Dscover 2차 세미나

# Git & GitHub

# Dscover 2021 2학기 2차 교육 세미나

CONTENTS

## Git & GitHub

# 1. Git vs GitHub



✅ 로컬에서 관리되는 버전 관리 시스템

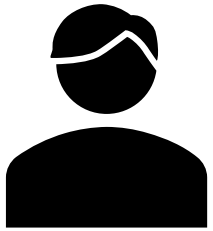
✅ 소스코드를 효율적으로 관리할 수 있게 해주는 도구

✅ 클라우드 방식으로 관리되는 버전 관리 시스템

✅ Git을 업로드 할 수 있는 웹사이트

✅ 개발자들의 버전 제어 및 공동 작업을 위한 플랫폼

# 2. Git & GitHub를 사용하는 이유



" 기말레포트_최종.docx "

" 기말레포트_진짜최종.docx "

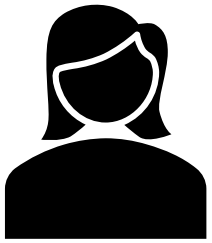
" 기말레포트_진짜진짜최종.docx "

" 기말레포트_진짜진짜찐최종.docx "

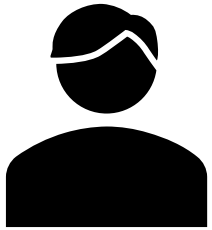
…

“기말레포트_진짜진짜찐찐찐최종_제발최종….docx”

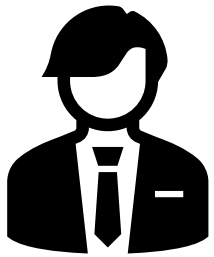
# 2. Git & GitHub를 사용하는 이유



" 조별과제_지인최종.docx "

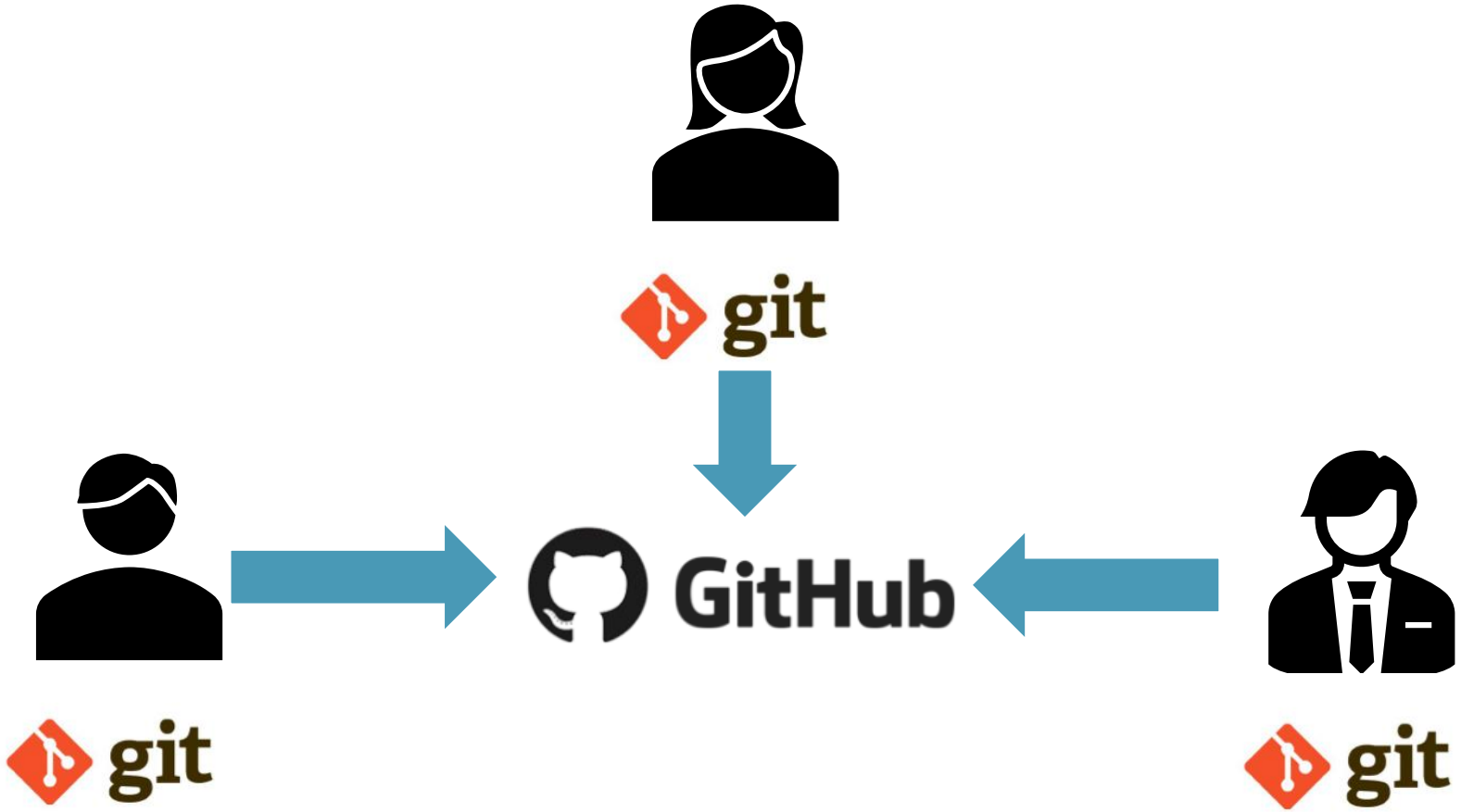


"조별과제_유진최종.docx "



" 조별과제_재현최종.docx "

# 2. Git & GitHub를 사용하는 이유



# 2. Git & GitHub를 사용하는 이유

## 오픈소스란(Open Source Software, OSS)?

공개적으로 액세스 할 수 있게 설계되어 누구나 자유롭게 확인, 수정, 배포할 수 있는 코드!

amazon facebook Google

NAVER SAMSUNG

## 체크섬

Git에서 사용하는 가장 기본적인 (atomic) 데이터 단위!
데이터 저장하기 전 항상 체크섬 구하고 그 체크섬으로 데이터를 관리

Git은 데이터를 **추가**할 뿐, 되돌리거나 삭제할 방법이 없음

## Repository (저장소)

실제 소스코드가 담겨 있으면서 commit 내역 등의 모든 작업 이력이 담겨 있는 공간

- ✓ 원격 저장소: 파일이 원격 저장소 전용 서버에서 관리되며 여러 사람이 공유
- ✓ 로컬 저장소: 개인 PC에 파일이 저장되는 개인 저장소

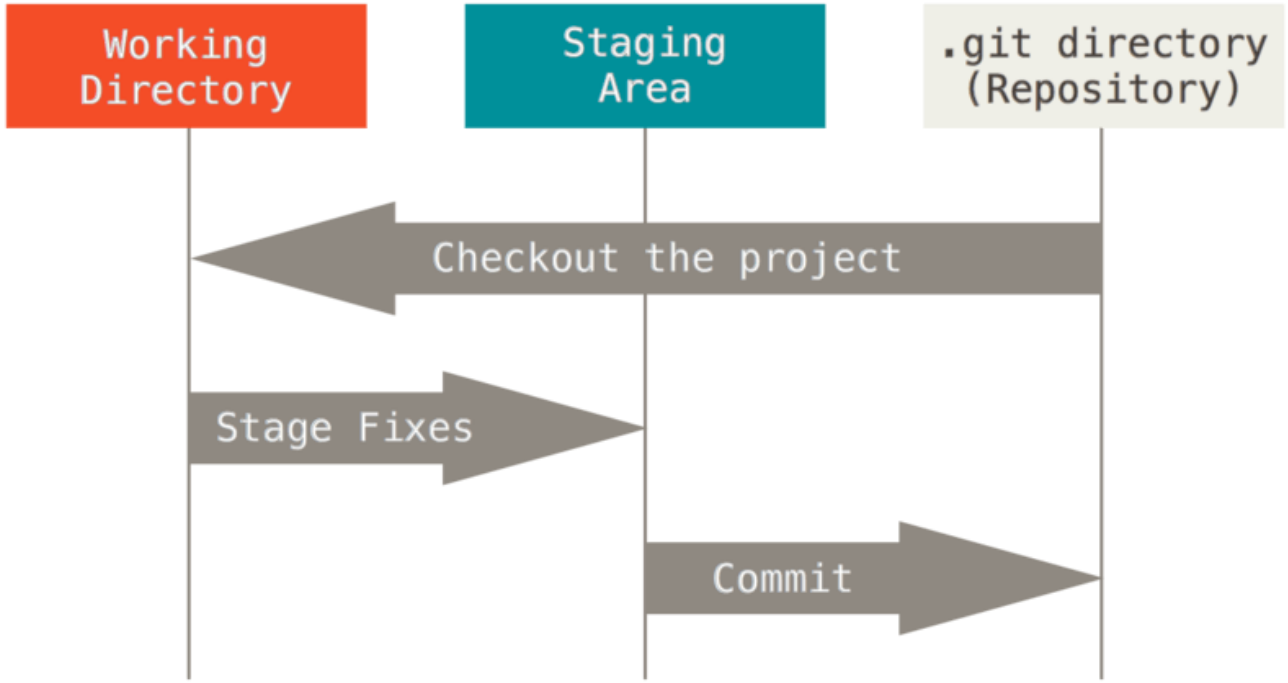
# 3. 관련 용어 및 작동 원리

## Git Status

- ✓ Committed: 데이터가 로컬 데이터베이스에 안전하게 저장됐다는 의미
- ✓ Modified: 수정한 파일을 아직 로컬 데이터베이스에 커밋하지 않은 것을 의미
- ✓ Staged: 현재 수정한 파일을 곧 커밋할 것이라고 표시한 상태를 의미

## 단계

Git 디렉토리에 있는 파일들 → committed 상태
파일 수정하고 staging area에 추가 → staged 상태
체크아웃하고 수정했지만, 아직 staging area에 추가 x → modified 상태



# 3. 관련 용어 및 작동 원리

## Add

작업한 내용을 staging area에 올림 `$ git add 파일명`

## Push

실제로 git에 반영하기 위함

`$ git push –u origin master`: 지역 저장소의 branch를 원격 저장소의 master branch와 연결 (한번만 하면 됨)

`$ git push`: 원격 저장소에 올리기

## Pull

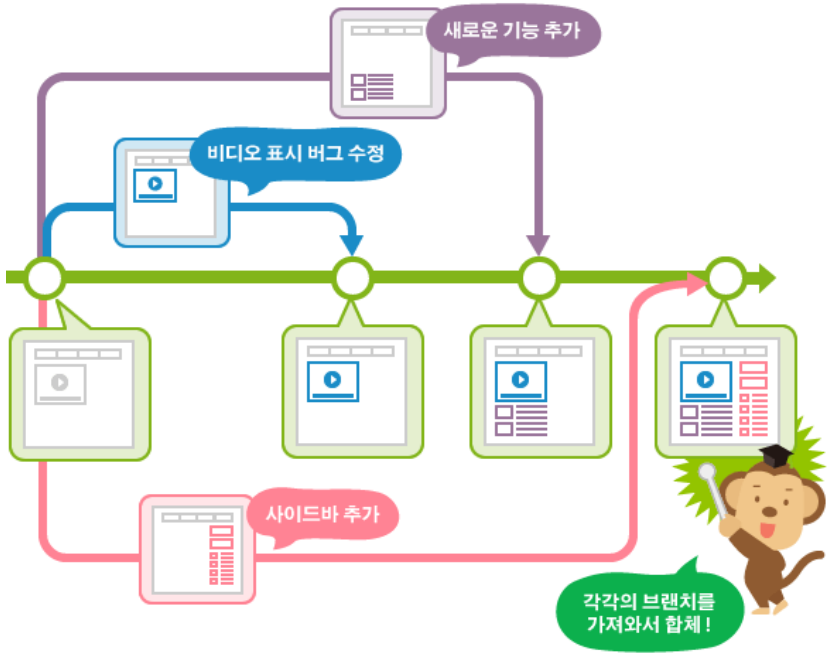
타인이 작업한 내용을 내 컴퓨터로 다운 받음 (`git fetch & git merge`) => `$ git pull`

# 3. 관련 용어 및 작동 원리

## Branch

독립적으로 어떤 작업을 진행하기 위한 개념
각각의 branch는 다른 branch의 영향을 받지 않아 여러 작업 동시에 진행 가능

다른 branch와 merge함으로써 작업한 내용을 다시 새로운 하나의 branch로 모을 수 있음



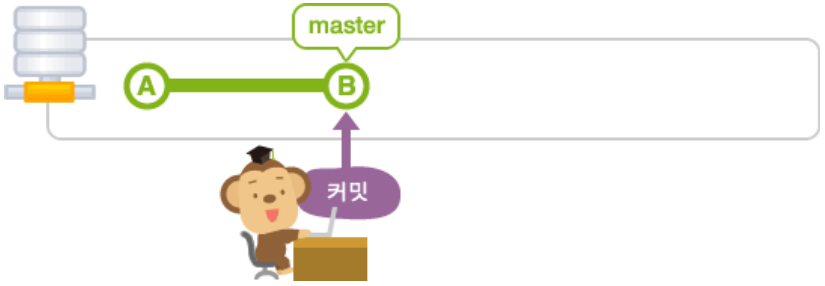

Main branch에서 자신의 작업 전용 branch 생성
각자 작업 진행 후 main branch에 자신의 branch 변경 사항 적용

# 3. 관련 용어 및 작동 원리

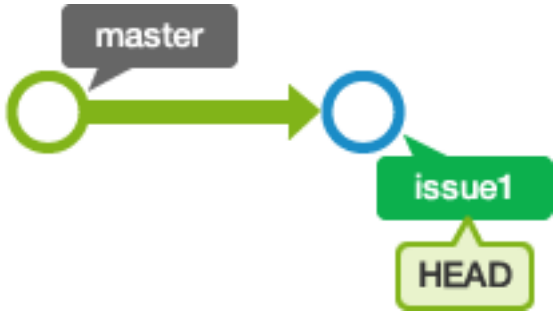
## Branch

✓ Master branch: 저장소 처음 만들 때 git은 'master'라는 이름의 branch 만들어 둠
다른 새로운 branch 만들어서 checkout하지 않는 이상, 모든 작업은 master에서 이루어짐

```
$ git branch 브랜치명 : 브랜치 만들기
$ git checkout 브랜치이름 : '브랜치이름'으로 브랜치 이동
```








## Merge

작업한 branch 내용을 master와 합치기

```
$ git merge 병합할 브랜치명
```

# 3. 관련 용어 및 작동 원리

## README.md

해당 프로그램에 대한 정보 (ex. 프로젝트 설명, 사용법, 저작권 등)를 기입하는 파일

## Markdown 문법

1. 제목(글머리) 작성

```
# 제목
## 부제목
### 소제목
#### 제목4
##### 제목5
###### 제목6
```

2. 리스트 작성

```
* 리스트1
- 리스트2
+ 리스트3
1. 리스트4
2. 리스트5
```

3. 이텔릭체

*텍스트* or _텍스트_

4. 굵은 글씨

**텍스트** or __텍스트__

참고: https://github.com/lsh424/MARKDOWN/blob/master/README.md

# 4. 실습 – git 설치

1. **Git 설치파일 다운로드 [링크]**

   자신의 OS에 맞는 Git 설치버전을 다운

2. **별다른 경우가 아니라면 계속 “Next”**

3. **“Install” 눌러 설치**

4. **설치 끝나면 “Finish”**

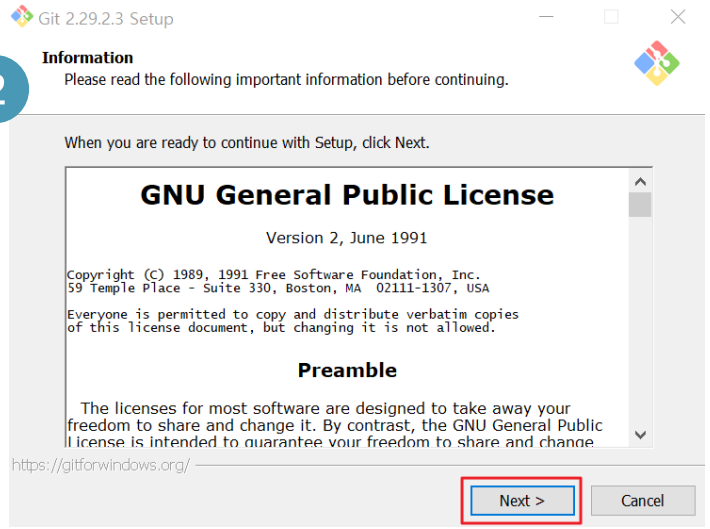
   정상적으로 Git이 설치되었다면 Git Bash가 설치되어 있을 것

참고: https://coding-factory.tistory.com/245

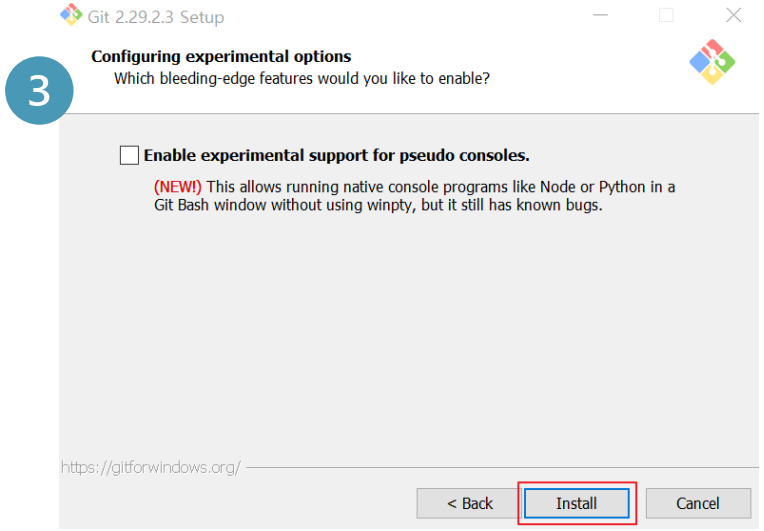
# 4. 실습 – git 설치

1

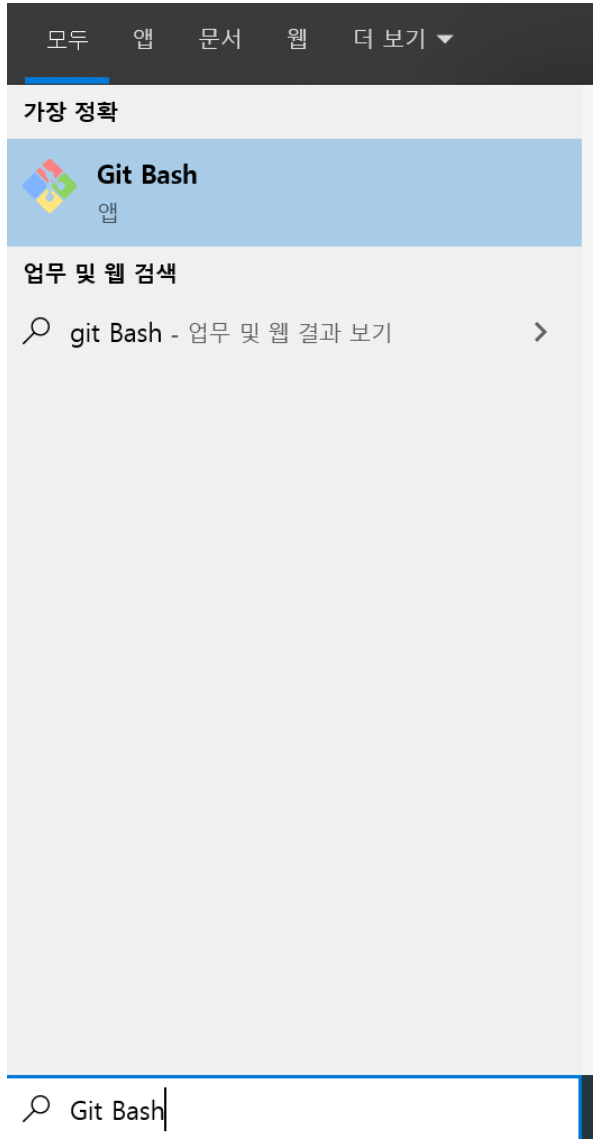

2




3




4

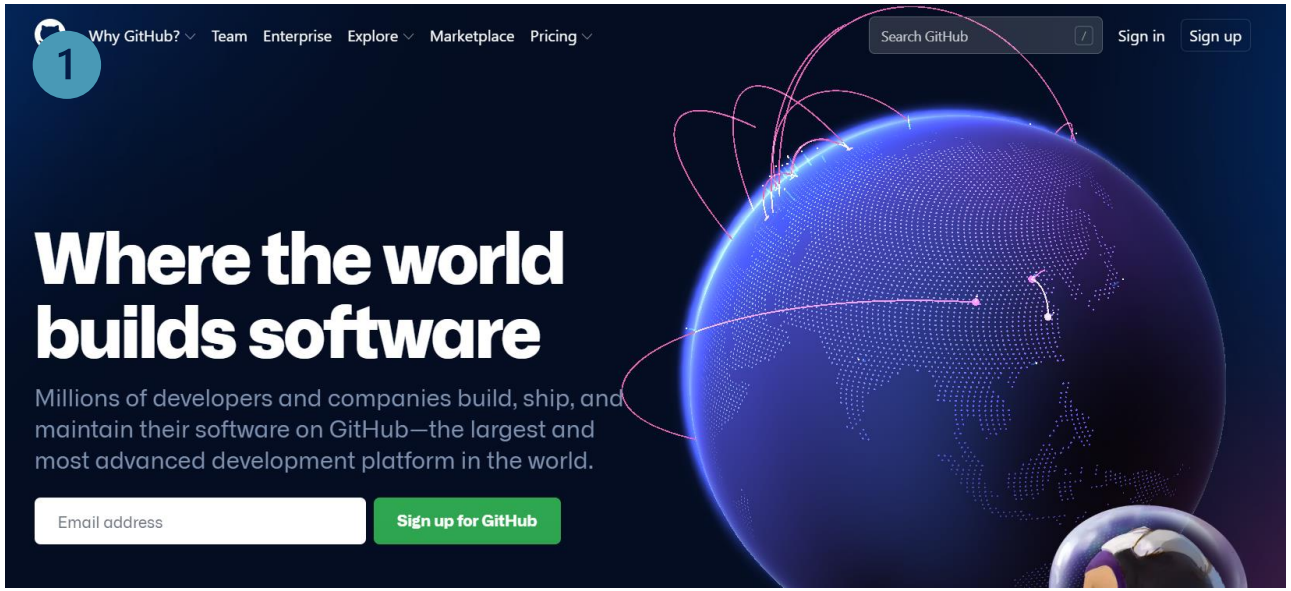


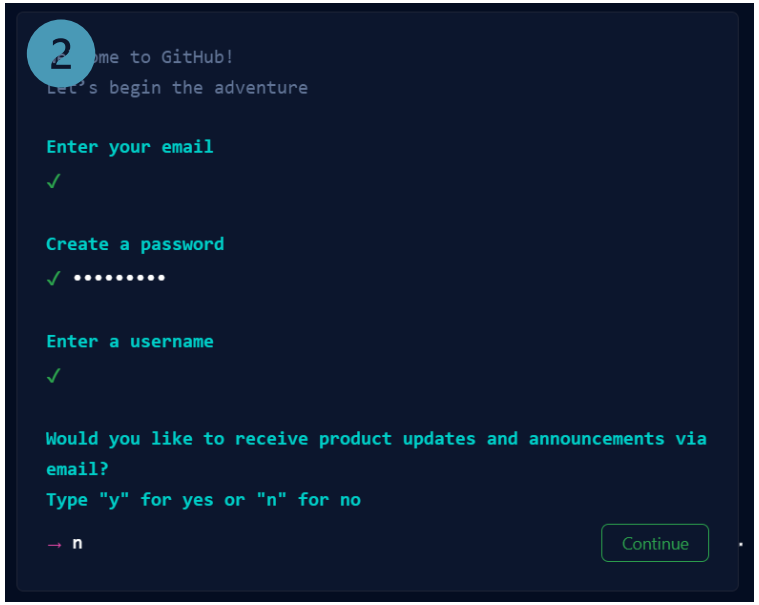
# 4. 실습 – 기본 명령어1 init, config, status, add, commit, log, reset, revert

1. git이 폴더 관리 시작 **git init**
2. 로컬에서 기본값으로 사용할 Git 사용자 이름,이메일 설정 **git config**
3. Git에 의해 관리되는 파일 상태 확인 **git status**
4. 'Working directory' 상의 변경 내용을 'staging area'에 추가 **git add <파일 위치>**
5. Git add로 추가된 파일을 설명과 함께 저장 **git commit –m "<커밋 내용>"**
6. Commit된 내역 확인 **git log**
7. 과거 commit으로 완전히 이동 **git reset <돌아갈 커밋의 log번호 6자리> --hard**
8. 과거 commit을 새로 commit **git revert <취소할 커밋의 log번호 6자리>**

참고: https://www.youtube.com/watch?v=FXDjmsiv8fI

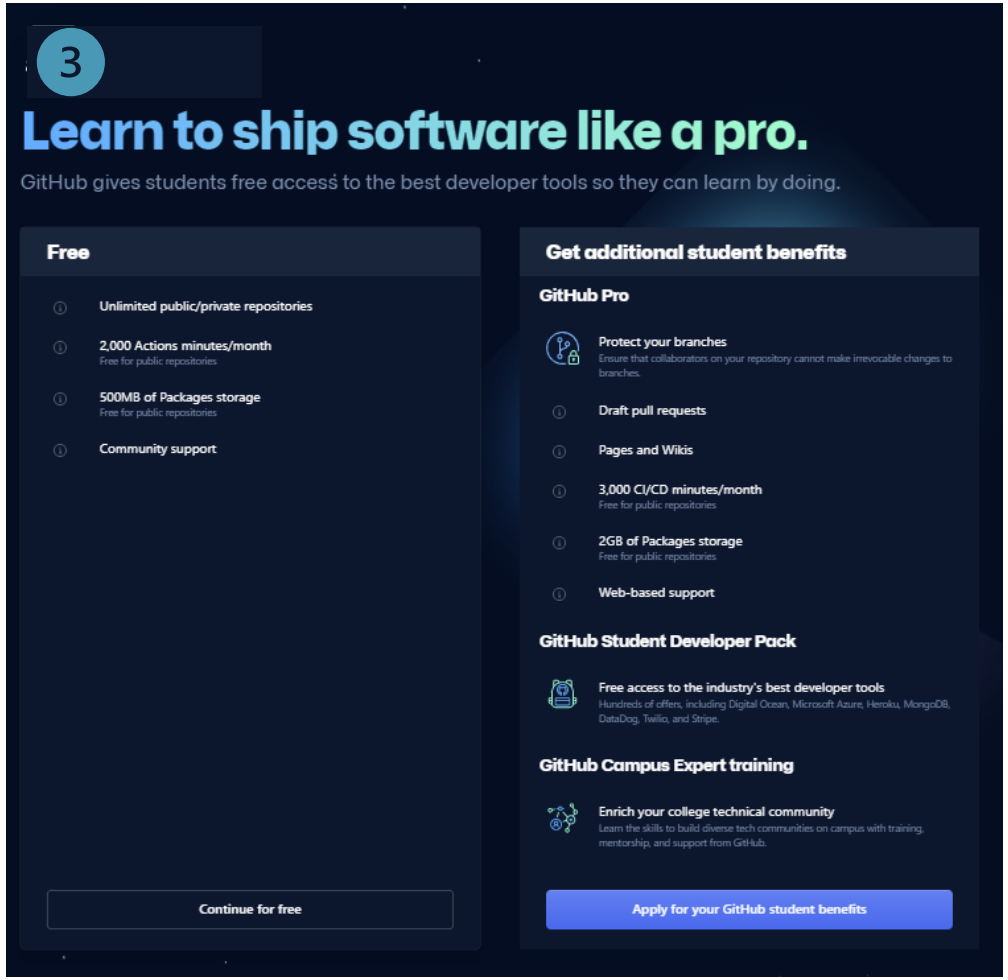
# 4. 실습 – GitHub 회원가입




[Github 홈페이지]

1. GitHub 계정 만들기
2. Create repository
3. url 복사




참고: https://tagilog.tistory.com/377

# 4. 실습 – 기본 명령어2 remote add, push, pull, clone

1. 관리하는 파일이 모두 커밋됐는지 확인 **git status**
2. 현 폴더의 원격 저장소 확인 **git remote**
3. 현 폴더의 원격 저장소 설정 **git remote add origin <원격 저장소 url>**

   아까 github 사이트에서 만든 repository가 현 폴더의 원격 저장소로 설정

   'origin'이라는 이름의 원격 저장소로 설정 (origin 바꿔도 되나 흔히 기본값으로 사용)
4. 현재 커밋된 내용들을 원격 저장소의 브랜치에 업로드 **git push –u origin master**

   원격 저장소 이름 'origin'이라고 가정한 명령어

   브랜치 이름 'master'라고 가정한 명령어

   처음 사용하는 유저라면 유저 이름과 비밀번호를 입력하라고 뜸
5. 원격 저장소의 내용 확인 **git fetch**
6. 원격 저장소의 변경 사항 현 폴더에 병합 **git pull origin master**
7. 다른 원격 저장소를 현재 폴더로 다운 **git clone <원격 저장소 url>**

   가져오려는 GitHub 원격 저장소에서 "Create or download"을 통해 url 복사

참고: https://www.youtube.com/watch?v=GaKjTjwcKQo

# 4. 실습 – branch 명령어 branch, checkout, merge, pull, CONFLICT

1. 로컬 저장소에 새로운 브랜치를 판다 **git branch <새로운 브랜치>**
2. 현재 작업하고 있는 HEAD를 옮긴다 **git checkout <옮길 브랜치>**

   git branch **–b** <브랜치명>을 하면 새로 만들고 바로 브랜치 변환
3. 작업을 수정하고, 로컬 저장소 issue1 브랜치에 커밋까지 완료한다
4. 원격 저장소 origin에도 새로운 브랜치를 만들어 저장한다 **git push origin <새로운 브랜치>**

   같은 방법으로 하나 더 만들어 둔다.

   주 사용 브랜치로 다시 이동한 후,
5. HEAD에서 다른 브랜치를 병합한다 **git merge <병합할 다른 브랜치>**

   만약 내용에 충돌이 없다면 자연스럽게 그 브랜치와 같은 내용을 가리키는 것

   CONFLICT 존재 시, 윗부분(현 위치(HEAD)에서의 내용)과 아랫부분(병합할 브랜치의 내용) 수정 필요.

   그 후 커밋을 통해 로컬 브랜치 병합
6. 원격저장소의 브랜치의 내용을 현재 로컬 HEAD에 불러와 병합 **git pull origin(원격저장소) <병합할 브랜치>**

   pull = fetch + merge

참고: https://backlog.com/git-tutorial/kr/stepup/stepup2_2.html

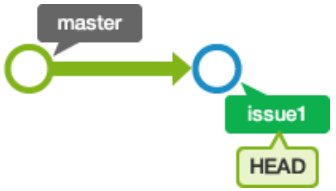
# 4. 실습 – branch 명령어 branch, checkout, merge, pull, CONFLICT

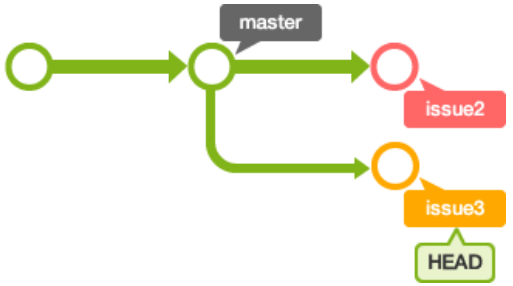
1. 로컬 저장소에 새로운 브랜치를 판다 **git branch <새로운 브랜치>**

2. 현재 작업하고 있는 HEAD를 옮긴다 **git checkout <옮길 브**

   git checkout **–b** <브랜치명>을 하면 새로 만들고 바로 브랜치 변환


3. 작업을 수정하고, 로컬 저장소 issue1 브랜치에 커밋까지 완료




4. 원격 저장소 origin에도 새로운 브랜치를 만들어 저장한다 **git push origin(원격저장소) <새로운 브랜치>**

   같은 방법으로 하나 더 만들어본다

   주 사용 브랜치로 다시 이동한 후 다음 페이지 진행

참고: https://backlog.com/git-tutorial/kr/stepup/stepup2_2.html

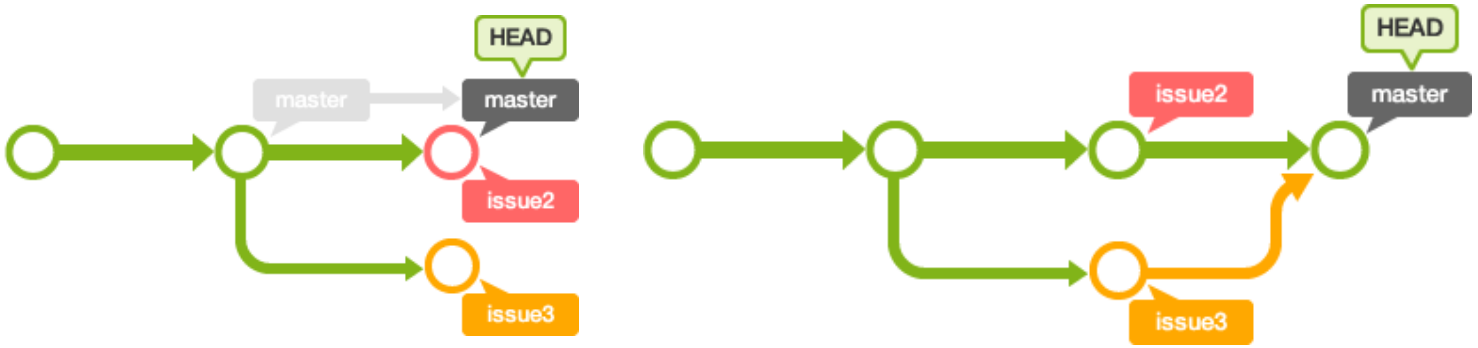
# 4. 실습 – branch 명령어 branch, checkout, merge, pull, CONFLICT

5. HEAD에서 다른 브랜치를 병합 **git merge <병합할 다른 브랜치>**

   만약 내용에 충돌이 없다면 그 브랜치와 HEAD가 동일해짐

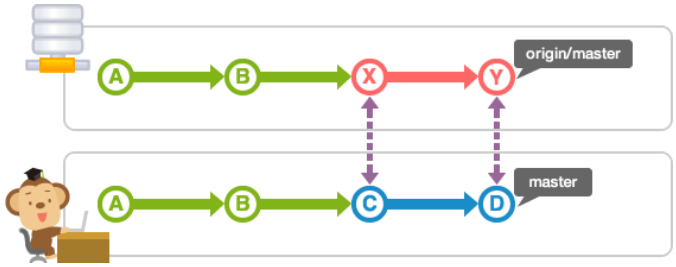
   CONFLICT 존재 시, 윗부분(현 위치(HEAD)에서의 내용)과 아랫부분(병합할 브랜치의 내용) 수정 필요

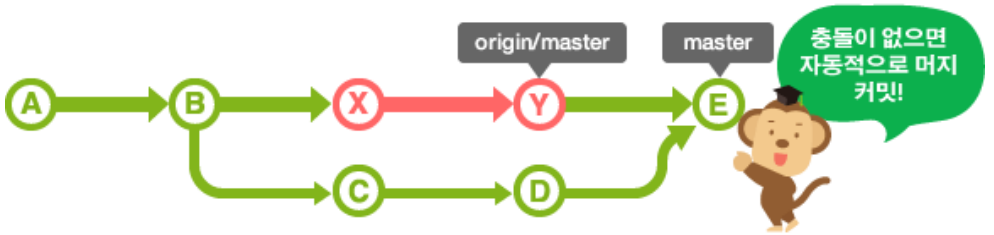
   그 후 커밋을 통해 로컬 브랜치 병합




6. 원격저장소의 브랜치의 내용을 현재 로컬 HEAD에 불러와 병합 **git pull origin <병합할 브랜치>**

   pull = fetch + merge




참고: https://backlog.com/git-tutorial/kr/stepup/stepup2_2.html

Dscover 2차 세미나

# 감사합니다