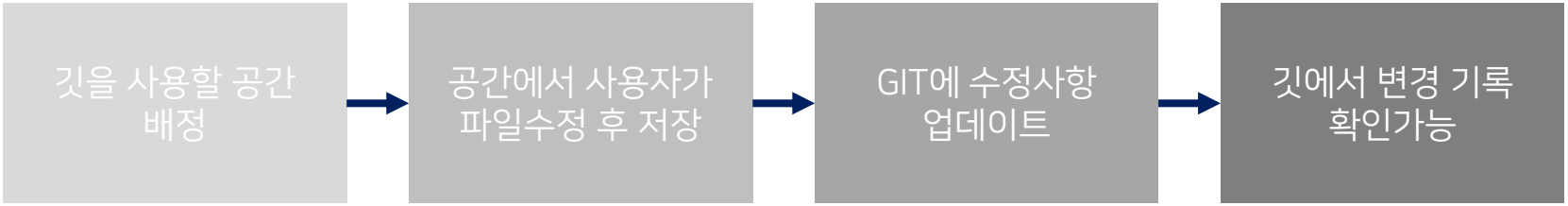
GIT & GIT HUB
VIM

GIT 이란?

- GIT → 버전 관리 툴(???)
  - Index.html,design.css,code.js,readme.md … 너무 많은 파일들
  - 각 파일을 수정하고 수정 기록을 남기는 방법이 없을까..?
  - 그래서 만든게 GIT

- GIT의 기본 Process

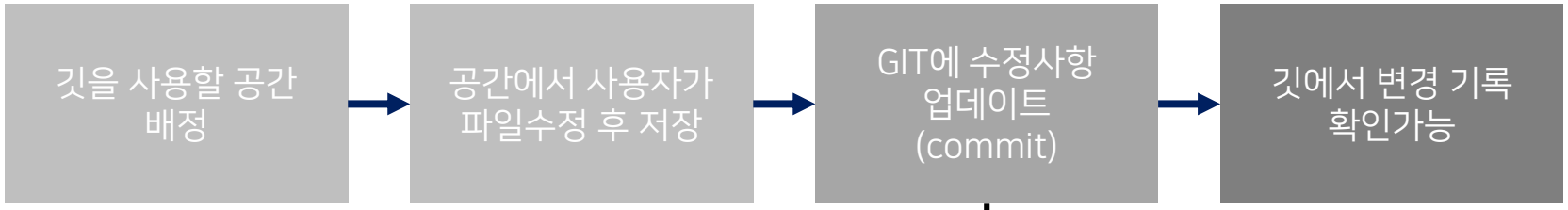


- GIT 의 사용이유!
  - 내가 작업했던, 다른 사람이 작업했던 내역 확인 가능!
  - 어떤게 최종버전인지 어떤게 테스트 버전인지 확인가능!
  - 교수님이 쓰라니 써야지!

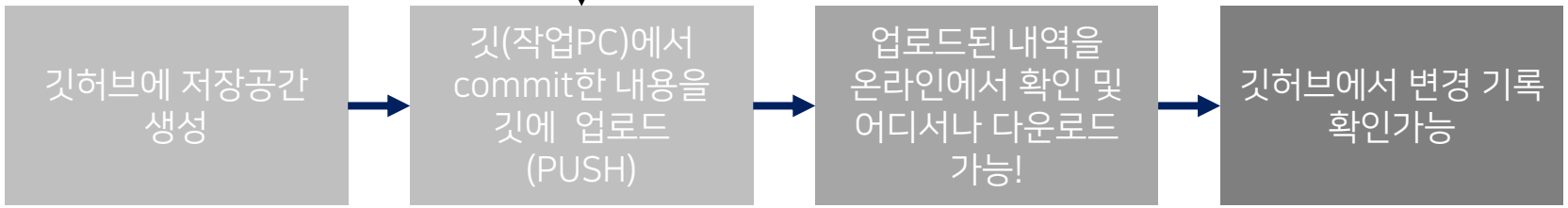
GIT HUB 이란?

- **GIT과는 다르다, 절대로 다르다!**
  - GIT이랑 GITHUB랑 하는일은 전혀 다릅니다!
  - **GIT이 PowerPoint면 GITHUB는 발표자료를 저장할 구글드라이브!**
    정말 구글드라이브 처럼 안에서 수정 및 내용확인도 가능하다!

- GIT의 기본 Process



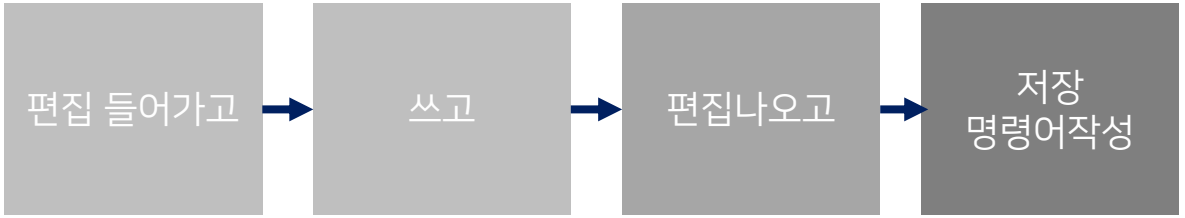
- GITHUB의 기본 Process



VIM 이란?

- VIM →cmd(명령어 치는 검정창 맞습니다)에서의 메모장
  - 그냥 메모장이 cmd로 들어온것
  - 왜 쓰나요? 일반적으로 서버OS에는 UI(마우스,아이콘 등등)이 없기 때문
  - 그럼 일반 PC에서는 그냥 텍스트 편집툴에 해도 되는 거 아닌가요? 맞습니다…

- VIM의 기본 Process



- VIM 의 이점
  - 가볍다! 어디서든 돌아간다!
- VIM 의 단점
  - 약 939852718789가지

언제 뭘 쓰지?

프로젝트
생성이후 파일
편집 때!

편집이후 로컬
PC(작업한 PC)상에서
저장(업데이트)할 때

작업한 PC에서
업데이트한 기록을
온라인으로 올릴 때!

반복하며 작업

혼자 하는데 굳이 깃헙 써야해?

*온라인 상에 업데이트를 해둬야 코드 망쳐서
다 안 돌아 갈 때 다시 이전버전으로 되돌릴 수 있다!

GIT 스타트

https://git-scm.com/ 접속하여 GIT 설치가능!

Mac OS X에는 이미 깔려 있을 수도 있으니 명령어 git --version을 쳐서 혹시 깔려 있나 보자!
(사실 각 환경 별 설치방법은 구글에 아주 자세히 기술되어 있다!)

설치가 되었다면
git config --global user.name “사용자이름”
git config --global user.email “이메일”
을 입력해 사용자 등록을 해주자!
여기서 등록을 하면 앞으로 하는 모든 commit기록에 해당 이름과 이메일이 보인다!
(누가 편집했는지 알게 하기 위함)

GIT 스타트

https://git-scm.com/ 접속하여 GIT 설치가능!

Mac OS X에는 이미 깔려 있을 수도 있으니 명령어 git --version을 쳐서 혹시 깔려 있나 보자!
(사실 각 환경 별 설치방법은 구글에 아주 자세히 기술되어 있다!)

설치가 되었다면
git config --global user.name “사용자이름”
git config --global user.email “이메일”
을 입력해 사용자 등록을 해주자! (cmd혹은 터미널 상에서)
여기서 등록을 하면 앞으로 하는 모든 commit기록에 해당 이름과 이메일이 보인다!
(누가 편집했는지 알게 하기 위함)

GIT 스타트

깃을 사용할 공간 배정

git init

앞으로 웹 작업할 git으로 관리될 폴더로 이동 후 git init을 해준다!

C:\Users\lukious\Desktop\webbuild>git init
Initialized empty Git repository in C:/Users/lukious/Desktop/ webbuild /.git/

앞으로 작업할 주소
*cd 명령어 -> 디렉토리 변경 (eg cd C:₩Users₩lukious₩Desktop ₩webbuild 하면 해당디렉토리로 이동)
ls 명령어 -> 디렉토리의 모든 파일 / 폴더명이 보이게 나온다.

정상적인 환경이라면 이제 branch (eg. (master))가 표시된다!

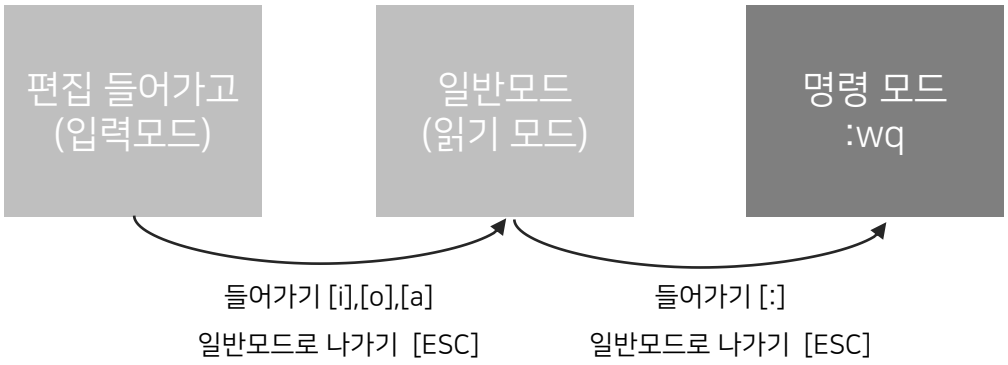
```
lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (master)
$
```

VIM 스타트

한 장으로 정리하는 VIM

공간에서 사용자가
파일수정 후 저장

- VIM에서 작성을 시작하려면 [I]키를 누르면 됩니다. 그러면 일반 모드에서 입력 모드로 전환이 되고, 커서가 바뀌면서 작성할 수 있는 상태라는 걸 알려줍니다.
- 작성후에 [ESC]키를 누르면 일반 모드(읽기 모드)로 돌아옵니다
- 일반모드에서 [:] 를 입력하면 명령어를 입력할 수 있는 명령모드로 전환 됩니다.
- 명령모드에서 wq를 누르면 저장(w) 후 종료(q)됩니다.



TIPS

- 파일이름이 abc.txt인데 수정하고 싶을 땐 command창에서 [vim abc.txt]입력

- 이름없는 파일을 집어넣으면 (eg. vim noname.txt) noname.txt 가 생성되고 작성할 수 있게 된다!

- [:wq]가 안될 땐 강제 명령어[:wq!]를 쓰자

- 밖에서 간단히 파일내용을 보고만 싶을땐 고양이를 부르자 [cat abc.txt] -> abc.txt의 내용 command에 출력

VIM GIT 스타트

공간에서 사용자가
파일수정 후 저장

git add

VIM에서 작성 및 편집한 문서를 '나 편집 했어! 하고 알려주는 역할'
정확히는 commit에 변경할 파일을 추가하는 방법

```
lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (master)
$ git add hello.py
```

Hello.py 라는 파일을 add 했다! 꼭 편집 및 생성하고 나서는 git add를 해주는 걸 잊지 말자!

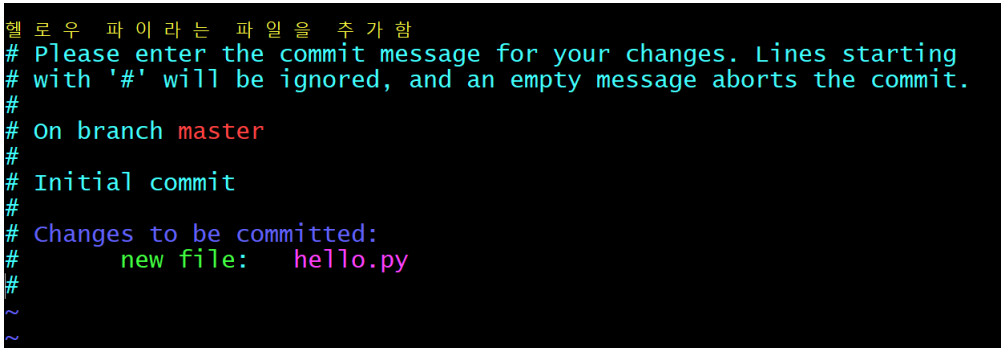
VIM GIT 스타트

GIT에 수정사항
업데이트

git commit

정상적으로 add가 된 상태에서 수정한 모든 내역을 하나로(version화) 해서 저장!

```
lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (master)
$git commit
```



위 사진과 같은 [커밋 메시지] 입력이 뜬다 [i]를 눌러 입력모드로 들어가고 맨 위에 무엇이 바뀌었는지 써준다.
이후 [ESC] -> [:] -> [wq]를 입력하여 저장하고 나가준다!

```
lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (master)
[master (root-commit) 502557f] 헬로우 파이라는 파일을 추가함
 1 file changed, 1 insertion(+)
 create mode 100644 hello.py
```

GIT 심화

GIT에 수정사항 업데이트

git branch

branch(브랜치)란 작업하고 있는 파일의 다른 버전
계통수와 비슷한 모습으로 각 브랜치를 쪼갠 다는건 현재
프로그램의 다른 부분을 강화한 버전이나 필요 없는 부분을
삭제한부분 버그가 나는 부분을 임시로 막아 둔 버전등 다양한
이유로 브랜치를 나눠야 할 때가 있다.

이 쪼개진 브랜치로 들어 가는 법을 checkout(체크아웃)이라고
하며 브랜치를 통합 하는 것을 merge(머지)라고 한다.

```
lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (master)
$ git branch
* master
```

git branch로 현재 존재하는 브랜치 확인

```
lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (master)
$ git branch extra_branch
```

extra_branch 라는 브랜치 생성

```
lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (master)
$ git branch
  extra_branch
* master
```

git branch를 통해 생성된
extra_branch를 확인 할 수 있다!

GIT 심화

GIT에 수정사항 업데이트

git checkout

```
lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (master)
$ git checkout extra_branch
Switched to branch 'extra_branch'

lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (extra_branch)
$
```

git checkout extra_branch
을 통해 브랜치가 변경된 것을 확인 가능!

- Commit 및 Vim에서의 저장은 내가 **작업하고 있는 branch에서만 일어난 일**!
- 만약 extra_branch에서 파일을 마구 수정하고 만들었다가 master로 돌아가면 모든 작업이 브랜치를 분기하기 전의 상태로 있다!

git merge

https://backlog.com/git-tutorial/kr/stepup/stepup2_4.html

GIT 기타 명령어

git log

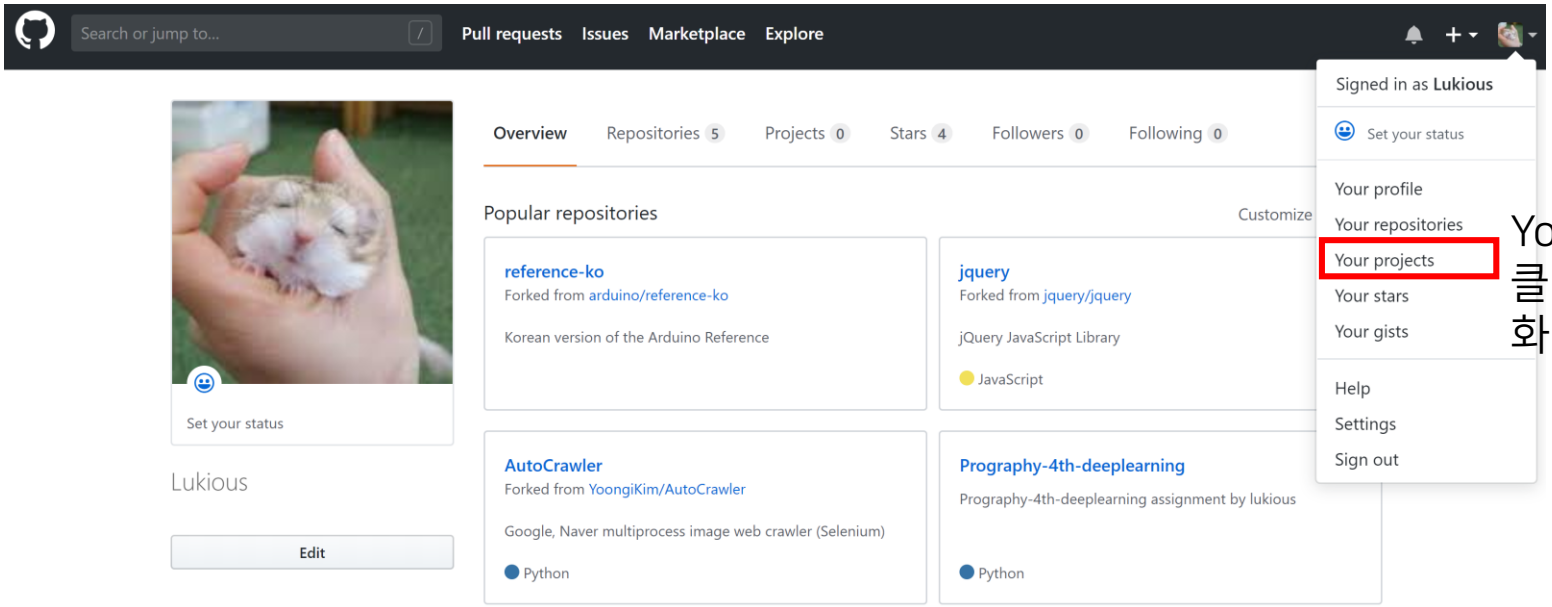
```
lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (extra_branch)
$ git log
commit 502557f60ed96577c3a0507d01665e9cf0cede83 (HEAD -> extra_branch, master)
Author: Lukious <zhsjzhsj@gmail.com>
Date:   Tue Mar 19 14:47:36 2019 +0900

    헬로우 파이라는 파일을 추가함

lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (extra_branch)
$ git log --graph
* commit 502557f60ed96577c3a0507d01665e9cf0cede83 (HEAD -> extra_branch, master)
  Author: Lukious <zhsjzhsj@gmail.com>
  Date:   Tue Mar 19 14:47:36 2019 +0900

      헬로우 파이라는 파일을 추가함
```

- Commit기록 확인 가능!
- --graph를 쓰면 branch 상황도 graphical 하게 볼 수 있다!

GITHUB 스타트

깃허브에 저장공간 생성

https://github.com/ 접속하여 회원가입!

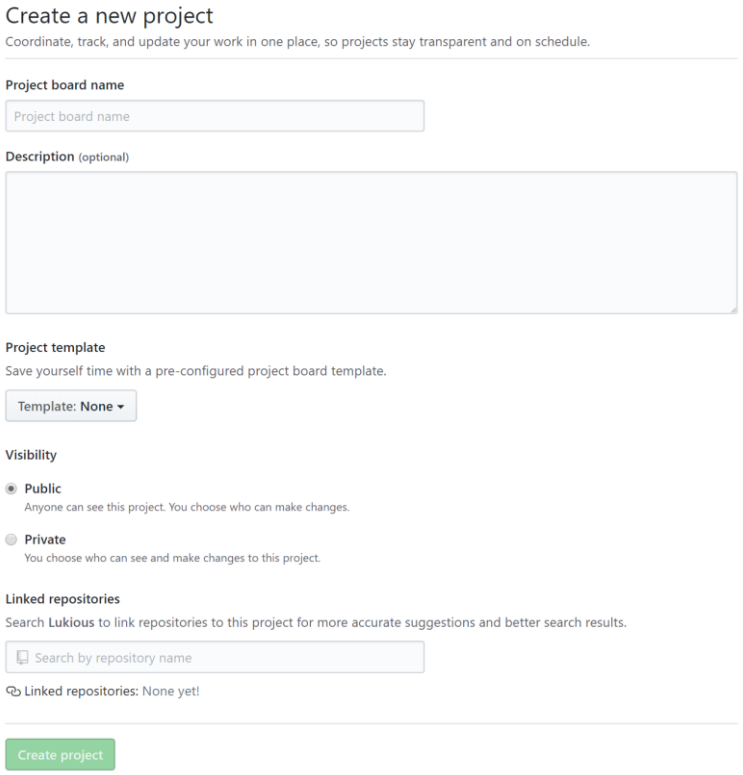


Your project를 클릭하여 프로젝트 화면으로!

GITHUB 스타트

깃허브에 저장공간 생성

New Project를 클릭해 새 프로젝트생성!

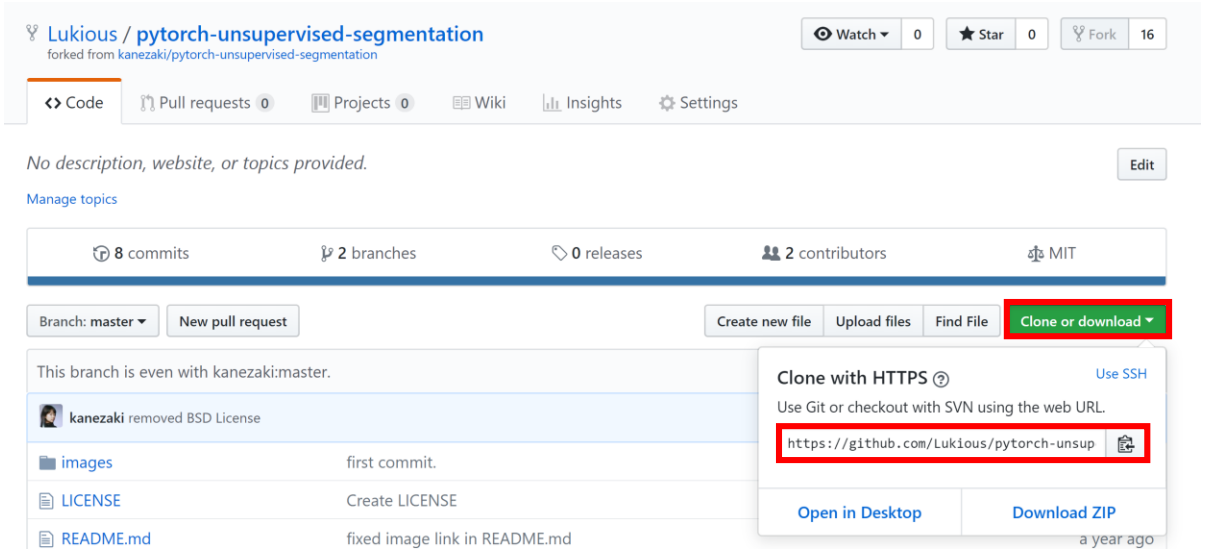


- 프로젝트이름 – 프로젝트이름 작성
- Description – 간단한 프로젝트 요약
- Templete – 기본세팅 none으로 두면 된다
- Visibility – 공개 or 비공개

GITHUB 스타트

깃허브에 저장공간 생성

git clone



- Git clone [저장소 주소]를 통해 현재 로컬(작업PC)디렉토리에 github저장소에 있는 내용을 복사해 올 수 있다!
- Clone 받아온 파일을 바탕으로 작업하면 로컬&깃허브 연결 작업을 하지 않아도 된다!
- **따라서 깃허브에서 프로젝트를 먼저 만들고 그걸 clone해서 작업하는 걸 추천!**

```
lukious@WINDOWS-T9R7TP0 MINGW64 ~/Desktop/webbuild (extra_branch)
$ git clone https://github.com/Lukious/pytorch-unsupervised-segmentation.git
```

GITHUB 스타트

깃(작업PC)에서
commit한 내용을
깃에 업로드
(PUSH)

중요!

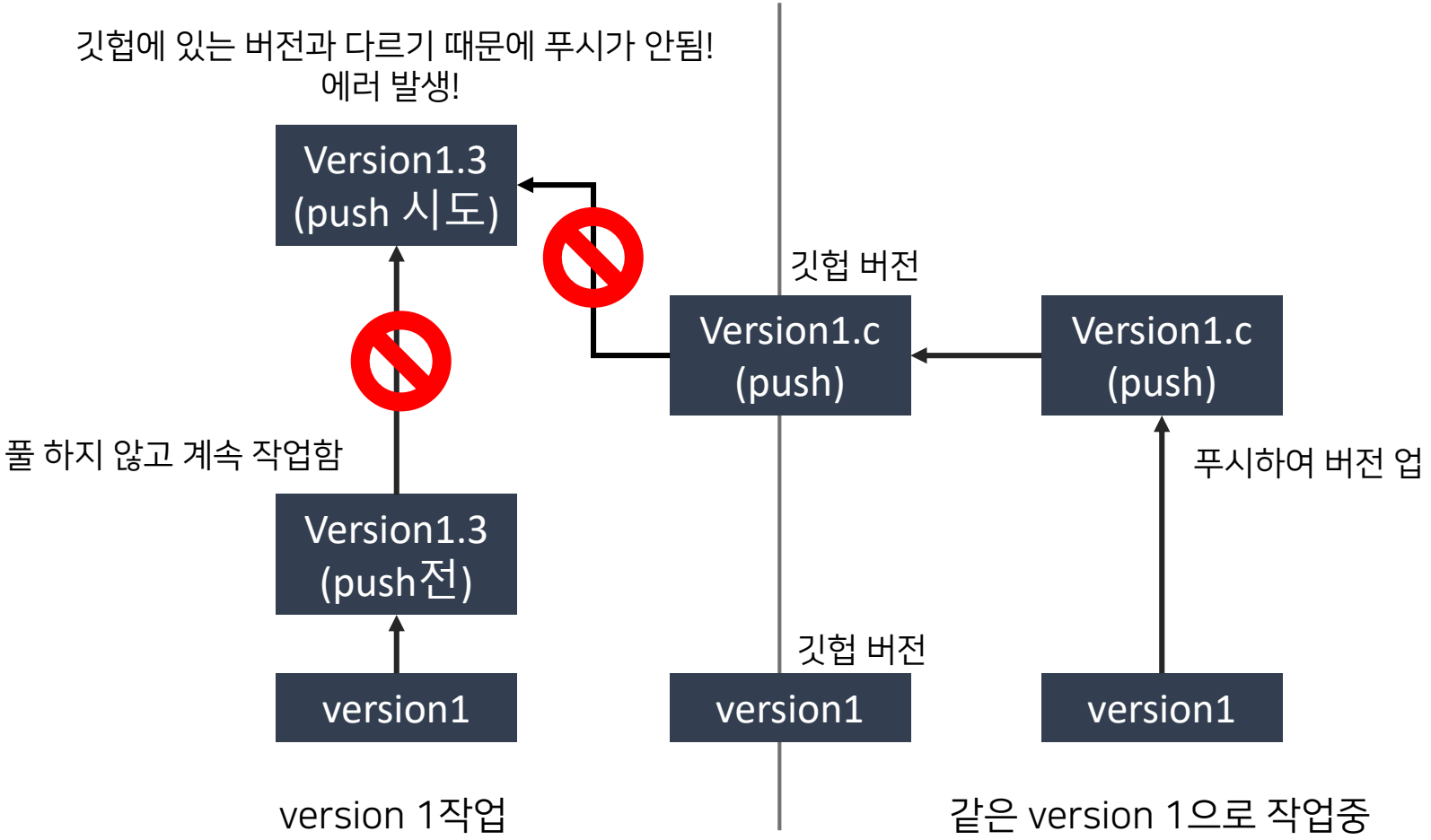
git push

- Clone하고 작업하고 commit한 내역을 연결된 github저장소로 보내는 명령어!
- 간단히 말해서 upload 작업이다!

git pull

- 내가 다른 곳에서 작업하고 push한 내용을 다시 받아 올때 혹은 다른 사람이 commit&push 한 내용을 반영 받고 싶을 때 git pull을 통해 받아 올 수 있다!

GITHUB 주의점!


깃헙에 있는 버전과 다르기 때문에 푸시가 안됨!
에러 발생!
Version1.3
(push 시도)
깃헙 버전
Version1.c
(push)
Version1.c
(push)
풀 하지 않고 계속 작업함
푸시하여 버전 업
Version1.3
(push전)
깃헙 버전
version1
version1
version1
version 1작업
같은 version 1으로 작업중