



Open Source Version 관리 Git

구명완교수

Office: K관 239호

Email: mwkoo9@gmail.com

Reference

- 참고 자료

Git 메뉴얼 (한글)

<http://git-scm.com/book/ko>

Learn Git Branching

<http://pcottle.github.io/learnGitBranching/>

Code School - Try git

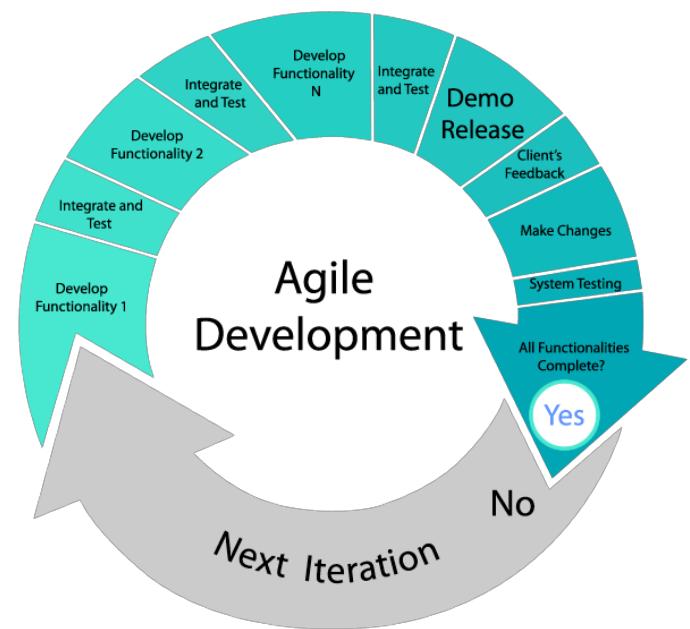
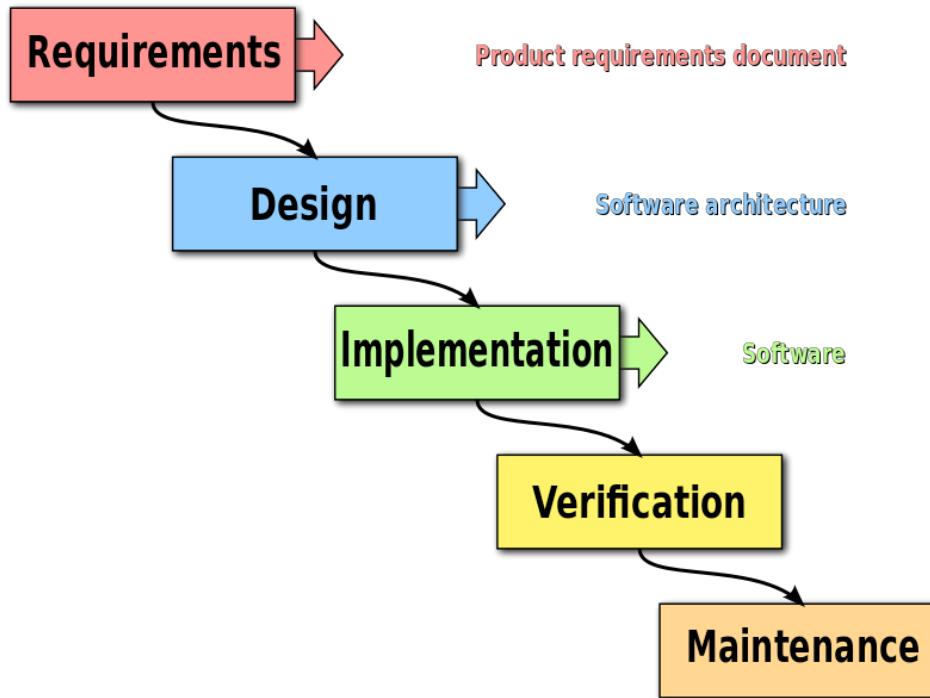
<https://try.github.io/levels/1/challenges/1>

Atlassian git tutorial

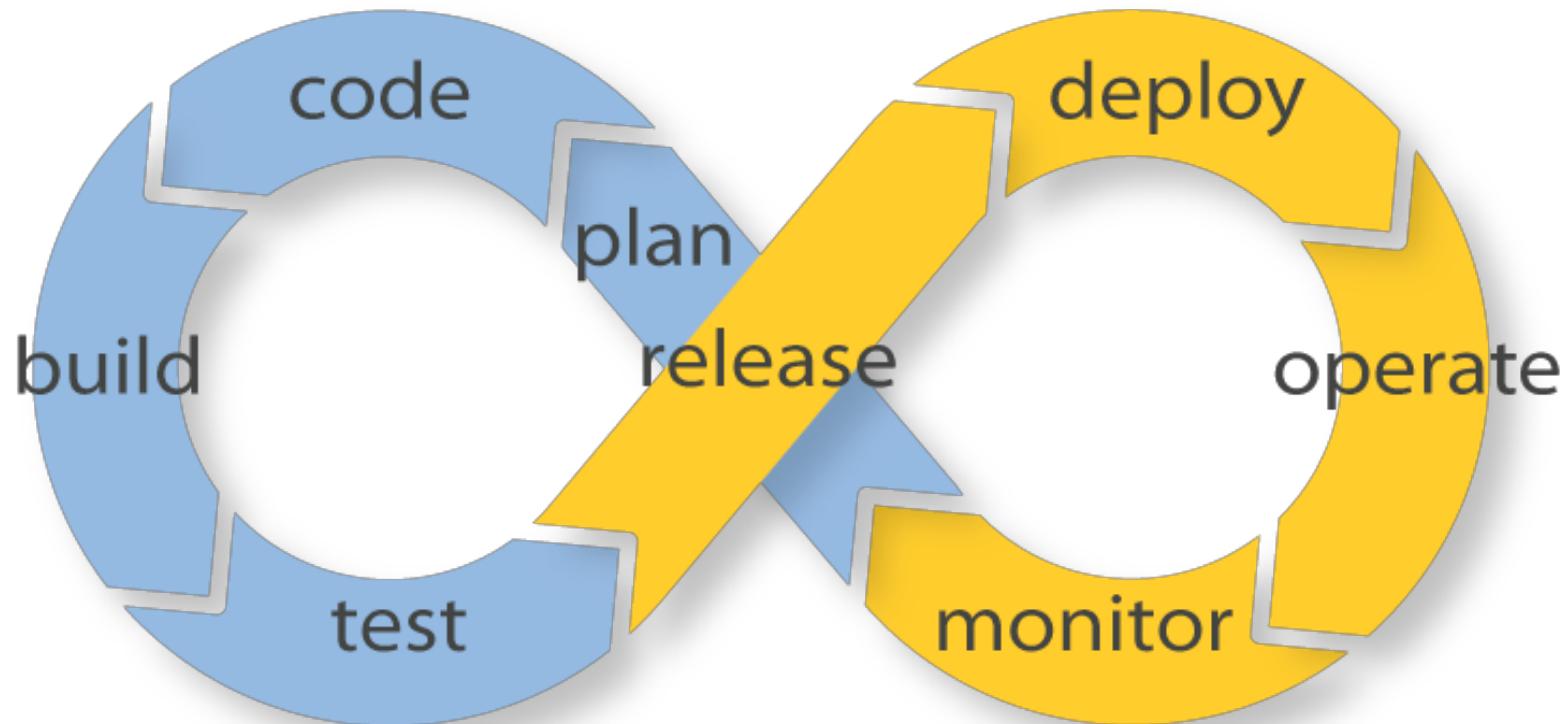
<https://www.atlassian.com/git/tutorial>



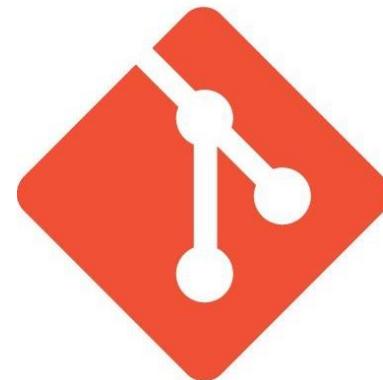
Waterfall vs. Agile



DevOps



Git & GitHub



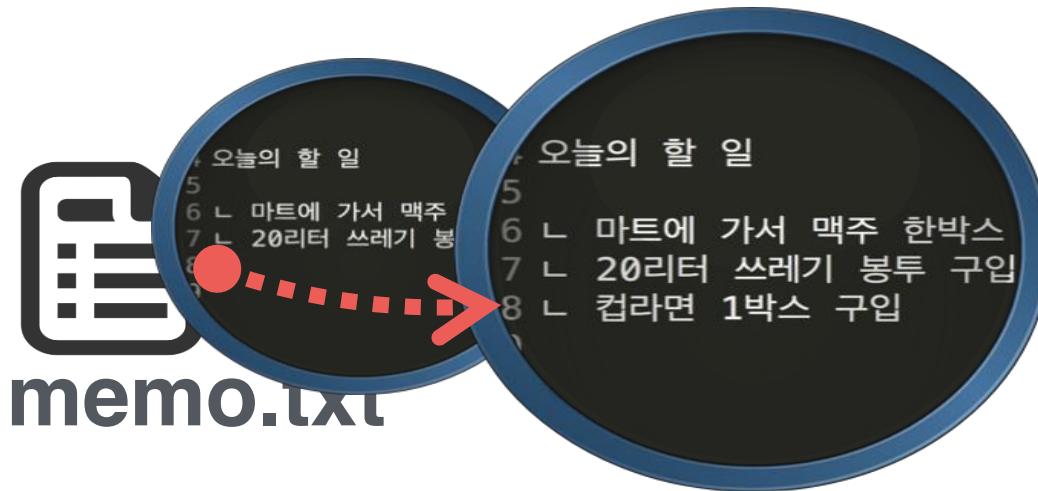
git

GitHub



Git 기본 개념

- Memo.txt 변경



“컵라면 1박스 구입”이라는 수정이 발생

Git 기본 개념

#Git - Commit

commit 3



2014년 7월 10일 17:00

컵라면 구매 할 일 추가
@홍길동



memo.txt

commit 2

2014년 7월 9일 10:00

쓰레기 봉투 구매 할 일 추가
@홍길동



memo.txt

commit 1

2014년 7월 6일 12:42

맥주 구매 할 일 추가
@홍길동



memo.txt

오늘의 할 일

- 5
- 6 ↘ 마트에 가서 맥주 한박
- 7 ↘ 20리터 쓰레기 봉투 구입
- 3 ↘ 컵라면 1박스 구입

오늘의 할 일

- 5
- 6 ↘ 마트에 가서 맥주 한박스
- 7 ↘ 20리터 쓰레기 봉투 구입

오늘의 할 일

- 5
- 6 ↘ 마트에 가서 맥주 한박스
- 7 ↘ 20리터 쓰레기 봉투 구입



Git 기본 개념

#Git - Commit [File Tree]

commit 3

2014년 7월 10일 17:00

수,목,금 교육 시간 수정
@홍길동



수요일.txt 목요일.txt 금요일.txt

commit 2

2014년 7월 9일 10:03

수요일 학습 내용 추가
@홍길동



수요일.txt

commit 1

2014년 7월 6일 12:42

월,화 자율 학습 일정 추가
@홍길동



월요일.txt 화요일.txt



#Git - Branch

commit 2



2014년 7월 9일 10:03

수요일 학습 내용 추가

@홍길동



수요일.txt

commit 1

2014년 7월 6일 12:42

월,화 자율 학습 일정 추가

@홍길동



월요일.txt 화요일.txt

master



#Git - Branch

branch lab1

lab1



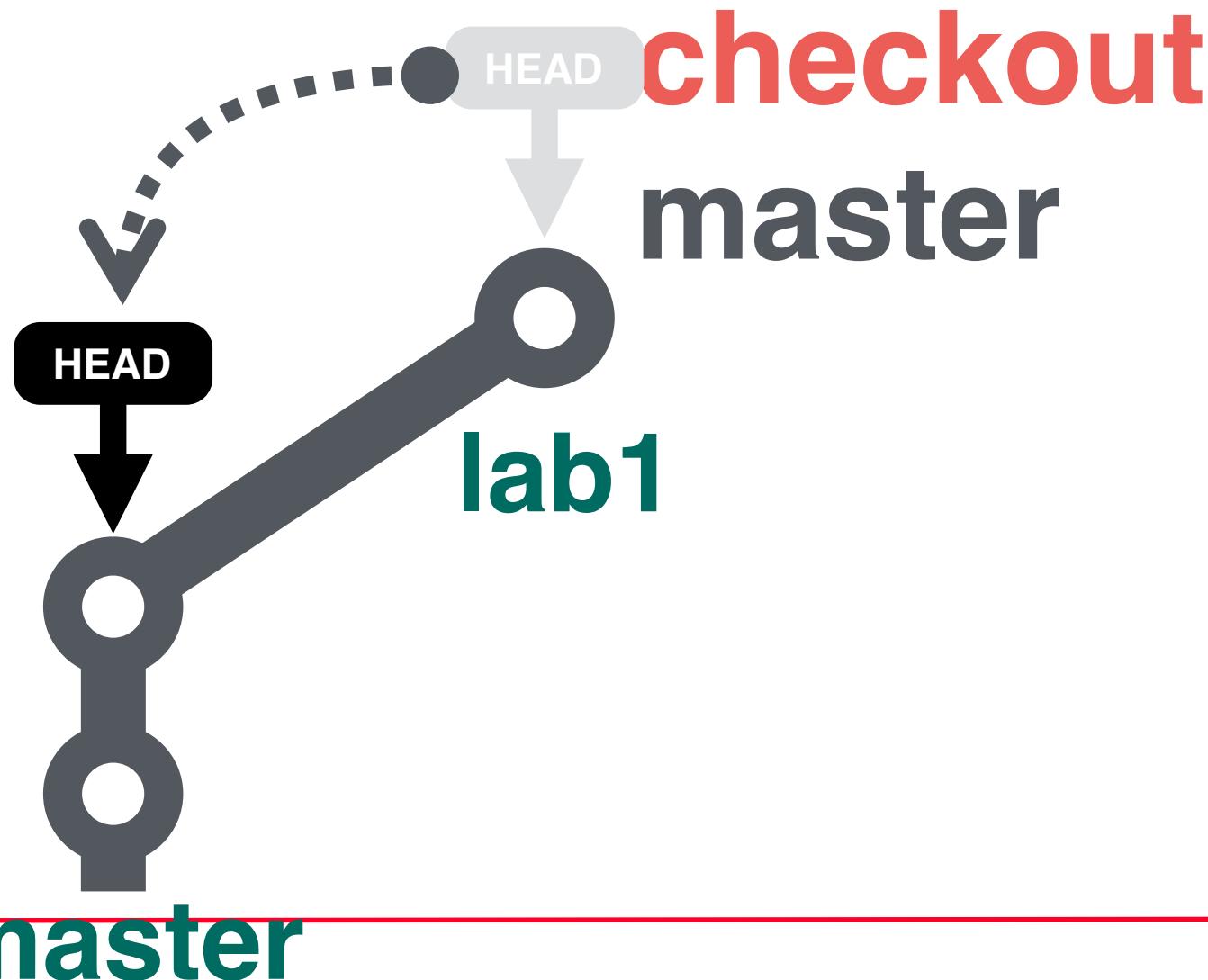
월요일.txt 화요일.txt 수요일.txt 목요일.txt 금요일.txt



월요일.txt 화요일.txt 수요일.txt 목요일.txt 금요일.txt



#Git - Checkout



#Git - Delete Branch



실험 실패
lab1에서 진행했던
실험이 예상과 달리
필요없는 작업이 되었다.
어떻게 하면 될까?
master로 이동 후 **lab1**
브랜치를 삭제한다. **lab1**의
모든 기록이 제거된다. 기록
보관 차원에서 삭제하지 않아도
아무 문제 없다

#Git - Merge



master

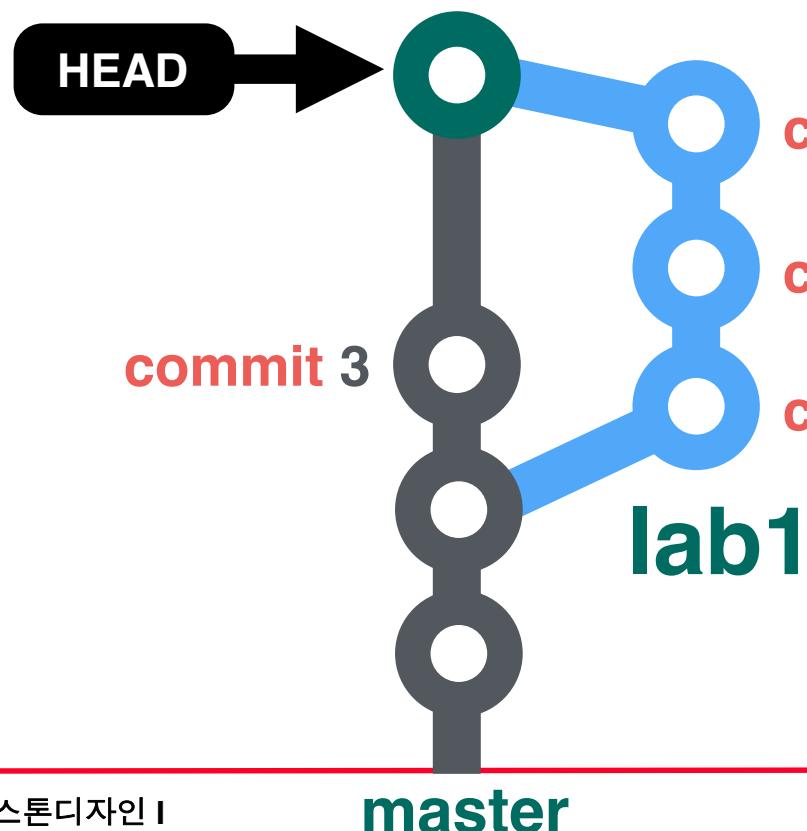
실험 성공

lab1에서 진행했던 실험이 성공적으로 끝났다. 실험의 결과를 master 브랜치에 옮기려한다. 어떻게 하면 될까?

lab1 브랜치의 내용을 마스터 브랜치와 병합(merge) 한다.



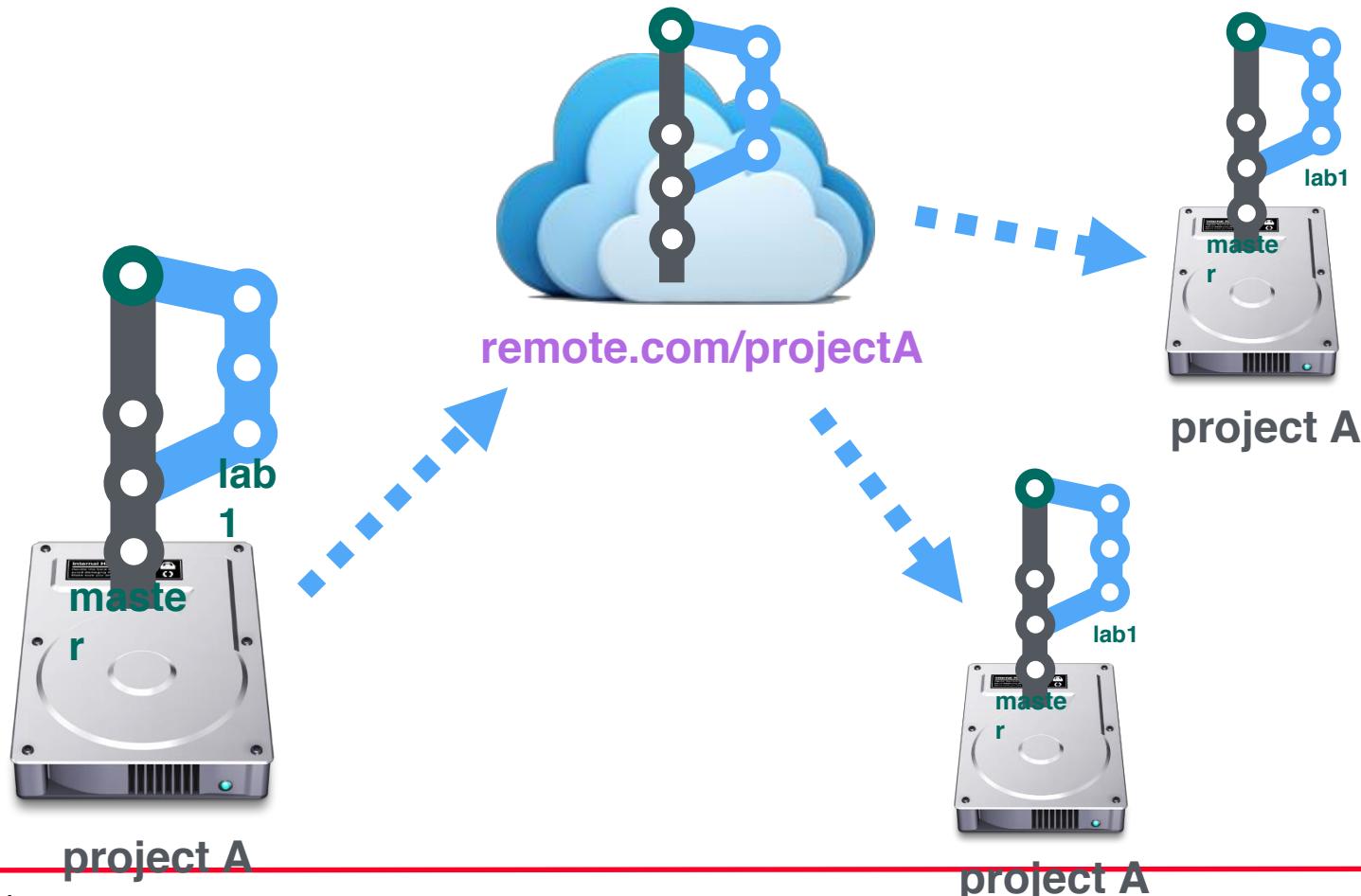
#Git - Merge



병합 결과

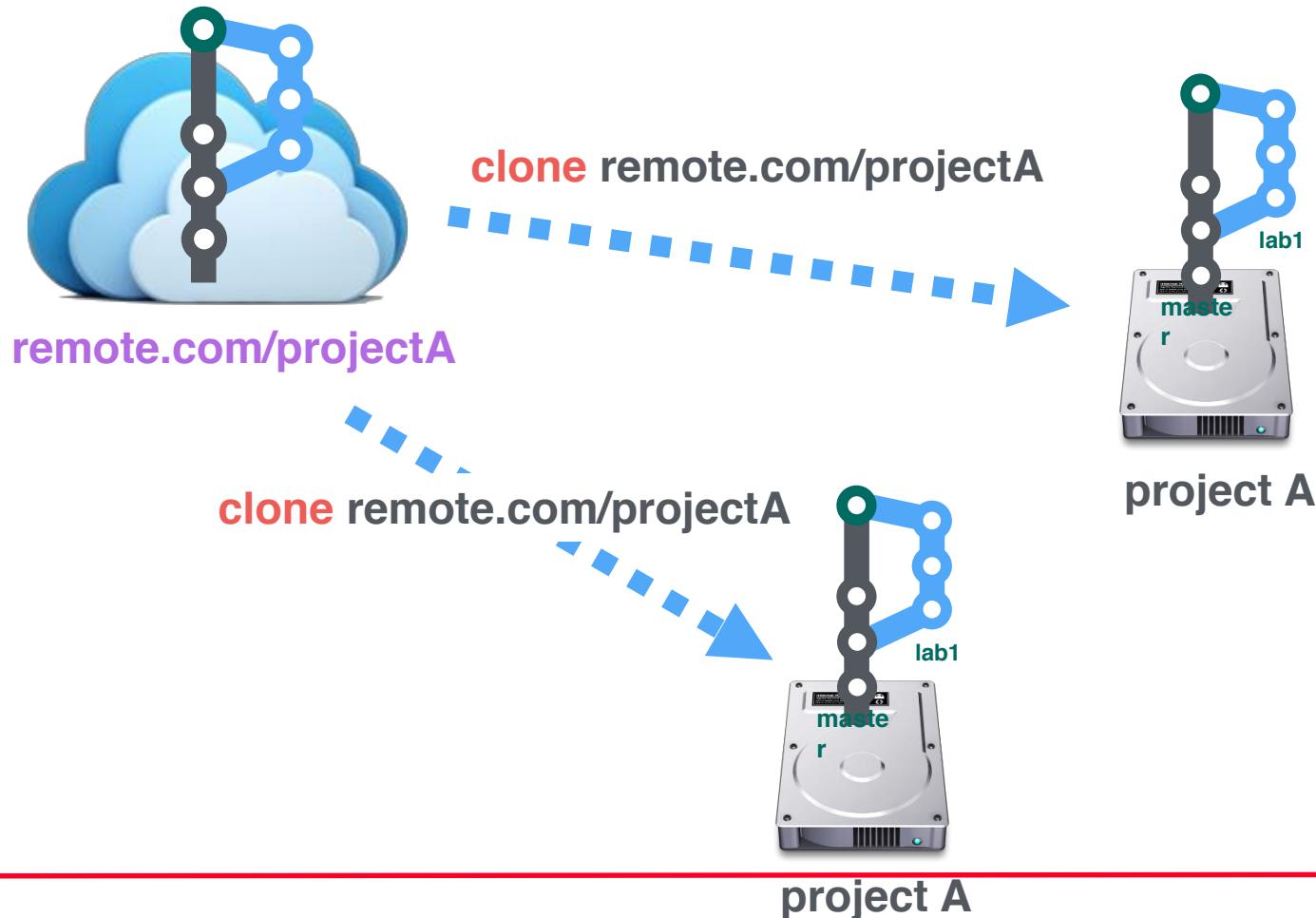
master 브랜치에 **lab1** 브랜치를 병합하면 **git** 은 **lab1** 브랜치의 내용과 **master** 브랜치의 **commit** 3 의 내용을 포함하여 두 브랜치를 병합한다
변경 내용에 따라 파일 내용이 변경되고 때론 파일이 삭제될 수도 있으며 추가될 수도 있다.

#Git - Remote Repo



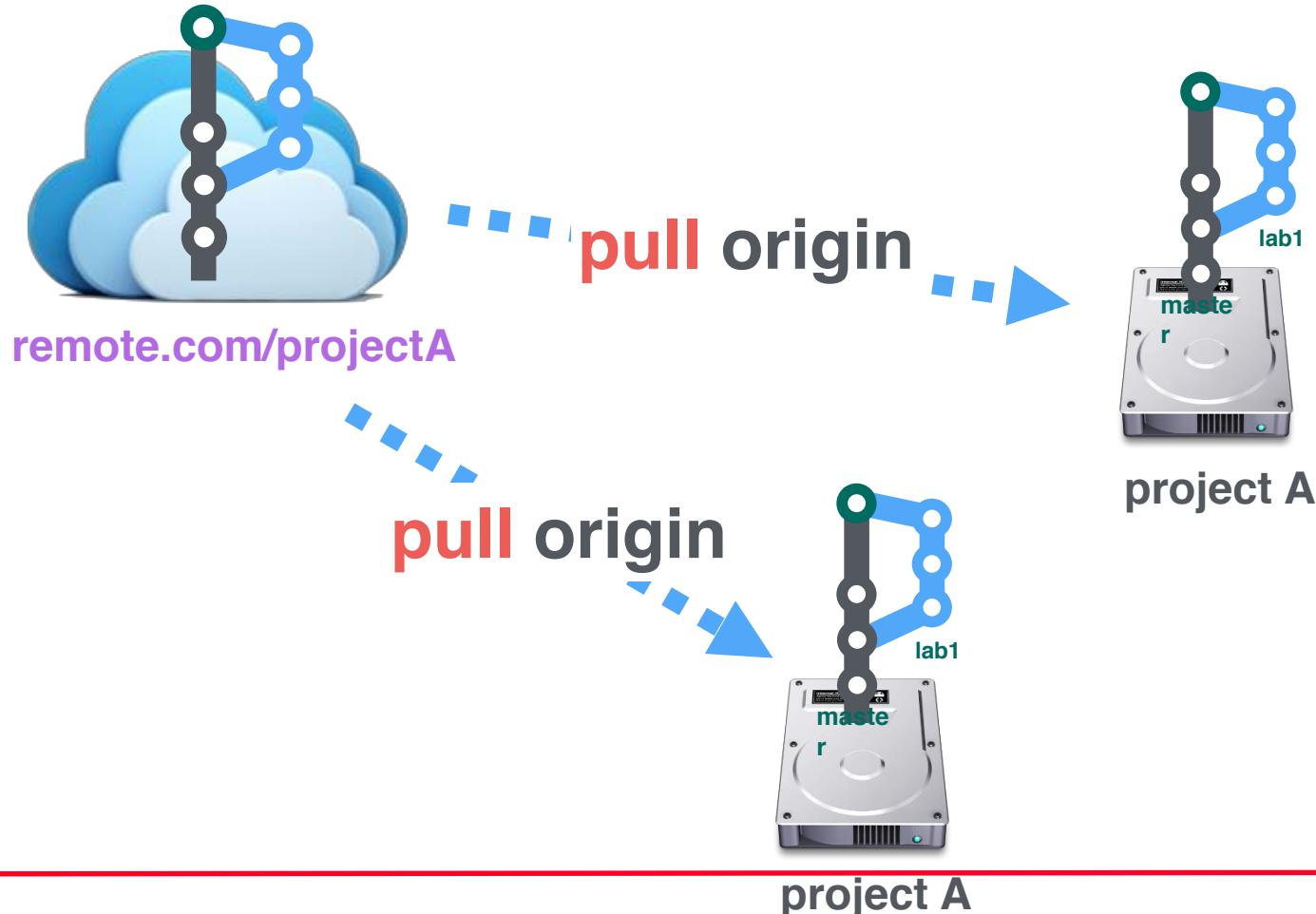
GitHub 기본 개념

#Git - Clone

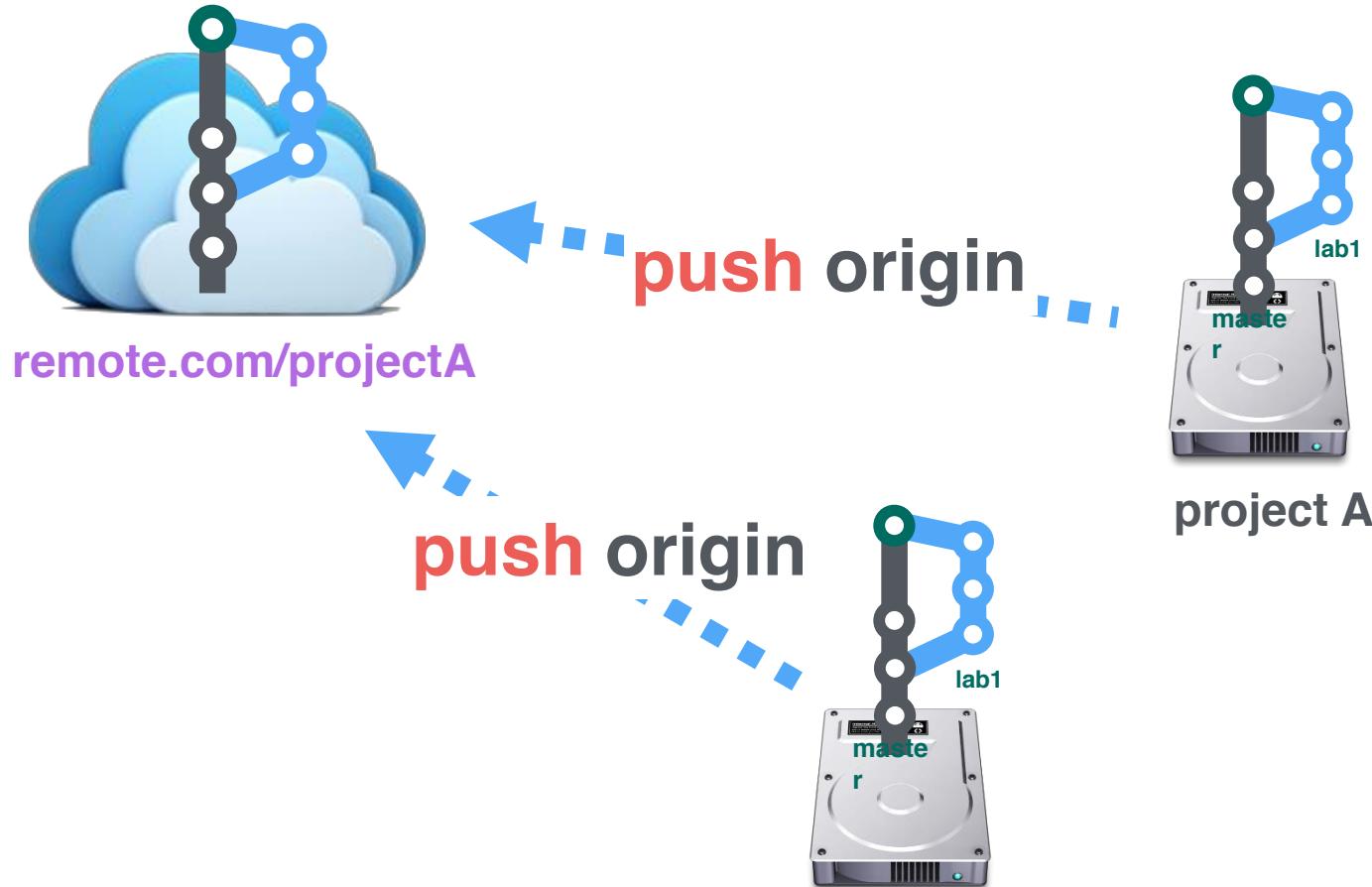


GitHub 기본 개념

#Git - Pull



#Git - Push



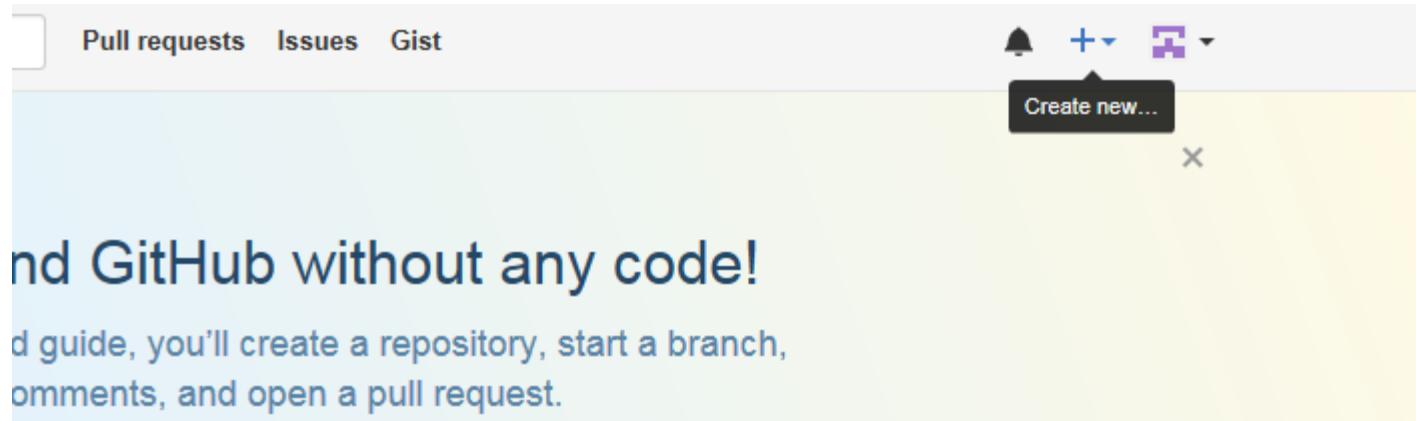
- GitHub is a platform for hosting and collaborating on projects
 - a collaborative and asynchronous workflow for building software better, together
- Hello World project
 - a time-honored tradition in computer programming
- Repositories, Branches, Commits, Issues and Pull Requests.
- Install & Code Free Zone
 - need a GitHub account

- Create a Repository

- repository is the basic unit of GitHub
- contain folders and files, including images
- recommend including a README
- also offers other common options such as a license file

- To create a new repository

- 1.Click the + icon next to your username, top-right



- Name your repository **HelloWorld**
- Write a short description.
- Select Initialize this repository with a README

Create a new repository

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

Owner



Repository name

hansori999

HelloWorld



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

Description (optional)

HelloWorld



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 is created

The screenshot shows a GitHub repository page for the user 'hansori999' named 'HelloWorld'. The repository has 1 commit, 1 branch, 0 releases, and 1 contributor. The 'Code' tab is selected. A green button labeled 'New pull request' is visible. The repository contains a single file, 'README.md', which displays the text 'HelloWorld'.

hansori999 / HelloWorld

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

HelloWorld — Edit

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request New file Find file HTTPS https://github.com/hansor Download ZIP

hansori999 Initial commit Latest commit 183fa51 a minute ago

README.md Initial commit a minute ago

README.md

HelloWorld

HelloWorld



- Open an Issue

- Issue is a note on a repository about something that needs attention
 - a bug, a feature request, a question or lots of other things
 - label, search and assign issues,
 - making managing an active project easier

- Open an Issue for README edits

- 1. Click the ! Issues tab from the sidebar

The screenshot shows a GitHub repository page for 'hansori999 / HelloWorld'. At the top, there's a red bar with the GitHub logo and the repository name. Below it, the repository details are shown: 'hansori999 / HelloWorld' with 1 unwatched, 0 stars, and 0 forks. The navigation tabs include 'Code', 'Issues 0' (which is highlighted in orange), 'Pull requests 0', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. Below the tabs, there are 'Filters' set to 'is:issue is:open', 'Labels' (empty), and 'Milestones' (empty). A prominent green button labeled 'New issue' is located on the right. The main content area is currently empty, showing a placeholder message: 'There are no issues yet.'

GitHub

- Click New Issue
- Give your Issue a title and description

The screenshot shows the GitHub interface for creating a new issue. At the top, there's a header with the repository name "hansori999 / HelloWorld" and buttons for "Unwatch" (with 1 watch), "Star" (0 stars), and "Fork" (0 forks). Below the header is a navigation bar with links for "Code", "Issues 0" (which is highlighted in orange), "Pull requests 0", "Wiki", "Pulse", "Graphs", and "Settings".

The main area is titled "Finish README" and contains a text editor with two tabs: "Write" (selected) and "Preview". The "Write" tab has a rich text editor toolbar with icons for bold, italic, underline, etc. A text input field contains the placeholder "Finish README|". Below the input field is a note: "Attach files by dragging & dropping or selecting them." At the bottom of the text editor area, it says "Styling with Markdown is supported". To the right of the text editor, there are three configuration sections: "Labels" (None yet), "Milestone" (No milestone), and "Assignee" (No one—assign yourself). At the bottom right of the entire form is a green "Submit new issue" button.

- Click Submit new Issue when you're done
 - a permanent home (URL) that you can reference



hansori999 / HelloWorld

Unwatch 1 Star 0 Fork 0

Code Issues 1 Pull requests 0 Wiki Pulse Graphs Settings

Finish README #1

Open hansori999 opened this issue just now · 0 comments

 hansori999 commented just now

Owner 

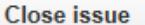
Finish README

 Write Preview 

Leave a comment

Attach files by dragging & dropping or [selecting them](#).

Styling with Markdown is supported

Labels None yet

Milestone No milestone

Assignee No one—assign yourself

Notifications 

You're receiving notifications because you authored the thread.

1 participant

- Next, work towards editing your README
- closing this issue

hansori999 / HelloWorld

Unwatch 1 Star 0 Fork 0

Code Issues 1 Pull requests 0 Wiki Pulse Graphs Settings

Finish README #1

Edit New issue

Closed hansori999 opened this issue 4 minutes ago · 0 comments

 hansori999 commented 4 minutes ago

Owner 

Finish README

  hansori999 closed this 2 minutes ago

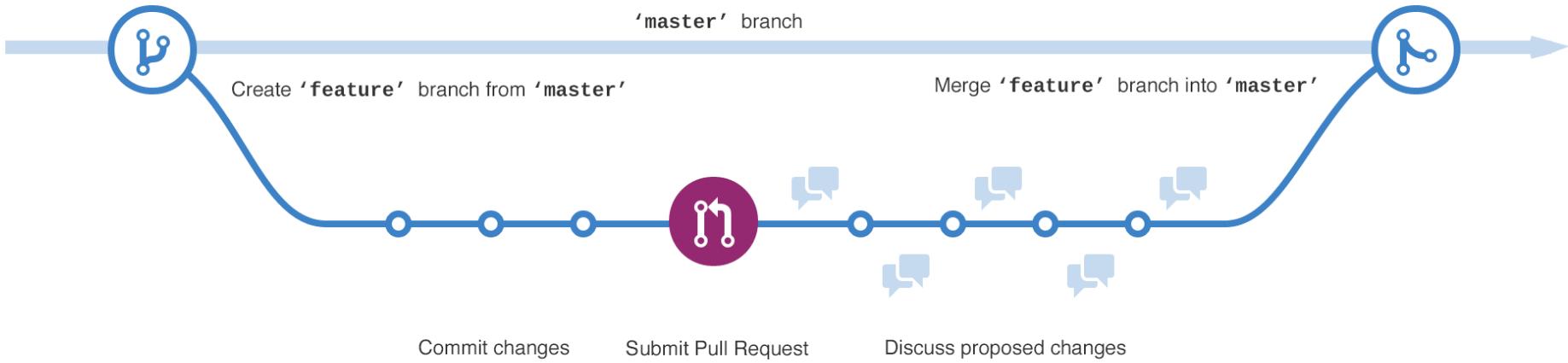
Labels 
None yet

Milestone 
No milestone

- Create a Branch

- the way to work on different parts of a repository at one time
- by default it has one branch “**master**” after creating repository
 - keep working on this branch and have only one
- if you have another feature or idea you want to work on
 - can **create another branch**, starting from master, so that you can leave master in its working state
 - making a copy of the original branch like a photo snapshot
- use branches for keeping bug fixes and feature work separate from our master (production) branch
 - When a feature or fix is ready,
 - the branch is merged into master.

- To create a new branch



- Go to your new repository **HelloWorld**
- Click the drop down at the top of the file list that says **branch: master**

GitHub

- Type a branch name, **ReadmeEdits**, into the new branch text box
- Select the blue **Create branch** box or hit “Enter” on your keyboard

The screenshot shows a GitHub repository page for 'hansori999 / HelloWorld'. The top navigation bar includes links for 'Pull requests', 'Issues', and 'Gist'. Below the repository name, there are buttons for 'Unwatch' (1), 'Star' (0), and 'Fork' (0). The main navigation tabs are 'Code', 'Issues 0', 'Pull requests 0', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The 'Code' tab is selected. Key statistics are displayed: 1 commit, 1 branch, 0 releases, and 1 contributor. A 'New pull request' button is visible. A modal window titled 'Switch branches/tags' is open, showing a text input field with 'ReadmeEdits' and a 'Create branch: ReadmeEdits from 'master'' button. The repository history shows an 'Initial commit' made 35 minutes ago. The URL is https://github.com/hansori999>HelloWorld.



- Now you have two branches, master and ReadmeEdits

The screenshot shows the GitHub repository page for 'hansori999 / HelloWorld'. At the top, there are buttons for Unwatch (1), Star (0), and Fork (0). Below the header, there are tabs for Code, Issues (0), Pull requests (0), Wiki, Pulse, Graphs, and Settings. The 'Code' tab is selected. Below the tabs, there are buttons for Overview (selected), Yours, Active, Stale, and All branches. A search bar for 'Search branches...' is also present. The main content area shows the 'Default branch' section, which lists the 'master' branch as 'Default'. There is a 'Change default branch' button. Below this, the 'Your branches' section lists the 'ReadmeEdits' branch, with a 'New pull request' button and a delete icon. Finally, the 'Active branches' section also lists the 'ReadmeEdits' branch, with a 'New pull request' button and a delete icon.

hansori999 / HelloWorld

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Wiki | Pulse | Graphs | Settings

Overview | Yours | Active | Stale | All branches | Search branches...

Default branch

master Updated 37 minutes ago by hansori999 | Default | Change default branch

Your branches

ReadmeEdits Updated 37 minutes ago by hansori999 | 0 | 0 | New pull request | Delete

Active branches

ReadmeEdits Updated 37 minutes ago by hansori999 | 0 | 0 | New pull request | Delete

- Make a commit

- Commits are saved changes On GitHub
- Each commit has an associated commit message
 - a description explaining why a particular change was made

- To commit changes

- Click the README file on Branch ReadMeEdits
- Click the pencil icon in the upper right corner of the file view to edit.

The screenshot shows a GitHub repository page for 'hansori999 / HelloWorld'. The top navigation bar includes links for Code, Issues (0), Pull requests (0), Wiki, Pulse, Graphs, and Settings. The repository name 'hansori999 / HelloWorld' is displayed, along with options to Unwatch (1), Star (0), or Fork (0). A dropdown menu shows the current branch is 'Read...'. Below this, the file 'HelloWorld / README.md' is selected. The commit history shows a single commit from 'hansori999' titled 'Update README.md' with the commit hash '06c5d6c' and a timestamp of '9 minutes ago'. There are also 'Find file' and 'Copy path' buttons.

GitHub

- In the editor, write some text, tell a bit about yourself
- Write a commit message that describes your changes

The screenshot shows a GitHub repository page for 'hansori999 / HelloWorld'. The repository has 1 issue, 0 pull requests, and 0 stars. The 'Code' tab is selected. Below it, the 'HelloWorld / README.md' file is open for editing. The code editor shows the following content:

```
1 # HelloWorld
2 HelloWorld
3 I am explaining about Commit at GitHub|
```

Below the editor, a 'Commit changes' dialog is open. It contains fields for 'Update README.md' and an optional 'extended description...'. At the bottom, there are two radio button options: one for committing directly to the 'ReadmeEdits' branch and another for creating a new branch and starting a pull request. A red horizontal bar is overlaid across the bottom of the dialog.

- Click Commit changes
- Changes in the README file on your **readme-edits branch** and now this branch contains different content and commits than master

The screenshot shows a GitHub repository page for 'hansori999 / HelloWorld'. The repository has 1 star and 0 forks. The navigation bar includes links for Code, Issues (0), Pull requests (0), Wiki, Pulse, Graphs, and Settings. The current branch is 'Read...'. The main content shows a single commit by 'hansori999' titled 'Update README.md' made 6 minutes ago. The commit message is 'HelloWorld I am explaining about Commit at GitHub'. Below the commit, there are options to Raw, Blame, or view History. The commit hash is 06c5d6c.

- **Open a Pull Request**

- allows you **to compare the content on two branches**
 - The changes, additions and subtractions, are shown in green and red and called **diffs**
- Use Pull Requests to start a discussion about commits (code review) even before the code is finished.
- a Pull Request in your own repository and merge it yourself

- Create a Pull Request for changes to the README
 - Click the Pull Request icon
 - click the green New pull request button.

The screenshot shows a GitHub repository page for 'hansori999 / HelloWorld'. The top navigation bar includes links for Code, Issues (0), Pull requests (0), Wiki, Pulse, Graphs, and Settings. The repository summary shows 2 commits, 2 branches, 0 releases, and 1 contributor. A section for 'Your recently pushed branches:' lists 'ReadmeEdits (about 1 hour ago)' with a 'Compare & pull request' button. Below this, a 'New pull request' button is highlighted in green. The main content area displays a branch status message: 'This branch is 1 commit ahead of master.' followed by a list of recent commits:

- hansori999 Update README.md** Latest commit 06c5d6c an hour ago
- README.md** Update README.md an hour ago

- Click the green New pull request button.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



The screenshot shows a commit history for a file named 'README.md' on February 2, 2016. The commit was made by 'hansori999' with the message 'Update README.md'. The commit hash is '06c5d6c'. Below the commit, it says 'Showing 1 changed file with 1 addition and 0 deletions.' There are 'Unified' and 'Split' buttons for viewing the diff. The diff itself shows the following code changes:

```
1 [green] README.md
...
1 1 @@ -1,2 +1,3 @@
2 2 # HelloWorld
3 3 HelloWorld
+I am explaining about Commit at GitHub
```

- Click **create pull request**

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub interface for creating a pull request. At the top, there are dropdown menus for 'base: master' and 'compare: ReadmeEdits'. A green checkmark indicates that the branches are 'Able to merge'. Below this, there's a large text input field containing the commit message 'Update README.md' with the commit details 'Fixed bugs #1'. To the right of the text area, there are sections for 'Labels' (None yet) and 'Milestone' (No milestone). Below the text area, there are tabs for 'Write' and 'Preview', and a toolbar with various icons for file operations.

- Merge your Pull Request
 - Click **Merge pull request**

Update README.md #2

[Open](#) hansori999 wants to merge 1 commit into `master` from `ReadmeEdits`

Conversation 0 Commits 1 Files changed 1

 hansori999 commented 7 minutes ago

Fixed bugs #1

 Update README.md 06c5d6c

Add more commits by pushing to the `ReadmeEdits` branch on [hansori999>HelloWorld](#).

 This branch has no conflicts with the base branch
Merging can be performed automatically.

[Merge pull request](#) You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

- Click **Confirm merge**

Update README.md #2

 Open hansori999 wants to merge 1 commit into master from ReadmeEdits

 Conversation 0  Commits 1  Files changed 1

 hansori999 commented 9 minutes ago

Fixed bugs #1

 Update README.md 06c5d6c

Add more commits by pushing to the ReadmeEdits branch on hansori999/HelloWorld.

 Merge pull request #2 from hansori999/ReadmeEdits

Update README.md

 Confirm merge 

- Go ahead and delete the branch **Delete branch**

Update README.md #2

Merged hansori999 merged 1 commit into master from ReadmeEdits 11 seconds ago

Conversation 0 Commits 1 Files changed 1

hansori999 commented 11 minutes ago

Fixed bugs #1

Update README.md 06c5d6c

hansori999 merged commit 53d1c74 into master 11 seconds ago

Revert

Avoid bugs by automatically running your tests.

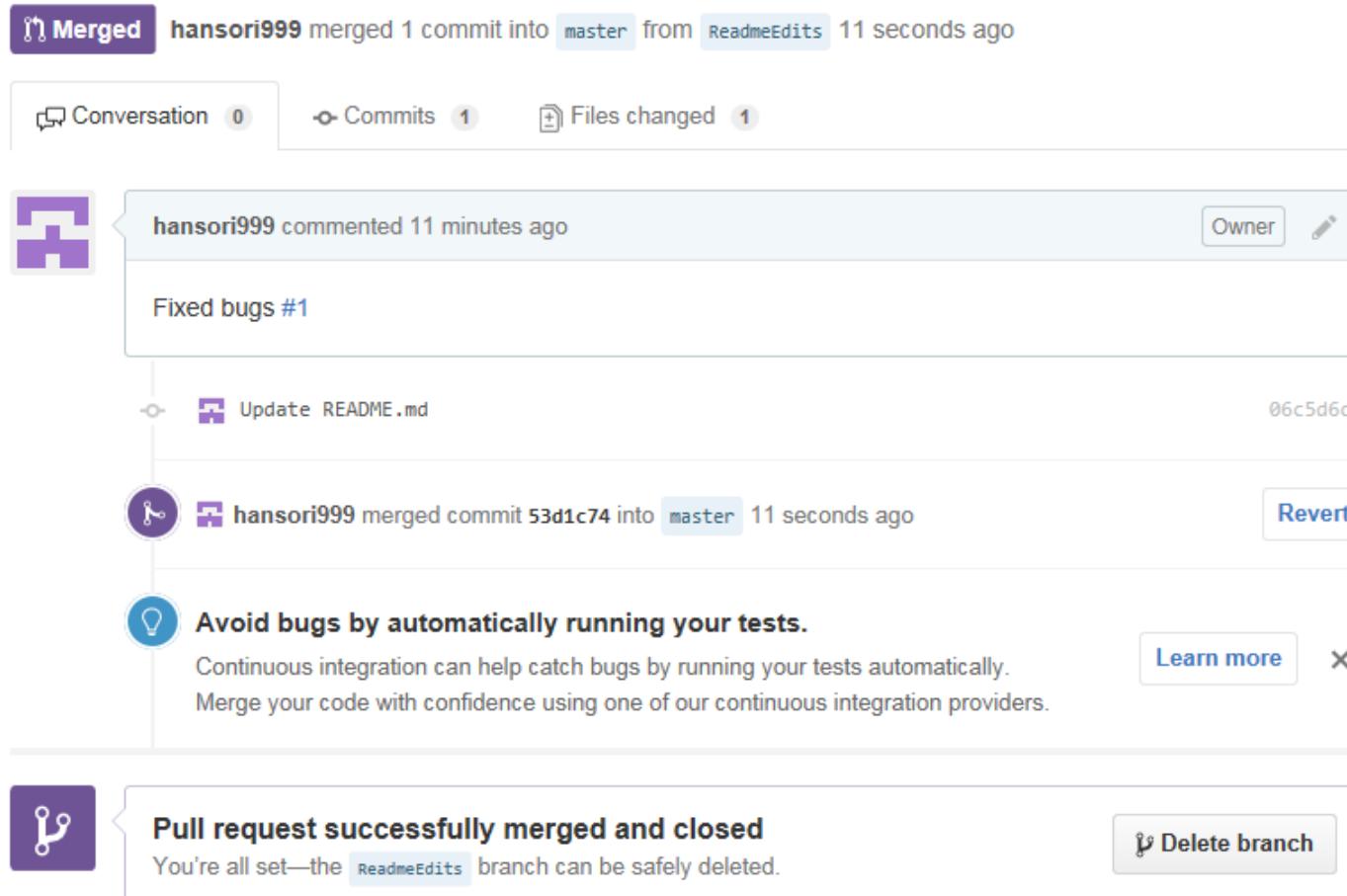
Continuous integration can help catch bugs by running your tests automatically.
Merge your code with confidence using one of our continuous integration providers.

Learn more X

Pull request successfully merged and closed

You're all set—the ReadmeEdits branch can be safely deleted.

Delete branch



● Final Result

Update README.md #2

 Merged [hansori999](#) merged 1 commit into [master](#) from [ReadmeEdits](#) a minute ago

 Conversation 0

 Commits 1

 Files changed 1



[hansori999](#) commented 12 minutes ago

Owner



Fixed bugs #1

 Update README.md

06c5d6c



 [hansori999](#) merged commit `53d1c74` into [master](#) a minute ago

Revert



Avoid bugs by automatically running your tests.

Continuous integration can help catch bugs by running your tests automatically.

[Learn more](#)



Merge your code with confidence using one of our continuous integration providers.



 [hansori999](#) deleted the [ReadmeEdits](#) branch just now

[Restore branch](#)



Git 기본

- **git help**
 - 기본적인 명령어 표시
- **git help -a**
 - 모든 명령어 표시
- **git help config 혹은 man git-config**
 - "config"라는 명령어 도움말 표시 사용자 이름과 이메일 설정
 - **git config --global user.name "Myoung-Wan Koo"**
 - **git config --global user.email mwkoo9@gmail.com**
- **설정확인**
 - **git config -list**
 - **git config user.name**



- Color 설정
 - `git config --global color.status auto`
 - `git config --global color.branch auto`
- Ignore 설정
 - Git에게 특정 파일을 무시하도록 하는 기능
 - 프로젝트의 root directory에 `.gitignore` 파일을 만들고, commit하고 싶지 않은 폴더나 파일명을 작성
 - 소스코드로 컴파일한 binary 파일이나, 대용량의 리소스 파일, 그리고 임시파일 등을 제외시키는 용도로 사용
 - 표준 Glob 패턴(정규표현식을 단순하게 만든 것)을 사용함

- How To : 프로젝트 다운 받아 개발시작하기

```
$ git clone http://github.com:hansori999/HelloWorld4.git      (1)  
$ cd HelloWorld4  
$ echo "first file" > first_file.txt  
$ git add first_file.txt  
$ git commit -m "add new file"                                (2)  
$ git push                                                 (3)
```

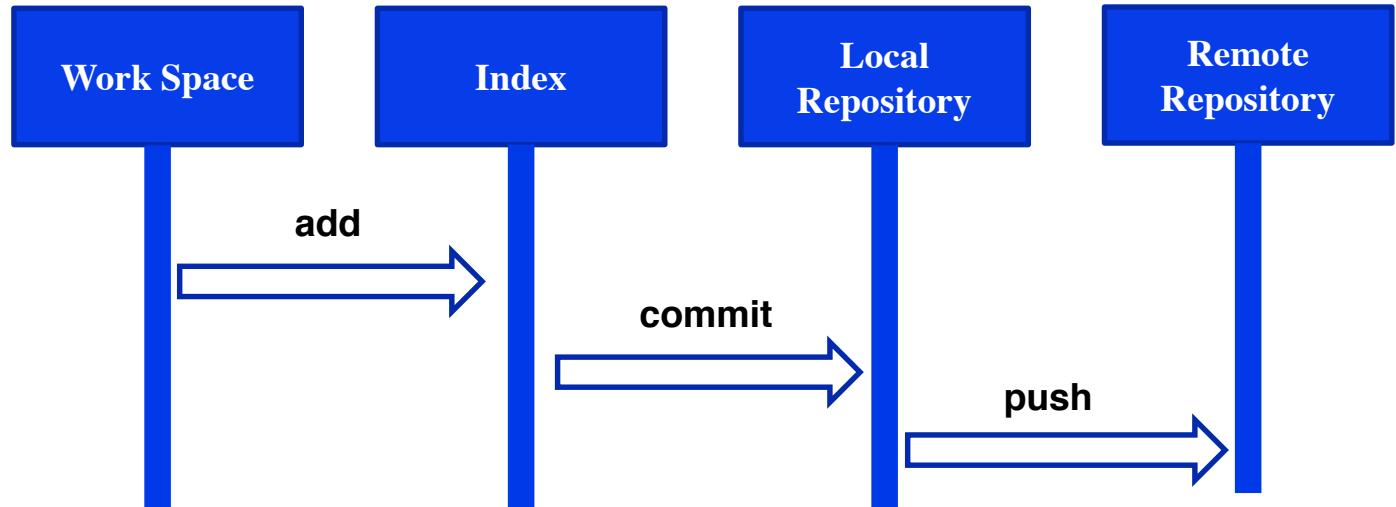
(1) 프로젝트 hansori999/HelloWord4.git를 HelloWorld 라는 폴더로 다운받는다.

(2) 변경된 것을 local repository에 commit하겠다.

(3) local repository에 commit 된 것을 remote repository에 반영하겠다.

- Work Space는 단지 개발하고 있는 파일시스템의 폴더일 뿐이다. ('.git' 폴더가 존재하는)
- 변경사항을 Local Repository에 적용.
- 그리고 이를 공유하기 위해 push로 Remote Repository(Server)에 올린다.

Git의 구성



- 앞의 예에서 ‘commit –a’라 한 것은 변경된 사항 전부를 commit의 대상으로 한다.
- 정확히는 commit의 대상은 오직 Index에 포함된 것만이다.
- ‘commit –a’는 변경된 사항 전부를 add하고 나서 commit을 실행한 것과 같다.
- 앞의 예에서는 전부 commit –a를 사용하였다. 설명의 간편함을 위한 것이다. 선별적으로 commit할 대상을 add하는 것이 바람직하다.

How To : 내가 수정한 것과 타인이 수정한 것을 merging하여 commit하기

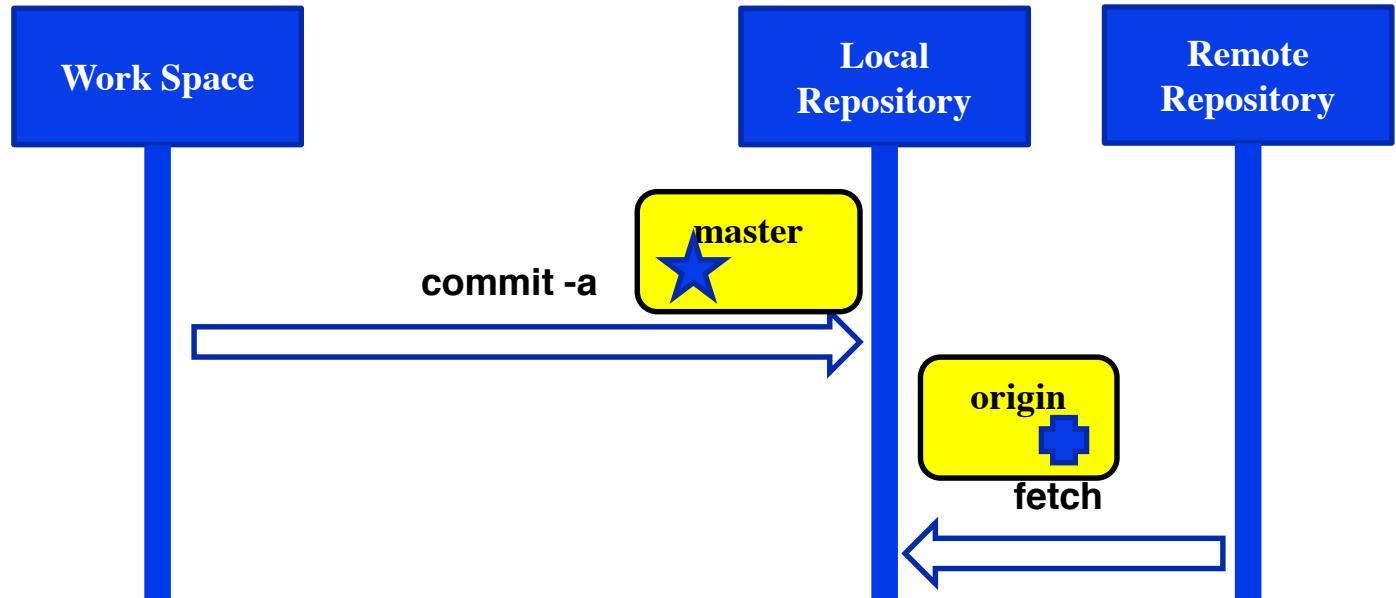
```
$ echo "some modification" >> first_file.txt (1)  
$ git commit -a -m "add first_file.txt" (2)  
$ git fetch (3)  
$ git diff origin  
$ git merge origin (4)  
$ git push (5)
```

commit의 -a 옵션은 git add 없이 모든 변경된 것을 대상으로 하라는 의미

- (1) 파일을 수정했다.
- (2) 변경된 사항 전부를 local repository에 commit한다.
- (3) remote repository에서 최신의 것을 local repository로 가져온다.
- (4) local repository에 가져온 최신의 것과 내가 commit한 내용을 merging한다.
- (3)/(4) git pull
- (5) merging된 사항을 remote repository에 올린다.

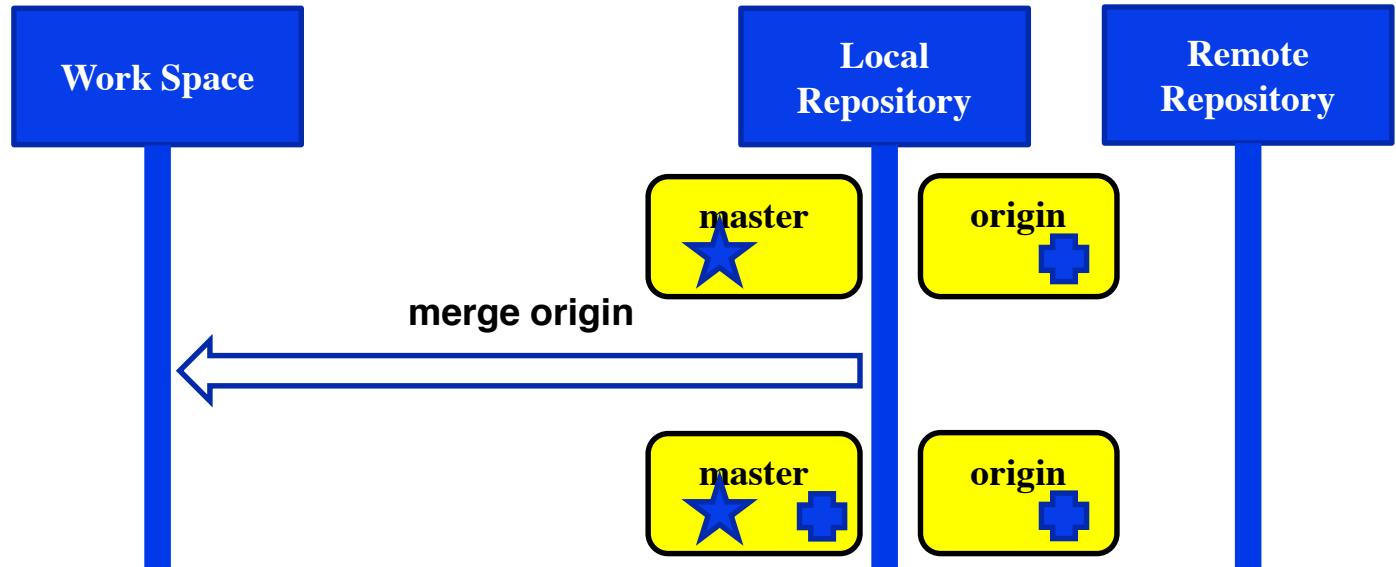


브랜치

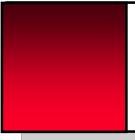


- Git는 브랜치라는 개념으로 작업본을 관리한다.
- 내가 commit한 본은 ‘master’라는 이름의 브랜치로, Remote Repository에서 가져온 것은 ‘origin’이라는 이름의 브랜치로 존재

브랜치 간 merge



- merge origin은 현재 작업중인 브랜치 master에 origin이라는 브랜치의 수정된 것을 병합하라는 것이다.
- merge origin을 실행한 후에는 origin의 수정 사항(십자가)가 HEAD에 반영되었다.
- 또한 merge된 결과는 다른 명령 없이 Work Space에 그대로 반영된다.



사용자가 정의하는 브랜치



How To : Local Repository에 새로운 branch 생성

```
$ ls first_file.txt
$ git checkout -b test          (1)
$ git branch
  master
* test
$ echo "my test" > test.txt    (3)
$ git status                     (4)
$ git add test.txt
$ git commit -m " Adding test"

$ git checkout master           (5)
$ git status                     (6)
```

(1) 새로운 브랜치 ‘test’를 생성하고, 그 브랜치로 변경한다.

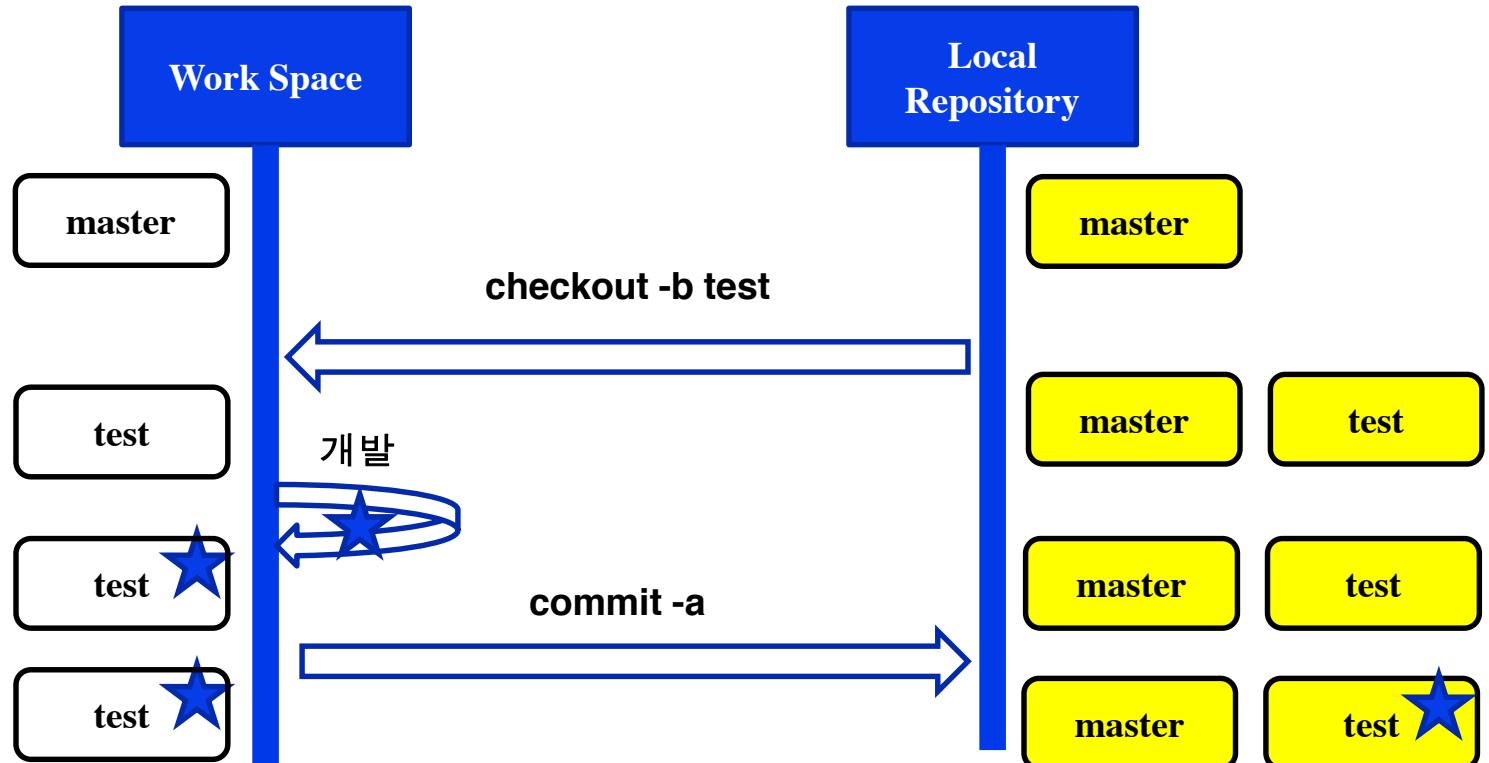
(2) 브랜치 리스트를 조회한다. 별표 (*)가 붙은 test 브랜치가 현재의 브랜치이다.

(3), (4) 새로운 파일 test.txt를 만들었다.

(5), (6) 기존의 브랜치 master로 변경했다..



branch 생성, 수정



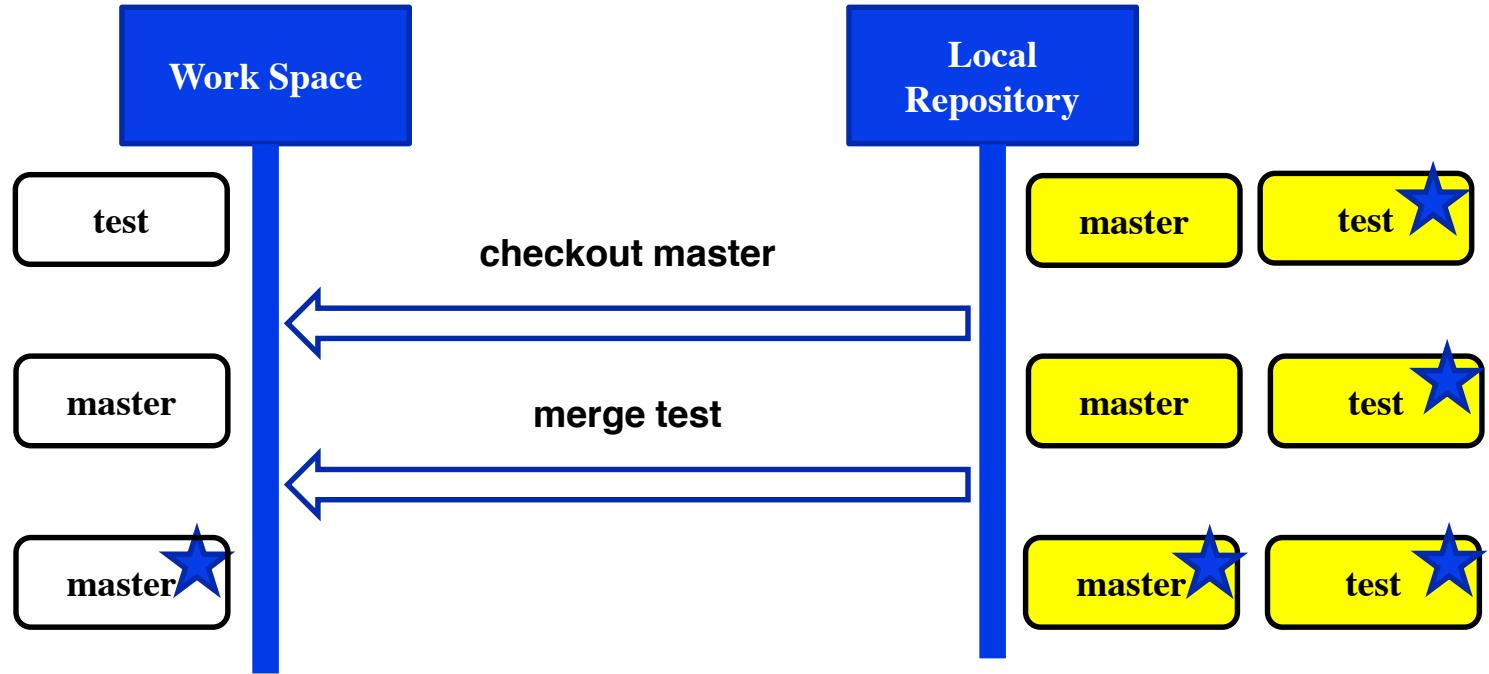
- 브랜치가 생성될 때 Work Space(그냥 폴더)의 내용은 master와 같다.
- 임의의 수정(별표)하고 commit 하면 Local repository의 test 브랜치에 그 내용이 적용된다. master하고는 관계없다.

branch merge

```
$ git branch (1)  
* master  
  test  
$ git merge test (2)  
$ git push (3)  
$ git branch -D test (4)
```

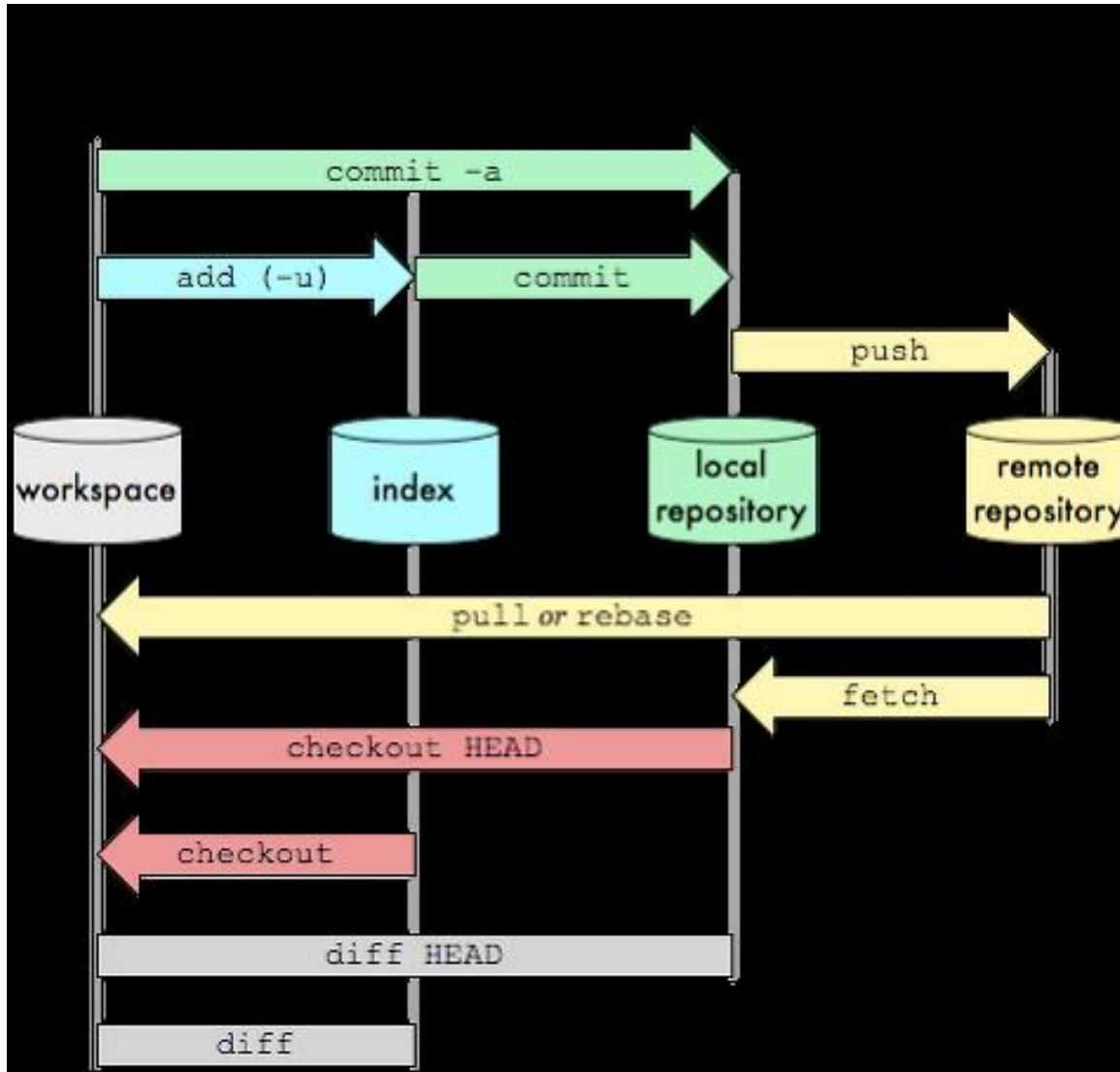
- (1) 브랜치 리스트를 조회한다. '*' 표 붙은 master가 현재의 브랜치이다.
- (2) 브랜치 test 를 master 로 merge 한다.
- (3) 서버로 push 한다.
- (4) 브랜치 test를 삭제한다.

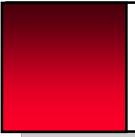
branch merge



- 새 브랜치 test에 적용한 사항을 master에 적용하려면 master 브랜치로 변환한 후 ‘merge test’를 한다.

Pull 과 Fetch





diff 하기



How To : 현재 작업된 것과 Local Repository과 의 차이를 비교하기

```
$ git commit -m "old change"  
$ echo "hi" >> test.txt  
$ git diff  
...  
+hi
```

(1)

(1) Local repository에 commit된 것과 현재 Work Space의 차이를 보여준다.

How To : 좀더 다양한 diff

\$ git diff	: Local Repository의 최근 commit된 상태와
\$ git diff --cached	: Index와 Local Repository간 차이
\$ git diff HEAD	: git diff와 동일
\$ git diff HEAD^	: Local Repository의 바로 전 revision과
\$ git diff HEAD^^	: Local Repository의 바로 전전 revision과
\$ git diff master test	: 브랜치 master와 test 간 차이
\$ git diff a1b2c3	: revision "a1b2c3"간 차이
\$ git diff HEAD^ a1b2c3	: Local Repository의 바로전 revision과 revision 'a1b2c3'간 차이
\$ git diff origin	: fetch해서 가져온 origin과 비교

- git diff에 사용되는 옵션은 브랜치 이름이거나 revision이다. 똘똘하게 알아서 찾아 준다.
- Remote Repository와 현재의 것을 비교하려면 우선 fetch 명령으로 Local Repository로 가져온 다음 diff origin으로 비교할 수 있다.



branch, HEAD, revision, master, origin

- Git는 revision에 순차적인 번호를 사용하지 않는다.
- 특정 revision에 숫자와 문자가 섞인 7 character를 사용한다.
- 그렇기 때문에 어떤 브랜치의 어떤 리비전이라고 하지 않고, diff 시에 그냥 리비전만 주면 된다.
- 만약 브랜치의 이름이 주어지면, 가장 최근의 리비전을 사용한다.
- HEAD는 현재 작업중인 브랜치의 가장 최근 리비전을 가리키는 예약어이다.
 - 만약 브랜치를 변경하지 않았으면 master가 현재의 브랜치가 되고 HEAD 대신 master로 하여도 결과는 같다.
- master는 기본 브랜치의 이름이다.
- origin은 Remote Repository에서 가져온 브랜치의 이름이다.

기타 팁들



How To : 파일 삭제, 파일 명 변경

```
$ git mv old.txt new.txt  
$ git rm useless.txt
```

(1)
(2)

(1) 파일명을 변경.

(2) 파일을 삭제

- Git는 파일명의 변경을 일일이 알려 주어야 한다. 지우고 새로 만든 것인지, 파일명만 변경되었는지 파악하지 못한다.
- 그래서 파일시스템의 rm, mv를 사용하면 곤란하다.



How To : 로그 보기

```
$ git log -5 --decorate --graph --oneline (1)  
$ git log -5 --author=hansori999 --oneline (2)  
$ git log --before={3.weeks.ago} --after={2015-04-18} (3)  
$ git log --grep="some message" (4)
```

- (1) 최근 5개의 리비전을 그래프로 보여준다.
- (2) 저자가 hansori999인 리비전을 5개 보여준다.
- (3) 주어진 기간내의 리비전만 보여준다.
- (4) commit 메시지에 “some message”가 포함된 것만 보여준다.

How To : commit 취소하기

```
$ git reset --soft HEAD^
```

(1)

- (1) Local Repository의 최신 commit 리비전을 바로전 리비전(HEAD[^])로 설정한다. Index와 Work Space는 건드리지 않는다.



How To : 특정 리비전 다운 받기

```
$ git checkout -b new a1b2c3
```

(1)

- (1) 새로운 브랜치 ‘new’를 생성하고 그 곳에 리비전 ‘a1b2c3’을 내려받는다.