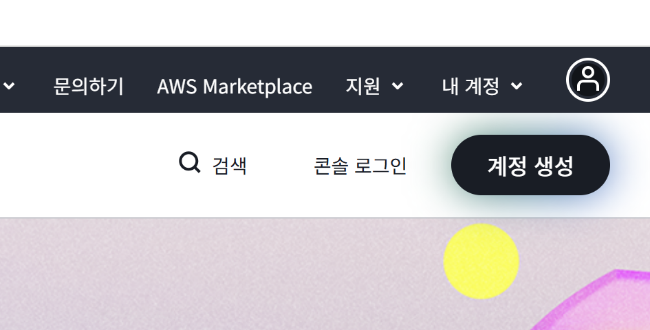
AWS EC2 장고 프로젝트 배포

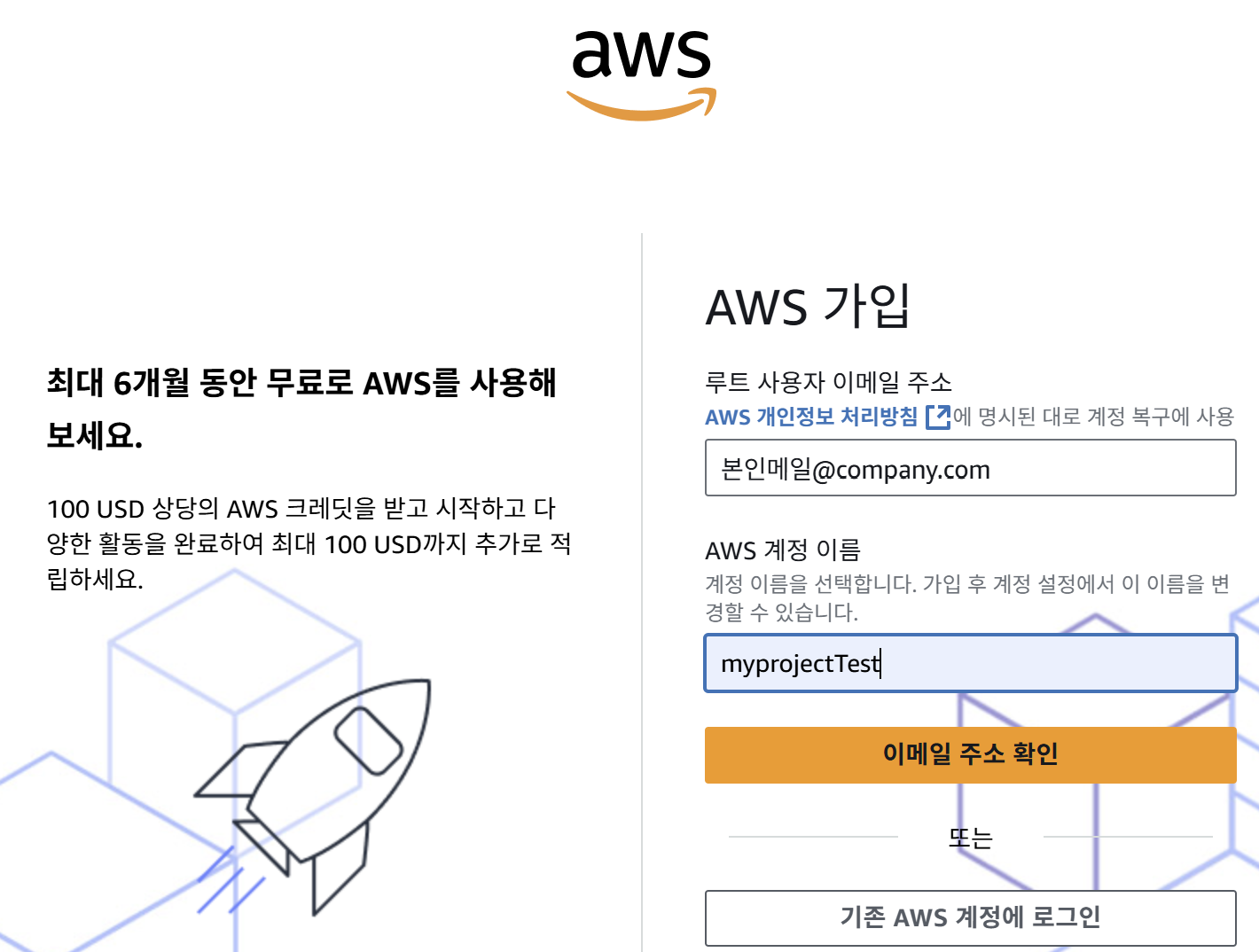
✅ AWS 회원 가입 및 계정 생성

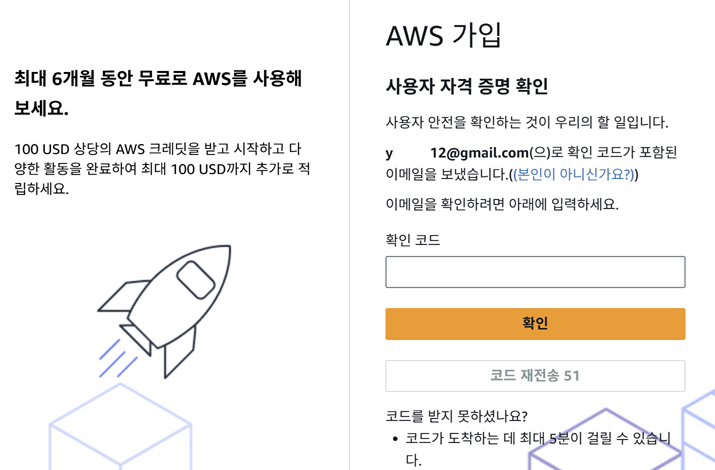
1. AWS 서비스에서 계정을 생성(Amazon Web Services)

참고 : <https://docs.aws.amazon.com/ko_kr/SetUp/latest/UserGuide/setup-AWSsignup.html>

<https://aws.amazon.com/ko/> 에서 계정 생성

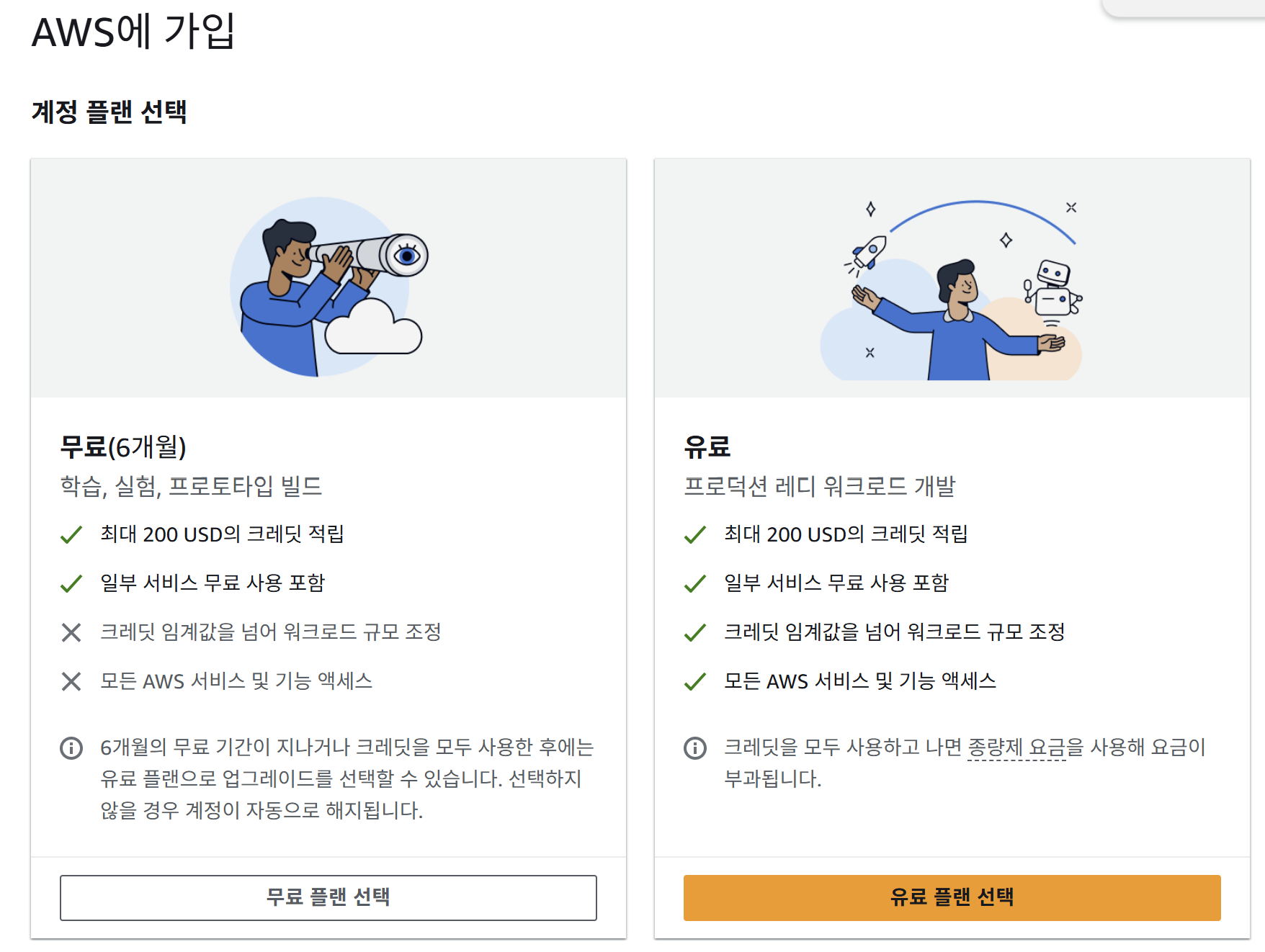






**①**

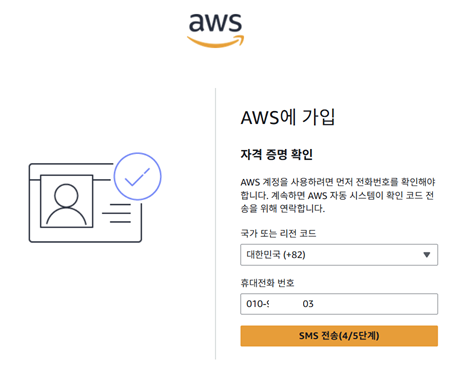
**②**

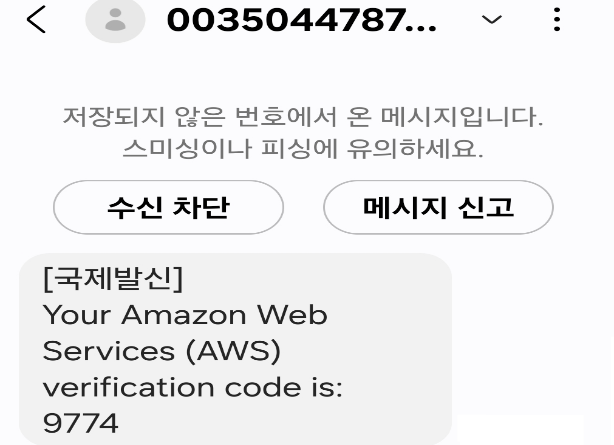




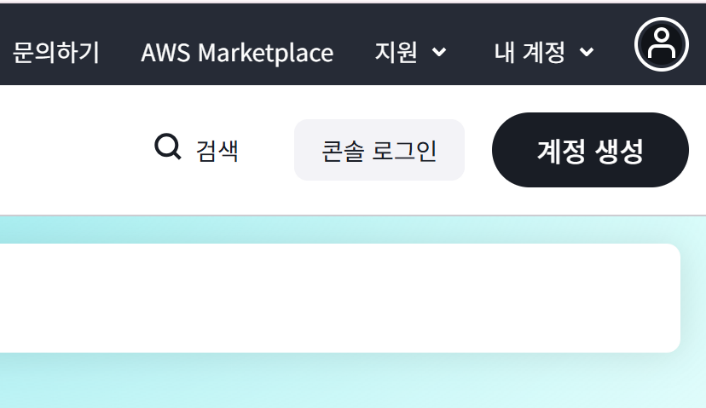


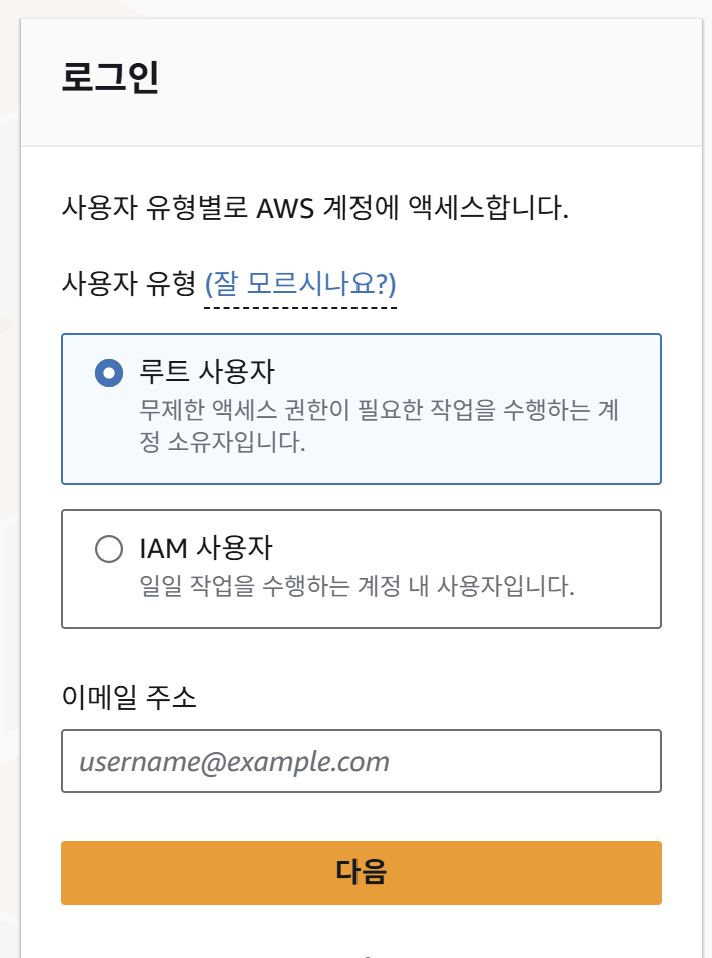
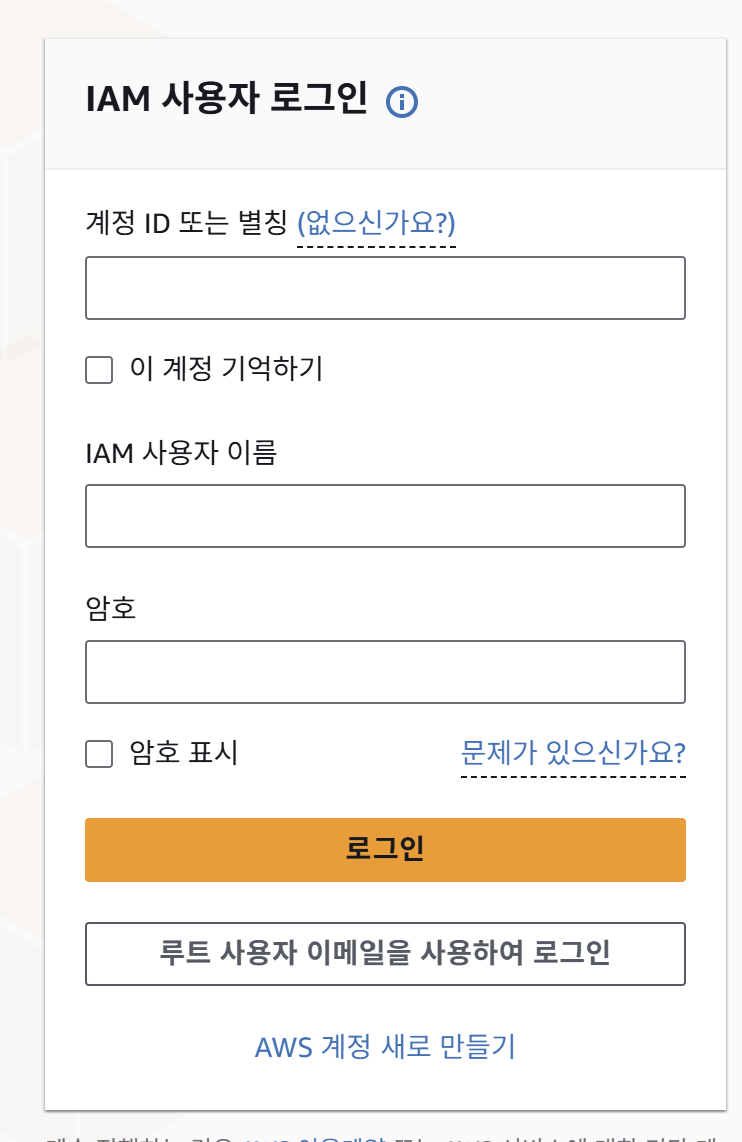




**콘솔 로그인의 경우**





휴대폰에 Authenticator 앱 설치할 경우도 있음

✅ 벡엔드 API 서버 배포란

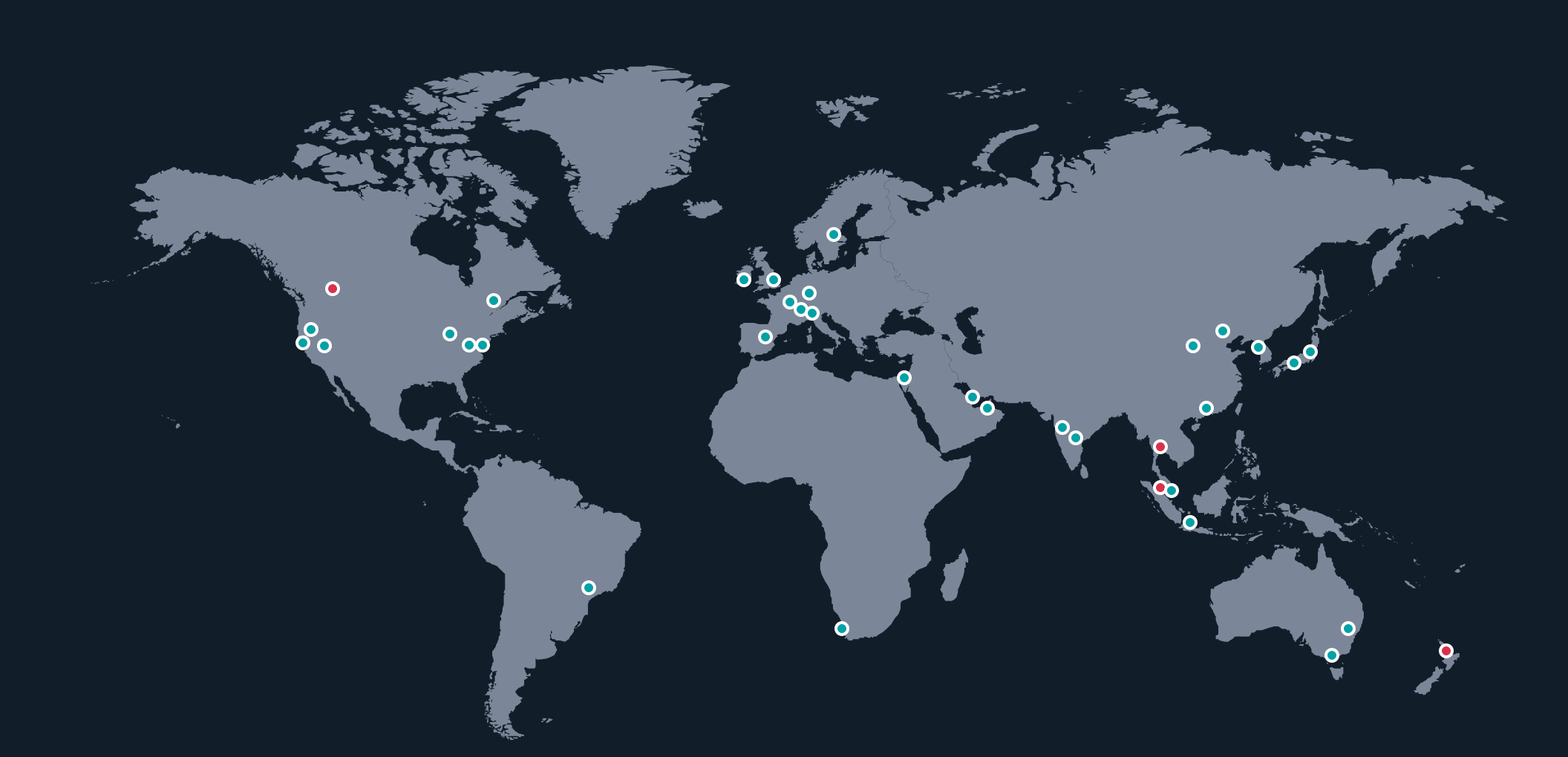
1. 배포란 다른 사용자들이 인터넷을 통해 사용할 수 있게 만드는 것.
2. 개발할 때는 localhost라는 주소로 테스트하지만, 다른 컴퓨터에서는 접근이 불가능한 주소.
3. 배포하게 되면, ip(182.209.94.186)나 도메인([www.xxx.co.kr](http://www.xxx.co.kr))과 같은 고유의 주소를 부여 받게 되어 다른 컴퓨터에서 그 주소로 접속할 수 있게 한다.

✅ EC2(Elastic Compute Cloud)

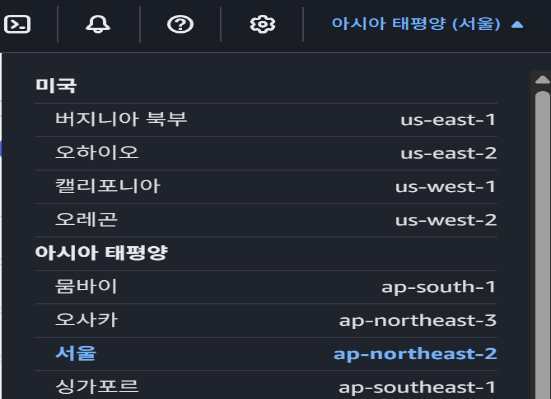
1. EC2 : 컴퓨터를 빌려서 원격으로 접속해 사용하는 서비스. 쉽게 애기하면 하나의 원격 컴퓨터
2. AWS EC2의 부가 서비스 : 로깅, 오토스케일링, 로드밸런싱 등

✅ EC2 실습

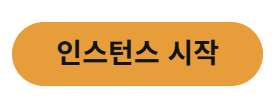
1. Region 선택 : 리전이란 인프라는 지리적으로 나누어 배포한 각각의 데이터 센터. EC2를 통해 빌려서 쓸 수 있는 컴퓨터들이 전세계적으로 다양하게 분포해 있다. 이렇게 컴퓨터들이 위치한 위치를 Region이라 함.



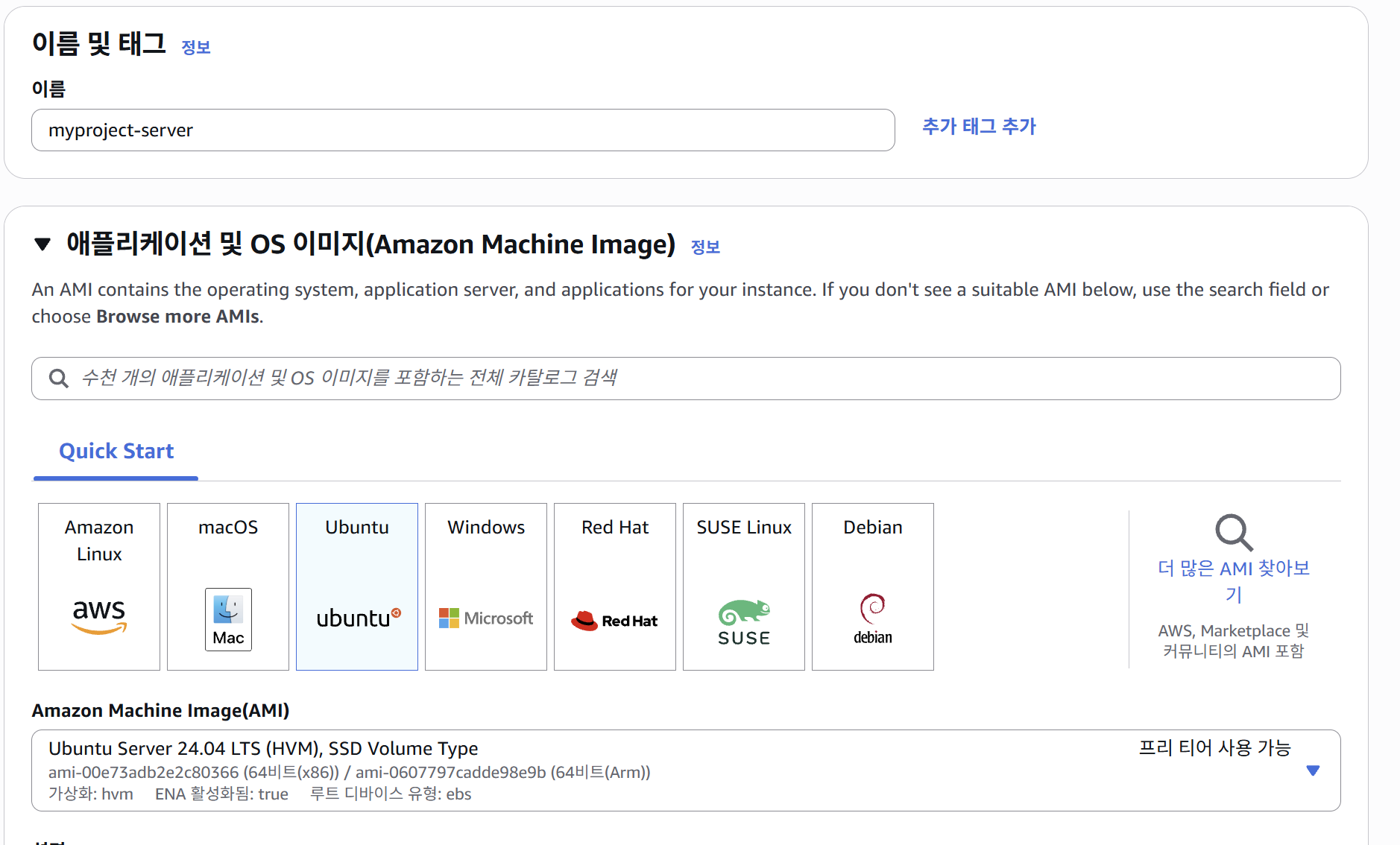
1. AWS EC2 서비스로 들어가서 Region 선택하기



1. Region은 애플리케이션의 주된 사용자들의 위치와 지리적으로 가까운 Region을 선택하는 것이 유리함
2. 인스턴스 시작 클릭



1. EC2의 이름을 설정. 이름을 지을 때는 이 컴퓨터가 어떤 역할을 하는지 알아볼 수 있게 작성. 운영체제는 Ubuntu를 선택해야 함. 맥이나 윈도우에는 user interface 부분 많은 용량 차지함. Ubuntu가 훨씬 가볍움. 다른 이유로는 장고 표준 운영 스택이 Nginx + Gunicorn인데, Gunicorn은 윈도우를 공식 지원하지 않아 Ubuntu면 한줄이면 끝날 걸 다른 운영체제는 우회 구성이 필요



1. 인스턴스 유형 : t2.micro (용량이 부족한 경우 t2.small)

인스턴스란, AWS EC2에서 빌리는 컴퓨터 1대를 의미.

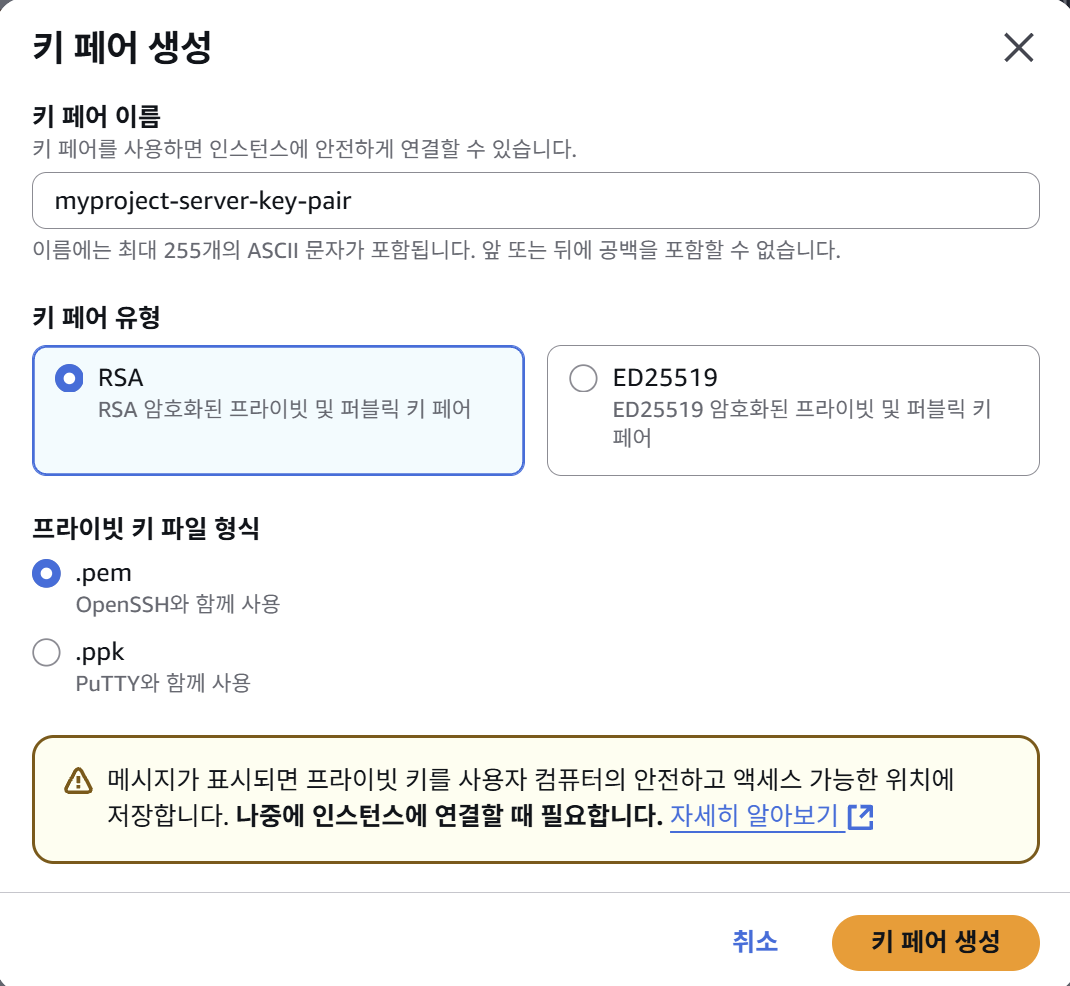
인스턴스 유형이란, 컴퓨터 사양을 의미. 컴퓨터 사양이 좋으면 좋을수록 많은 수의 요청을 처리할 수 있고, 무거운 서버나 프로그램을 돌릴 수 있다.

ex1. t3.small (조금 더 좋은 사양) vCPU 2개, 메모리 1GB200달러로: 약 10개월 사용 가능

ex2. t3.medium (중간 사양) 사양: vCPU 2개, 메모리 2GB 200달러로: 약 5개월 사용 가능

1. 키 페어(로그인) : EC2 컴퓨터에 접근할 때 사용하는 비밀번호. 열쇠(key) 역할

“새 키 페어 생성”클릭

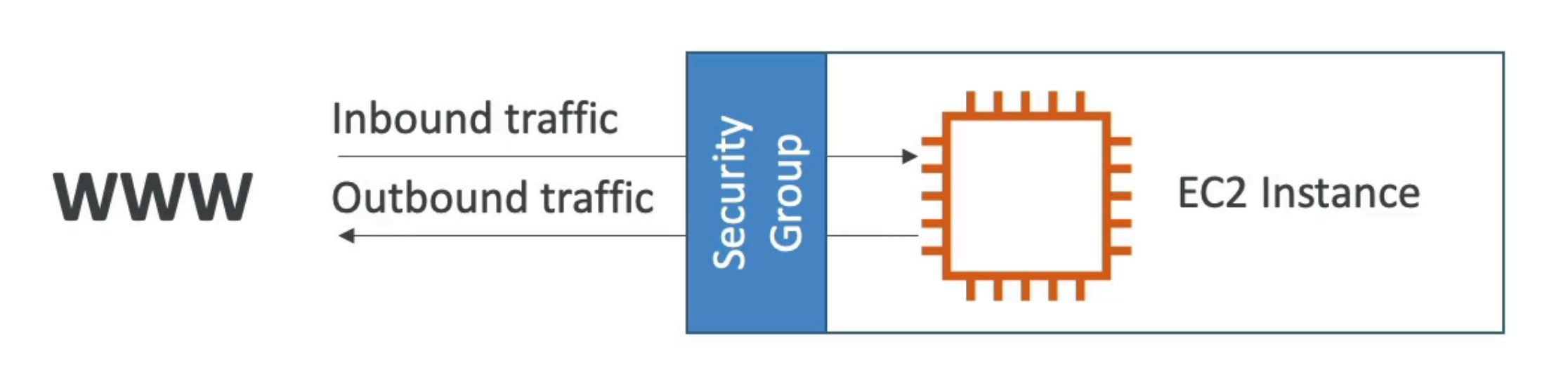


**다운받은 파일 myproject-server-key-pair.pem는 C:\Users\<내컴퓨터명>\.ssh에 보관(물론 이 파일을 안 쓸 수도 있음)**

1. 네트워크 설정



1. 보안 그룹(Security Group)이란 AWS 클라우드에서의 네트워크 보안을 의미



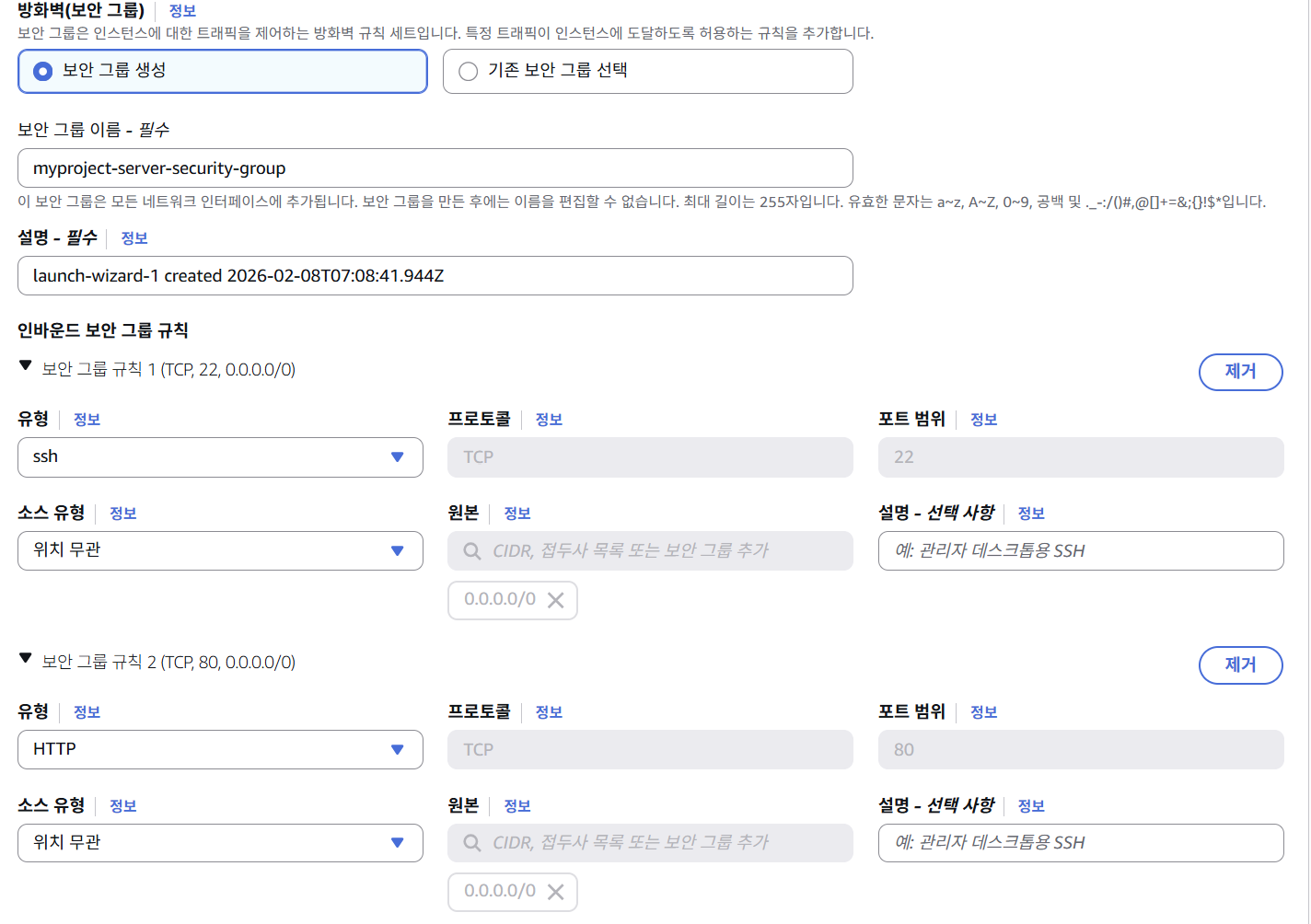
EC2 인스턴스를 집이라고 생각한다면, 보안 그룹은 집 바깥 쪽에 쳐져있는 울타리와 대문이라고 생각하면 된다. 집에 접근할 때 울타리의 대문에서 접근해도 되는 요청인지 검사를 하는 것과 비슷하다.

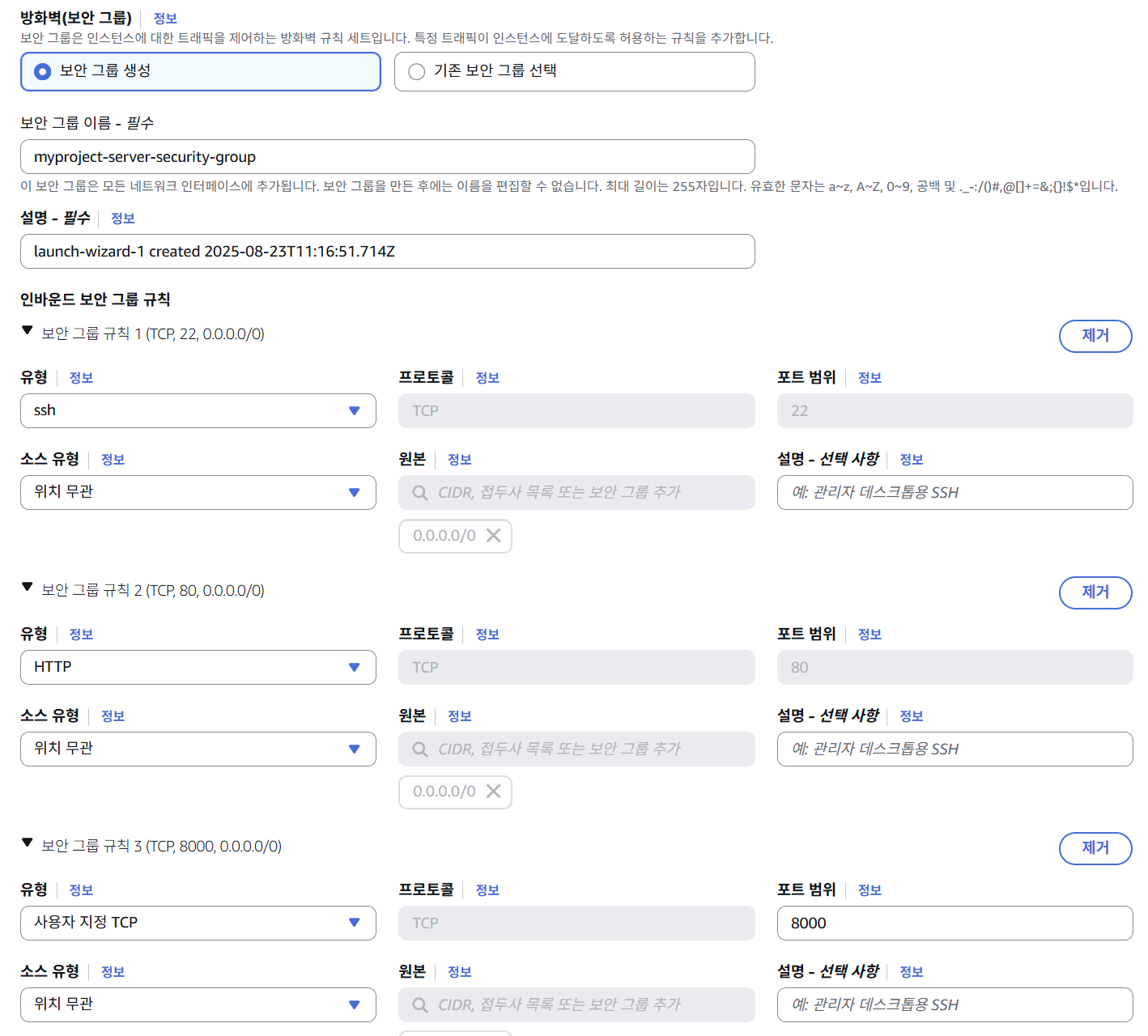
인터넷에서 일부 사용자가 EC2 인스턴스에 접근(액세스)하려고 한다고 가정해보자. 위 그림과 같이 EC2 인스턴스 주위에 방화벽 역할의 보안 그룹(Security Group)을 만들고 보안 그룹에 규칙을 지정한다. 이 보안 규칙에는 인바운드 트래픽(즉, 외부에서 EC2 인스턴스로 보내는 트래픽) 에서 어떤 트래픽만 허용할 지 설정할 수 있고, 아웃바운드 트래픽(즉, EC2 인스턴스에서 외부로 나가는 트래픽)에서 어떤 트래픽만 허용할 지 설정할 수 있다.

보안 그룹을 설정할 때는 허용할 IP 범위와 포트(port)를 설정할 수 있다

1. 보안 그룹(Security Group) 설정

외부에서 EC2로 접근할 포트는 22번 포트와 80번 포트라고 생각해서 이 2가지에 대해 인바운드 보안 그룹 규칙을 추가. 22번 포트는 우리가 EC2에 원격 접속할 때 사용하는 포트임.





* + 잘 알려진 포트(well-known port)란?

포트 번호는 0~65,535번까지 사용할 수 있다. 그 중에서 0~1023번까지의 포트번호는 주요 통신을 위한 규약에 따라 이미 정해져 있다. 이렇게 규약을 통해 역할이 정해져 있는 포트 번호를 보고 잘 알려진 포트(well-known port)라고 부른다.

규약이 정해져 있는 포트 중 자주 사용되는 포트번호

22(SSH, Secure Shell Protocal) : 원격 접속을 위한 포트번호

80(HTTP) : http 통신할 때 사용

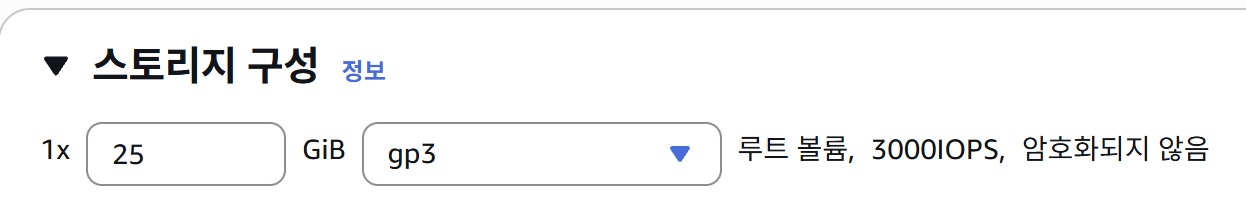
443(HTTPS) : https로 통신할 때 사용

물론 정해놓은 규약을 지키지 않아도 됨. http통신을 할 때 8000번 포트를 써도 무방

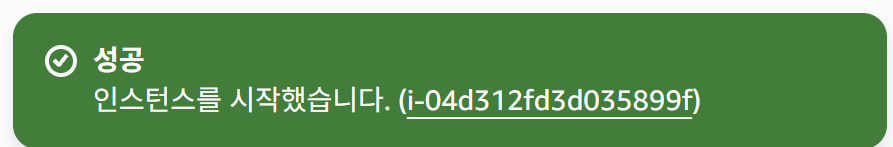
1. EC2 셋팅하기 – 스토리지 구성

우리가 쓰고 있는 데스크톱 컴퓨터는 전부 하드디스크를 가지고 있다. 하드디스크는 컴퓨터에서 파일을 저장하는 공간이다. EC2도 하나의 컴퓨터이다보니 여러 파일들을 저장할 저장 공간이 필요하다. 이 저장 공간을 보고

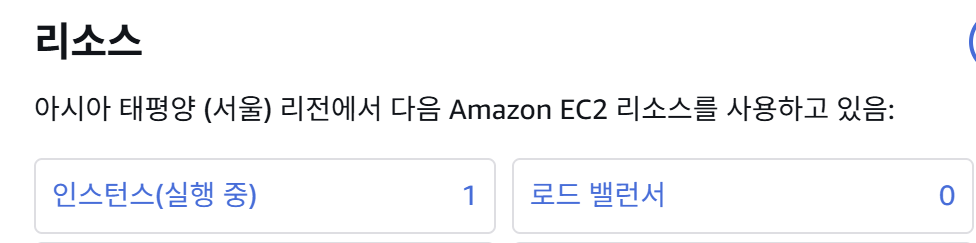
EBS(Elastic Block Storage)라고 부른다. 즉, EBS란 EC2 안에 부착되어 있는 일종의 하드디스크라고 생각하면 된다. EBS와 같은 저장 공간을 조금 더 포괄적인 용어로 스토리지(Storage), 볼륨(Volume)이라고 부른다.

 => 

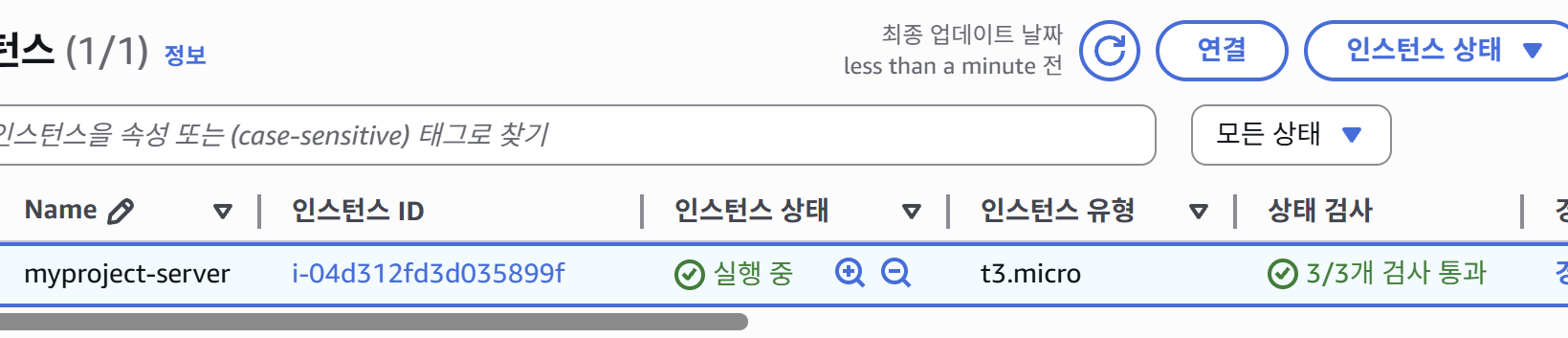
1. 모든 내용을 셋팅하고 인스턴스시작 클릭하면 다음 메시지 출력

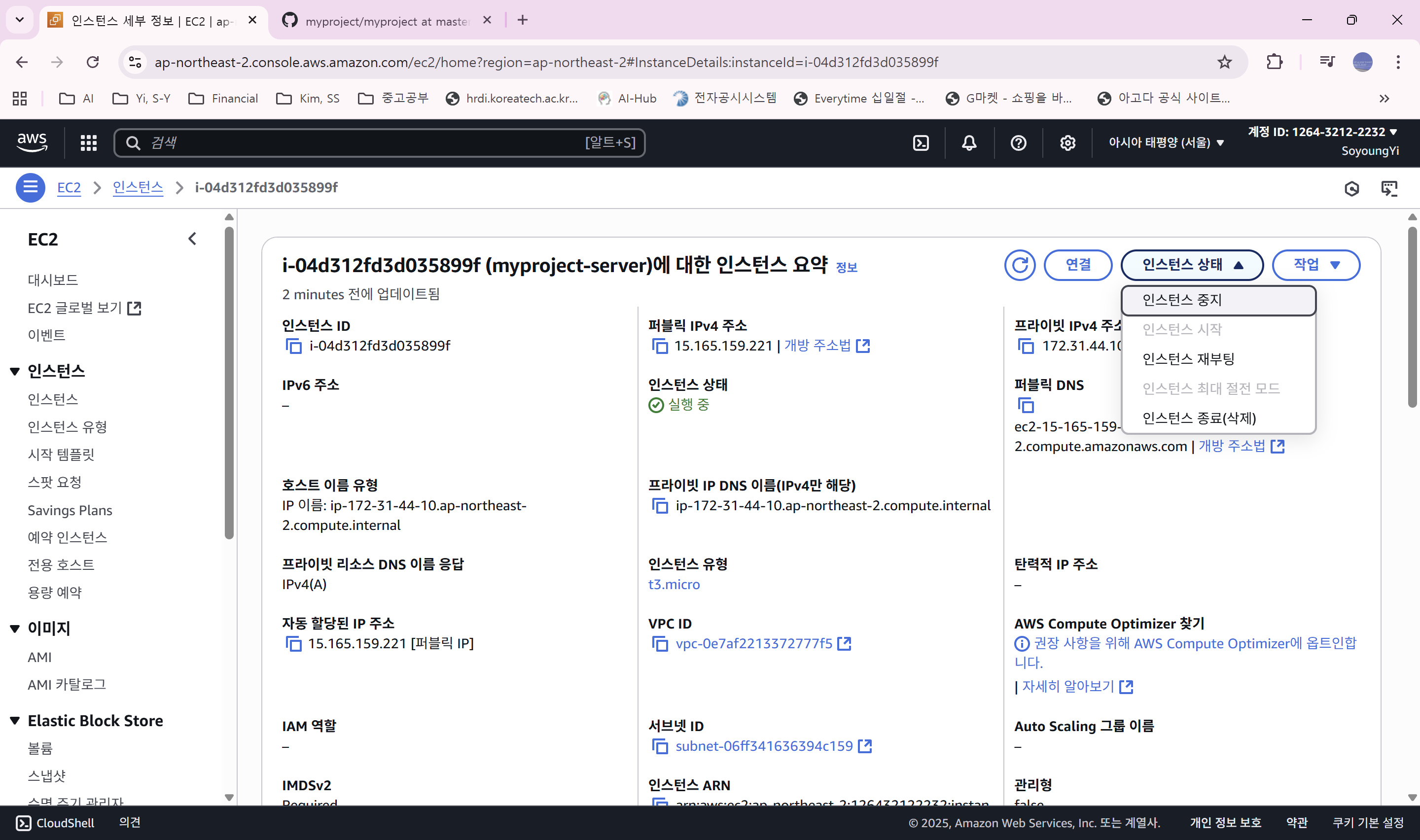


✅ EC2 대시보드 (<https://ap-northeast-2.console.aws.amazon.com/ec2/home?region=ap-northeast-2#Home>)

****

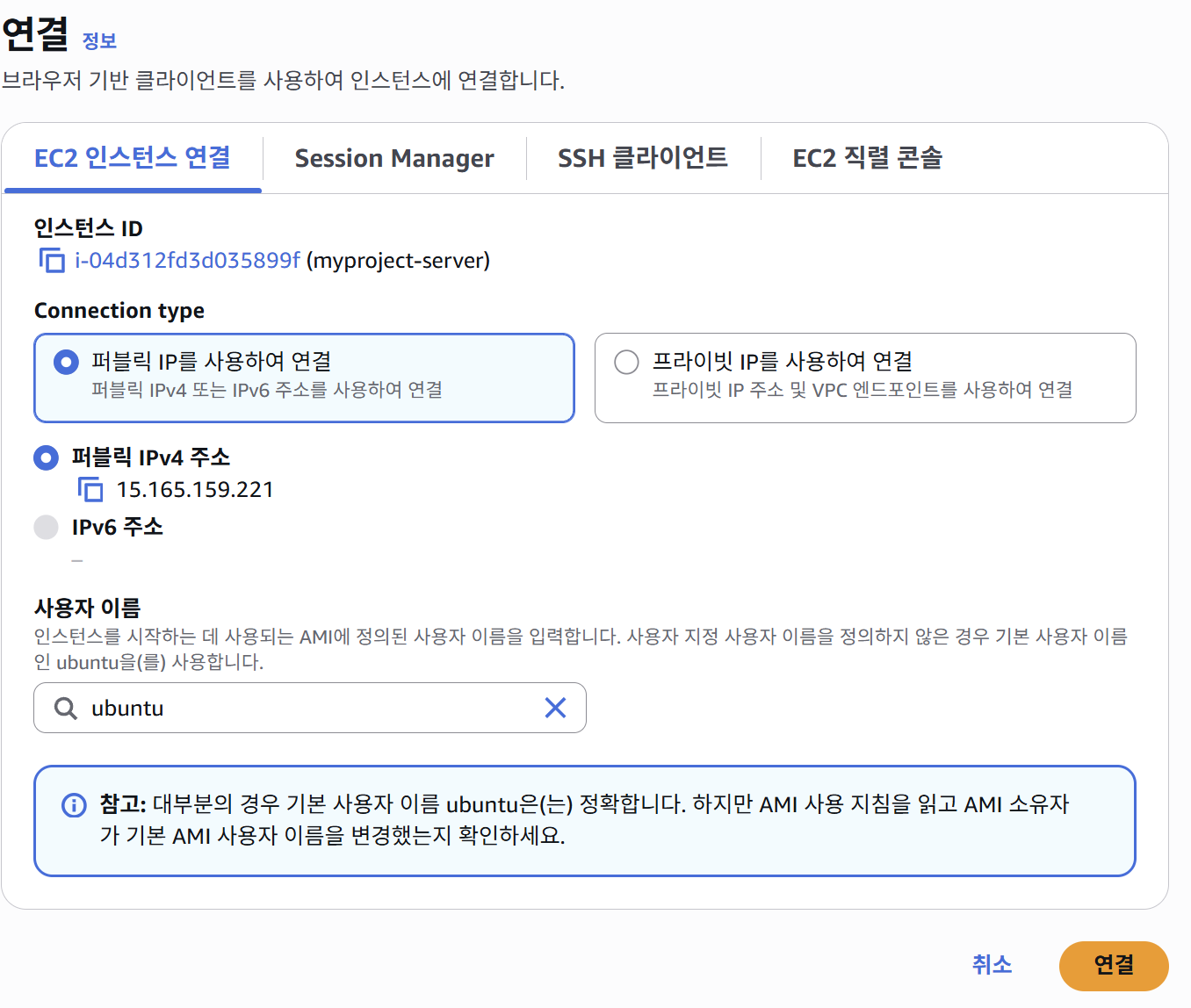
✅ **EC2 접속하기**





실제 서버를 운영할 때는 모니터링 탭을 자주 보게 됨

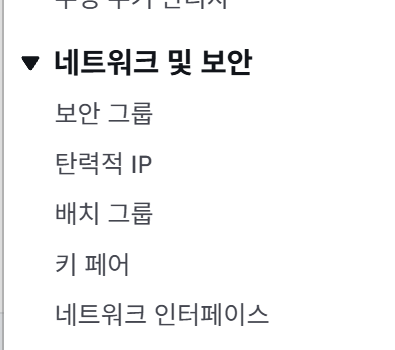
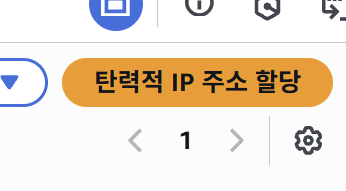
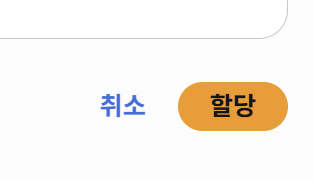
1. **연결클릭**

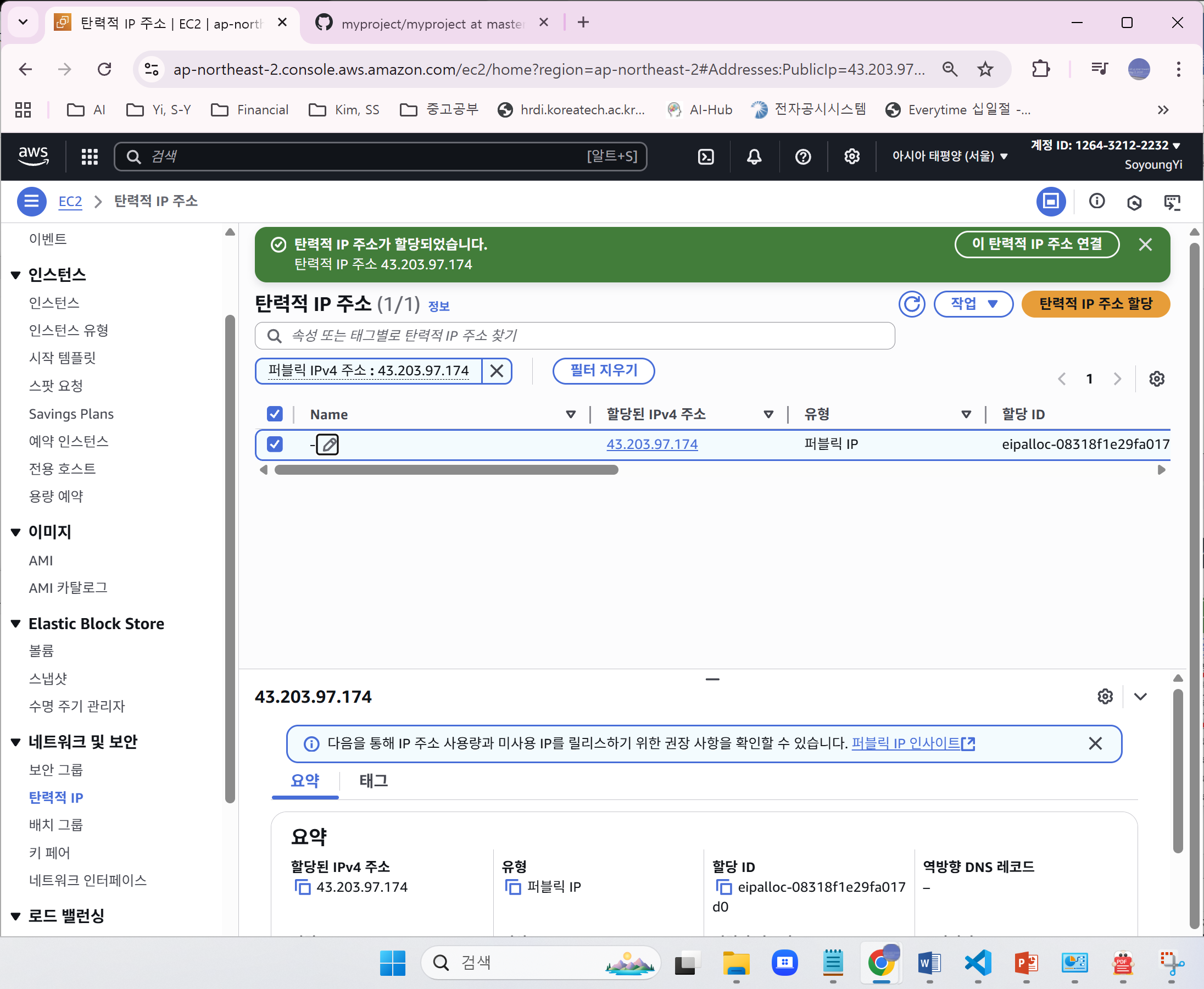
****

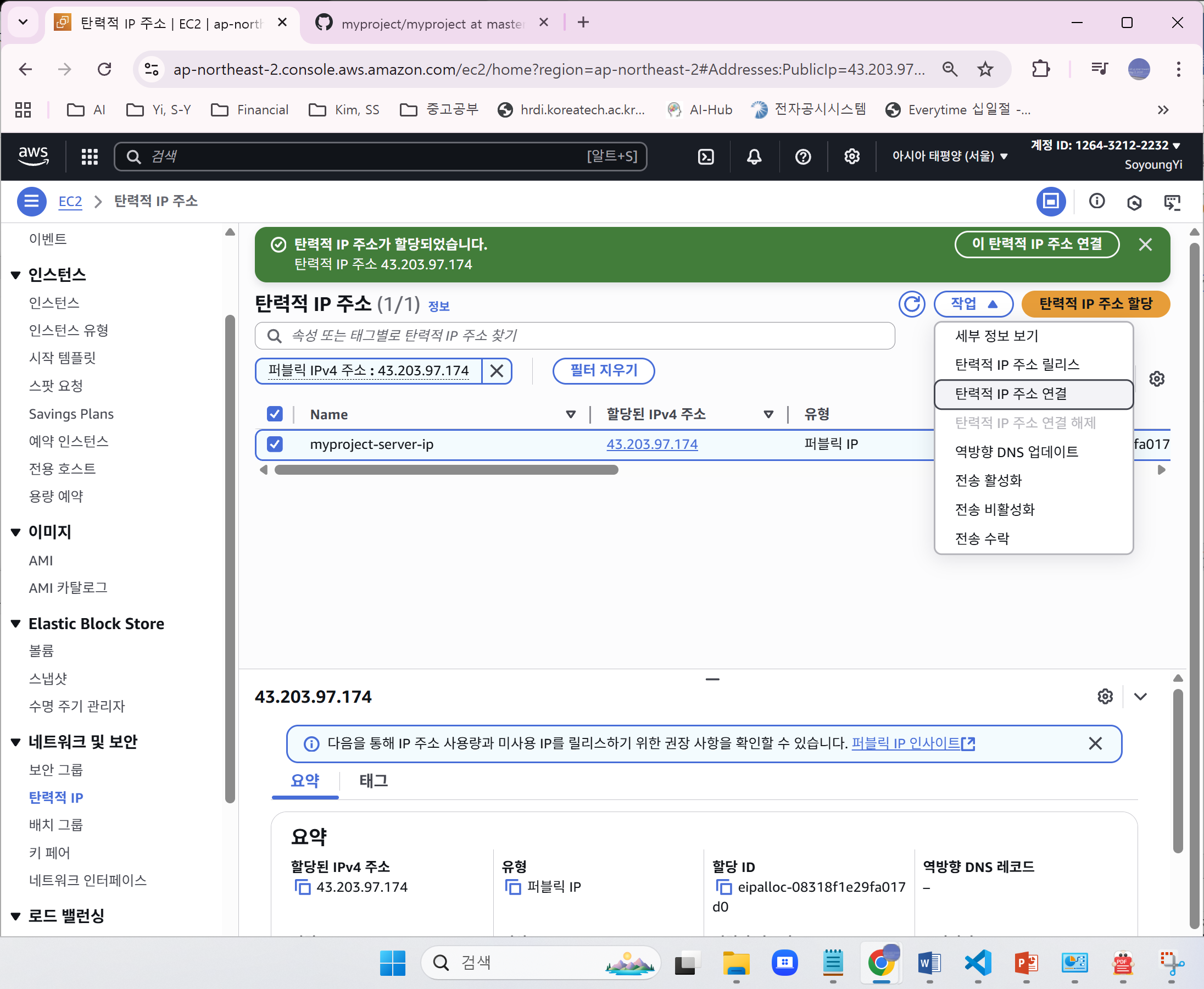
✅ 탄력적(Elastic) IP 연결하기

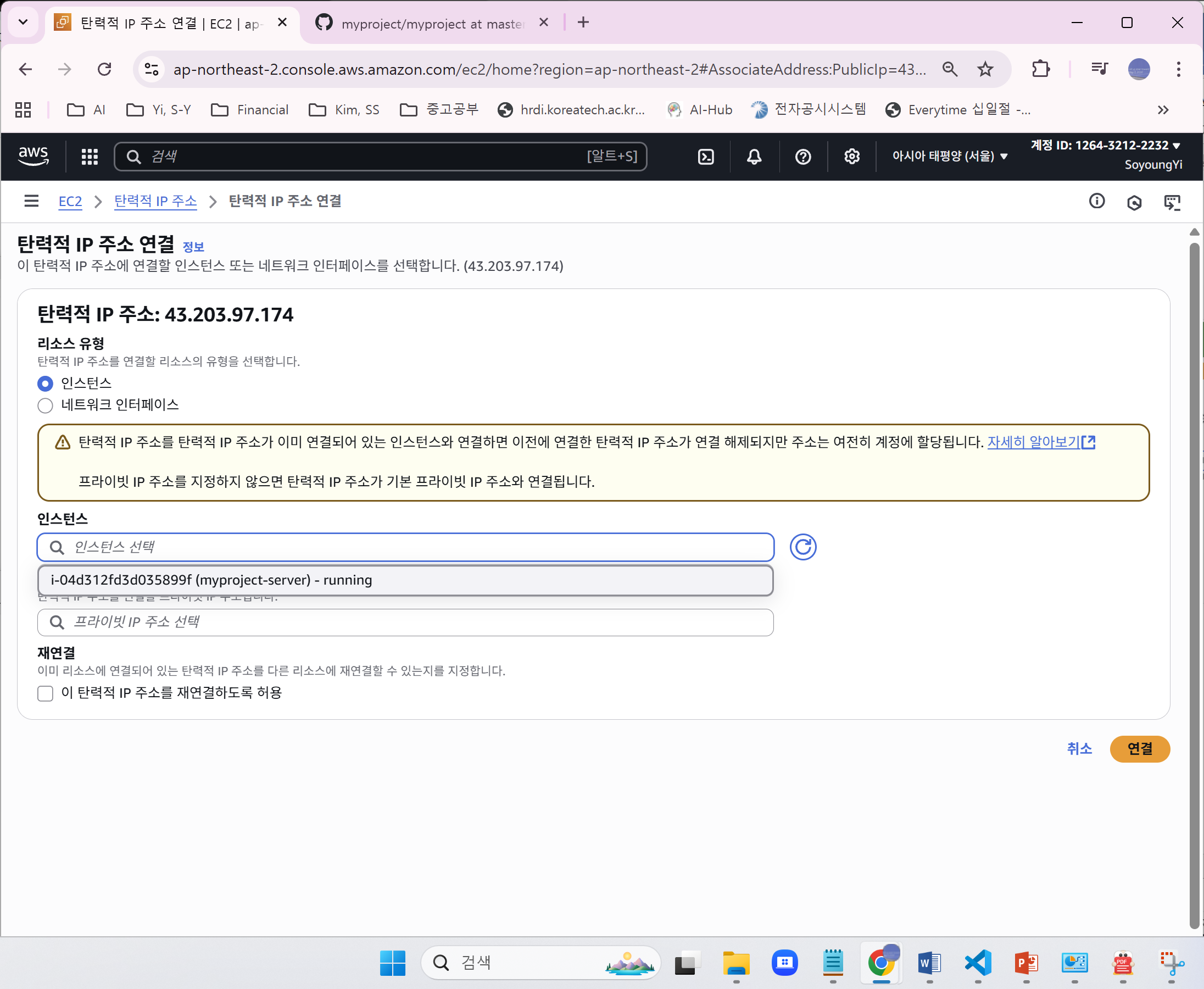
EC2 인스턴스를 생성하면 IP를 할당받는다 하지만 이렇게 할당받은 IP는 임시적인 IP이다. EC2 인스턴스를 잠깐 중지시켰다가 다시 실행시켜보면 IP가 바뀌어 있다. EC2 인스턴스를 중지시켰다가 다시 실행시킬때마다 IP가 바뀌면 굉장히 불편하다. 그래서 중지시켰다가 다시 실행시켜도 바뀌지 않는 고정 IP를 할당받아야 한다. 그것을 탄력적 IP라고 한다.

pwd, hostname -i

** =>  => **







✅ 장고 프로젝트에서

pip install whitenoise 한 후,

settings.py 에 ALLOWED\_HOSTS = [“\*”]

MIDDLEWARE = [

    …

    "whitenoise.middleware.WhiteNoiseMiddleware", 추가

]

pip freeze > requirements.txt

python manage.py collectstatic

✅ git repository 생성후 EC2에 배포(.gitignore에 .env와 .venv/추가)

✅ windows에서 SSH에 접속하는 방법은

1. PowerShell 또는 CMD로 접속(수업 시간에는 이 방법 사용할 예정)
2. PuTTY 사용(GUI 선호시)
3. WSL 사용시(Linux 환경 선호시)

✅ EC2 인스턴스에 SSH 접속(PEM 키 파일 권한 설정후 Powershell 관리자 권한으로 실행)

1. **Power shell 관리자 권한에서 아래 실행(PEM 파일 위치로 이동)**
2. **Pem key 권한 높이기**

**icacls "C:\Users\[사용자명]\.ssh\myproject-server-key-pair.pem" /inheritance:d**

**icacls "C:\Users\space\.ssh\myproject-server-key-pair.pem" /grant:r "%username%:F"**

1. SSH 접속

ssh -i "C:\Users\[사용자명]\.ssh\myproject-server-key-pair.pem" ubuntu@[EC2-퍼블릭-IP]

처음 접속 시 Are you sure you want to continue connecting (yes/no/[fingerprint])? yes 입력

✅ 시스템 초기 설정

1. Ubuntu 서버 초기 설정(시스템 업데이트)

sudo apt upgrade -y

1. Python 3.10 저장소 추가

*# 필요한 패키지 설치*

sudo apt install software-properties-common -y

*# deadsnakes PPA 추가* (Python 3.10용)

sudo add-apt-repository ppa:deadsnakes/ppa

1. sudo apt update
2. Python 3.10 및 필수 패키지 설치

*# Python 3.10과 관련 패키지들 설치*

sudo apt install python3.10 python3.10-venv python3.10-dev

python3.10 --version

*# pip 수동 설치*

python3.10 -m ensurepip --upgrade

*# python3 명령어가 3.10을 가리키도록 설정*

sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.10 1

*# 설치 확인*

python3 --version *# Python 3.10.x가 출력되어야 함*

python3.10 -m pip --version

1. 필수 소프트웨어 설치

*# 컴파일 도구와 웹서버 설치 (SQLite는 Django에 기본 포함)*

sudo apt install build-essential git nginx

*# 설치 확인*

git --version

nginx -v

gcc --version *# 컴파일 도구 확인*

1. pwd프로젝트 디렉토리 생성

pwd 결과가 /home/ubuntu가 아닐 경우 cd /home/ubuntu

mkdir myproject # 폴더 생성 (反 rm -rf myproject는 폴더 삭제)

ls # dir 역할

cd myproject

1. Git 에서 프로젝트 클론

git clone 내깃주소 **.**

1. 가상환경 생성 및 활성화

python3.10 -m venv .venv

# 권한 문제로 source붙여야 함

source .venv/bin/activate

# 활성화 확인(프롬프트에 .venv가 시작되는지도 봄) .venv폴더가 숨긴폴더여서 ls -la

which python

python –V

1. 의존성 설치

pip install --upgrade pip

pip install –r requirements.txt

1. .env파일 생성

# nano .env 해서 파일을 아래와 같이 입력하고 저장(Ctrl+o), 엔터, 종료(ctrl+x)

SECRET\_KEY=

EMAIL\_HOST\_PASSWORD=

EMAIL\_HOST\_USER=

ALLOWED\_HOSTS= 43.203.97.174,localhost

# 확인

cat .env

1. settings.py 파일에 있어야 할 내용

from decouple import config

import os

SECRET\_KEY = config('SECRET\_KEY')

ALLOWED\_HOSTS = config('ALLOWED\_HOSTS', default='').split(',') 또는 [“\*”]

# SQLite 데이터베이스 설정 (기본값 - 수정할 필요 없음)

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

# 정적 파일 설정

STATIC\_URL = '/static/'

STATIC\_ROOT = os.path.join(BASE\_DIR, '\_staticfiles')

1. Django 애플리케이션 초기 설정

# db.sqlite3파일 확인 및 정적 파일 수집 확인, 관리자 계정 확인

ls -la db.sqlite3

없으면 python manage.py makemigrations와 migrate실행

python manage.py collectstatic

python manage.py shell

User.objects.first()

1. 개발 서버 테스트

python manage.py runserver 0.0.0.0:8000

*# 브라우저에서 http://EC2\_PUBLIC\_IP:8000 접속 확인* *# Ctrl+C로 서버 종료*

1. SSH를 닫아도 계속 서버 연결

*# screen 설치*

sudo apt install screen -y

*# 2. screen 세션 생성*

screen -S django

*# 3. 가상환경 활성화 후 서버 실행*

source .venv/bin/activate

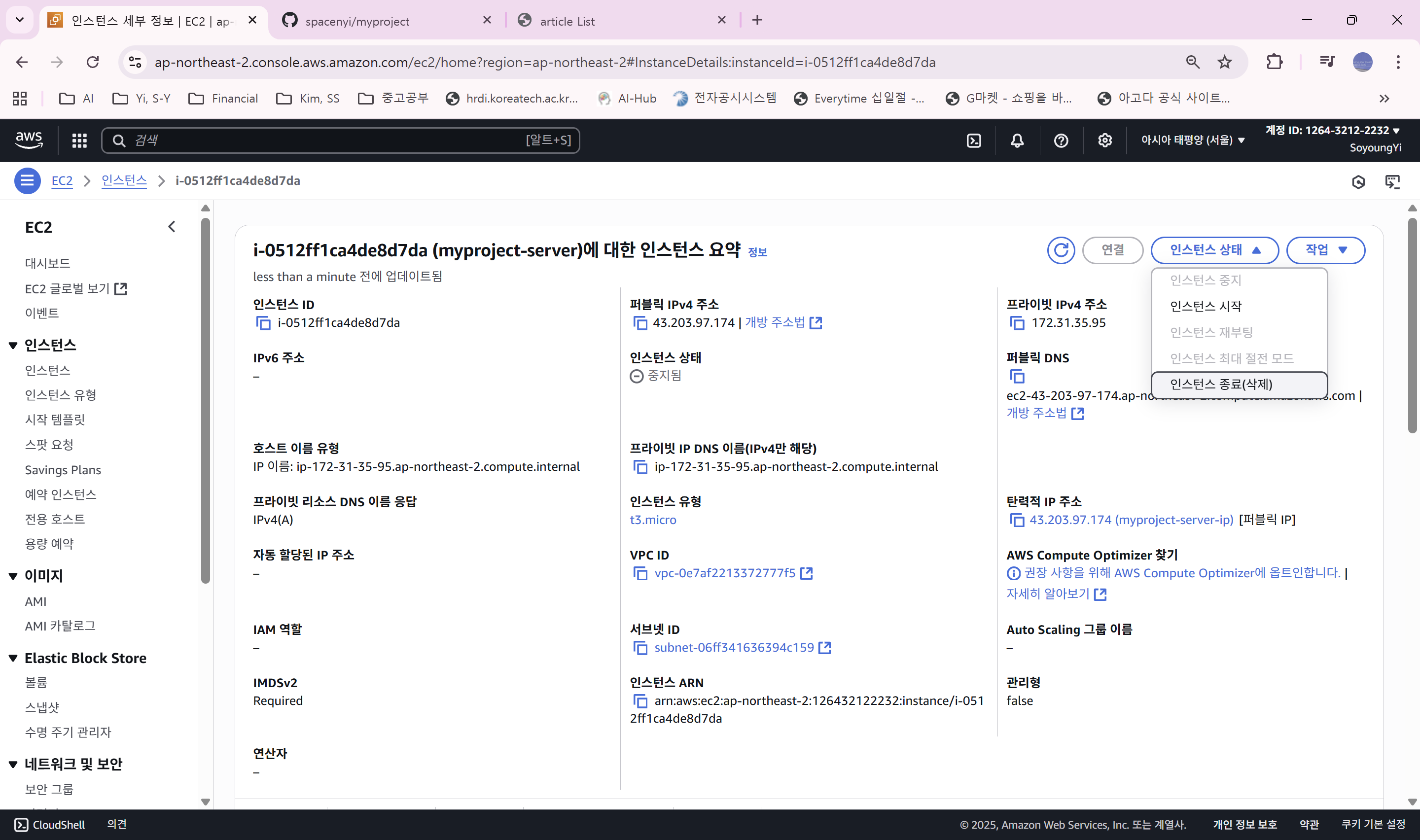
python manage.py runserver 0.0.0.0:8000

*# 4. Ctrl+A, D로 빠져나오기 (서버는 계속 실행)*

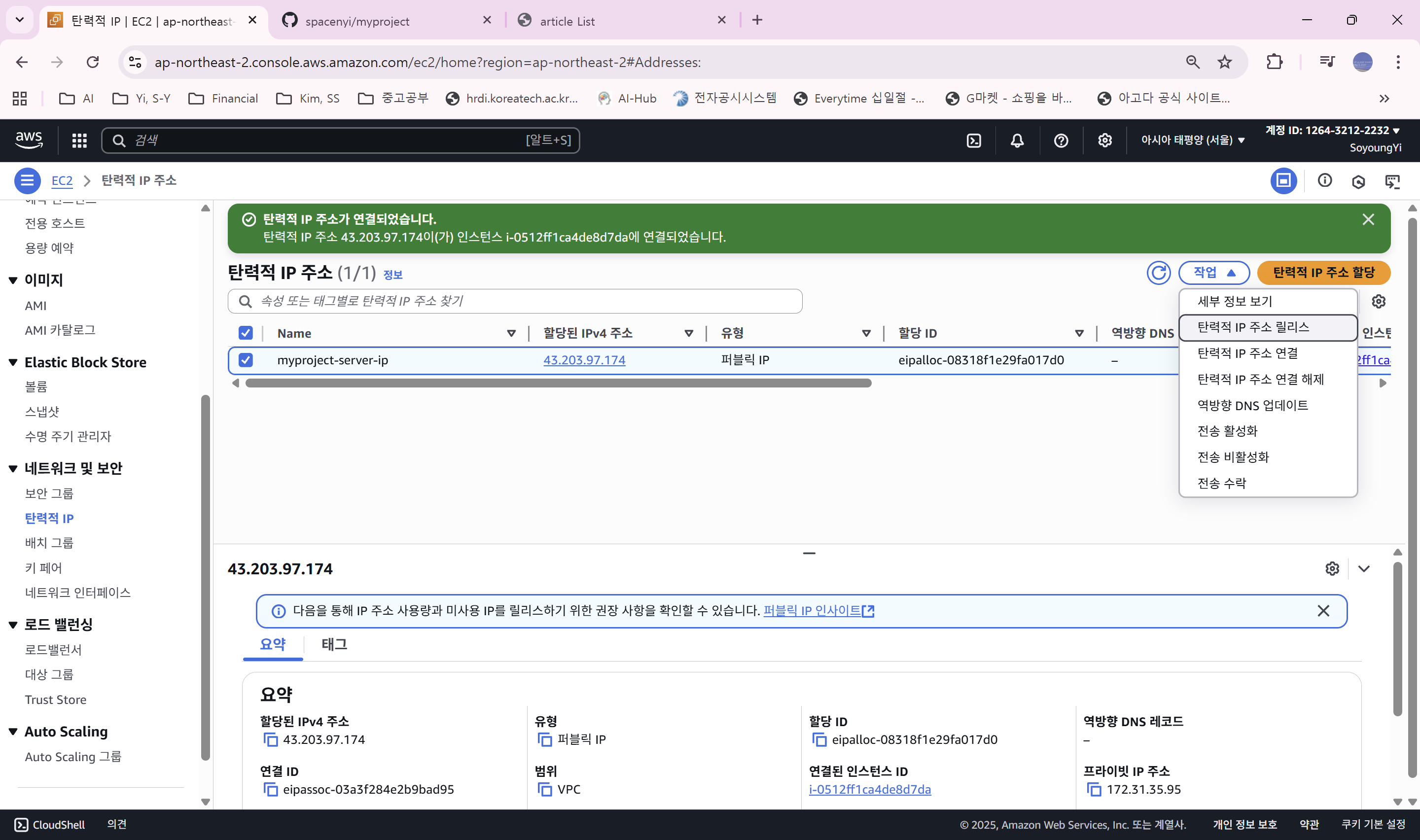
*# 5. 나중에 다시 접속하려면*

screen -r django

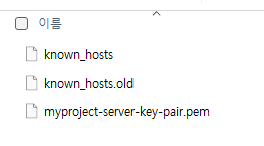
**✅ 과금 방지를 위한 인스턴스 종료, 탄력적 ip 릴리즈, 보안그룹, 데스크톱의 pem 파일 삭제, 웹에서 키페어 삭제**



1. 탄력적 ip 주소 연결 해제 -> 탄력적 ip 주소 릴리스



1. **c:\users\사용자명\.ssh\폴더에 있는 파일들 삭제**

 **모두 삭제**

1. 키페어 삭제

