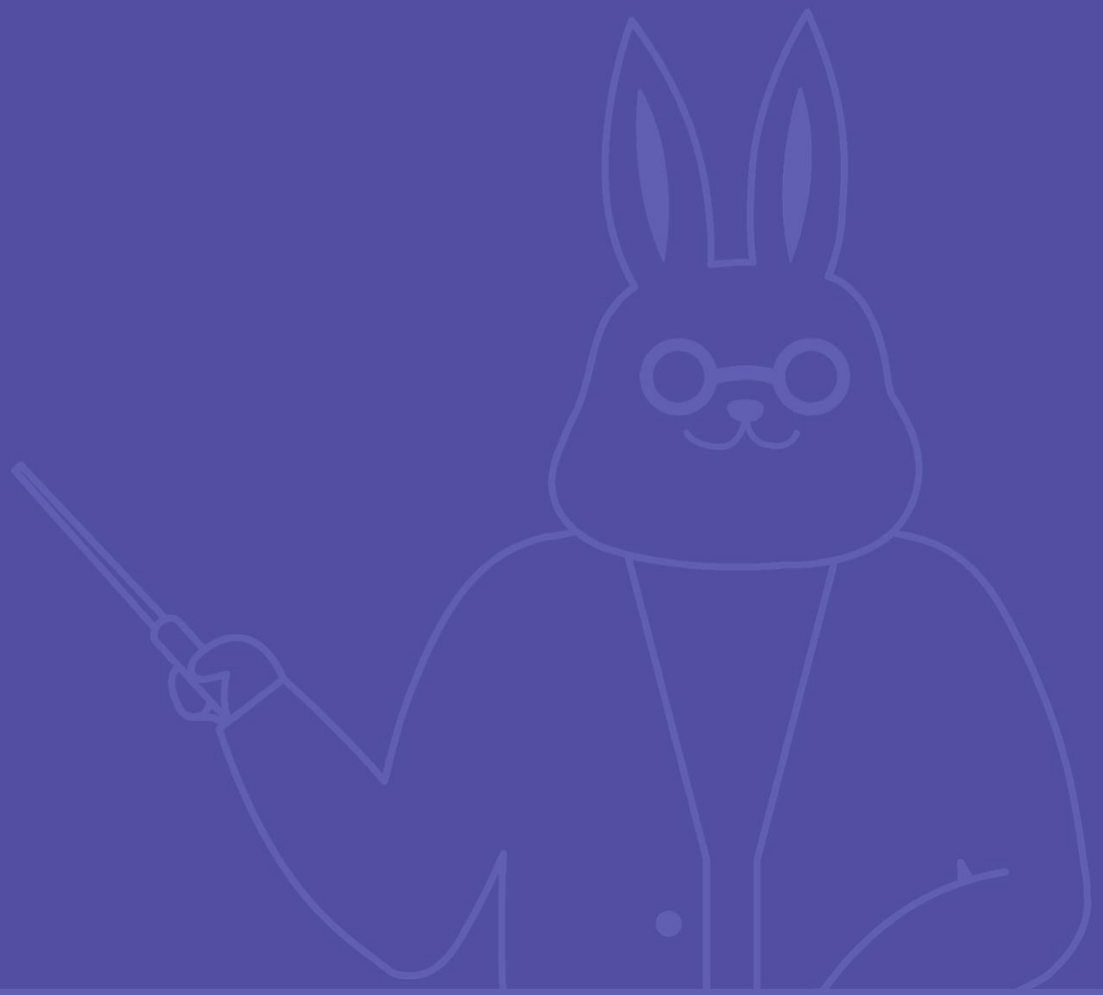
Flask 기초

08 Flask 서버 배포하기



/* elice */

목차

🎯 수강목표

Azure서버를 사용 할 수 있다.

Azure 서버를 생성하고 ssh를 사용하여 Azure 서버에 접근 할 수 습니다.

리눅스 환경에서 개발환경을 구축 할 수 있다.

Azure 서버 내에서 개발환경을 구축 해 봅니다. python 개발 환경과 mysql서버를 구축 해 보고, Flask를 실행 해 봅니다.

Uwsgi, Nginx 의 모듈을 Flask서버에 적용 할 수 있다.

Uwsgi, Nginx와 같은 서버를 실행 시켜 주는 모듈을 설정하고 적용 할 수 있습니다.

01

Flask 배포 코드 준비하기

Flask 배포 코드 준비하기

Flask 서버 코드를 준비해야 합니다.

Flask 서버를 실행하기 위해서는 플라스크 서버 코드가 준비되어야 합니다.
Flask 서버 코드를 준비하세요.

Gitlab에 소스코드 올리기

Gitlab을 사용해서 어디에서든 소스코드를 받을 수 있도록 준비 해야 합니다.
배포 할 서버에서 Gitlab의 소스코드를 다운받아 실행 해 주려고 합니다.

Gitlab에 업로드

myproject

Flask 서버 코드를 Gitlab에 올려서 어디에서든 다운받을 수 있도록 준비 해 주세요.

02

Azure 가상서버 만들기

Azure 가상서버 만들기

Azure 계정 (Microsoft 계정) 준비하기

Microsoft에서 제공하는 클라우드 서비스인 Azure를 배포에 사용하기 위해서는 계정이 필요합니다. 접속해서 Azure를 사용 할 수 있도록 계정을 준비해 주세요.

URL : https://azure.microsoft.com/ko-kr/free/

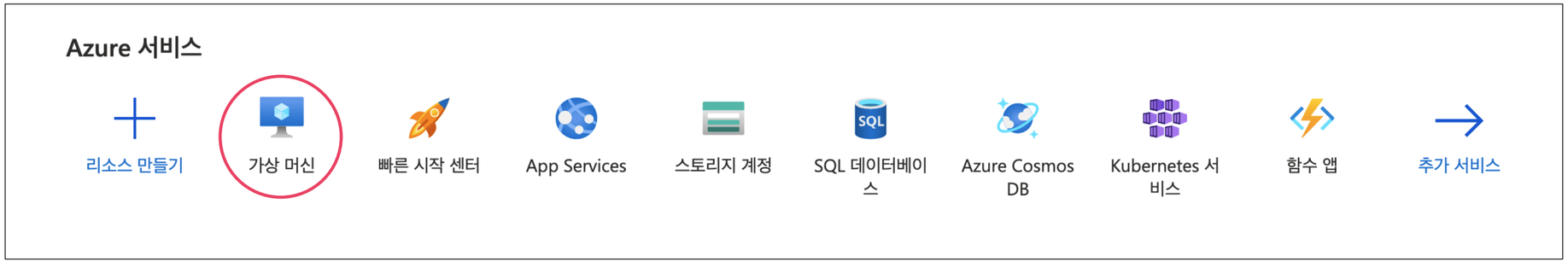
Azure Portal 접속하기

계정이 준비가 되고, 무료 서비스를 사용 할 준비가 되었다면 Azure 포털로 접속해서 Azure 가상서버를 생성하세요.

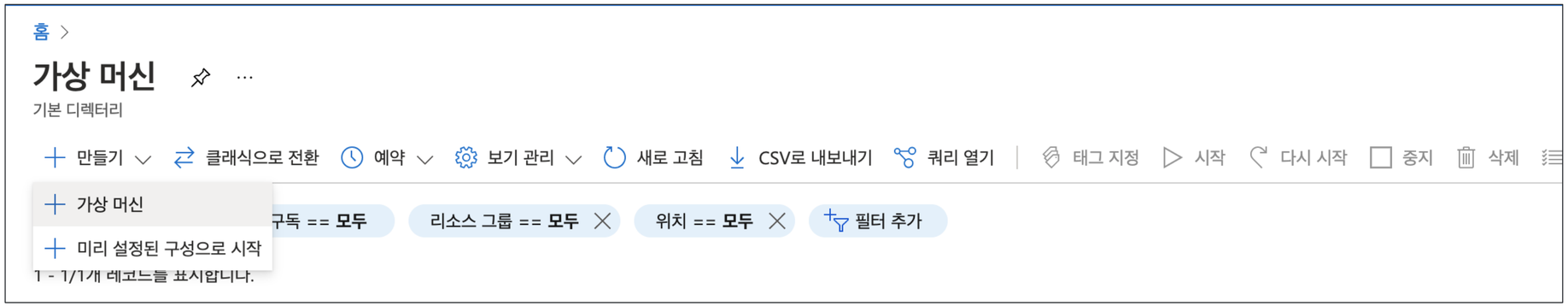
URL : https://portal.azure.com/

Azure 포털에서 내가 사용할 '가상머신' 서비스를 눌러 생성

1. 가상머신 클릭

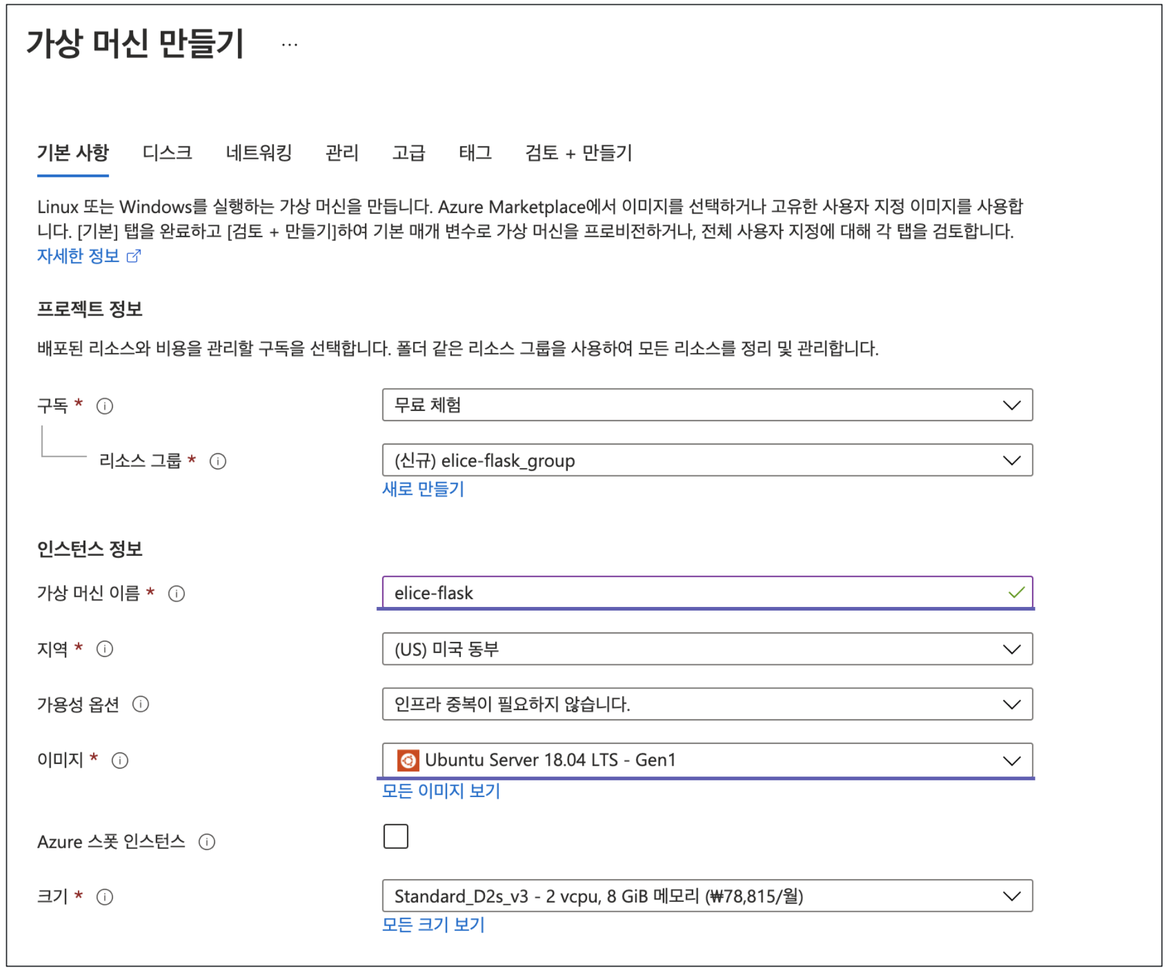


2. 만들기 > 가상 머신 클릭

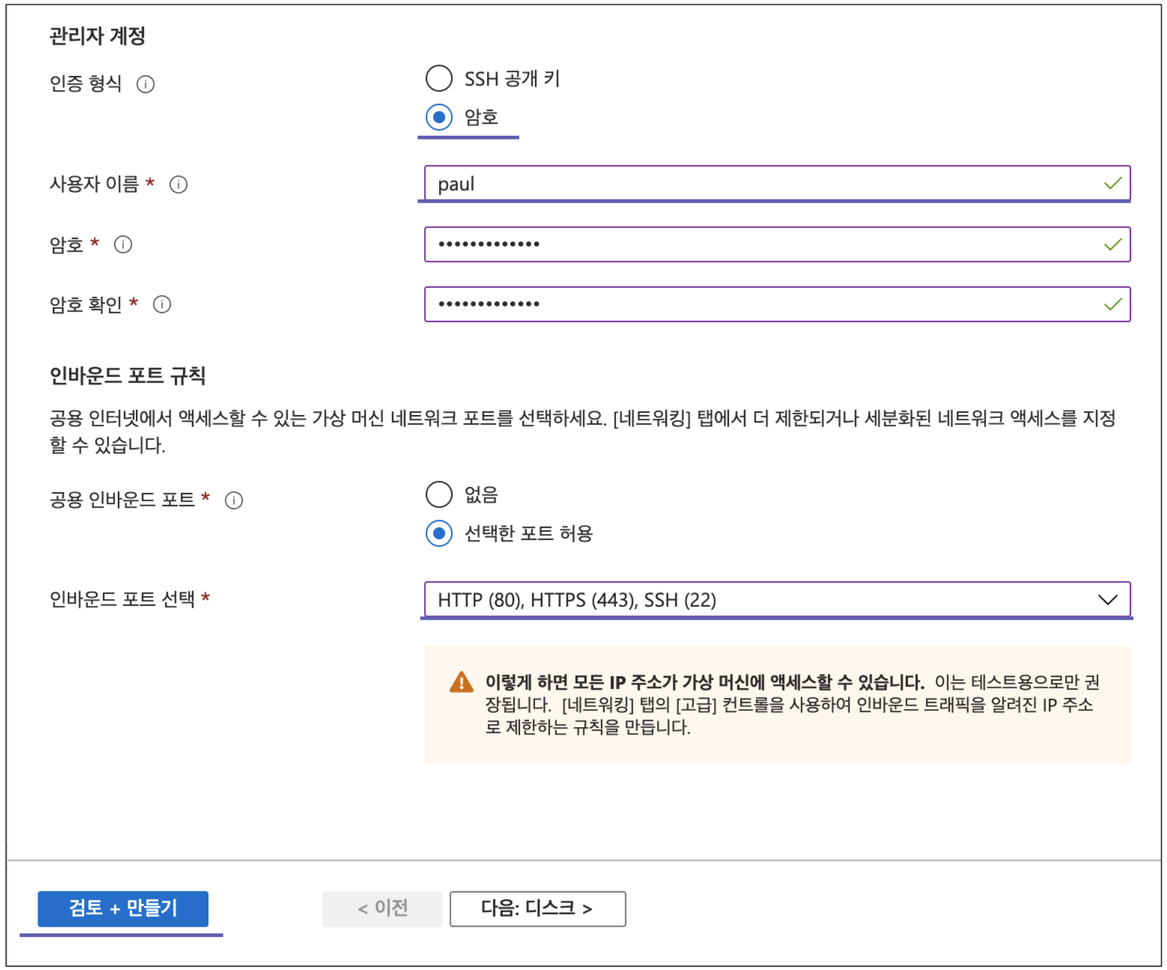


가상머신 설정하기

1. 가상머신 이름 및 OS 설치

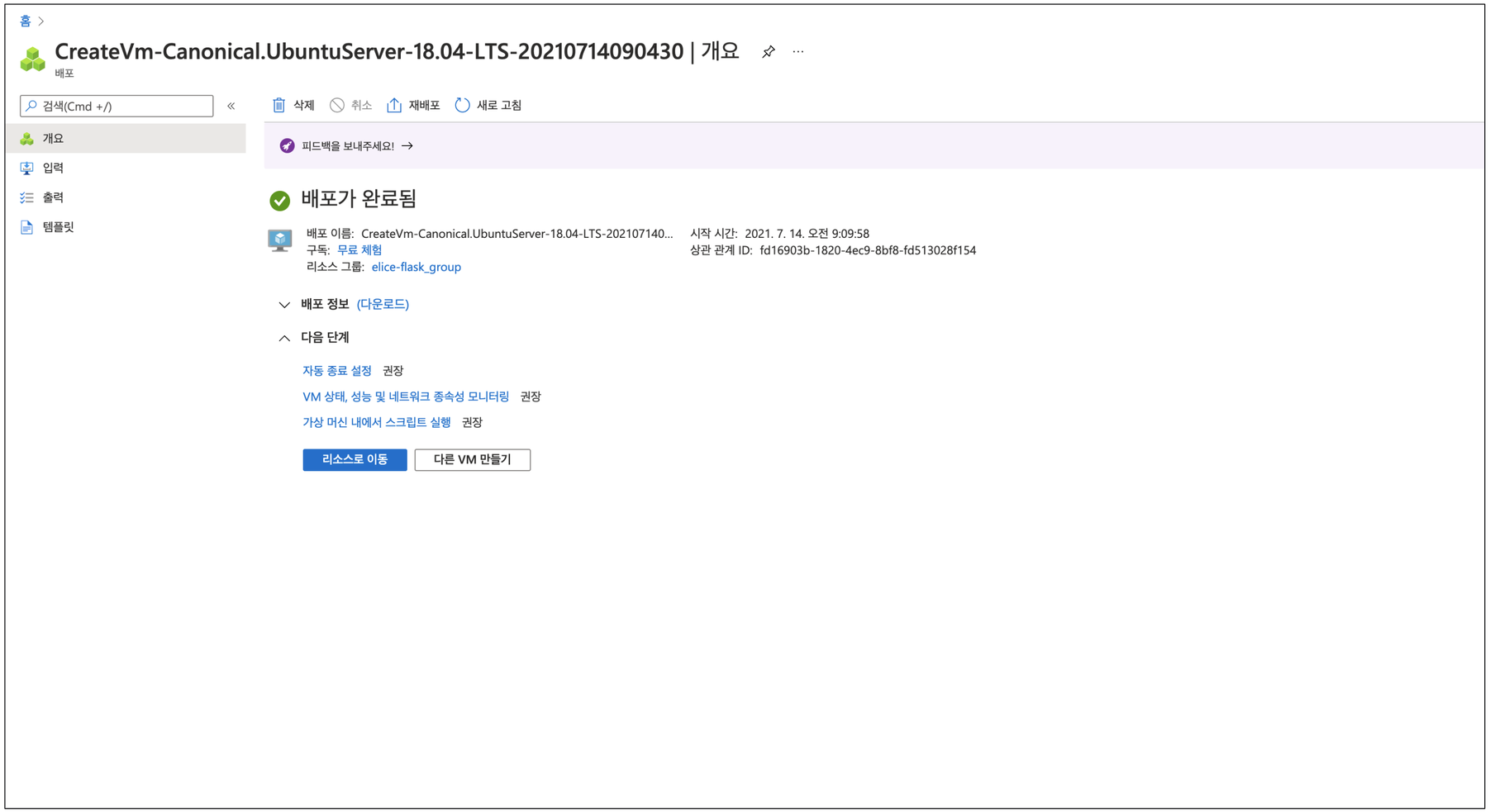


2. 가상머신 접속 계정 설정



가상머신 설정 완료

아래의 '리소스로 이동' 을 눌러 서버의 주소를 확인하세요.



가상머신 원격으로 접속하기

서버의 주소를 가지고 원격으로 접속 할 준비를 하세요.
- **window : gitbash**
- **mac : terminal**

명령어 : ssh [이름] @ [Azure서버 주소]
서버 주소는 리소스 페이지에 나오는 '공용 IP주소' 를 적어주면 됩니다.

입력 후 나오는 첫 질문에 yes 입력 한 후, Azure 생성 할 때 만들었던 패스워드 입력

```
wool ~ ssh paul@20.185.40.224
The authenticity of host '20.185.40.224 (20.185.40.224)' can't be established.
ECDSA key fingerprint is SHA256:RDRcryTVyV/GR7CBzewwYgQzxozSeb9mq9COCDfuWy4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '20.185.40.224' (ECDSA) to the list of known hosts.
paul@20.185.40.224's password:
```

03

Azure에서 Flask 실행하기

Azure에 개발환경 구축하기 – 개발도구 설치하기

아래의 명령어를 ssh로 연결 된 Azure 환경에서 실행하세요.

sudo apt update
sudo apt install python3-pip python3-dev build-essential libssl-dev libffi-dev python3-setuptools
sudo apt install python3-venv

***Tips - '&&' 으로 명령어를 한번에 실행 할 수 있어요.**

```
paul@elice-flask:~$ sudo apt update && sudo apt install python3-pip python3-dev build-essential libssl
-dev libffi-dev python3-setuptools && sudo apt install python3-venv
```

Azure에 개발환경 구축하기 – mysql 설치하기

Flask 서버 코드를 받기 전, 데이터베이스를 먼저 구축 해야 합니다.

sudo apt-get install mysql-server

설치가 완료되면 systemctl을 사용해서 mysql 서버를 실행 하세요.

sudo systemctl start mysql
sudo systemctl enable mysql

mysql에 접속해서 사용자와 사용자 비밀번호를 세팅해 주세요.

sudo mysql –u root ← mysql접속

use mysql;
update user set plugin='mysql_native_password' where user='root'; ← root 비밀번호를 바꿔주기 위함
flush privileges;
alter user 'root'@'localhost' IDENTIFIED BY 'qwerqwer123';
flush privileges;

Azure에 개발환경 구축하기 – mysql에 데이터베이스와 테이블 만들기

우리가 사용할 elice 테이블을 만들어주세요.

```
create database elice;
```

elice 데이터베이스 내에 테이블을 생성하세요.

```
use elice;

CREATE TABLE if not exists `post` (
        `id` int NOT NULL AUTO_INCREMENT,
        `author` varchar(256) NOT NULL,
        `content` text NOT NULL,
        `created_at` datetime DEFAULT NULL,
        PRIMARY KEY (`id`)
) ;


exit;
```

```
CREATE TABLE if not exists `user` (
        `id` int NOT NULL AUTO_INCREMENT,
        `user_id` varchar(100) NOT NULL,
        `user_pw` varchar(100) NOT NULL,
        PRIMARY KEY (`id`)
);
```

가상머신에서 Gitlab 코드 불러오기

명령어 : git clone [git lab 주소]
*** 첫 git clone 때 에는, git lab의 id, pw 입력이 필요합니다.**

```
paul@elice-flask:~$ git clone https://gitlab.com/paullee714/myproject.git
Cloning into 'myproject'...
Username for 'https://gitlab.com': paullee714
Password for 'https://paullee714@gitlab.com':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
paul@elice-flask:~$ ls
myproject
```

Azure에 개발환경 구축하기 – 가상환경 만들기

Gitlab으로 다운받은 나의 폴더로 이동하세요.

```
paul@elice-flask:~$ ls
myproject
paul@elice-flask:~$ cd myproject/
paul@elice-flask:~/myproject$
```

나의 폴더 안에 가상환경을 설정하고 실행하세요.

```
paul@elice-flask:~/myproject$ ls
app.py
paul@elice-flask:~/myproject$ python3 -m venv venv
paul@elice-flask:~/myproject$ ls
app.py  venv
paul@elice-flask:~/myproject$ source venv/bin/activate
(venv) paul@elice-flask:~/myproject$
```

Azure에 개발환경 구축하기 – 가상 환경에 패키지 다운받기

실행 된 가상환경에서 패키지를 다운로드 하세요.

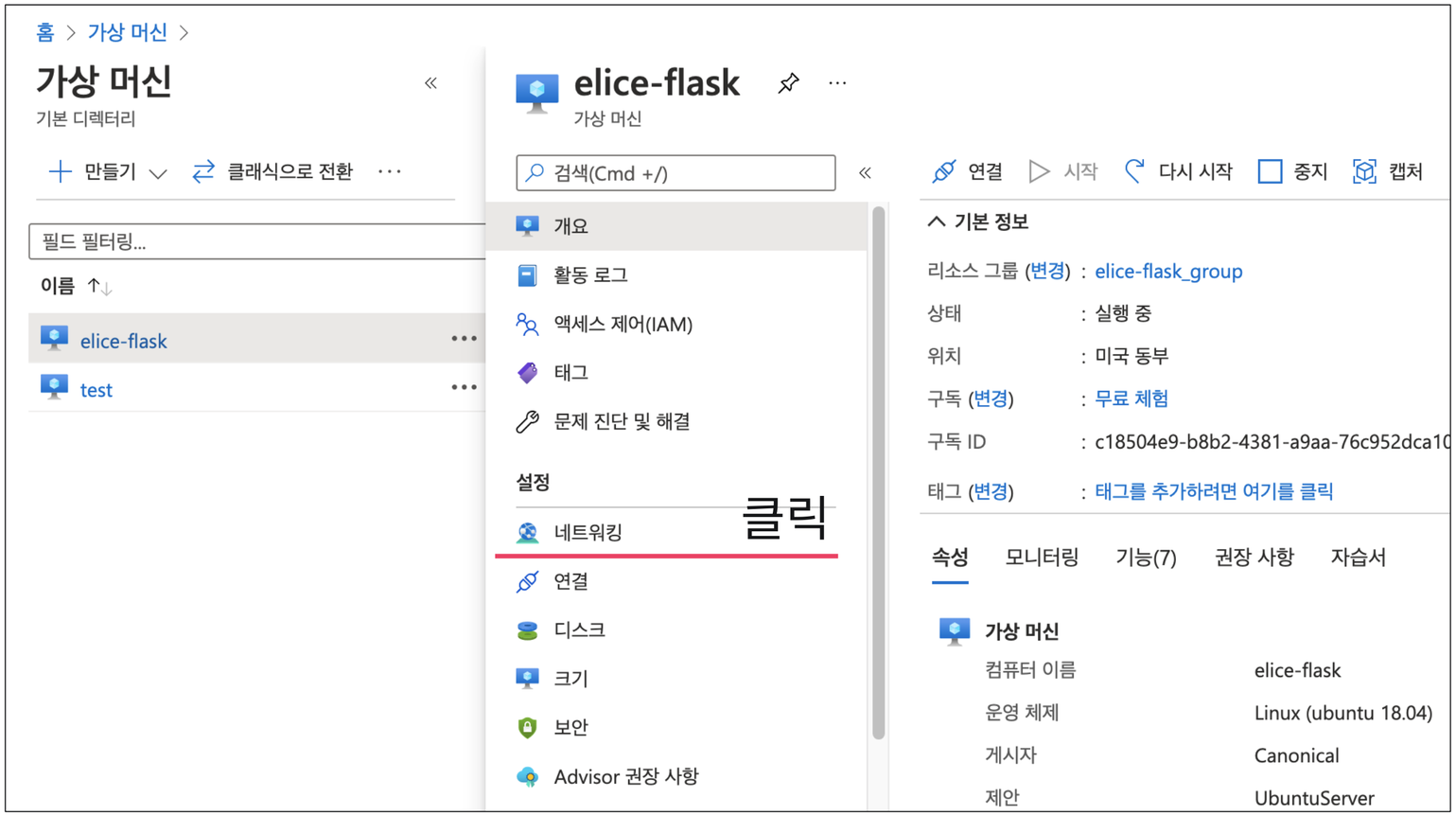
```
(venv) paul@elice-flask:~/myproject$ pip3 install flask flask-sqlalchemy pymysql Flask-Bcrypt uwsgi
```

설치가 완료되었다면 app.py를 실행 시켜 서버를 구동하세요.

```
(venv) paul@elice-flask:~/myproject$ python app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://10.3.0.4:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 258-021-079
```

Azure에 개발환경 구축하기 – 포트 열어주기

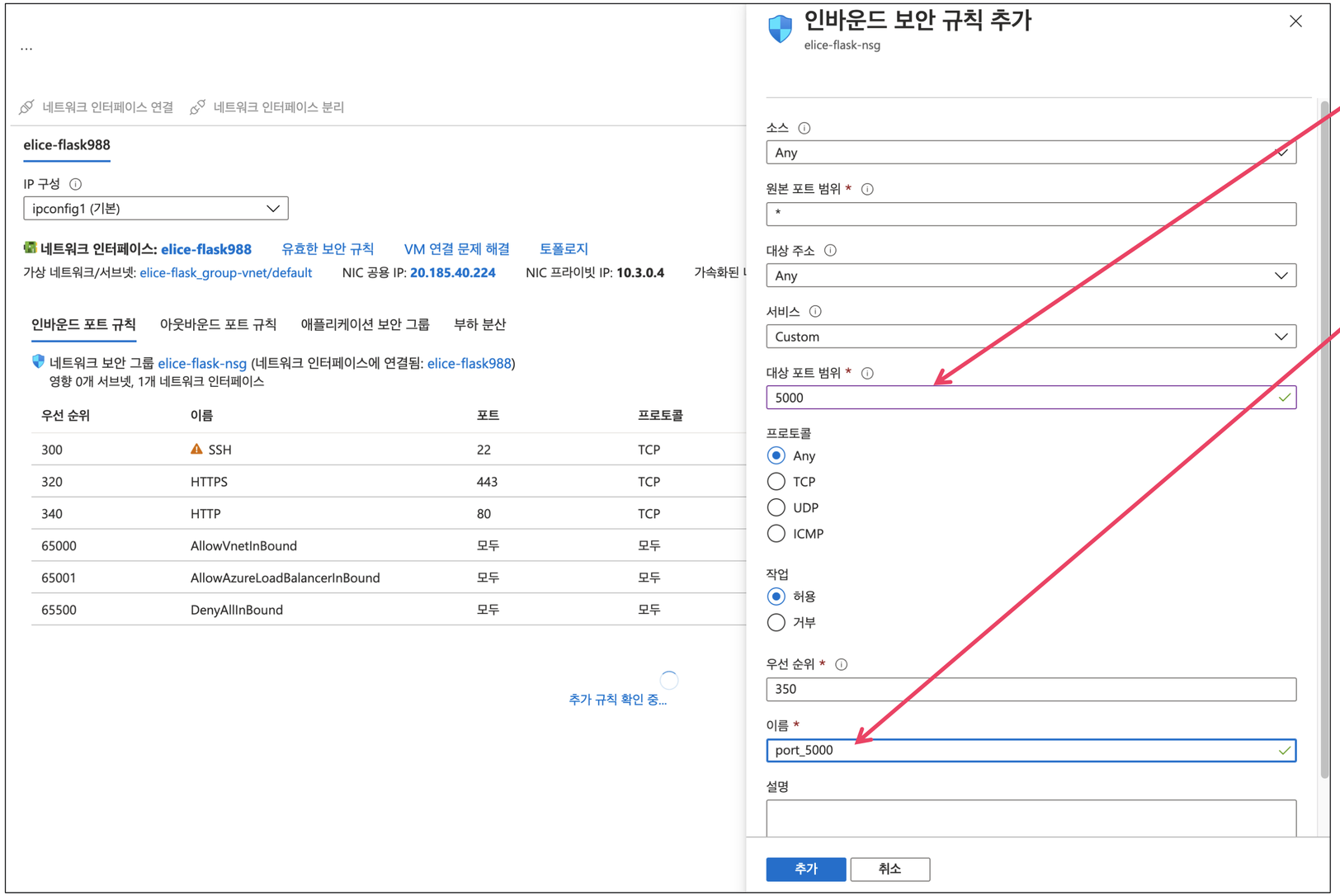
Azure는 무분별한 접근을 막기 위해서 허용된 포트 외에는 접근이 불가능합니다.
Flask 서버는 5000번을 사용하고 있기 때문에, 5000번 포트를 열어주도록 하겠습니다.



Azure에 개발환경 구축하기 – 포트 열어주기

기존의 포트 외의 새로운 포트를 열어주기 위해 '인바운드 포트 규칙 추가'를 클릭하세요.

Azure에 개발환경 구축하기 – 포트 열어주기

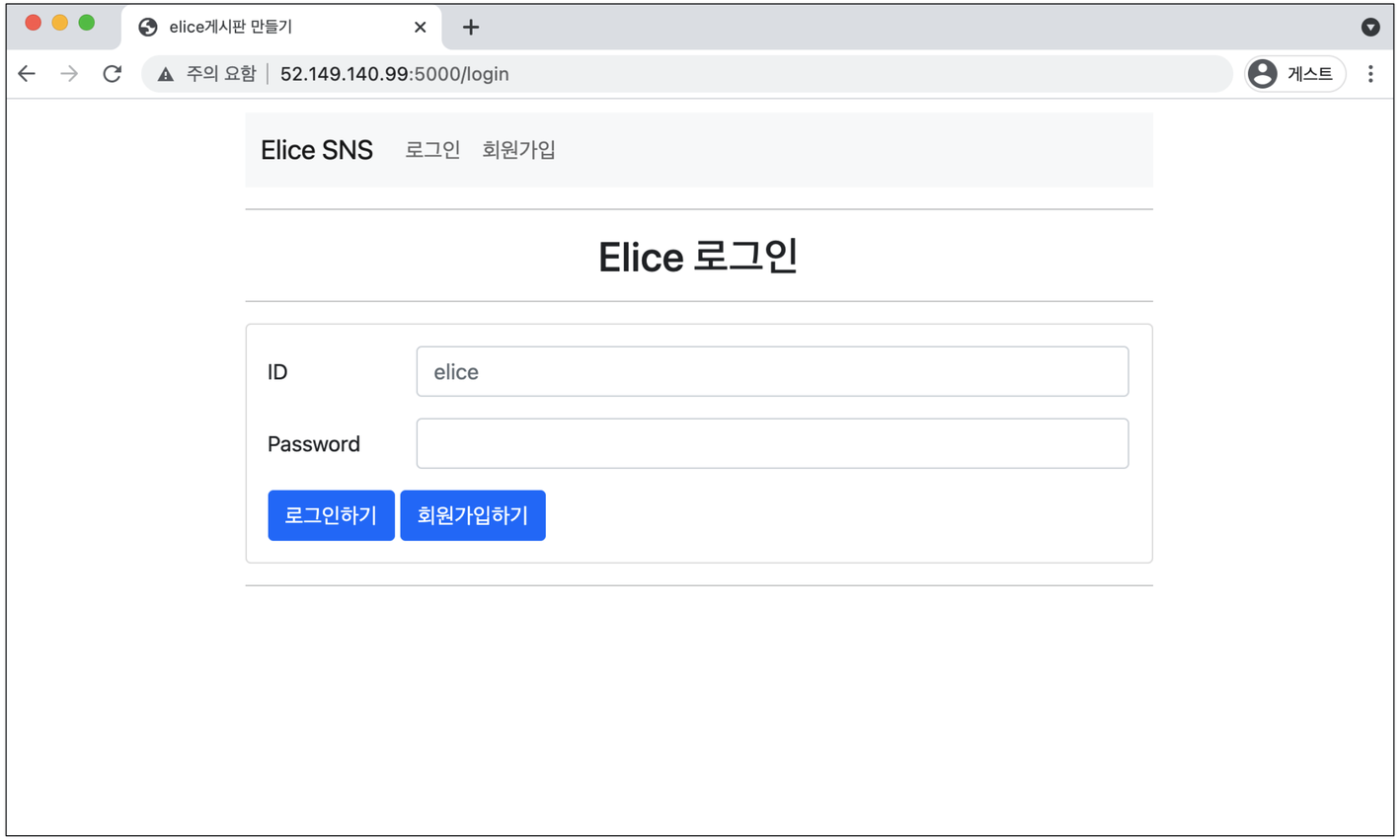


대상 포트 범위
5000

이름
port_5000

내 Flask 서버에 접근하기

Flask 서버는 5000번 포트를 사용합니다.
이전에 열어주었던 5000번 포트가 잘 열렸다면 플라스크 서버에 접근이 잘 되는 것을 볼 수 있습니다.
접근했던 서버의 공용 IP 뒤에 :5000 을 붙여주면 플라스크 서버에 접근 할 수 있습니다.



04

Azure에서 Flask + Uwsgi 실행하기

UWSGI?

app.py를 실행해서 열린 서버는
개발서버 이기 때문에 성능이 좋지 않습니다.
성능을 개선하기 위해서 사용 하는 모듈이 uwsgi 입니다.

wsgi.py 모듈 생성하기

uwsgi가 직접 실행 할 wsgi.py 를 생성해 주세요.
리눅스의 vim으로 코드를 작성해 주세요 (들여쓰기 주의!)

```
(venv) paul@elice-flask:~/myproject$ vim wsgi.py
```

```
from app import app

if __name__ == "__main__":
    app.run()
```

작성 시작은 'a' 를 눌러서 에디팅 모드로 접근 후 하세요.
작성이 완료 된 후에는 esc 키를 누르고 ':wq' 를 입력하고 enter를 누르세요.

uwsgi 로 실행하기

uwsgi –socket 0.0.0.0:5000 --protocol=http –w wsgi:app

```
(venv) paul@elice-flask:~/myproject$ uwsgi --socket 0.0.0.0:5000 --protocol=http -w wsgi:app
```

실행이 성공 했을 경우 -> 동일한 주소로 접근 했을 경우에, 서버에 로그가 올라온다.

```
detected number of CPU cores: 2
current working directory: /home/paul/myproject
detected binary path: /home/paul/myproject/venv/bin/uwsgi
!!! no internal routing support, rebuild with pcre support !!!
*** WARNING: you are running uWSGI without its master process manager ***
your processes number limit is 31702
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
uwsgi socket 0 bound to TCP address 0.0.0.0:5000 fd 3
Python version: 3.6.9 (default, Jan 26 2021, 15:33:00)  [GCC 8.4.0]
*** Python threads support is disabled. You can enable it with --enable-threads ***
Python main interpreter initialized at 0x555b3b4cee60
your server socket listen backlog is limited to 100 connections
your mercy for graceful operations on workers is 60 seconds
mapped 72920 bytes (71 KB) for 1 cores
*** Operational MODE: single process ***
WSGI app 0 (mountpoint='') ready in 0 seconds on interpreter 0x555b3b4cee60 pid: 20356 (default app)
*** uWSGI is running in multiple interpreter mode ***
spawned uWSGI worker 1 (and the only) (pid: 20356, cores: 1)
```

05

Azure에서 Flask + Uwsgi + Nginx 실행하기

Azure에서 Flask + Uwsgi + Nginx 실행하기

웹 서버 Nginx

웹 서버에는 다양한 종류가 있습니다.
Apache, IIS, Nginx 등의 많은 웹 서버의 종류가 있는데, 실행과 구축이 조금 어렵고, 최근 버그가 발견된 Apache 보다는 Nginx를 선호하는 경향이 있습니다.

Nginx는 외부에서 들어오는 다량의 요청을 쉽게 처리 할 수 있는 장점을 가지고 있습니다.

uwsgi 실행을 파일로 옮기기

uwsgi –socket 0.0.0.0:5000 --protocol=http –w wsgi:app

```
(venv) paul@elice-flask:~/myproject$ uwsgi --socket 0.0.0.0:5000 --protocol=http -w wsgi:app
```

위의 명령어는, 실행 할 때 마다 너무 길기 때문에 위의 명령어를 ini 파일로 만들어 주도록 하겠습니다.

```
(venv) paul@elice-flask:~/myproject$ vim myproject.ini
[uwsgi]
module = wsgi:app

master = true
processes = 5

socket = myproject.sock
chmod-socket = 660
vacuum = true

die-on-term = true
```

*** Tips**
작성 시작은 ‘a’ 를 눌러서 에디팅 모드로 접근 후 하세요.
작성이 완료 된 후에는 esc 키를 누르고 ‘:wq’ 를 입력하고 enter를 누르세요.

uwsgi 을 systemctl로 옮기기

systemctl은 서버가 실행이 되고 백그라운드에서 자동으로 실행 되게 해 주는 기능입니다.
uwsgi를 systemctl에 등록하고 서버가 자동으로 실행 될 수 있도록 하겠습니다.

```
(venv) paul@elice-flask:~/myproject$ sudo vim /etc/systemd/system/myproject.service
```

```
[Unit]
Description=uWSGI instance to serve myproject
After=network.target

[Service]
User=paul
Group=www-data
WorkingDirectory=/home/paul/myproject
Environment="PATH=/home/paul/myproject/venv/bin"
ExecStart=/home/paul/myproject/venv/bin/uwsgi --ini myproject.ini

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

User <- 로그인 때 사용 한 이름
/home/paul/myproject <- 다운받은 폴더 이름

* paul과 myproject는 로그인 할 때와 Gitlab에서 소스를 받을 때 에 따라서 달라 질 수 있습니다

*** Tips**
작성 시작은 'a' 를 눌러서 에디팅 모드로 접근 후 하세요.
작성이 완료 된 후에는 esc 키를 누르고 ':wq' 를 입력하고 enter를 누르세요.

uwsgi 을 systemctl로 옮기기

sudo systemctl start myproject **<- myproject 시작**
sudo systemctl enable myproject **<- myproject 등록**
sudo systemctl status myproject **<- myproject 상태 확인하기**

```
(venv) paul@elice-flask:~/myproject$ sudo systemctl start myproject
(venv) paul@elice-flask:~/myproject$ sudo systemctl enable myproject
(venv) paul@elice-flask:~/myproject$ sudo systemctl status myproject
● myproject.service - uWSGI instance to serve myproject
   Loaded: loaded (/etc/systemd/system/myproject.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-07-14 01:46:48 UTC; 11s ago
 Main PID: 21049 (uwsgi)
    Tasks: 6 (limit: 4915)
   CGroup: /system.slice/myproject.service
           ├─21049 /home/paul/myproject/venv/bin/uwsgi --ini myproject.ini
           ├─21072 /home/paul/myproject/venv/bin/uwsgi --ini myproject.ini
           ├─21073 /home/paul/myproject/venv/bin/uwsgi --ini myproject.ini
           ├─21074 /home/paul/myproject/venv/bin/uwsgi --ini myproject.ini
           ├─21076 /home/paul/myproject/venv/bin/uwsgi --ini myproject.ini
           └─21077 /home/paul/myproject/venv/bin/uwsgi --ini myproject.ini

Jul 14 01:46:48 elice-flask uwsgi[21049]: mapped 437520 bytes (427 KB) for 5 cores
Jul 14 01:46:48 elice-flask uwsgi[21049]: *** Operational MODE: preforking ***
Jul 14 01:46:49 elice-flask uwsgi[21049]: WSGI app 0 (mountpoint='') ready in 1 seconds on interpreter
Jul 14 01:46:49 elice-flask uwsgi[21049]: *** uWSGI is running in multiple interpreter mode ***
Jul 14 01:46:49 elice-flask uwsgi[21049]: spawned uWSGI master process (pid: 21049)
Jul 14 01:46:49 elice-flask uwsgi[21049]: spawned uWSGI worker 1 (pid: 21072, cores: 1)
Jul 14 01:46:49 elice-flask uwsgi[21049]: spawned uWSGI worker 2 (pid: 21073, cores: 1)
Jul 14 01:46:49 elice-flask uwsgi[21049]: spawned uWSGI worker 3 (pid: 21074, cores: 1)
Jul 14 01:46:49 elice-flask uwsgi[21049]: spawned uWSGI worker 4 (pid: 21076, cores: 1)
```

Nginx 설치하기

Nginx 설치를 위해서 아래의 명령어를 실행해 주세요.

sudo apt-get install nginx

```
(venv) paul@elice-flask:~/myproject$ sudo apt-get install -y nginx
```

이제 Azure 서버에서 Nginx를 사용 할 수 있습니다.
기본 설정을 바꾸어서 우리가 실행 한 uwsgi와 연동 시켜주도록 하겠습니다.

Nginx 설정하기

```
(venv) paul@elice-flask:~/myproject$ sudo vim /etc/nginx/sites-available/default
```

여러 설정이 나오는데, 스크롤을 내려서 'location' 이라는 부분을 찾아서 수정 해 주도록 하겠습니다.

```
location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        #try_files $uri $uri/ =404;
        include uwsgi_params;
        uwsgi_pass unix:/home/paul/myproject/myproject.sock;
}
```

*** Tips**
작성 시작은 'a' 를 눌러서 에디팅 모드로 접근 후 하세요.
작성이 완료 된 후에는 esc 키를 누르고 ':wq' 를 입력하고 enter를 누르세요.

Nginx 설정하기

sudo nginx –t 명령어로, nginx 설정이 잘 되어있는지 확인 해 주세요.

```
(venv) paul@elice-flask:~/myproject$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

우리가 수정 한 환경설정이 잘 적용 되기 위해서 nginx를 restart 해 주세요.

```
(venv) paul@elice-flask:~/myproject$ sudo systemctl restart nginx
```

접근 했었던 [ip주소]:5000 에서, 포트번호를 떼고 ip 주소만 입력 해도 접근이 됩니다!

크레딧

/* elice */

코스 매니저
이재성

콘텐츠 제작자
이바울, 이재성

강사
이바울

감수자
이바울

디자이너
강혜정

연락처

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io