**GCP 2tier 만들기**

Web 단을 생략하고 바로 was로 접속하여 CICD-test 창이 열리도록한다

WAS 서버 접속시 LB를 통해 접속하도록 설정한다

LB를 제외하고 모든 서비스는 private으로 만들어 외부접속이 불가능하게 한다

Autoscale 가능하도록 구성하고, 세션처리를 위해 redis 서버를 만들어 사용한다.

DB는 sql 서비스의 mysql 5.7을 사용한다

방화벽은 전부 다 열지 않고 필요한경우에 firewall rule을 추가한다

# VPC생성

web-svc VPC 생성

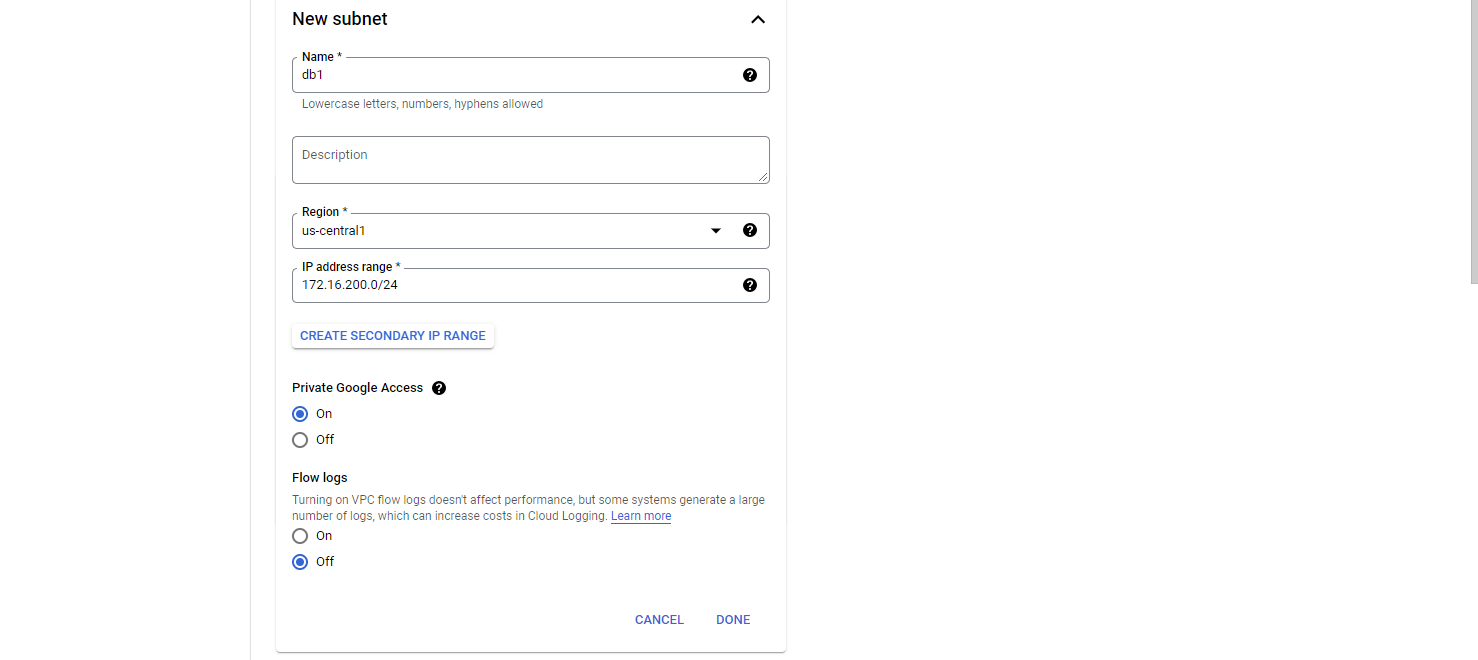
## **subnet**

Name: DB

Region: us-central1

IP address range: 172.16.200.0/24

Private Google Access: On

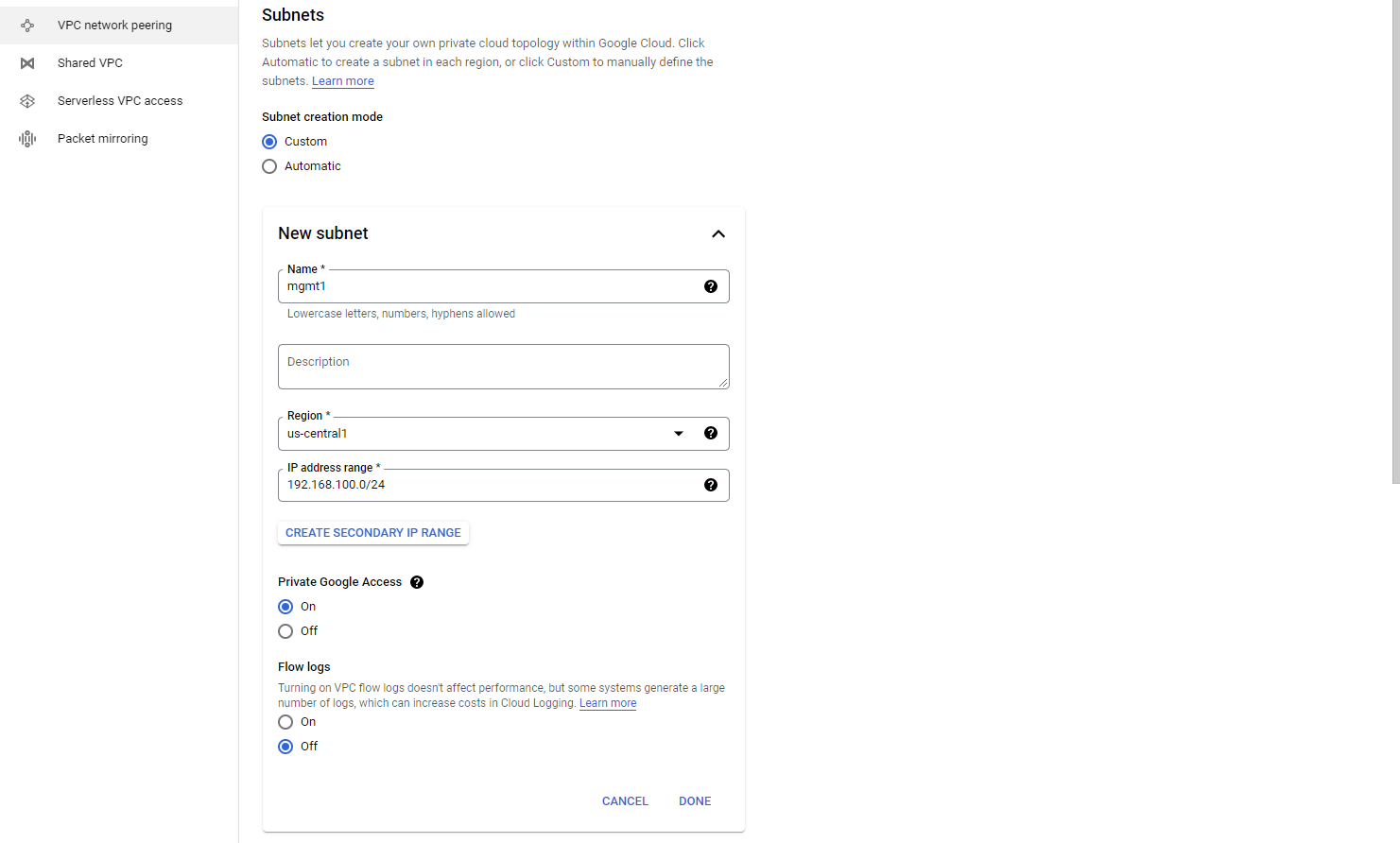


Name: mgmt

Region: us-central1

IP address range: 192.168.100.0/24

Private Google Access: On

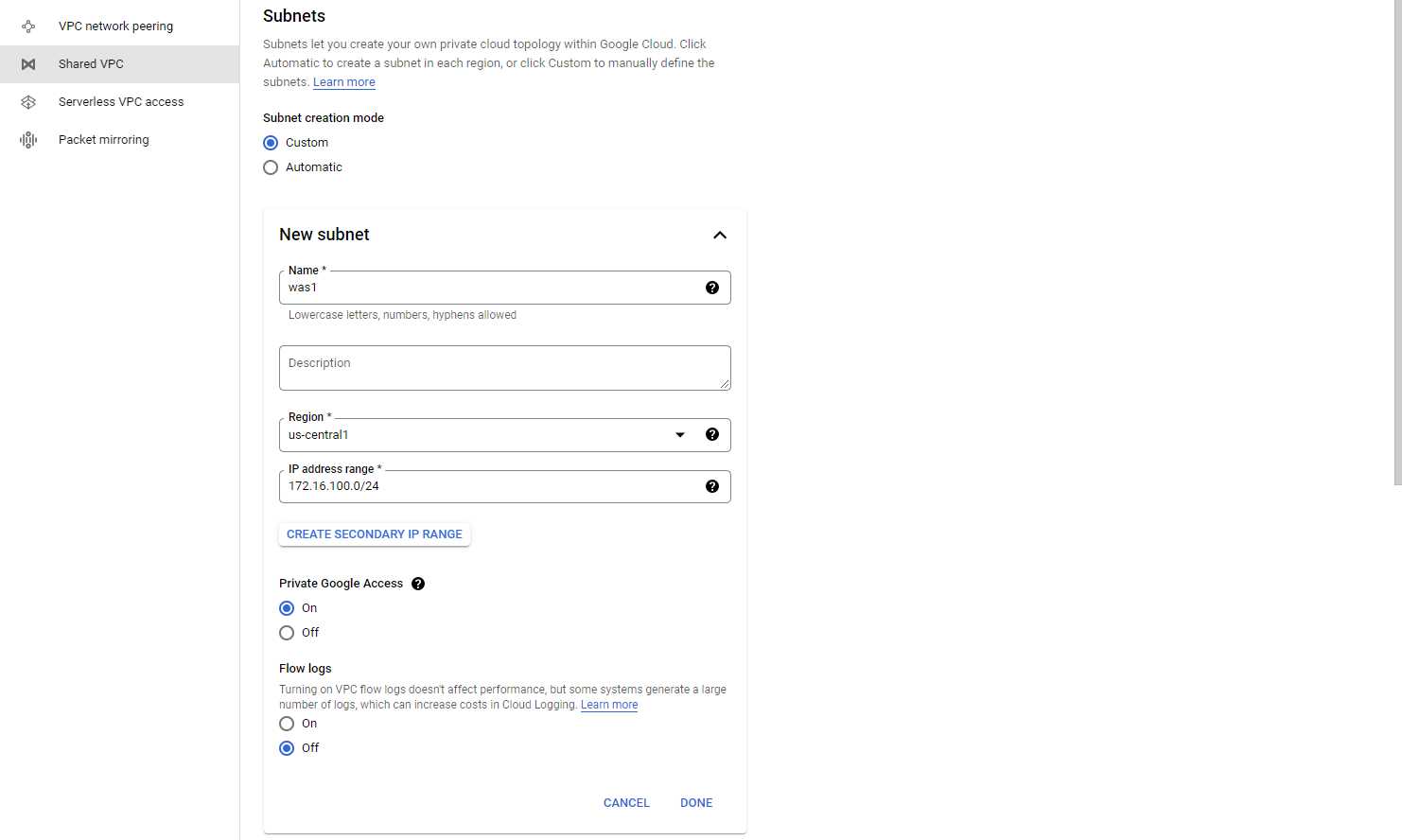


Name: was

Region: us-central1

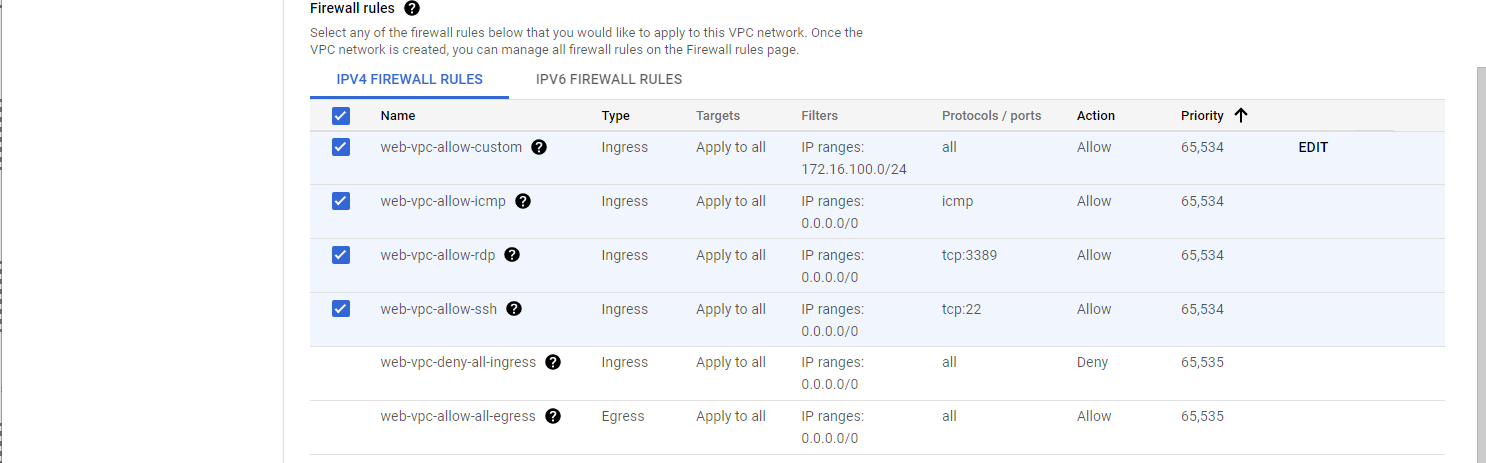
IP address range: 172.16.100.0/24

Private Google Access: On

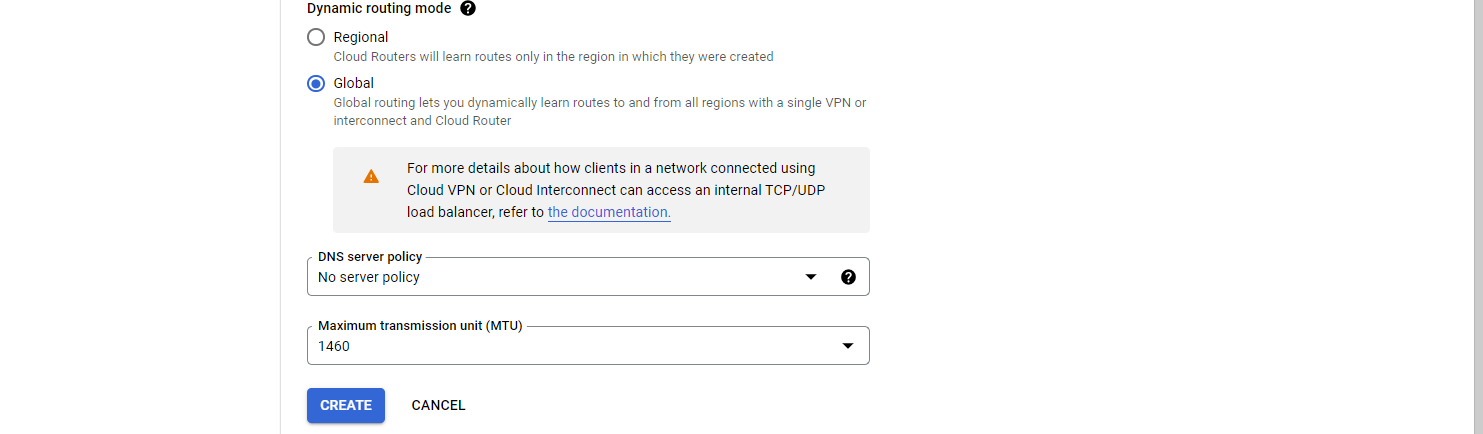


sql 서비스를 사용할 예정이라 DB 서브넷을 사용하지는 않지만 DB 서버를 만든다면 이렇게 따로 빼놓는게 좋음

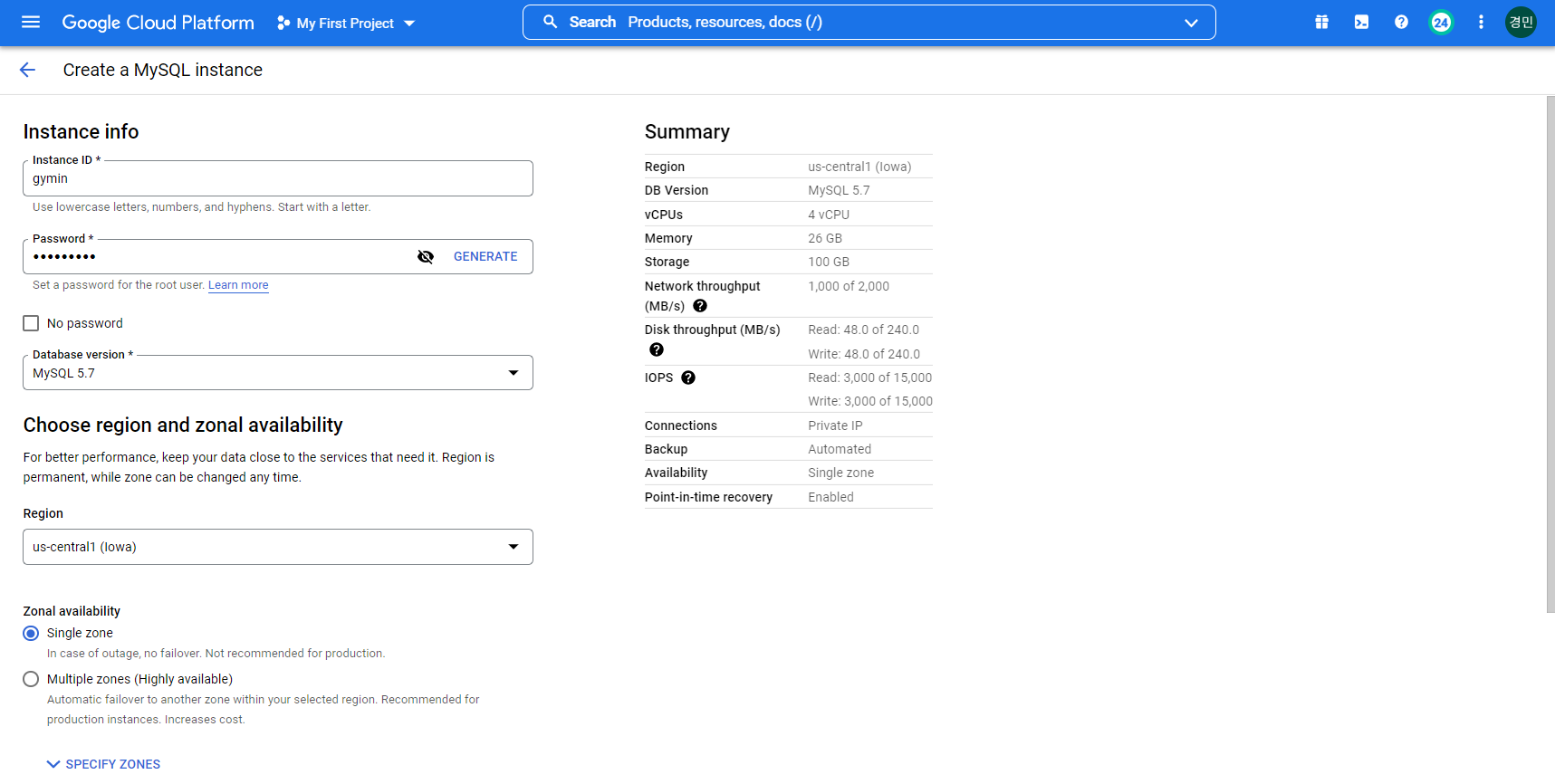
## Firewall rules

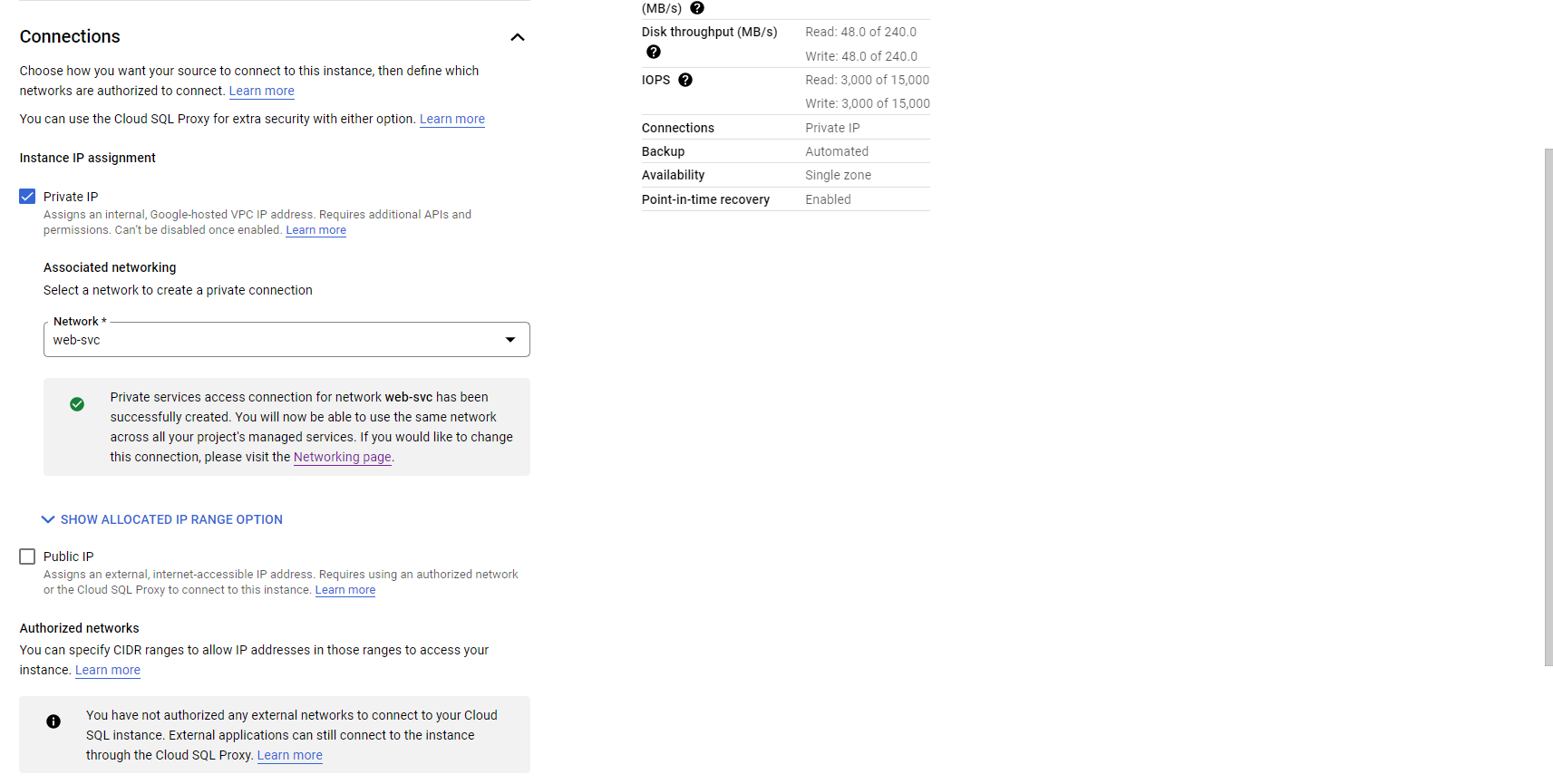


## Dynamic routing mode

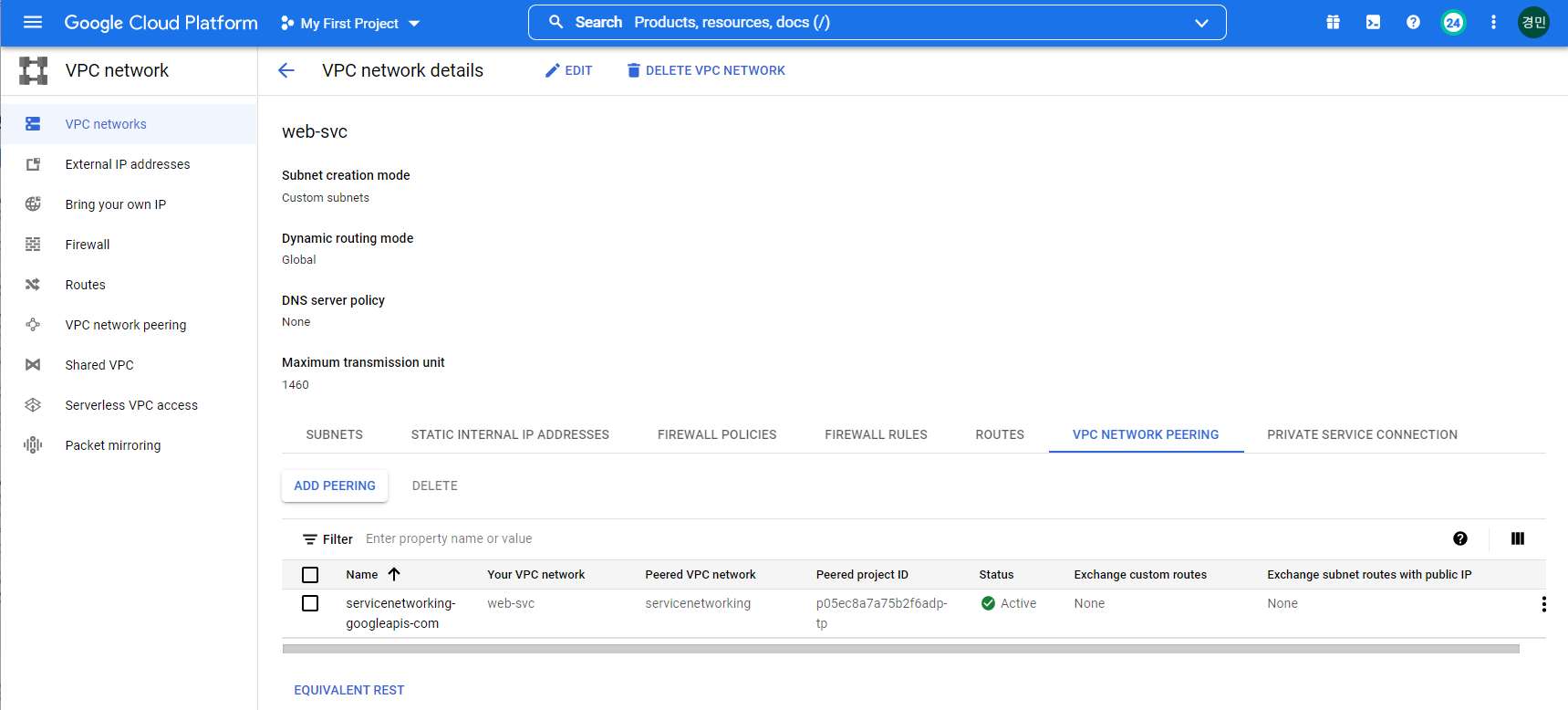


# SQL





* Ip release가 안된다면 VPC들어가서 private service network release

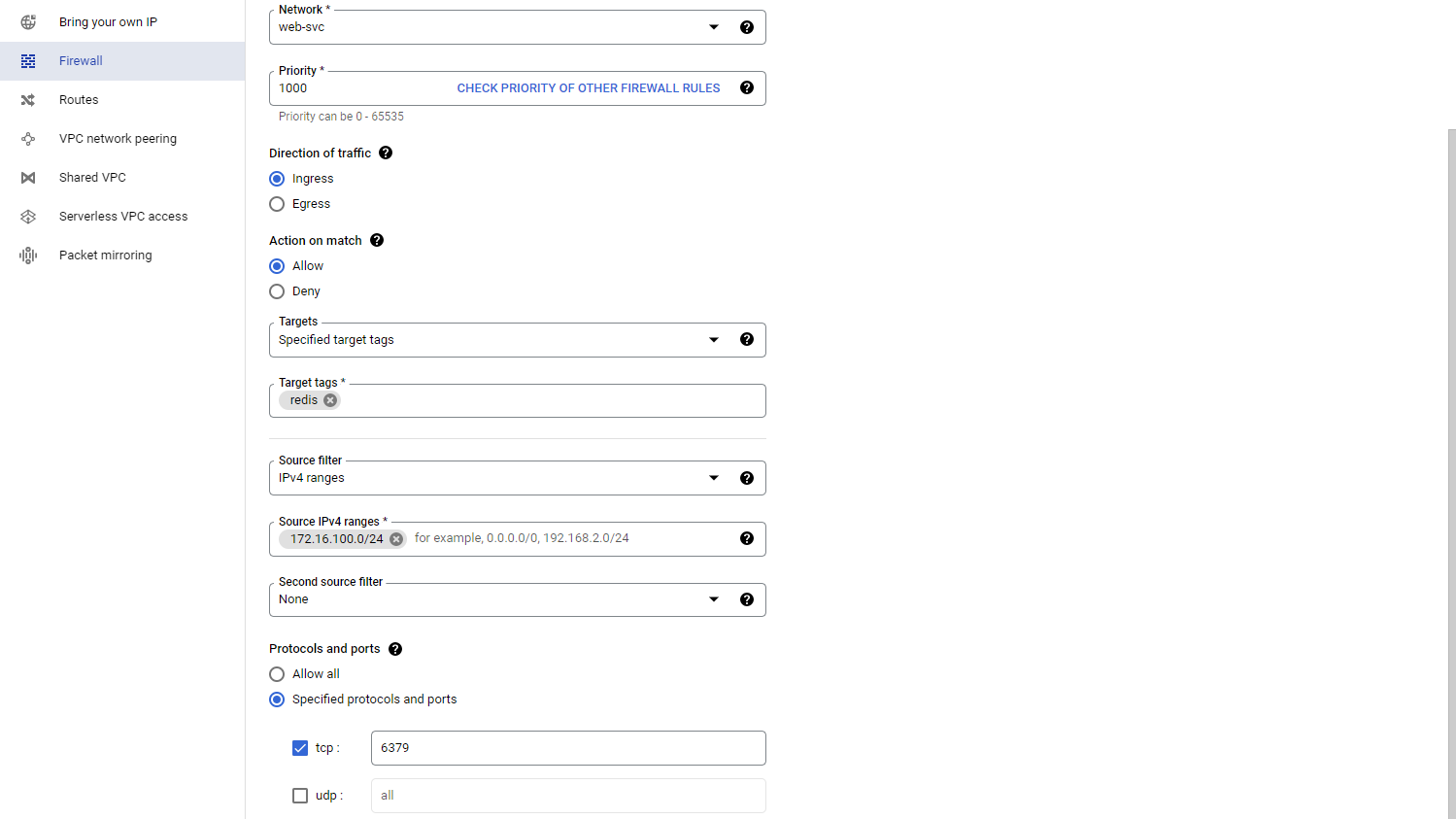


# Redis 서버

## 방화벽 열기

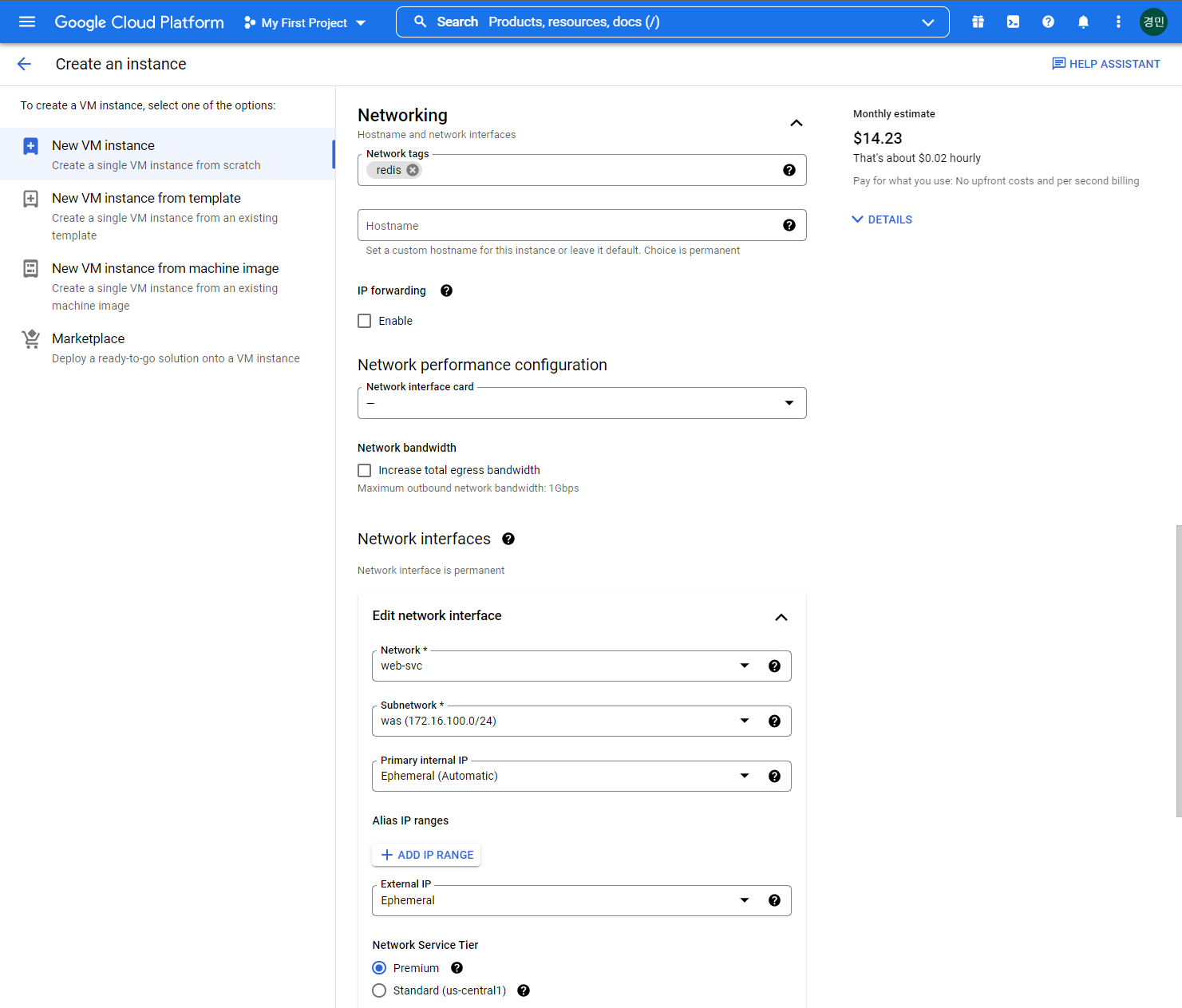
Create a firewall rule: tcp-6379

target - redis : redis 태그가 달린 서버에 이 rule을 적용



## VM 생성

* VPC 선택: web-svc
* Subnet: was
* Network tag: was



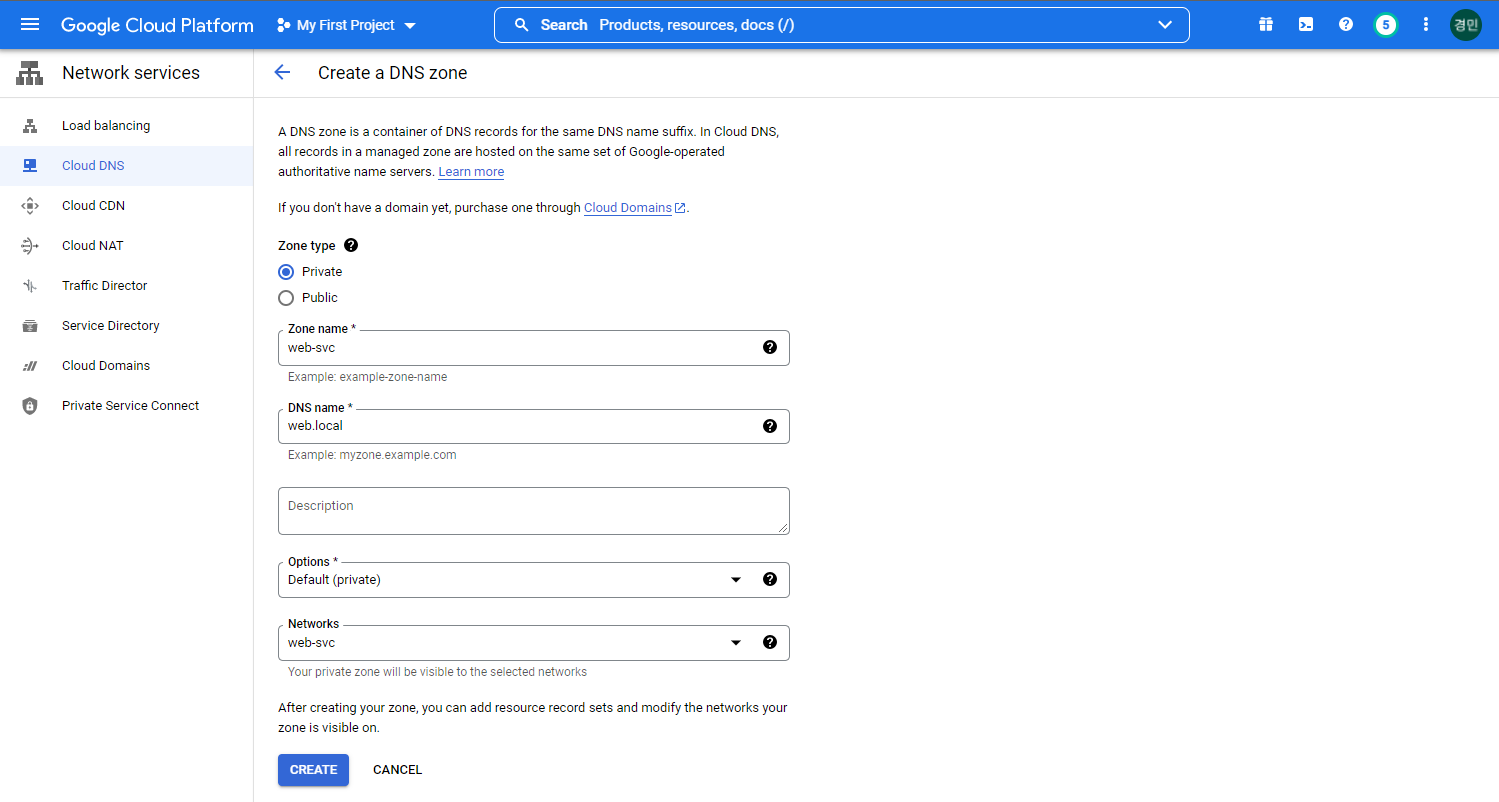
## redis 설치

|  |
| --- |
| yum install epel-release  yum install <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>  yum install -y redisfirewall-cmd --permanent --zone=public --add-port=6379/tcp  firewall-cmd --reloadfirewall-cmd --list-all  systemctl enable redis  systemctl start redis  systemctl status redisvi /etc/redis.conf  bind 0.0.0.0 ==> 수정  requirepass frodo5020!!  systemctl restart redis |

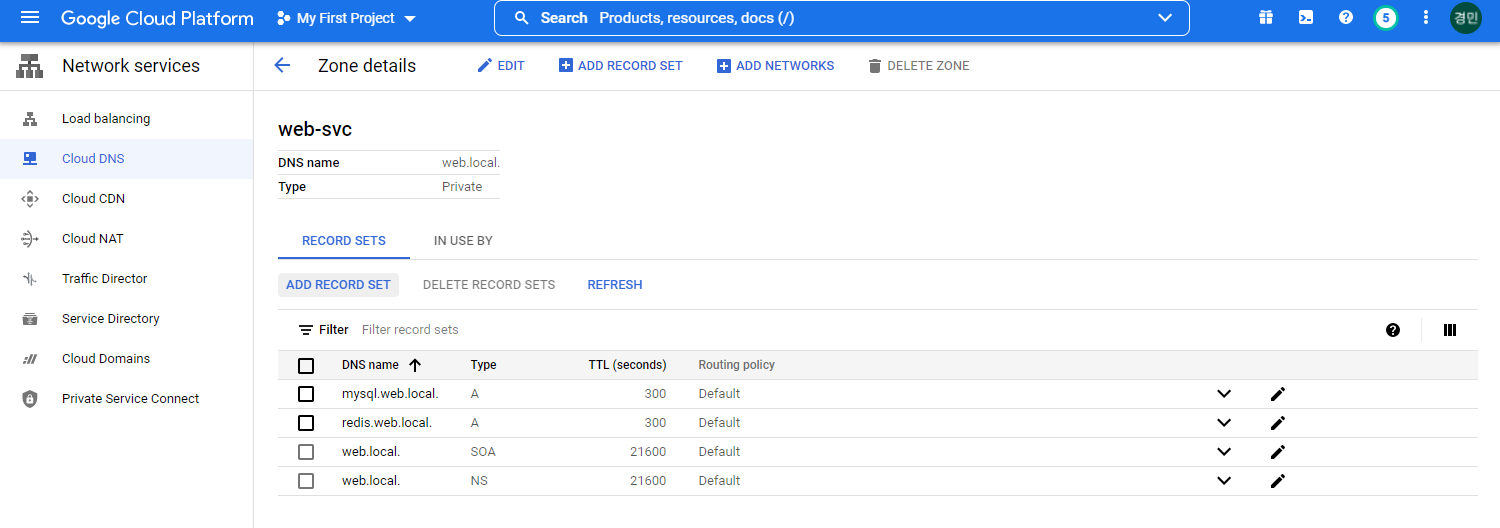
# Cloud DNS

같은 VPC내에서만 사용할 수 있는 DNS를 만들어 관리할 수 있다

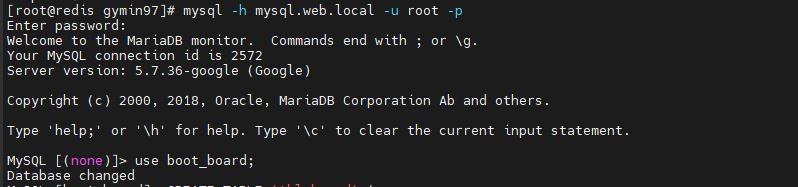
Network Service > Cloud DNS - Create Zone



2개 레코드 생성



해당 DNS로 sql 접속이 되는지 테스트

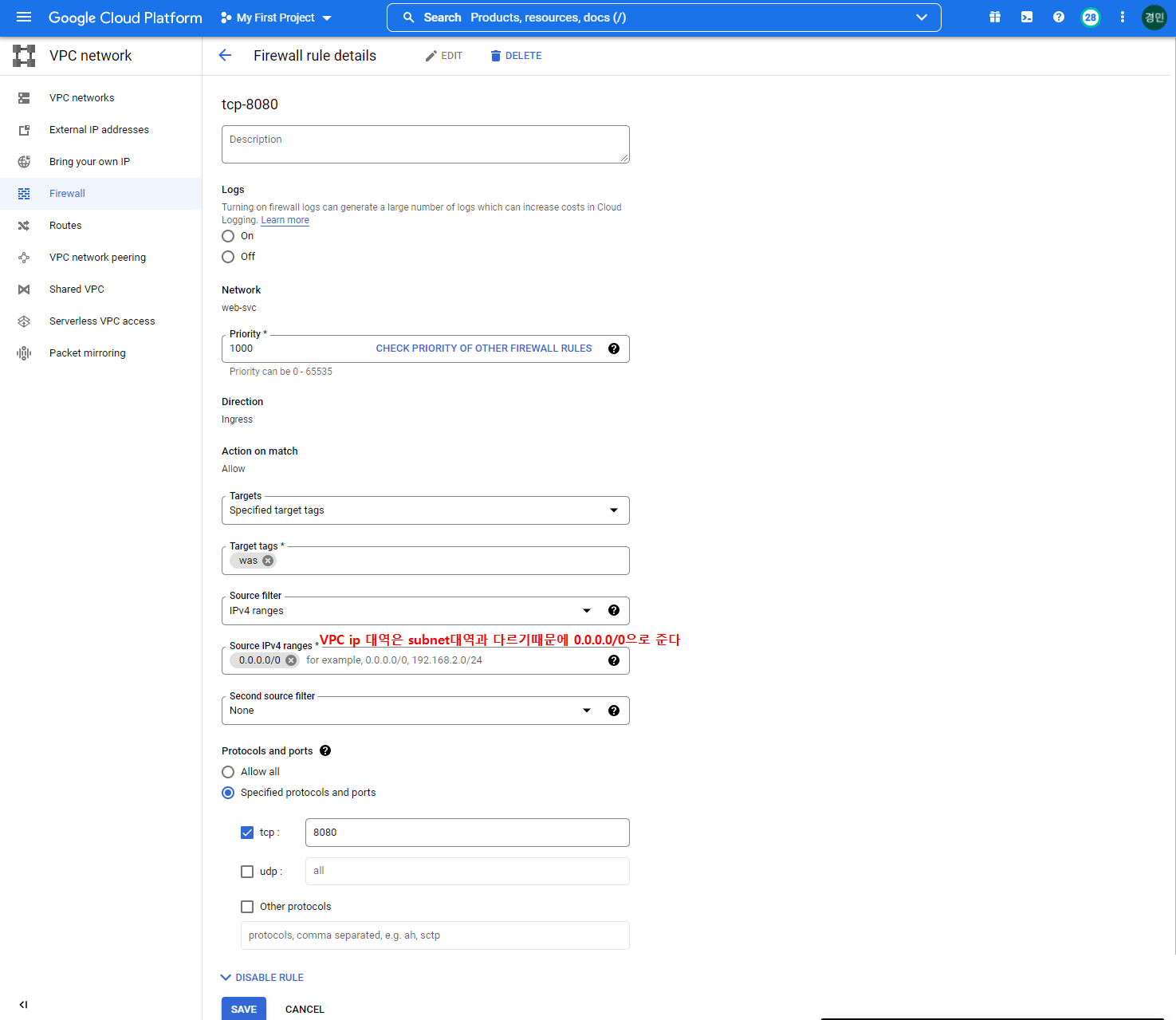


클라우드 vm들은 ip가 바뀔 가능성이 얼마든지 있다

--> 그럴때 마다 소스코드를 수정할 수 없기때문에 dns처리를 하고 dns 설정을 바꾸는것이 좋다

# WAS 서버 만들기

## 방화벽 열기



## Tomcat image 만들기

### **1. JDK, tomcat 설치**

VM 생성 후 접속하여 다음 명령어 입력

<https://gammistory.tistory.com/191>

### **2. war file build**

|  |
| --- |
| server.port=8080  spring.main.allow-bean-definition-overriding=true  spring.datasource.hikari.driver-class-name=com.mysql.cj.jdbc.Driver  spring.datasource.hikari.jdbc-url=jdbc:mysql://mysql.web.local:3306/boot\_board?serverTimezone=UTC  spring.datasource.hikari.username=root  spring.datasource.hikari.password=rlarudals  spring.datasource.hikari.connection-test-query=SELECT 1  #root  mybatis.configuration.map-underscore-to-camel-case=true  spring.session.store-type=redis  spring.redis.host=redis.web.local  spring.redis.password=frodo5020!!  spring.redis.port=6379  spring.session.redis.flush-mode=on\_save  spring.session.redis.namespace=spring:session  spring.thymeleaf.cache=false  spring.thymeleaf.enabled=true  spring.thymeleaf.prefix=classpath:/templates/  spring.thymeleaf.suffix=.html |

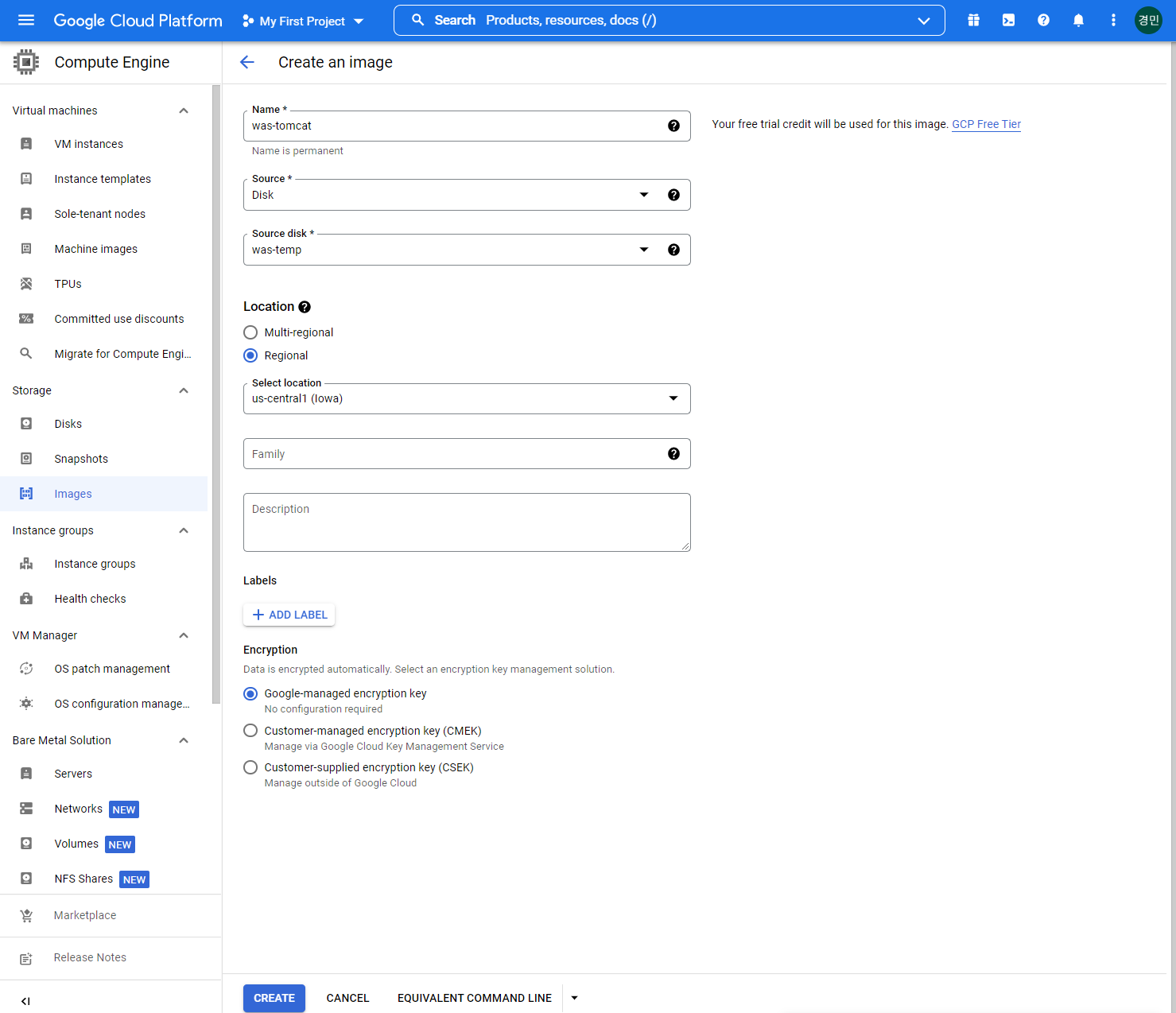
War file 빌드

|  |
| --- |
| gradle clean  gradle build |

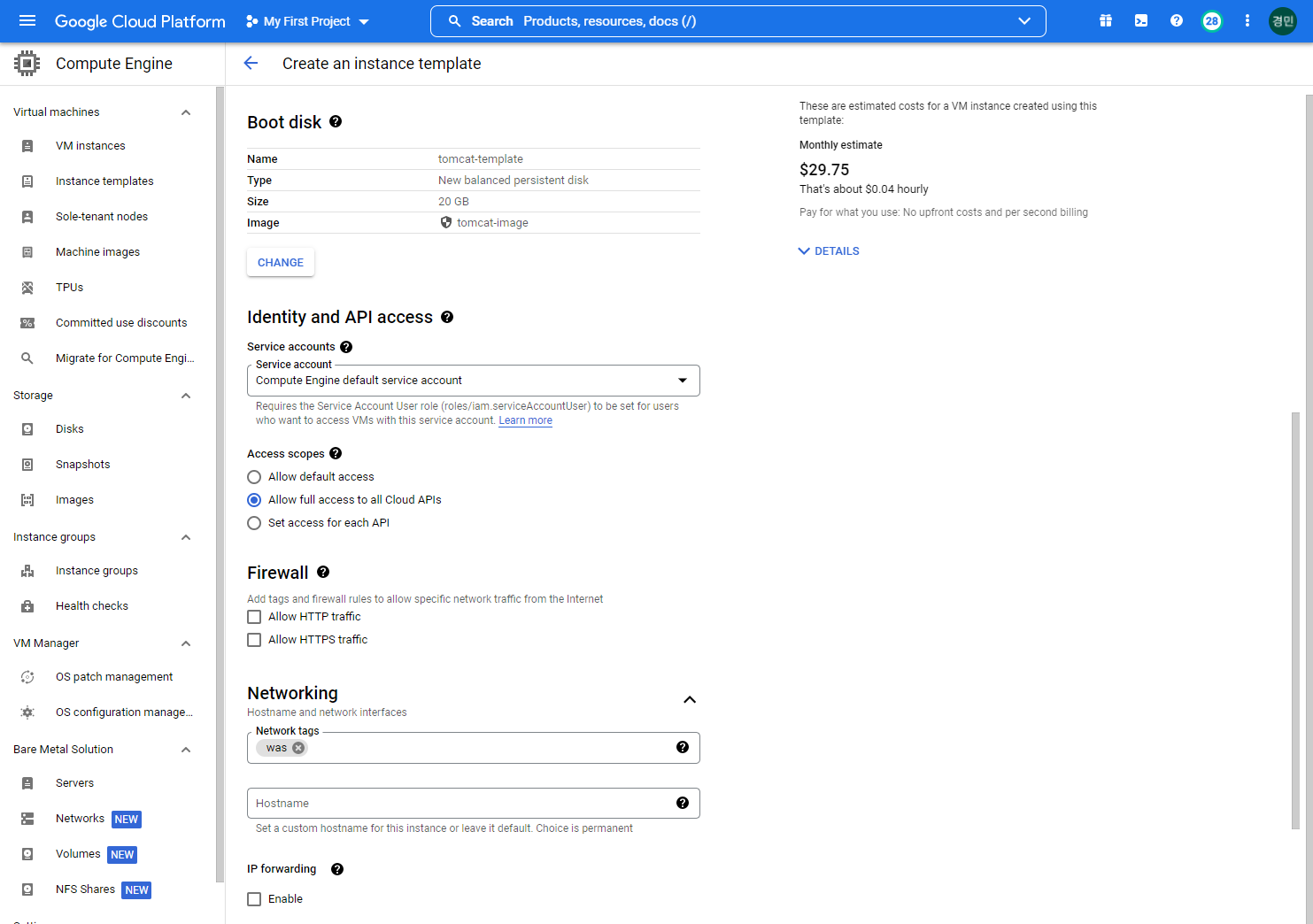
파일 옮기기

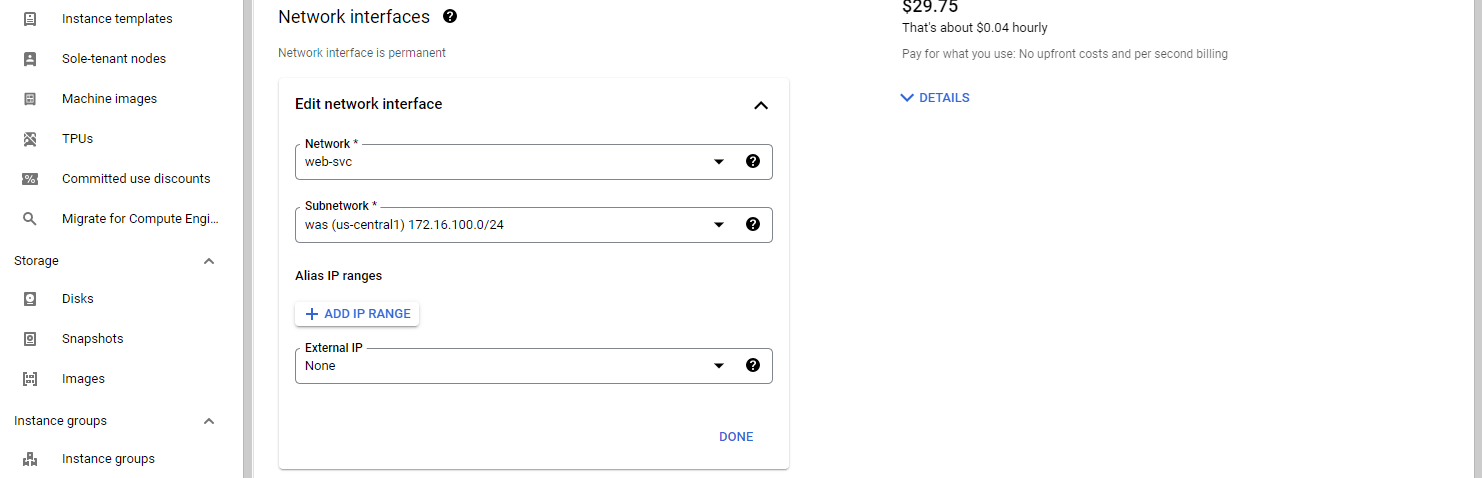
|  |
| --- |
| cp my-spring-board-0.0.1-SNAPSHOT.war /opt/tomcat/apache-tomcat-9.0.62/webapps/ROOT.war  systemctl restart tomcat  # 설치 확인  curl localhost:8080 |

### **3. 이미지 생성**



## Instance template 생성

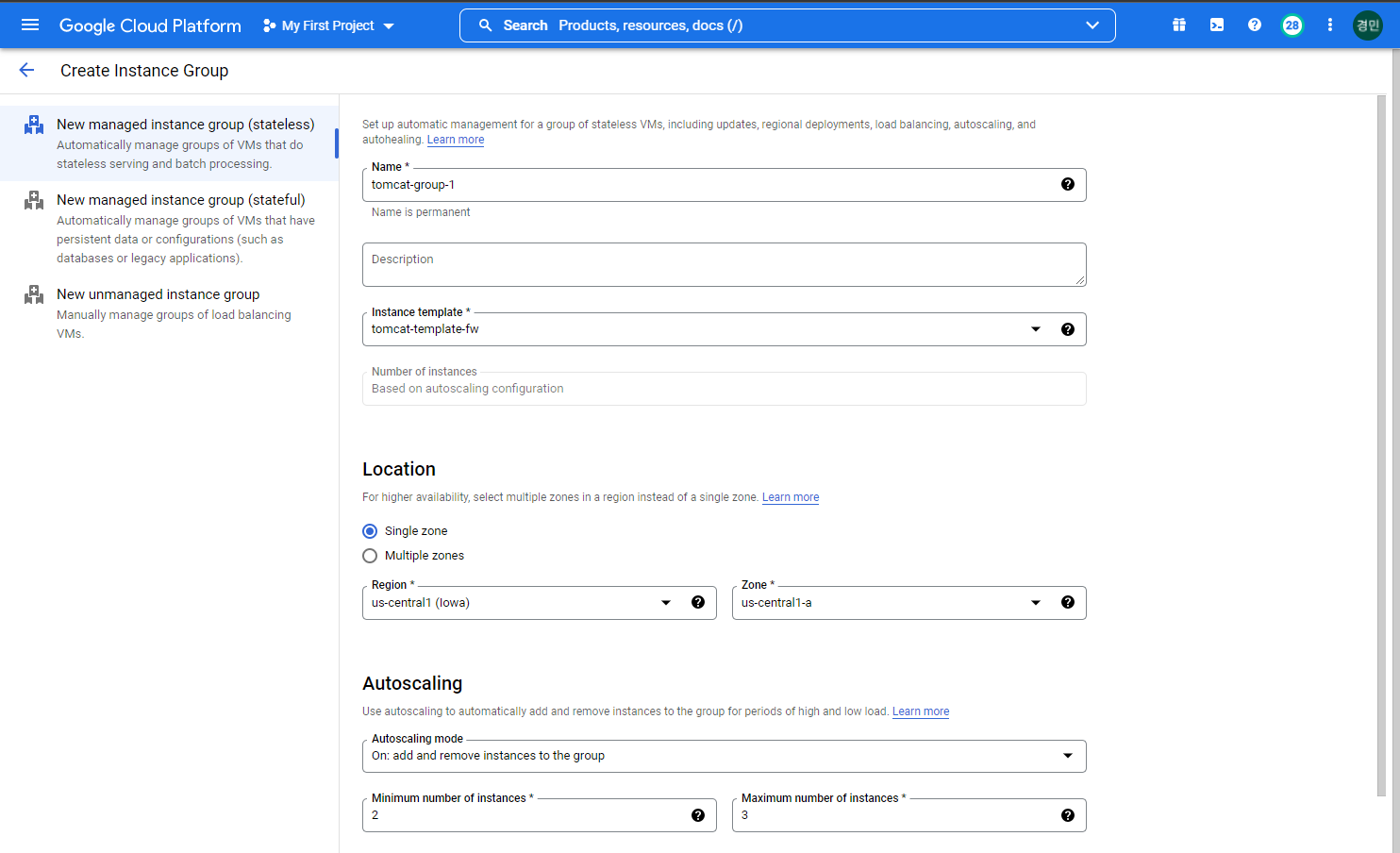




이미지 선택, network tag - was, VCP 선택, Extarnal IP -None

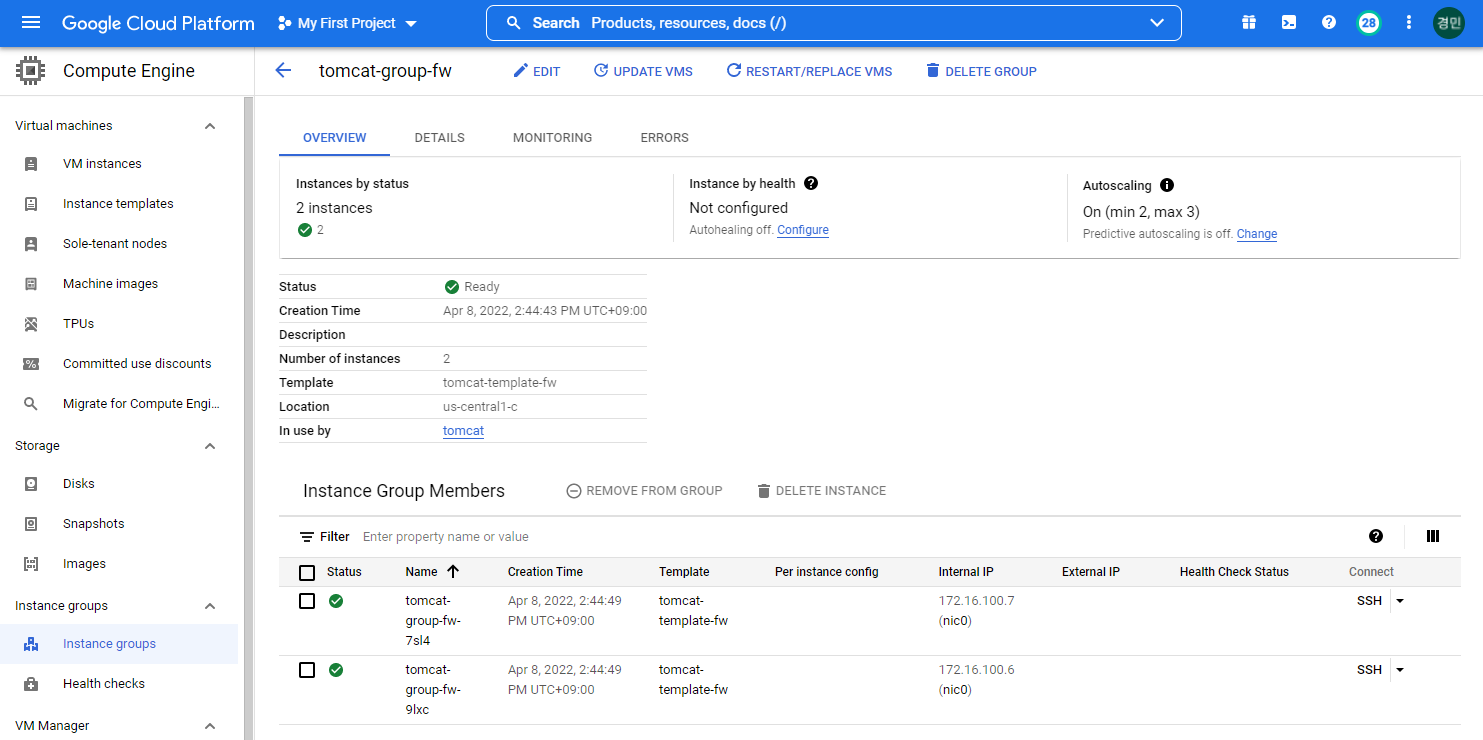
생성

## Instance group, Autoscaleing



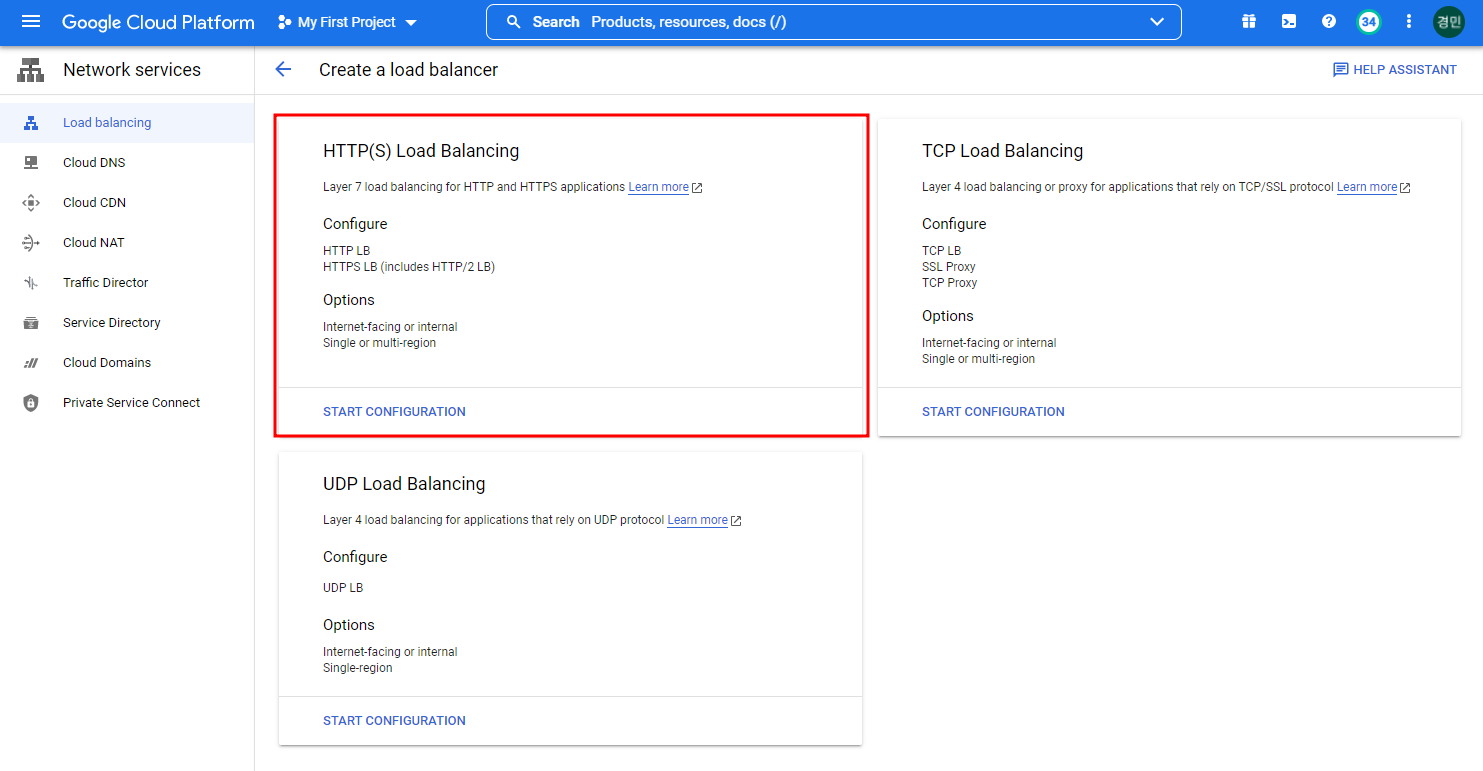
Template 선택 후 create

자동으로 최소개수만큼의 VM이 생성된다

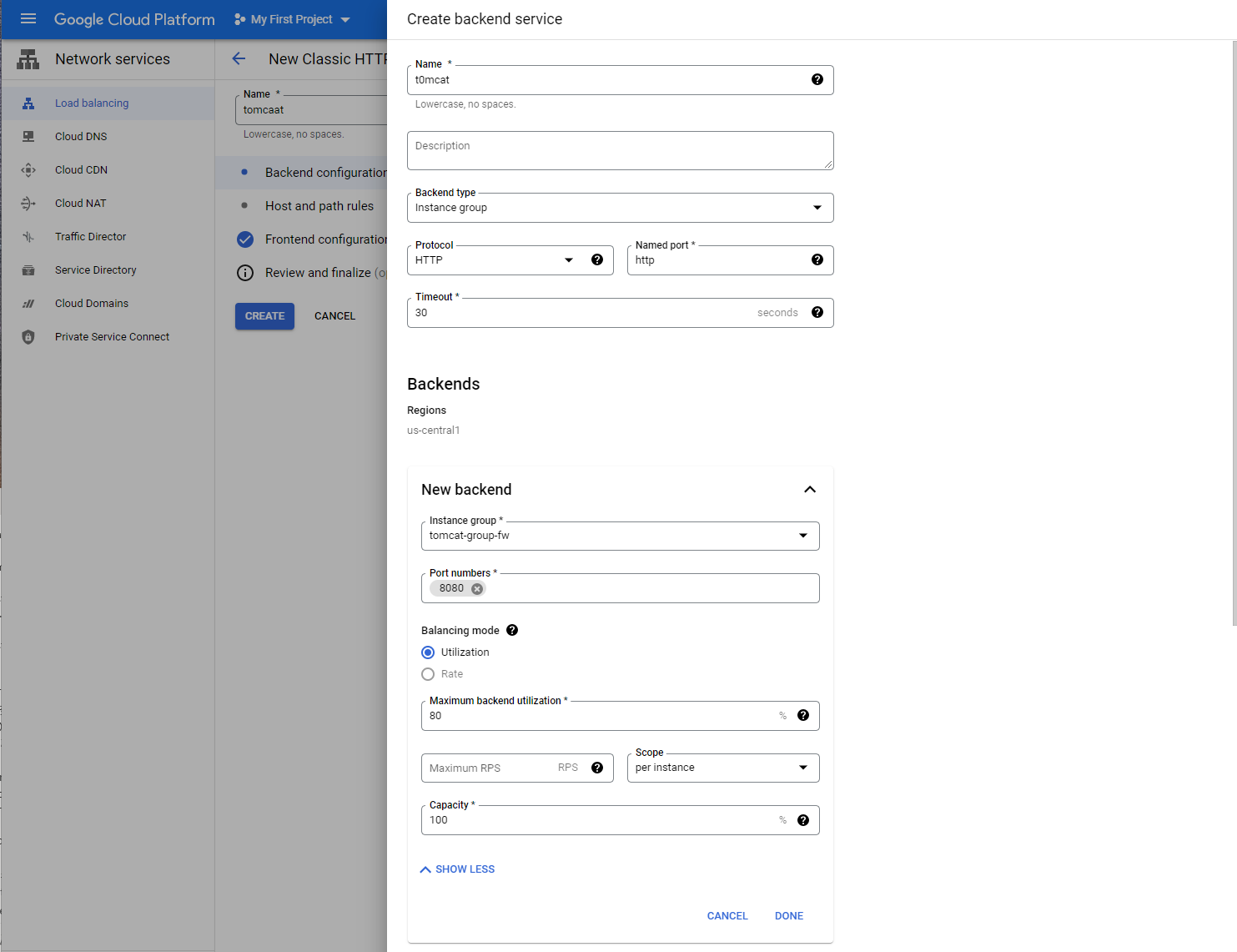


# LB

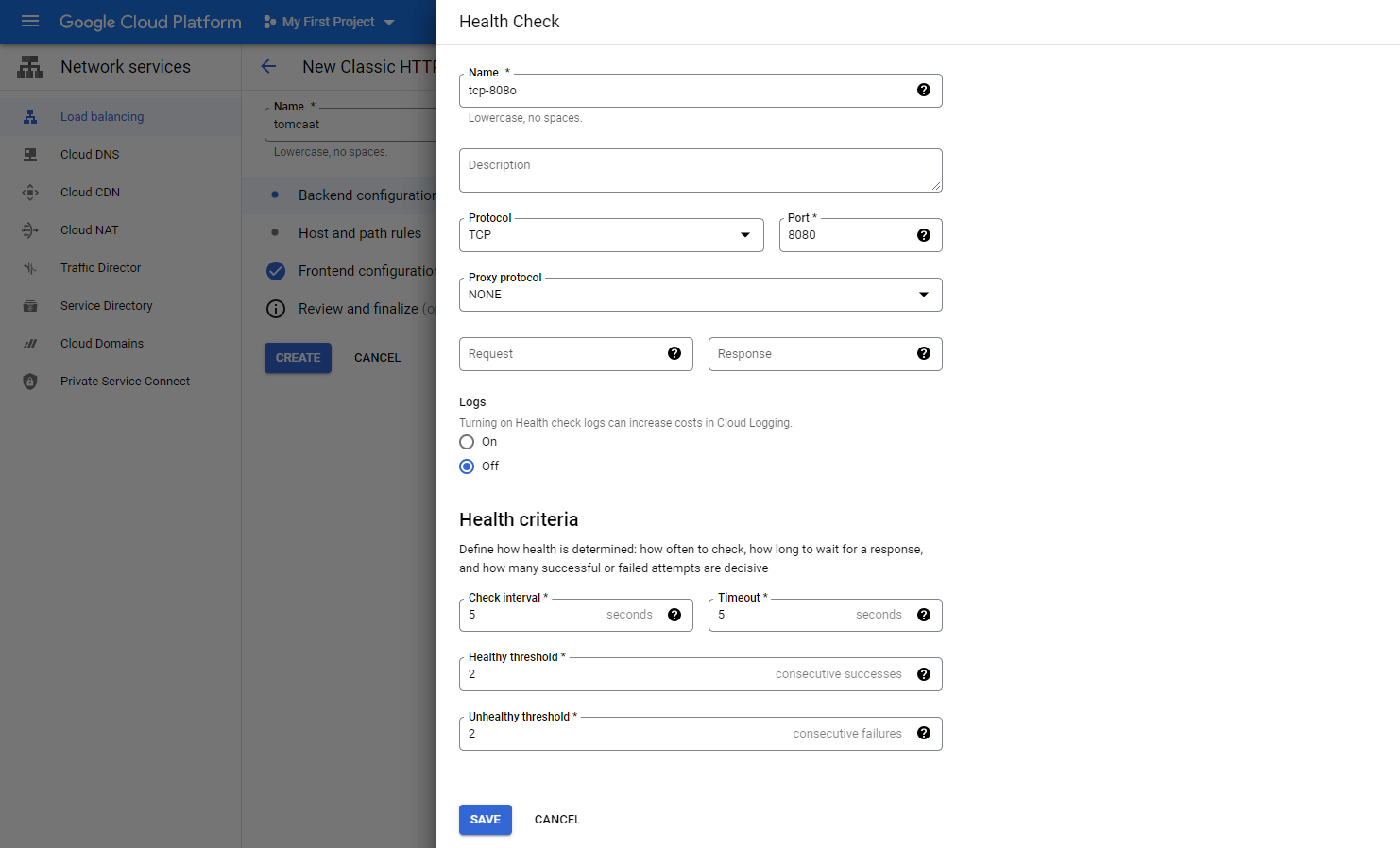
Network services > Load balancing - Create a load balancer



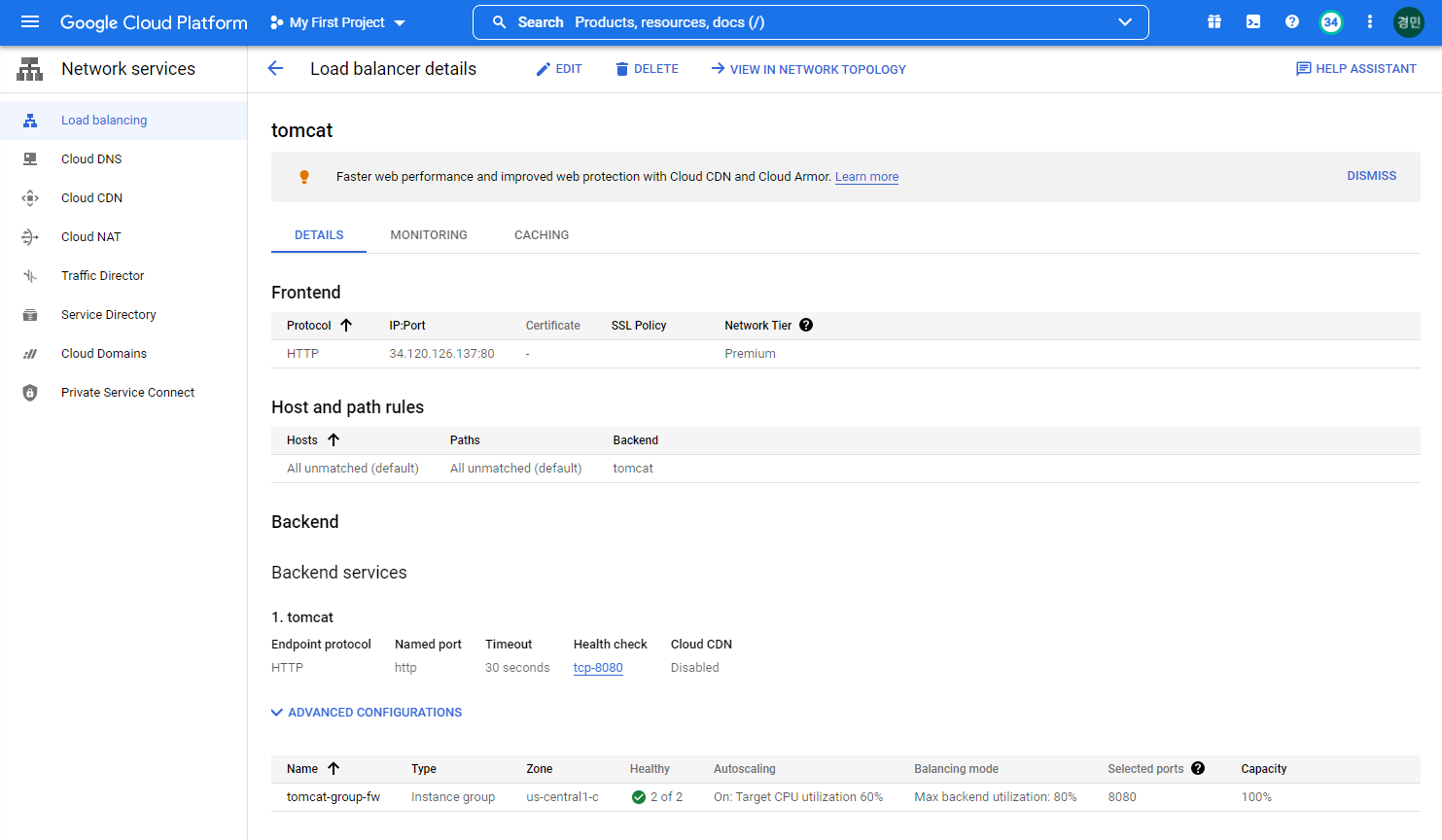
## Backend config



## Health check create



## 접속 확인



IP접속

