

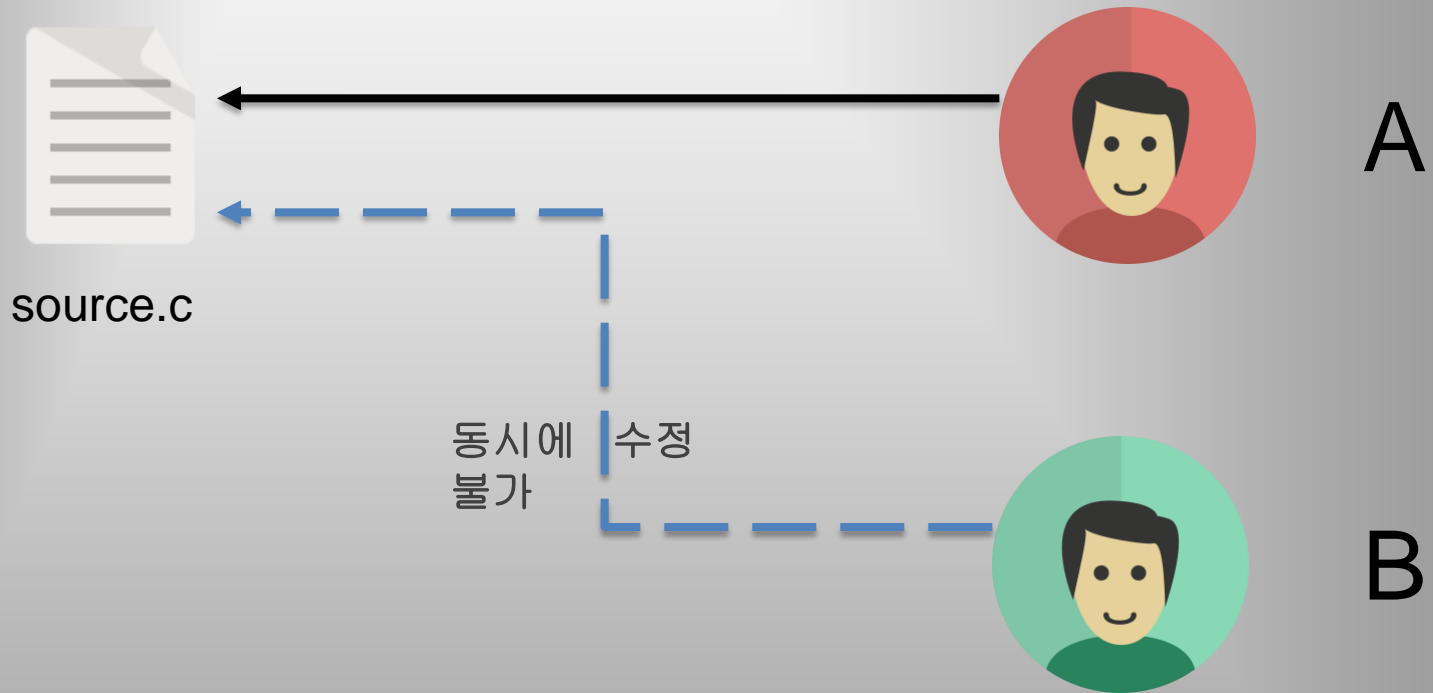
# Github

20071086 임지민  
20071155 정명학  
20081349 최연상  
20110391 김해연

Why    What    How    Problem

Why?

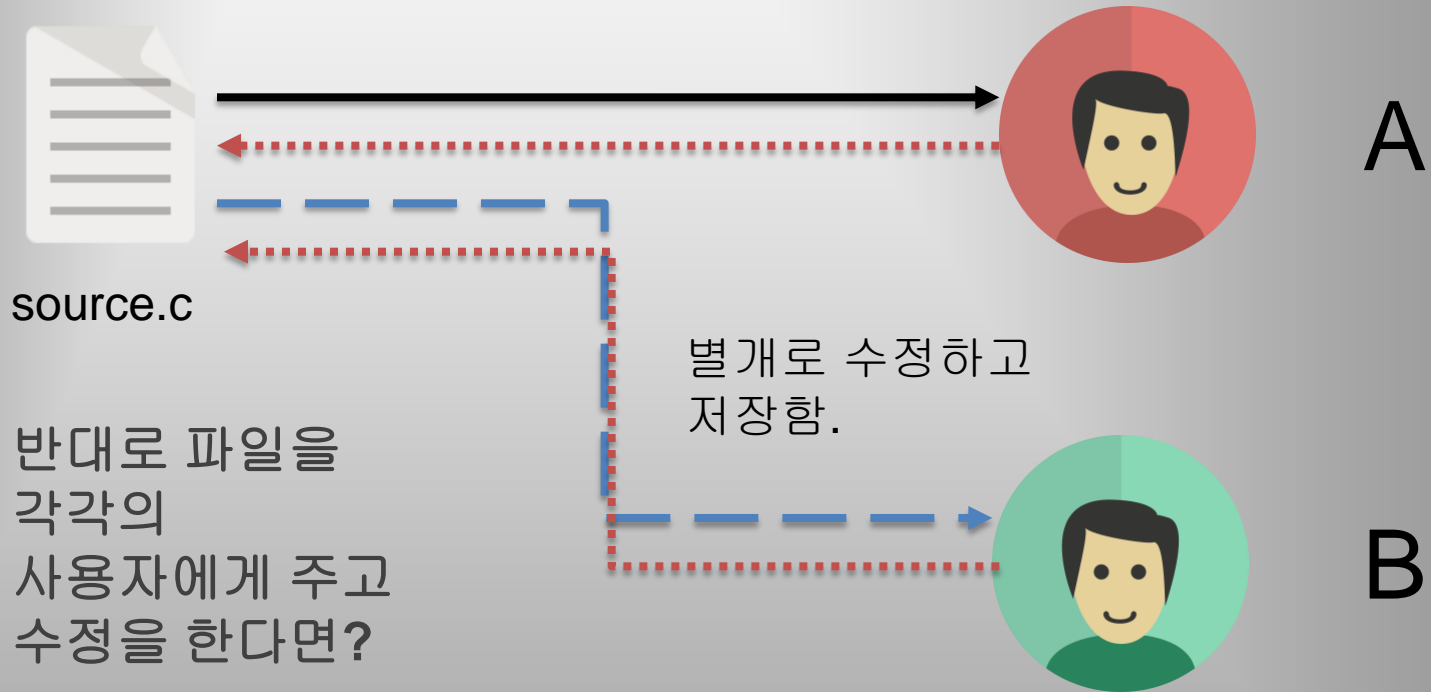
# 상황 #1



- B는 A가 source.c 파일을 모두 수정하기까지 기다려야함

>> 시간적인 효율이  
적다  
>> 사기  
저하

# 상황 #1



## ○ 요구사항

A 와 B가 작업 할 수 있는 각각의 **저장소**가 있어야함

`source.c` 파일을 사용자에게 주기 위해 **저장소**가 있어야함

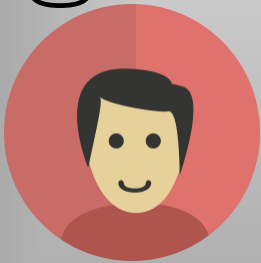
- 각각 작업 후 `source.c` 파일을 병합(merge)하는 과정이 필요

**\*\* Git을 이용하면 아주 간단한 방법으로 합치고, 바로 적용**

## 상황 #2 로그인 프로그램 개발



팀장



A

HTML 개발만 가능



Login.html  
Sign.html



Proc.java



B

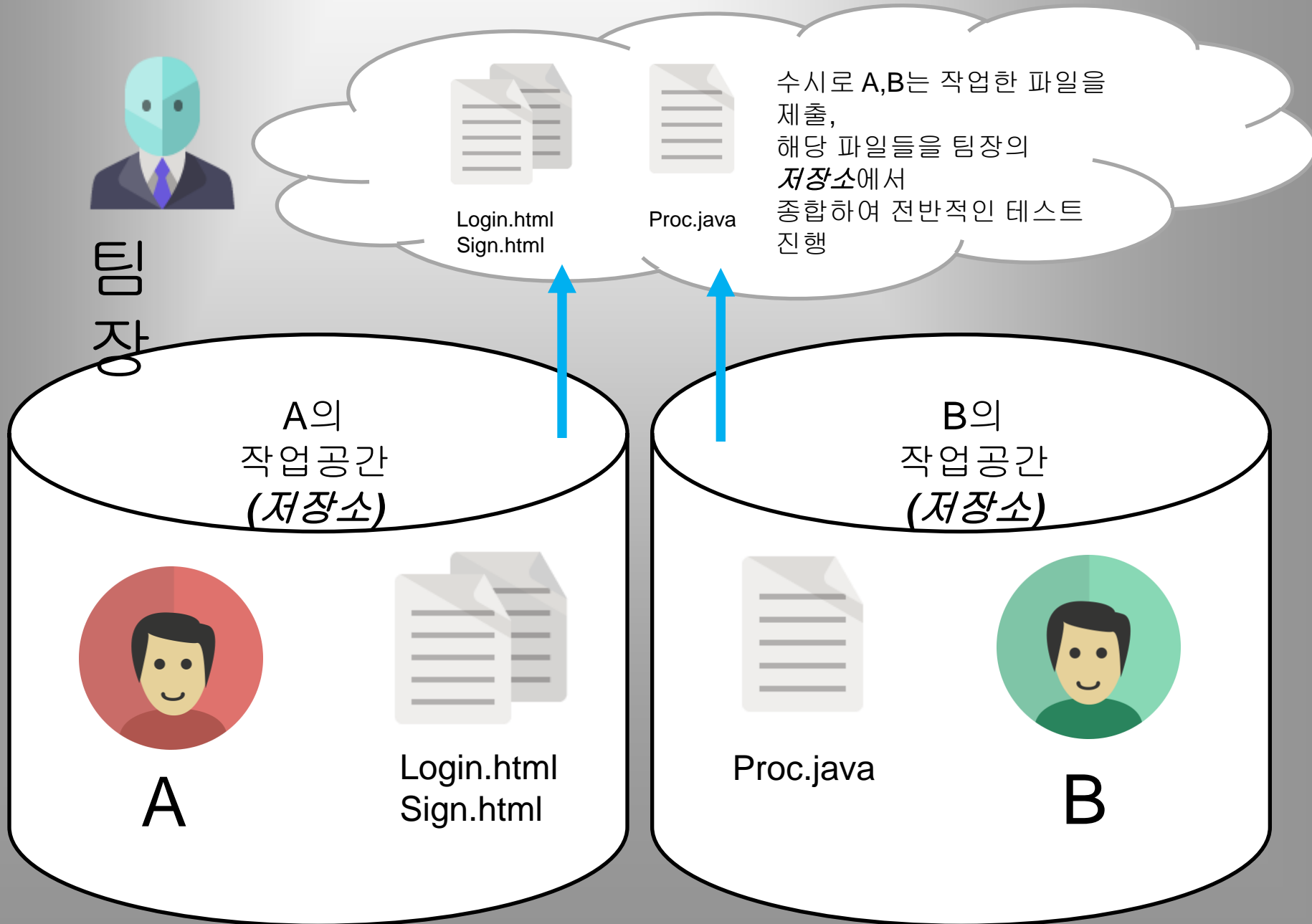
Java 개발만 가능

**\*\* A와 B사용자는 개발 전 서로가 사용 할 함수명, 변수명에 대해서는 정의 된 상태**

## 상황 #2 로그인 프로그램 개발

어떻게 하면 효율적으로 개발과 테스트를 할 수 있을까?

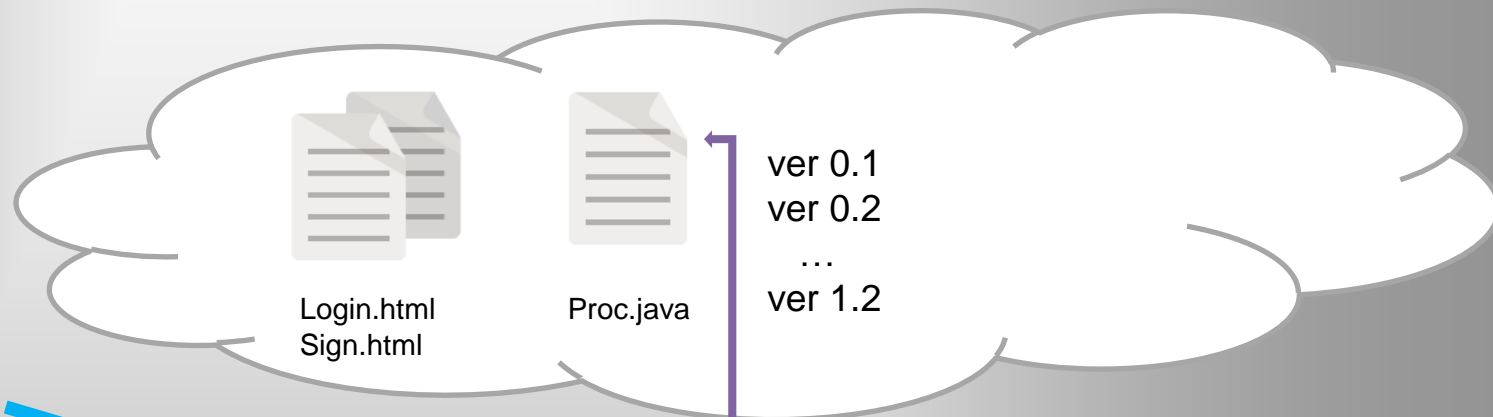
## 상황 #2 로그인 프로그램 개발



# 상황 #2 로그인 프로그램 개발



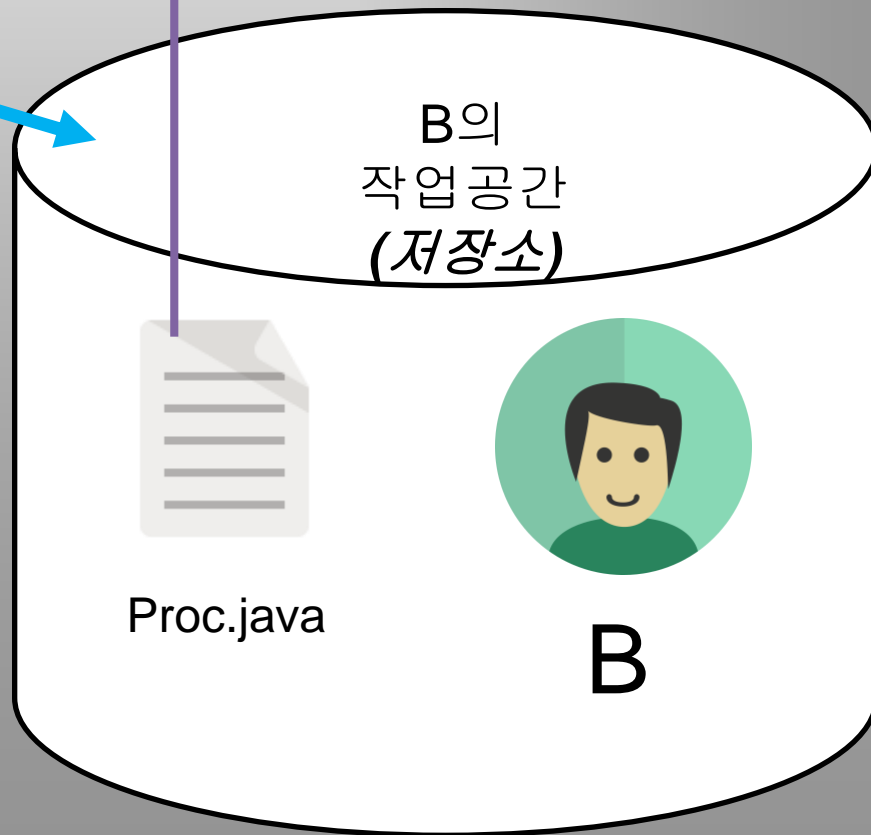
팀장



이거  
다시해!

1. B가 작성한 파일의 변경을 요구할 때도 팀장은 프로그램을 테스트하는데 전혀 지장없고
2. B는 변경완료 된 파일을 제출함과 동시에 팀장은 그것을 간단한 작업을 통해 다시 테스트 가능
3. 팀장은 예전에 B가 작성했던 파일도 불러올 수 있음

이러한 일련의 과정을 거쳐  
소스코드를 **형상관리**하게  
된다.



What?

# Git & Github

## Git

- 소스버전 (형상관리) 프로그램
- 로컬 저장소 (Stage)
- 원격 저장소 (Repositories)
- 리누스토발즈가 리눅스 커널개발에 이용하려고 만들었음

## Github

- 원격 저장소를 지원해주는 대표적인 공간
- 무료로 사용하기 위해서는 저장소를 공개해야함
- Github를 통해 비공개로 개발 하려면 비용부과

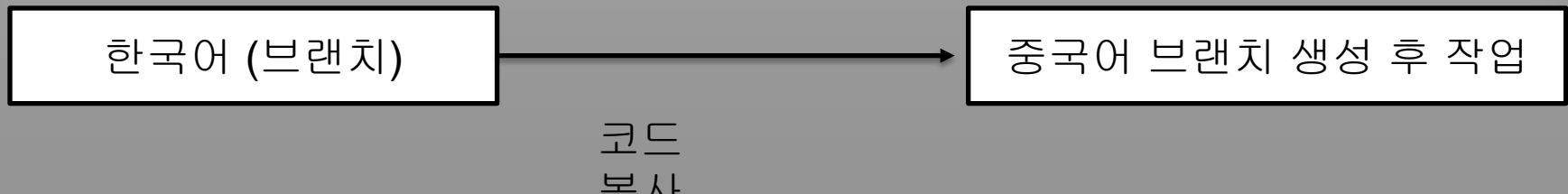


# Git 주요 개념

## Branch

- Git 에서 가장 중요한 개념
- 독립적인 개발 작업 공간
  - => 원격 저장소에 있는 코드를 모두 복사 한 후에 독립적인 공간에서 개발이 가능

Ex) 하나의 앱이 있는데, 기본 언어는 한국어로 되어있고 중국어 버전을 개발하려고 한다.

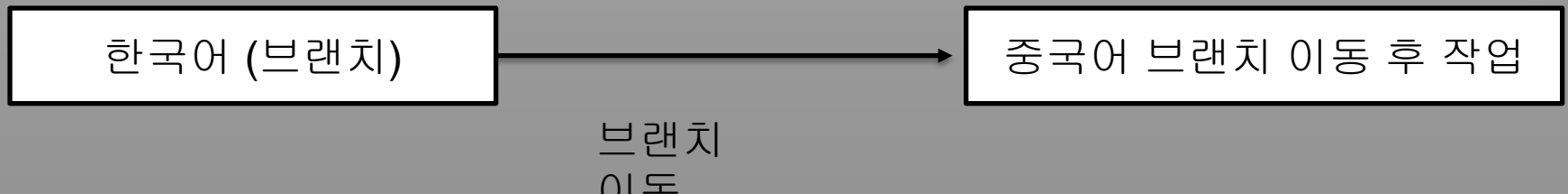


# Git 주요 개념

## Checkout

- 브랜치로 이동하기 위한 절차 (명령어)
- Checkout 과 동시에 새로운 브랜치 생성도 가능 (`git checkout -b LangCN`)

Ex) 중국어 브랜치로 이동 후 작업을  
진행



# Git 주요 개념

## Commit

- 개발 작업 후 로컬저장소에 반영하는 과정
- **Commit** 할 땐 항상 정보를 남기게 된다. (해쉬코드)  
=> 각각의 브랜치에서 어느 부분의 내용이 수정되었는지,
- 필요할 땐 예전에 **Commit** 했던 내용들을 불러올 수 있음.

Ex) 중국어 브랜치로 개발 진행을 하며 작업 중간중간에 저장소로

Commit

...

중국어 브랜치

...

commit

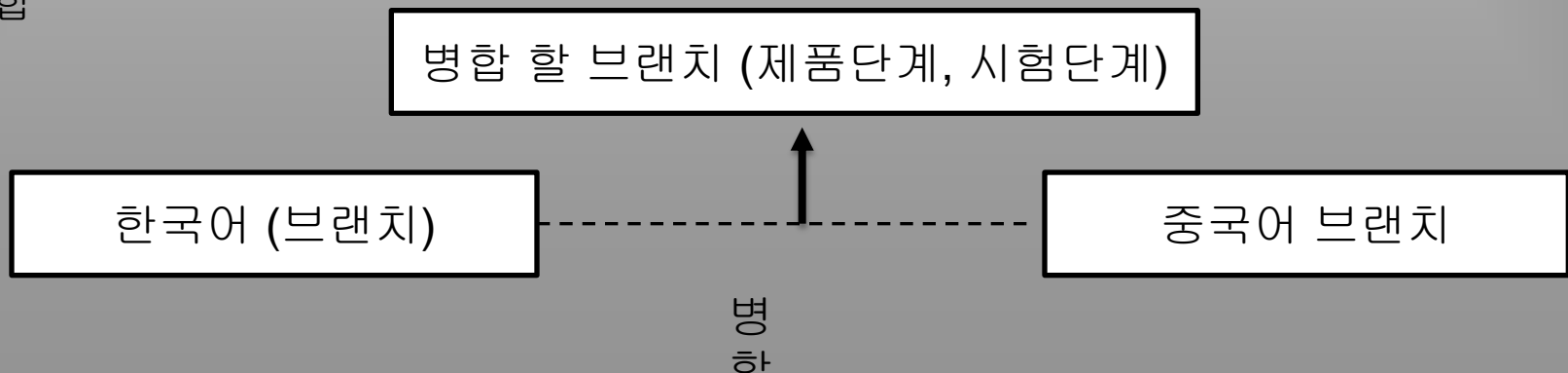
로컬 저장소

# Git 주요 개념

## Merge

- Commit 된 브랜치들의 파일들을 합치는 작업을 할 때 사용
- 필요한 브랜치의 파일들을 합친 후 개발을 다시 진행할 수 있으며 테스트도 가능

Ex) 중국어 개발 작업 후 파일을  
병합



# Git 주요 개념

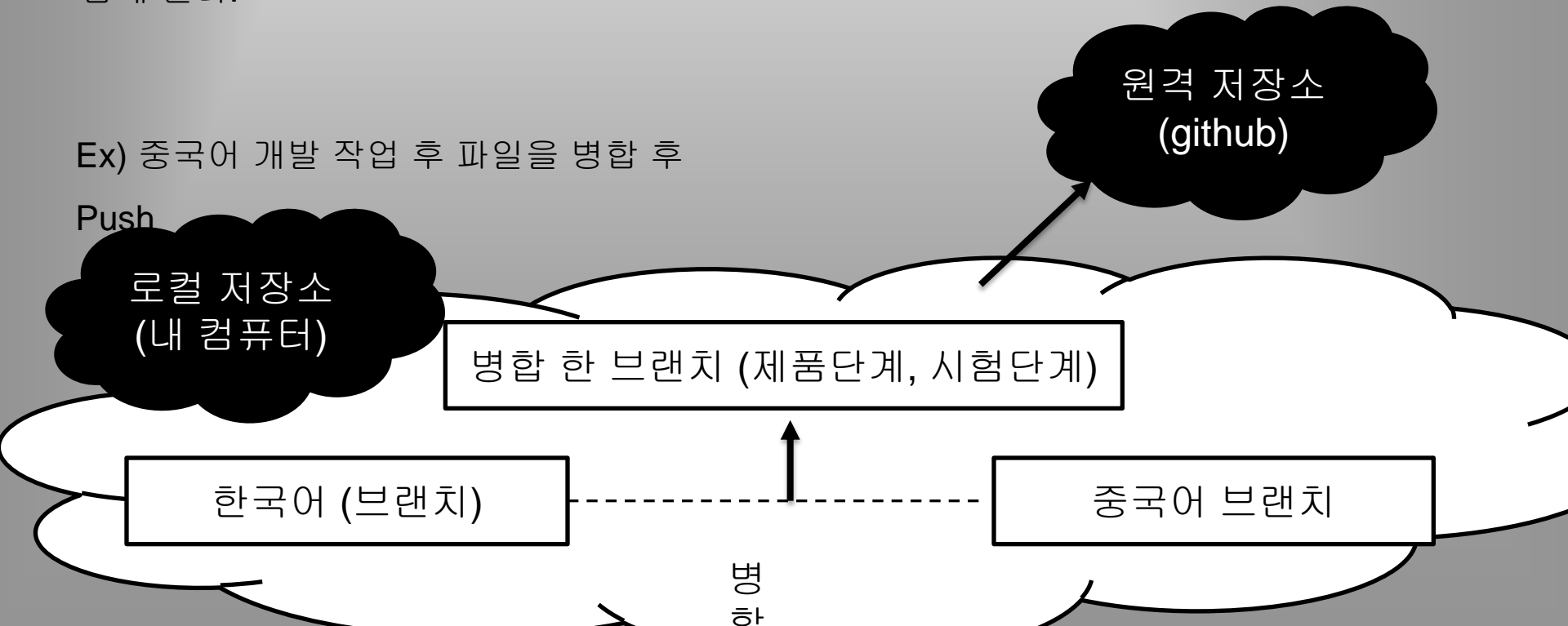
## Push & Pull

### Push

- Commit 된 브랜치들을 Commit 정보와 함께 원격 저장소로 업로드하는 것
- 원격저장소에서는 로컬저장소와 동일한 Commit 정보와 File들이 남게 된다.

Ex) 중국어 개발 작업 후 파일을 병합 후

Push



### Pull

- 원격 저장소에 존재하는 **Commit** 정보와 **File**들을 모두 로컬 저장소로 다운받는 것
- 로컬저장소에서는 원격저장소와 동일한 **Commit** 정보와 **File**들이 남게 된다.



## bug fix lang

[Browse code](#)

master



yunchiri authored 2 months ago

1 parent 64592a9

commit 64a4eb3cc0e150b33564e441cbac0fe4eb36e232

Showing 1 changed file with 3 additions and 3 deletions.

[Show Diff Stats](#)

6 routes/iedor.js

[View](#)

```
@@ -100,7 +100,7 @@ exports.create = function(req,res){
100 100     var news_      = req.body.news;
101 101     var fontColor_ = req.body.fColor;
102 102     var backColor_ = req.body.bColor;
103 -    var lang_      = req.body.lang;
+103 +    var langcode_   = req.body.lang;

104 104     console.log("into Create MD5 : " + iedorMD5_);
105 105     if(typeof iedorMD5_=="undefined" ) return;
106 106     @@ -108,8 +108,8 @@ exports.create = function(req,res){
108 108         //client.connect();
109 109         // console.log("got create");
110 110
111 -    client.query("insert into iedormaster (id,name,tname,twitter,news,fontColor ,backColor ,lang ,vote,down,updat
112 -    ,[iedorMD5_,name_,tname_,twitter_,news_,fontColor_, backColor_,lang_, twitter_,news_,fontColor_,
+111 +    client.query("insert into iedormaster (id,name,tname,twitter,news,fontColor ,backColor ,lang ,vote,down,updat
+112 +    ,[iedorMD5_,name_,tname_,twitter_,news_,fontColor_, backColor_,langcode_, twitter_,news_,fontColo
    ,function(err,rows,fileds){
113 113         if(err) throw console.log(err);
114 114         // console.log('result :', rows);
115 115     }
```

# Summary

How?

# Github 가입 및 저장소 생성

+ 그룹 생성

# GitHub 가입



<https://github.com/>

GitHub 홈페이지 방문  
**<https://github.com>**

Username, Email,  
Password(회원 정보) 입력  
후,  
**Sign up for GitHub**  
클릭

Pick a username

Your email



Create a password



Use at least one lowercase letter, one numeral, and seven characters.

**Sign up for GitHub**

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

# GitHub 가입

계정 생성 확인  
가능

## Welcome to GitHub

You've taken your first step into a larger world, @YoniKim.



Completed  
Set up a personal account



Step 2:  
Choose your plan




Step 3:  
Go to your dashboard


Finish sign up

계정 생성 완료

# GitHub 저장소 생성(개인)


 YoniKim ▾

ProTip Know a non-profit that wants to use git? We'd love to help.




News FeedPull RequestsIssuesStars

**GitHub Bootcamp** If you are still new to things, we've provided a few walkthroughs to get you started.




**Set up Git**  
A quick guide to help you get started with Git.

1




**Create repositories**  
Repositories are where you'll work and collaborate on projects.

2




**Fork repositories**  
Forking creates a new, unique project from an existing one.

3



**Be social**  
Send pull requests, follow friends. Star and watch projects.


4



**Welcome to GitHub! What's next?** (2 minutes ago)

- Create a repository
- Tell us about yourself
- Browse interesting repositories
- Follow @github on Twitter

Your repositories (0)



 New repository

You don't have any repositories yet.  
Create your first repository or learn more about Git

New repository  
클릭


# GitHub 저장소 생성(개인)


**Owner** **Repository name**

PUBLIC  YoniKim / Project 


Great repository names are short and memorable. Need inspiration? How about **massive-octo-wallhack**.

**Description (optional)**

☒  **Public**  
Anyone can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

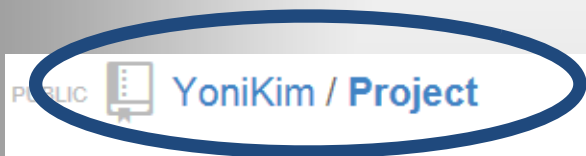
☐ **Initialize this repository with a README**  
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** | Add a license: **None** 

**Create repository**

저장소 이름 입력  
후, **Create  
repository** 클릭

# GitHub 저장소 생성(개인)



개인 저장소 생성 완료

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



Set up in Desktop

or

HTTP

SSH

`https://github.com/YoniKim/Project.git`



We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

## Create a new repository on the command line

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/YoniKim/Project.git
git push -u origin master
```

## Push an existing repository from the command line

```
git remote add origin https://github.com/YoniKim/Project.git
git push -u origin master
```

# GitHub 저장소 생성(그룹)

The screenshot displays the GitHub user interface for a user named YoniKim. The top navigation bar includes the GitHub logo, a search bar, and links for Explore, Gist, Blog, and Help. The user's profile name 'YoniKim' is in the top right, with a dropdown menu containing a plus icon, a wrench icon, and a document icon. The left sidebar lists various account settings: Profile, Account Settings, Emails, Notification Center, Billing, Payment History, SSH Keys, Security, Applications, Repositories, and Organizations. The 'Organizations' link is circled in blue. The main content area shows two buttons: 'Create new organization' and 'Turn YoniKim into an organization', both circled in blue. Below these buttons is a section titled 'Organizations' with the message 'You are not a member of any organizations'. A green rounded rectangle highlights a list of steps: 'Account Settings', 'Organizations', and 'Create new organization'.

Search or type a command

Explore Gist Blog Help

YoniKim

Account settings

Create new organization Turn YoniKim into an organization




Organizations

You are not a member of any organizations

Account Settings  
Organizations  
Create new organization

# GitHub 저장소 생성(그룹)

## Create an organization

-  Completed  
Set up a personal account
-  Step 2:  
Set up the organization
-  Step 3:  
Invite team members

### Set up the organization

Organization name

The organization will live at <https://github.com/>

Billing email

Receipts will be sent here

### Choose the organization's plan

 SECURE

| Plan        | Cost        | Private repos |                         |
|-------------|-------------|---------------|-------------------------|
| Diamond     | \$450/month | 300           | <button>Choose</button> |
| Platinum    | \$200/month | 125           | <button>Choose</button> |
| Gold        | \$100/month | 50            | <button>Choose</button> |
| Silver      | \$50/month  | 20            | <button>Choose</button> |
| Bronze      | \$25/month  | 10            | <button>Choose</button> |
| Open Source | \$0/month   | 0             | <button>Choose</button> |

### Organizations

- ✓ Repository management
- ✓ Fine-grained permissions
- ✓ Focused dashboard

저장소 이름 입력  
mail 주소 입력  
Create Organization 클릭

Create Organization

# GitHub 저장소 생성(그룹)

## Invite team members



Completed

Set up a personal account



Completed

Set up the organization



Step 3:

Invite team members

### Add owners to the YoniProject organization



YoniKim

Type a username

Add

그룹 멤버 추가

### Permissions

In addition to the Owners team, you'll be able to create **unlimited teams** with fine-grained permissions to your repositories.

You'll also be able to add new repositories to your organization and manage who has access to which repositories.

Anyone you add to this team will be able to **modify the organization's billing information** and will have **complete access to all repositories and organization information**.

Finish

# GitHub 그룹 저장소


The screenshot shows the GitHub homepage with several annotations:

- HOME 버튼**: A green box highlights the GitHub logo in the top left corner, with a green arrow pointing to it from the text.
- 저장소 확인**: A green box highlights the 'Family-of-loves' repository dropdown menu in the top left, with a green arrow pointing to it from the text.
- 선택 저장소 안에 파일 만들기**: A green box highlights the 'New repository' button in the 'Repositories (2)' section, with a green arrow pointing to it from the text.

The interface includes a search bar, navigation links (Explore, Gist, Blog, Help), a user profile (YoniKim), and a list of repositories (Family-of-loves/TestLab, Family-of-loves/RunningMan).

# GitHub 그룹 저장소 파일 생성


**Owner** **Repository name**


PUBLIC  Family-of-loves / 생성 파일 이름

Great repository names are short and memorable. Need inspiration? How about [laughing-octo-ironman](#).

**Description (optional)**

파일에 대한 설명

☒  **Public**  
Anyone can see this repository. You choose who can push to it.

☐  **Private**  
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**  
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** | Add a license: **None** ⓘ

**Create repository** | 파일 생성 완료

파일 public/private 여부 결정  
**Private**으로 할 경우  
추가요금 발생

# GitHub 그룹 저장소 파일생성



## Family-of-loves



 New repository

### Test

This is a storage for test

Updated a minute ago

★ 0  0

### TestLab

Git 맘대로 사용할 수 있는 테스트 저장소

Updated 21 hours ago

★ 0  0

생성완료

# GitHub 그룹 저장소 setting



사용자

개인 저장소

그룹 저장소

그룹 안에서 일어나는 일 확인

그룹 멤버 확인/제거/추가

그룹 멤버 확인\_private/public

Get started!

# Git for Windows

- Git for Windows 두 가지 방식의 환경을 제공 ( GUI, CLI )

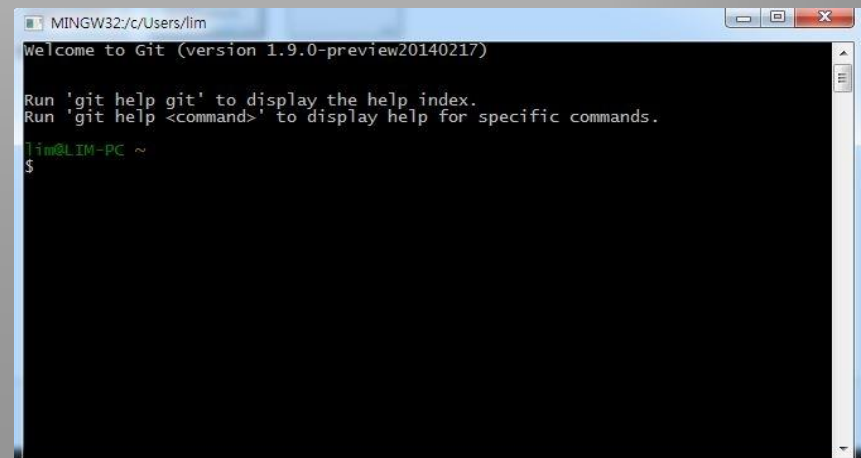
- **GUI** (Graphic User Interface) - Git GUI

사용자가 **그래픽**을 통해 컴퓨터와 정보를 교환하는 작업환경  
배우기 쉽고 사용이 간편하나, 사용이 제한적

GUI 구성요소 : 윈도우창, 스크롤바, 아이콘, 이미지, 버튼 등

- **CLI** (Command Line Interface) - Git Bash

사용자가 **명령어**를 입력단위로 컴퓨터와 정보를 교환하는 작업환경  
명령어 사용법과 기능을 숙지하여야 하나, 명령 변환과 조합 기능으로 사용이  
비제한적



# Git for Windows – 설치 (1)

**git** --fast-version-control

Search entire site...

**About**  
**Documentation**  
**Blog**  
**Downloads**  
GUI Clients  
Logos  
**Community**

## Downloads

클릭! 다운로드

Mac OS X Windows Linux Solaris

Older releases are available and the Git source repository is on GitHub.

**GUI Clients**  
Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.  
[View GUI Clients →](#)

**Logos**  
Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.  
[View Logos →](#)

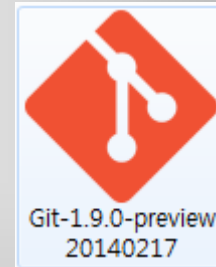
Latest source release  
**1.9.0**  
Release Notes (2014-02-14)  
[Download for Windows](#)

The entire **Pro Git book** written by Scott Chacon is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

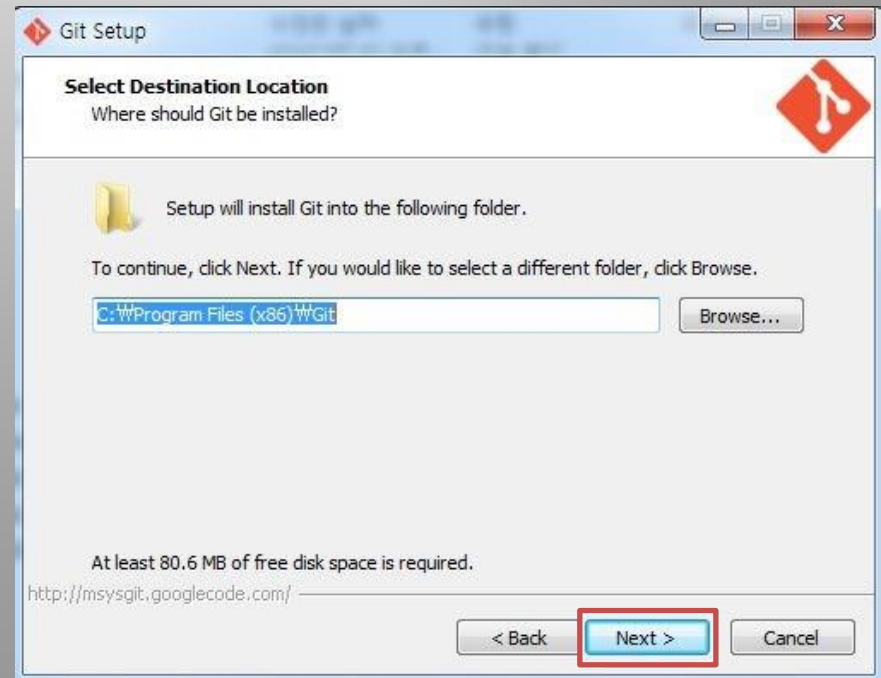
다운로드 주소 : <http://www.git-scm.com/downloads>

# Git for Windows – 설치 (2)

- 다운받은 설치파일 실행 (Git-1.9.0-preview20140217)

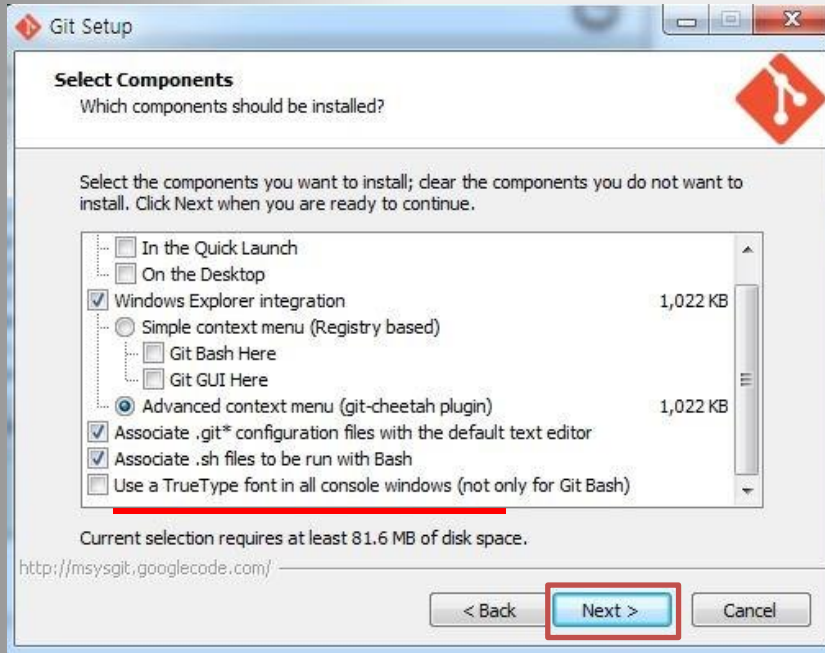


설치파일 실행  
초기화면



설치할 폴더  
정하기

# Git for Windows – 설치 (3)



[Advanced context menu] 체크 시,  
마우스 우 클릭 시, Git에 대한 **썬-메뉴** 사용  
가능



Git 명령어 사용을 위한 **환경 선택**하는 화면

## Use Git Bash Only :

해당프로그램에서 제공해주는 Git Bash에서만 Git 명령어들을 사용할 경우 (별도의 환경변수 설정 X)

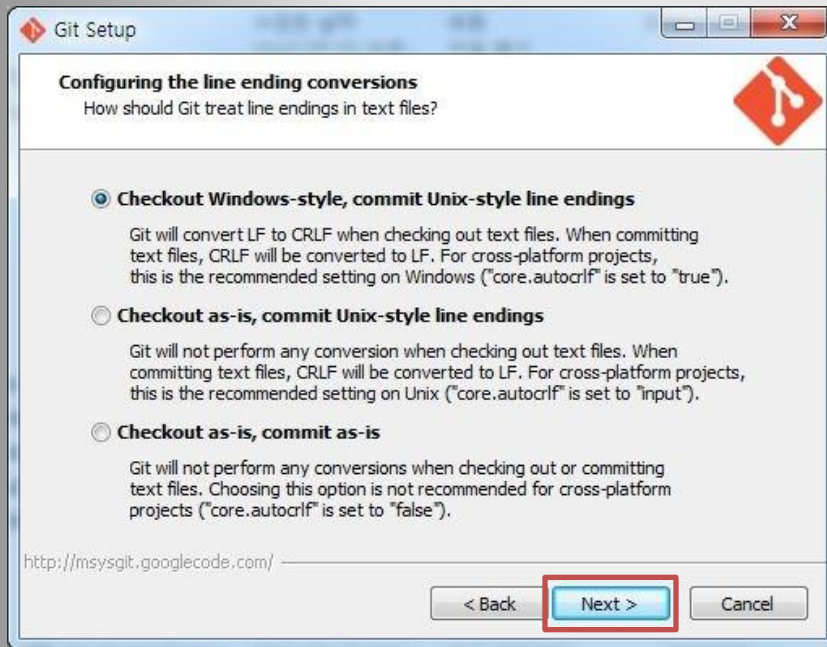
## Run Git from the Windows Command Prompt :

윈도우의 cmd프롬프트에서 실행하는 경우 (Git을 실행하기 위한 명령어들의 경로가 환경변수에 추가됨)

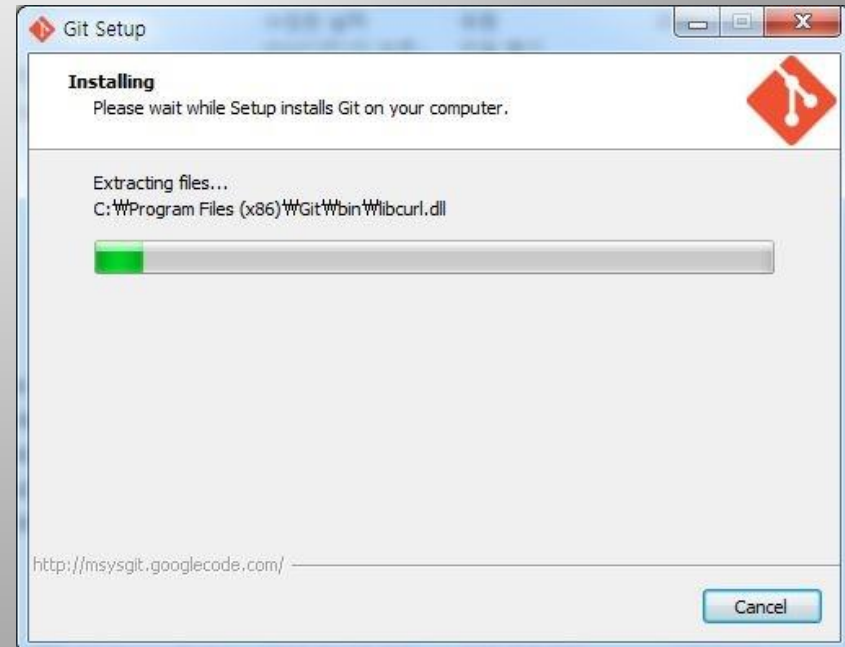
## Run Git and included Unix tools from the Windows Command Prompt :

Git명령어 뿐만 아니라 여러 유닉스 명령어들을 윈도우 cmd 프롬프트에서 사용하는 것

# Git for Windows – 설치 (4)



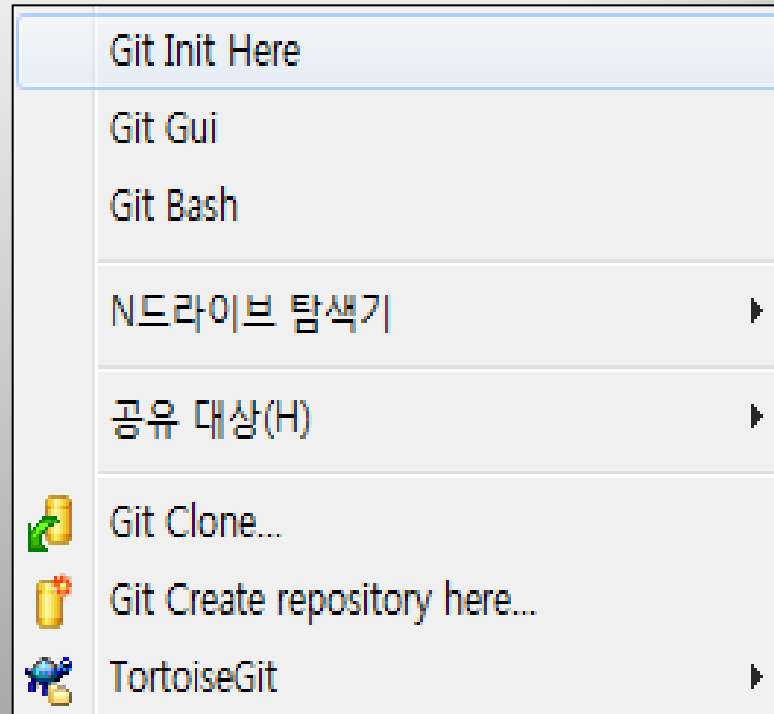
소스파일 내, **행 변환 문자에 대한 설정**



모든 설정 선택 후,  
설치화면

- ✦Windows : 행 변환 시, CRLF 사용 (1)
- ✦UNIX : 행 변환 시, LF만 사용 (2)

# Git for Windows – 구성 (1)

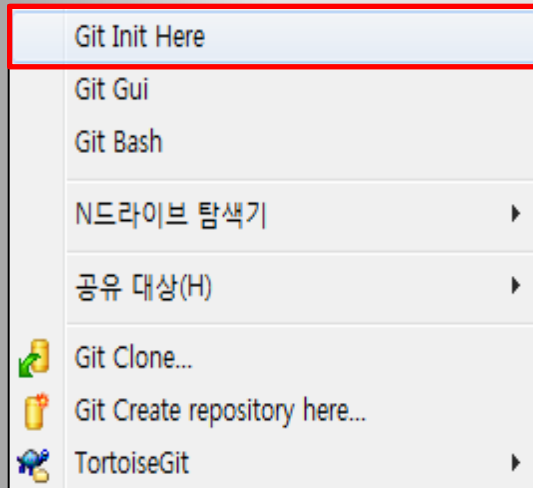


⇒ Git Init Here : 해당 폴더를 Git Repository로 생성

⇒ Git Gui : Git을 GUI 형태로 이용 (아이콘 위주)

⇒ Git Bash : Git을 CLI 형태로 이용 (명령어 위주)

# Git for Windows – 구성 (2)



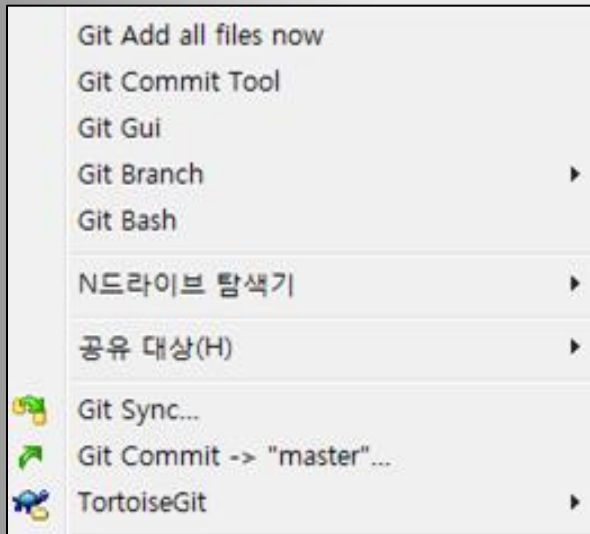
## ※ Git Repository 생성

- 1) Git Repository 로 만들고자 하는 폴더에 접근
- 2) 마우스 우 클릭 후, Git Init Here 이용
- 3) Git Init Here은 해당 폴더를 Git Repository로 생성

| 이름    | 수정한 날짜           | 유형     | 크기  |
|-------|------------------|--------|-----|
| .git  | 2014-03-13 오후... | 파일 폴더  |     |
| text1 | 2014-03-13 오후... | 텍스트 문서 | 1KB |
| text2 | 2014-03-13 오후... | 텍스트 문서 | 1KB |

해당 폴더가 Git Repository가 되어 숨겨진 폴더 '.git'폴더가 생성

# Git for Windows – 구성 (3)



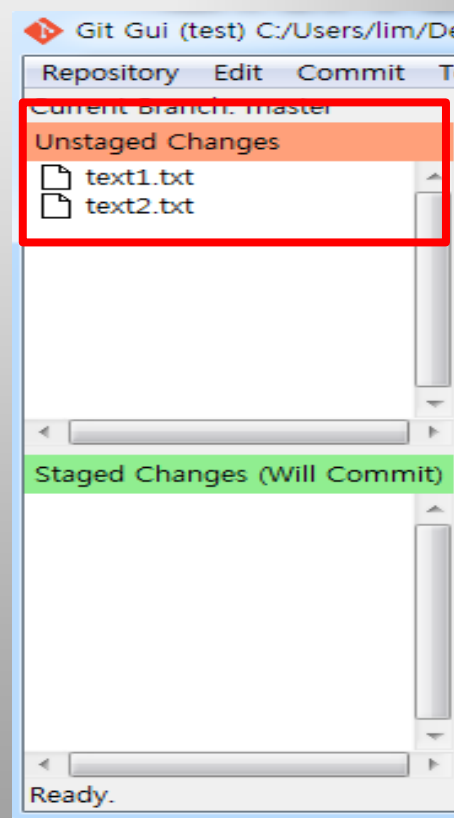
Git Add all files now

모든 새롭게 변경된 사항들을 스테이지에 등록

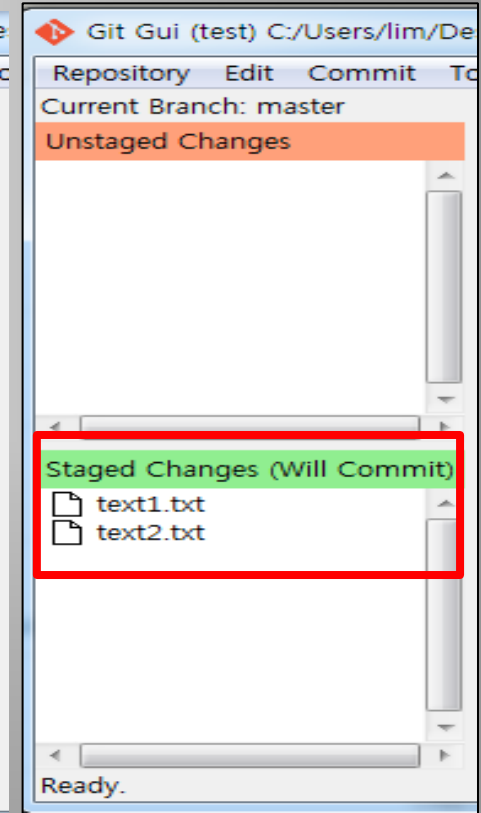
Git Commit Tool

커밋을 할 수 있는 Tool을 활성화 시킴

이때, Tool은 Git Gui 버튼을 눌렀을 때와 같음

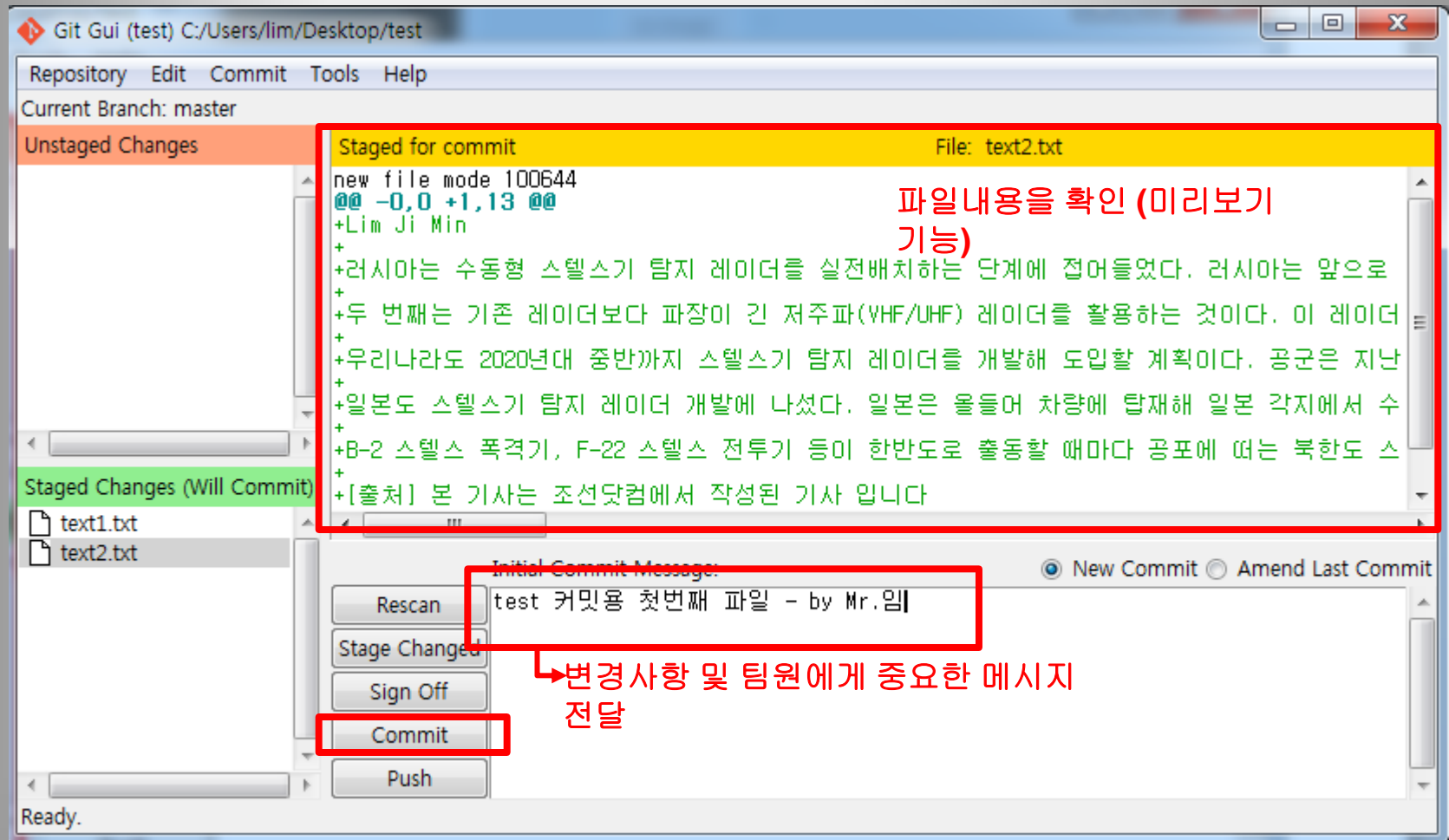


변경된 사항이  
스테이지 등록 되기  
전



변경된 사항이  
스테이지 등록 된  
후

# Git for Windows – 구성 (4)



# Git for Windows – 구성 (5)

Commit History 정보확인

The screenshot displays the Git GUI interface for a repository named 'gitk: test'. The top section shows the commit history with a single commit on the 'master' branch. A red box highlights the commit message 'test 커밋용 첫번째 파일 - by Mr.임', with a red arrow pointing to the text '변경사항에 대한 메세지' (Message about the change).

The bottom section shows the details of the selected commit. A red box highlights the commit message and the file list, with a red arrow pointing to the text 'Commit 된 파일목록' (File list after commit). The file list includes 'text1.txt' and 'text2.txt'. Another red box highlights the file content for 'text1.txt', with a red arrow pointing to the text '파일내용을 확인 (미리보기 기능)' (Check file content (preview feature)).

The commit details section includes the following information:

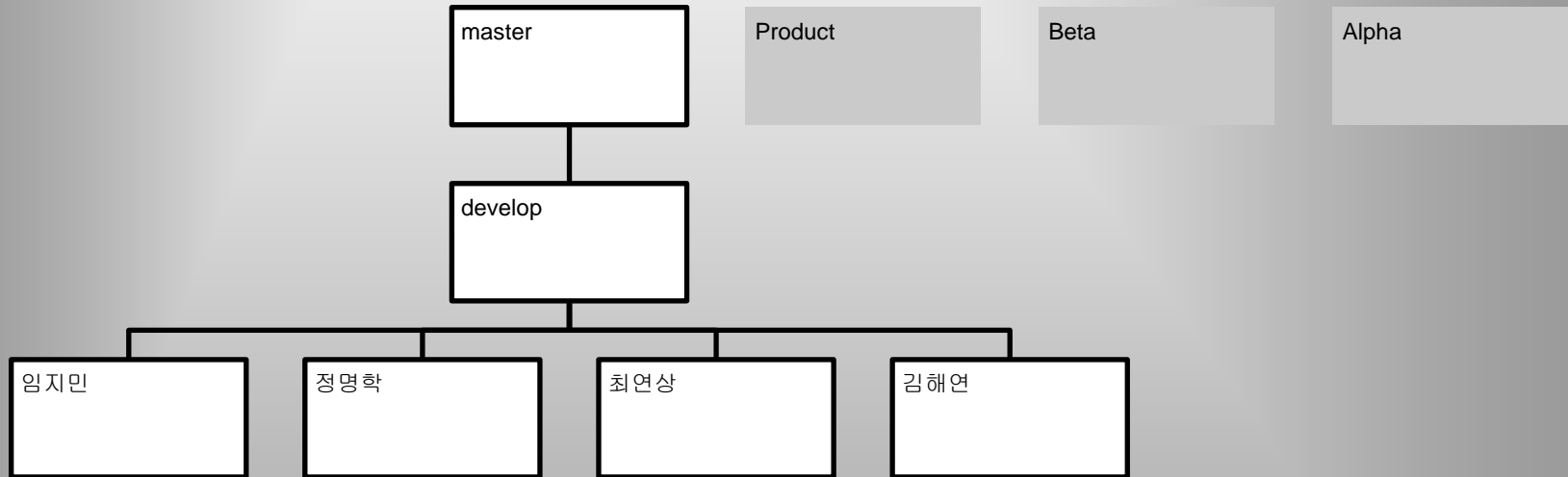
- SHA1 ID: 0c3ad239ca58470a9b775318d8624bb72a6b49c7
- Author: Jimin Lim <jimin4026@naver.com> 2014-03-13 19:10:54
- Committer: Jimin Lim <jimin4026@naver.com> 2014-03-13 19:10:54
- Branch: master
- Follows: (empty)
- Precedes: (empty)
- test 커밋용 첫번째 파일 - by Mr.임
- Signed-off-by: Jimin Lim <jimin4026@naver.com>

The file content for 'text1.txt' is displayed below the commit details, showing a diff view with green lines indicating new content.

# Workflow

# Branch strategy

# Workflow – Branch strategy



단계별 Branch 생성, Branch는 많이 있어도 지나침이 없다.

Develop Branch에서는 Alpha 레벨의 모든 브랜치를 종합하며 Develop Branch의 개발과 테스트가 완료되면, master Branch로 관리

# Workflow - Init

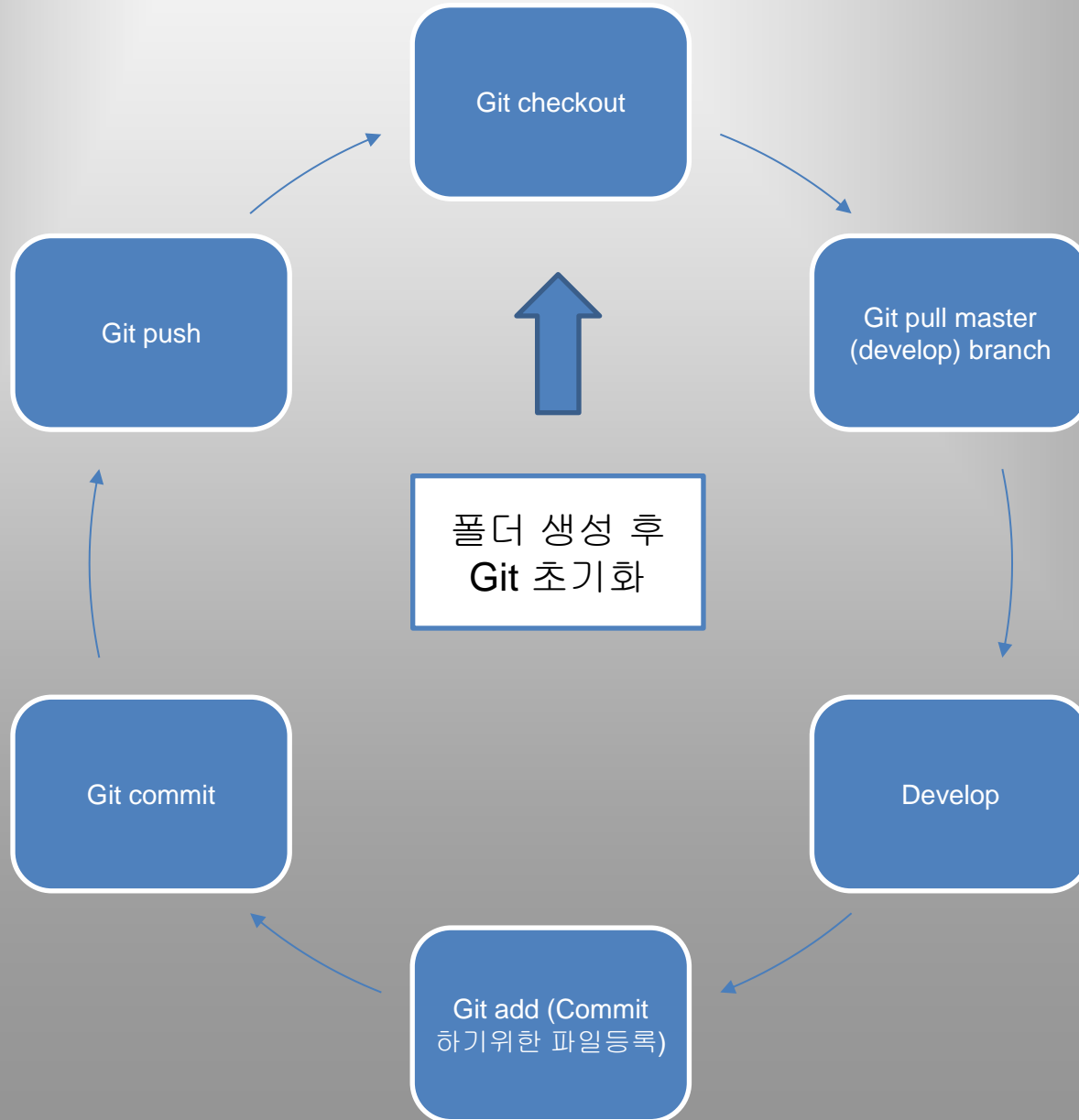
작업 폴더  
생성

```
>> git init  
>> git remote add <별칭> <git저장소주소>  
    ( git remote add origin https://github.com/Family-of-loves/TestLab.git )
```

OR

```
>> git clone https://github.com/Family-of-loves/TestLab.git
```

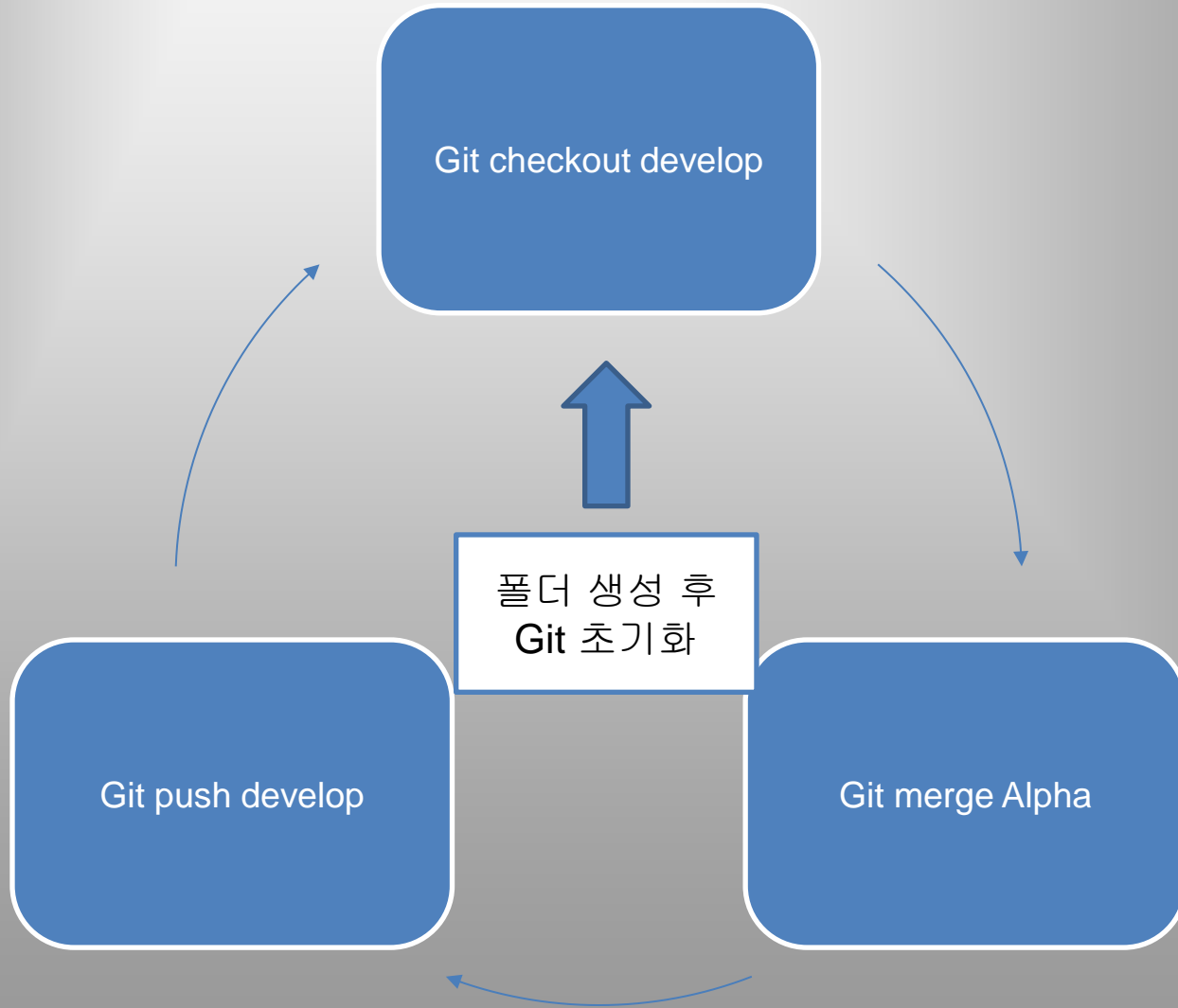
# Workflow - Employee



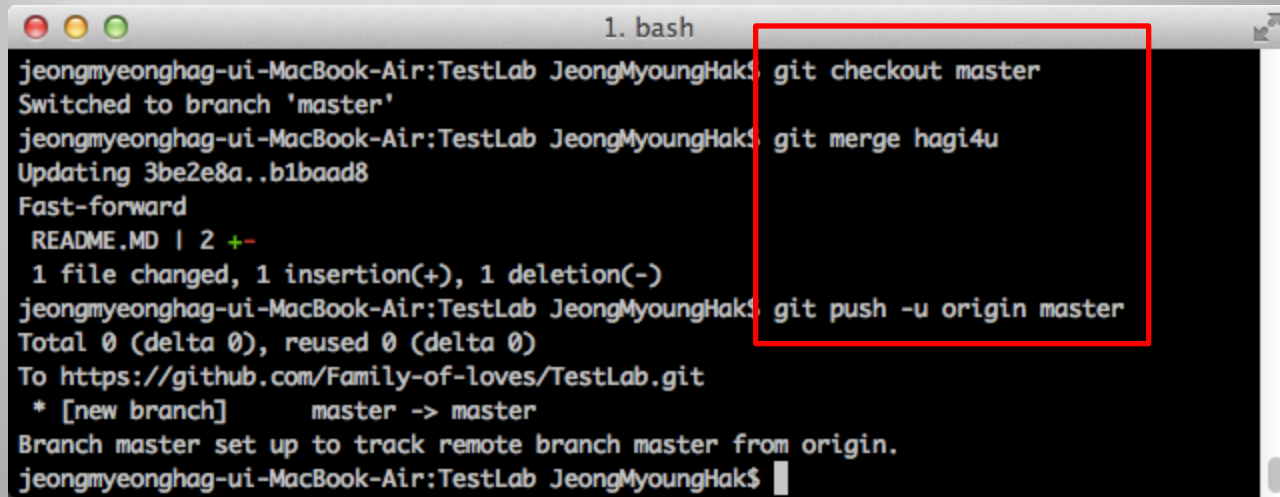
# Workflow - Employee

```
1. bash
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git checkout hagi4u
Switched to branch 'hagi4u'
Your branch is up-to-date with 'origin/hagi4u'.
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git pull -u origin hagi4u
From https://github.com/Family-of-loves/TestLab
* branch          hagi4u      -> FETCH_HEAD
Already up-to-date.
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ vi README.MD
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git add .
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git commit -m "Commit MSG!"
-bash: !": event not found
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git commit -m "Commit MSG"
[hagi4u b1baad8] Commit MSG
1 file changed, 1 insertion(+), 1 deletion(-)
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git push -u origin hagi4u
Counting objects: 6, done.
Writing objects: 100% (3/3), 285 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Family-of-loves/TestLab.git
3be2e8a..b1baad8 hagi4u -> hagi4u
Branch hagi4u set up to track remote branch hagi4u from origin.
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$
```

# Workflow - Manager



# Workflow - Manager

A terminal window titled "1. bash" with a standard macOS window header (red, yellow, green buttons). The terminal shows a sequence of Git commands and their output. A red rectangular box highlights the commands "git checkout master", "git merge hagi4u", and "git push -u origin master". The output shows a fast-forward merge of the hagi4u branch into master, updating the README.md file.

```
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git checkout master
Switched to branch 'master'
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git merge hagi4u
Updating 3be2e8a..b1baad8
Fast-forward
 README.MD | 2 + -
 1 file changed, 1 insertion(+), 1 deletion(-)
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git push -u origin master
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/Family-of-loves/TestLab.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$
```

# 상황 별 Tips

코딩이 산으로 갔을 때, 이전 **Commit** 상태로 되돌리기

# 코딩이 산으로 갔을 때, 이전 Commit 상태로 되돌리기

코드를 작성하다가 실수로 잘못된 수정들을 많이 해서 문제가 생겼을 경우, 이것들을 지난 **Commit** 상태로 돌려놓고 다시 코드를 테스트해보고 싶은 경우들이 많이 있다.

`git reset--hard HEAD~3`



현재 저장소의 **HEAD**로부터 과거  
3번째 **Commit**으로 돌아간다.

약 변경하려는 로컬 저장소의 **Commit**이 이미 **remote** 저장소에 **push**가 된 상태라면?

현재 작업중인 내용을 저장소 상태를 저장해두고  
이전 **Commit** 상태로 돌리기

# 현재 작업중인 내용을 저장소 상태를 저장해두고 이전 **Commit** 상태로 돌리기

코드를 작성하다 보면 새로 작성했거나 수정한 코드에 뭔가 문제가 생겨서 부분적으로 동작이 하지 않는 경우가 생긴다. 이때 어떤 부분이 문제가 되는지 확인하거나, 기존코드에서도 동일한 문제가 생기는지 확인하려면 저장소 상태를 이전 **Commit** 상태로 **checkout** 하게된다.

---

이때 새로 작성한 부분들 또한 잃어버리지 않고 안전하게 보관해 두려면 어떻게 해야 할까? (혹은 **git stash save**)      현재 상태 저장

|                        |   |
|------------------------|---|
| <b>git stash pop</b>   | 저장된 가장 최근 상태를 불러오고, 해당 <b>stash</b> 삭제              |
| <b>git stash apply</b> | <b>Stash</b> 적용 후에도 해당 <b>stash</b> 를 보관 하고 싶을 때 사용 |
| <b>git stash clear</b> | <b>Stash</b> 된 내용들을 모두 삭제                           |

---

‘**stash**’란 ‘안전한 곳에 넣어두다’라는 뜻을 가졌다. 이 말 뜻 그대로 **stash** 명령은 현재 변경 상태를 잠시 안전하게 보관해 두었다가 나중에 필요할때 꺼내서 사용할 수 있게 해주는 기능을 의미한다.

Problem

# Conflict

원격 저장소와 로컬 저장소의 commit 정보가 다른 경우  
생기는 문제

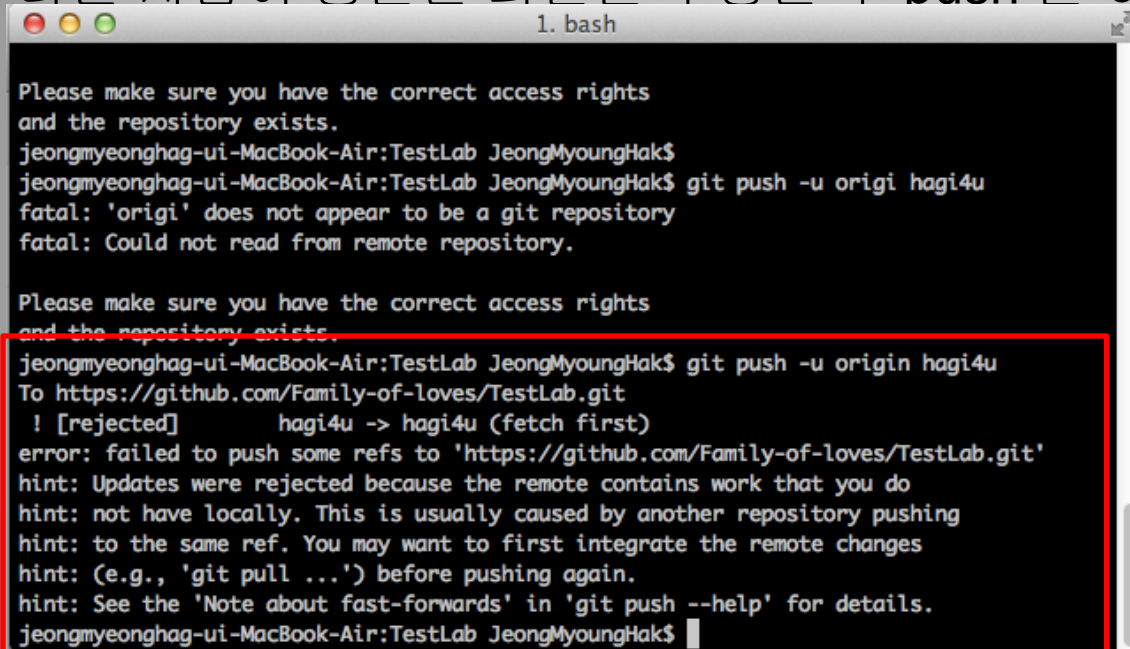
# Problem - Conflict

원격 저장소에 push 할 경우 발생할 수 있는 conflict (rejected)

조

같은 사람이 하나의 파일을 동시에 수정 중이며 한 사람이 Push 하고

다른 사람이 동일한 파일을 수정한 후 push 를 하게 되면

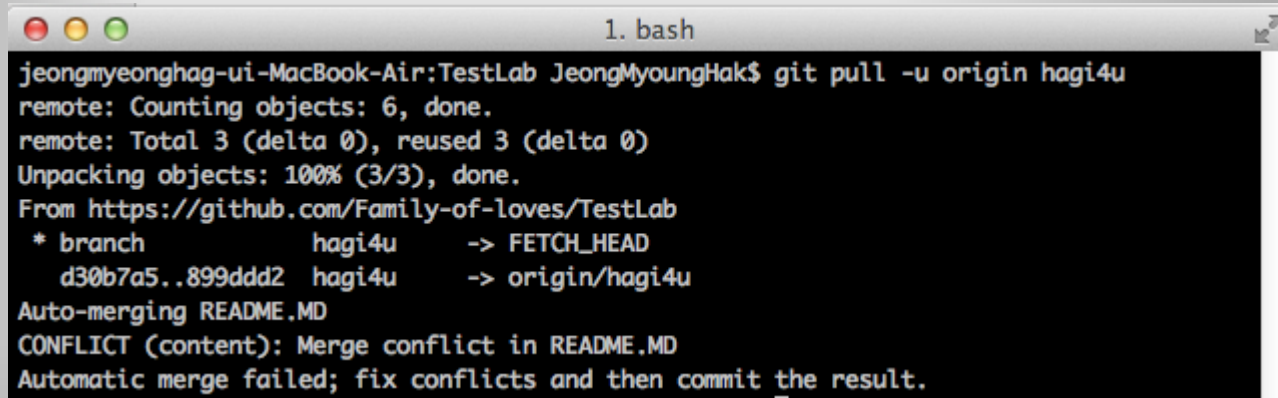
A terminal window titled '1. bash' showing a sequence of git commands and their outputs. The first command is 'git push -u origi hagi4u', which fails with a 'fatal: 'origi' does not appear to be a git repository' error. The second command is 'git push -u origin hagi4u', which fails with a 'rejected' error because the remote contains work that is not locally available. The terminal text is as follows:

```
1. bash
Please make sure you have the correct access rights
and the repository exists.
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git push -u origi hagi4u
fatal: 'origi' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git push -u origin hagi4u
To https://github.com/Family-of-loves/TestLab.git
! [rejected]        hagi4u -> hagi4u (fetch first)
error: failed to push some refs to 'https://github.com/Family-of-loves/TestLab.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.
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$
```

Solution

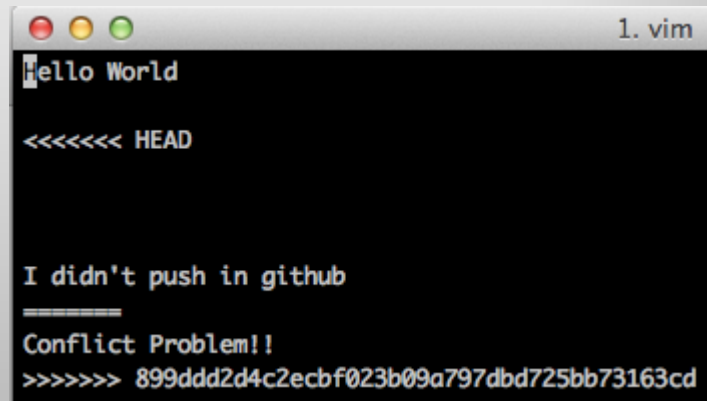
# Problem - Solution



```
1. bash
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git pull -u origin hagi4u
remote: Counting objects: 6, done.
remote: Total 3 (delta 0), reused 3 (delta 0)
Unpacking objects: 100% (3/3), done.
From https://github.com/Family-of-loves/TestLab
* branch          hagi4u      -> FETCH_HEAD
   d30b7a5..899ddd2 hagi4u      -> origin/hagi4u
Auto-merging README.MD
CONFLICT (content): Merge conflict in README.MD
Automatic merge failed; fix conflicts and then commit the result.
```

Github 에 Commit 된 내용을 로컬 저장소과 일치  
시켜야함.

# Problem - Solution



```
1. vim
Hello World

<<<<<<<< HEAD

I didn't push in github
=====
Conflict Problem!!
>>>>>>>> 899ddd2d4c2ecbf023b09a797dbd725bb73163cd
```

충돌나는 부분

편집

<<<<<<<<<

Push에 실패한 파일의 내용

=====

Commit이 되어있던 파일의 내용

>>>>>>>> Commit 시 생성 되었던  
해시코드

여러군대라면 각각 위 형식처럼 표시됨

# Problem - Solution

```
1. bash
Automatic merge failed; fix conflicts and then commit the result.
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ vi README.MD
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git add .
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git commit -m "Conflict fixed"
[hagi4u 3be2e8a] Conflict fixed
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$ git push -u origin hagi4u
Counting objects: 9, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 477 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/Family-of-loves/TestLab.git
   899ddd2..3be2e8a  hagi4u -> hagi4u
Branch hagi4u set up to track remote branch hagi4u from origin.
jeongmyeonghag-ui-MacBook-Air:TestLab JeongMyoungHak$
```

편집 후 commit 과 push 작업을  
진행

항상 작업 전 **pull** 작업을 하세요.

Q & A

감사합니다.