

9/21/2019

분석환경 세팅과 GIT

5기 최보경

1

CONTENTS

THEME A. 분석환경 세팅

- Anaconda / Jupyter Notebook / AWS

THEME B. Git & GitHub

- 버전관리 / Git / GitHub

THEME A. 분석환경 세팅

1. Anaconda

기본명령 / 패키지 관리 / 가상환경 생성

2. Jupyter Notebook

IDE 개념 / 가상환경 커널 오픈

3. AWS

제공 서비스 소개/ EC2

1. Anaconda?

- Anaconda를 사용하는 이유

- Python 기반 데이터 분석에 필요한 주요 패키지 포함
- 주요 **IDE** 포함 (ex. Jupyter notebook)
- 가상환경 및 패키지 관리자 제공
- Tensorflow의 문서가 conda 기준으로 작성됨



1. Anaconda – 기본 명령 [참고]

- 아나콘다 버전 확인 (conda --version)

```
Anaconda Prompt
(base) C:\Users\reno\>conda --version
conda 4.6.8
```

- 아나콘다 업데이트 (conda update conda)

```
(base) C:\Users\reno\>conda update conda
Collecting package metadata: done
Solving environment: done

# All requested packages already installed.
```

1. Anaconda – 패키지 관리

- 설치된 패키지 목록 확인 (conda list)

```
(base) C:\Users\wrenoi>conda list
# packages in environment at C:\Anaconda3:
#
# Name                        Version      Build    Channel
ipyw_jlab_nb_ext_conf        0.1.0        py37_0
alabaster                     0.7.11       py37_0
anaconda                      5.3.1        py37_0
anaconda-client               1.7.2        py37_0
anaconda-navigator            1.9.2        py37_0
anaconda-project              0.8.2        py37_0
appdirs                       1.4.3        py37h28b3542_0
asn1crypto                    0.24.0       py37_0
```

- 패키지 설치 (conda install 패키지이름)

```
(base) C:\Users\BokyoungChoi>conda install seaborn
Collecting package metadata: done
Solving environment: ⚡
```

(참고) conda 기본 서버에 존재하지 않는 패키지들은 pip install 패키지이름으로 설치
명령을 강제 종료하고 싶으면 Ctrl+C

1. Anaconda – 가상환경

• 가상환경 (개발환경, 독립 개발환경)

개념

자신이 원하는 Python 환경을 구축하기 위해, 필요한 모듈 (패키지)만 담아 놓는 바구니

목적

각 환경에서 동일한 패키지 버전과 의존성을 유지

상황

각 모듈은 다른 모듈에 의존성이 달라서, 마구잡이로 설치하면 이유 모를 충돌 발생 가능! 따라서, 같은 모듈을 사용해도 다른 버전이 필요하거나 하는 상황이 발생함

1. Anaconda – 가상환경

- 새로운 가상환경 생성하기 (conda create -n 가상환경이름 python=원하는 버전)

```
(base) C:\Users\reno>conda create -n growthhackers python=3.7.0  
Collecting package metadata: done  
Solving environment: done
```

```
Proceed ([y]/n)? y
```

(참고) 파이썬 버전 확인 방법

- Windows: 명령 프롬프트에 들어가서 python --version, anaconda 프롬프트에서도 python --version 으로 확인 가능
- Mac: 터미널에 들어가서 python --version

C:\> 명령 프롬프트

```
Microsoft Windows [Version 10.0.17134.648]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Users\reno>python --version  
Python 3.7.0
```


1. Anaconda – 가상환경

- 가상환경 목록 확인 (conda info --envs)

```
(base) C:\Users\renoi>conda info --envs
# conda environments:
#
base                * C:\Anaconda3
growthhackers        C:\Anaconda3\envs\growthhackers
```

1. Anaconda – 가상환경

- 가상환경 활성화 (activate 가상환경명 or conda activate 가상환경명)

```
(base) C:\Users\reno>activate growthhackers  
(growthhackers) C:\Users\reno>
```

- 가상환경에 설치된 패키지 목록 확인해보기 (conda list), 아무것도 설치하지 않으면 이 패키지들 보유

```
(growthhackers) C:\Users\reno>conda list  
# packages in environment at C:\Anaconda3\envs\growthhackers:  
#  
# Name                        Version      Build    Channel  
certifi                       2019.3.9     py37_0  
pip                           19.0.3       py37_0  
python                        3.7.0        hea74fb7_0  
setuptools                    40.8.0       py37_0  
vc                             14.1         h0510ff6_4  
vs2015_runtime                14.15.26706  h3a45250_0  
wheel                         0.33.1       py37_0  
wincertstore                   0.2          py37_0
```

1. Anaconda – 가상환경

- 가상환경 비활성화 (conda deactivate)

```
(growthhackers) C:\Users\reno i>conda deactivate  
(base) C:\Users\reno i>
```

- 가상환경 삭제(Conda remove --name 지울가상환경명 --all)

```
(base) C:\Users\BokyoungChoi>conda remove --name growthhackers --all  
  
Remove all packages in environment C:\Users\BokyoungChoi\conda\envs\growthhackers:  
  
## Package Plan ##  
  
environment location: C:\Users\BokyoungChoi\conda\envs\growthhackers
```

Proceed ([y]/n)? 하면 y

2. Jupyter Notebook - IDE란?

- 파이썬을 위한 통합개발환경 (Integrated Development Environment)
- 종류:



PyCharm



Spyder



Jupyter
notebook

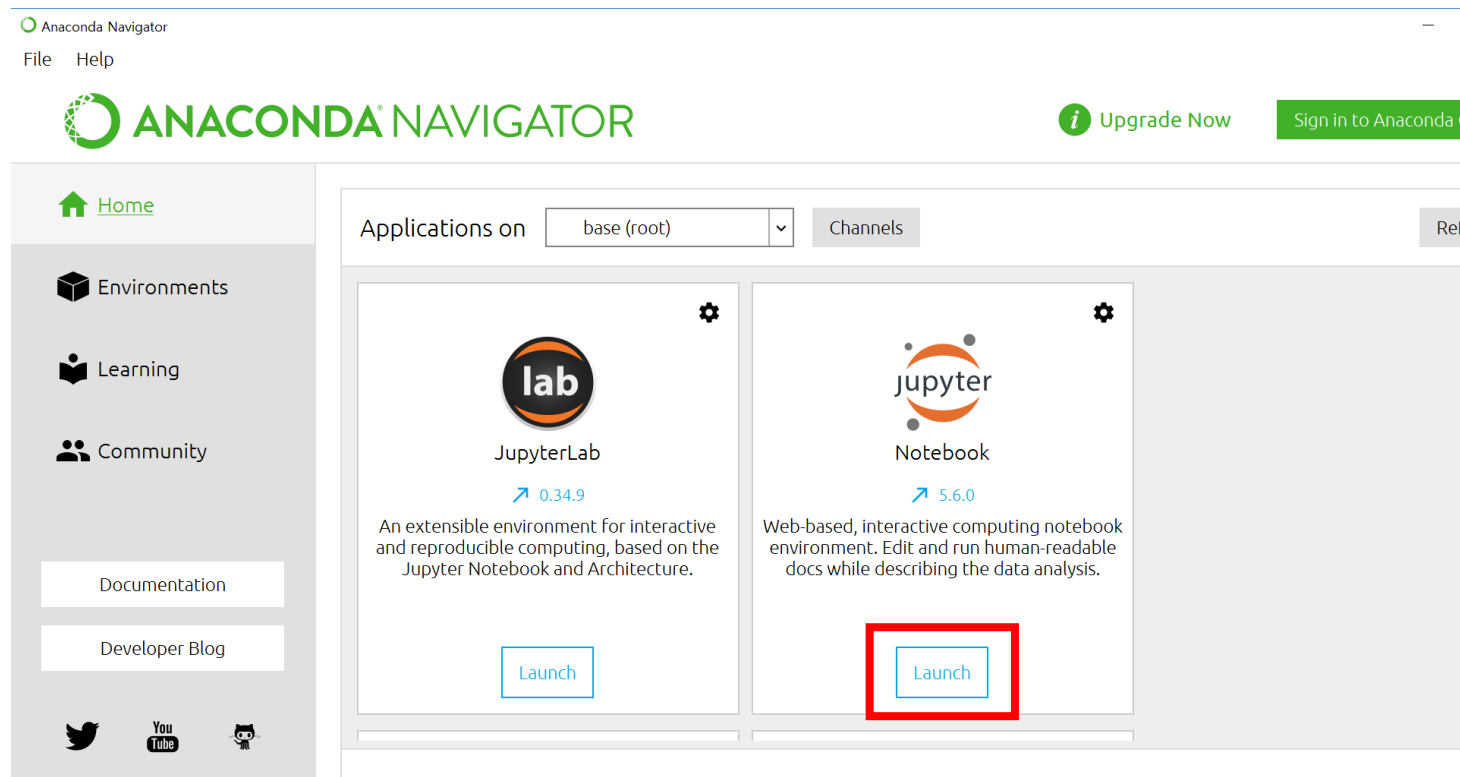
2. Jupyter Notebook?

- Html이나 pdf로 변환해서 공유하기 쉬움
- 셀 단위의 순차적인 실행 가능
- Github과 연동 가능



2. Jupyter Notebook – 실행 방법 1

1-1. Anaconda Navigator > 'Jupyter notebook'



2. Jupyter Notebook – 실행 방법 2

1-2. Anaconda Prompt > 'jupyter notebook' 입력

```
Anaconda Prompt - jupyter notebook
(base) C:\Users\poo0> jupyter notebook
[I 21:02:58.496 NotebookApp] JupyterLab beta preview extension loaded from C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab
[I 21:02:58.496 NotebookApp] JupyterLab application directory is C:\ProgramData\Anaconda3\share\jupyter\lab
[W 21:02:58.539 NotebookApp] Error loading server extension jupyterlab
Traceback (most recent call last):
  File "C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab\commands.py", line 321, in __init__
    self._run(['node', 'node-version-check.js'], cwd=HERE, quiet=True)
  File "C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab\commands.py", line 1165, in _run
    proc = Process(cmd, **kwargs)
  File "C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab\process.py", line 73, in __init__
    self.proc = self._create_process(cwd=cwd, env=env)
  File "C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab\process.py", line 131, in _create_process
    cmd[0] = which(cmd[0], kwargs.get('env'))
  File "C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab\jupyterlab.py", line 59, in which
    raise ValueError(msg)
ValueError: Please install nodejs 5+ and npm before continuing installation. nodejs may be installed using conda or directly from the nodejs website.

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "C:\ProgramData\Anaconda3\lib\site-packages\notebook\notebookapp.py", line 1454, in init_server_extensions
    func(self)
  File "C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab\extension.py", line 111, in load_jupyter_server_extension
    info = get_app_info(app_dir)
  File "C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab\commands.py", line 244, in get_app_info
    handler = _AppHandler(app_dir, logger)
  File "C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab\commands.py", line 324, in __init__
    raise ValueError(msg)
ValueError: Please install nodejs 5+ and npm before continuing installation. nodejs may be installed using conda or directly from the nodejs website.
[I 21:02:59.324 NotebookApp] Serving notebooks from local directory: C:\Users\poo0\
[I 21:02:59.325 NotebookApp] 0 active kernels
[I 21:02:59.326 NotebookApp] The Jupyter Notebook is running at:
[I 21:02:59.329 NotebookApp] http://localhost:8888/?token=8cbe525b622230149bba5eea9c43c48a2b7779154a0ddbc
[I 21:02:59.330 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 21:02:59.333 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=8cbe525b622230149bba5eea9c43c48a2b7779154a0ddbc
[I 21:03:00.018 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

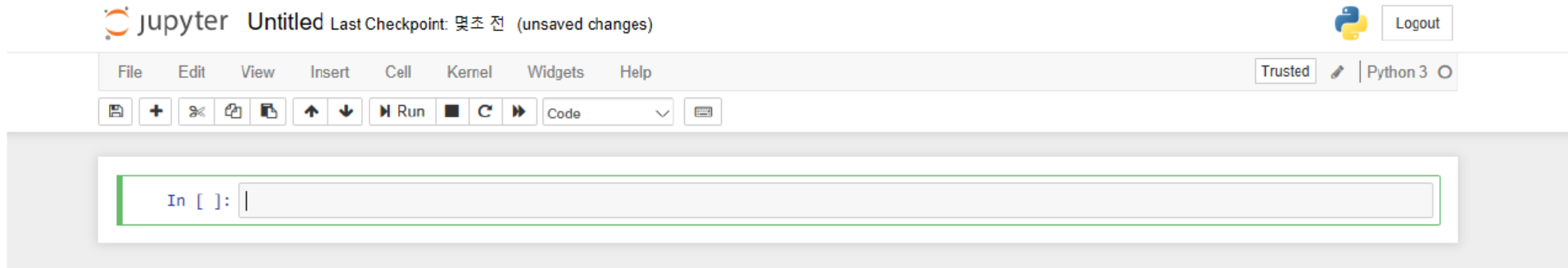
2. Jupyter Notebook – 창이 뜨면

2. 오른쪽 상단의 New > Python3



2. Jupyter Notebook – 실행 완료

3. Jupyter Notebook 실행 완료



2. Jupyter Notebook – 가상환경 커널 오픈

- 현재 가상환경 리스트들 확인: `Conda -info envs` (생략)
- 아까 만든 growthhackers 가상환경을 활성화해보자 **Conda activate 가상환경명**

```
(base) C:\Windows\system32>conda activate growthhackers
```

Conda install jupyter notebook
Conda install ipykernel

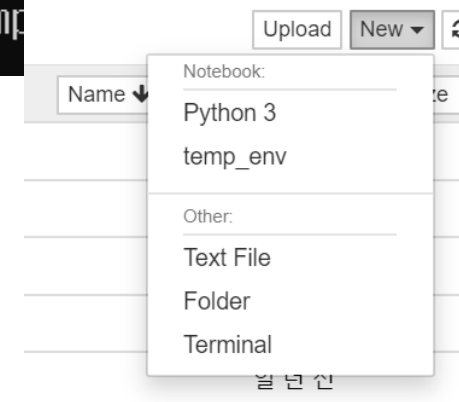
```
(growthhackers) C:\Windows\system32>conda install ipykernel  
Collecting package metadata: /
```

python -m ipykernel install --user --name= 새로 만들고 싶은 노트북 커널 이름

```
(growthhackers) C:\Windows\system32>python -m ipykernel install --user --name=temp_env  
Installed kernelspec temp_env in C:\Users\korse\AppData\Roaming\jupyter\kernels\temp
```

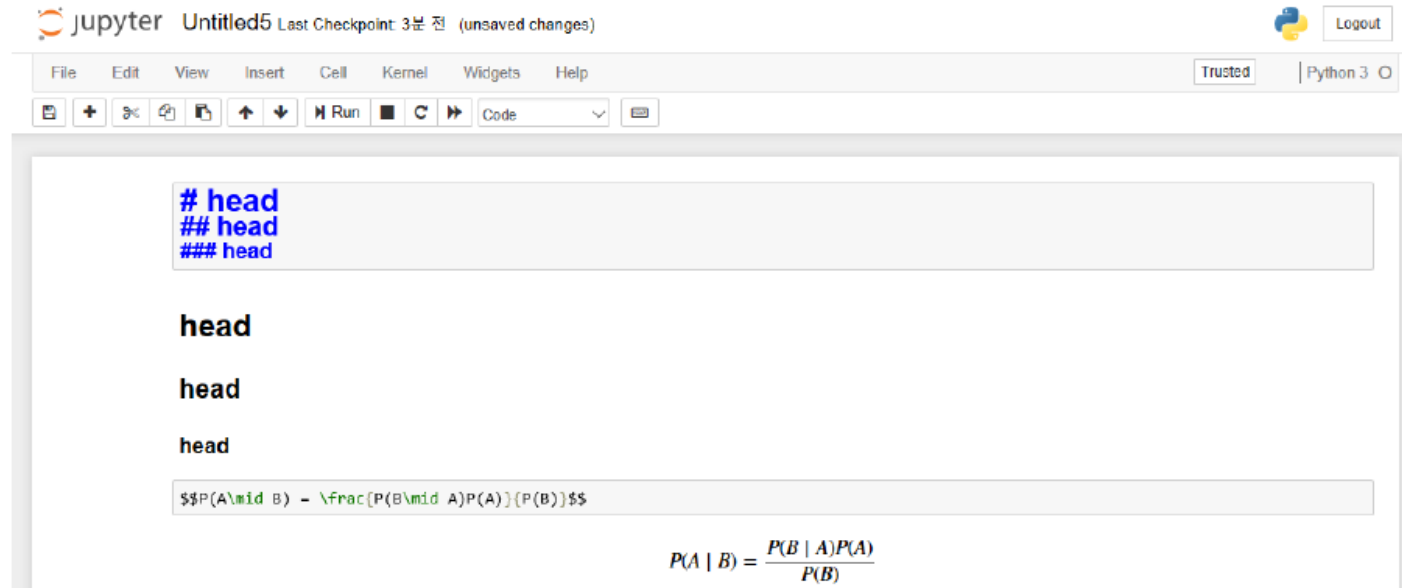
마지막으로 jupyter notebook

```
(growthhackers) C:\Windows\system32>jupyter notebook  
[I 00:17:11.358 NotebookApp] Serving notebooks from local directory: C:\Windows\system32
```



[참고] Jupyter notebook 코드 & 마크다운

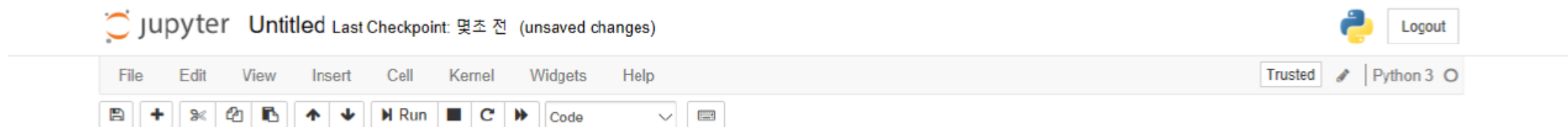
- Code & Markdown






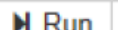






Markdown에 대해 더 알고 싶다면?

<https://kevinthegrey.tistory.com/74?category=793117>:

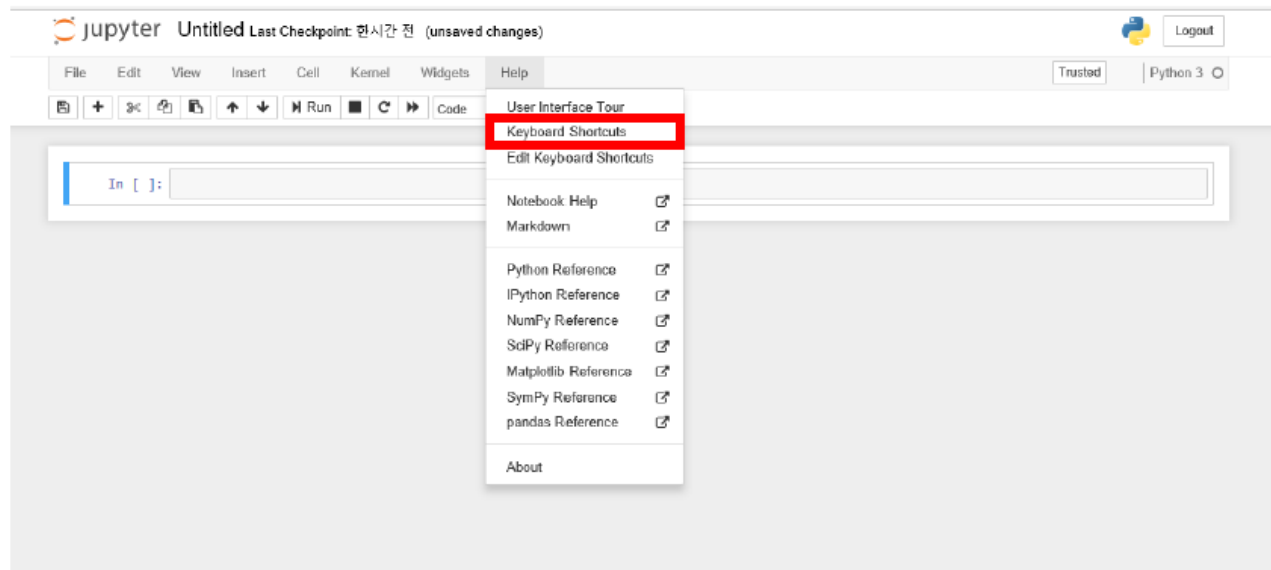
[참고] Jupyter notebook 사용법



- | | |
|--|--|
|  Save: 저장 [S] |  위의 셀로 이동 [Up], [K] |
|  Add: 새로운 셀 추가 [A], [B] |  아래의 셀로 이동 [Down], [J] |
|  Cut: 잘라내기 [X] |  Run: 입력한 코드 실행 [Shift] + [Enter] |
|  Copy: 복사 [C] |  Code [Y], Markdown [M] 등 모드 변경 |
|  Paste: 붙여넣기 [V] |  Command Palette: 다양한 명령을 검색할 수 있는 검색창 패널 [P] |

[참고] Jupyter notebook 사용법

- 단축키 (Help > Keyboard Shortcuts)



단축키에 대해 더 알고 싶다면?

<https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts>

[참고] Jupyter notebook 사용법

• 기본 단축키

- Shift + Enter : 해당 Cell 실행, 그 아래 Cell 선택
- Ctrl + Enter : 해당 Cell 실행
- Alt + Enter: 해당 Cell 실행, 그 아래에 새로운 Cell 하나를 추가
- ➔ 작성된 코드를 실행할 때는 주로 Shift + Enter / 자신이 코드를 작성할 때는 주로 Alt + Enter
- a : 현재 cell 바로 위에 cell 하나 추가
- b : 현재 cell 바로 아래에 cell 하나 추가
- c : cell 복사하기
- v : 현재 cell의 아래에 붙여넣기
- shift + v : 현재 cell의 위에 붙여넣기
- d : 셀 삭제
- ctrl + z : 되돌리기

```
In [1]: print("Hello World!")
```

Hello World!

3. AWS (Amazon Web Service)

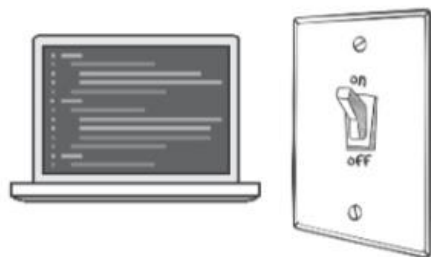
- 무슨 서비스?



<https://www.slideshare.net/awskorea/aws-for-alpha-users-1>

3. AWS (Amazon Web Service)

내 로컬 너무 느려..
메모리 모자라..
램이 모자라..
용량도 모자라..
요런 고민들의 SOLUTION!!



몇 번의 클릭과 간단한 명령만으로
IT 자원을 언제 어디서나
바로 사용할 수 있습니다!



CPU, 메모리, 스토리지, 네트워크, 데이터베이스...

주 정의:

인터넷을 통해 (물리적 서버와 네트워크 같은) IT 리소스와 (데이터 분석과 같은) 애플리케이션을 원할 때 언제든지(On-Demand) 사용한 만큼만 요금을 내는 서비스

<https://www.slideshare.net/awskorea/aws-for-alpha-users-1>

3. AWS (Amazon Web Service)

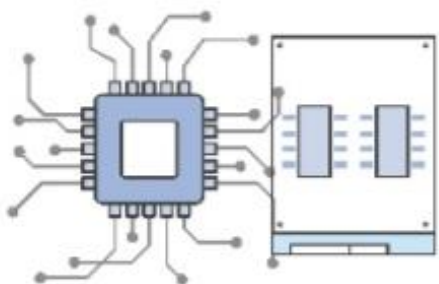
AWS 국내 고객 현황

잘 나감. 기업에서 많이 씀.
이게 뭔지는 알아두자.



3. AWS (Amazon Web Service)

물리적 장치를 가상 서비스로...



CPU/메모리

**Amazon Elastic
Compute Cloud
(EC2)**



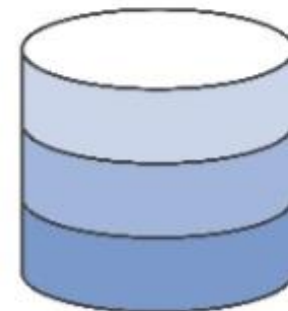
하드디스크

**Amazon Elastic
Block Store
(EBS)**



스토리지

**Amazon Simple
Storage Service
(S3)**



데이터베이스

**Amazon
Relational DB
Services (RDS)**

2006년 클라우드 컴퓨팅 기반(첫 시작 서비스는 EC2, S3, SQS) 으로 서비스 확대

3. AWS – 제공 서비스 크게 2종류

AWS 기초 서비스

1. 컴퓨팅(서버) : EC2, Lambda, Auto Scaling(서버 자동증설) 등
2. 네트워킹 : VPC, Route 53
3. 스토리지 : S3, EBS, Glacier, CloudFront
4. 관리 및 보안 (계정 관리와 모니터링) : IAM, CloudWatch
5. 어플리케이션

AWS 플랫폼 서비스

1. 데이터베이스
2. 분석 (머신러닝, 딥러닝 ++)
3. 앱 서비스
4. 배포 및 관리
5. 모바일 서비스

더 자세히 알고 싶으면

<https://brunch.co.kr/@topasvga/1>
https://cloud-img.hosting.kr/wp-content/uploads/2018/06/29180750/AWS_%ED%95%B5%EC%8B%AC_%EC%84%9C%EB%B9%84%EC%8A%A4_%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0.pdf

3. AWS – 기초 용어

- 리전(Region): 서버의 물리적인 위치 (ex. 도쿄, 북경, 싱가포르, 시드니)
- 가용 영역(Availability Zone) : 리전 내부에 분리된 서버군, 데이터센터 (서버 클러스터 ex. Ap-northeast-1a)
- 엣지 로케이션(Edge location): 리전이나 AZ가 없는 곳에 더 빠르게 서비스하기 위한 전송 서버군

<https://web-front-end.tistory.com/74>

3. AWS – EC2 위주 용어

- EC2 (Elastic Compute Cloud) : 가상 인스턴스를 운영하는 서비스
- 가상 서버: CPU와 메모리를 가진 클라우드 내 서버
- 인스턴스: AWS에서 가상 서버를 부르는 용어 (하나의 컴퓨터다!)
- AMI: 가상머신, 운영체제 등 설정해둔 템플릿
- 관리 콘솔 : AWS 서비스를 모두 관리하는 사용자 인터페이스
- 클라우드 와치 : AWS 자원을 모니터링하는 서비스

3. AWS – EC2 (Elastic Compute Cloud)

- GH는 이번학기 EC2 컴퓨팅 파워를 사용할 예정!
- 콘솔을 들어가보자!

THEME B. Git 과 GitHub

1. 버전관리

개념

2. Git

로직

3. GitHub

로직과 단계 / 용어 / 추가 기능

4. ☆★ 실습 ★★

일단, 버전 관리부터 알자.


영어로는, Version Control이구요.

151110_코드_(1).txt	2015-11-11 오후...	텍스트 문서
151110_코드_(2) - 수정.txt	2015-11-11 오후...	텍스트 문서
151110_코드_(3) - 기존코드백업.txt	2015-11-11 오후...	텍스트 문서
151110_코드_(4) - 수정2.txt	2015-11-11 오후...	텍스트 문서
151110_코드_(5) - 수정3.txt	2015-11-11 오후...	텍스트 문서
151111_코드_(6) - 완성.txt	2015-11-11 오후...	텍스트 문서
151112_코드_(7) - 완성 - 수정.txt	2015-11-11 오후...	텍스트 문서

버전 관리는 사실 우리가 자주 하던 짓

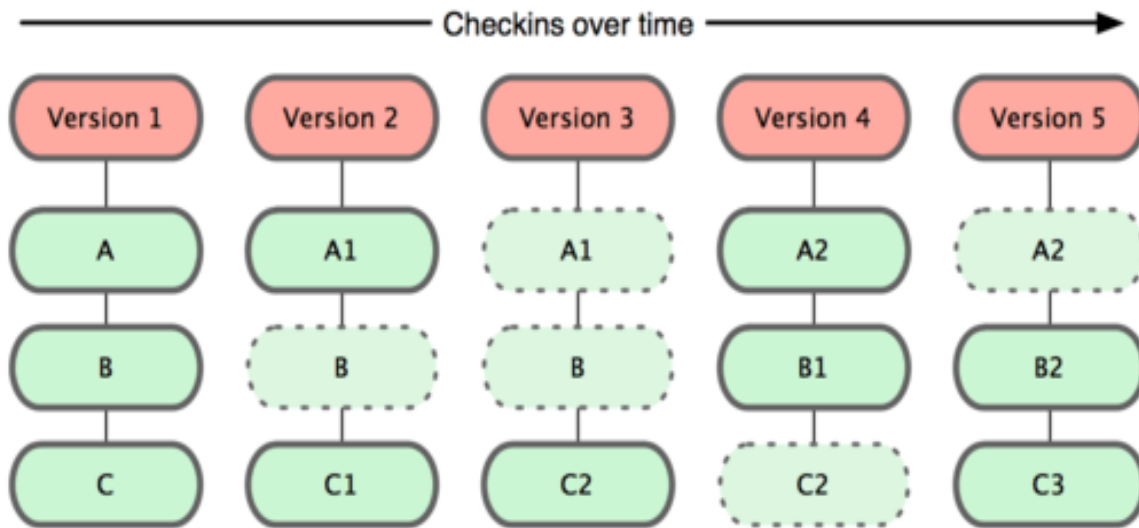
버전 관리를 위한 프로그램

- 버전 관리 시스템(Version Control System, 줄여서 VCS)

- 주 사용 분야 : 소프트웨어 개발 현장에서 사용하는 프로그램
- 필요성: 여러 개발자가 협업하여 코드를 작성할 때, 코드를 추가 & 변경하는 과정을 모두 기록하여 특정한 시점으로 돌아가거나, 문제 생기는 파일을 복원하기 위함 + 효율적인 소스코드 관리
- 예시 : CVS, Subversion(SVN), Mercurial, 그리고.. 

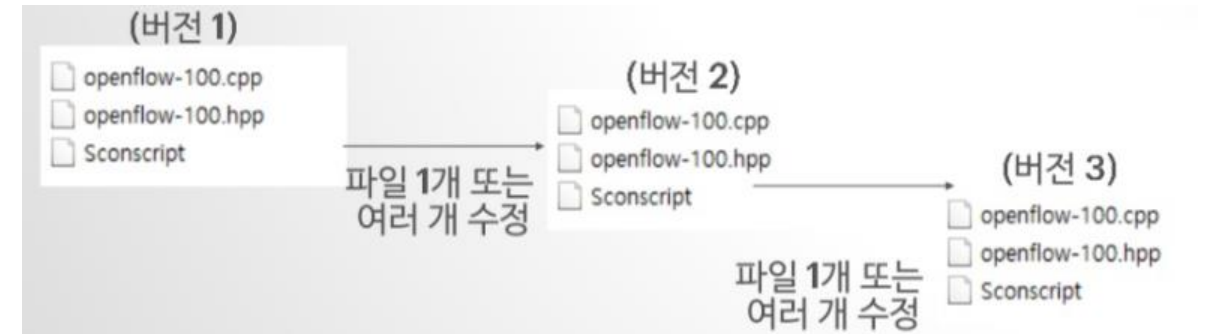
Git 의 로직

로직



1. 시간 순으로 프로젝트의 스냅샷을 저장한다.
2. 각 파일이 변화했을 때만 저장한다.
3. 파일이 변화하지 않았을 때는 파일에 대한 링크만 저장한다.

예시



[참고] Git이 왜 특별해?

• Git 특징

- Git은 다른 도구들과는 달리 '**분산**' 버전 관리 시스템.
- 여기서 **분산의 의미**는 각 개발자가 중앙 서버에 접속하지 않은 상태에서 코드 작업 가능하다는 것이 핵심
- 서버와 통신 (네트워크와 연결) 되지 않아도, 우선 자신의 컴퓨터에서 버전 관리한 후에 통신할 수 있다는 것

더 알고 싶다? <https://goddaehee.tistory.com/91>

Git 그래! 이해했어 그럼 GitHub은 ?



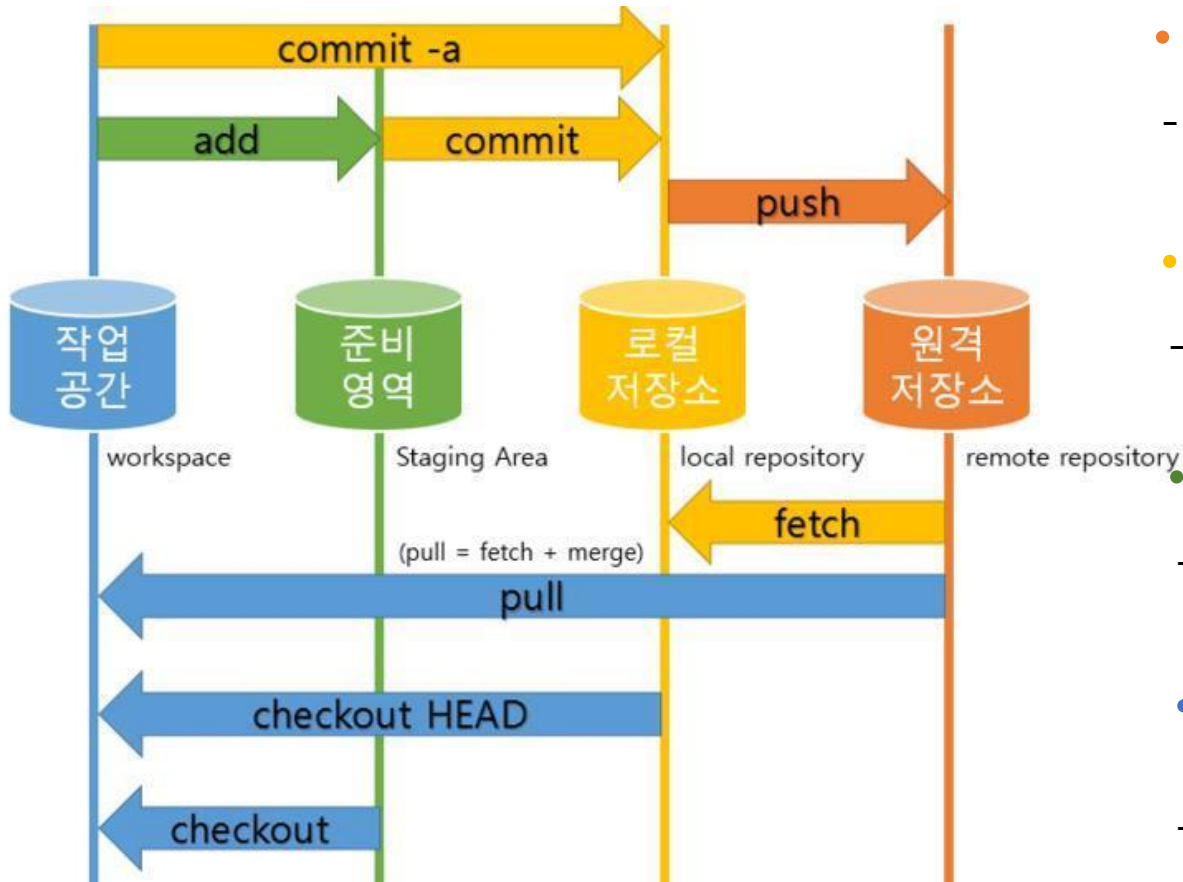
<http://www.github.com>

Git에 이어 한 단계를 더 추가하는 것.

Git에 **서버 단계**를 추가한 것.

로컬 PC → **원격 저장소 서버를 제공하는 서비스**

GitHub 로직과 단계 (중요)



- **원격 저장소 (Remote Repository) - Server**
 - 스냅샷이 2차로 원격 저장소로 저장되고 전용 '서버'에서 관리되며 여러 사람이 함께 접근, 공유할 수 있는 저장소
- **로컬 저장소 (Local Repository) - Local**
 - 스냅샷들이 1차로 개인 Local repository에 저장
- **준비 영역 (Staging Area) - Local**
 - 파일의 스냅샷을 찍어 두고 Repository들에 올리기 전 저장소
- **작업 공간(Workspace, Working Directory) - Local**
 - 개인이 파일을 수정하고 작업하는 곳

로직 이해 위해 익혀 둘 용어

- 핵심적으로 알아야 하는 용어 (중요!)

파일의 움직임 중 3가지

- **Commit**
 - Staging area에서 로컬 저장소로 보내 둔다
- **Push**
 - 로컬 저장소에서 원격저장소로 최종적으로 보낸다
- **Pull**
 - 원격 저장소에서 working directory로 받아온다

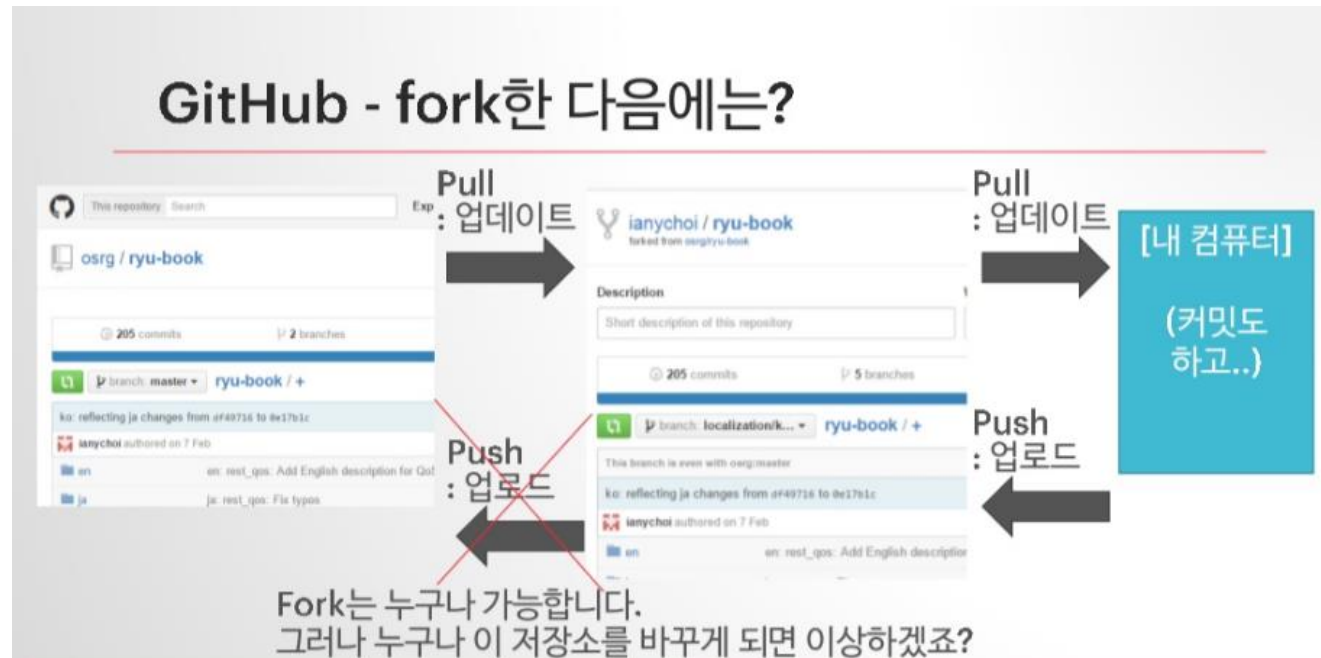
3가지 파일의 상태

- **Committed (좋음!, push만 하면 끝)**
 - 데이터가 로컬 저장소에 안전하게 저장됨
- **Modified (로컬 저장소에 있는 파일과 다름)**
 - 수정한 파일이 로컬 저장소에 안전하게 저장 안 됨
- **Staged (staging area에 보내 둔 상태)**
 - 수정한 파일이 곧 로컬 저장소로 저장되기 직전 상태

[참고] GitHub 추가 기능

GitHub에서 제공하는 서비스 (오픈 소스이자 커뮤니티 공간으로의 특징)

- Fork
 - 다른사람 원격 저장소를 가져와 제 저장소로 만들어 놓는 기능

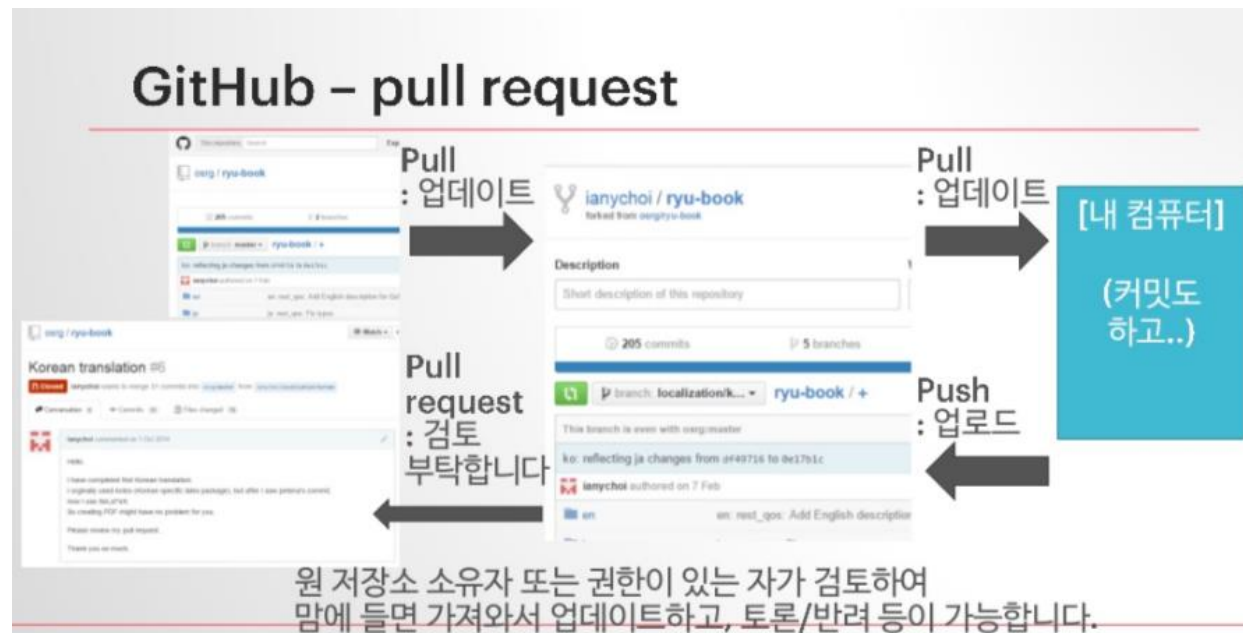


출처 :
<https://www.sli-deshare.net/ianychoi/git-github-46020592>

[참고] GitHub은 하나의 문화

GitHub에서 제공하는 서비스 (오픈 소스이자 커뮤니티 공간으로의 특징)

- Pull Request (개발자들의 코드 리뷰 문화)
 - 원 저장소 소유자 또는 권한이 있는 자에게 수정한 파일을 검토해주세요.
 - 그리고 업데이트 해주세요. 하는 것



출처 :
<https://www.sli-deshare.net/ianychoi/git-github-46020592>

[참고] GitHub은 커뮤니티

GitHub에서 제공하는 서비스 (오픈 소스이자 커뮤니티 공간으로의 특징)

- 블로그로도 사용!



좋은 예시 변성윤님 블로그 ^^*

[https://www. https://zzsza.github.io/](https://www.https://zzsza.github.io/)

- 개발자/ 디자이너 포트폴리오

Awesome Creative Portfolios

https://github.com/iRaul/awesome-portfolios?source=post_page-----5e773c68d300-----

9/21/2019

☆ ★ 4 GIT BASH 통해 실습

미션이 6가지다.

42

회원가입부터 하자

빨리 해보자!

- www.github.com

Would you like to set up an organization account for your team? ⓘ

☐ Yes, we need an organization account.



Please verify your email address

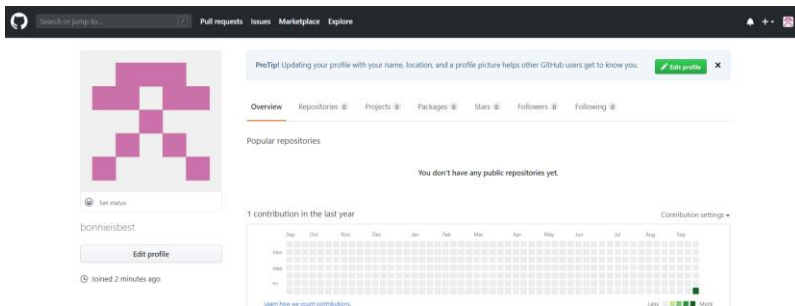
Before you can contribute on GitHub, we need you to verify your email address.

체크 아니요!

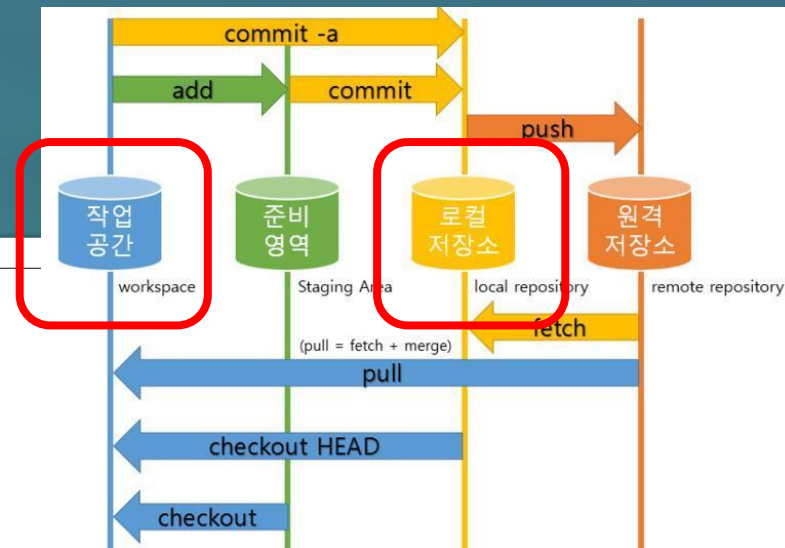
Free Account로 해주세요

이메일 verify해주세요.

자기 페이지 뜨면 완성!



9/21/2019



1) GIT 로컬저장소

(Mission1) 로컬저장소를 만들고, 이 로컬저장소에 파일을 추가하자!

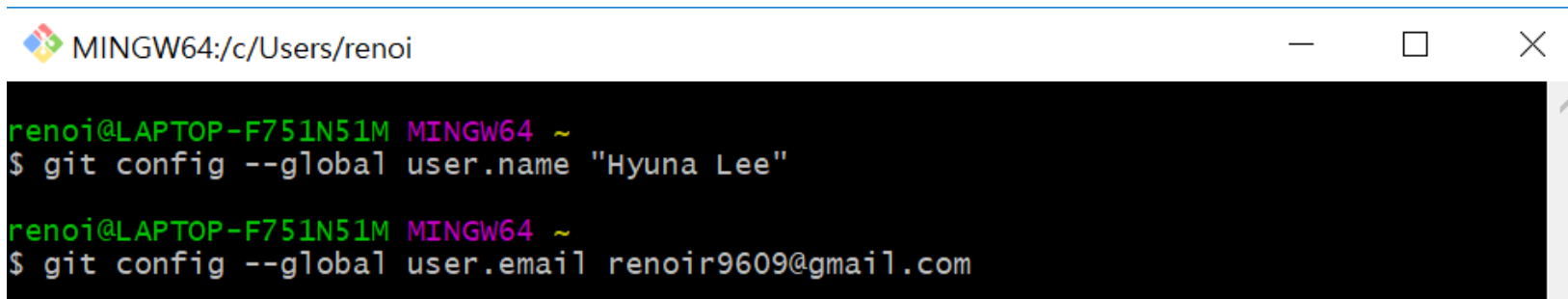
44

1. Git 사용 준비 – Git Bash 열고 정보 입력

- 전역 사용자 이름 및 이메일 설정, 확인 [최초 1회만 실행하면 됩니다!]

- git config --global user.name "사용자이름"
- git config --global user.email 이메일 주소
- git config --global --list

명령어

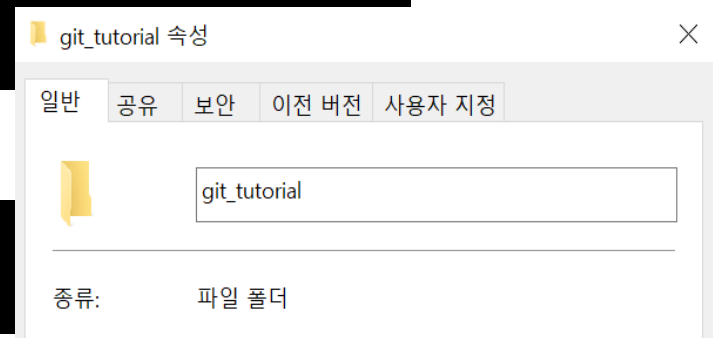


```
MINGW64:/c/Users/renoi
renoi@LAPTOP-F751N51M MINGW64 ~
$ git config --global user.name "Hyuna Lee"
renoi@LAPTOP-F751N51M MINGW64 ~
$ git config --global user.email renoir9609@gmail.com
```

2. Working directory 생성 후 로컬저장소로 만들기

mkdir
: 디렉토리 생성

```
renoi@LAPTOP-F751N51M MINGW64 ~  
$ mkdir git_tutorial
```



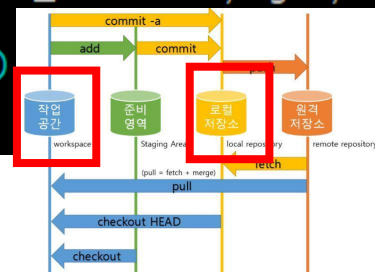
cd
: 디렉토리로 이동

```
renoi@LAPTOP-F751N51M MINGW64 ~  
$ cd git_tutorial
```

mkdir 명령어로, 컴퓨터에 폴더가 생겨났다.

git init
: 실행위치를
로컬저장소로 만들기

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial  
$ git init  
Initialized empty Git repository in C:/Users/renoi/git_tutorial/.git/  
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ |
```

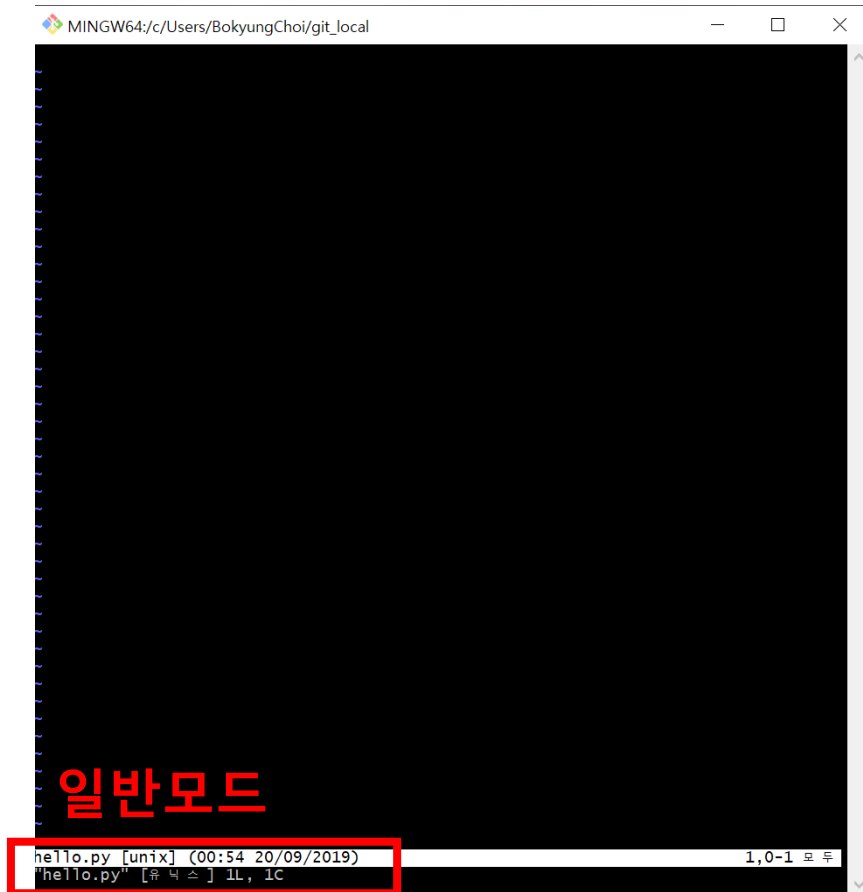


3-1. 파일 – Git Bash에서 생성 (vim)

- 이제 파일 올리기; git bash에서 가상으로 hello.py라는 파일을 만들어봅시다

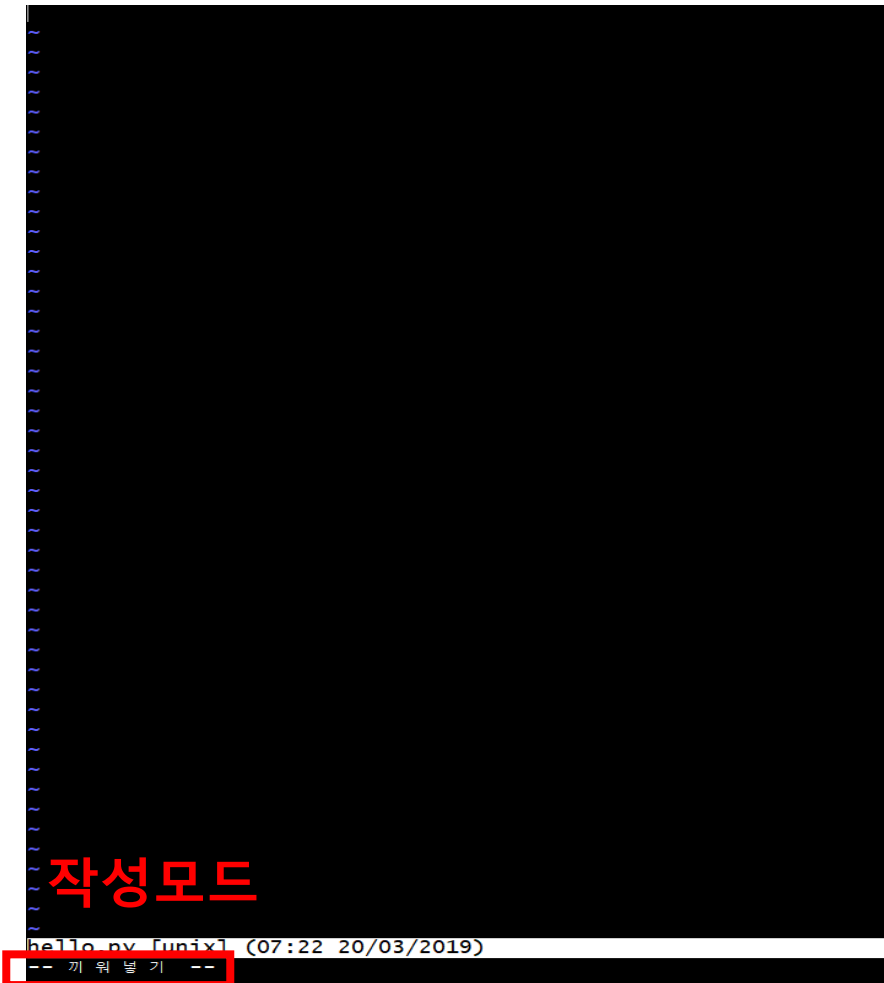
```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ vim hello.py
```

3-1. 파일 – Git Bash에서 생성 (vim)



- **Vim이란?**
 - 리눅스나 Unix에서 사용되는 텍스트 편집기
- **모드**
 - 일반 모드, 작성 모드, 명령 모드

3-1. 파일 – Git Bash에서 생성 (vim)



- Vim이란?
 - 리눅스나 Unix에서 사용되는 텍스트 편집기
- Vim 사용법
 - Vim에서 작성을 시작하려면
키보드의 [i]키를 눌러 작성모드로 바꾸기
 - 일반모드로 돌아오려면
작성 후에 [ESC]키를 누르기
 - : 를 입력하면 명령모드로 바뀌고,
 - 명령모드에서 wq를 입력하면 저장(w) 후 종료(q)됨
즉, :wq 를 입력하면 vim이 저장 종료됨

3-1. 파일 – Git Bash에서 생성 (vim)



- **Vim이란?**
 - 리눅스나 Unix에서 사용되는 텍스트 편집기
- **Vim 사용법**
 - Vim에서 작성을 시작하려면
키보드의 [i]키를 눌러 작성모드로 바꾸기
 - 일반모드로 돌아오려면
작성 후에 [ESC]키를 누르기
 - **:를 입력하면 명령모드로 바뀌고,**
 - 명령모드에서 wq를 입력하면 저장(w) 후 종료(q)됨
즉, :wq 를 입력하고 엔터!하면 vim이 저장 종료됨

3-2. 파일 – Git Bash에서 내용 조회 (cat)

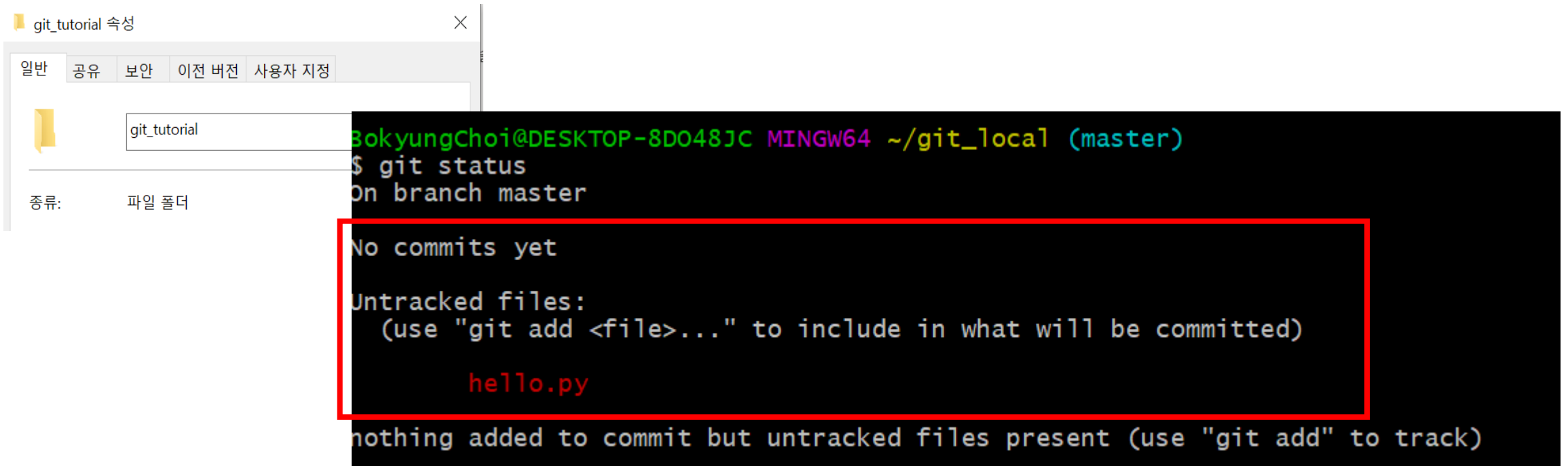
- 파일 내용을 확인해봅니다.

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ cat hello.py  
print("Hello GH")
```

(참고) 내 컴퓨터의 디렉토리에 직접 .py 파일을 만들어도 됨

4. 저장소 상태 확인 (git status)

- 로컬 저장소에 hello.py 파일을 만들었으나, 파일이 없다네?? Untracked !



The screenshot shows a Windows File Explorer window titled 'git_tutorial 속성' (git_tutorial Properties) with tabs for '일반' (General), '공유' (Sharing), '보안' (Security), '이전 버전' (Previous Versions), and '사용자 지정' (Customization). The '일반' tab is selected, showing the folder name 'git_tutorial' and its type '파일 폴더' (File Folder). Overlaid on the right is a terminal window with a black background and green text. The terminal shows the command '\$ git status' and its output, which is highlighted with a red rectangle. The output indicates that there are no commits yet and that 'hello.py' is an untracked file.

```
sokyoungchoi@DESKTOP-8D048JC MINGW64 ~/git_local (master)
$ git status
On branch master

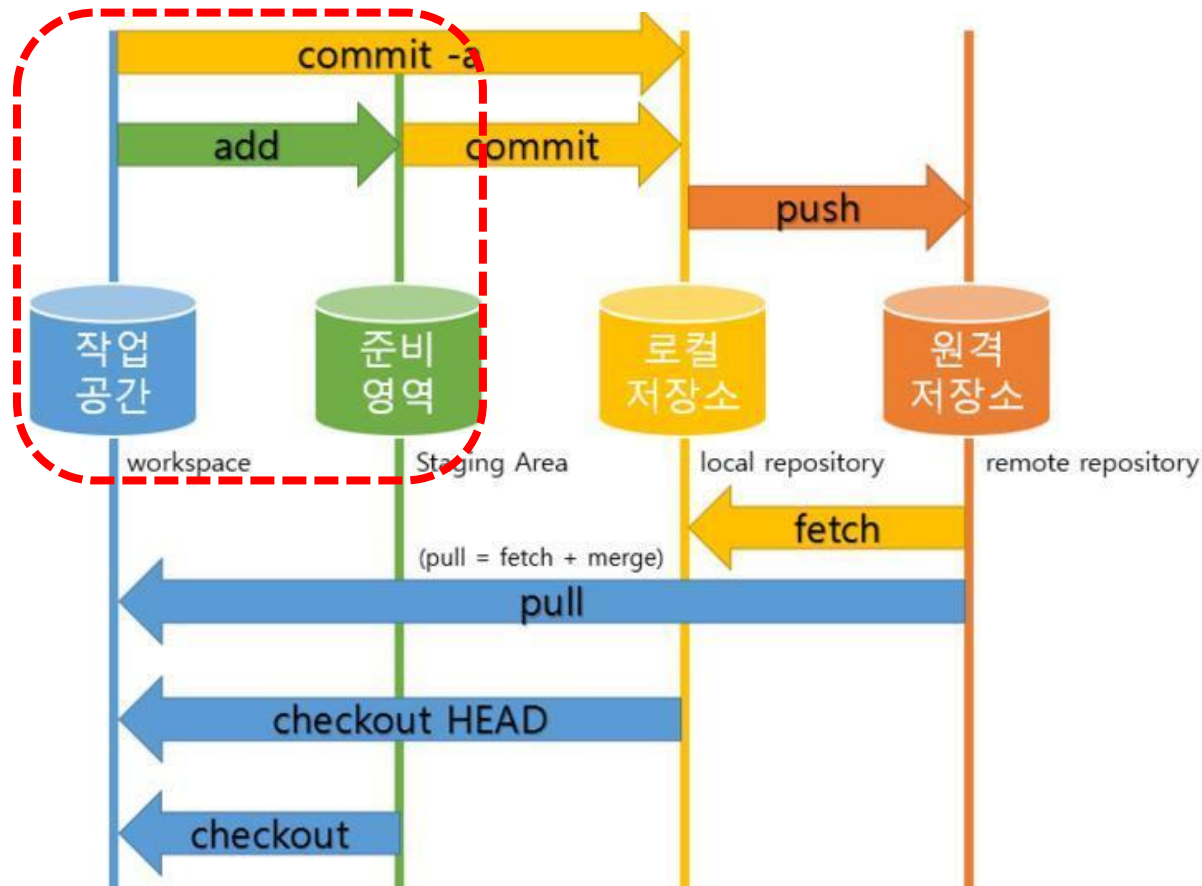
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        hello.py

nothing added to commit but untracked files present (use "git add" to track)
```

* 아직 덜 올려져서 그래



- 단순히 파일을 bash에서 생성하는 것만으로는 staging area로 올려지지 않는다.

- 1) vim으로 파일을 생성한 후 git add
- 2) 워킹 디렉토리에 있던 실존 파일을 git add를 통해 staging area로 올려줘야 한다.

* 그래서 저장소에 파일 추가 (git add)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ git add hello.py
```

git add

: 해당 파일을 git이 추적할 수 있게 staging area 저장소에 추가하기

5. 저장소 상태 다시 확인 (git status)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

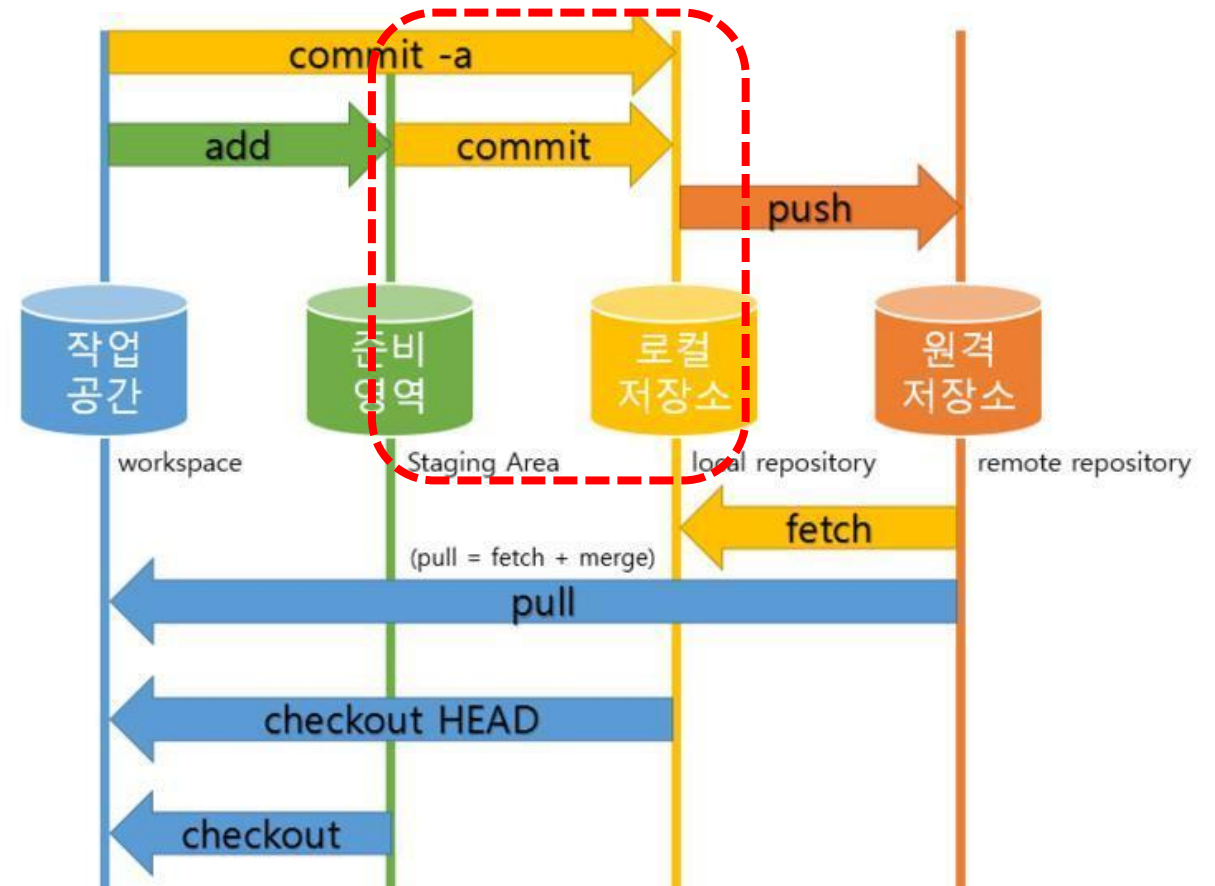
    new file:   hello.py
```

- 이젠 파일이 있다

6. Staging area 에서 커밋 (git commit)

```
renoi@LAPTOP-F751N51M MINGW64 ~  
$ git commit
```

git commit
: 변경된 파일을 로컬 저장소에 저장하기



6. 늘 뜨는 커밋 메시지 입력 (vim)

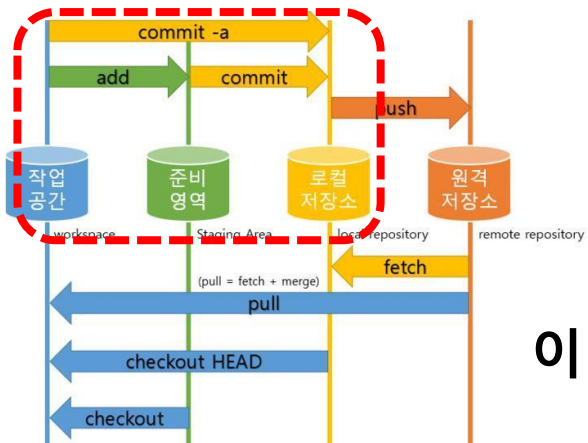
[illegible]

- **Vim이란?**
 - 리눅스나 Unix에서 사용되는 텍스트 편집기
- **Vim 사용법**
 - Vim에서 작성을 시작하려면 키보드의 [i]키를 눌러 작성모드로 바꾸기
 - 작성 후에 [ESC]키를 누르면 일반모드로 돌아옴
 - :를 입력하면 명령모드로 바뀌고, 명령모드에서 wq를 입력하면 저장(w) 후 종료(q)됨

Mission1 성공!

(Mission1) 로컬저장소를 만들고, 이 로컬저장소에 파일을 추가하자!

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git commit
[master (root-commit) 7da5211] create "hello world" program
1 file changed, 1 insertion(+)
create mode 100644 hello.py
```



이 빨간 단계까지 하셨어요

9/21/2019

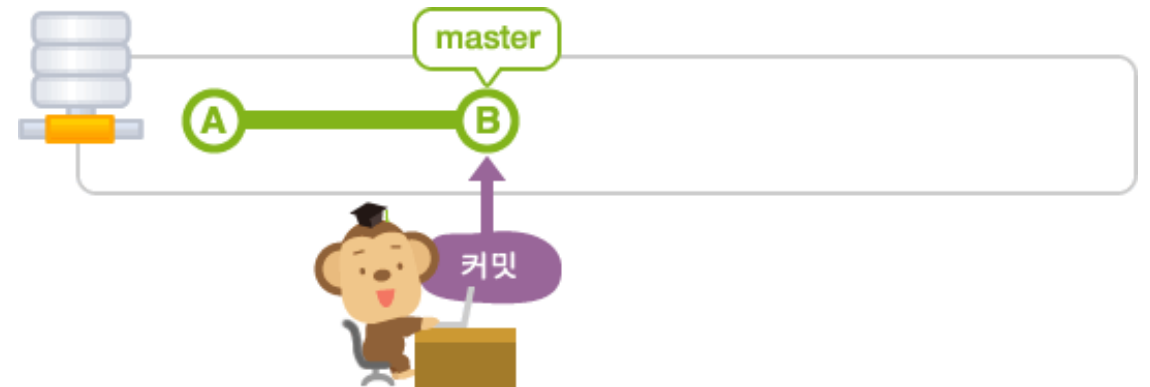
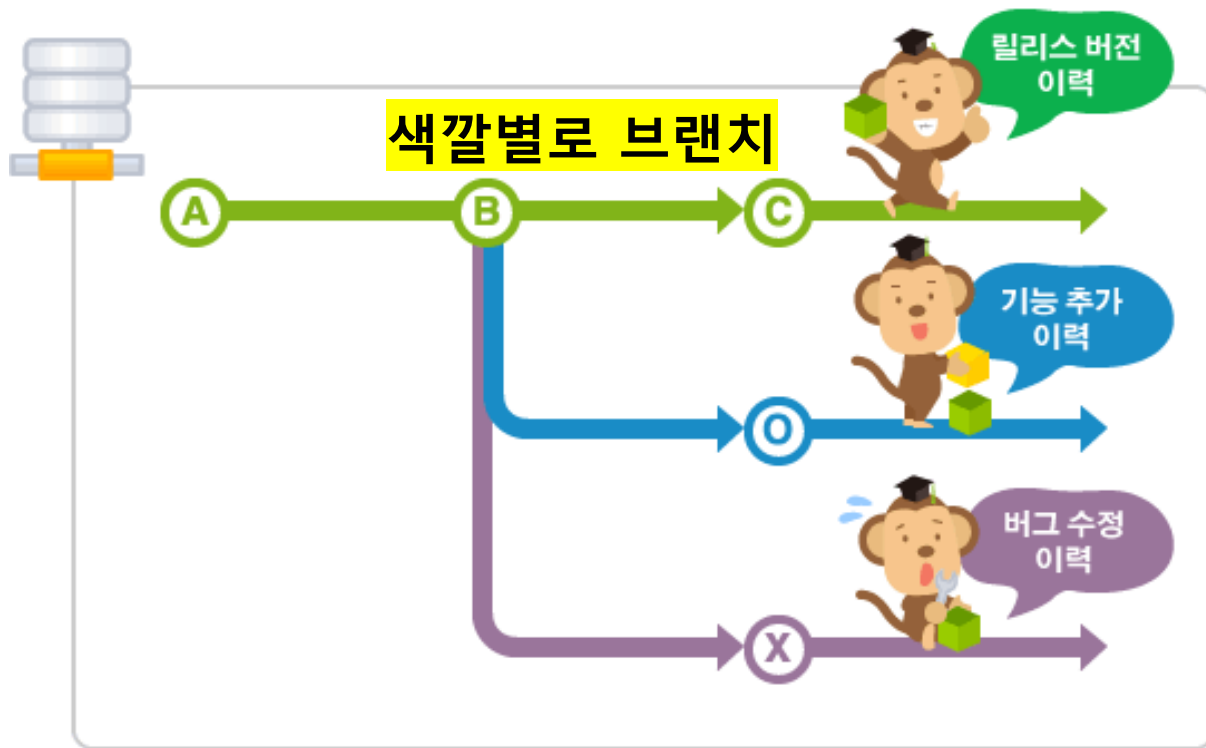
2) BRANCH와 MERGE

(Mission2) 새로운 branch를 만들고,
이 branch에서 파일을 수정해서 다시 master branch에 반영시키자!

59

Git – Branch 개념

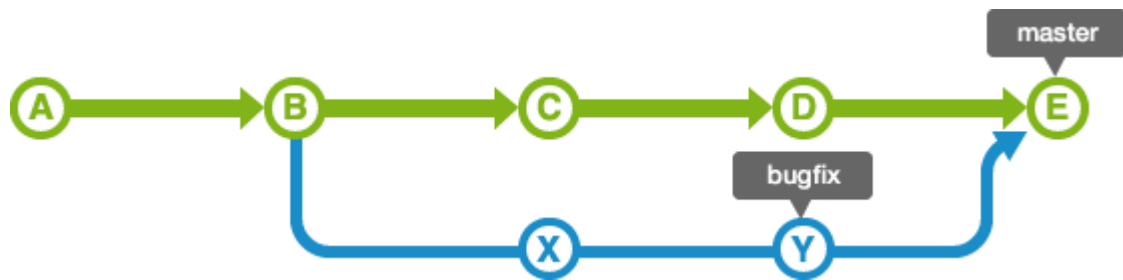
여러 명에서 동시에 작업을 할 때의 작업 흐름 눈에 파악



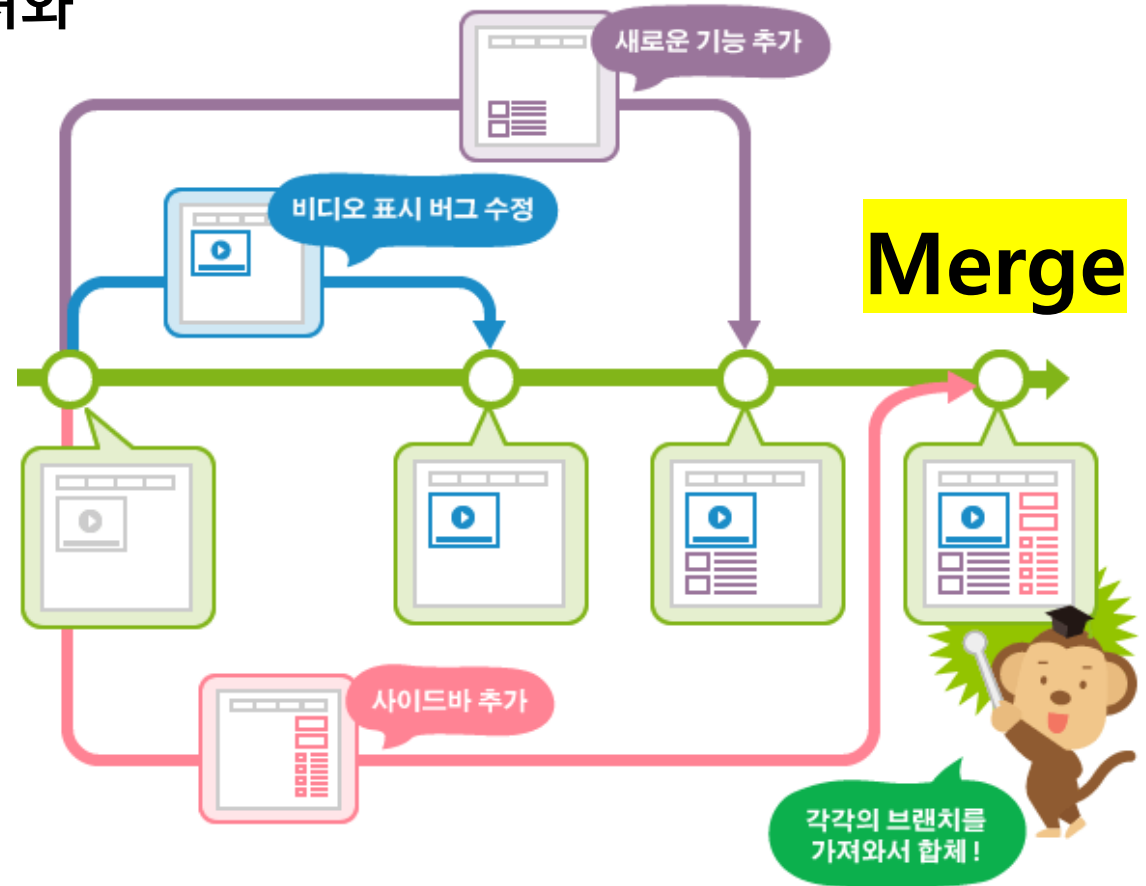
혼자 사용할 때는 새로운 브랜치를 만들지 않아도 충분히 master 하나로 처리할 수 있다

Git – Merge 개념

여러 명이서 동시에 작업한 브랜치들을 각각 가져와
통합 브랜치에 합체



https://backlog.com/git-tutorial/kr/stepup/stepup1_1.html



Branch 확인하기 (git branch)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git branch
* master
```

Branch 만들기 (git branch 이름)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ git branch hotfix
```

(참고) hotfix라는 이름은 주로
: 프로그램 사용 중 발견된 버그(bug)의 수정이나 취약점 보완, 성능 향상을 위해 긴급 배포되는 응급 패치 프로그램.

Branch 다시 확인하기 (git branch)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git branch
  hotfix
* master
```


다른 branch로 이동 (git checkout)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git checkout hotfix
Switched to branch 'hotfix'

renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (hotfix)
$ |
```

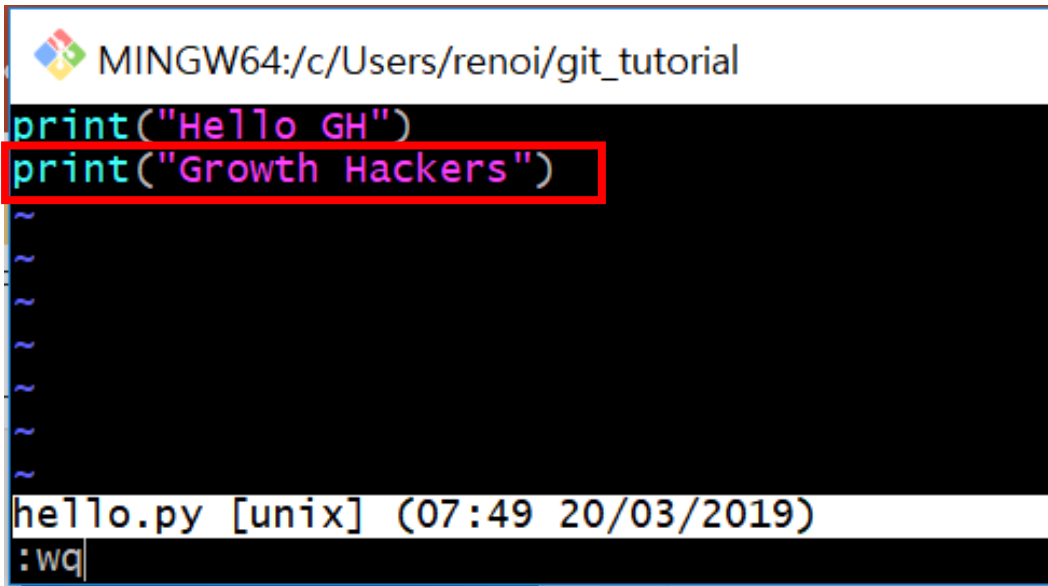
파일 확인 (ls)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (hotfix)
$ ls
hello.py
```

파일 수정 (vim)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (hotfix)  
$ vim hello.py|
```

파일 수정 (vim)



```
MINGW64:/c/Users/renoi/git_tutorial
print("Hello GH")
print("Growth Hackers")
~
~
~
~
hello.py [unix] (07:49 20/03/2019)
:wq
```

- **Vim이란?**

- 리눅스나 Unix에서 사용되는 텍스트 편집기

- **Vim 사용법**

- Vim에서 작성을 시작하려면
키보드의 [i]키를 눌러 작성모드로 바꾸기
- 일반모드로 돌아오려면
작성 후에 [ESC]키를 누르기
- : 를 입력하면 명령모드로 바뀌고,
- 명령모드에서 wq를 입력하면 저장(w) 후 종료(q)됨
즉, :wq 를 입력하면 vim이 저장 종료됨

파일내용 확인 (cat)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (hotfix)
$ cat hello.py
print("Hello GH")
print("Growth Hackers")
```

저장소 상태 확인 (git status)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (hotfix)
$ git status
On branch hotfix
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   hello.py

no changes added to commit (use "git add" and/or "git commit -a")
```

- 파일이 수정되어 있는 상태라 modified, 변경사항이 반영되지 않음

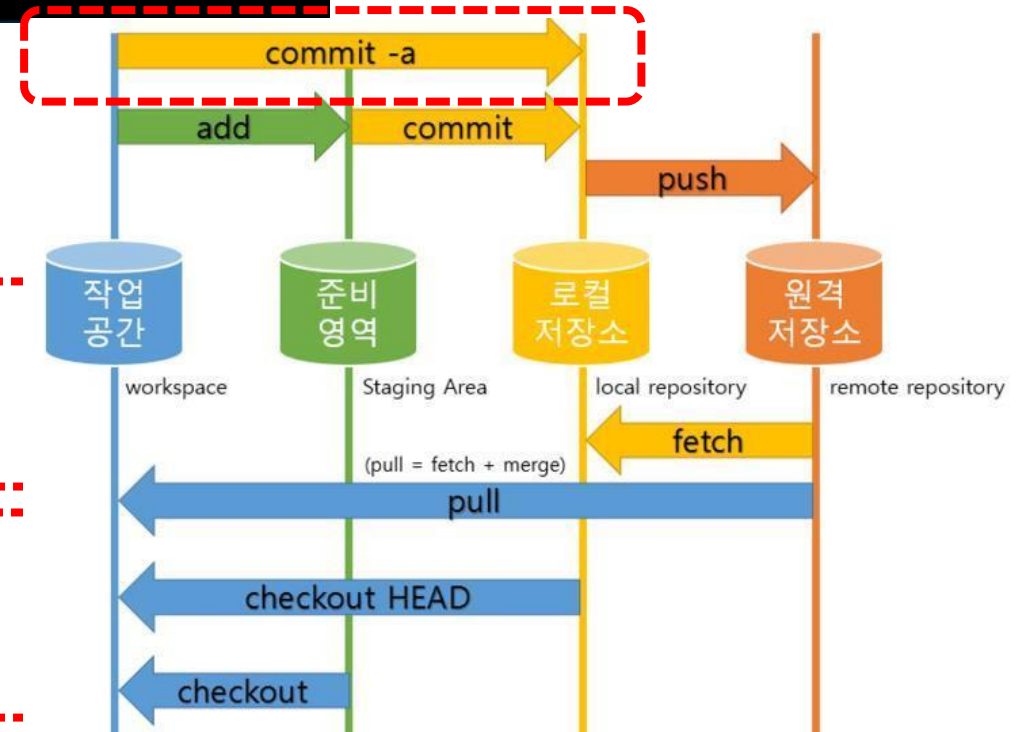
커밋 (git commit -a 또는 add & commit)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (hotfix)
$ git commit -a
```

Hotfix branch를 통해 바로 로컬 저장소에 저장

git commit -a
: 변경된 저장소 파일을 모두 저장소에 제출(커밋)

git add 변경한 파일
git commit 변경한 파일
: 변경된 파일 하나만 골라서 제출(커밋)



커밋 메시지 작성 (vim)

```
MINGW64:/c/Users/renoi/git_tutorial
added output "Growth Hackers"
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch hotfix
# Changes to be committed:
#   modified:   hello.py
#
~
<rs/renoi/git_tutorial/.git/COMMIT_EDITMSG[+] [unix] (07:54 20/03/2019):
:wq
```

• Vim이란?

- 리눅스나 Unix에서 사용되는 텍스트 편집기

Vim 사용법

- Vim에서 작성을 시작하려면
키보드의 [i]키를 눌러 작성모드로 바꾸기
- 일반모드로 돌아오려면
작성 후에 [ESC]키를 누르기
- : 를 입력하면 명령모드로 바뀌고,
- 명령모드에서 wq를 입력하면 저장(w) 후 종료(q)됨
즉, :wq 를 입력하면 vim이 저장 종료됨

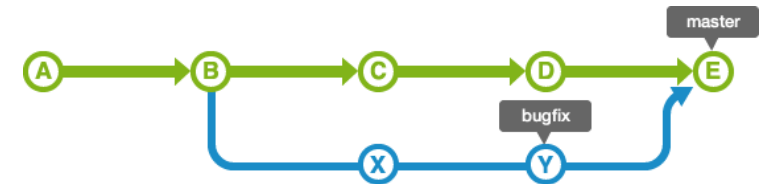
다른 branch로 이동 (git checkout)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (hotfix)  
$ git checkout master  
Switched to branch 'master'
```

저장소 파일 상태 확인 (ls / cat)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ ls
hello.py

renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ cat hello.py
print("Hello GH")
```



hotfix 브랜치에서 수정한 내용이 master 브랜치에는 반영되지 않은 상태!
➔ 반영시키고 싶다면?

Branch 병합 (git merge)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git merge hotfix
Updating 7da5211..443df5b
Fast-forward
 hello.py | 1 +
 1 file changed, 1 insertion(+)
```

(주의) master 브랜치에서 "git merge hotfix"를 하는 것과
hotfix 브랜치에서 "git merge master"를 하는 것은 **다르다**
→ 현재 위치한 브랜치가 수정사항을 모으는 브랜치(통합 브랜치라고 함)여야 함!

Mission2 성공!

- (Mission2) 새로운 branch를 만들고,
이 branch에서 파일을 수정해서 다시 master branch에 반영시키자!

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ ls
hello.py

renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ cat hello.py
print("Hello GH")
print("Growth Hackers")
```

* [참고] 커밋 내역 확인하기 (git log)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git log
commit 4ccb158870338a77e1715e36325c464235a22c9f (HEAD -> master)
Merge: e96095d e93f43a
Author: Hyuna Lee <renoir9609@gmail.com>
Date:   Wed Mar 20 23:40:25 2019 +0900

    conflict resolved GitHub

commit e96095dbfcf97d4a4e7303528c98e6ba53c8aef9
Author: Hyuna Lee <renoir9609@gmail.com>
Date:   Wed Mar 20 23:34:18 2019 +0900

    hello.py modified on Local repository

commit e93f43a0dce56baa1179034a12d18d0317eb849b
Author: Hyuna Lee <43376842+hysophie@users.noreply.github.com>
Date:   Wed Mar 20 23:33:46 2019 +0900

    hello.py modified on GitHub
```

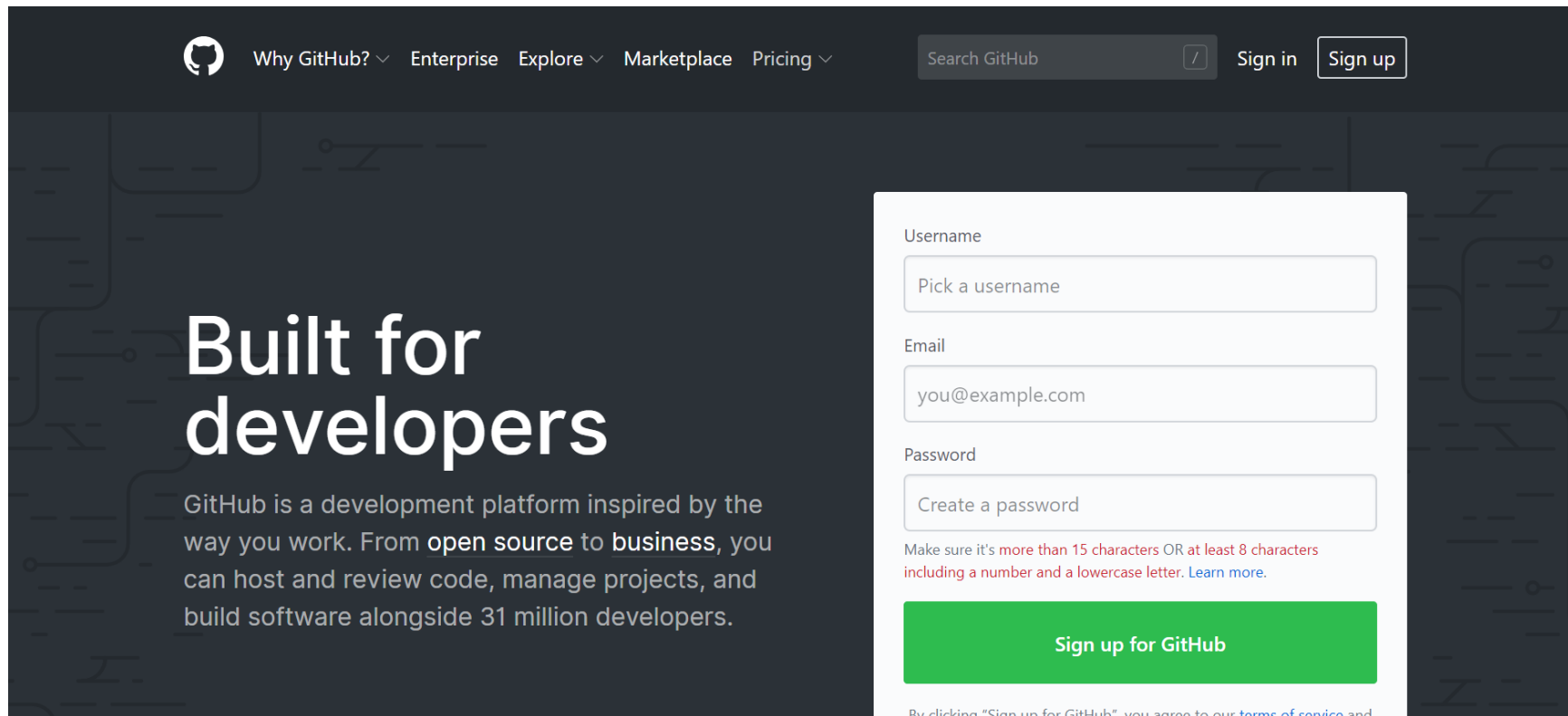
9/21/2019

3) GITHUB -원격저장소 파일 복사 받아오기

(Mission3) GitHub에 원격저장소를 만들고, 원격저장소를 로컬저장소에 복사해오자!

78

GitHub repository(저장소) 만들기



The image shows the GitHub sign-up page. The header includes the GitHub logo, navigation links (Why GitHub?, Enterprise, Explore, Marketplace, Pricing), a search bar, and 'Sign in' and 'Sign up' buttons. The main content area features the text 'Built for developers' and a description of GitHub as a development platform. On the right, there is a sign-up form with fields for Username, Email, and Password, followed by a green 'Sign up for GitHub' button. Below the button, there is a small disclaimer about agreeing to terms of service.

Why GitHub? ▾ Enterprise Explore ▾ Marketplace Pricing ▾ Search GitHub / Sign in Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.

Username
Pick a username

Email
you@example.com

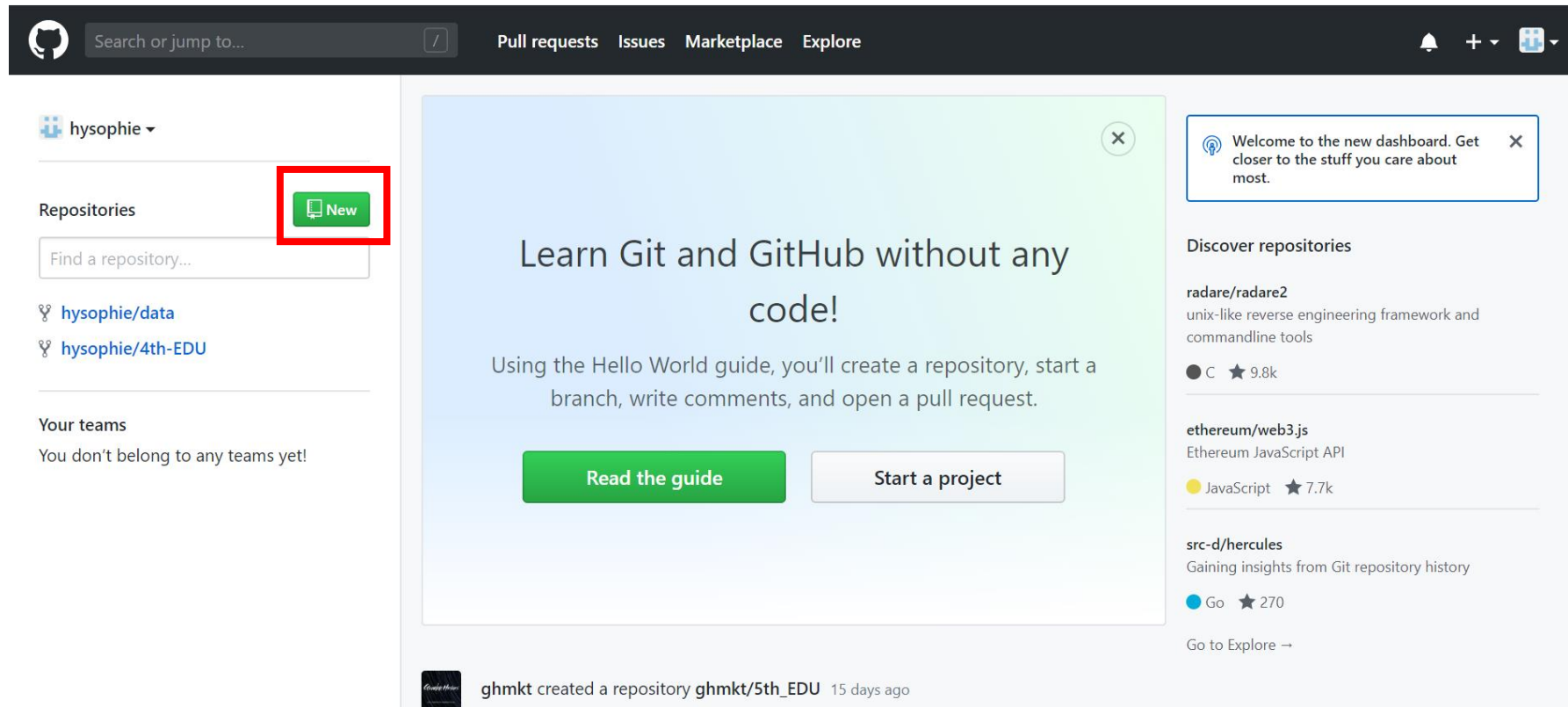
Password
Create a password

Make sure it's **more than 15 characters** OR **at least 8 characters** including a number and a lowercase letter. [Learn more.](#)

Sign up for GitHub

By clicking "Sign up for GitHub" you agree to our [terms of service](#) and

GitHub repository 만들기




GitHub repository 만들기

Create a new repository

A repository contains all project files, including the revision history.

Owner

 hysophie ▾

Repository name *

/ study ✓

Great repository names are short and memorable. Need inspiration? How about **cautious-fiesta**?

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☒ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾



Create repository

- README로 Initialize하기에 체크 눌러주세요

GitHub repository 만들기 (완성~)

The screenshot shows the GitHub interface for a repository named 'study' by user 'hysophie'. At the top, the repository name and user are displayed, along with buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. Below this is a navigation bar with tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. The 'Code' tab is selected. A message states 'No description, website, or topics provided.' with an 'Edit' button. Below this, statistics show '1 commit', '1 branch', '0 releases', and '1 contributor'. A row of buttons includes 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find File', and 'Clone or download'. The commit history shows 'hysophie Initial commit' as the latest commit. Below the history, a file named 'README.md' is listed as the 'Initial commit'. The file content is displayed in a large text area, showing the word 'study'.

hysophie / study

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. Edit

Manage topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find File Clone or download

hysophie Initial commit Latest commit 3821284 just now

README.md Initial commit just now

README.md

study

원격저장소를 로컬저장소로 복사 (git clone)

The screenshot shows the GitHub interface for a repository named 'study' by user 'hysophie'. At the top, there are buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. Below these are tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. The 'Code' tab is selected. A message states 'No description, website, or topics provided.' with an 'Edit' button. Below this, statistics show '1 commit', '1 branch', '0 releases', and '1 contributor'. A 'Branch: master' dropdown and a 'New pull request' button are visible. A row of buttons includes 'Create new file', 'Upload files', 'Find File', and a highlighted 'Clone or download' button. A dropdown menu is open from the 'Clone or download' button, showing 'Clone with HTTPS' (selected), 'Use SSH', and the repository URL 'https://github.com/hysophie/study.git'. A red box highlights the 'Clone or download' button, and another red box highlights the 'Copy to clipboard' icon next to the URL. Below the URL are buttons for 'Open in Desktop' and 'Download ZIP'. The repository content shows an 'Initial commit' with a 'README.md' file. The file content displays the word 'study'.

원격저장소를 로컬저장소로 복사 (git clone)

```
renoi@LAPTOP-F751N51M MINGW64 ~  
$ mkdir github_tutorial
```

```
renoi@LAPTOP-F751N51M MINGW64 ~  
$ cd github_tutorial
```

mkdir

: 디렉토리 생성

cd

: 디렉토리로 이동

원격저장소를 로컬저장소로 복사 (git clone)

```
renoi@LAPTOP-F751N51M MINGW64 ~/github_tutorial  
$ git clone https://github.com/hysophie/study.git
```

git clone

: 원격저장소의 모든 내용을 로컬저장소로 복사하기

CTRL + V 못 씁니다! 오른쪽 마우스 누르세요

* 이전에 있던 파일 + 복사해서 늘어납니다

- 저장소를 연결하는 것 X
- 파일을 받아온다 정도의 복사를 의미함

Mission3 성공!

- (Mission3) GitHub에 원격저장소를 만들고, 원격저장소를 로컬저장소에 가져오자!

```
renoi@LAPTOP-F751N51M MINGW64 ~/github_tutorial
$ cd study

renoi@LAPTOP-F751N51M MINGW64 ~/github_tutorial/study (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

9/21/2019

3. GITHUB – 원격저장소 와 로컬저장소 연결

(Mission4) GitHub에 원격저장소를 만들고, 원격저장소와 로컬저장소를 연결하자!


87

Github repository 만들기 [아까 했다]

Create a new repository

A repository contains all project files, including the revision history.

Owner

 hysophie ▾

Repository name *

/ command_hello ✓

Great repository names are short and memorable. Need inspiration? How about [fictional-invention?](#)

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

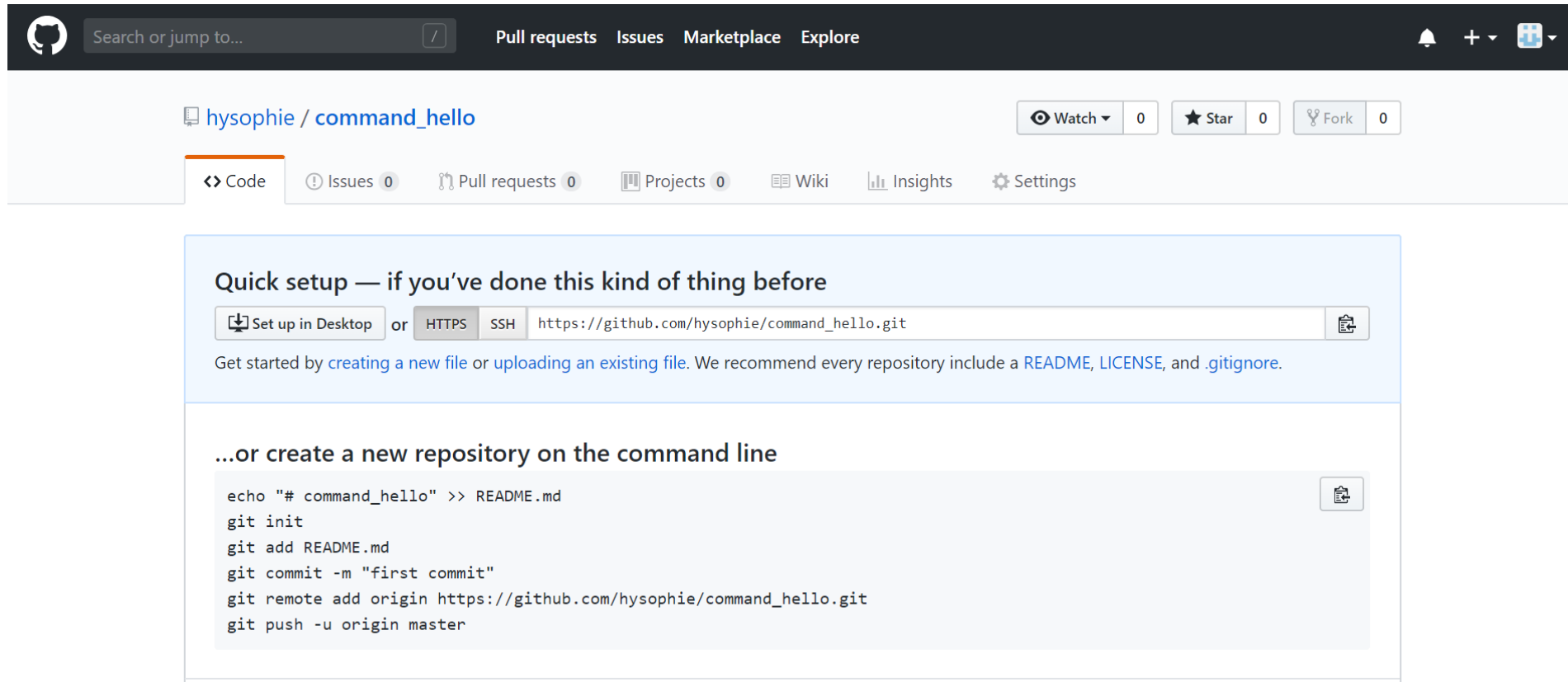
Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

Github repository 만들기 [아까 했다]



The screenshot shows the GitHub interface for a new repository named 'command_hello' by user 'hysophie'. The top navigation bar includes the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. The repository header shows the name 'hysophie / command_hello' and buttons for Watch (0), Star (0), and Fork (0). Below the header, tabs for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings are visible. The main content area features a 'Quick setup' section with options to 'Set up in Desktop' or use 'HTTPS' or 'SSH' with the repository URL 'https://github.com/hysophie/command_hello.git'. A note suggests starting by creating a new file or uploading an existing one, and recommends including a README, LICENSE, and .gitignore. Below this, a section titled '...or create a new repository on the command line' provides a list of terminal commands to initialize and push the repository.

Quick setup — if you've done this kind of thing before

or `https://github.com/hysophie/command_hello.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# command_hello" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/hysophie/command_hello.git
git push -u origin master
```

다른 로컬저장소로 이동하기

```
renoi@LAPTOP-F751N51M MINGW64 ~/github_tutorial/study (master)
$ cd ..

renoi@LAPTOP-F751N51M MINGW64 ~/github_tutorial
$ cd ..

renoi@LAPTOP-F751N51M MINGW64 ~
$ cd git_tutorial
```

cd

: 디렉토리로 이동

cd ..

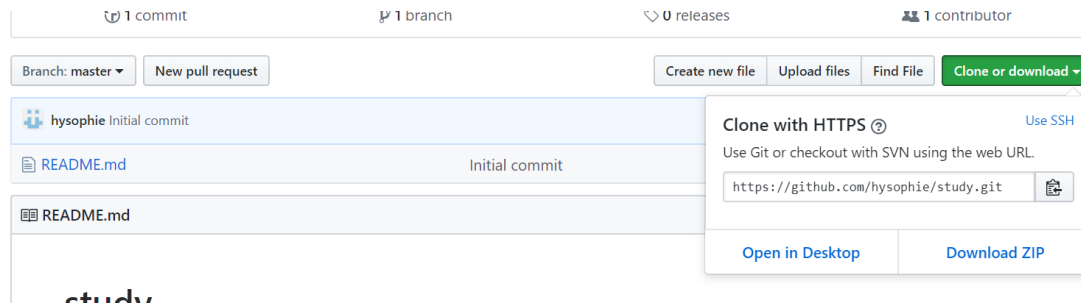
: 한 단계 위의 디렉토리로 이동

로컬저장소를 원격저장소에 연결 (git remote)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git remote add origin https://github.com/hysophie/command_hello.git
```

`git remote add` **원격저장소별칭** `https://github.com/사용자이름/원격저장소이름.git`

➔ 저장소별칭은 어떤 이름이라도 상관 없으나, 관례로 origin을 사용하는 경우 많음.



Mission4 성공!

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git remote -v
origin https://github.com/hysophie/command_hello.git (fetch)
origin https://github.com/hysophie/command_hello.git (push)
```

git remote -v

: 어떤 원격저장소와 연결 되어 있는지 확인

[참고]

명령어 헛갈리면 **git --help**
또는 **git help -a** 또는 **git help -g**

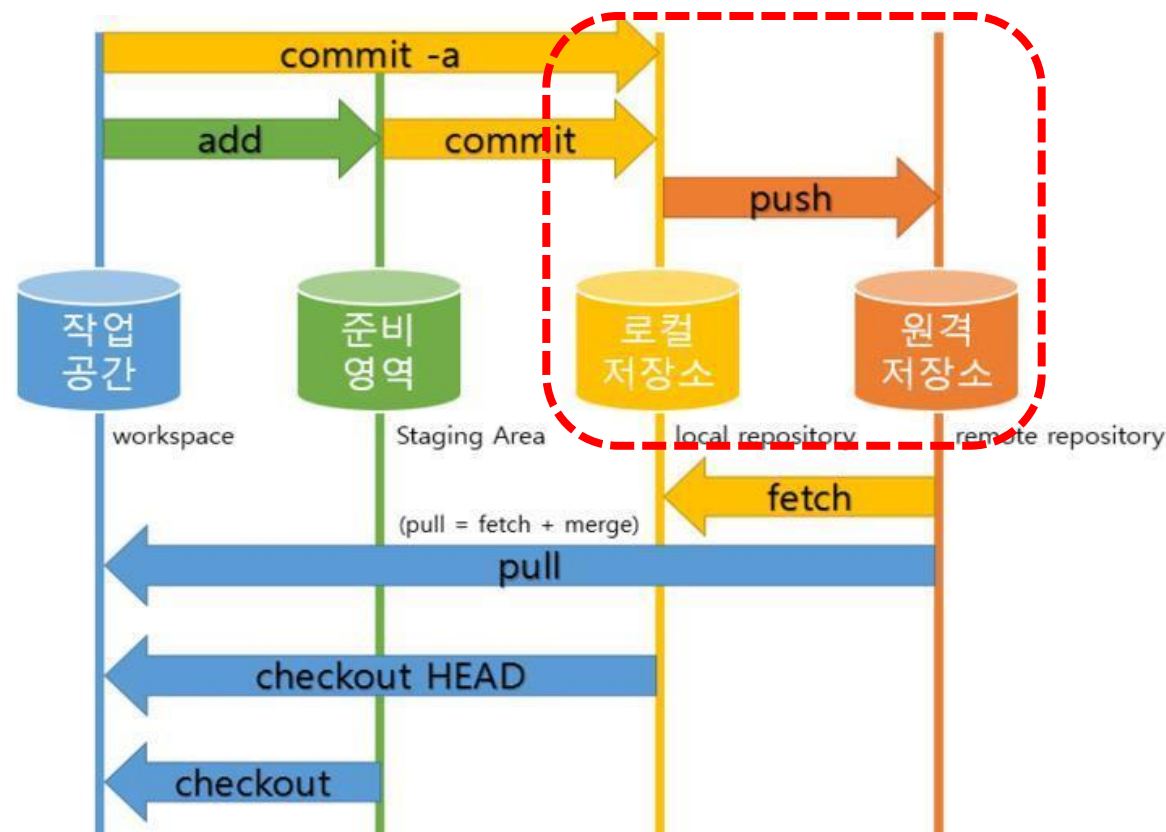
9/21/2019

5) 로컬 파일 변경을 원격저장소로 ★☆

(Mission5) 로컬저장소의 변경사항을 원격저장소로 공유해보자!

93

원격&로컬저장소 간의 변경사항 공유



로컬 작업내역을 원격 저장소에 올리기 (git push)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ git push origin --all
```

git push 원격저장소별칭 로컬브랜치이름

: 로컬브랜치의 내용 or 수정사항을 현재 연결된 원격저장소에 보내는 것

➔ 로컬브랜치이름에 "--all"을 쓰는 것은 로컬의 "모든 브랜치"를 의미

로컬 작업내역을 원격 저장소에 올리기 [참고]

일단 먼저 init 되어 있는 로컬 저장소로 이동

```
cd 로컬저장소 폴더이름
```

원격 저장소에서 모든 파일을 받아온다 = 변화가 있었던 파일을 update 한다

```
git pull 원격저장소별칭 로컬브랜치이름
```

Working directory(실제 폴더) 의 실제 파일에 변화를 주고 그대로 저장한다.

```
git add 변경한 파일 : 변경된 파일 하나만 골라서 제출  
하고 커밋하는 방식
```

바꾼 파일을 add - commit - push

```
git commit 변경한 파일 또는 git commit -m '메모'  
Git push 원격저장소별칭 로컬브랜치이름
```


로컬 작업내역을 원격 저장소에 올리기 (git push)

 GitHub Login



GitHub
Login



Login



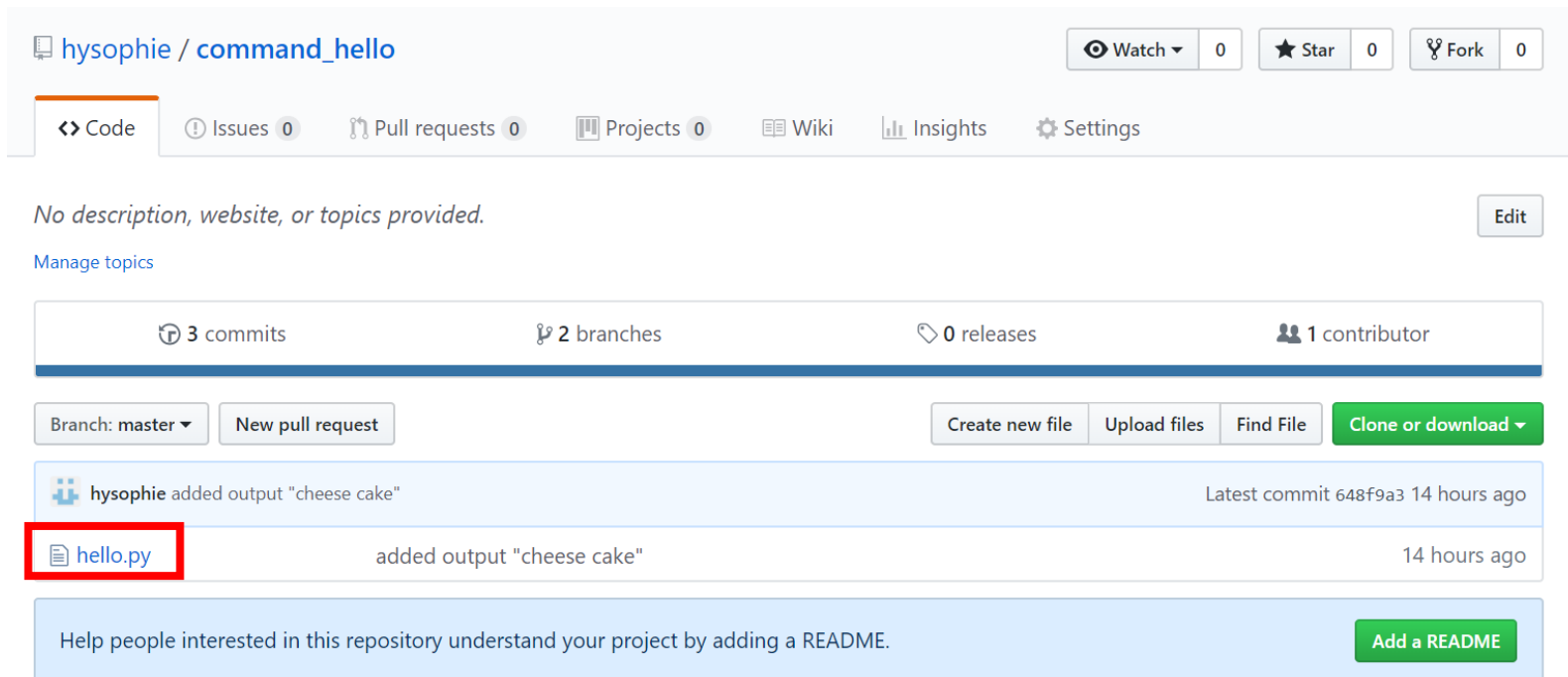
Cancel

Don't have an account? [Sign up](#)

[Forgot your password?](#)

Mission5 성공!

- (Mission5) 로컬저장소의 변경사항을 원격저장소로 공유해보자! (협업)



Mission5 완료!

hysophie / command_hello

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Overview Yours Active Stale All branches Search branches...

Default branch

master	Updated 14 hours ago by hysophie	Default	Change default branch
--------	----------------------------------	---------	-----------------------

Your branches

hotfix	Updated 14 hours ago by hysophie	1 0	New pull request
--------	----------------------------------	-------	------------------

Active branches

hotfix	Updated 14 hours ago by hysophie	1 0	New pull request
--------	----------------------------------	-------	------------------

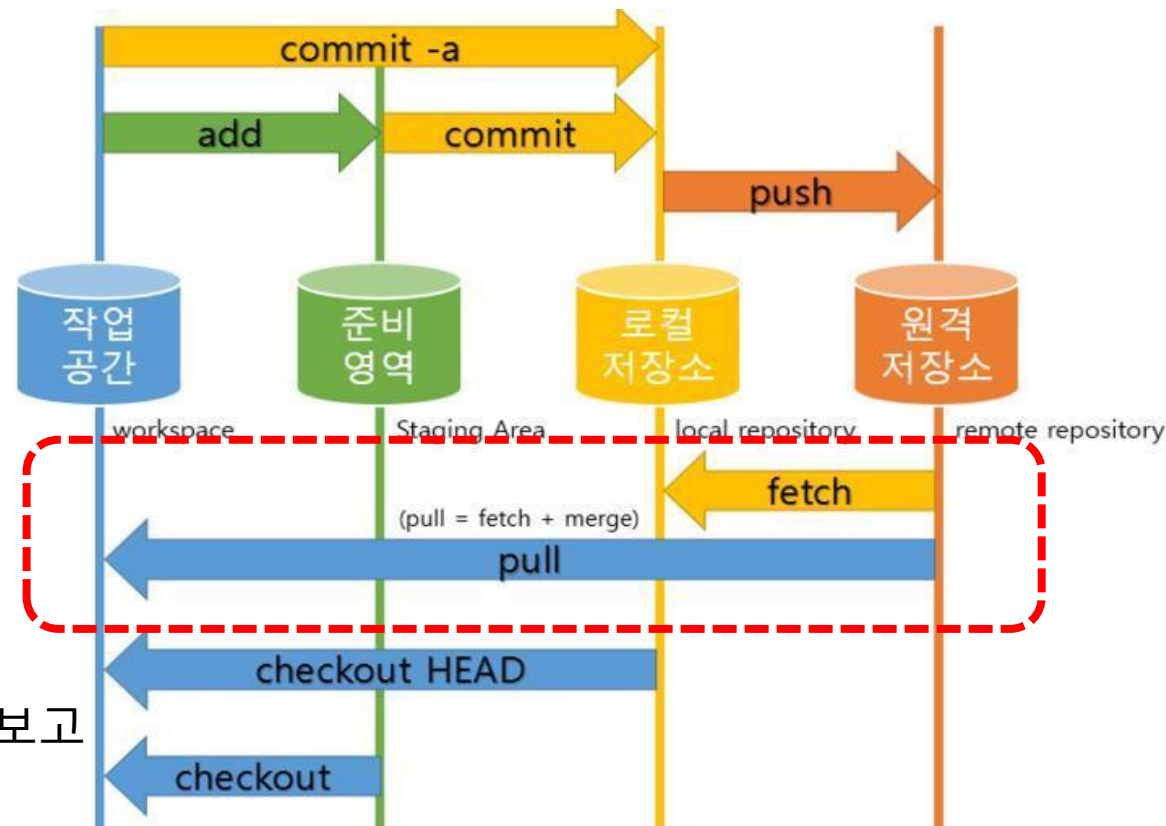
9/21/2019

6) 원격저장소 파일 변경을 로컬 저장소로

(Mission6) 원격저장소의 변경사항을 로컬저장소에 가져오자!

100

원격&로컬저장소 간의 변경사항 공유



원격저장소에서 변경, 커밋해보고
그걸 로컬로 가져옵니다
그 방식은 fetch or pull

git pull vs. git fetch

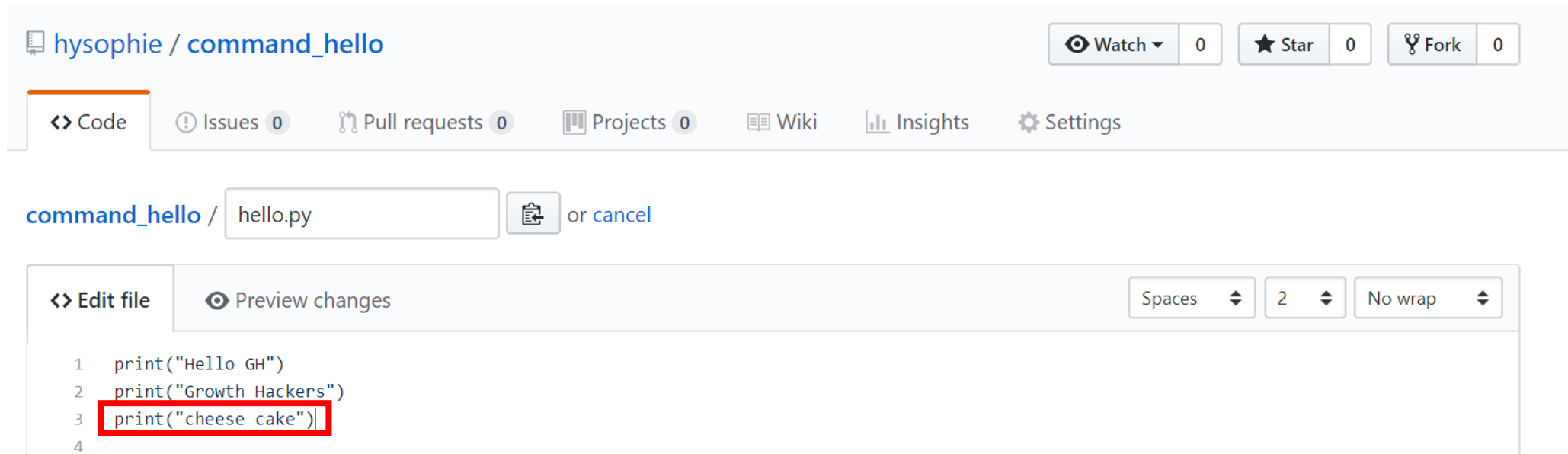
Git pull = git fetch + git merge

- ➔ Git fetch는 원격저장소의 커밋들을 로컬저장소로 가져오는 것
- ➔ Git pull은 원격저장소의 커밋들을 로컬저장소로 가져오는(git fetch) 동시에, 로컬 브랜치에 병합(git merge)까지 시키는 것

Git pull의 장단점

- ➔ 장점: git fetch와 merge를 한꺼번에 해주는 편리함 (그래서 많이 사용됨)
- ➔ 단점: fetch와 merge를 함께 하다 보니 변경사항을 세세하게 파악하기 어렵다

먼저, 원격저장소에서 commit해보기

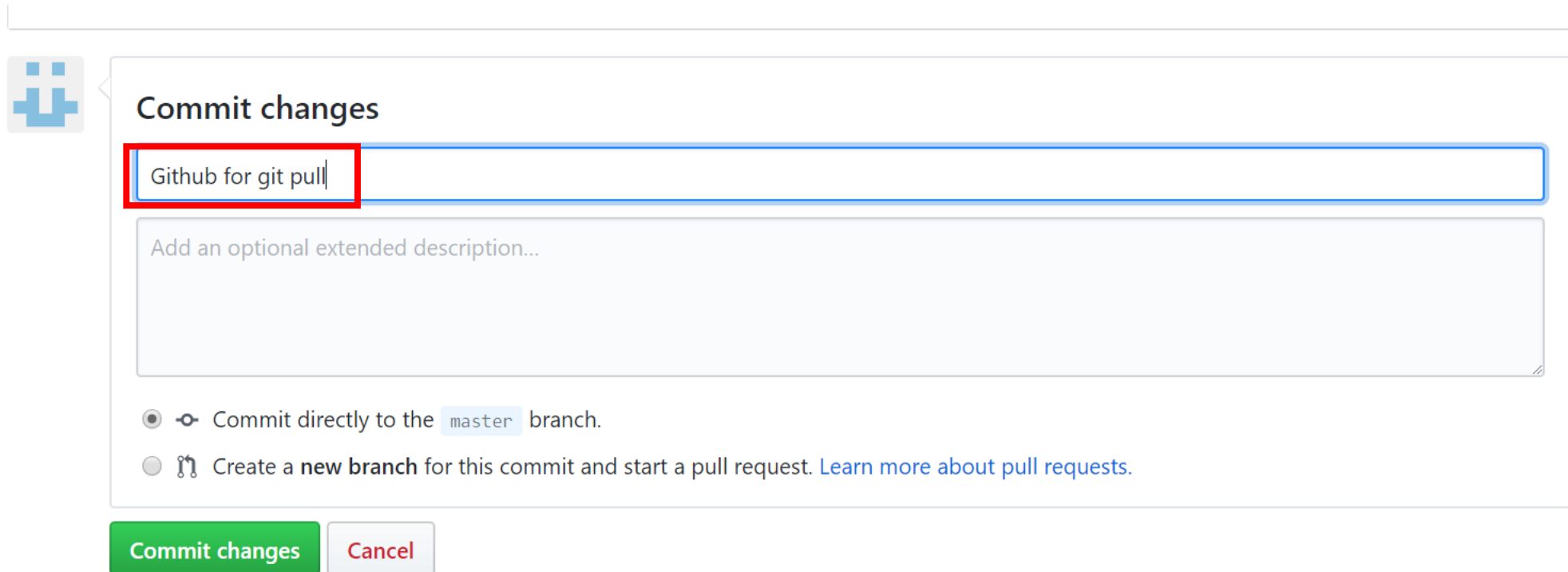


The screenshot shows the GitHub interface for a repository named 'command_hello' by user 'hysophie'. The repository has 0 Watchers, 0 Stars, and 0 Forks. The 'Code' tab is selected, showing the file 'hello.py'. The file content is as follows:


```
1 print("Hello GH")
2 print("Growth Hackers")
3 print("cheese cake")
4
```

The third line of code, `print("cheese cake")`, is highlighted with a red box. The interface also shows options to 'Preview changes', 'Spaces' (set to 2), and 'No wrap'.

먼저, 원격저장소에서 commit해보기



The image shows a Git commit dialog box. On the left is a small icon of a person with a plus sign. The main area is titled "Commit changes". Below the title is a text input field containing "Github for git pull", which is highlighted with a red rectangular border. Underneath the input field is a larger text area with the placeholder text "Add an optional extended description...". At the bottom of the dialog, there are two radio button options: the first is selected and says "Commit directly to the master branch.", and the second is unselected and says "Create a new branch for this commit and start a pull request. Learn more about pull requests.". At the very bottom are two buttons: a green "Commit changes" button and a grey "Cancel" button.

 **Commit changes**

Github for git pull

Add an optional extended description...

☒ Commit directly to the master branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel

그 후 Git pull

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ git pull origin master
```

git pull 원격저장소별칭 로컬브랜치이름

: 원격저장소의 내용을 로컬저장소로 불러오고, 로컬브랜치에 병합
(git fetch + git merge)

Mission6 성공!

- (Mission6) 원격저장소의 변경사항을 로컬저장소에 가져오자!

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master
$ ls
hello.py

renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master
$ cat hello.py
print("Hello GH")
print("Growth Hackers")
print("cheese cake")
```

★Quest★ 퀘스트 제출 방법 익히기

- Jupyter notebook에서 "Hello, I'm 본인이름"을 출력하는 코드 작성하고, 이를 .ipynb 파일로 저장하기
➔파일 이름: "본인이름_Quest1"
- Github에 퀘스트를 제출할 repository 만들기
➔Repository 이름: "본인이름_GH_Quest"
- Jupyter notebook에서 실습한 파일을 git을 통해 github repository에 push하기

참고자료

- 만들면서 배우는 Git, GitHub 입문 (윤웅식)
- 생활코딩 “지옥에서 온 git” 강의
(링크: <https://www.youtube.com/watch?v=hFJZwOfme6w&list=PLuHgQVnccGMA8iwZwrGyNXCGy2LAAsTXk>)
- 시각 자료들은 각 슬라이드에 링크 첨부

9/21/2019

고생하셨습니다!

109


9/21/2019




부록

Git fetch의 사용법 & 충돌 해결

110


원격저장소에서 파일 수정

 [hysophie](#) / [command_hello](#)




 Watch ▾ 0  Star 0  Fork 0

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Insights](#) [Settings](#)

Branch: master ▾ [command_hello](#) / [hello.py](#) [Find file](#) [Copy path](#)

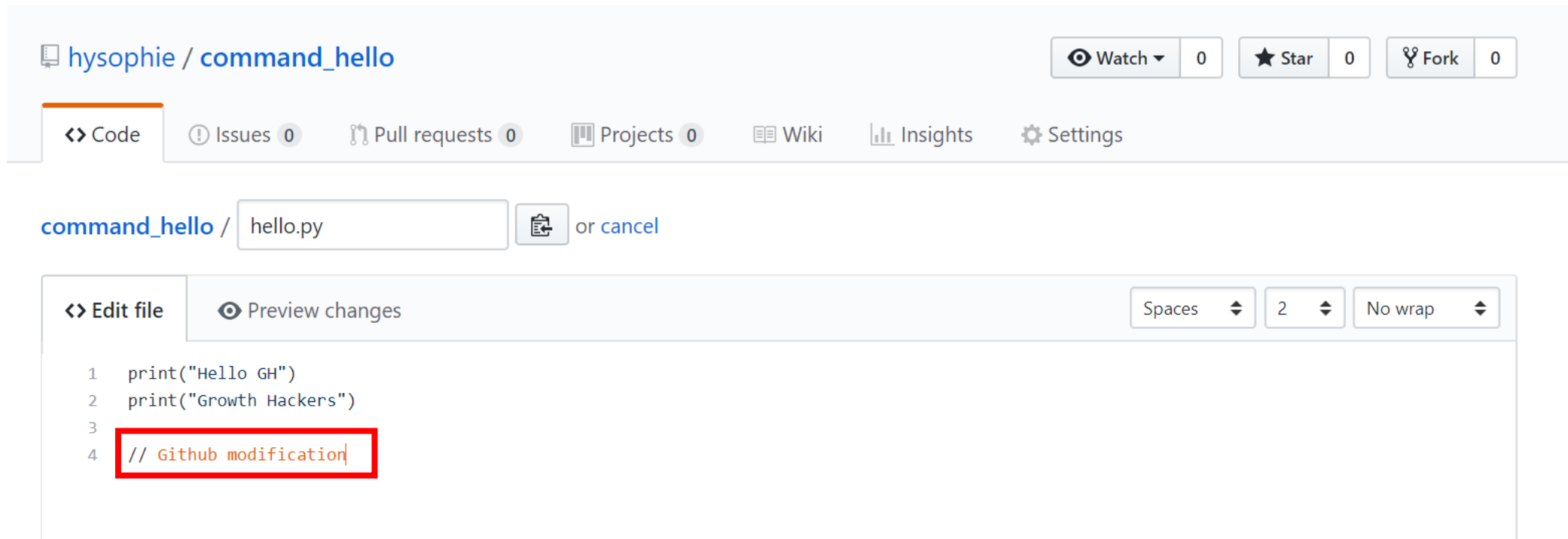
 [hysophie](#) Update hello.py 1c9265c 10 seconds ago

1 contributor

3 lines (2 sloc) 42 Bytes [Raw](#) [Blame](#) [History](#)   

```
1 print("Hello GH")
2 print("Growth Hackers")
```

원격저장소에서 파일 수정




The screenshot shows the GitHub interface for a repository named 'command_hello' by user 'hysophie'. The repository has 0 Watchers, 0 Stars, and 0 Forks. The 'Code' tab is selected, showing a file named 'hello.py'. The file content is as follows:

```
1 print("Hello GH")
2 print("Growth Hackers")
3
4 // Github modification
```

The fourth line of code, `// Github modification`, is highlighted with a red rectangular box. The interface also shows options to 'Edit file' or 'Preview changes', and settings for 'Spaces' (2) and 'No wrap'.

원격저장소에서 파일 수정



Commit changes

hello.py modified on GitHub

Add an optional extended description...

☒ Commit directly to the `master` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

원격저장소에서 파일 수정 (완료~)

The screenshot shows the GitHub interface for the repository 'hysophie / command_hello'. At the top, there are buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. Below these are tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. The 'Code' tab is selected. Below the tabs, there is a dropdown for 'Branch: master' and a link to 'command_hello / hello.py'. To the right are buttons for 'Find file' and 'Copy path'. The main content area shows a commit by 'hysophie' titled 'Update hello.py' with commit hash '95b8306' and the text 'just now'. Below the commit information, it says '1 contributor'. The file 'hello.py' is shown with a size of '66 Bytes' and '5 lines (3 sloc)'. There are buttons for 'Raw', 'Blame', and 'History'. The code content is as follows:

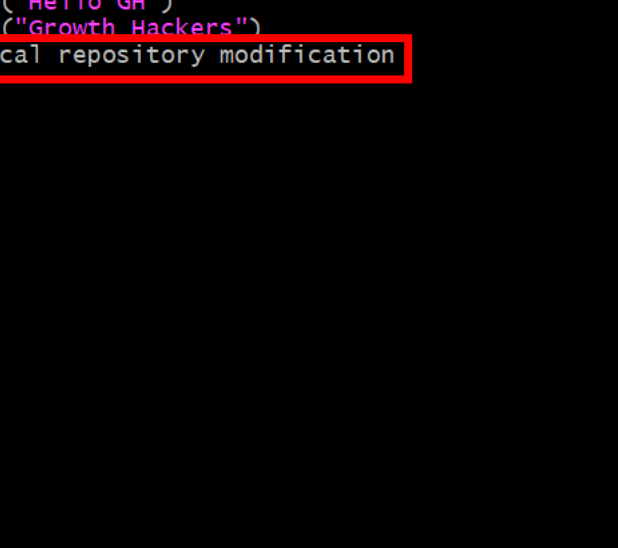
```
1 print("Hello GH")
2 print("Growth Hackers")
3
4 // Github modification
```

The line '4 // Github modification' is highlighted with a red box.

로컬저장소에서 파일 수정

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ vim hello.py
```

로컬저장소에서 파일 수정



The screenshot shows a Windows command prompt window with the title bar "MINGW64:/c/Users/renoi/git_tutorial". The prompt is at a Python shell, indicated by "hello.py[+] [dos]". The user has entered the following commands:

```
print("Hello GH")
print("Growth Hackers")
// Local repository modification
```

The output of the Python script is visible on the left side of the window, showing a series of tilde characters (~) representing the output of the print statements.

- Vim 사용법

- Vim에서 작성을 시작하려면 키보드의 [i]키를 눌러 작성모드로 바꾸기
- 작성 후에 [ESC]키를 누르면 일반모드로 돌아옴
- :를 입력하면 명령모드로 바뀌고, 명령모드에서 wq를 입력하면 저장(w) 후 종료(q)됨

로컬저장소에서 파일 수정

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ git commit -a
```

git commit -a

: 변경된 로컬저장소 파일을 모두 로컬저장소에 제출(커밋)

Git push → BUT 실패

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git push origin master
To https://github.com/hysophie/command_hello.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/hysophie/command_hello.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'note about fast-forwards' in 'git push --help' for details.
```

왜 실패?

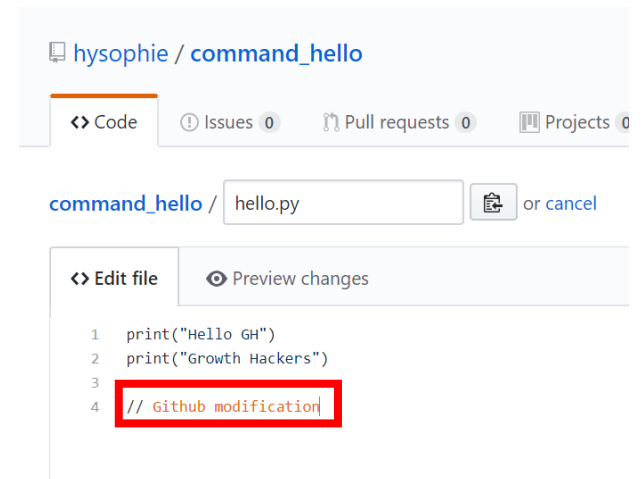
- 로컬저장소의 hello.py와 원격저장소의 hello.py의 내용이 다르기 때문
- Push하려는 원격저장소에 같은 이름의 브랜치가 있는데 서로의 내용이 다르면, push가 거부됨

git fetch & git merge로 해결하기!

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git fetch
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 11 (delta 1), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (11/11), done.
From https://github.com/hysophie/command_hello
   3b4ae5e..e93f43a  master    -> origin/master
```

git fetch

: 원격저장소의 커밋들을 로컬저장소로 가져옴



git fetch & git merge로 해결하기!

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)
$ git branch -a
hotfix
* master
remotes/origin/hotfix
remotes/origin/master
```

git branch -a

: 모든 로컬저장소, 원격저장소의 브랜치 정보를 볼 수 있음

Git merge → BUT 충돌(conflict) 발생!

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ git merge origin/master
```



```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ git merge origin/master  
Auto-merging hello.py  
CONFLICT (content): Merge conflict in hello.py  
Automatic merge failed; fix conflicts and then commit the result.
```

충돌 해결 (git diff)

```
reno@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master | MERGING)
$ git diff
diff --cc hello.py
index 9651424,88b4c95..0000000
--- a/hello.py
+++ b/hello.py
@@ -1,3 -1,4 +1,8 @@
    print("Hello GH")
    print("Growth Hackers")
++<<<<<< HEAD
+// Local repository modification
+=====
+
+ // Github modificationn
++>>>>>> origin/master
```

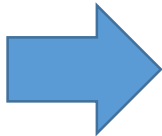
hello.py의 오류내용

git diff

: 로컬저장소의 브랜치와
원격저장소의 브랜치 사이에
어떤 차이점이 있는지 확인

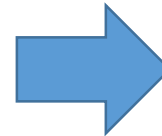
충돌 해결 (vim)

```
renoi@LAPTOP-F75  
$ vim hello.py
```



```
MINGW64:/c/Users/renoi/git_tutorial  
print("Hello GH")  
print("Growth Hackers")  
<<<<<< HEAD  
// Local repository modification  
=====  
  
// Github modificationnn  
>>>>>> origin/master  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
hello.py [dos] (23:37 20/03/2019)  
"hello.py" [⚡ ⏎] 8L, 151C
```

해결 전

A terminal window titled "MINGW64:/c/Users/renoi/git_tutorial". The prompt character is "~>". The user has entered several commands:

```
print("Hello GH")  
print("Growth Hackers")  
// Local repository modification  
// Github modification
```

The last two lines are enclosed in a red rectangular box. Below the box, there are approximately ten more lines starting with "~>", each followed by a blank space.

hello.py[+] [dos] (23:37 20/03/2019)
:wq|

해결 후

충돌 해결 (git commit -a)

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master|MERGING)  
$ git commit -a
```

git commit -a
: 변경된 저장소 파일을 모두 저장소에 제출(커밋)

커밋 메시지 작성

```
MINGW64:/c/Users/renoi/git_tutorial
conflict resolved GitHub
# Conflicts:
#   hello.py
#
# It looks like you may be committing a merge.
# If this is not correct, please remove the file
#   .git/MERGE_HEAD
# and try again.
#
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# All conflicts fixed but you are still merging.
#
# Changes to be committed:
#   modified:   hello.py
#
~
~
~
C:/Users/renoi/git_tutorial/.git/COMMIT_EDITMSG[+] [unix] (23:40 20/0
:Wq|
```

충돌 해결 후 git push 시도

```
renoi@LAPTOP-F751N51M MINGW64 ~/git_tutorial (master)  
$ git push origin master
```

Git push (성공~)

The screenshot shows a GitHub repository page for 'command_hello' by user 'hysophie'. The repository has 0 Watchers, 0 Stars, and 0 Forks. The 'Code' tab is selected, showing the file 'hello.py' on the 'master' branch. The file was modified 10 minutes ago by 'hysophie' with commit 'e93f43a'. The file size is 100 Bytes and it contains 6 lines (4 sloc). The code content is as follows:

```
1 print("Hello GH")
2 print("Growth Hackers")
3
4 // Local repository modification
5 // Github modification
```