제목

문서 기록

(표)

작성자 날짜 설명

이찬행 2022-02-07 ver 1.0 처음 생성

목차

[VPC 생성 4](#_Toc95146691)

[VPC CIDR 4](#_Toc95146692)

[서브넷 생성 4](#_Toc95146693)

[NACL , SG 생성 4](#_Toc95146694)

[인터넷 게이트웨이 & 라우팅 테이블 연결 4](#_Toc95146695)

[EC2 생성 4](#_Toc95146696)

[인스턴스 생성 4](#_Toc95146697)

# VPC 생성

## VPC CIDR

우선 VPC 생성을 위해선 IP주소를 어떻게 할당하는지가 중요하다.

우리는 3-tier로 연결할 예정이므로, www(web) – was(api) – database에 적절한 IP 범위를 할당해주어야 한다. 이를 위해서 CIDR 계산기를 활용할 것이다.

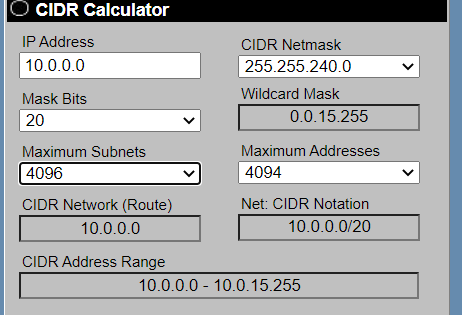
텍스트이(가) 표시된 사진

자동 생성된 설명

<subnet calculator>

사이트 이름은 subnet calculator로, IP address와 할당하고자 하는 주소를 정해주면 알아서 계산을 해주는 아주 유용한 사이트다. 이제 IP주소를 얼마나 할당해야 하는지 정해야 한다.

뭐든지 넉넉하게 잡아야 하기 때문에 총 할당량은 4096개로 정했다. 해당 주소는 vpc생성에 사용된다.



공인 IP는 통신사에서 관리하고 있으므로, 우리는 사설 IP를 사용해야 한다. 사설 IP에도 나름의 기준이 있는데,

10.0.0.0~ - 서버용

172.0.0.0 - 장치 (디바이스 VPN 장치 같은 것)

192.0.0.0 - 사용자pc

이렇게 사용된다. 우리가 흔히 사용하는 공유기가 192.~으로 하는 이유가 바로 이것이다.

이제, IP주소로 VPC생성을 하겠다. 텍스트이(가) 표시된 사진

자동 생성된 설명

아마존 AWS 사이트에 들어와서 VPC 생성버튼을 누른 후 첫 화면이다.

이름을 정하고, IPv4 CIDR을 10.0.0.0/20으로 정했다.

이제 4096개의 IP주소를 WWW-WAS-DB에 적절히 분배할 것이다.

## 서브넷 생성

서브넷은 웹 , WAS 그리고 DB1 & DB2 총 4개를 만들 것이다. 또한 웹에 가장 많은 IP주소가 필요하고, DB가 가장 적게 필요하기 때문에,

웹 – 약 1000개

WAS – 약 200개

DB – 약 50개

를 각각 할당한다.

우선, 웹을 담당할 서브넷을 생성한다.

약 1000개가 필요하기 때문에 서브넷 계산기로 계산한 결과, 10.0.0.0/22를 할당하면 약 1019개 정도를 사용할 수 있음을 확인했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

VPC와 동일하게 이름과 CIDR 블록을 적절히 설정해주었고, 가용영역은 웹을 다룰 땐 크게 상관 없으나, WAS와 DB를 연결할 땐 같은 영역에 있어야 하기 때문에 어느정도 신경을 써주어야한다.

위와 마찬가지로 WAS와 DB의 서브넷을 생성한다. 참고로, WAS와 DB의 IP주소는

WAS – 10.0.8.0/23 (약 510개)

DB\_1 – 10.0.13.0/24 (약 255개)

DB\_2 – 10.0.14.0/24 (약 255개)

이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

여기서, WAS와 DB1을 연결해야 하므로, 가용영역을 같게 했다.

마지막으로 DB2 서브넷까지 생성했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

서브넷을 모두 만든 후의 모습이다.

모두 고유의 서브넷 ID를 가지고 있고, VPC와도 연결되어있음을 확인할 수 있다.

## NACL , SG 생성

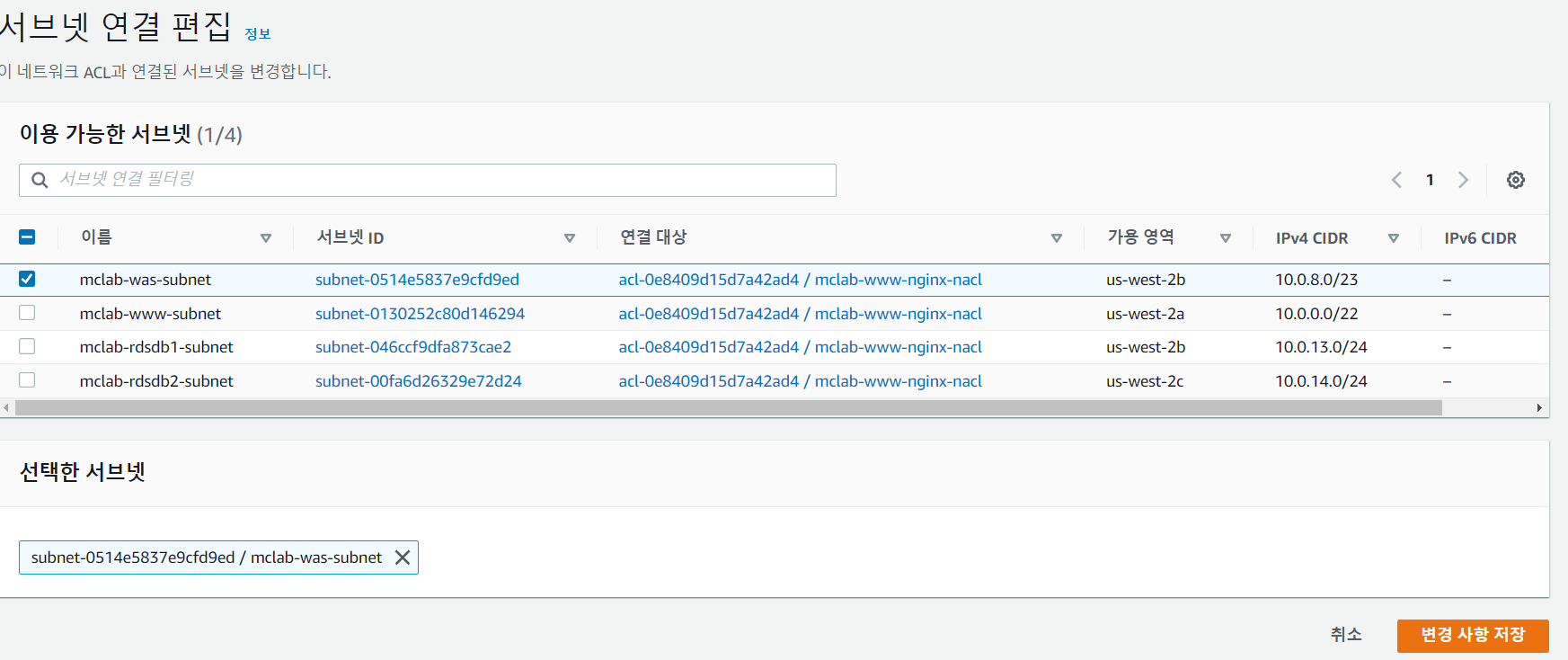
서브넷까지 만들었으면, NACL(Network Access Control List)라고 불리는 네트워크 접근제어목록을 생성하여, 무분별한 타 IP의 접근을 차단할 것이다. 또한 Security Group(보안그룹)도 같이 만들어서 심층 보안을 가동할 수 있도록 한다.

NACL 그리고 보안그룹은 각 서브넷마다 하나씩 만들고, DB는 하나로 통합해서 관리하기 때문에 각각 3개씩 생성한다. 여기서 웹 서브넷은 인스턴스 연결의 원활한 접근을 위해 두개 모두 default 값을 수정하여 작업한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

WAS 서버는 Flask로 연결할 예정인데, 쉽게 알아보기 위해 이름에 flask를 기입했다. 생성 후에 서브넷 연결 편집에서 WAS 서브넷을 연결해준다.



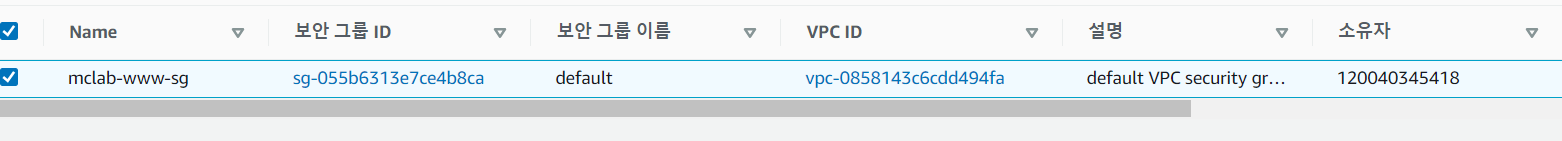
저장을 누르면, 서브넷에 NACL이 생성된다. 같은 방법으로 DB에도 적용하면,

텍스트이(가) 표시된 사진

자동 생성된 설명

이렇게 각 서브넷에 NACL이 연결되는 모습을 확인했다.

이번엔 보안그룹을 생성해보자.



NACL과 마찬가지로, default값에 웹 보안그룹을 생성하고, 나머지 2개의 보안그룹도 생성해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

생성을 모두 마쳤으면, 이제 인바운드와 아웃바운드를 편집해주어야 한다.

인바운드는 외부에서 EC2로 접근할 때를 말하고, 아웃바운드는 외부로 나갈 때를 말한다.

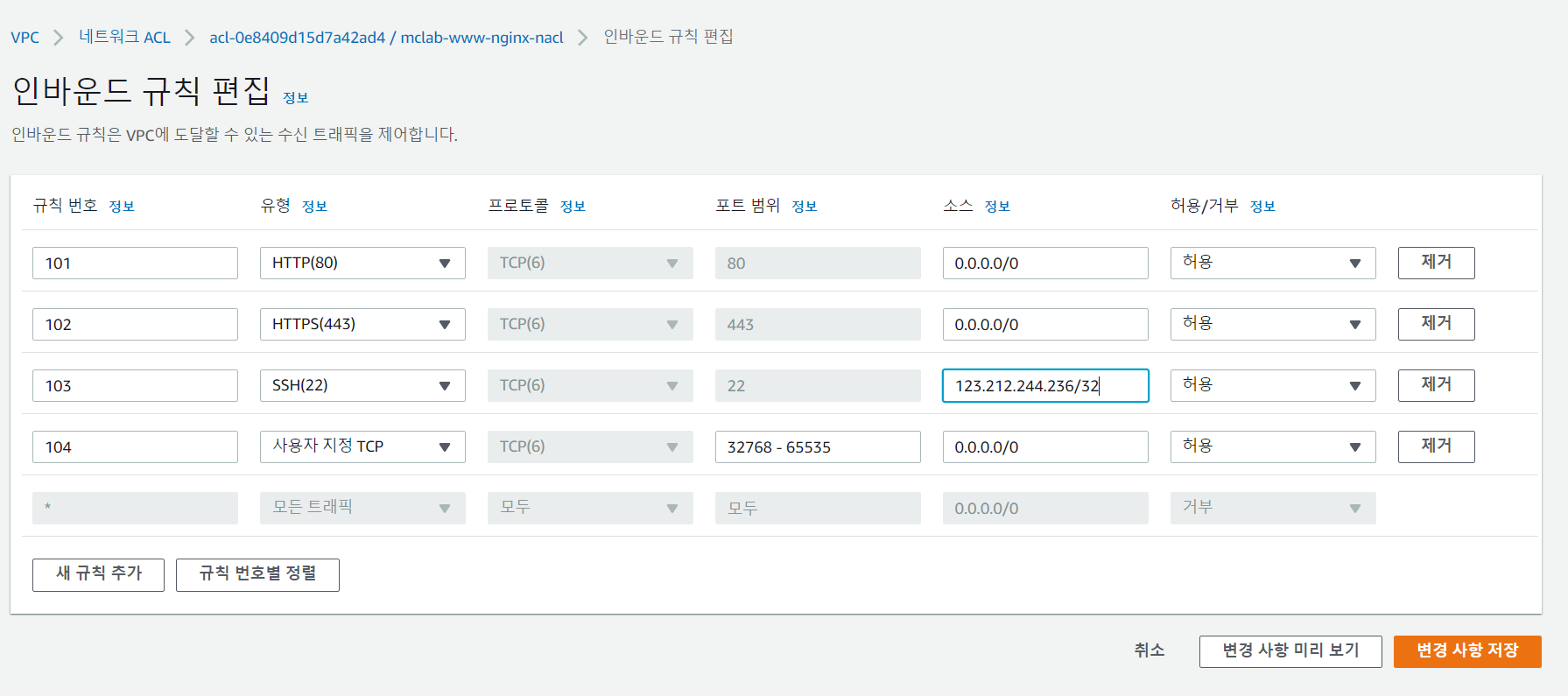
웹 서버는 HTTP와 HTTPS로 웹 통신을,

WAS 서버는 Flask를 통한 통신을

그리고 DB는 MariaDB를 활용하기 때문에 각 해당하는 값을 입력해주어야한다.

이때, 웹 서버의 아웃바운드는 모두 허용으로 한다. 이는 인바운드규칙을 통해 들어온 트래픽이 정상적으로 나갈 수 있도록 하는 것이다.

웹 서버의 인바운드 규칙을 설정한다.



웹 통신은 모든 서버를 통과시켰고, SSH를 내 IP로 제한시켜 내 컴퓨터에서만 웹서버가 작동할 수 있도록 했다.

웹에 EC2의 인스턴스를 생성하여 연결할 예정이므로 동적포트를 할당 해줬다.

아웃바운드는 모두 허용으로 설정했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

WAS에도 인바운드 규칙을 편집했다.

텍스트, 스크린샷, 실내이(가) 표시된 사진

자동 생성된 설명

Flask의 포트번호인 5000번을 범위로 정하고, WAS 서브넷의 IP주소를 입력했다. 또한 웹 서버의 동적포트에서 넘어온 포트의 접근을 허용하기 위해 마찬가지로 동적포트를 할당했다. 아웃바운드는 인바운드와 동일하게 했다.

마지막으로 DB의 인바운드 규칙을 편집한다.

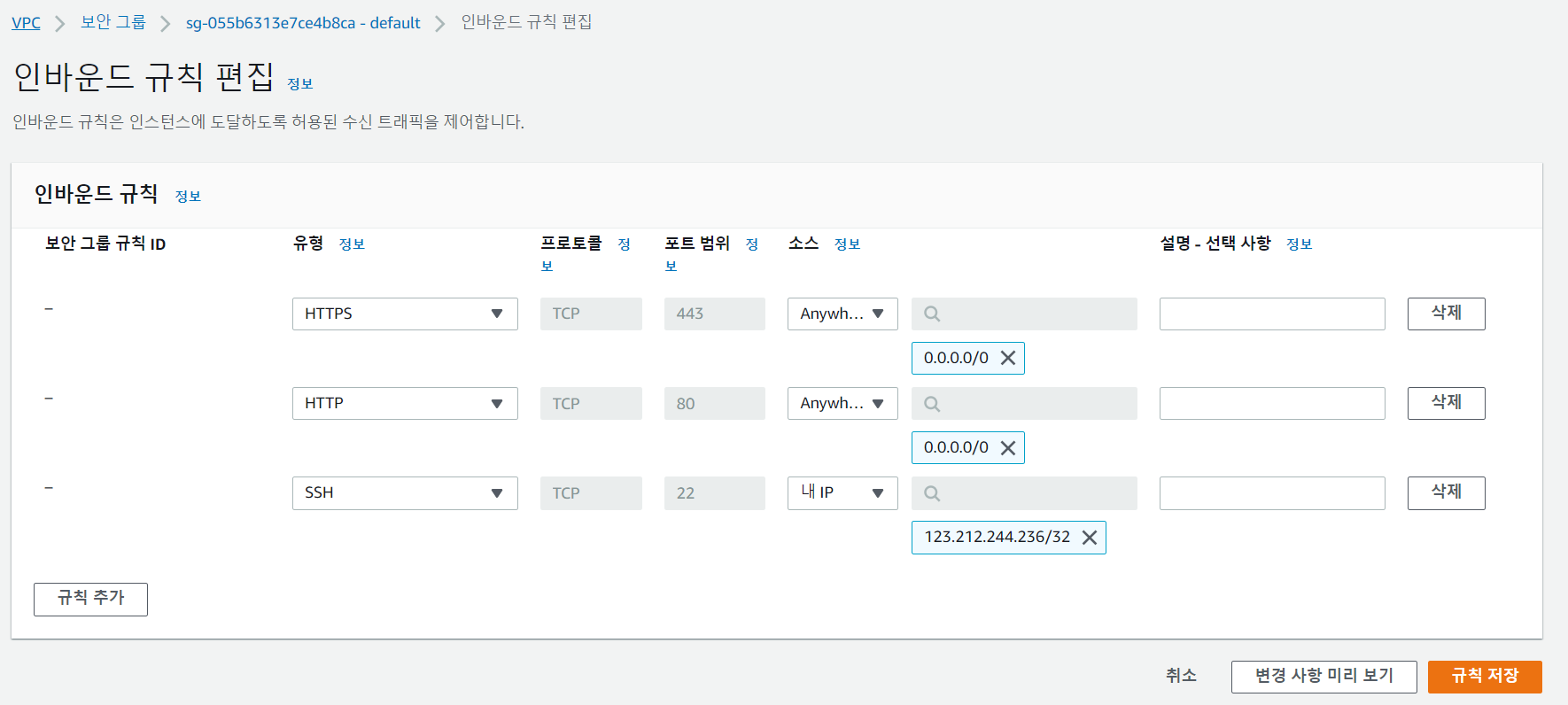
텍스트, 스크린샷, 실내이(가) 표시된 사진

자동 생성된 설명

고유포트인 3306과 동적포트를 할당했고, 아웃바운드도 동일하다.

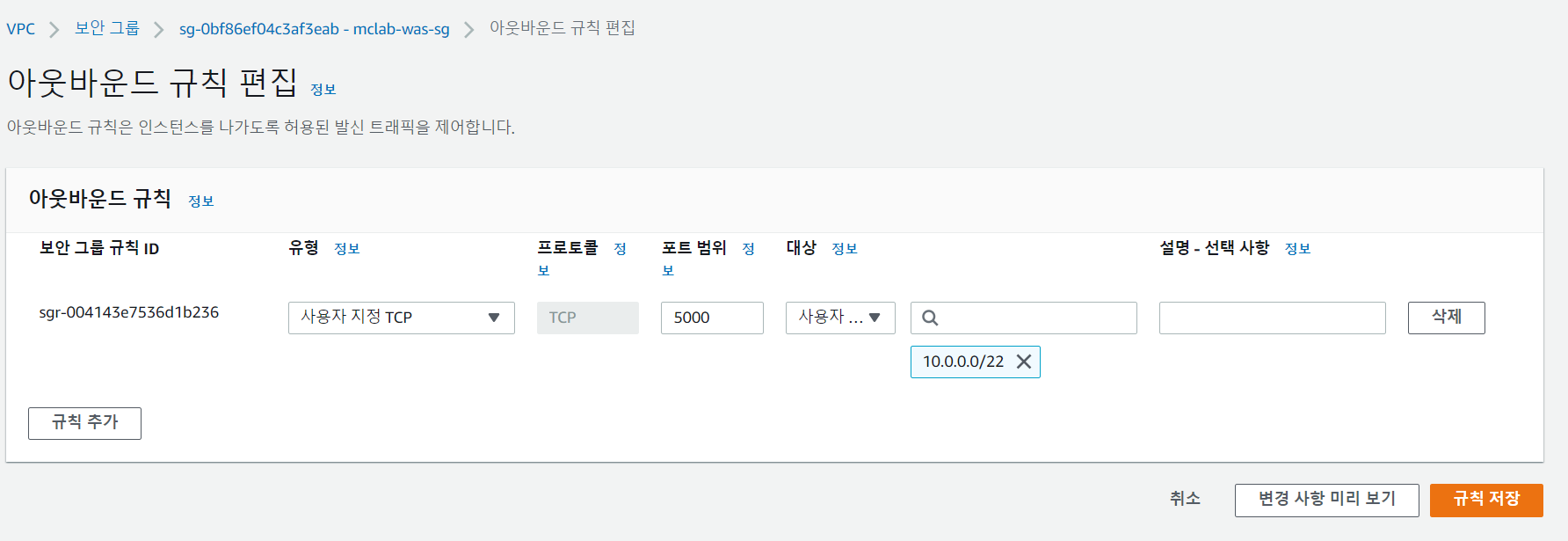
이제 보안그룹을 설정하는데, 보안그룹은 EC2 인스턴스의 보안그룹과도 연관 있기 때문에 적절한 순서에 맞게 해줘야한다.

웹 보안그룹의 인바운드는 동적포트를 제외하고 NACL과 같다.



아웃바운드도 전체 허용한다.

WAS는 인바운드,아웃바운드 모두 Flask만 허용하고 DB는 MariaDB만 허용한다.



텍스트이(가) 표시된 사진

자동 생성된 설명

WAS와 DB 모두 인바운드&아웃바운드 설정은 동일하다.

## 인터넷 게이트웨이 & 라우팅 테이블 연결

이제 host 및 guest가 외부에서 인터넷을 통해 접근하기 위한 과정을 생성할 것이다.

인터넷 게이트웨이를 생성하고 VPC에 연결한 후, 이를 라우팅 테이블에 연결해준다.

텍스트이(가) 표시된 사진

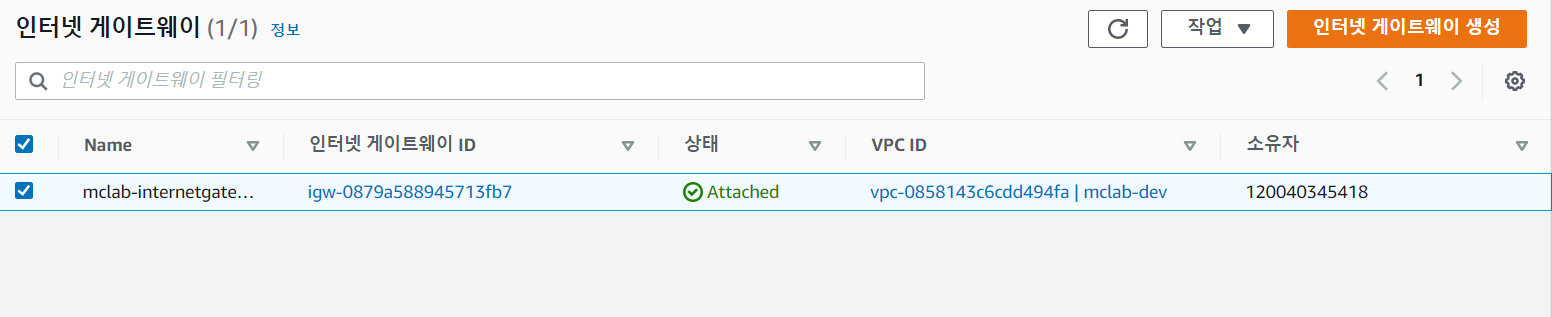
자동 생성된 설명

인터넷 게이트웨이를 생성하면 상단에 VPC 연결 버튼이 뜨는데, 이를 클릭하면

텍스트이(가) 표시된 사진

자동 생성된 설명

이렇게 내가 생성했던 VPC와 연결할 수 있도록 해준다. 연결을 누르면,

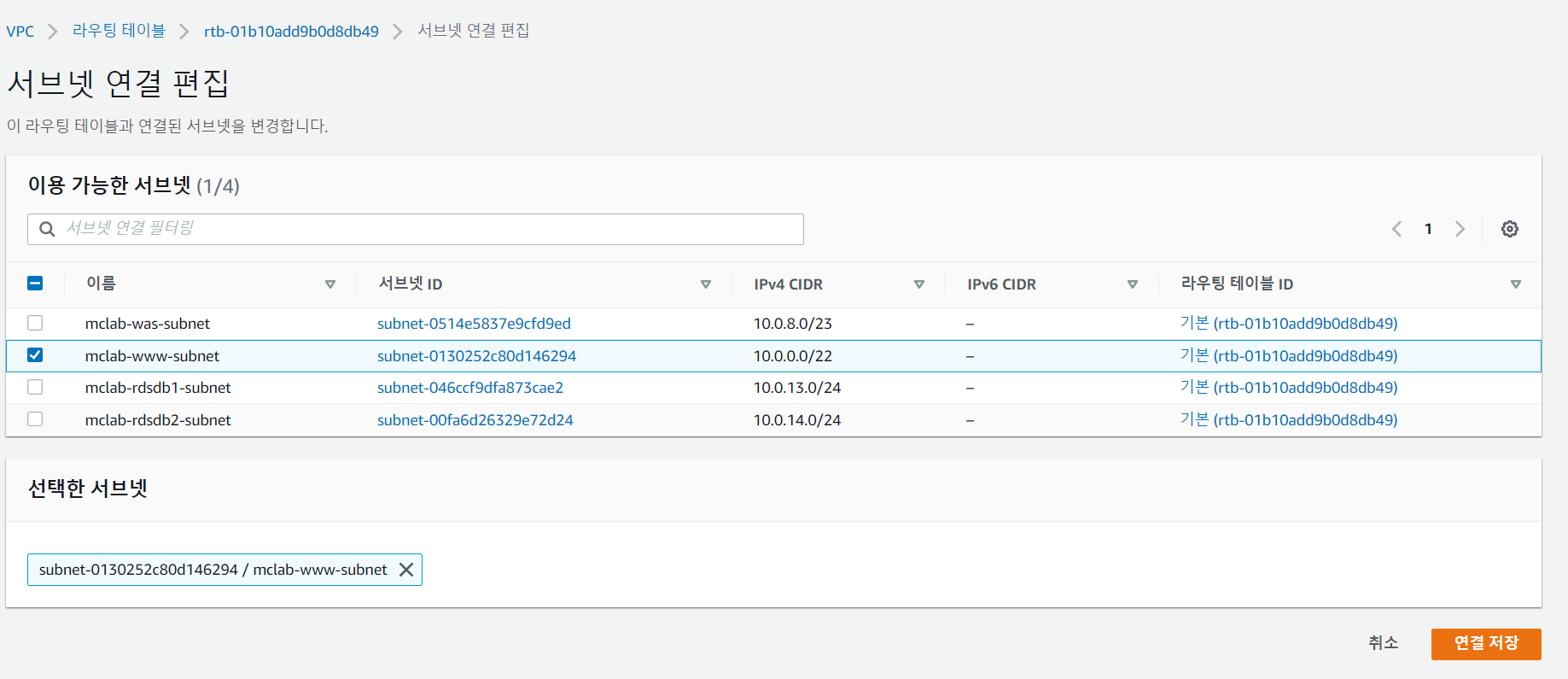


내 VPC와 연결 됐다는 사실을 확인시켜준다. 이제 라우팅 테이블을 생성한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

라우팅 테이블도 default 값을 수정하는데, 라우팅 편집에 들어가서 0.0.0.0/0 즉 내가 설정한 인터넷게이트웨이를 통해 들어온 모든 인터넷의 통신이 가능하도록 설정한다. 그 이후에, 웹 서버와 연결해야 하므로 웹 서브넷과 연결해준다.



웹 서브넷을 연결해주고 나면, 외부의 인터넷 접근을 통해 게이트웨이를 거쳐 라우팅 테이블을 통과하여 나의 웹과 통신이 가능해지는 것이다.

# EC2 생성

## 인스턴스 생성

이제 도로를 깔았으니 건물을 세울 것이다. 프리티어 계정을 사용해 돈이 빠져나가지 않는 선에서 생성할 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우린 가장 밑에 있는 Ubuntu Server 20.04를 사용한다.

텍스트, 스크린샷, 실내이(가) 표시된 사진

자동 생성된 설명

유형도 프리티어에서 사용가능한 그룹을 선택하고 넘어간다.

텍스트이(가) 표시된 사진

자동 생성된 설명

네트워크에 내가 생성한 VPC를 선택하고, 웹 서버로 첫 접근을 하므로 웹 서브넷을 선택한 다음에 넘어간다. 스토리지와 태그는 넘어가고, 보안설정에서 기존 그룹을 선택한 후에 웹 보안그룹을 클릭한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

시작 버튼을 누르면, 키 페어 설정이 나오는데, 새로운 키 페어를 생성한 후에 다운로드를 받는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

인스턴스 시작을 누르면, 생성이 진행된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

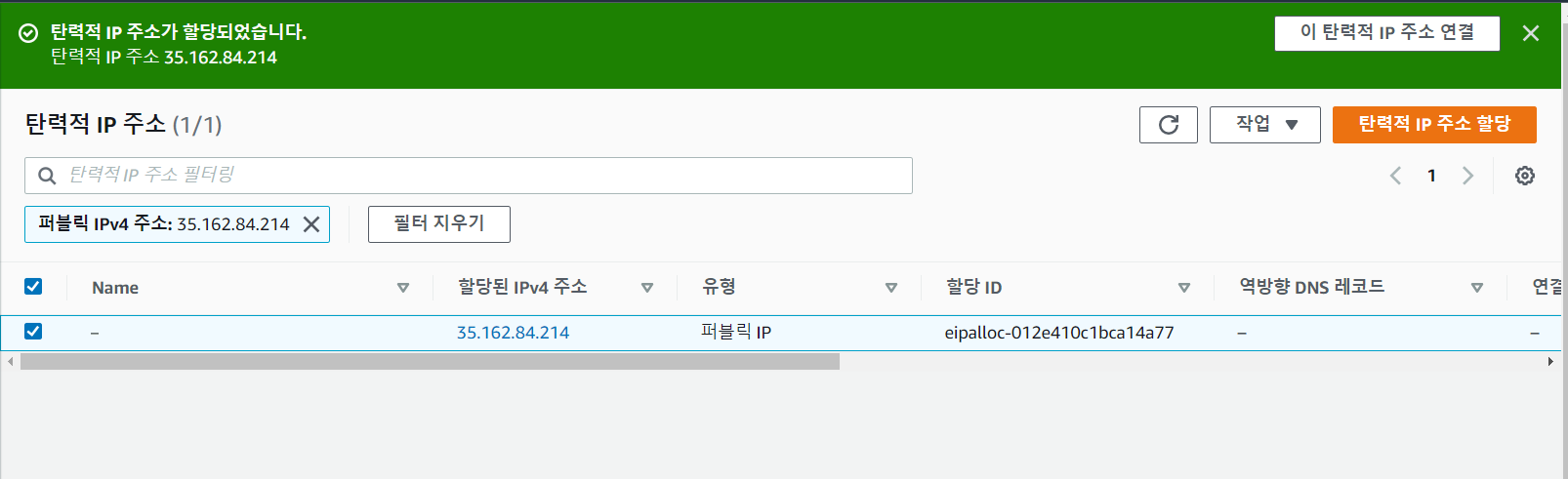
이런 화면이 뜨면 정상적으로 생성이 된 것이고, 우측 하단에 인스턴스 보기 버튼을 누르면 인스턴스 상태를 확인할 수 있다. “실행 중”이라고 뜨면 정상적으로 실행되고 있다는 것이다.

## SSH 프로그램 접속 후 Nginx 설치

인스턴스를 실행하면, 이제 탄력적IP주소를 할당하여 웹서버에 접근을 시도할 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

내 가용영역을 확인한 후, 할당을 누르면 생성이 된다. 생성 후에, IP를 연결해줘야 하는데, 

상단에 있는 연결 버튼을 누른다.

텍스트이(가) 표시된 사진

자동 생성된 설명

내가 생성한 인스턴스와 인스턴스의 프라이빗IP주소를 확인하고 연결한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

퍼블릭 주소 생성된 것을 보아, 탄력적 IP주소가 정상적으로 연결 됐다는 것을 알 수 있다.

이제 SSH 프로그램으로 접속을 시도할 것이다. 우선 인스턴스 상단에 있는 연결을 누른다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이런 화면이 뜨면 성공이고, 이제 컴퓨터의 명령프롬프트를 실행시킨다.

텍스트이(가) 표시된 사진

자동 생성된 설명

나는 다운로드 폴더에 키 페어를 저장했으므로, 다운로드 폴더로 이동했고, 디렉터리에 mclab-webserver.pem 파일이 있다는 것을 확인했다.

AWS 연결 창에 있던 주소를 복사해서 입력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우분투 서버에 정상 접속 됐음을 알 수 있다.

우선 sudo apt update를 통해 최신화를 하고, 그 이후에 sudo apt install nginx 를 입력하여 서버를 설치한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

업데이트를 마치고, nginx 서버를 설치한다.

설치를 마치면, 퍼블릭 ip주소를 주소창에 입력시, nginx 서버가 등장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명