**SW사관학교 정글 - Chapter 5**

**[수업 목표]**

1. Flask 프레임워크를 통해 웹서비스를 완성할 수 있다.
2. AWS의 EC2 서버를 이용해 웹서비스를 배포할 수 있다.

**01. 서버 구매**

웹서비스 런칭에 필요한 개념

* 👉 이제 내가 만든 프로젝트를 배포해봅니다. 배포는 누구나 내 서비스를 사용할 수 있게 하기 위해서 작업들이에요. 웹 서비스를 런칭하는 것이죠!
* 웹 서비스를 런칭하기 위해 클라이언트의 요청에 항상 응답해줄 수 있는 서버에 프로젝트를 실행시켜줄 것입니다. AWS 라는 클라우드 서비스에서 편하게 서버를 관리하기 위해서 항상 켜 놓을 수 있는 컴퓨터인 EC2 사용권을 구입해 서버로 사용하겠습니다.
* 클라우드 환경이 필요한 이유?
  + 1.컴퓨터가 항상 켜져있고 프로그램이 실행되어 있어야하고,
  + 2.모두가 접근할 수 있는 공개 주소인 공개 IP 주소(Public IP Address)로 나의 웹 서비스에 접근할 수 있도록 해야하기 때문이죠.
  + 3.내 컴퓨터를 남이 접속할 수 있게 계속 켜둘 수는 없으니까요.
* IP 주소와 포트
  + 사실 우리가 접속하는 컴퓨터는 숫자로 되어있는 주소(IP 주소)가 붙어있습니다. 그치만 매번 172.217.. 과 같이 입력하는 것은 매우 번거롭겠죠. 그래서 IP와 읽기좋은 주소를 매칭해주는 '도메인'이라는 개념이 탄생하게 되었습니다.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

* + IP주소란?
    - 인터넷 위에서 컴퓨터가 통신할 수 있도록 컴퓨터마다 가지는 고유한 주소라고 생각하면 됩니다. 각 서버는 하나의 주소를 가지고 있습니다.
  + 포트(port)란?
    - 하나의 IP에 여러 포트가 있습니다. 하나의 포트에 하나의 프로그램을 실행시킬 수 있습니다.
* AWS EC2 서버 구매하기
  + 💡 리눅스 OS, 정확히는 Ubuntu 배포판의 컴퓨터를 사서 서버로 사용하겠습니다. 오픈소스 운영체제로, 우리 주변 대부분의 서비스가 리눅스 OS로 서버를 운영하고 있습니다.
  + <https://ap-northeast-2.console.aws.amazon.com/ec2/v2/home?region=ap-northeast-2> (Seoul Region으로 들어가기)
  + EC2 서버 종료하는 방법 (1년 후 자동결제 방지)
    - 💡 **무료 기간(1년) 후 결제가 되기 전에, 이렇게 종료하세요**
    - 대상 인스턴스에 마우스 우클릭 → 인스턴스 상태 → 종료를 클릭하면 인스턴스를 완전히 삭제할 수 있습니다. 중지는 컴퓨터를 잠시 꺼놓은 것이고, 종료는 컴퓨터를 버리는 것으로 이해해주시면 됩니다.
* AWS EC2 접속 준비
  + 💡 EC2 접속을 위해, 윈도우는 Git bash 라는 프로그램을 먼저 설치해주세요.
  + [윈도우만] Git bash : ([설치하기](https://gabii.tistory.com/entry/Git-Git-Bash-219-%EC%84%A4%EC%B9%98%ED%95%98%EA%B8%B0))
  + 준비 상식 - SSH와 22번 포트
    - 💡 SSH란 다른 컴퓨터에 접속할 때 쓰는 프로그램이며, 서로 22번 포트가 열려있어야 접속 가능합니다. EC2의 경우, 이미 22번 포트가 열려있음을 확인할 수 있습니다.

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

* AWS EC2 접속
  + Mac OS: Mac은 ssh가 있어서, 명령어로 바로 접근 가능
    - 터미널을 열기 (spotlight에 terminal 입력)
    - 방금 받은 내 Keypair의 접근 권한을 바꿔주기
      * sudo chmod 400 받은키페어를끌어다놓기
    - SSH로 접속하기
      * ssh -i 받은키페어를끌어다놓기 ubuntu@AWS에적힌내아이피
    - 예) 아래와 비슷한 생김새입니다.
      * ssh -i /path/my-key-pair.pem [ubuntu@13.125.250.20](mailto:ubuntu@13.125.250.20)
    - Key fingerprint 관련 메시지가 나올 경우 Yes를 입력해주세요!
  + Window: ssh가 없으므로, git bash라는 프로그램을 이용
    - gitbash를 실행하고, 아래를 입력!
      * ssh -i 받은키페어를끌어다놓기 ubuntu@AWS에적힌내아이피
    - 예) 아래와 비슷한 생김새!
      * ssh -i /c/Users/bumky/Desktop/myec2test.pem [ubuntu@13.125.149.68](mailto:ubuntu@13.125.149.68)
    - Key fingerprint 관련 메시지가 나올 경우 Yes를 입력해주세요!
    - git bash를 종료할 때는 exit 명령어를 입력하여 ssh 접속을 먼저 끊어주세요.
* 접속 완료 된 화면
  + 💡 아래와 같은 화면을 마주하면, EC2 원격 접속에 성공한 것입니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**02. 서버 설정**

리눅스 명령어

* 리눅스는 윈도우 같지 않아서, '쉘 명령어'를 통해 OS를 조작한다. (일종의 마우스 역할)
* [가장 많이 쓰는 몇 가지 명령어]
  + 팁! 리눅스 커널에서 윗화살표를 누르면 바로 전에 썼던 명령어가 나옵니다.
  + ls: 내 위치의 모든 파일을 보여준다.
  + pwd: 내 위치(폴더의 경로)를 알려준다.
  + mkdir: 내 위치 아래에 폴더를 하나 만든다.
  + cd [갈 곳]: 나를 [갈 곳] 폴더로 이동시킨다.
  + cd .. : 나를 상위 폴더로 이동시킨다.
  + cp -r [복사할 것] [붙여넣기 할 것]: 복사 붙여넣기
  + rm -rf [지울 것]: 지우기
  + sudo [실행 할 명령어]: 명령어를 관리자 권한으로 실행한다.
  + sudo su: 관리가 권한으로 들어간다. (나올때는 exit으로 나옴)

filzilla 이용하기

* 👉 우리 EC2에 파일 업로드 해보겠습니다.
* 설치 하기
  + 다운로드: <https://filezilla-project.org/download.php>
  + 다운로드 클릭 후 가장 기본 버전 (스크린샷 기준 왼쪽) 다운로드!

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

* python 파일 올리기
  + 1.서버에 업로드해 실행할 간단한 파일을 만듭니다. [hello.py](<http://hello.py>) 로 파일을 만든 후 sparta 폴더 안에 아래 코드를 복사 붙여넣기 해주세요.
    - print('Hello World')
  + 2.파일질라 실행, 다음과 같이 설정
    - EC2 서버에 ubuntu라는 user로 접속하는 설정입니다. 받아둔 키파일을 사용합니다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* + 3.[hello.py](<http://hello.py>) 파일을 EC2서버에 업로드합니다.
    - 마우스로 드래그 해서 파일을 업로드/다운로드할 수 있습니다. 기본 설정은 왼쪽이 내 컴퓨터, 오른쪽이 원격접속한 EC2 인스턴스 입니다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

파이썬 파일 실행

* 💡 아래 명령어를 차례대로 입력합니다.
  + # python 이라는 명령어로 3 버전 이상을 실행하도록 하는 명령어입니다.
  + sudo apt install python-is-python3
  + # home 디렉토리로 이동
  + cd ~
  + # 실행. 콘솔창에 hellow world!가 뜨는 것을 확인 할 수 있습니다.
  + python hello[.py](<http://test1.py>)

**03. 서버 실행하기**

flask 서버를 실행해보기

* 패키지 설치를 도와줄 패키지, pip 설치하기
  + # pip3 설치
  + sudo apt update
  + sudo apt install -y python3-pip
  + # 버전 확인
  + pip –version
* 설치한 pip를 이용해 flask 설치
  + 💡 앞으로 패키지 설치할 때에는 아래처럼 명령어를 작성하면 됩니다.
  + pip install [패키지이름]
  + pip install flask
* 기초적인 flask 서버를 실행해보겠습니다. [app.py](<http://app.py>) 로 아래 내용을복사 붙여넣기 해주세요. 그 후, 파일질라에서 드래그 드롭으로 EC2 인스턴스의 home/ubuntu/jungle 폴더에 해당 파일을 업로드합니다.
  + from flask import Flask
  + app = Flask(\_\_name\_\_)
  + @app.route('/')
  + def home():
  + return 'This is Home!'
  + if \_\_name\_\_ == '\_\_main\_\_':
  + app.run('0.0.0.0', port=5000, debug=True)
* jungle 폴더에 들어가서, 쉘에 아래와 같이 입력하면, flask가 실행됩니다.
  + # app.py 파일 파이썬으로 실행
  + python app.py

AWS에서 5000포트를 열어주기

* 💡 그러나, 아직 ip로 접속하면 오류가 발생합니다. AWS에서 아직 포트번호를 막고 있기 때문입니다.
* EC2에서 5000포트로 들어오는 요청을 받을 수 있게 설정해야합니다. AWS EC2 Security Group에서 요청 포트를 열어주면 됩니다.
* 일단, EC2 관리 콘솔로 들어갑니다. 그리고 보안그룹(영문: Security Group)을 눌러 들어갑니다. 여기선 launch-wizard-1 이라고 쓰여 있네요

텍스트, 소프트웨어, 컴퓨터 아이콘, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

* 해당 보안그룹을 클릭합니다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명- Edit inbound rules를 선택합니다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명- 세 가지 포트를 추가해봅니다. (80, 5000포트는 미리 추가해두겠습니다)

→ 5000포트: flask 기본포트

→ 80포트: HTTP 접속을 위한 기본포트

→ 27017포트: 외부에서 mongoDB 접속을 하기위한 포트

텍스트, 소프트웨어, 컴퓨터 아이콘, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

* 다시 접속해봅니다!
  + http://[내 EC2 퍼블릭 IP]:5000/
  + → 잘 작동하는 것을 확인할 수 있습니다.

**04. 서버에 나홀로메모장 업데이트**

나홀로메모장 완성본을 filezilla로 EC2에 업로드해봅니다.

* 파일질라에서 나홀로메모장 파일들을 jungle 폴더에 업로드합니다.
  + 👉 app.py, templates 폴더, static 폴더를 모두 옮겨주세요.
* 완성본을 실행해봅니다.
  + 1.파일을 실행하기 전, 사용하는 패키지를 설치합니다.
    - # 패키지 설치하기
    - pip install requests beautifulsoup4 pymongo
  + 2.실행하기
    - # 해당 디렉토리에서 파이썬으로 코드를 실행합니다.
    - python app.py

접속해봅니다.

- <http://내AWS아이피:5000/>

**05. 서버 완성**

포트 포워딩

* 포트 번호 없애기 - 기본 개념
  + 지금은 5000포트에서 웹 서비스가 실행되고 있습니다. 그래서 매번 :5000 이라고 뒤에 붙여줘야 하죠. 뒤에 붙는 포트 번호를 없애려면 어떻게 해야할까요?
  + http 요청에서는 80포트가 기본이기 때문에, 굳이 :80을 붙이지 않아도 자동으로 연결이 됩니다.
  + 포트 번호를 입력하지 않아도 자동으로 접속되기 위해, 우리는 80포트로 오는 요청을 5000 포트로 전달하게 하는 포트포워딩(port forwarding) 을 사용하겠습니다.
  + 리눅스에서 기본으로 제공해주는 포트포워딩을 사용할 것입니다. 그림으로 보면 아래와 같습니다.

텍스트, 스크린샷, 폰트, 로고이(가) 표시된 사진

자동 생성된 설명

* 포트 번호 없애기 - 리눅스 자체 포트포워딩을 작동시키기
  + 띄워둔 터미널에서 새롭게 설정을 적용하기 전에 돌아가고 있던 서비스(Flask)는 일단 중지합니다. (터미널에서 ctrl + c)
  + 포트포워딩 룰을 입력합니다.
    - sudo iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 5000
  + 다시 서비스 시작
    - python app.py
  + 웹 브라우저에서 접속해보기
    - <http://내AWS아이피/>

nohup 설정

* 현재 상황
  + 👉 Git bash 또는 맥의 터미널을 종료하면 (=즉, SSH 접속을 끊으면) 프로세스가 종료되면서, 서버가 돌아가지 않고 있습니다. 그러나 우리가 원격접속을 끊어도, 서버는 계속 동작해야겠죠?
* 원격 접속을 종료하더라도 서버가 계속 돌아가게 하기
  + # 아래의 명령어로 실행하면 된다
  + nohup python app.py &
* 서버 종료하기 - 강제종료하는 방법
  + 👉 프로세스 번호는 빨간 화살표를 참고해주세요

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* + # 아래 명령어로 미리 pid 값(프로세스 번호)을 본다
  + ps -ef | grep 'app.py'
  + # 아래 명령어로 특정 프로세스를 죽인다
  + kill -9 [pid값]
* 다시 켜기
  + nohup python app.py &
* 브라우저에서 접속하기
  + <http://내AWS아이피/>