공통 프로젝트 - 웹 IoT

포팅매뉴얼

A109

임진경 김유민 문요성 이지수 정진아

목차

1. MobaXterm

a. 버전

i. MobaXterm Home Edition v22.1 installer edition

b. 설치 및 설정

i. Session - SSH - Remote host에 호스트 주소 입력 - Specify username에 ubuntu 입력

ii. Advanced SSH settings - Use private key 체크 - pem 키 설정 - OK

2. MySQL

a. 버전

i. 8.0.20

b. 설치

i. 업데이트 후 설치

```
$ sudo apt-get update
$ sudo apt-get install mysql-server
$ sudo systemctl start mysql
```

c. 설정

i. 초기 root 계정 설정 (비밀번호 설정)

```
$ mysqladmin -u root -p password '비밀번호'
```

ii. DB 생성

```
> CREATE DATABASE {DB이름};
# CREATE DATABASE Plantinum;
```

iii. plantinum 사용자 계정 생성

```
> CREATE USER '{username}'@'%' IDENTIFIED BY '{password}';
# CREATE USER 'plantinum'@'%' IDENTIFIED BY 'your password';
```

iv. 권한 부여

```
> GRANT ALL PRIVILEGES ON {database}.* TO '{username}'@'%';
# GRANT ALL PRIVILEGES ON Plantinum.* TO 'plantinum'@'%';
```

3. Redis

a. 버전

i. 5.0.7

b. 설치

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install redis-server
```

c. 설정

i. maxmemory 설정을 위해 redis.conf 파일 편집

```
$ sudo vim /etc/redis/redis.conf
```

ii. maxmemory와 maxmemory-policy의 주석 부분을 편집

1. `i`를 누르면 입력 가능

```
maxmemory 1g  # 최대 메모리 사용량 1G
maxmemory-policy allkeys-lru  # 초과시 오래된 데이터를 지워서 메모리 확보
```

iii. - 편집이 끝나면 `Esc` + `:wq`

4. S3

a. **IAM** 정책 생성

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
    {
      "Sid": "S3statement",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObjectAcl",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject",
        "s3:PutObjectAcl"
      ],
      "Resource": [
        "arn:aws:s3:::{bucket_name}/*",
        "arn:aws:s3:::{bucket_name}"
      ]
    }
  ]
}
```

b. IAM 사용자 생성

i. 기존 정책 직접 연결

ii. 고객관리형에서 생성한 정책 선택

iii. 사용자 엑세스 키 ID와 비밀 엑세스 키를 저장

c. 버킷생성 및 설정

i. plantinum

ii. 퍼블릭 엑세스 차단 해제

iii. 권한 - 버킷정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "StatementSid1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "{user_arn}"
            },
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::{bucket_name}/*"
        },
        {
            "Sid": "StatementSid2",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::{bucket_name}/*"
        }
    ]
}
```

iv. CORS 구성

```
[
  {
      "AllowedHeaders": [
          ""
      ],
      "AllowedMethods": [
          "GET",
          "PUT",
```

```
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            ""
        ],
        "ExposeHeaders": [
            "ETag"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

5. Python

a. 버전

i. 3.9.1

b. 설치

i. 개발 라이브러리 다운

```
$ sudo apt-get install build-essential checkinstall
$ sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev
libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev libffi-dev zlib1g-dev
```

ii. 파이썬 설치

```
$ cd /opt
$ sudo wget https://www.python.org/ftp/python/3.9.1/Python-3.9.1.tgz
$ sudo tar xzf Python-3.9.1.tgz
```

iii. 파이썬 컴파일

```
$ cd Python-3.9.1
$ sudo ./configure --enable-optimizations
$ sudo make altinstall
```

iv. 버전 확인

```
$ python -V
Python 3.9.1
```

v. 디폴트 설정

```
$ sudo update-alternatives --install /usr/bin/python python
/usr/local/bin/python3.9 1
```

6. Django BE

a. 버전

i. 3.2.14

b. 설치

i. MobaXterm 접속

ii. Session - SFTP - Remote host에 호스트 주소 입력 - Username에 ubuntu 입력

iii. Advanced Sftp settings - Use private key 체크 - pem 키 설정 - OK

iv. Django 프로젝트 폴더 복사 (back)

v. my_settings.py 파일 생성

Django 프로젝트의 manage.py 와 동일한 레벨에서 생성

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'Plantinum',  # db name
        'USER': 'plantinum',  # user name
        'PASSWORD': 'your password',  # user password
        'HOST': '127.0.0.1',
        'PORT': '3306',
```

```
  }
}

SECRET_KEY = '<your secret key>'

AWS_ACCESS_KEY_ID = '<your aws access key id>'
AWS_SECRET_ACCESS_KEY = '<your secret access key>'
```

vi. 가상환경 설치

```
# 경로 /home/ubuntu/
$ python -m venv venv
```

vii. 가상환경 활성화

```
# 경로 /home/ubuntu/
$ source venv/bin/activate
```

viii. Django 프로젝트 폴더로 이동

```
$ cd back
```

ix. 라이브러리 설치, DB 테이블 생성, 초기데이터 입력

```
$ pip install -r requirements.txt
$ python manage.py migrate
$ python manage.py loaddata plants.json juso.json
```

7. Vue FE

a. 버전

i. 3.2.13

b. 설치

i. 빌드생성

```
$ npm run build
```

ii. 생성된 dist 폴더를 /var/www/html/ 에 복사

c. 재실행

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart uwsgi nginx redis-server.service
```

8. 채팅 서버 - Node.js

a. 버전

i. node : 16.15.0
ii. express : ^4.17.2
iii. socket.io : ^4.4.0
iv. mysql2 : ^2.3.3

b. 설치

i. 폴더에 있는 node_modules 설치

```
$ npm i
```

c. 설정

i. ./db파일 안의 index.js 수정

```
const mysql = require("mysql2/promise");
const pool = mysql.createPool({
    host: "db 서버 주소",
    port : "포트번호",
    user: "db 접속하려는 유저",
    password: "db 비밀번호 ",
    database: "접속하려는 db 이름",
    waitForConnections:true,
    connectionLimit:10,
    queueLimit:0,
});
module.exports = { pool };
```

d. 채팅 서버 실행

i. 실행

백그라운드 및 터미널 종료시에도 실행 필요

```
$ cd chatServer
$ nohup node app.js &
```

ii. 터미널 종료

```
$ exit
```

9. NGINX

a. 버전

i. nginx 1.18.0(ubuntu)

b. 설치

```
$ sudo apt-get install nginx
```

c. 설정 파일 변경

i. nginx.conf 파일 변경

```
$ sudo vi /etc/nginx/nginx.conf
```

http {} 블록 끝에 추가

```
include /etc/nginx/sites-enabled/*.conf;
server_names_hash_bucket_size 64;
```

ii. default.conf 파일 변경

```
$ sudo vi /etc/nginx/sites-available/default
```

root에 작성해 둔 dist 파일의 경로 입력
서버네임에 도메인 입력 없다면 '_' 입력

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

        root /var/www/html/dist;
        index index.html index.htm;

        server_name plantinum.co.kr www.plantinum.co.kr;
```

```
        ocation / {
                try_files $uri $uri/ /index.html;
        }

        location /api {
                uwsgi_pass      unix:///tmp/back.sock;
                include         uwsgi_params;
        }

}
```

iii. 링크 및 활성화

```
$ sudo ln -s /etc/nginx/sites-available/default.conf
/etc/nginx/sites-enabled/default.conf
```

재시작은

```
$ sudo systemctl restart nginx
```

/var/log/nginx 에서 로그 확인 가능

10. uWSGI

django는 nginx에서 바로 프록시로 쓰지 못하기 때문에 uwsgi를 통하여 연결한다.

a. 버전

i. uwsgi 2.0.20

b. 설치

i. 가상환경 활성화

```
$ source venv/bin/activate
```

ii. uwsgi 설치

```
$ pip install uwsgi
```

c. 파일 설정

i. django의 설정파일

프로젝트 폴더 안 manage.py 가 있는 폴더에서 작업

```
$ cd ./.config
$ mkdir uwsgi
$ cd uwsgi
```

이 안에서 back.ini 생성

```
$ vi back.ini
```

```
[uwsgi]
#프로젝트 폴더 절대 경로
chdir = /home/ubuntu/back
#절대 경로 안 wsgi파일 경로 (프로젝트 폴더 안)
module = PLANTinum_back.wsgi:application
#가상환경 경로
home = /home/ubuntu/venv/

uid = ubuntu
gid = ubuntu

#1
#http = :8000

#2 소켓 연결
```

```
socket = /tmp/back.sock
chmod-socket = 666
chown-socket = ubuntu:ubuntu

enable-threads = true
master = true
vacuum = true
pidfile = /tmp/mysite.pid
logto = /var/log/uwsgi/back/@(exec://date "+%%Y-%%m-%%d").log
log-reopen = true
```

작성 후 저장
프로젝트 내의 wsgi.py의 파일의 위치 확인 필요
로그는 /var/log/uwsgi/back/에서 확인

ii. 서비스 등록 스크립트 생성

```
$ sudo vi /etc/systemd/system/uwsgi.service
```

스크립트 작성

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

[Service]
ExecStart=/home/ubuntu/venv/bin/uwsgi -i
/home/ubuntu/back/.config/uwsgi/back.ini

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

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

iii. uwsgi 구동

```
$ sudo systemctl start uwsgi
$ sudo systemctl enable uwsgi
```

iv. nginx 설정 파일 변경

```
$ sudo vi /etc/nginx/sites-available/default
```

이동 후 작성

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

        root /var/www/html/dist;
        index index.html index.htm;

        server_name plantinum.co.kr www.plantinum.co.kr;

        location / {
                try_files $uri $uri/ /index.html;
        }

        location /api {
                uwsgi_pass      unix:///tmp/back.sock;
```

```
        include         uwsgi_params;
    }
}
```

이후 변경사항이 있을 때는 다음 명령어를 실행

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart uwsgi
```

11. 라즈베리파이

a. 버전

i. 라즈베리파이 4b

ii. python 3.9.2

b. 필요 패키지 설치 및 파일 설정

i. vim

```
#.vimrc 내부

set nocompatible
filetype off

set rtp+=~/.vim/bundle/Vundle.vim
call vundle#begin()

Plugin 'VundleVim/Vundle.vim'
Plugin 'delimitMate.vim'

call vundle#end()
```

```
filetype plugin indent on



set nu
set ts=4
set sw=4
set ls=2
set cindent
set autoindent
set encoding =utf-8
syntax on

let delimitMate_expand_cr=1
```

이후 vim 을 열어 PlugInstall 로 Vundle에서 불러온 Plugin을 설치

```
$ sudo vim
:PlugInstall
```

ii. PySide2 설치 및 앞으로의 설치를 위한 도구들 설치

```
$ pip3 install PySide2
$ sudo apt-get install build-essential python-dev scons swig
```

iii. 온습도 센서 dht11 - adafruit_dht

```
$ git clone https://github.com/adafruit/Adafruit_Python_DHT
$ cd Adafruit_Python_DHT/Adafruit_Python_DHT

//라즈베리파이4 이기 때문에 설정을 바꾸어주어야 한다.
$ sudo vi platform_detect.py
```

마지막 부분 pi_version()에 라즈베이파이 4를 위한 코드 삽입

```
if match.group(1) == 'BCM2708':
    # Pi 1
    return 1
elif match.group(1) == 'BCM2709':
    # Pi 2
    return 2
elif match.group(1) == 'BCM2835':
    # Pi 3 or Pi 4
    return 3
elif match.group(1) == 'BCM2837':
    # Pi 3b+
    return 3
# 추가할 부분
elif match.group(1) == 'BCM2711':
    return 3
else:
    # Something else, not a pi.
    return None
```

이후 설치

```
$ sudo python3 setup.py install
```

iv. neo-pixel LED 모듈

```
$ sudo pip3 install rpi_ws281x adafruit-circuitpython-neopixel
$ sudo python3 -m pip install --force-reinstall adafruit-blinka
```

명령어 두개를 입력