

컴퓨터 네트워크

-3주차-

학번: 201404377

이름: 진승언

이번 과제의 목표는 도커내에서 웹 프레임워크를 활용해서 웹 페이지를 서빙하는 것이다. 이를 위해 nginx, uwsgi, django를 사용한다. uwsgi는(Web Server Gateway Interface) 웹서버와 웹 애플리케이션의 인터페이스를 위한 파이썬 프레임워크다. nginx(서버)가 보낸 요청을 django가 알아들을 수 없다. 즉 둘은 직접 연동이 안된다. 그래서 uwsgi를 사용해서 nginx 보낸 요청을 파이썬으로 해석해서 django와 연동되게 해주고 알맞게 웹페이지를 불러오게 해준다. 또한 nginx는 정적인 웹만 지원하므로 동적 웹, 데이터베이스 연동을 위해 웹 어플리케이션인 django가 필요하다.

Client - Server(nginx) - Socket - Uwsgi - Django

그러므로 이러한 구조가 된다.

```
FROM ubuntu:latest
MAINTAINER Kim SooHyun <shkim950921@cs-cnu.org>

RUN apt-get update
RUN apt-get install -y net-tools
RUN apt-get install -y nginx
RUN apt-get -y install locales
RUN apt-get install -y mysql-server
RUN apt-get install -y python3
RUN apt-get install -y python3-pip
RUN apt-get install -y python3.6-dev
RUN apt-get install -y vim
RUN apt-get install -y python3-venv
RUN apt-get -y install sudo
RUN pip3 install virtualenv
RUN pip3 install django
RUN pip3 install uwsgi
RUN useradd -m network && echo "network:network" | chpasswd && adduser network sudo
USER network

CMD /bin/bash

ENV LC_ALL=C.UTF-8
ENV TERM=xterm-256color

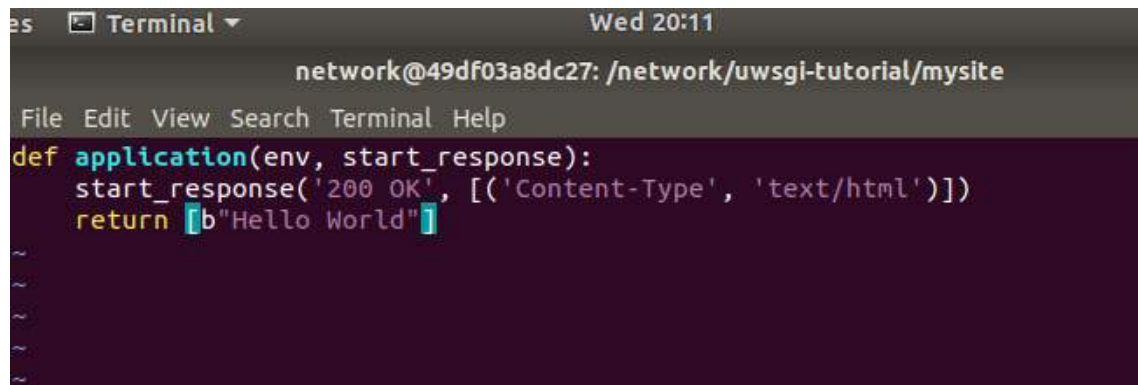
EXPOSE 80
EXPOSE 8000
EXPOSE 8080
```

먼저 위와 같이 Dockerfile을 만들어 이미지 생성 후 docker run -it -v /home/u201404377/u201404377/network:/network -p 1234:80 -p 443:443 -p 2345:8000 --name net_container_v0.5 network-class:0.1 /bin/bash 로 도커 컨테이너를 만들었다. 80포트를 1234로 8000포트를 2345로 mapping 시켰다.

uwsgi 설정을 시작하기에 앞서 가상환경에서 작업하기 위해 virtualenv와 source bin/activate 명령어를 통해 가상환경에 들어갔다. virtualenv는 python의 가상환경 모듈이라 볼 수 있다. 작은 python을 새로 설치해서 내가 원하는 모듈만 운용하는 바구니라고 생각하면 된다.

가상환경을 생성하고 접속 후 해당 가상환경을 만든 폴더로 들어간다음 Django 프로젝트를 생성해야한다. django-admin.py startproject mysite 명령어를 통해 mysite라는 이름의 Django 프로젝트를 생성하였다.

그 후 pip3 install uwsgi로 uwsgi를 설치 후 생성한 Django 프로젝트에 들어가 uwsgi를 테스트해봤다. tes.py는 다음과 같다.

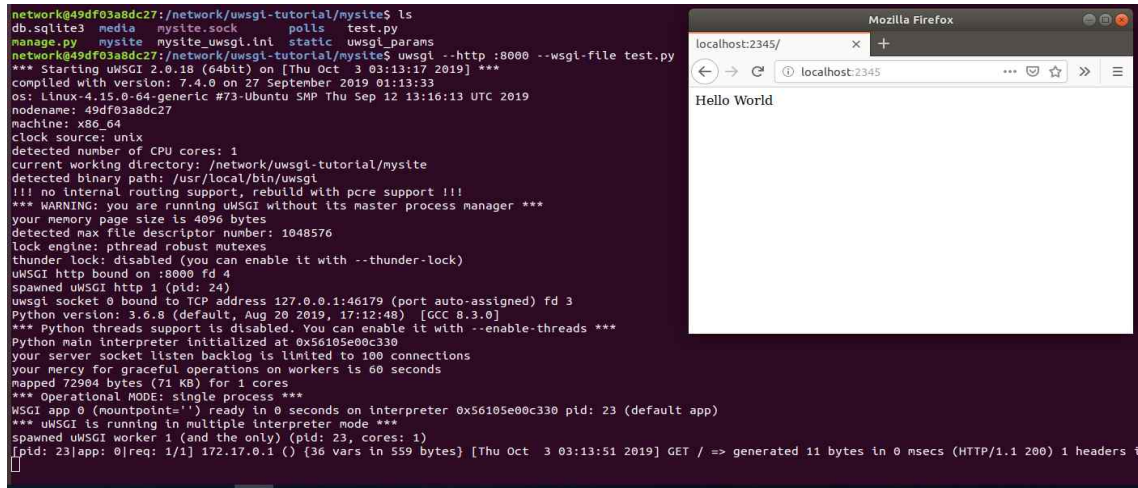
A terminal window titled 'Terminal' with a timestamp 'Wed 20:11'. The prompt is 'network@49df03a8dc27: /network/uwsgi-tutorial/mysite'. The terminal shows a Python function definition for 'application' that takes 'env' and 'start_response' as arguments. It calls 'start_response' with '200 OK' and a list of headers, then returns the bytes string 'Hello World'.

```
network@49df03a8dc27: /network/uwsgi-tutorial/mysite
File Edit View Search Terminal Help
def application(env, start_response):
    start_response('200 OK', [('Content-Type', 'text/html')])
    return [b"Hello World"]
```

다음은 uwsgi 테스트한 결과이다. 잘 동작함을 볼 수 있다.

uwsgi --http : 8000 --wsgi-file test.py

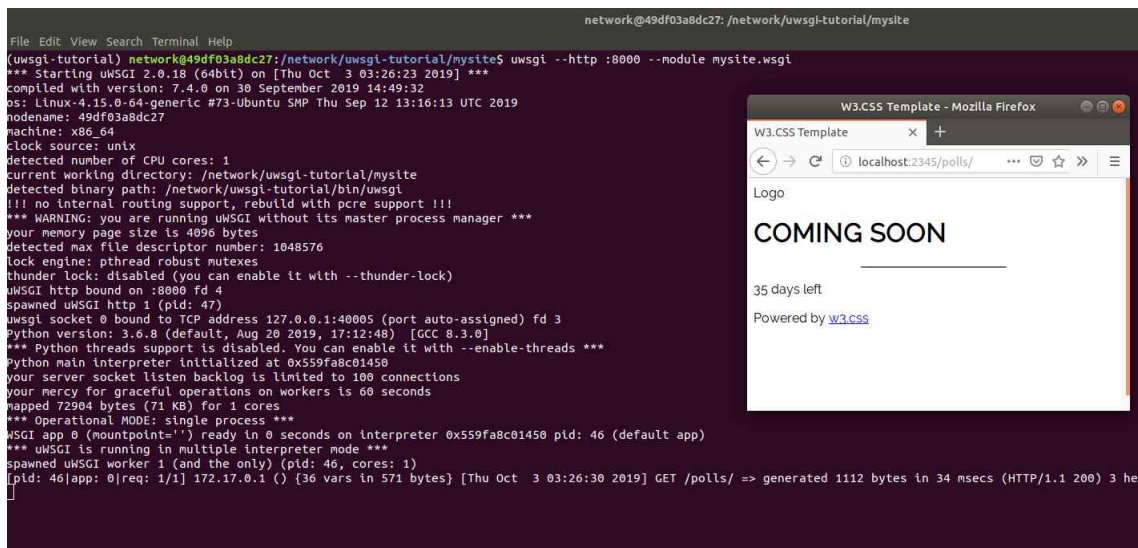
에서 http프로토콜을 사용하고 8000 포트를 사용하며 지정된 파일인 test.py를 로드한다.



The image shows a terminal window on the left and a Mozilla Firefox browser window on the right. The terminal displays the output of running 'uwsgi --http : 8000 --wsgi-file test.py'. It shows various system information, warnings, and a successful GET request from 172.17.0.1. The browser window shows 'localhost:2345/' with the text 'Hello World'.

이 테스트가 성공함으로써 client - uWsgi- python 간의 동작이 된다는 것을 의미한다. 다음은 test.py 모듈이 아니라 Djano 사이트를 실행 및 테스트를 해봤다. 다음과 같이 잘 동작함을 볼 수 있다.

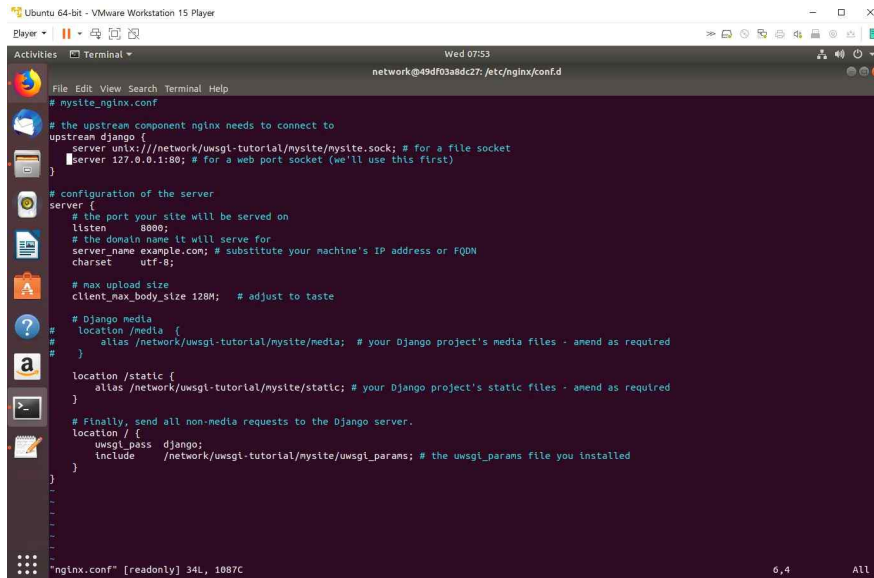
uwsgi --http :8000 --module mysite.wsgi



The image shows a terminal window on the left and a Mozilla Firefox browser window on the right. The terminal displays the output of running 'uwsgi --http :8000 --module mysite.wsgi'. It shows various system information, warnings, and a successful GET request for '/polls/' from 172.17.0.1. The browser window shows 'W3.CSS Template - Mozilla Firefox' with the text 'COMING SOON' and '35 days left'.

client - uWSGI - Django가 잘 연동됨을 볼 수 있다. 이제는 client와 Uwsgi가 직접 대화(연결)하지 않고 웹 서버가 할 역할이므로 nginx를 클라이언트와 연결해주면 된다.

nginx를 사용하려면 다음과 같은 설정이 필요하다.



```
# mysite_nginx.conf

# the upstream component nginx needs to connect to
upstream django {
    server unix:///network/uwsgi-tutorial/mysite/mysite.sock; # for a file socket
    server 127.0.0.1:80; # for a web port socket (we'll use this first)
}

# configuration of the server
server {
    # the port your site will be served on
    listen 8000;
    # the domain name it will serve for
    server_name example.com; # substitute your machine's IP address or FQDN
    charset utf-8;

    # max upload size
    client_max_body_size 128M; # adjust to taste

    # Django media
    location /media {
        alias /network/uwsgi-tutorial/mysite/media; # your Django project's media files - amend as required
    }

    location /static {
        alias /network/uwsgi-tutorial/mysite/static; # your Django project's static files - amend as required
    }

    # Finally, send all non-media requests to the Django server.
    location / {
        uwsgi_pass django;
        include /network/uwsgi-tutorial/mysite/uwsgi_params; # the uwsgi_params file you installed
    }
}
```

위의 conf파일은 nginx에게 파일 시스템에서 미디어 및 정적 파일을 제공하고 Django의 개입이 필요한 요청을 처리하도록 지시한다. (즉, nginx가 어떻게 동작해야 하는가를 지정하는 기능으로 파일에 설정 값을 기술한다)

여기서 server 블록은 하나의 웹사이트를 선언하는데 사용된다. location 블록은 server 블록 안에 등장하면서 특정 url을 처리하는 방법을 정의한다.

location /static 은 static한 resource를 서빙해야한다면 alias를 통해 서빙할 수 있다. 다른 static한 resource를 사용하고 싶으면 주석처리한 media처럼 location /media하고 적어주면 된다. 그리고 static한 자원들 외의 모든 요청은 location / 로 오게되고 이 블록에 있는 설정된 django로 보내지게 된다. (이 django가 위에 upstream django로 되어있는 곳으로 보내진다는 뜻이다.) 그러므로 uwsgi_pass를 위에 upstream으로 설정한 블록의 이름을 적어주면된다. uwsgi를 사용하려면 uwsgi_params파일이 필요하고 이 파일이 있는 경로를 include에 지정해주면 된다.

upstream django블록은 nginx가 요청을 전달할 django 서버의 정보를 적어주면 된다. 위에 2개가 있는데 하나는 unix파일 소켓을 사용하는 방법이고 다른 하나는 http 통신을 사용하는건데 전자가 더 가볍게 동작할 수 있어 좋다고 한다. 위 문단에서 말한 것처럼 location / 으로 온 요청들은 이 upstream으로 오게 된다.

다음은 uwsgi 설정한 사진이다.

```
[uwsgi]

chdir = /network/uwsgi-tutorial/mysite/

module = mysite.wsgi:application

home = /network/uwsgi-tutorial

master = true
process = 10
socket = /network/uwsgi-tutorial/mysite/mysite.sock
chmod-socket = 666
chown-socket = network:network
uid = network
gid = network
vacuum = true
enable-threads = true
pidfile = /tmp/mysite.pid

~
~
~
~
~
~
~
~
~
~

"mysite_uwsgi.ini" 18L, 334C
```

중요한 것만 설명하자면

먼저 chdir은 장고 프로젝트의 base directory를 의미한다.

module은 장고의 wsgi파일을 의미한다.

home은 가상환경인 virtualenv의 path를 의미한다.

socket은 해당 mysite.sock가 있는 mysite/ 디렉토리에 대해서 모든 권한을 갖게 해준다. 그 밑의 chown과 chmod는 socket의 소켓 소유자와 고유권한을 의미한다.

vacuum은 uWSGI 종료 시 socket파일 삭제 여부를 의미한다.

이렇게 ini파일들과 함께 실행되도록 uWSGI를 구성할 수 있다. 즉 uWSGI에 사용한 것과 동일한 옵션을 파일에 넣은 다음 uWSGI에 해당 파일을 실행하도록 요청할 수 있고 구성을보다 쉽게 관리 할 수 있다.

다음은 조교님의 영상에 있던 최종결과를 따라서 실행한 결과이다.

통신 구조는 다음과 같다.

nginx - unix socket - uWSGI - django - python, db

