



# 스파르타코딩클럽 8기 6주차



매 주차 강의자료 시작에 PDF파일과 영상 링크를 올려두었어요!

## ▼ PDF 강의자료 다운받기

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/9c5b031c-f784-4b8f-8745-fa2749928781/scc-8-1.pdf>

## ▼ 영상강의 참고하기

- 1주차 복습용 영상강의 링크

## [수업 목표]

1. EC2 서버를 구매 & 접속한다.
2. EC2 서버에서 flask 서버를 돌릴 수 있다.

전반 3시간

## [시작] : 체크인 & 출석체크



튜터님은 체크인과 함께 출석 체크(링크)를 진행해주세요!

## ▼ "15초 체크인"을 진행합니다.

- 튜터님은 타이머를 띄워주세요! (링크)
- 본인의 감정상태와 오늘 있었던 소소한 일을 공유하는 시간입니다.

## [2시간]: 수업 (서버 구매부터 간단한 flask 서버 올리기까지)

### ▼ 1) 6~8주차 수업 설명: "웹서비스를 런칭하기 위한 작업들"

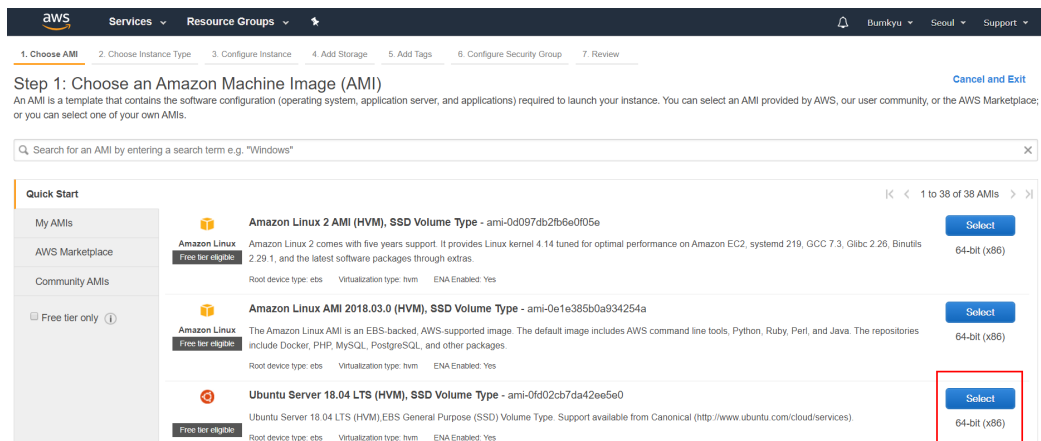
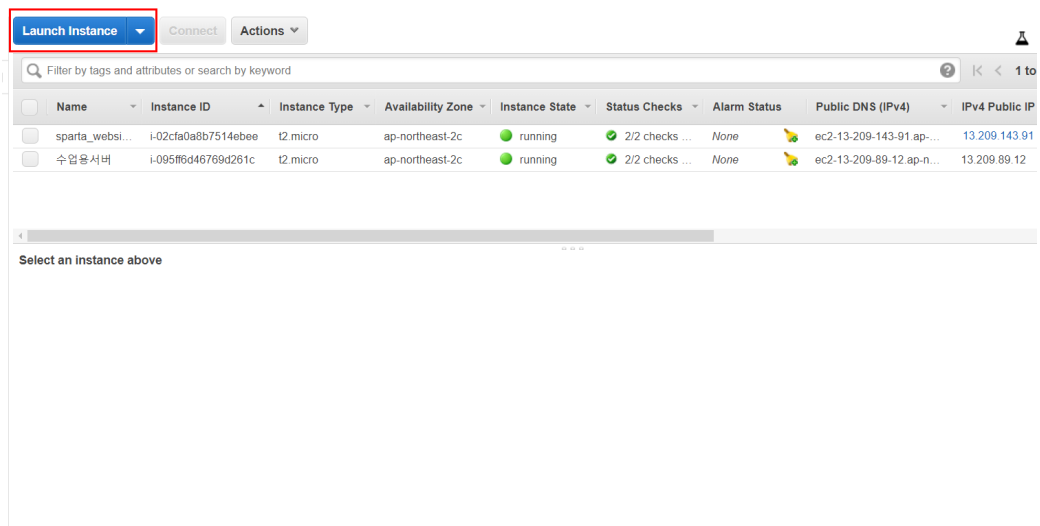
- 6주차(오늘): AWS에서 EC2 서버 구매하기~EC2 서버에서 간단한 flask 서버 실행해보기
- 7주차: mongoDB 설치하기+5주차 완성본을 서버에서 실행하기
- 8주차: 도메인 붙이기, 포트 번호 떼기

### ▼ 2) EC2 서버 구매하기

#### ▼ AWS EC2 서버 사기

- <https://ap-northeast-2.console.aws.amazon.com/ec2/v2/home?region=ap-northeast-2>  
(Seoul Region으로 들어가기)

#### ▼ 구매 화면들 따라하기



1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

### Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types Current generation Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	General purpose	t2.micro	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

Cancel Previous **Review and Launch** Next: Configure Instance Details

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

### Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

AMI Details [Edit AMI](#)

**Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0fd02cb7da42ee5e0**  
 Free tier eligible Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).  
 Root Device Type: ebs Virtualization type: hvm

Instance Type [Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups [Edit security groups](#)

Security group name: launch-wizard-10  
 Description: launch-wizard-10 created 2019-10-03T15:25:50.845+09:00

Type	Protocol	Port Range	Source	Description
This security group has no rules				

Cancel Previous **Launch**

### Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. [Learn more about removing existing key pairs from a public AMI.](#)

Create a new key pair

**Key pair name**  
 sparta-newkey

**Download Key Pair**

You have to download the **private key file** (\*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel **Launch Instances**

▼ 3) EC2에 접속하기

▼ 4) 간단한 리눅스 명령어 연습하기

- 리눅스는 윈도우 같지 않아서, '셸 명령어'를 통해 OS를 조작한다. (일종의 마우스 역할)

#### [가장 많이 쓰는 몇 가지 명령어]

```
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)
```

- 함께 해보기: 임의의 폴더를 만들어보고, 지워보기

#### ▼ Tips !

- 리눅스 커널에 붙여넣기를 할 때는 마우스 오른쪽 버튼을 이용하면 됩니다.
- 리눅스 커널에서 윗화살표를 누르면 바로 전에 썼던 명령어가 나옵니다.

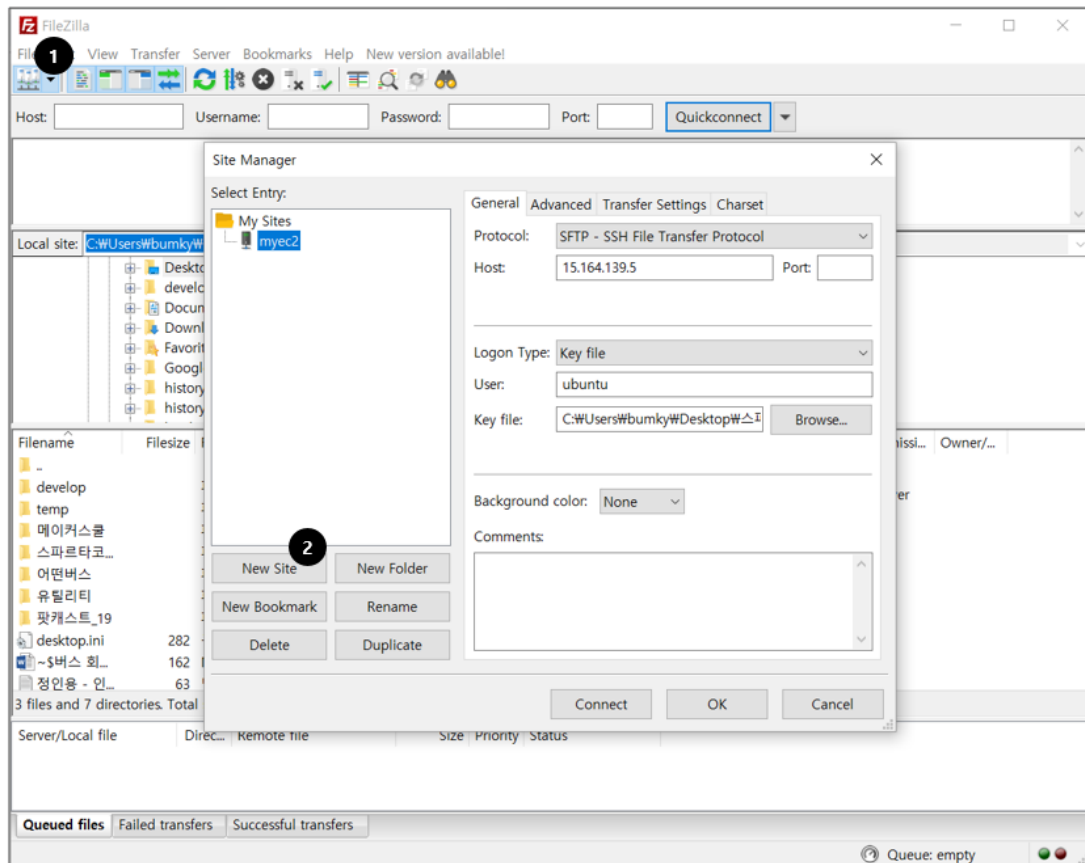
#### ▼ 6) filezilla를 이용해서, 간단한 python 파일을 올려봅니다.

- 서버에 업로드 할 간단한 파일을 만듭니다.

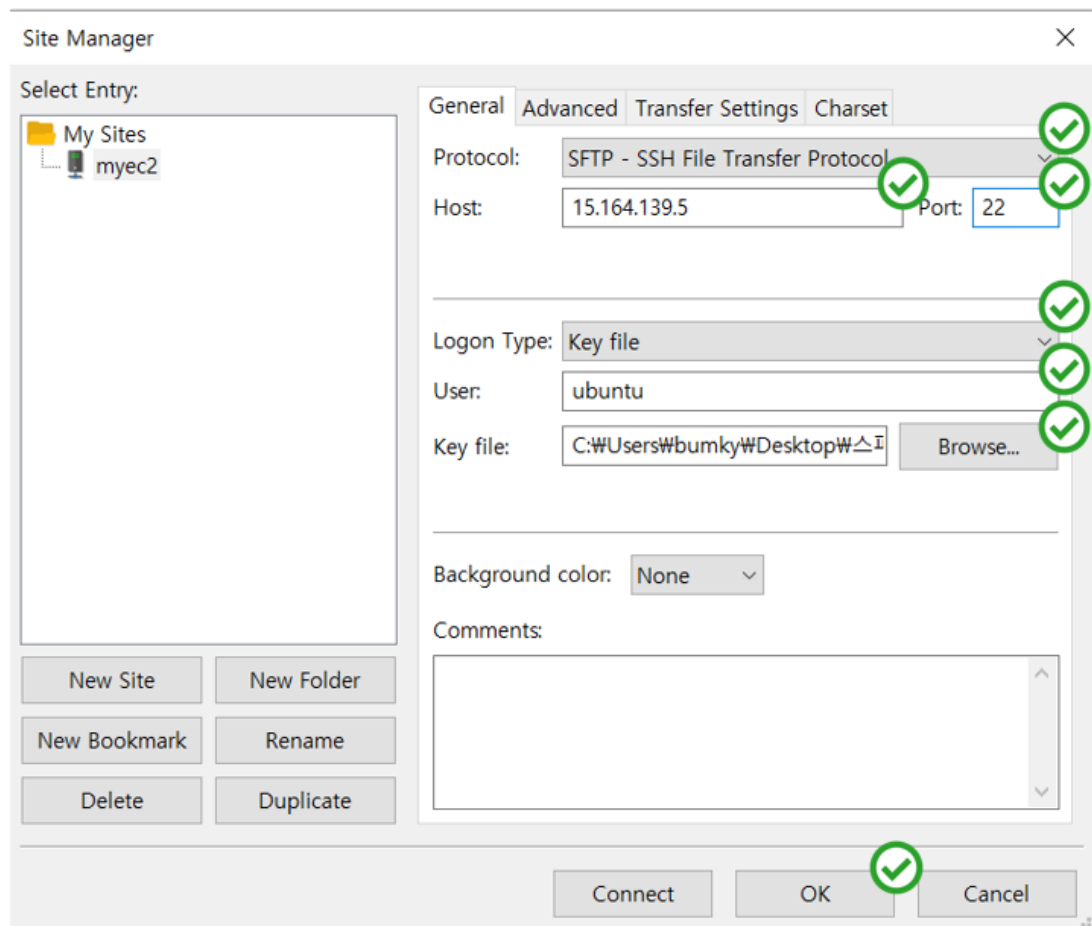
예) test.py

```
# 아주 간단하게, 이 정도만 적어볼까요?
# 그리고 적당한 곳에 파일을 저장해봅니다.
print('hello word!')
```

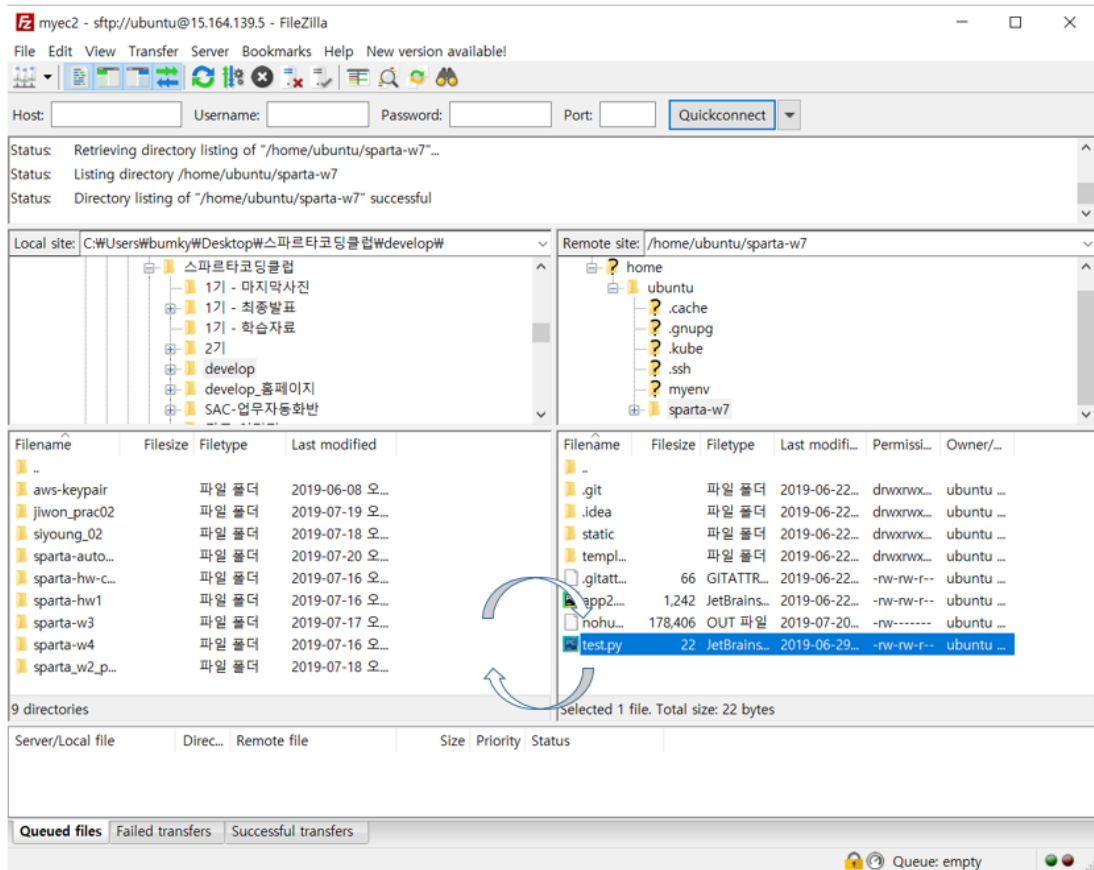
- 파일질라 실행, 다음과 같이 설정



- 정보들을 입력하고, ok 누르면 서버의 파일들을 볼 수 있음  
(Host: 내 EC2서버의 ip // User: ubuntu 로 입력)



- 마우스로 드래그 해서 파일을 업로드/다운로드하면 됩니다!  
(자, 그럼 이제 간단한 파이썬 파일을 하나 만들어서 업로드 해볼까요?)



## ▼ 7) 파이썬 파일 실행해보기

- EC2 콘솔창에서 아래와 같이 입력합니다.

```
# python 이라는 명령어로 3 버전 이상을 실행하도록 하는 명령어입니다.
sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 10

# home 디렉토리로 이동
cd ~

# 실행. 콘솔창에 hellow world!가 뜨는 것을 확인 할 수 있습니다.
python test.py
```

## ▼ 8) flask 서버를 실행해보기

- 기초적인 flask 서버 파일을 하나 만들어봅니다.

예) app.py 파일

```
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)
```

- filezilla를 통해 EC2에 업로드한 다음, 실행해봅니다.

```
# 실행  
python app.py
```



에러가 납니다! 뭐라고 에러가 나는가요?



정답: flask 패키지가 없는데? - 라는 에러입니다.  
그럼, 패키지를 설치해볼까요?

#### ▼ 9) 리눅스에서 패키지 설치하기

- 우리가 여태까지 패키지 설치에 사용하던 pip 는 AWS에 설치되어있지 않아서 설치해주어야 합니다. python 버전 3을 위해서는 pip3을 설치해야 합니다.
- 바로 pip3를 설치해보겠습니다.

```
# pip3 설치  
sudo apt-get update  
sudo apt-get install -y python3-pip  
  
# 버전 확인  
pip3 --version  
  
# pip3 대신 pip 라고 입력하기 위한 명령어  
# 아래 명령어를 입력하면 pip 라고 쳐도 pip3를 작동시킬 수 있습니다.  
sudo update-alternatives --install /usr/bin/pip pip /usr/bin/pip3 1
```

- 설치한 pip를 이용해 해당 가상환경에 flask 설치

```
pip install flask
```

#### ▼ 10) 다시 flask 서버를 실행해보기

- 아래 명령어로 flask 서버를 실행합니다.



```
python app.py
```

- 서버 실행이 되면, 크롬에서 접속을 해봅니다.

크롬 브라우저 창에 아래와 같이 입력합니다.

```
http://[내 EC2 IP]:5000/
```



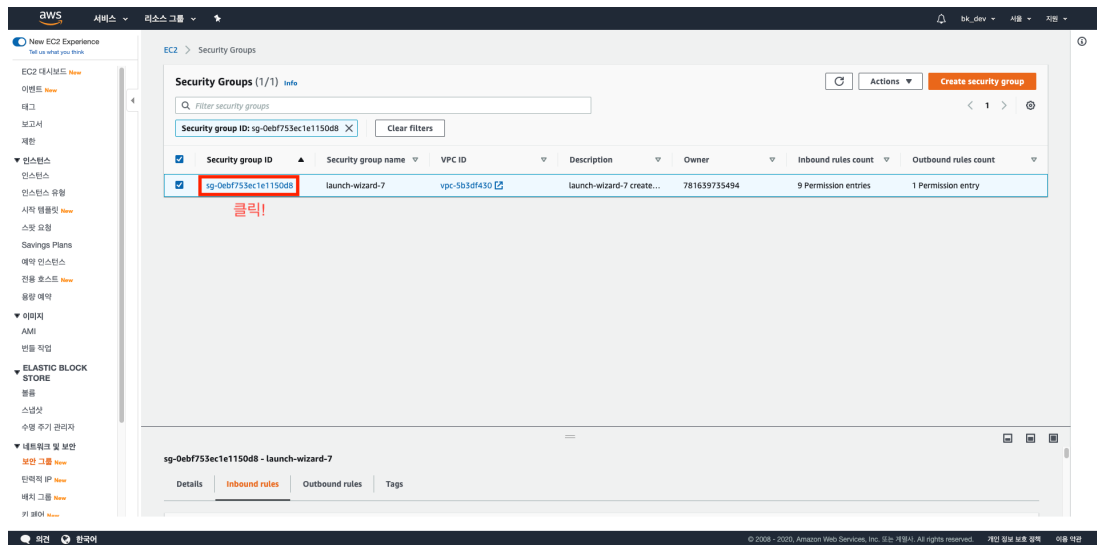
아직, 작동하지 않을 걸요! → AWS에서 약간의 설정이 더 필요합니다.

## ▼ 11) AWS에서 5000포트를 열어주기

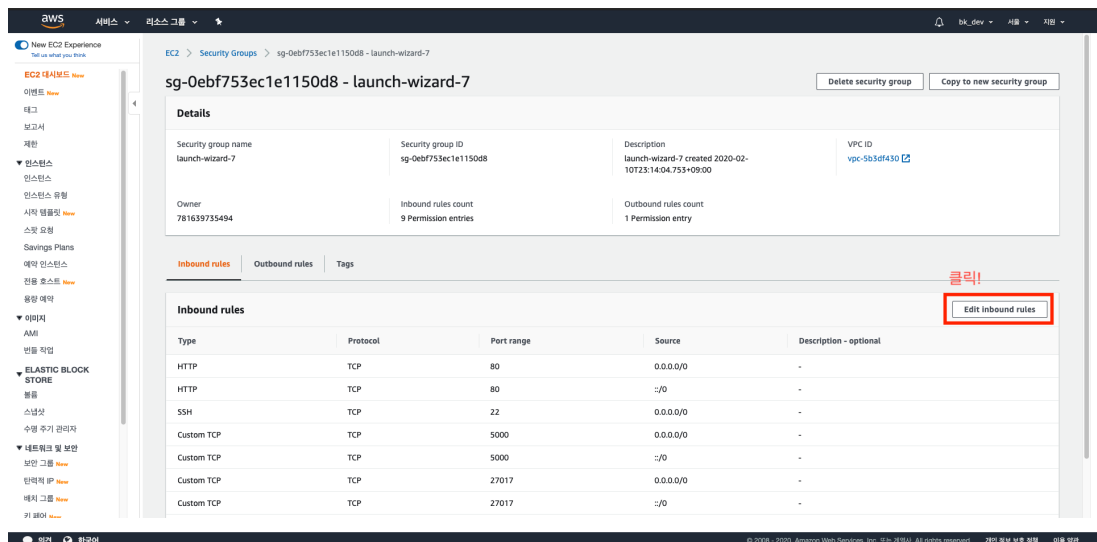
- EC2 서버(=가상의 내 컴퓨터)에서 포트를 따로 설정하는 것 외에도, AWS EC2에서도 자체적으로 포트를 열고/닫을 수 있게 관리를 하고 있습니다.  
→ 그래서 AWS EC2 Security Group에서 인바운드 요청 포트를 열어줘야 합니다.
- 일단, EC2 관리 콘솔로 들어갑니다. 그리고 보안그룹(영문: Security Group)을 눌러 들어갑니다. 여기서 launch-wizard-1 이라고 쓰여 있네요

The screenshot shows the AWS Management Console interface. On the left, the navigation menu includes 'EC2 대시보드', '인스턴스', 'AMI', 'ELASTIC BLOCK STORE', '스냅샷', '수명 주기 관리자', '네트워크 및 보안', '보안 그룹', and '탄력적 IP'. The '인스턴스' section is selected, showing a table of instances. The instance 'sparta\_website' (ID: i-02cfa0a8b7514ebee) is highlighted. Below the table, the '인스턴스: i-02cfa0a8b7514ebee (sparta\_website\_new)' details are shown. The '설명' tab is active, displaying various attributes: 인스턴스 ID, 인스턴스 상태 (running), 인스턴스 유형 (t2.micro), 탄력적 IP (13.209.143.91), 가용 영역 (ap-northeast-2), 보안 그룹 (launch-wizard-1), 예약된 이벤트 (없음), AMI ID (ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20190722.1), 플랫폼 (없음), IAM 역할 (sparta\_bk\_190805), 퍼블릭 DNS (IPv4), IPv4 퍼블릭 IP (13.209.143.91), 프라이빗 DNS (없음), 프라이빗 IP (172.31.17.74), 보조 프라이빗 IP (없음), VPC ID (vpc-39d41150), 서브넷 ID (subnet-1c8f156), 네트워크 인터페이스 (eth0), 소스/대상 확인 (예), T2/T3 무제한 (비활성).

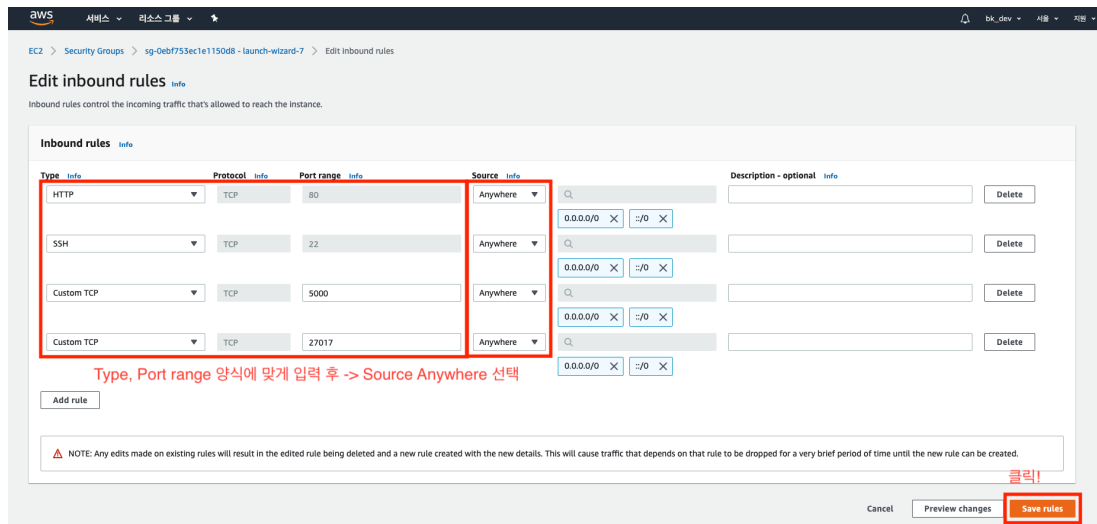
- 해당 보안그룹을 클릭합니다.



- Edit inbound rules를 선택합니다.



- 세 가지 포트를 추가해봅니다.
- 80포트: HTTP 접속을 위한 기본포트
- 5000포트: flask 기본포트
- 27017포트: 외부에서 mongoDB 접속을 하기위한 포트 (7주차에 쓰여요!)



## ▼ 12) 다시 접속해봅니다!

- `http://내아이피:5000`  
→ 잘 작동하는 것을 확인할 수 있습니다.
- 이제 `app.py` 파일을 수정하고, 다듬어서 올리면 진짜 프로젝트가 되겠죠?

## [1시간]: API 설계해보기

### ▼ 13) API 설계란?

- API는 클라이언트와 서버 간의 약속이라는 점 기억하실까요?
- 우리는 벌써 많은 API를 계획하고 만들었습니다.



### 모두의책리뷰

목적	API주소 (요청방법)	클라이언트는 (클라이언트 → 서버)	서버는 (서버 → 클라이언트)
리뷰조회	/reviews (GET)	줄 것 없음	리뷰를 전부 찾아 돌려준다.
리뷰작성	/reviews (POST)	제목, 저자, 리뷰를 주고	DB 추가 후 성공 메시지를 돌려준다.



### 나홀로메모장

목적	API주소 (요청방법)	클라이언트는 (클라이언트 → 서버)	서버는 (서버 → 클라이언트)
메모조회	/memo (GET)	줄 것 없음	메모를 전부 찾아 돌려준다.
메모작성	/memo (POST)	주소, 메모를 주고	주소 기반으로 크롤링하고, DB 추가 후 성공 메시지를 반환



### 마이페이보릿무비스타

목적	API주소 (요청방법)	클라이언트는 (클라이언트 → 서버)	서버는 (서버 → 클라이언트)
배우조회	/api/list (GET)	줄 것 없음	영화인을 찾아 돌려준다.
좋아요 +1	/api/like (POST)	배우 이름을 주고	좋아요 증가 후 성공 메시지를 돌려준다.
삭제하기	/api/delete (POST)	배우 이름을 주고	삭제 후 성공 메시지를 돌려준다.

## ▼ 14) 내 프로젝트 API 설계해보기



### 그리고보니 공통점이 있네요!

클라이언트는 (~) 을 주고) - 서버는 (~)하고 (~)를 돌려준다



지금부터 API 설계를 시작하고 궁금한 점이 있으면 언제든지 튜터에게 질문합니다.

아래 박스를 채워주세요!

목적	API주소 (요청방법)	클라이언트는 (클라이언트 → 서버)	서버는 (서버 → 클라이언트)
⋮ (필요한 만큼)			

### 후반 3시간

#### [월수/화목반] : 체크인 & 출석체크



튜터님은 체크인과 함께 출석 체크(링크)를 진행해주세요!

▼ "15초 체크인"을 진행합니다.

- 튜터는 타이머를 띄워주세요! (링크)
- 본인의 감정상태와 오늘 있었던 소소한 일을 공유하는 시간입니다.

#### [2시간]: 프로젝트 개발

▼ 14) 본격적으로 개발을 시작해볼까요!

- API 별로 차례차례 구현을 시작합니다.
- 필요한 기술이 있을 경우 자유롭게 공부합니다.

#### [1시간]: 튜터 피드백 - Todo 리스트

▼ 15) 개별 면담을 통해 Todo 리스트를 튜터와 함께 선정합니다.

- 각자 코딩을 하고, 튜터가 5분 씩 개별 면담을 진행하여 Todo 리스트를 함께 선정합니다.

- 예시1) "일단 HTML, CSS부터 뽐내고 갑시다! 다음주까지 프론트엔드 완성해주세요!"
- 예시2) "조회하려면 데이터가 있어야 하니, 다음 주까지 저장 API를 완성해보시죠!"
- 예시3) "이 프로젝트는 크롤러가 무척 중요하니 다음 주까지는 완성해주세요. 그 다음 바로 API 완성으로 넘어가시죠!"

## [끝]

▼ "15초 체크아웃"을 진행합니다.

- 튜터는 타이머를 띄워주세요! ([링크](#)).
- 다음 시간까지 해올 Todo 리스트에 대한 계획을 말해도 좋습니다!

## [숙제] - 다음 수업까지 개발일지 써오기

- 튜터와 함께 정한 Todo 리스트 개발일지를 작성합니다.



개발일지는, 본인을 위해 작성하는 것입니다! 꼭 일주일에 한 편씩만 작성하지 않아도 되며, 사실은 개발을 할 때마다 매일 작성하는 것이 가장 좋은 방법입니다.

TIL (today I learned)를 검색하면, 많은 분들의 개발일지를 볼 수 있습니다.

1. 한 주 동안의 회고
2. 한 주 동안의 배운 것들
3. 이번주의 목표

## [설치 해와야 할 것]

없음!