

Django

Django 서버 배포

django 추가코드

0. settings.py

- staticfiles 경로 추가
 - `STATICFILES_DIRS` 을 사용하는 경우 동일한 경로 사용불가(이름변경 ex. `staticfiles=>static`)

```
# settings.py
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
```

- ALLOWED_HOSTS
 - EC2 서버주소를 등록
 - 편하게 배포하기 위하여 * 로 등록 후 추후 수정가능

```
# settings.py
ALLOWED_HOSTS = [
    '.compute.amazonaws.com',
    '*',
]
```

1. 의존성 저장

- freeze

```
pip freeze > requirements.txt
```

2. git push

- 원격저장소에 업로드 (add, commit, push)

0. 준비

- 완성된 django프로젝트
- 해외결제가 가능한 체크카드 or 신용카드
- 여유로운 마음

1. <https://aws.amazon.com/ko/>

- AWS 계정 생성
- 기본정보입력
- 카드정보입력
- 휴대폰인증
- 완료후 로그인

2. aws cloud9

- AWS Management Console 에서 Cloud9 검색

AWS Management Console

AWS 서비스

서비스 찾기

이름, 키워드 또는 약어를 입력할 수 있습니다.

Cloud9

코드의 작성, 실행 및 디버깅을 위한 클라우드 IDE

▼ 최근 방문한 서비스

 Cloud9

 AWS Cloud Map

▼ 전체 서비스

컴퓨팅

EC2

Lightsail 

Lambda

Batch

Elastic Beanstalk

블록체인

Amazon Managed Blockchain

위성

Ground Station

보안, 자격 증명 및 규정 준수

IAM

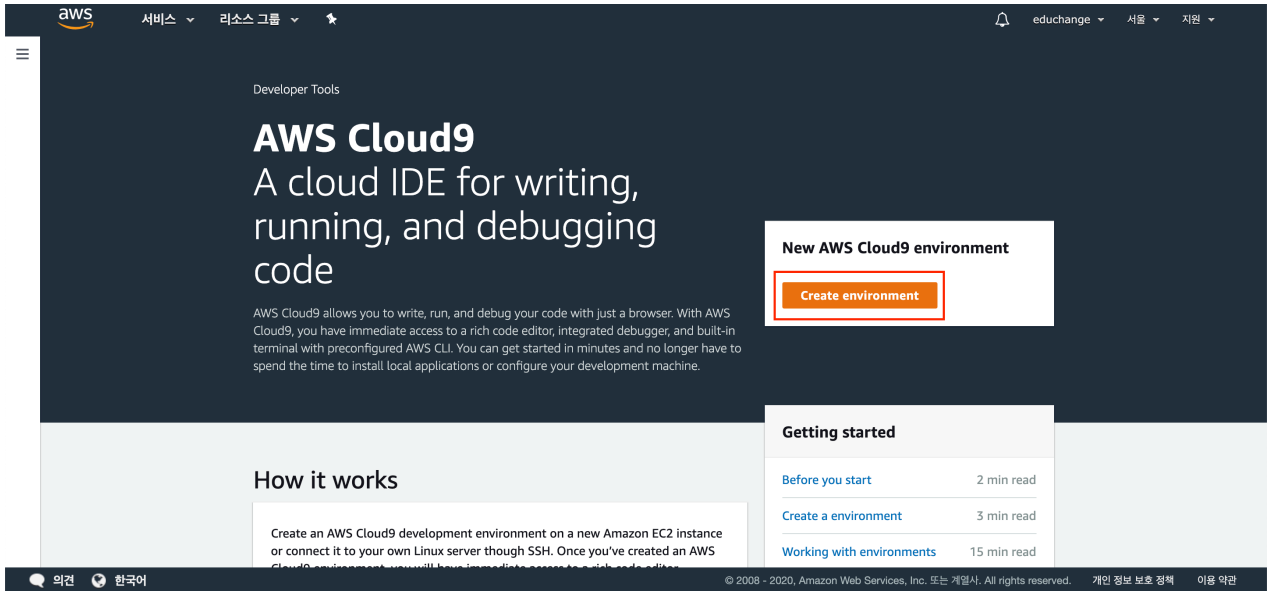
Resource Access Manager

Cognito

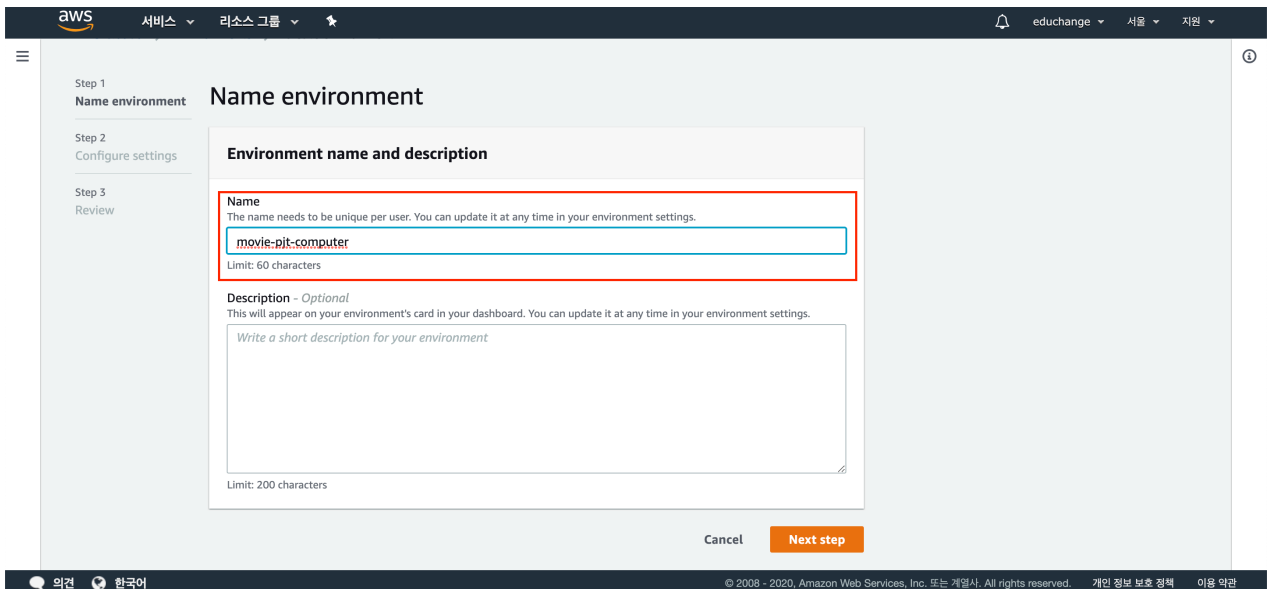
Secrets Manager

GuardDuty

- 생성



- 이름입력



- 설정

- 서버를 사용하지 않을 때 끄려면 default인 30분으로 설정

Platform

- ☐ Amazon Linux
- ☒ Ubuntu Server 18.04 LTS

Cost-saving setting

Choose a predetermined amount of time to auto-hibernate your environment and prevent unnecessary charges. We recommend a hibernation settings of half an hour of no activity to maximize savings.

Never

IAM role

AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. [Learn more](#)

AWSServiceRoleForAWSCloud9

► Network settings (advanced)

No tags associated with the resource.

Add new tag

You can add 50 more tags.

● 생성완료

AWSServiceRoleForAWSCloud9 (generated)

Tags (0)

Q Search tags

Key



Value



No tags



We recommend the following best practices for using your AWS Cloud9 environment

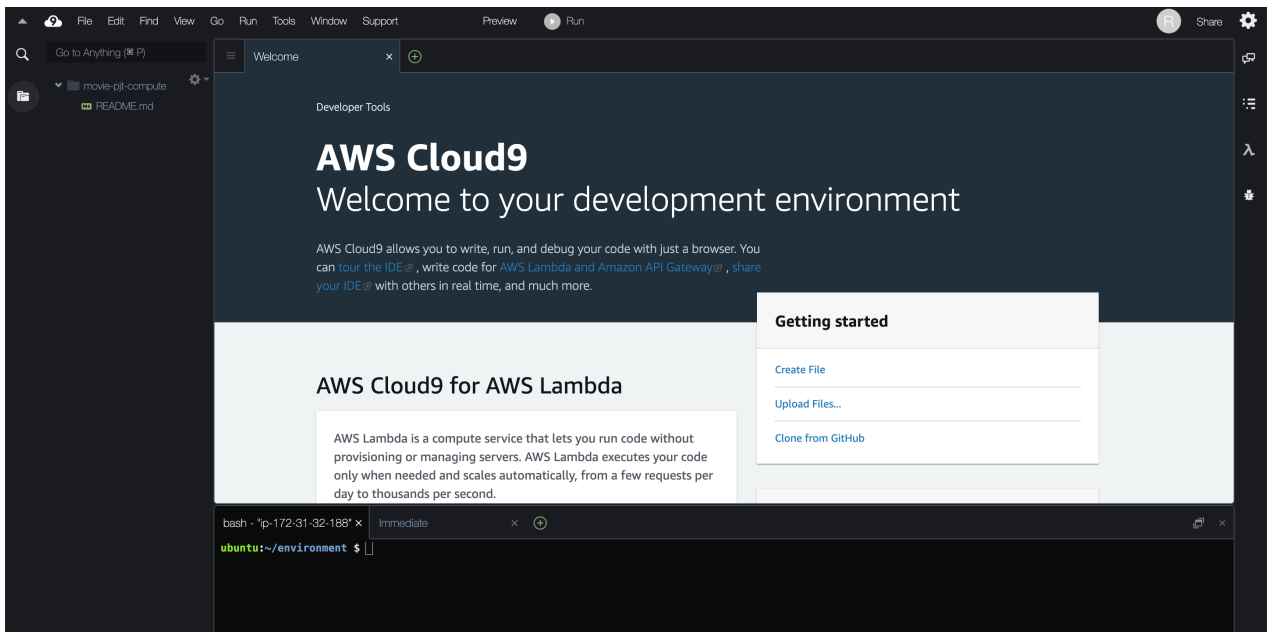
- Use **source control and backup** your environment frequently. AWS Cloud9 does not perform automatic backups.
- Perform regular **updates of software** on your environment. AWS Cloud9 does not perform automatic updates on your behalf.
- **Turn on AWS CloudTrail in your AWS account** to track activity in your environment. [Learn more](#)
- Only share your environment with **trusted users**. Sharing your environment may put your AWS access credentials at risk. [Learn more](#)

Cancel

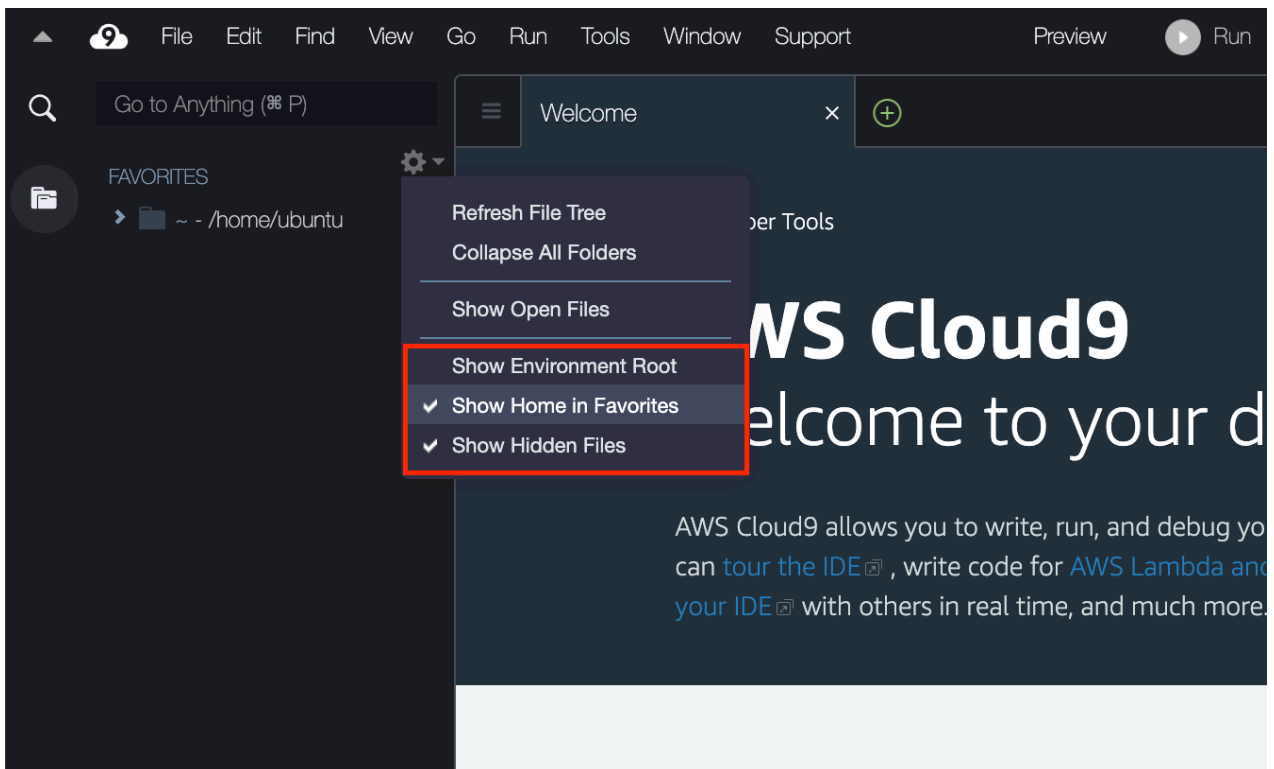
Previous step

Create environment

● cloud9확인



● 파일트리설정

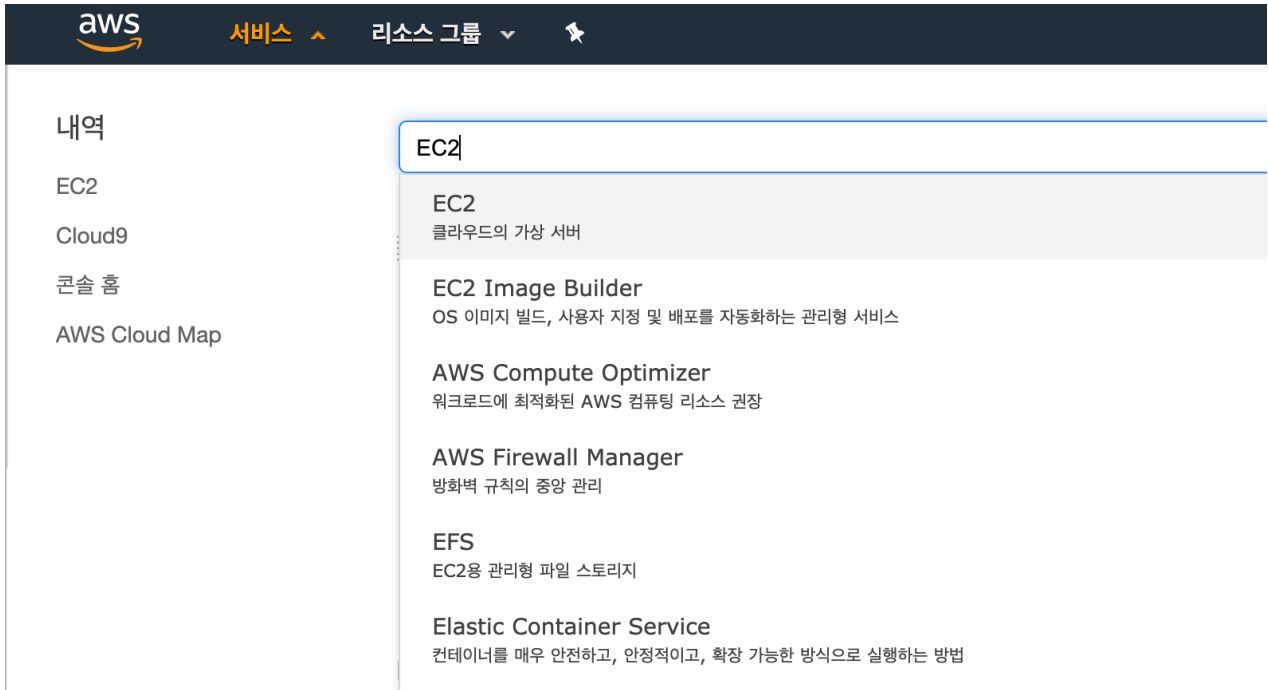


EC2

새로운 탭에서 진행

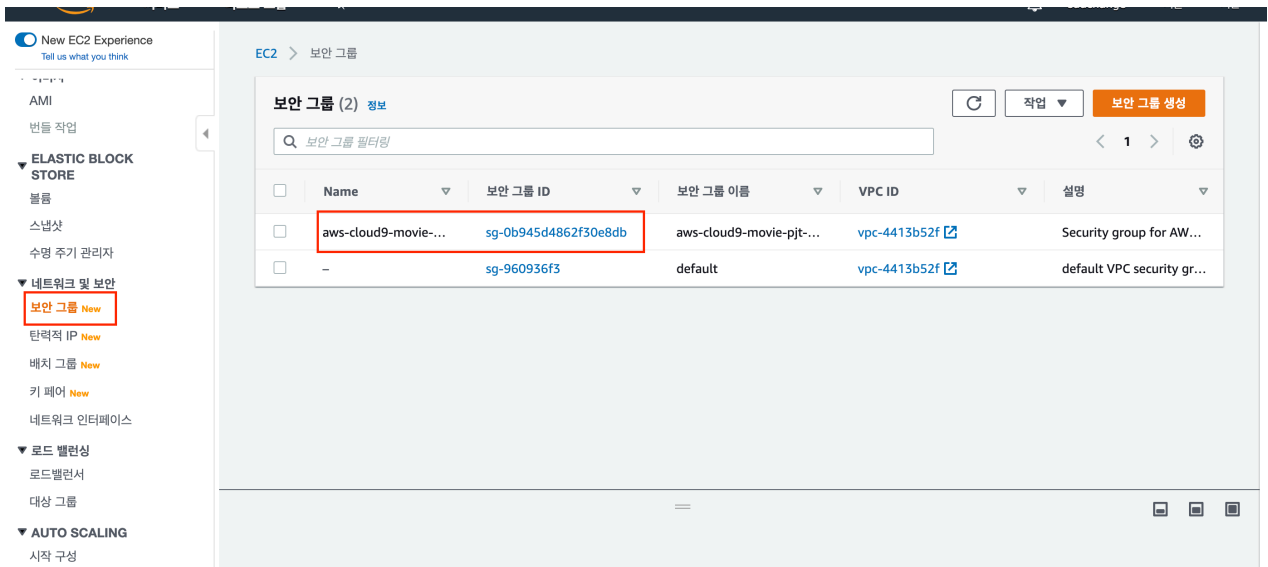
EC2는 cloud9 생성시 자동생성

0. 서비스검색



1. 보안그룹

- 생성된 ID 클릭



2. 인바운드 설정

- 편집

인바운드 규칙

아웃바운드 규칙

태그

인바운드 규칙

인바운드 규칙 편집

유형	프로토콜	포트 범위	소스	설명 - 선택 사항
SSH	TCP	22	15.164.243.192/27	-
SSH	TCP	22	15.164.243.32/27	-

인바운드 규칙

정보

유형 정보

프로토콜 정보

포트 범위 정보

소스 정보

설명 - 선택 사항 정보

SSH

TCP

22

사용자 ...

15.164.243.192/27

삭제

SSH

TCP

22

사용자 ...

15.164.243.32/27

삭제

사용자 지정 TCP

TCP

80

위치 무관

0.0.0.0/0

::/0

삭제

규칙 추가

참고: 기존 규칙을 편집하면 편집된 규칙이 삭제되고 새 세부 정보로 새 규칙이 생성됩니다. 이렇게 하면 새 규칙이 생성될 때까지 해당 규칙에 의존하는 트래픽이 잠시 중단될 수 있습니다.

취소

변경 사항 미리 보기

규칙 저장

python

0. pyenv & pyenv-virtualenv

- 설치 & 설정
 - 전체 복사 후 터미널에서 실행

```
git clone https://github.com/pyenv/pyenv.git ~/.pyenv
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
echo -e 'if command -v pyenv 1>/dev/null 2>&1; then\n  eval "$(pyenv init -)\nfi' >> ~/.bashrc
exec "$SHELL"

git clone https://github.com/pyenv/pyenv-virtualenv.git $(pyenv
root)/plugins/pyenv-virtualenv
echo 'eval "$(pyenv virtualenv-init -)"' >> ~/.bashrc
exec "$SHELL"
```

1. python 설치&전역등록

프로젝트 진행한 버전에 맞게 설치

```
pyenv install 3.7.4
pyenv global 3.7.4
python -V
#=> Python 3.7.4
```

2. 가상환경생성

가상환경이름 기억!

```
pyenv virtualenv {가상환경이름}
```

```
ubuntu:~ $
ubuntu:~ $ pyenv virtualenv myenv
Looking in links: /tmp/tmpont8r9ba
Requirement already satisfied: setuptools in /home/ubuntu/.pyenv/versions/3.7.4/lib/python3.7/site-packages
Requirement already satisfied: pip in /home/ubuntu/.pyenv/versions/3.7.4/lib/python3.7/site-packages
ubuntu:~ $ pyenv versions
system
* 3.7.4 (set by /home/ubuntu/.pyenv/version)
  3.7.4/envs/myenv
  myenv
ubuntu:~ $
```

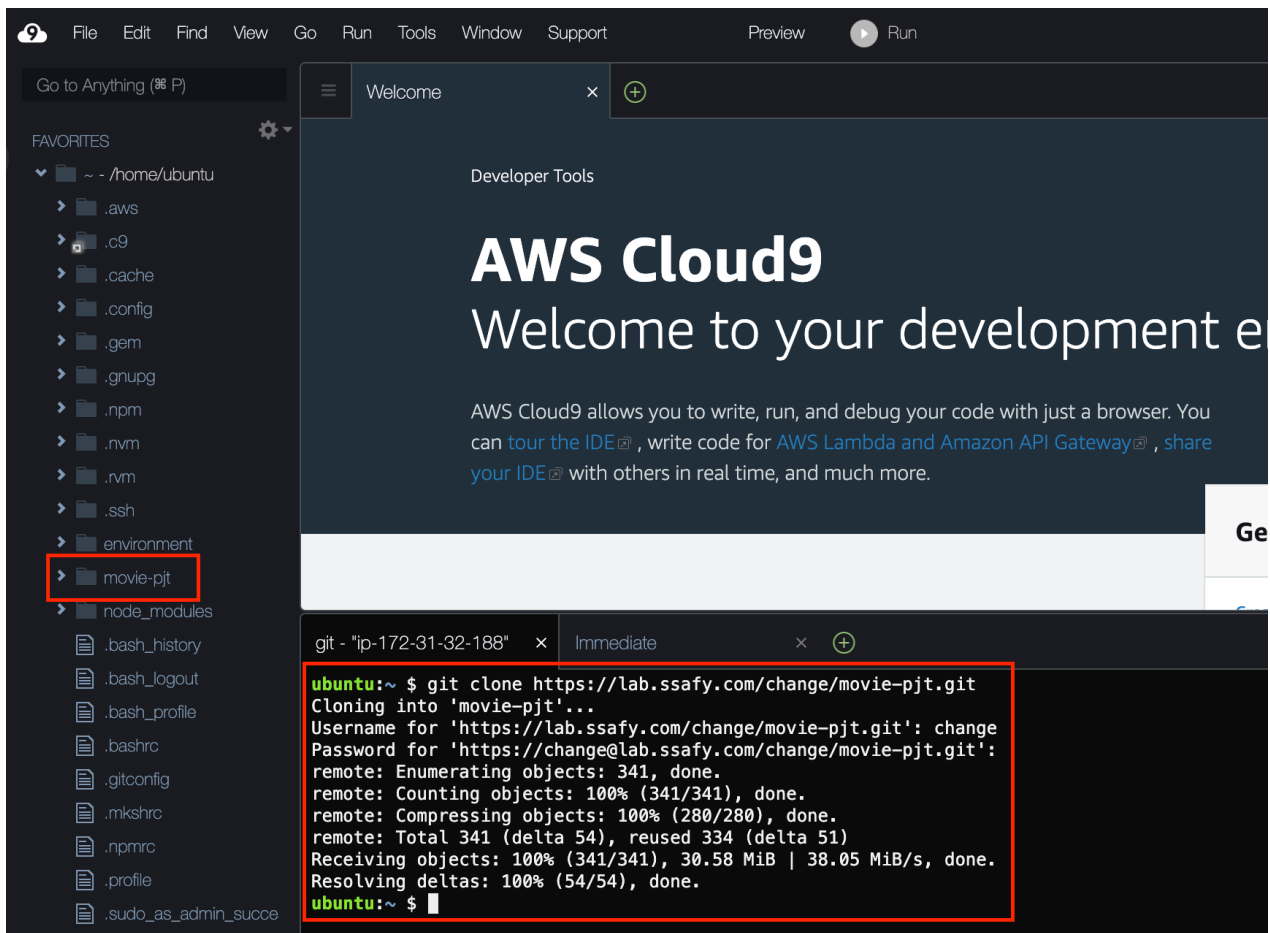

project clone

루트폴더와 프로젝트, 두개의 폴더 이름에 주의하며 진행해주세요. 두 폴더의 이름을 통일하면 조금더 편하게 설정할 수 있습니다.

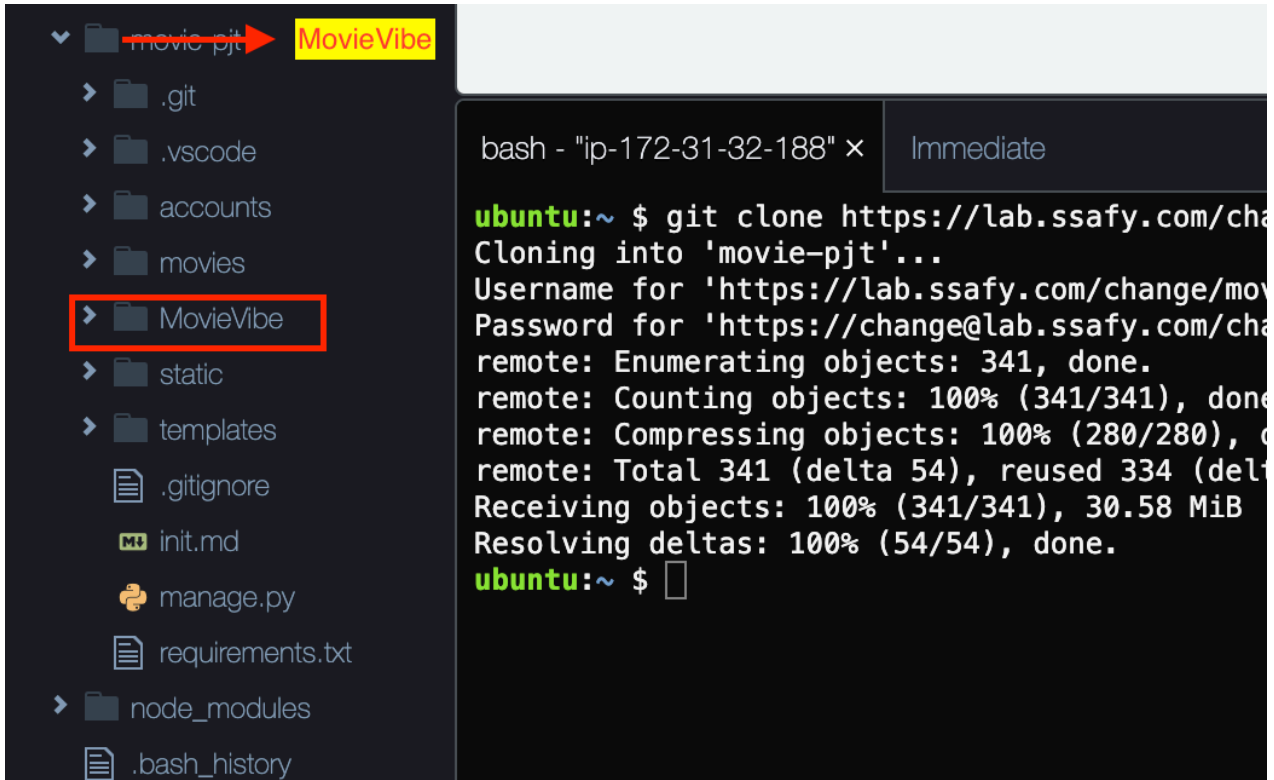
0. 준비

- ~로 이동 => `cd ~`
 - 명령어를 작성하는 위치 주의!
- clone

```
git clone {project_remote_url}
```



- 편의를 위해 폴더명 변경
 - 프로젝트 전체 폴더의 이름을 프로젝트이름과 동일하게 변경



- 폴더구조
 - 프로젝트이름은 변수처럼 사용예정 이름 기억!

```
home
  ubuntu
    {프로젝트이름}
      {프로젝트이름}
        {앱}
          manage.py
```

- 클론한 폴더로 이동

```
cd {프로젝트이름}
```

- 가상환경 활성화

```
pyenv local {가상환경이름}
```

```
bash - "ip-172-31-32-188" ×
```

Immediate

```
ubuntu:~/MovieVibe $ pyenv local myenv  
(myenv) ubuntu:~/MovieVibe $ █
```

- 라이브러리 설치

```
pip install -r requirements.txt
```

- 마이그레이션

```
python manage.py migrate
```

- createsuperuser

```
python manage.py createsuperuser
```

- loaddata (fixture가 있는경우)

```
python manage.py loaddata {data.json}
```

- collectstatic

```
python manage.py collectstatic
```

nginx

0. 설치

```
sudo apt-get update  
sudo apt-get install -y nginx
```

1. 설정

vi를 사용하여 터미널에서 파일을 수정합니다. 사용법을 숙지하고 진행해주세요.

- 복사할 코드 미리 작성하기
 - 아래의 코드에서 각자의 프로젝트이름에 맞게 수정 후 아래에 붙여넣기
 - staticfiles의 경우 다른 폴더를 썼다면 이름수정

```
server_name *.compute.amazonaws.com;

location / {
    uwsgi_pass unix:///home/ubuntu/{프로젝트이름}/tmp/{프로젝트이름}.sock;
    include uwsgi_params;
}

location /static/ {
    alias /home/ubuntu/{프로젝트이름}/staticfiles/;
}
```

- 결과 붙여넣기 위한 빈칸

- 파일 수정

```
sudo vi /etc/nginx/sites-enabled/default
```

- 아래의 표시된 부분 수정
 - `i` 버튼으로 수정모드로 전환
 - 아래의 부분으로 방향키를 이용하여 이동
 - 수정
 - `esc` 로 수정모드 빠져나오기
 - `:wq` 명령어로 저장 후 종료

```
#
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }

    # pass PHP scripts to FastCGI server
    #

```

- 수정 결과
 - 주석부분을 지워도 무방

```
#server_name _;

#location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    #
    try_files $uri $uri/ =404;
#}

server_name *.compute.amazonaws.com;

location / {
    uwsgi_pass unix:///home/ubuntu/MovieVibe/tmp/MovieVibe.sock;
    include uwsgi_params;
}

location /static/ {
    alias /home/ubuntu/MovieVibe/staticfiles/;
}

```

uWSGI

0. 설치

```
pip install uwsgi
```

1. 폴더&파일 생성

- 프로젝트 폴더 이동 (기존의 위치와 동일)

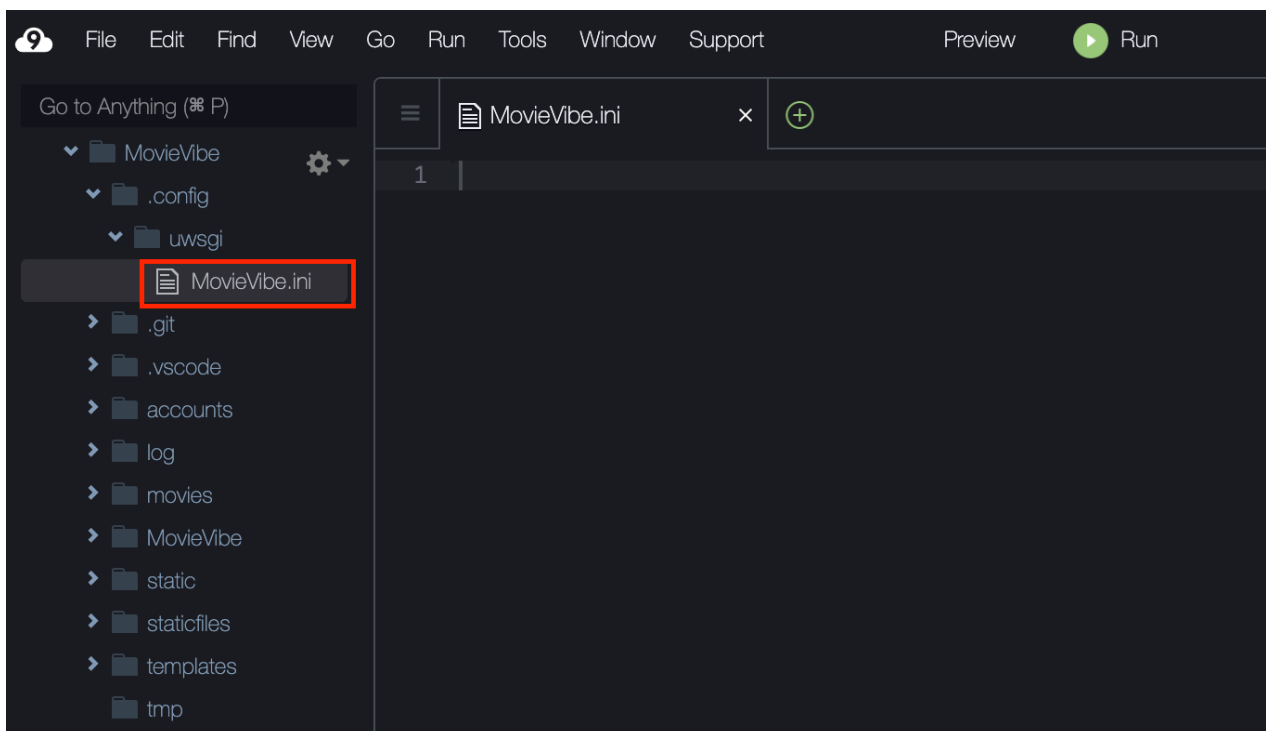
```
cd ~/ {프로젝트이름}
```

- uwsgi 설정, 로그 저장할 폴더 생성 (파일트리에서 생성해도 무방)

```
mkdir tmp
mkdir -p log/uwsgi
mkdir -p .config/uwsgi/
```

- uwsgi 설정파일 생성 (파일트리에서 생성해도 무방)

```
touch .config/uwsgi/{프로젝트이름}.ini
```



3. 수정

- `.config/uwsgi/{프로젝트이름}.ini` 설정파일 수정

```
# {프로젝트이름}/.config/uwsgi/{프로젝트이름}.ini

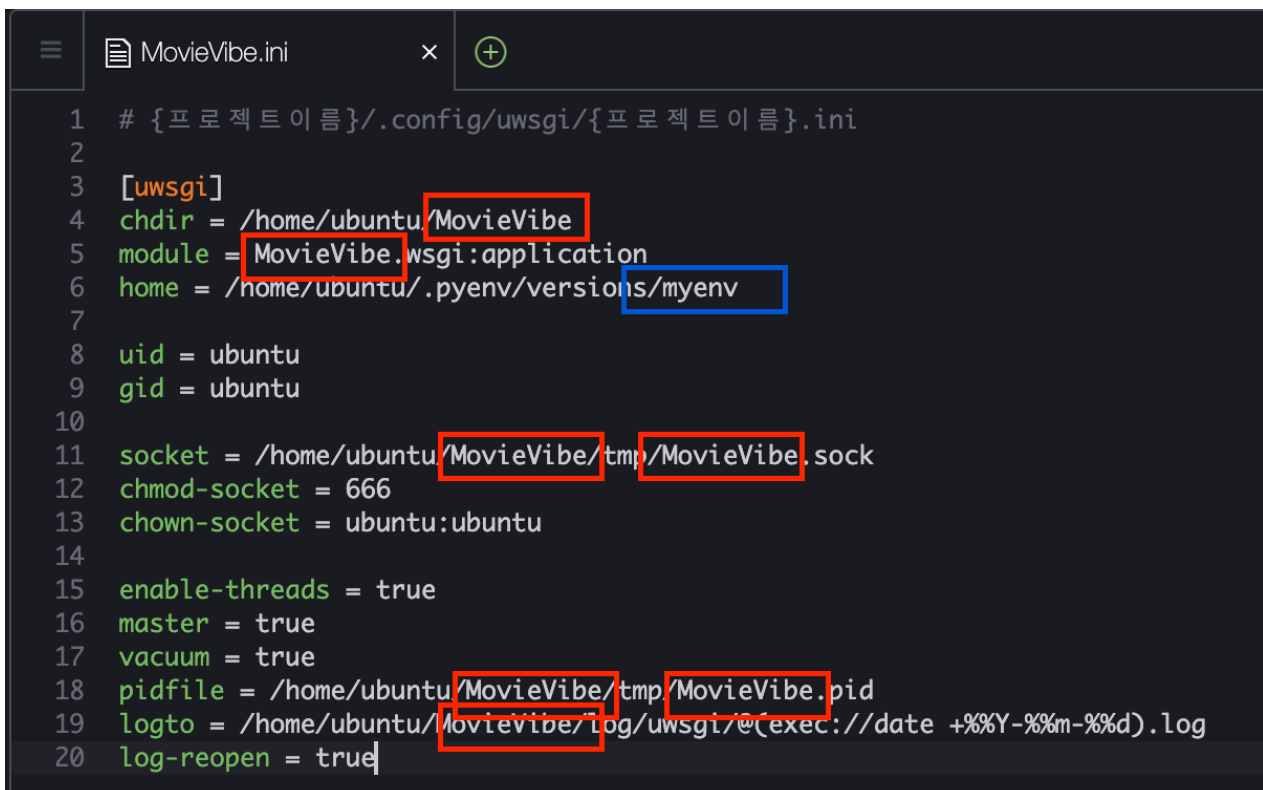
[uwsgi]
chdir = /home/ubuntu/{프로젝트이름}
module = {프로젝트이름}.wsgi:application
home = /home/ubuntu/.pyenv/versions/{가상환경이름}

uid = ubuntu
gid = ubuntu

socket = /home/ubuntu/{프로젝트이름}/tmp/{프로젝트이름}.sock
chmod-socket = 666
chown-socket = ubuntu:ubuntu

enable-threads = true
master = true
vacuum = true
pidfile = /home/ubuntu/{프로젝트이름}/tmp/{프로젝트이름}.pid
logto = /home/ubuntu/{프로젝트이름}/log/uwsgi/@(exec://date +%Y-%m-%d).log
log-reopen = true
```

- 결과



```
1 # {프로젝트이름}/.config/uwsgi/{프로젝트이름}.ini
2
3 [uwsgi]
4 chdir = /home/ubuntu/MovieVibe
5 module = MovieVibe.wsgi:application
6 home = /home/ubuntu/.pyenv/versions/myenv
7
8 uid = ubuntu
9 gid = ubuntu
10
11 socket = /home/ubuntu/MovieVibe/tmp/MovieVibe.sock
12 chmod-socket = 666
13 chown-socket = ubuntu:ubuntu
14
15 enable-threads = true
16 master = true
17 vacuum = true
18 pidfile = /home/ubuntu/MovieVibe/tmp/MovieVibe.pid
19 logto = /home/ubuntu/MovieVibe/log/uwsgi/@(exec://date +%Y-%m-%d).log
20 log-reopen = true
```

4. daemon

- 설정파일 생성 (파일트리에서 생성해도 무방)

```
touch .config/uwsgi/uwsgi.service
```

- `.config/uwsgi/uwsgi.service` 설정파일 수정

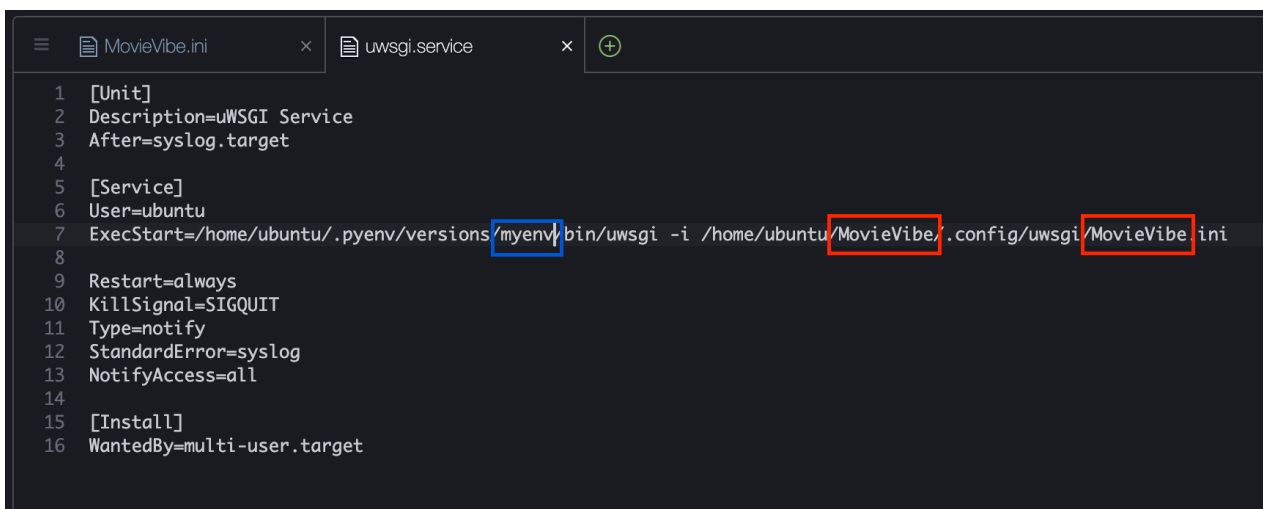
```
[Unit]
Description=uWSGI Service
After=syslog.target

[Service]
User=ubuntu
ExecStart=/home/ubuntu/.pyenv/versions/{가상환경이름}/bin/uwsgi -i
/home/ubuntu/{프로젝트이름}/.config/uwsgi/{프로젝트이름}.ini

Restart=always
KillSignal=SIGQUIT
Type=notify
StandardError=syslog
NotifyAccess=all

[Install]
WantedBy=multi-user.target
```

- 결과



```
1 [Unit]
2 Description=uWSGI Service
3 After=syslog.target
4
5 [Service]
6 User=ubuntu
7 ExecStart=/home/ubuntu/.pyenv/versions/myenv/bin/uwsgi -i /home/ubuntu/MovieVibe/.config/uwsgi/MovieVibe.ini
8
9 Restart=always
10 KillSignal=SIGQUIT
11 Type=notify
12 StandardError=syslog
13 NotifyAccess=all
14
15 [Install]
16 WantedBy=multi-user.target
```

- 심볼릭링크 생성


```
sudo ln -s ~/프로젝트이름/.config/uwsgi/uwsgi.service
/etc/systemd/system/uwsgi.service
```

- 등록

```
# daemon reload
sudo systemctl daemon-reload

# uwsgi daemon enable and restart
sudo systemctl enable uwsgi
sudo systemctl restart uwsgi.service

# check daemon
sudo systemctl | grep nginx
sudo systemctl | grep uwsgi

# nginx restart
sudo systemctl restart nginx
sudo systemctl restart uwsgi
```

- 아래의 에러 상황에서 80번 포트 프로세스 종료

```
(myenv) ubuntu:~/MovieVibe $ sudo systemctl restart uwsgi
(myenv) ubuntu:~/MovieVibe $
(myenv) ubuntu:~/MovieVibe $ systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: failed (Result: exit-code) since Wed 2020-06-17 16:53:23 UTC; 10s ago
     Docs: man:nginx(8)
  Process: 2180 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=1/FAILURE)
  Process: 2169 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)

Jun 17 16:53:21 ip-172-31-32-188 nginx[2180]: nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
Jun 17 16:53:21 ip-172-31-32-188 nginx[2180]: nginx: [emerg] bind() to [::]:80 failed (98: Address already in use)
Jun 17 16:53:22 ip-172-31-32-188 nginx[2180]: nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
Jun 17 16:53:22 ip-172-31-32-188 nginx[2180]: nginx: [emerg] bind() to [::]:80 failed (98: Address already in use)
Jun 17 16:53:22 ip-172-31-32-188 nginx[2180]: nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
Jun 17 16:53:22 ip-172-31-32-188 nginx[2180]: nginx: [emerg] bind() to [::]:80 failed (98: Address already in use)
Jun 17 16:53:23 ip-172-31-32-188 nginx[2180]: nginx: [emerg] still could not bind()
Jun 17 16:53:23 ip-172-31-32-188 systemd[1]: nginx.service: Control process exited, code=exited status=1
Jun 17 16:53:23 ip-172-31-32-188 systemd[1]: nginx.service: Failed with result 'exit-code'.
Jun 17 16:53:23 ip-172-31-32-188 systemd[1]: Failed to start A high performance web server and a reverse proxy server.
(myenv) ubuntu:~/MovieVibe $
```

```
sudo lsof -t -i tcp:80 -s tcp:listen | sudo xargs kill
```

- 최종확인

- EC2대시보드에서 DNS혹은 IP확인

