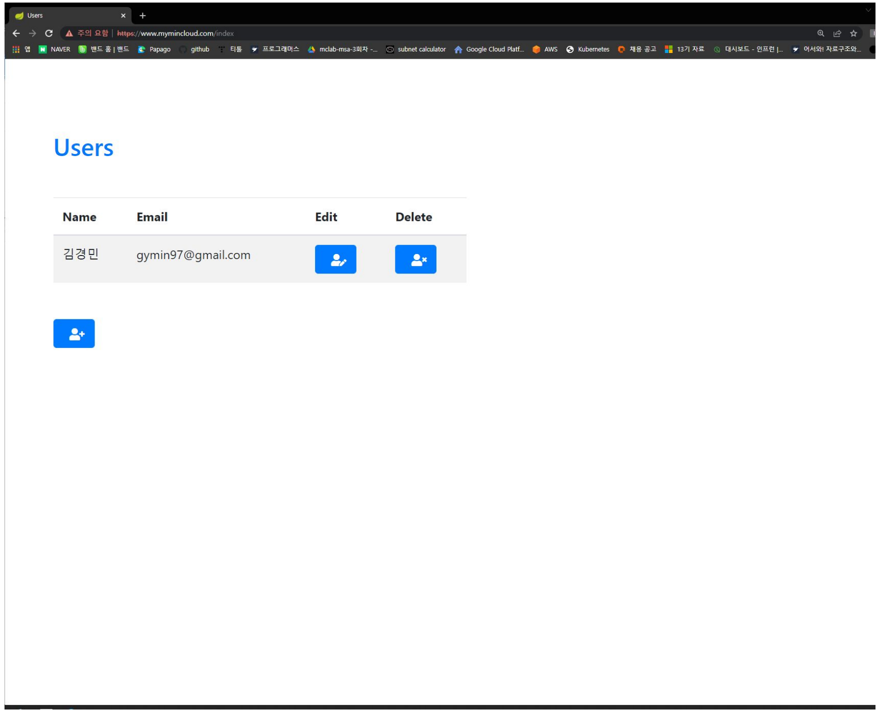
인증서가 올바르지 않음

쿠키 2개 사용 중

사이트 설정

Vertically Centered Masthead Content

A great starter layout for a landing page



https://www.mymincloud.com/index

Users

Name	Email	Edit	Delete
김경민	gymin97@gmail.com		