



스파르타코딩클럽 6주차



매 주차 강의자료 시작에 PDF파일과 영상 링크를 올려두었어요!

▼ PDF 강의자료 다운받기

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b6b1b639-095f-4531-87cd-fe9866acde83/week06.pdf>

▼ 영상강의 참고하기

- 곧 업로드 될 예정입니다!



모든 토클을 열고 닫는 단축키

Windows : **Ctrl** + **alt** + **t**

Mac : **⌘** + **option** + **t**

목차

수업 목표

체크인

배우고 적용하기 - 전반부

5분 꿀팁 - 클라우드란?

웹 기초 동작 원리 - 총 복습

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

[1시간]: API 설계해보기

배우고 적용하기 - 후반부

[0.5시간] 최종 프로젝트 기획안 발표하기

[0.5시간] 프로젝트 Github repository 만들기

[1시간] 튜터 피드백 - Todo 리스트

[남은 시간] 프로젝트 개발

[숙제 - 단짠단짠 🍕🍰] 개발일지 쓰기!

[숙제 제출]



체크아웃

[설치] - 다음 시간을 위해 미리 설치해와야 할 것들



수업 목표

1. 내 서비스 공개하기! 배포 첫 걸음
 - 1) 서버(EC2) 구매 후 접속하기
 - 2) 구입한 서버(EC2)에서 Flask 프로젝트 실행하기
2. 내 프로젝트 발전시키기

전반 3시간



체크인



튜터님은 체크인과 함께 출석 체크([링크](#))를 진행해주세요!

스파르타코딩클럽 출석체크

<http://spartacodingclub.shop/attendance>

▼ "15초 체크인"을 진행합니다.

- 튜터님은 타이머를 띄워주세요! ([링크](#))
- 본인의 감정상태와 오늘 있었던 소소한 일을 공유하는 시간입니다.
 - 예. 지난주에 등산을 했었는데요. 갑자기 아, 가보고 싶은 산들을 모아놓는 나만의 메모장을 만들고 싶더라고요! 제 프로젝트로 그거 해보려고요!



배우고 적용하기 - 전반부



5분 꿀팁 - 클라우드란?

▼ 클라우드란?

- 그동안 우리는 클라이언트와 서버가 같은 컴퓨터에 있기 때문에 별 다른 설정없이 접근할 수 있었어요.
- 만약 다른 컴퓨터(클라이언트)에서도 내가 만든 프로그램이 실행되고 있는 서버에 접근할 수 있게 하려면 어떻게 해야할까요?

- 모두가 접근할 수 있는 공개 주소인 공개 IP 주소(Public IP Address)로 나의 웹 서비스에 접근할 수 있도록 해야해요.
- 그리고 서버가 요청에 언제나 응답할 수 있게 서버가 항상 켜져있고, 웹 서비스가 항상 실행되어 있어야하죠.
- 여러가지 설정을 해서 내 컴퓨터를 항상 켜두어도 되지만, 우리는 웹 서버를 실행하기 편하도록 클라우드에서 서버 사용권을 구입할 거에요.
- ? 클라우드 많이 들어보긴 했는데 무슨 뜻인가요?
 - ! 클라우드(인터넷)을 통해 컴퓨터의 리소스를 사용할 수 있는 것입니다. 여기서 컴퓨터의 리소스는 컴퓨터를 이루고 있는 메모리, 저장장치(하드디스크, SSD), CPU 등을 이야기합니다.
 - 네이버 클라우드, 구글 드라이브에 파일을 저장하고 읽어올 수 있죠? 바로 원격으로 쓸 수 있는 저장장치를 산 것과 같죠.
- 우리는 AWS(Amazon Web Service)라는 곳의 EC2라는 서비스를 사용할 거에요. 원격으로 어딘가에 내가 사용할 수 있는 컴퓨터를 샀다고 생각하면 됩니다.

▼ [열 걸음 더 🚶] 클라우드 컴퓨팅의 특징 - 스킵하셔도 괜찮아요!

- 미국 국립표준연구소(NIST)가 정의한 클라우드 컴퓨팅은 아래와 같은 특징을 가지고 있어요. (원문. [The NIST Definition of Cloud Computing](#))

5 가지 주요 특징 Essential Characteristics

- On-demand self-service: consumer가 컴퓨팅 자원을 요구하는 즉시 자동으로 제공 (인간의 개입이 불필요).
- Broad network Access: 어디 있던 인터넷을 통해 리소스에 액세스
- Resource pooling: provider는 리소스 풀을 확보해서 multi-tenant model로 제공. 규모의 경제. 고객은 리소스 위치에 대해 신경쓸 필요없음.
- Rapid elasticity: 탄력적으로 리소스를 줄이거나 늘릴 수 있음(scale up and down). Capabilities can be elastically provisioned and released
- Measured service: 리소스 사용량이 측정되어서 쓴 만큼만 지불함. 투명성, 리소스 모니터링, 제어 및 보고 가능

3가지 서비스 모델 Service Models

- Software as a Service (SaaS) : 소프트웨어처럼 바로 사용할 수 있음. 예를 들면, MS 오피스 365, 구글 클라우드, 네이버 클라우드

- Platform as a Service (PaaS) : 응용 프로그램(Application)을 작성하고 실행할 수 있는 환경을 제공하는 것. 예를 들면, Google App Engine, Heroku
- Infrastructure as a Service (IaaS) : 사용할 수 있는 인프라를 제공하는 것. 예를 들면, 우리가 사용할 AWS, GCP

4가지 배포 모델 Deployment Models

- Private cloud : 비공개 클라우드. 단일 조직에서만 독점적으로 사용하는 것.
- Community cloud : 특정 용도로만 제한된 조직에서만 사용하는 것.
- Public cloud : 공개 클라우드. 일반인이 공개적으로 사용할 수 있는 것. 대부분의 클라우드 서비스는 여기에 속하겠죠?
- Hybrid cloud : 두 종류 이상의 클라우드로 구성된 것.

웹 기초 동작 원리 - 총 복습

▼ 총 복습 - HTTP 요청(GET/ POST), HTTP 응답, API, Flask

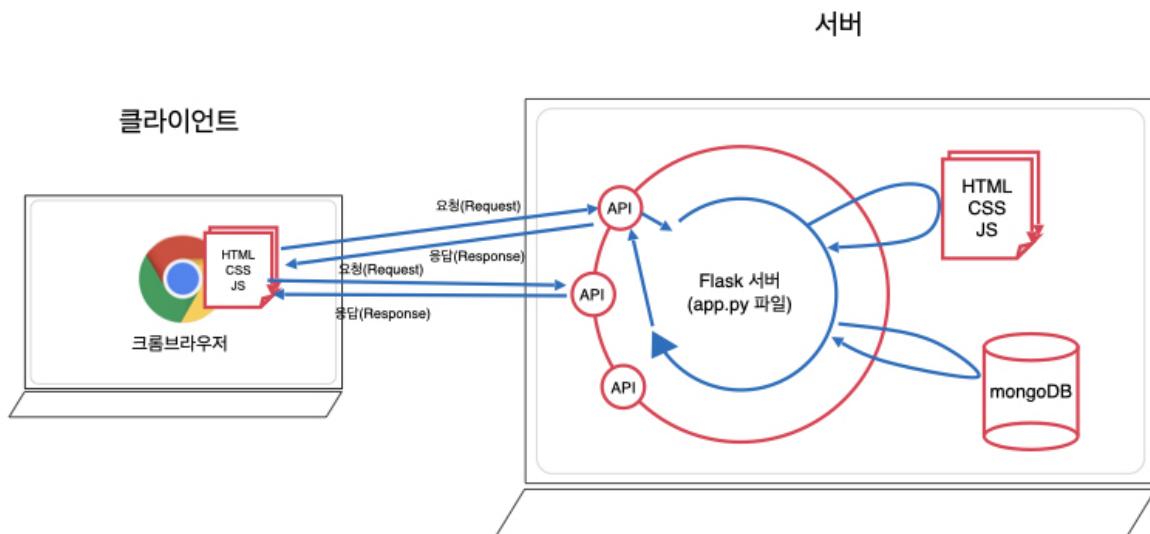
- HTTP: 웹은 HTTP라는 규약(규칙)을 따릅니다. url에서 `http://` 가 바로 HTTP라는 규약을 따른다는 표시예요.
- 클라이언트 : HTTP에서 요청을 하는 쪽
- 서버 : HTTP에서 요청을 받아 응답하는 쪽입니다. 응답하는 데이터로는 화면 코드(HTML, CSS, JS) 와 JSON같은 데이터 파일 등이 있습니다.
- 클라이언트 요청 방식 - GET, POST
 - GET → 통상적으로! 데이터 조회(Read)를 요청할 때
예) 영화 목록 조회
→ 데이터 전달 : URL 뒤에 물음표를 붙여 key=value로 전달
→ 예: google.com?q=북극곰
 - POST → 통상적으로! 데이터 생성(Create), 변경(Update), 삭제(Delete) 요청 할 때
예) 회원가입, 회원탈퇴, 비밀번호 수정
→ 데이터 전달 : 바로 보이지 않는 HTML body에 key:value 형태로 전달
- API(Application Programming Interface): 응용 프로그램(application, 애플리케이션)에서 기능을 사용하거나 데이터를 주고 받기 위한 기능

- API 를 사용할 땐, 미리 정해둔 **약속** 을 따라야 작동합니다. 약속들은 API 페이지(문서)에 적혀있습니다.
- 우리가 사용하는 API는 아래 적힌 모든 것을 미리 약속해두고 그대로 동작합니다.
 - 1. 요청 정보 :** 요청 URL, 요청 방식 (GET / POST /...)
 - 2. 서버가 제공할 기능 :** 데이터 조회(Read), 데이터 생성(Create) 등
 - 3. 응답 데이터 :** 응답 데이터 형식 (어떤 key 로 어떤 데이터를 줄지, 예. response['img'])
- Flask : 웹을 만들고 서버를 구동시키기 편하게 하는 프레임워크(Framework) ([Flask 공식 문서](#) / [비공식 한글 번역문서](#))
 - `@app.route('/')` 부분을 수정해서 URL을 부여할 수 있습니다
 - **templates 폴더** : HTML 파일을 담아둡니다. 실행할 때에는 이 폴더에서 화면을 불러오죠.
 - **static 폴더** : 이미지나 css파일과 같은 정적 파일을 담아두는 역할을 하지요!



서버와 클라이언트가 하나의 컴퓨터에 있던 로컬 개발 환경에서 한 걸음 더 나아가

오늘은 클라우드에서 구입한 서버에서 Flask 프로젝트를 실행시켜 보겠습니다.
실제 서비스 환경처럼 말이죠!



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

▼ 1) "웹서비스를 런칭하기 위한 작업"에 필요한 개념 소개

- 배포

- 이제 내가 만든 프로젝트를 배포해봅니다. 배포는 누구나 내 서비스를 사용할 수 있게 하기 위해서 작업들이에요. 웹 서비스를 런칭하는 거죠!
- 웹 서비스를 런칭하기 위해 클라이언트의 요청에 항상 응답해줄 수 있는 서버에 프로젝트를 실행시켜줄 거예요.
언제나 요청에 응답하려면,
 - 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 해야해요.
- 서버는 그냥 컴퓨터라는거 기억나시죠? 외부 접속이 가능하게 설정한 다음에 내 컴퓨터를 서버로 사용할 수도 있어요.
- 우리는 AWS라는 클라우드 서비스에서 편하게 서버를 관리하기 위해서 항상 켜놓을 수 있는 컴퓨터인 EC2 사용권을 구입해 서버로 사용할 겁니다.

▼ [열 걸음 더] IP 주소와 포트

- 사실 우리가 접속하는 컴퓨터는 숫자로 되어있는 주소(IP 주소)가 붙어있어요. 우리가 아는 URL은 우리가 알아보기 쉽게 하는 등의 이유로 IP 주소를 알파벳으로 바꾼 거에요. 이렇게 변환해주는 시스템을 DNS라고 합니다.

http://google.com

URL

DNS(Domain Name System)

http://172.217.25.14:80

IP

Port

- IP 주소(줄여서 IP라고 부릅니다)
 - 컴퓨터가 통신할 수 있도록 컴퓨터마다 가지는 고유한 주소라고 생각하면 됩니다. 정확히는 네트워크가 가능한 모든 기기가 통신할 수 있도록 가지고 있는 특수한 번호입니다. 서버는 하나의 주소를 가지고 있습니다.
 - 포트(port) : 하나의 IP에 여러 포트가 있습니다. 하나의 포트에 하나의 프로그램을 실행시킬 수 있습니다.



출처: 연합뉴스 <https://www.yonhapnews.co.kr/view/AKR201412310900000004>

- 하나의 주차장에 A1, A2, A3, 처럼 여러 차들이 각 섹션에 주차할 수 있죠? 주차장 주소를 IP, 포트를 각 섹션, 서버에 실행되고 있는 프로그램을 자동차로 비유할 수 있습니다.
- 하나의 서버(하나의 IP를 가짐)에 여러 포트에 각각 프로그램이 실행될 수 있습니다. 하나의 주차 자리에 하나의 자동차만 주차할 수 있는 것처럼, 하나의 포트에는 하나의 프로그램만 실행될 수 있습니다. 만약 하나의 포트에 여러가지 프로그램이 있다면, 어떤 프로그램에 요청을 했는지 알 수 없겠죠?
- 즉, 클라이언트는 서버의 주소(URL 또는 IP)를 통해 프로그램의 API에 요청(Request)하는 것입니다.

▼ 2) 6~8주차 수업 설명

- **6주차: 서버 구입해서 프로젝트 실행시켜보기**
 - 클라우드 서비스 AWS에서 사용할 서버(EC2) 구매하기
 - 구입한 서버에서 간단한 flask 서버 실행해보기
- **7주차: 클라우드 서버에 mongoDB 설치하기 + 나홀로 메모장 완성본을 서버에서 실행하기**
- **8주차: 도메인 붙이기**
 - 숫자로 된 IP 주소 대신 영문으로 된 주소 (www.spartacodingclub.co.kr 처럼요!)로 접속할 수 있게 됩니다.

▼ 3) EC2 서버 구매하기

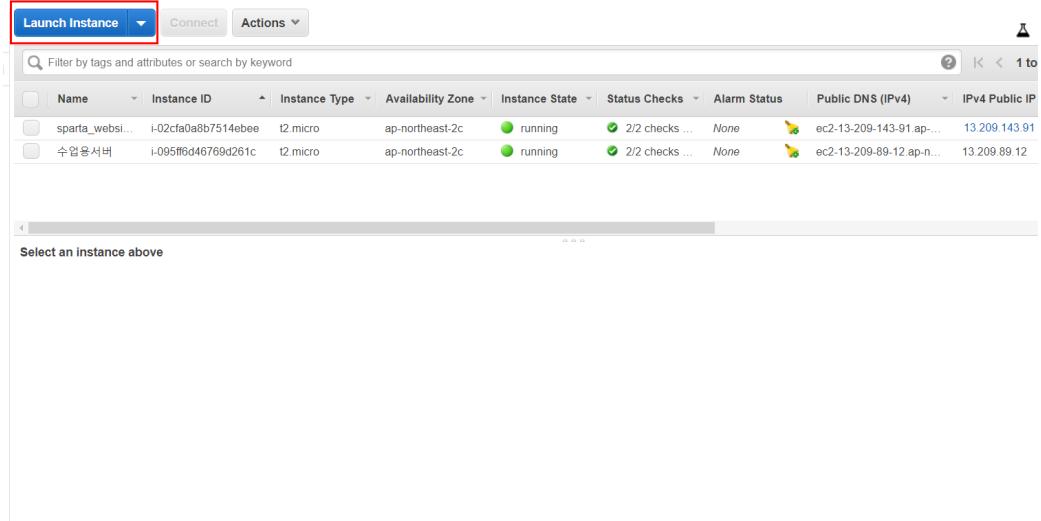
▼ AWS EC2 서버 사기 (준비 상식 편)

- 클릭 클릭 만으로 내가 원하는 서버(컴퓨터)의 설정을 정하고, 거기에 컴퓨터를 실행시키기 위한 운영체제(OS)를 설치할 거예요.
- 우리는 OS로 리눅스의 Ubuntu를 설치합니다. 윈도우와 같이, OS가 여러개 존재합니다. 리눅스도 그 중의 하나이며, 오픈소스로 발전되는 OS입니다.

▼ AWS EC2 서버 사기

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

▼ 구매 화면들 따라하기



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

Quick Start
My AMIs
AWS Marketplace
Community AMIs
<input checked="" type="checkbox"/> Free tier only ⓘ
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0d097db2fb6e0f05e
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
Select 64-bit (x86)
Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-0e1e385b0a934254a
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
Select 64-bit (x86)
Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0fd02cb7da42ee5e0
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 ENA Enabled: Yes
Select 64-bit (x86)

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 ⓘ
General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
General purpose	t2.micro ⓘ	1	1	EBS only	-	Low to Moderate	Yes
General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

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

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

- 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

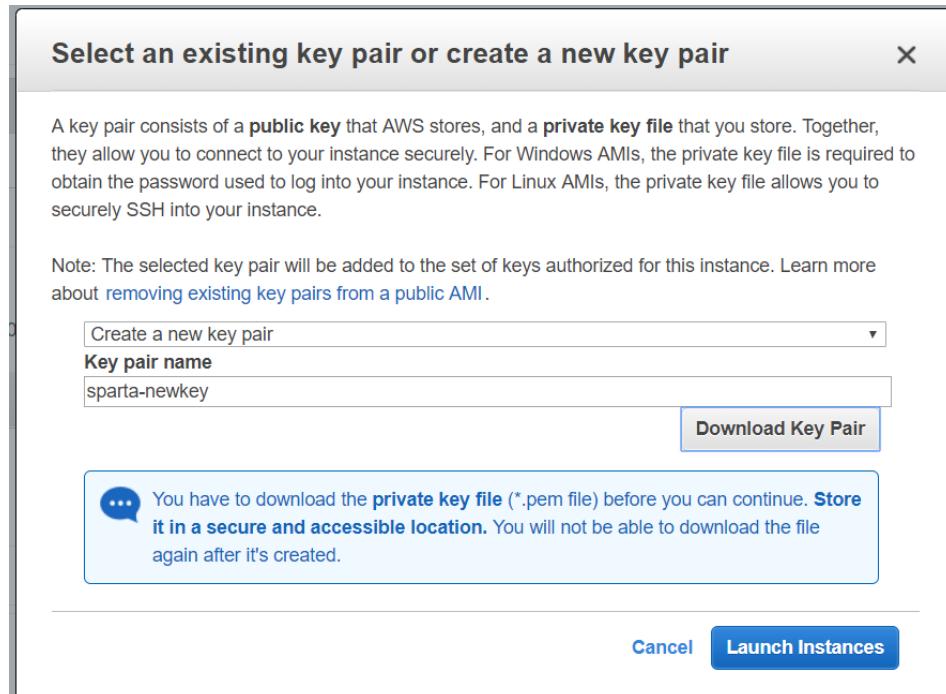
Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups

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

Buttons: Cancel, Previous, **Launch** (highlighted)

- 이 keypair는 절대 잃어버리면 안됩니다! 다시 다운로드 할 수 없습니다.

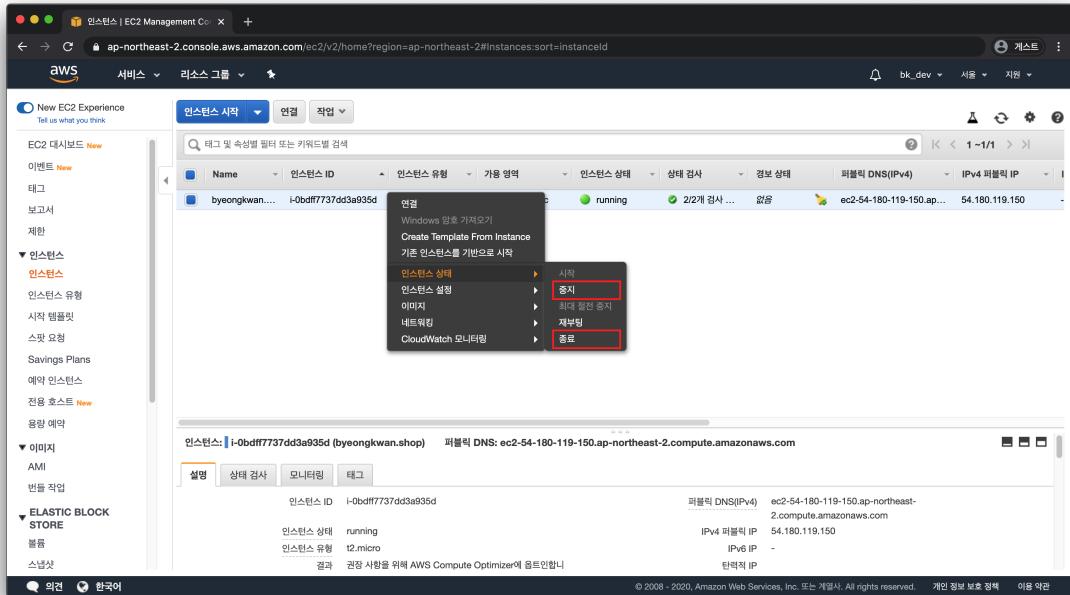


▼ 4) EC2 서버 종료하는 방법 (1년 후 자동결제 방지!)



중지 또는 종료하는 법. 무료 기간(1년) 후 결제가 되기 전에, 이렇게 종료하세요!

- 대상 인스턴스에 마우스 우클릭 > '인스턴스 상태'를 클릭합니다. 종료를 클릭하면, 내가 사용한 컴퓨터가 사라지는 것과 같습니다. 이 상태여야 추가 과금이 되지 않습니다. '중지'는 컴퓨터를 끄는 것과 같습니다. 사용량은 올라가지 않지만 인스턴스가 존재하는 것만으로 약간의 비용이 과금될 수 있습니다.



▼ 5) EC2에 접속하기

▼ AWS EC2에 접속하기 (준비 상식 편)

- SSH(Secure Shell Protocol)
 - 다른 컴퓨터에 접속할 때 쓰는 프로그램입니다. 다른 것들 보다 보안이 상대적으로 뛰어납니다.
 - 접속할 컴퓨터가 22번 포트가 열려있어야 접속 가능합니다. AWS EC2의 경우, 이미 22번 포트가 열려있습니다. 확인해볼까요?

Launch Instance ▾ Connect Actions ▾

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Sta
<input checked="" type="checkbox"/> i-0776319a3220cc87e	i-0776319a3220cc87e	t2.micro	ap-northeast-2c	running	2/2 checks ...	None
<input type="checkbox"/> mj	i-0ed332d8634bb8ff0	t2.micro	ap-northeast-2c	running	2/2 checks ...	None

Instance: i-0776319a3220cc87e Public DNS: ec2-13-125-250-20.ap-northeast-2.compute.amazonaws.com

Description Status Checks Monitoring Tags

Instance ID	i-0776319a3220cc87e	Public DNS (IPv4)	ec2-13-125-250-20.ap-northeast-2.compute.amazonaws.com		
Instance state	running	IPv4 Public IP	13.125.250.20		
Instance type	t2.micro	IPv6 IPs	-		
Elastic IPs		Private DNS	ip-13-125-250-20.ec2.internal		
Availability zone	ap-northeast-2c	Private IPs	172.31.1.10		
Security groups	launch-wizard-1, view inbound rules, view outbound rules	Secondary private IPs	-		
Scheduled events	No scheduled events	Security Groups associated with i-0776319a3220cc87e			
AMI ID	ubuntu-eks/k8s_18.04-amd64-server-20200727_02c9728b733d27	Ports	Protocol	Source	Destination
Platform	-	22	tcp	0.0.0.0/0	launch-wizard-1

▼ AWS EC2에 접속하기

▼ 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 생김새!

```
ssh -i /path/my-key-pair.pem ubuntu@13.125.250.20
```

▼ Window: ssh가 없으므로, git bash라는 프로그램을 이용!

- gitbash를 실행하고, 아래를 입력!

```
ssh -i 받은키페어를끌어다놓기 ubuntu@AWS에적힌내아이피
```

예) 아래와 비슷한 생김새!

```
ssh -i /path/my-key-pair.pem ubuntu@13.125.250.20
```

- Key fingerprint 관련 메시지가 나올 경우 Yes를 입력해주세요!
- git bash를 종료할 때는 exit 명령어를 입력하여 ssh 접속을 먼저 끊어주세요.

▼ 6) 간단한 리눅스 명령어 연습하기

- 우리는 클릭 클릭 마우스를 사용해서 윈도우즈 또는 맥을 사용할 수 있었어요. 그런데 우리가 쓰는 리눅스는 마우스 없이 명령어(쉘 명령어)를 통해 사용합니다.

[리눅스에서 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]: 강제로 지우기. 이 명령어로 지우면 복구가 안되니 조심하세요!  
sudo [실행 할 명령어]: 명령어를 관리자 권한으로 실행한다.  
sudo su: 관리가 권한으로 들어간다. (나올때는 exit으로 나옴)
```

• 사용 팁

- 윈도우즈에서 GitBash 를 사용해 명령어를 붙여넣기를 할 때는 마우스 오른쪽 버튼 - paste를 이용하면 됩니다.
- 위 아래 화살표를 누르면 전에 썼던 명령어가 자동완성됩니다.
- 파일/폴더명을 두 세 글자를 입력하고 tab키를 눌러보세요. 해당되는 파일명 또는 폴더명이 자동완성됩니다.
- **【✍튜터와 함께】 임의의 폴더를 만들어보고, 지워보세요!**

▼ 7) 파일질라(FileZilla)를 이용해서, 간단한 python 파일을 올려봅니다.

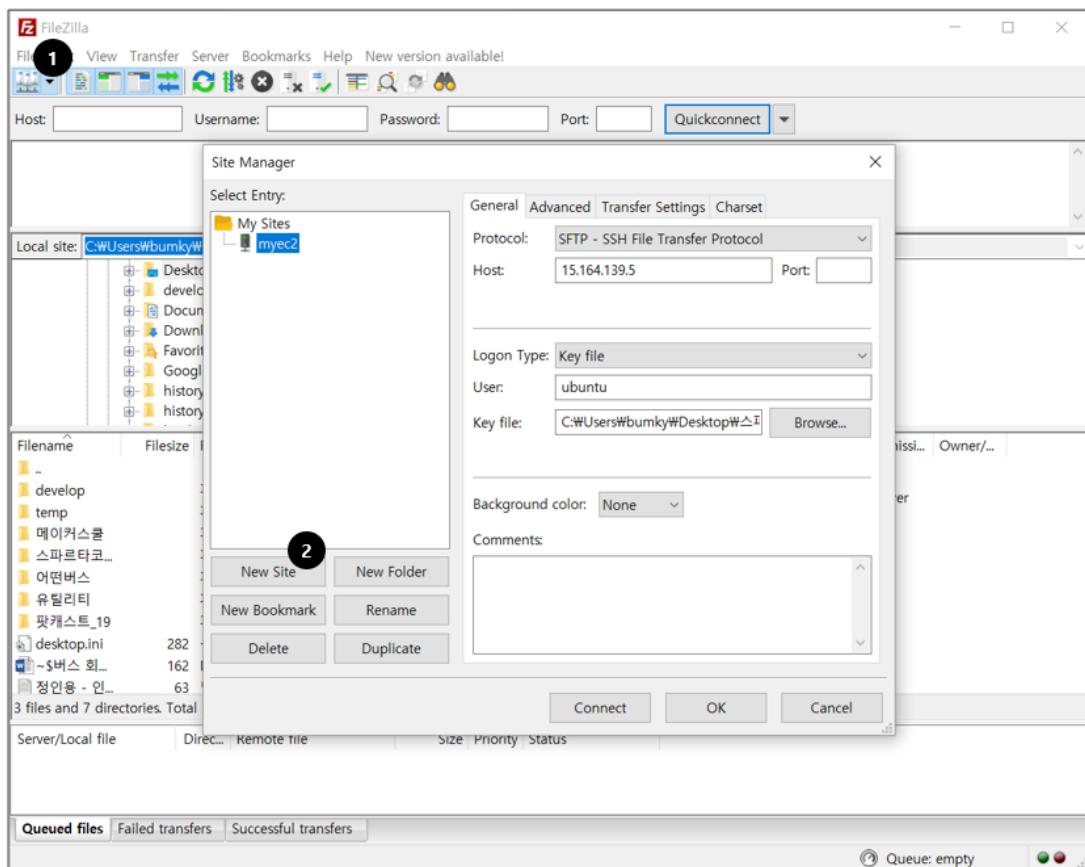
- 먼저 오늘 만드는 파일을 저장할 `week06` 폴더를 바탕화면의 `sparta` 폴더 안에 만듭니다.

- 서버에 업로드해 실행할 간단한 파일을 만듭니다. `hello.py`로 파일을 만든 후 `week06` 폴더 안에 아래 코드를 복사 붙여넣기 해주세요.

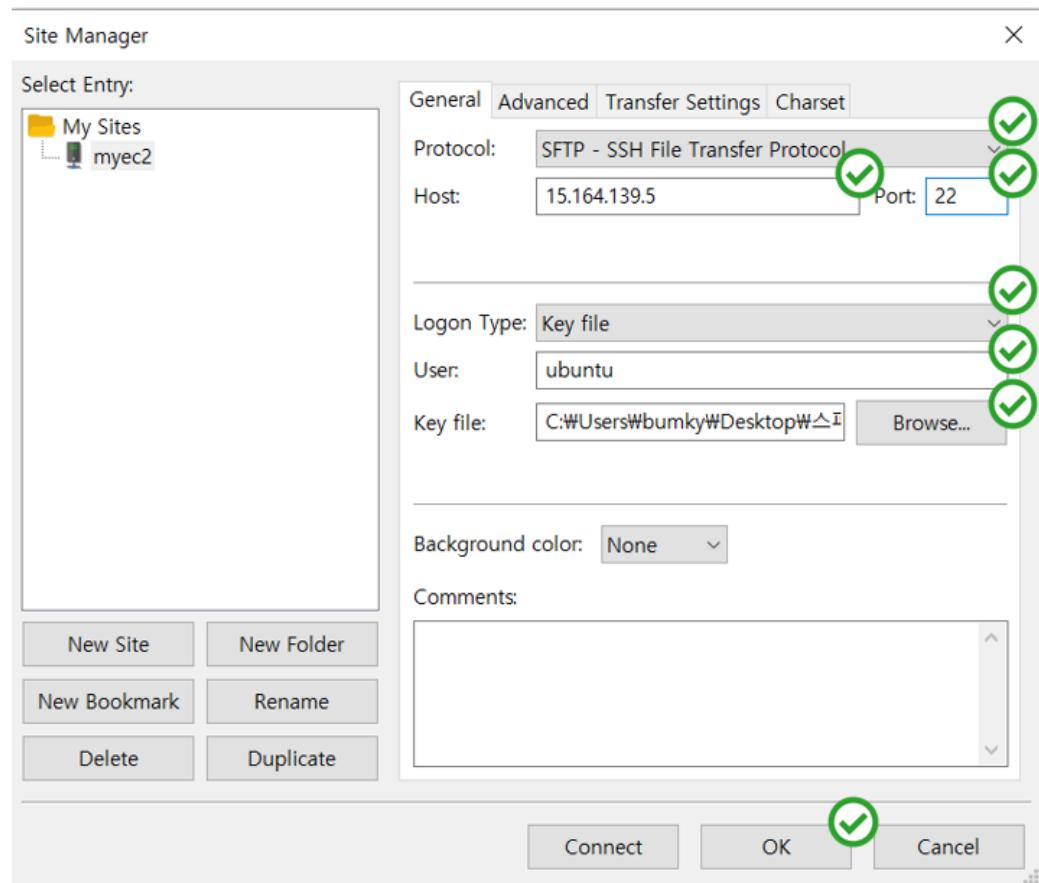
▼ [ 코드]

```
# 아주 간단하게, 이 정도만 적어볼까요?
print('Hello, Sparta!')
```

- 파일질라 실행, 다음과 같이 설정

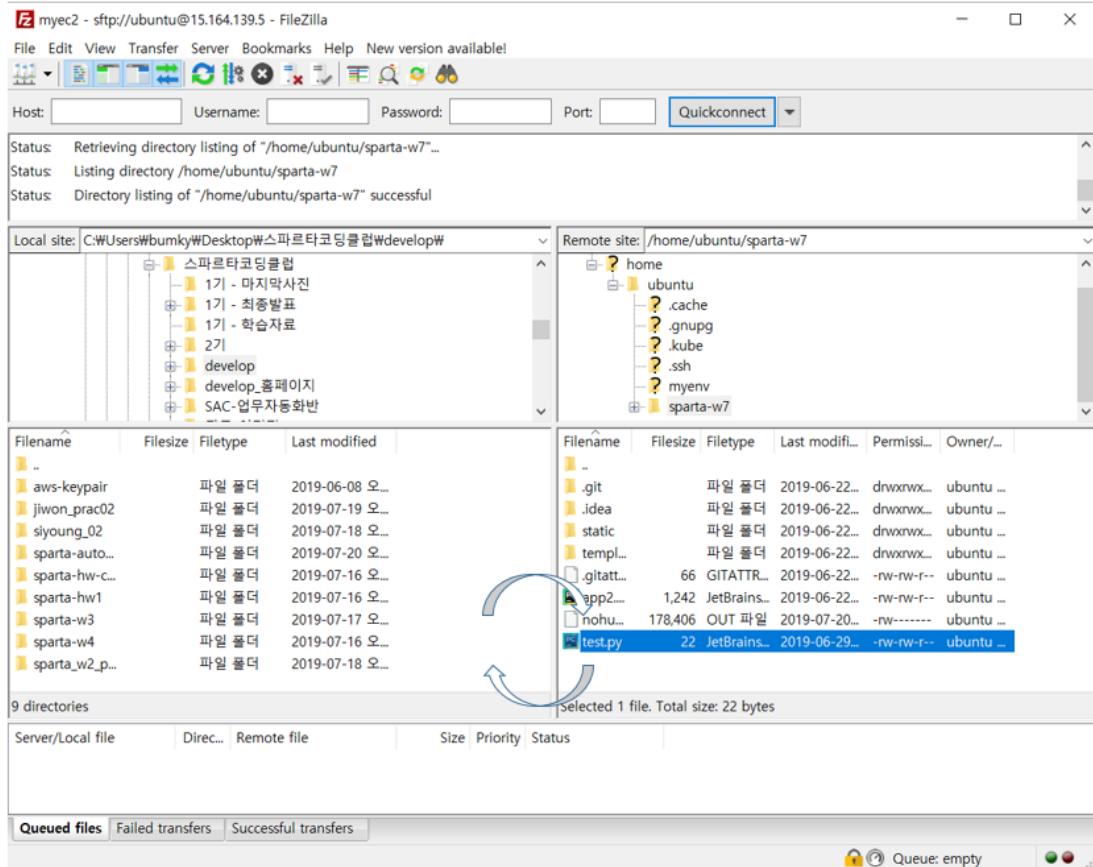


- 정보들을 입력하고, ok 누르면 서버의 파일들을 볼 수 있음
(Host: 내 EC2서버의 ip // User: ubuntu로 입력)
 - EC2 서버에 ubuntu라는 user로 접속하는 설정입니다. 이때 아까 3)에서 다운로드 받아둔 `sparta-newkey.pem`를 사용합니다.



3. 아까 만든 hello.py 파일을 EC2서버에 업로드합니다.

- 마우스로 드래그 해서 파일을 업로드/다운로드할 수 있습니다. 기본 설정은 왼쪽이 내 컴퓨터, 오른쪽이 원격접속한 EC2 인스턴스입니다.



▼ 8) 파일 실행해보기

- 아래 파일을 `week06` 폴더에 다운받습니다. 그런 후 파일질라에서 드래그 드롭으로 EC2 인스턴스의 `home/ubuntu` 폴더에 업로드합니다.

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/567f205d-a0f8-46a9-96b1-52ae5d9c2fa5/runpython.sh>

- 위 파일은 쉘 명령어를 적어둔 파일입니다. 파일을 실행시키면 안에 적어둔 명령어들이 실행됩니다. 위 파일에는 파이썬 파일을 실행하는 명령어들을 모아 두었습니다.
- EC2에 접속한 터미널(윈도우즈는 GitBash)에 아래와 같이 입력합니다. 쉘에서 `runpython.sh` 파일이 있는 `ubuntu` 폴더에 위치하고 있어야합니다.

```
. runpython.sh
```

▼ 파일 내용 보기

```

# 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 hello world!가 뜨는 것을 확인 할 수 있습니다.
python hello.py

```

- 실행결과는 아래처럼 터미널에 출력됩니다.



```

ubuntu@ip-10-0-10-14:~$ . runpython.sh
Hello Sparta
ubuntu@ip-10-0-10-14:~$ 

```

▼ 9) flask 서버를 실행해보기

- 기초적인 flask 서버를 실행해보겠습니다. week06 폴더 안에 [app.py](#) 로 아래 내용을복사 붙여넣기 해주세요. 그 후,파일질라에서 드래그 드롭으로 EC2 인스턴스의 home/ubuntu 폴더에 해당 파일을 업로드합니다.

▼ [💻 코드]

```

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)

```

- EC2 에 접속한 터미널(윈도우즈는 GitBash) 에 아래와 같이 입력합니다. 쉘에서는 [runpython.sh](#) 파일이 있는 ubuntu 폴더에 위치하고 있어야합니다.

```

# app.py 파일 파이썬으로 실행
python app.py

```



파일이 실행되지 않고, 에러가 발생합니다! 잠깐 에러메시지를 구글링 해보겠습니다. 에러가 나는 이유는 무엇일까요?



정답: flask 패키지가 없는데? - 라는 에러입니다.

그럼, 패키지를 설치해볼까요?

▼ 10) 리눅스에서 패키지 설치하기

- 우리가 여태까지 패키지 설치에 사용하던 pip(python install package) 는 EC2 서버에 자동으로 설치되어있지 않아서 추가로 설치해주어야 합니다. 우리는 파이썬 중에서도 python3를 쓰고 있습니다. python3를 위한 pip3을 설치하겠습니다.

1. 실행 파일 업로드

- 아래 파일을 `week06` 폴더에 다운받습니다. 그런 후 파일질라에서 드래그 드롭으로 EC2 인스턴스의 home/ubuntu 폴더에 업로드합니다.

```
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5b4c8228-7f3c-4eed-8b0b-e7c055060de9/installpip3.sh
```

▼ 파일 내용

```
# 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
```

- 위 파일은 pip3를 실행시키기 위한 쉘 명령어를 적어둔 파일입니다. 파일을 실행시키면 안에 적어둔 명령어들이 실행됩니다.

2. EC2 에 접속한 터미널(윈도우즈는 GitBash) 에 아래와 같이 입력합니다. 쉘에서 실행시킬 파일이 있는 ubuntu 폴더에 위치하고 있어야합니다.

```
. installpip3.sh
```

- 이제 패키지 설치를 하기 위한 준비가 끝났습니다. 프로그램을 설치하는 것처럼 이 과정은 한 번만 해주면 됩니다.

3. 설치한 pip를 이용해 flask 설치

- 앞으로 패키지 설치할 때에는 아래처럼 명령어를 작성하면 됩니다.

```
pip install flask
```

▼ 11) 다시 flask 서버를 실행해보기

- 아래 명령어로 flask 서버를 실행합니다.

```
python app.py
```

- 서버 실행이 되면, 크롬 브라우저 창에 아래와 같이 입력합니다.

```
http://[내 EC2 퍼블릭 IP]:5000/
```



아직, 작동하지 않을 것입니다! EC2에서 접속을 허용하기 위한 추가 설정이 필요합니다.

▼ 12) AWS에서 5000포트를 열어주기

- EC2 서버(=가상의 내 컴퓨터)에서 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

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various navigation options like Instances, Security Groups, and VPCs. The main area displays two instances:

- Instance 1:** Name: sparta_website_new, Instance ID: i-02cfa0a8b7514ebbe, Type: t2.micro, State: running, Public DNS: ec2-13-209-143-91.ap-northeast-2.compute.internal, Public IP: 13.209.143.91.
- Instance 2:** Name: sparta_website, Instance ID: i-0a0fc904b368d003a, Type: t2.micro, State: stopped, Public DNS: -.

Below the instances, detailed information is provided for the running instance, including its launch time (2020-04-04 17:17:40), AMI ID (ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20190722.1), and network details (VPC: vpc-39d41150, Subnet: subnet-1cf0f156, Network Interface: eth0).

- 해당 보안그룹을 클릭합니다.

The screenshot shows the AWS Security Groups page. The sidebar includes options like Instances, Security Groups, and VPCs. The main area shows a list of security groups:

Security group ID	Security group name	VPC ID	Description	Owner	Inbound rules count	Outbound rules count
sg-0ebf753ec1e1150d8	launch-wizard-7	vpc-5b3df430	launch-wizard-7 create...	781639735494	9 Permission entries	1 Permission entry

A red box highlights the row for the security group with ID sg-0ebf753ec1e1150d8. Below the table, there's a detailed view for this specific group.

- Edit inbound rules를 선택합니다.

aws 서비스 리소스 그룹

EC2 > Security Groups > sg-0ebf753ec1e1150d8 - launch-wizard-7

Details

Security group name launch-wizard-7	Security group ID sg-0ebf753ec1e1150d8	Description launch-wizard-7 created 2020-02-10T23:14:04.753+09:00	VPC ID vpc-5b3df430
Owner 781639735494	Inbound rules count 9 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules Outbound rules Tags

Inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	-
HTTP	TCP	80	:/0	-
SSH	TCP	22	0.0.0.0/0	-
Custom TCP	TCP	5000	0.0.0.0/0	-
Custom TCP	TCP	5000	:/0	-
Custom TCP	TCP	27017	0.0.0.0/0	-
Custom TCP	TCP	27017	:/0	-

클릭! Edit inbound rules

- 세 가지 포트를 추가해봅니다.

- 80포트: HTTP 접속을 위한 기본포트
- 5000포트: flask 기본포트
- 27017포트: 외부에서 MongoDB 접속을 하기위한 포트 (7주차에 쓰여요!)

aws 서비스 리소스 그룹

EC2 > Security Groups > sg-0ebf753ec1e1150d8 - launch-wizard-7 > Edit inbound rules

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	Anywhere	
SSH	TCP	22	Anywhere	
Custom TCP	TCP	5000	Anywhere	
Custom TCP	TCP	27017	Anywhere	

Type, Port range 양식에 맞게 입력 후 -> Source Anywhere 선택

Add rule

⚠ NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

클릭! Cancel Preview changes Save rules

▼ 13) 다시 접속해봅니다!

```
http://[내 EC2 퍼블릭 IP]:5000/
```

→ 잘 작동하는 것을 확인할 수 있습니다.

- 이제 app.py 파일을 수정하고, 서버에 업로드하면 외부에서 접속할 수 있는 서비스가 되겠죠?

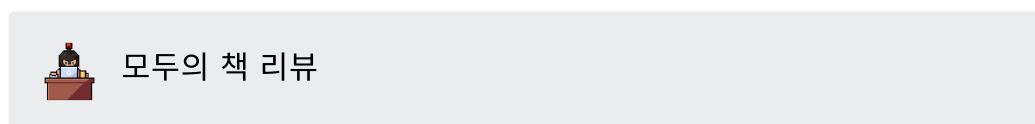
[1시간]: API 설계해보기

▼ 14) API 설계란?

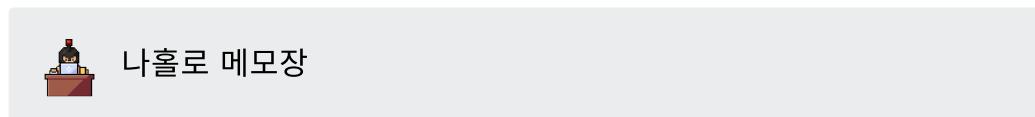
- API는 기능을 사용하기 위한 클라이언트와 서버 간의 약속입니다!
- 우리는 벌써 많은 API를 계획하고 만들었습니다.
- 수업시간에 만든 API 정보 문서 보러가기

👉 <http://spartacodingclub.shop/doc>

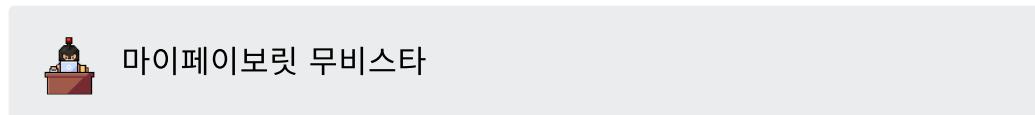
▼ API 문서 화면



기능	요청			응답
	요청 URL	요청 방식(GET/ POST)	요청 데이터	응답 데이터
리뷰 작성(Create)	/review	POST	제목(title), 저자(author), 리뷰(review)	(JSON 형식) 'result'= 'success', 'msg'= '리뷰가 성공적으로 작성되었습니다.'
리뷰 불러오기(Read)	/review	GET	없음	(JSON 형식) 'result'= 'success', 'reviews'= 리뷰리스트



기능	요청			응답
	요청 URL	요청 방식(GET/ POST)	요청 데이터	응답 데이터
메모 작성(Create)	/memo	POST	URL(url_give), 코멘트(comment_give)	(JSON 형식) 'result'= 'success'
메모 조회(Read)	/memo	GET	없음	(JSON 형식) 'result'= 'success', 'articles': 아티클 정보



마이페이지보릿무비스타

기능	요청			응답
	요청 URL	요청 방식(GET/ POST)	요청 데이터	응답 데이터
영화인 정보 조회(Read)	/api/list	GET	없음	(JSON 형식) 'result'= 'success', 'stars_list'= 영화인 정보 리스트
영화인 정보 좋아요 수 업데이트 (Update)	/api/like	POST	영화인 이름 (name_give)	(JSON 형식) 'result'= 'success'
영화인 정보 삭제(Delete)	/api/delete	POST	영화인 이름 (name_give)	(JSON 형식) 'result'= 'success'

▼ 15) 내 프로젝트 API 설계해보기



수업시간 프로젝트에서 API를 만들기 전 설계를 먼저 했었죠?

지금부터 API 설계를 시작하고 궁금한 점이 있으면 언제든 튜터에게 질문합니다.

아래 박스를 채워주세요!

기능	요청 정보			응답 정보
	요청 URL	요청 방식 (GET/ POST)	요청 데이터	
(예시) 리뷰 작성(Create)	(예시) /review	(예시) POST	(예시) 제목(title), 저자(author), 리뷰(review)	(예시) (JSON 형식) 'result'= 'success', 'msg'= '리뷰가 성공적으로 작성되었습니다.'

- 여기 API 문서 양식을 만들어두었습니다. 꼭 엑셀파일에 작성하지 않아도 됩니다! 항목들이 빠지지 않게 자유롭게 작성하시면 됩니다.

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/63b2815c-71d4-4532-9ba8-da289579a695/API_.xlsx

후반 3시간

[월수/화목반] : 체크인 & 출석체크



튜터님은 체크인과 함께 출석 체크(링크)를 진행해주세요!

스파르타코딩클럽 출석체크

<http://spartacodingclub.shop/attendance>

▼ "15초 체크인" 진행합니다

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



배우고 적용하기 - 후반부

[0.5시간] 최종 프로젝트 기획안 발표하기

▼ 16) 내 프로젝트 동료들에게 소개하기

- 3분씩 돌아가면서 지난주 숙제인 프로젝트 기획안을 발표합니다.
- 슬랙에 공유했던 블로그를 토대로 발표합니다. 지난주 숙제를 못했던 분들은 아래 내용을 중심으로 발표해주세요.



스파르타코딩클럽 내 프로젝트 기획안

1. 프로젝트 이름/설명
2. 프로젝트 생김새(레이아웃)
3. 개발해야 하는 기능 (우선순위별 정렬)
4. 프로젝트에 필요한 데이터 소스 (3주차 공통 숙제 참고)
4. 다음 시간까지 해야 하는 Todo 리스트

[0.5시간] 프로젝트 Github repository 만들기

▼ 17) 프로젝트 Github 설정하기

- 앞으로 sparta 폴더 안 my_projects 폴더에 프로젝트를 진행하겠습니다.
- my_projects 안에 아래 `README.md` 파일을 넣어주세요.

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/d1d24ce2-db05-4978-a675-666d02fdb90c/README.md>

- 프로젝트를 소개하는 파일을 관행적으로 `README.md` 라고 하며, 프로젝트 최상위 폴더 안에 넣어둡니다.
- 파일을 편집할 때는 일반 텍스트 에디터(메모장 등)를 사용하면 됩니다. markdown이라는 형식을 따르는 일반 텍스트 파일입니다.
- markdown이라는 형식을 따르면 굵은 글씨 등의 서식을 쉽게 적용할 수 있습니다. (참고. [markdown 튜토리얼](#))

2. my_projects 안에 아래 `.gitignore` 파일 넣기

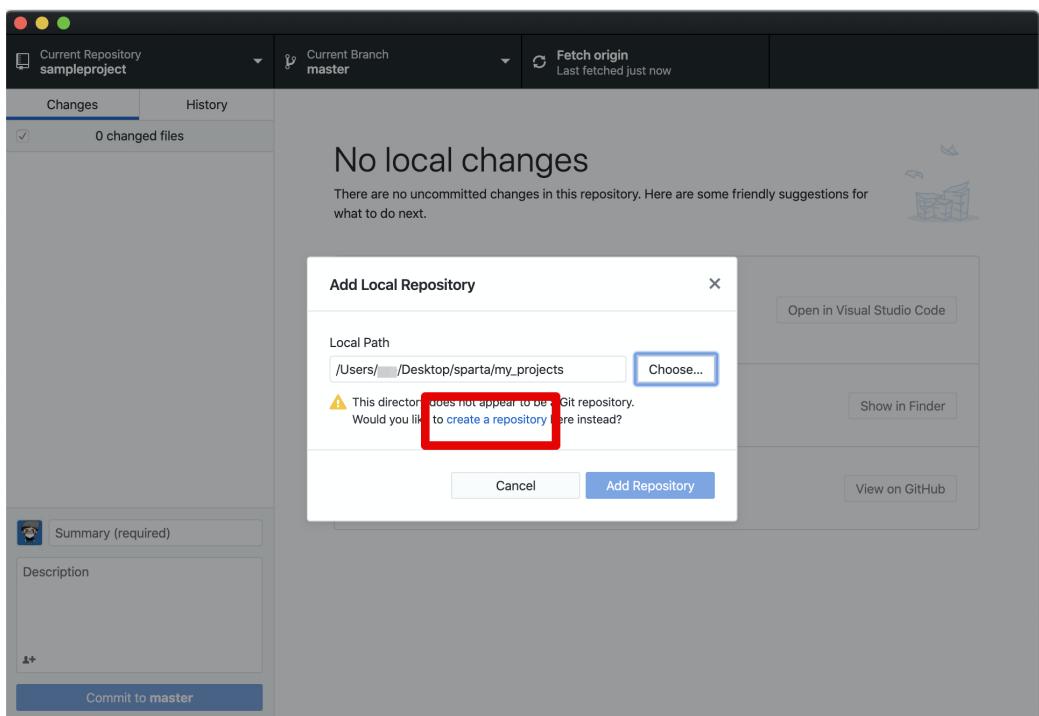
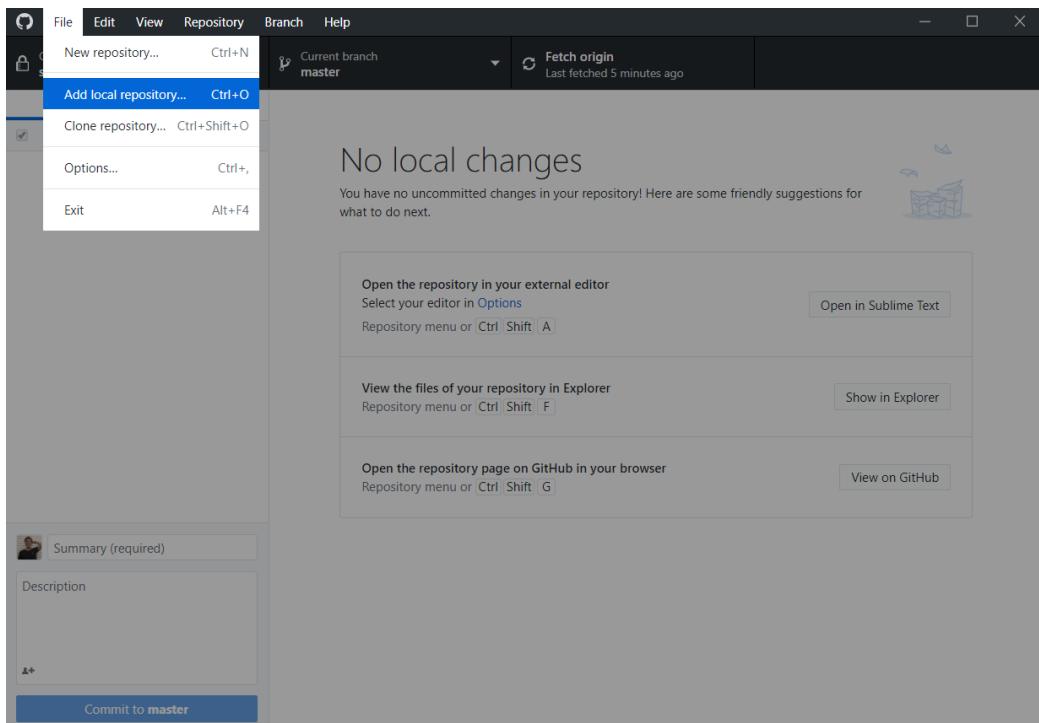
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/99a2eb42-8044-4c23-98f7-c86e6424c73f/_gitignore

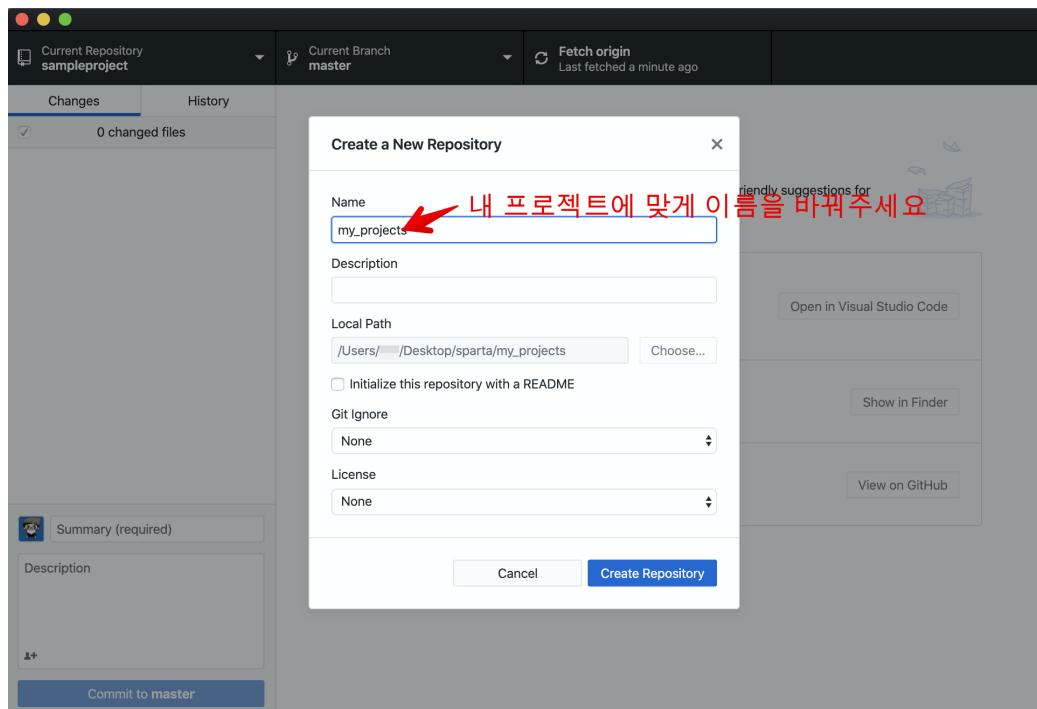
- `.gitignore`에 적힌 폴더와 파일은 Git이 추적(tracking)하지 않습니다. 내 컴퓨터에 파일은 존재하지만 Git은 없는 것처럼 취급하기 때문에, github에도 업로드가 되지 않습니다.
- 보통 컴퓨터마다 다르기 때문에 다른 사람들이 필요없는 불필요한 설정파일(예. venv 폴더), 보안을 위해 공개되면 안되는 파일 목록을 여기에 적어둡니다.

3. 내 로컬 저장소(Repository) 만들기

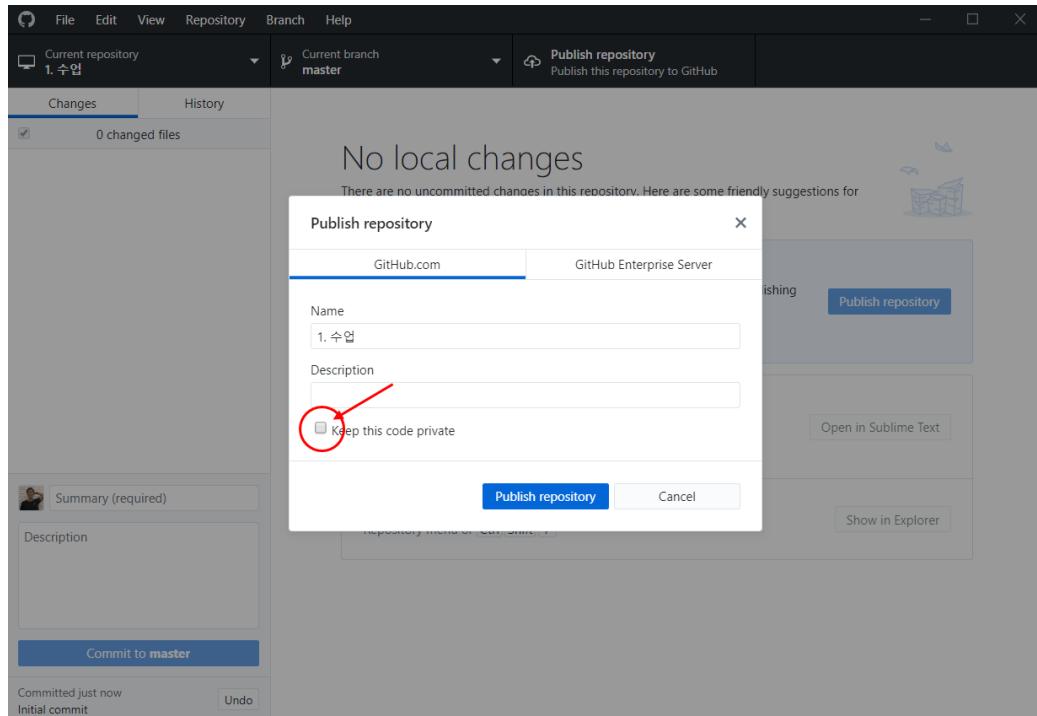
- Github Desktop 을 켜고 로그인 후, add local repository 클릭, 바탕화면에 있는 sparta/my_projects 폴더를 클릭하고, 파란색 글자/버튼을 따라 누르면 repository가 완성됩니다.
- [리마인드] 기존에 존재하던 폴더(my_projects)를 git이 추적(tracking)할 수 있게 설정했다고 생각하면 됩니다.

▼ 화면

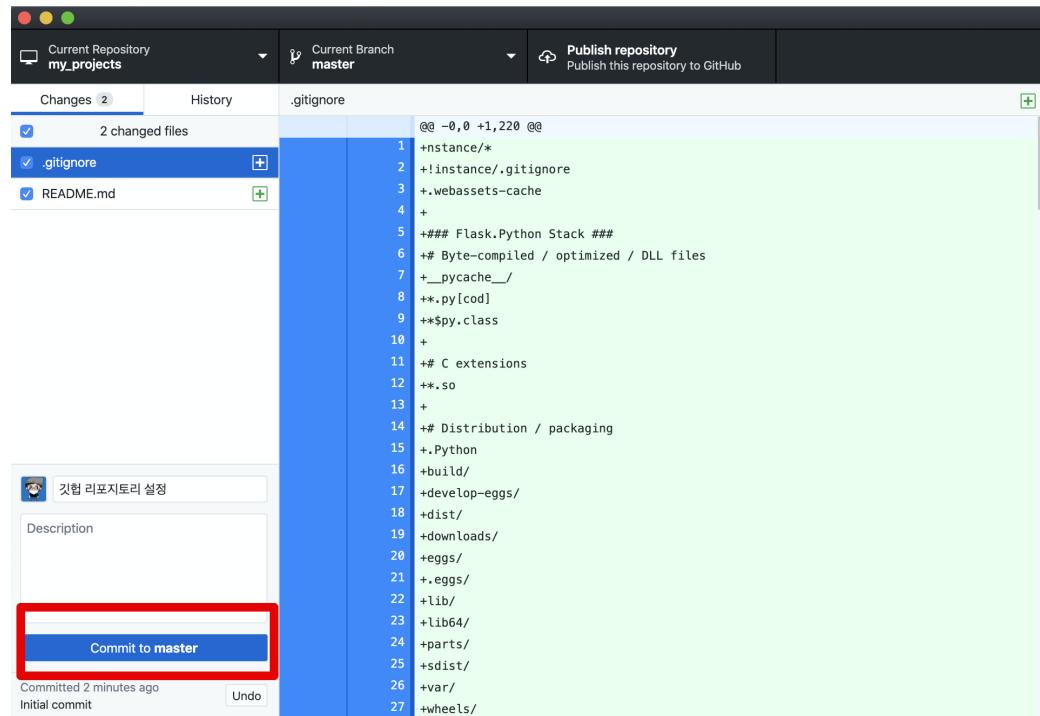




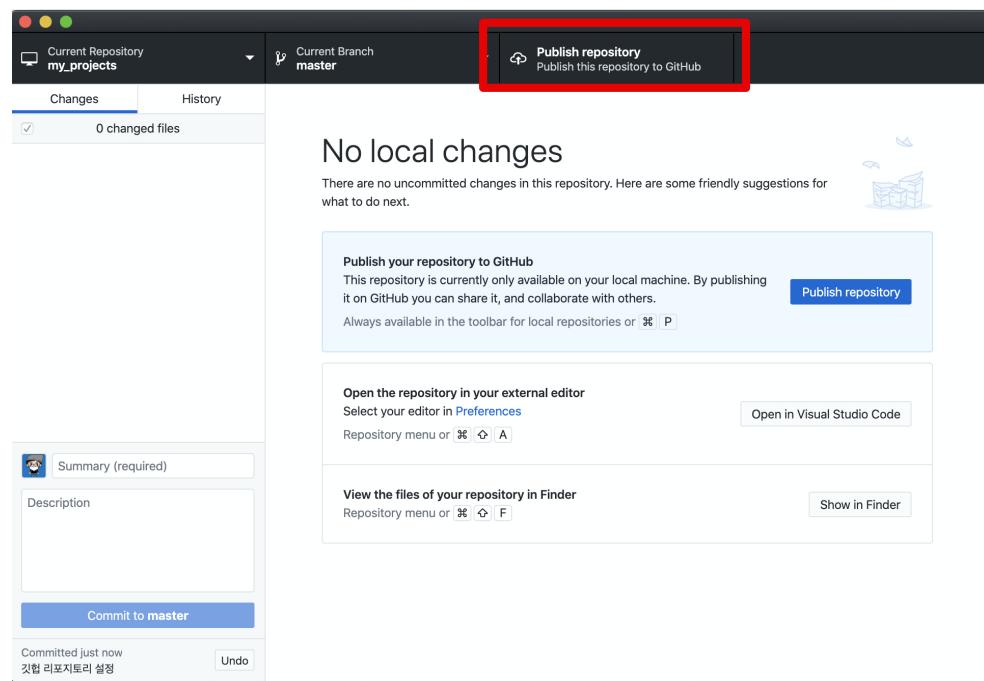
반드시 'keep this code private'을 해제해주세요!
공개되어 있어야 튜터도 프로젝트 확인이 가능하답니다.



- `.gitignore` 와 `README.md` 파일을 commit 합니다.



- Github에 Publish(초기 공개) 합니다



4. github.com에 로그인 후, 원격 저장소(repository)가 생성된 것 확인

▼ 화면

The screenshot shows a GitHub repository page for 'ohahohah / scc-homework'. The repository has 1 branch and 0 tags. It contains files like .gitignore, LICENSE, README.md, and homework_sample.html. The README.md file shows the text 'scc-homework' and '스파르타코딩클럽 숙제 리포지토리입니다.' The right sidebar includes sections for About, Releases, and Packages.

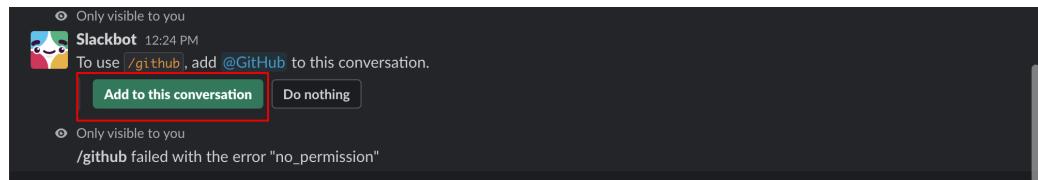
▼ 18) 슬랙 채널에 알람 설정

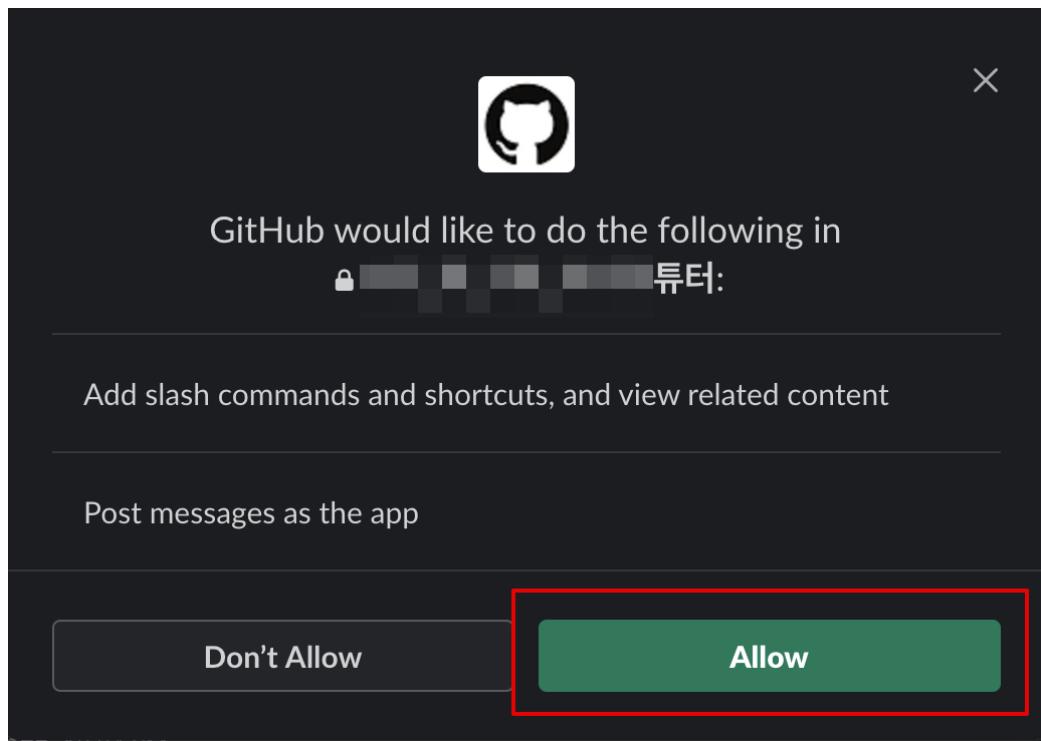
- 우리반 슬랙 채널에 나의 원격 저장소(Github repository) 수정내역을 알림오게 설정해볼게요.
- 슬랙에 있는 github 봇을 사용하는 겁니다. 슬랙 채널 메시지창에 아래처럼 입력해보세요.

```
/github subscribe 내_github_사용자명/repositories명
```

- 만약 아래와 같은 메시지가 나온다면 초록 버튼 클릭 클릭!

▼ 화면





[1시간] 튜터 피드백 - Todo 리스트

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

- 각자 코딩을 시작 합니다. 튜터님이 5분씩 개별 면담을 진행하여 Todo 리스트를 함께 정합니다.
 - 예 1. "일단 HTML, CSS부터 뾰개고 갑시다! 다음주까지 프론트엔드 완성해주세요!"
 - 예 2. "조회하려면 데이터가 있어야 하니, 다음 주까지 데이터를 저장 API를 완성해보시죠!"
 - 예 3. "이 프로젝트는 웹 스크래핑이 무척 중요하니 다음주까지는 완성해주세요. 그 다음 바로 API 완성으로 넘어가시죠!"

[남은 시간] 프로젝트 개발

▼ 20) Todo 리스트에 맞게 개발을 합니다.

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

[숙제 - 단짠단짠 🍕🍜] 개발일지 쓰기!

- 튜터님과 함께 정한 Todo 리스트 순서대로 개발합니다. 즐겁게! 💡💻

- 개발하면서 익힌 내용들을 개발일지에 자유롭게 기록합니다.



개발일지는, 본인을 위해 과정을 기록하는 것입니다. 잘 안되는 것, 개발하면서 배운 것들을 편하게 작성해보세요. 이렇게 능동적으로 기록하면 배운 것들이 정리되면서 학습효율이 더 높아진다고 해요.

여러분이 해낸 것들 을 꼭 기록해두세요!

- 개발일지에는 아래와 같은 내용이 들어가면 좋아요!



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

▼ 참고. 이전 기수 개발일지

- 매 번 개발할 때마다 개발일지를 쓰고,
<https://travel-hama.tistory.com/3?category=874450>
<https://medium.com/@annasim0318/tile-%ED%84%B0%EC%A0%80%EC%8A%A4%EB%8D%BC%EB%8B%8C%EB%85%84-438e6e40e846>
<https://medium.com/@sujinhan.me/habit-tracker-2020-04-12-04-15-86d34872e274>
- TIL 이 모이면 나중엔 종합 개발일지가 되죠!
<https://medium.com/@sujinhan.me/habit-tracker-%ED%84%B0%EC%A0%80%EC%8A%A4%EB%8D%BC%EB%8B%8C%EB%85%84-8%EC%9E%91%EC%9D%BC-7382ff1f004d>
<https://blog.naver.com/dalgona92/222043274296>
- 이렇게 개발일지 쓰는 게 습관이 되면 또 새로운 프로젝트를 시도하기 쉬워질 수도 있고요!
https://blog.naver.com/yonnie_k/222037366116
- 꼭 일주일에 한 편씩만 작성하지 않아도 되며, 사실은 개발을 할 때마다 매일 작성하는 것이 가장 좋은 방법입니다.
- 지난주 🎂 5분 꿀팁 🎂에서도 봤었죠? TIL (Today I Learned)를 검색하면, 많은 분들의 개발일지를 볼 수 있습니다.

[숙제 제출]

- 슬랙에 개발일지 링크를 공유합니다.

체크아웃

▼ "15초 체크아웃"을 진행합니다.

- 튜터는 타이머를 띄워주세요! ([링크](#)).
- 체크인처럼, 현재 본인의 감정상태와 수업후기에 관해 이야기합니다.
 - 예. 오늘 내 프로젝트를 시작해서 재밌었어요!
 - 자유롭게 해주셔도 좋고, **KPT**에 맞춰해주셔도 좋아요.
 - Keep : 오늘 수업을 하면서 좋았던 것, 앞으로도 할 행동 / Problem : 아쉬워서 고쳐보면 좋을 것 / Try : 고치기 위해 내가 할 시도
 - 예) (Keep) 웹사이트를 빠르게 만들어서 좋았어요. (Problem) 아직 API 사용하는 부분이 헷갈려요. (Try) 집에 가서 숙제하면서 복습해볼게요!
- **튜터님은 체크아웃과 함께 출석 체크([링크](#))를 진행해주세요!**
 - 출석체크 - (변동이 있다면 다시 제출해주세요!)

스파르타코딩클럽 출석체크

<http://spartacodingclub.shop/attendance>

[설치] - 다음 시간을 위해 미리 설치해와야 할 것들

- 없음!