

효과적인 프로젝트 협업을 위한
Git & Github

2023.08.23
4기 교육부장 노명은

이번 특강의 목표

**토이 프로젝트에서 Git, Github를
활용하여 프로젝트를 체계적으로 관리**

**어떻게 하면 1시간 안에 쉽고 간단하고
이해하기 쉽게 알 수 있을까?**



**개발자 유머로 배우는
효과적인 프로젝트 협업을 위한
깃 & 깃허브**



- 1. git 기초 명령어**
- 2. git branch 활용 방법**
- 3. github 활용 방법**

Git = 분산형 버전 관리 시스템



10일 →



초안 ver

git이 왜 중요할까요?

1. 버전 관리
2. 타인과 협업

In case of fire 🔥



1. git commit

2. git push

3. leave building

git이 왜 중요할까요?

- 1. 버전 관리**
- 2. 타인과 협업**

문서집계표(최종).HWP
문서집계표(최종수정).HWP
문서집계표(최종수정컨펌).HWP
문서집계표(컨펌V1).HWP
문서집계표(컨펌V2).HWP
문서집계표(컨펌V3).HWP
문서집계표(진짜최종).HWP
문서집계표(진짜진짜최종).HWP
문서집계표(진짜진짜진짜최종).HWP
문서집계표(회장님).HWP
문서집계표(회장님지시수정).HWP
문서집계표(회장님수정.V1).HWP.....

git이 왜 중요할까요?

1. 버전 관리
2. 타인과 협업



- 1. git 기초 명령어**
- 2. git branch 활용 방법**
- 3. github 활용 방법**

CLI 실습 중 오류가 났을 때 대처법

First step in learning Programming



Learn Basic
Syntax,
Data Types and
Variables.



Learn how to
Google.

DAY1 OF PROGRAMMING

Google

A screenshot of a Google search results page. The search bar contains the query "regex for email validation". Below the search bar are two cards: "Google Search" and "I'm Feeling Lucky".

10 YEARS OF PROGRAMMING

Google

A second screenshot of a Google search results page, identical to the first one, showing the same search query and results.

**git init
git remote
git clone
fork**



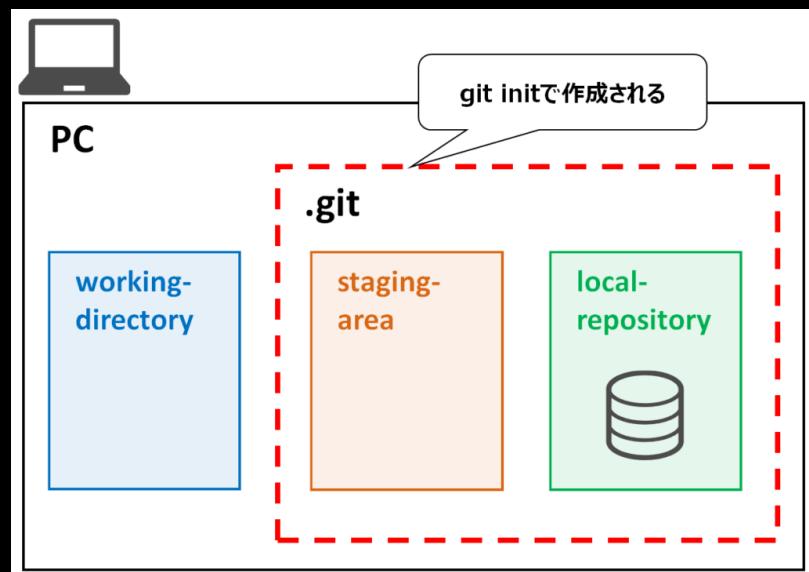
git init = git 저장소 생성

```
test — -zsh — 80x24  
n-ui-MacBookAir test % git init
```

1. ls -a 명령어 실행
2. git init 명령어 실행
3. 3. ls -a 명령어 실행

'.git' 폴더가 생겼다면 성공

cf. mkdir : 폴더 생성 명령어
ls : 현재 폴더에 있는 모든 파일 목록
(-a : 숨겨진 파일, 디렉토리(.이나 ..로 시작)까지 표시)

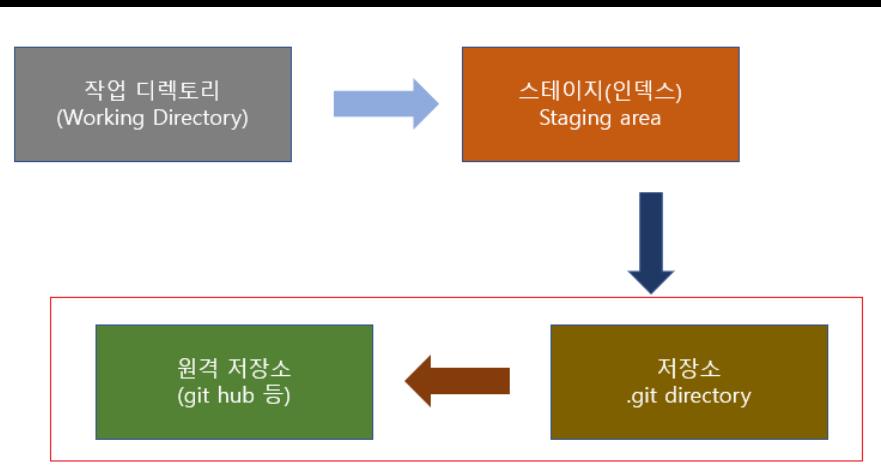


git remote = 원격 저장소와 연결

git remote add [원격저장소 이름] [url]

1. **git remote -v 명령어 실행**
2. **git remote add ... 명령어 실행**
3. **git remote -v 명령어 재실행**

3을 했을 때 설정한 저장소 이름과 url이름이 표시되면 성공

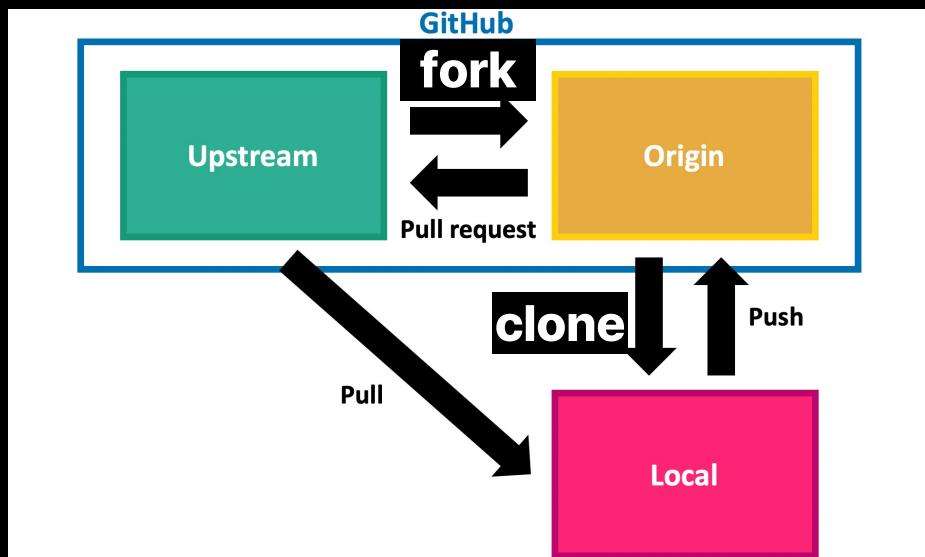


ex: % git remote add test https://github.com/khuda-4th/git-tutorial.git

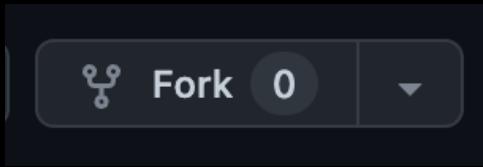
cf. 원격 연결 방식에는 HTTP와 SSH의 두 가지가 있습니다. 시간상 생략했지만, 관심 있으신 분은 찾아보시면 좋을 것 같습니다 14

git clone = 원격 to 로컬 복붙

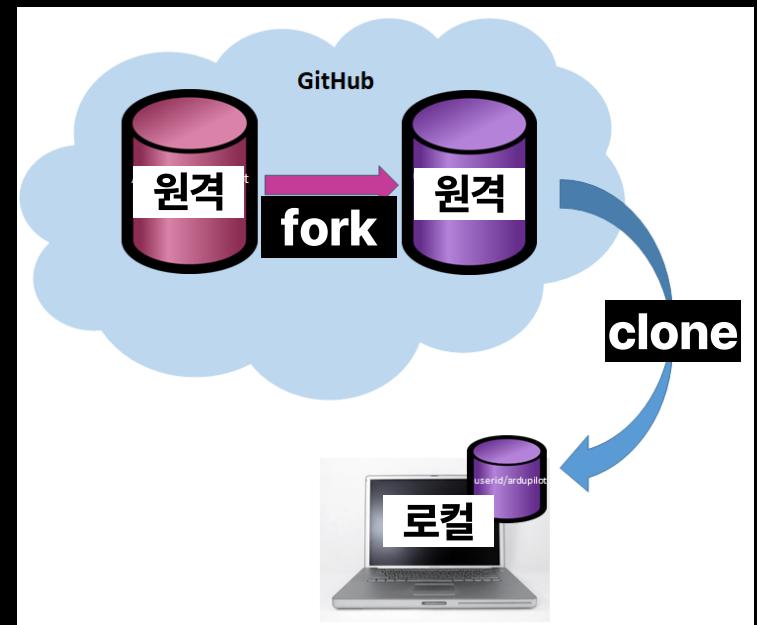
1. 디렉토리를 만든다.
2. 생성한 디렉토리로 이동하여 **git init** 명령으로 빈 git 저장소를 만든다.
3. 입력한 URL을 origin이라는(기본값) 이름의 리모트로 추가(**git remote add**)
4. **git fetch** 명령으로 리모트 저장소에서 데이터를 가져온다.



fork = 원격 to 원격 복붙



Fork된 저장소의 수정 사항을
원본에 반영하기 위해서 pull request 필요



**git add
git commit
git push
git pull
+ git fetch**



cf. 파일 관련 쉘 명령어

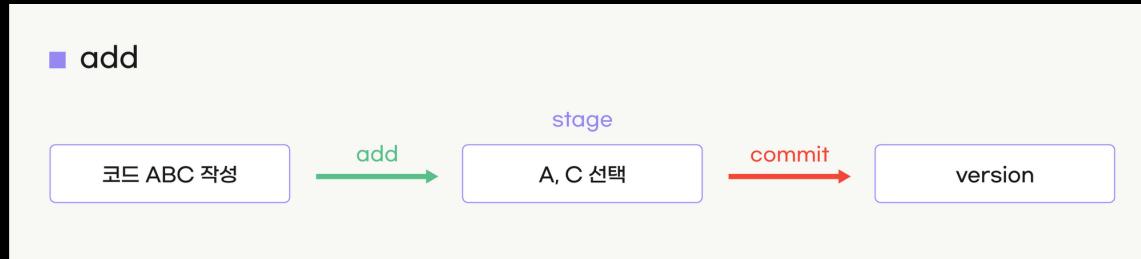
vim = 텍스트 편집기

vim [filename] : 파일 생성 및 편집

- **I : insert mode**
- **!wq : 저장 및 종료**
(esc 누르고 모드 변경 가능)

git add = stage area로 전송

git add [file or directory name]



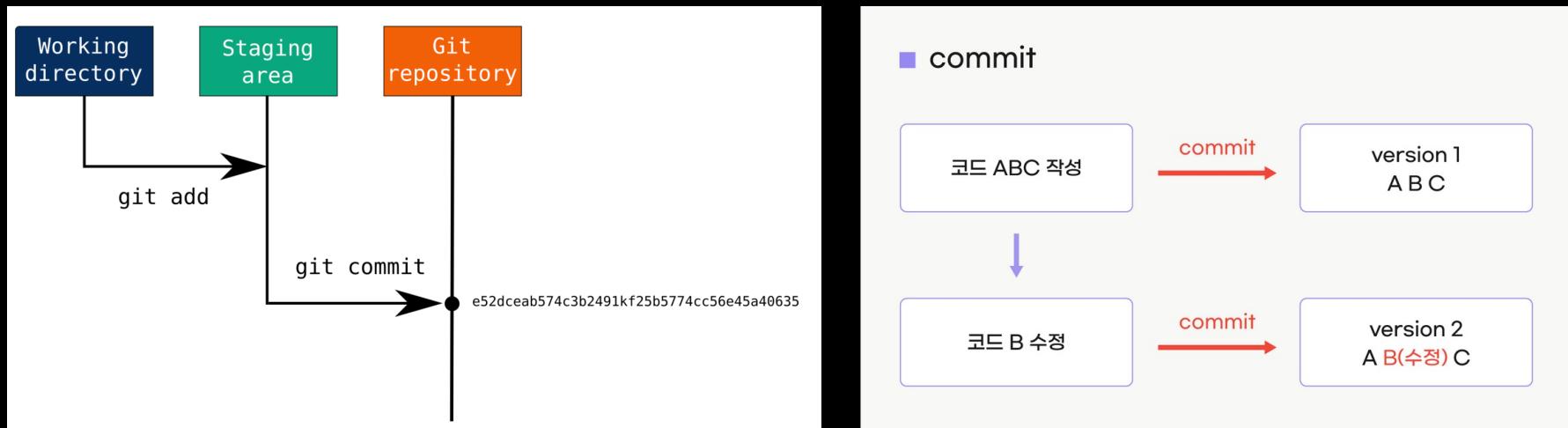
변경한 파일 목록 중 스테이지에 올리기 원하는 파일만 선택

1. **git add .** 명령어 실행
2. **git status** 명령어 실행하여 확인

cf. file이나 directory 이름에 . 이라고 작성하게 되면, 현재 디렉토리에 위치한 모든 파일들을 선택한다는 의미입니다.

git commit = 변경 사항 저장

git commit -m [commit message]



Stage area에 올라와 있는 파일들을 로컬에 기록. 수정사항에 대한 버전 생성

ex : git commit -m "added info (name, ID)

git push = 커밋 내용을 원격에 전송

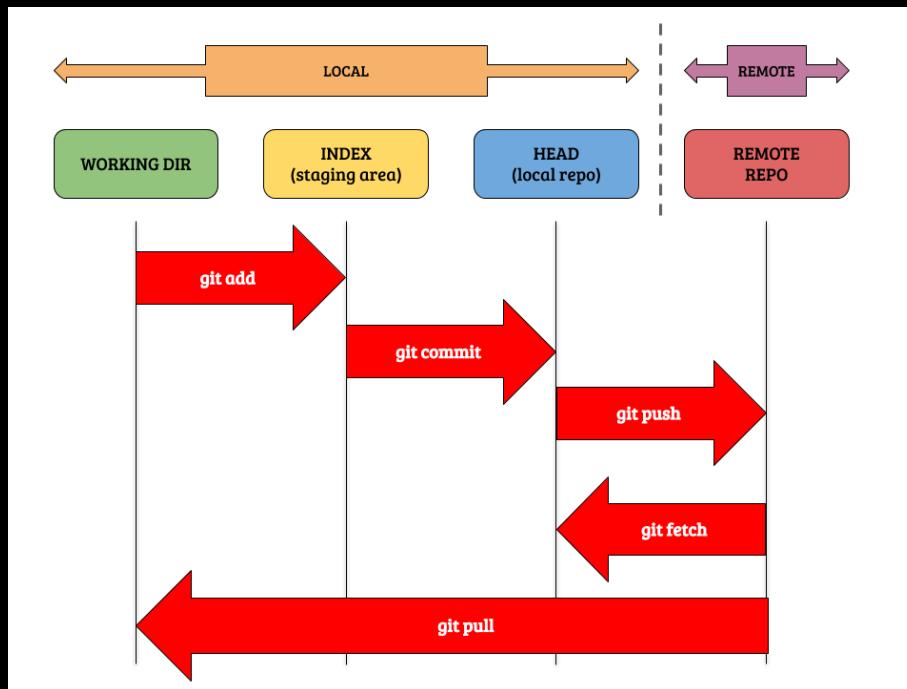
git push



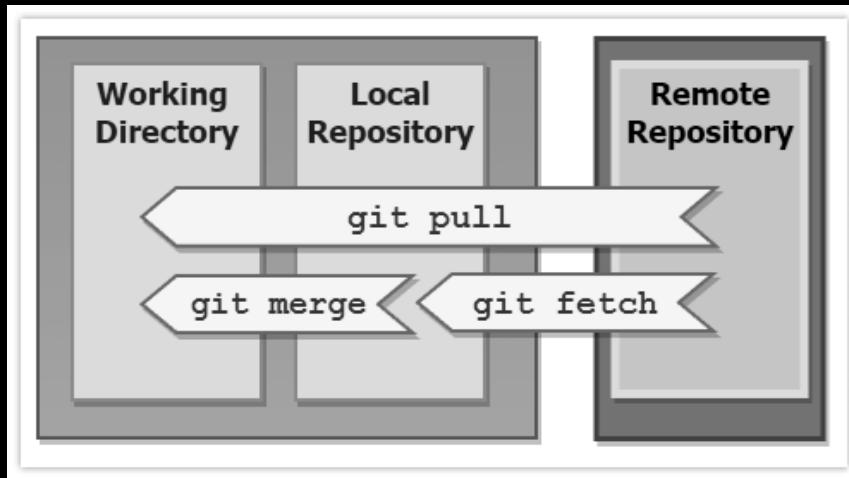
원격 저장소에 '실질적으로' 반영되도록 하는 명령어, 따라서 주의해야 한다

cf : --force 옵션을 주면 충돌 등을 무시하고 강제로 덮어쓴다

git pull = 원격 저장소에서 변경된 내용을 가져와 로컬 저장소에 병합



git fetch = merge 없이 변경사항만 임시로 가져온다



로컬 저장소에 가져오기 전에 확인(충돌, 변경사항 등)하고 싶을 때 사용
이후에 수동으로 병합 작업 필요

순서의 중요성



**git log
git diff
git status
.gitignore
.gitconfig**

git log = 커밋 히스토리 조회

```
[nomyeong-eun@nomyeong-eun-ui-MacBookAir test % git log
commit c0f1a75c48afdb70d0e7ff21d6c72a1cd65406ad (HEAD -> main, test/main)
Author: Co-DDING <90135669+NoMyeongEun@users.noreply.github.com>
Date:   Wed Aug 23 04:54:41 2023 +0900

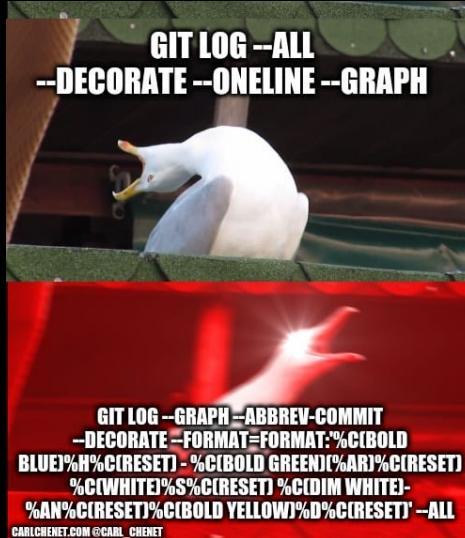
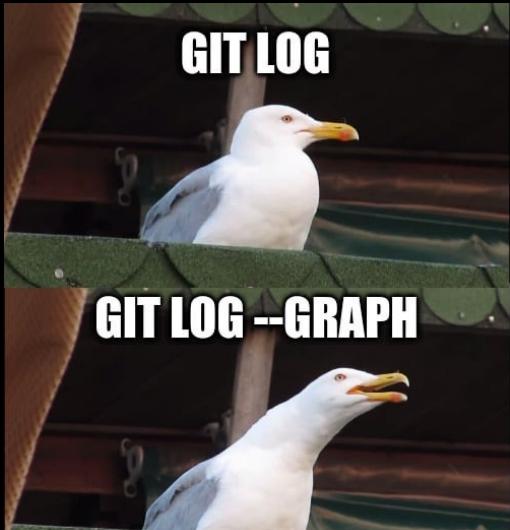
    push test

commit 0e27cb4557fe5c1cec959ffc151b7d39ea1b83c9 (test/master, master)
Author: Co-DDING <90135669+NoMyeongEun@users.noreply.github.com>
Date:   Wed Aug 23 04:37:25 2023 +0900

    added name, id
```

최근 순으로 commit ID, branch, author, 일자, commit message 표시

git log = 커밋 히스토리 조회



옵션	설명
<code>-p</code>	각 커밋에 적용된 패치를 보여준다.
<code>--word-diff</code>	diff 결과를 단어 단위로 보여준다.
<code>--stat</code>	각 커밋에서 수정된 파일의 통계정보를 보여준다.
<code>--shortstat</code>	<code>--stat</code> 명령의 결과 중에서 수정한 파일, 추가된 줄, 삭제된 줄만 보여준다.
<code>--name-only</code>	커밋 정보중에서 수정된 파일의 목록만 보여준다.
<code>--name-status</code>	수정된 파일의 목록을 보여줄 뿐만 아니라 파일을 추가한 것인지, 수정한 것인지, 삭제한 것인지도 보여준다.
<code>--abbrev-commit</code>	40자 짜리 SHA-1 체크섬을 전부 보여주는 것이 아니라 처음 몇 자만 보여준다.
<code>--relative-date</code>	정확한 시간을 보여주는 것이 아니라 2 주전처럼 상대적인 형식으로 보여준다.
<code>--graph</code>	브랜치와 머지 히스토리 정보까지 아스키 그래프로 보여준다.
<code>--pretty</code>	지정한 형식으로 보여준다. 이 옵션에는 oneline, short, full, fuller, format이 있다. format은 원하는 형식으로 출력하고자 할 때 사용한다.
<code>--oneline</code>	<code>--pretty=oneline --abbrev-commit</code> 옵션을 함께 사용한 것과 동일하다.

다양한 옵션을 통해 원하는 방식으로 표시 가능

git diff = 파일 변경 사항 확인



커밋 전

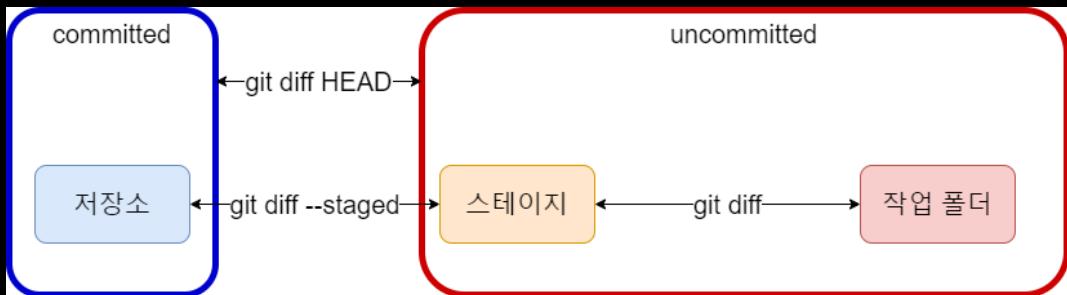
git diff = 파일 변경 사항 확인

```
nomyeong-eun@nomyeong-eun-ui-MacBookAir test % git diff
diff --git a/khuda_member_list b/khuda_member_list
index 5b9f6a8..6b90786 100644
--- a/khuda_member_list ← 원래 파일
+++ b/khuda_member_list ← 수정 파일
@@ -2,6 +2,6 @@
 name : No MyeongEun
 id : 2021105692
 -----
-name : Kim Khuda
+name : Park Khuda
 id : 2023100000
-----
```

원래 파일, 수정 파일의 2번째 줄부터 6개 줄을 가져왔다

수정 내용 : -는 사라진 내용, +는 추가된 내용

git diff = 파일 변경 사항 확인



1. **git diff** : 작업폴더 vs 스테이지
2. **git diff HEAD** : 작업폴더 vs HEAD
3. **git diff --staged** : HEAD vs 스테이지

옵션	뜻
--cached	staging area diff
--staged	staging area diff
--word-diff	
--color=words	
-w, --ignore-all-space	
--diff-filter	
--stat	수정된 사항의 통계를 보여줍니다.
-U, --unified=N	N 개의 라인을 포함한 diff
--output=<file>	stdout 이 아닌 file로 diff 결과 저장

다양한 옵션을 통해 원하는 조건에 맞게 비교 가능 (man git help로 확인 가능)

git status = 파일의 상태 확인

```
nomyeong-eun@nomyeong-eun-MacBookAir test % git status
현재 브랜치 main
브랜치가 'test/main'에 맞게 업데이트된 상태입니다.

커밋하도록 정하지 않은 변경 사항 :
  (무엇을 커밋할지 바꾸려면 "git add <파일>..."를 사용하십시오 )
  (use "git restore <file>..." to discard changes in working directory)
    수정 함 : khuda_member_list

커밋 할 변경 사항을 추가하지 않았습니다 ("git add" 및 / 또는 "git commit -a"를
사용하십시오 )
nomyeong-eun@nomyeong-eun-MacBookAir test %
```

1. **Untracked**(관리 대상 X) : 파일 생성 후 한번도 git add하지 않은 상태를

2. **Tracked**(관리 대상 O) : git이 관리하는 파일임

- **Unmodified** : 최근의 커밋과 비교했을 때 바뀐 내용이 없는 상태
- **Modified** : 최근 커밋과 비교했을 때 바뀐 내용이 있는 상태
- **Staged** : 파일이 수정되고 나서 스테이지 공간에 올라와 있는 상태(git add 후의 상태)

.gitignore = git에 추가되면 안 되는 파일 설정



mute



```
echo $'\n'<br>person' >> .gitignore
```

:funny.co



.gitignore = git에 추가되면 안 되는 파일 설정

```
[nomyeong-eun@nomyeong-eun-ui-MacBookAir test % git status --ignored
현재 브랜치 main
브랜치가 'test/main'에 맞게 업데이트된 상태입니다.

커밋하도록 정하지 않은 변경 사항 :
(무엇을 커밋할지 바꾸려면 "git add <파일>..."을 사용하십시오)
(use "git restore <file>..." to discard changes in working directory)
수정함 : khuda_member_list

추적하지 않는 파일 :
(커밋 할 사항에 포함하려면 "git add <파일>..."을 사용하십시오)
.gitignore

무시한 파일 :
(커밋 할 사항에 포함하려면 "git add -f <파일>..."을 사용하십시오)
ignore_test
```

The screenshot shows a terminal window titled 'test – vim .gitignore – 45x39'. It contains the following text:

```
ignore_test
```

Below the terminal window, the text 'gitignore 파일 내용' is displayed. To the right, another terminal window shows the command 'ls -a' output:-eun-ui-MacBookAir test % ls -a
.git ignore_test
.gitignore khuda_member_list

Below this, the text '현재 디렉토리 전체 파일, 폴더' is shown. Further down, the command 'git ls-files' output is shown:okAir test % git ls-files
khuda_member_list

Finally, the text 'Git에서 추적하는 파일' is displayed.

의도적으로 git에서 특정 파일이나 디렉토리를 추적하지 않도록 설정

.gitconfig = config 설정

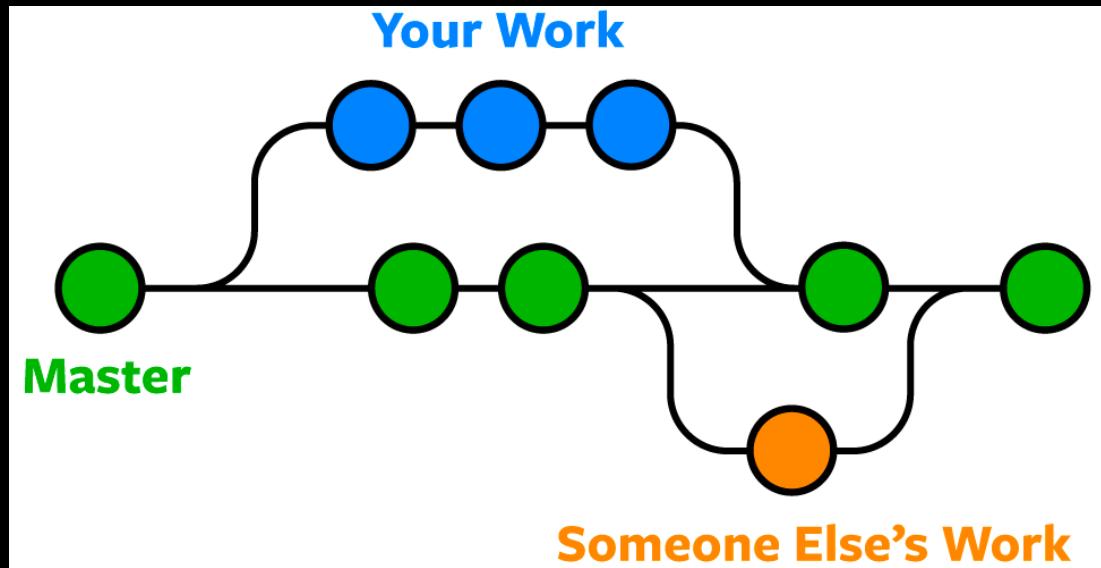
```
nomyeong-eun@nomyeong-eun-ui-MacBookAir test % git config --list
credential.helper=osxkeychain
user.name=Co-DDING
user.email=90135669+NoMyeongEun@users.noreply.github.com
filter.lfs.required=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true
remote.test.url=https://github.com/khuda-4th/git-tutorial.git
remote.test.fetch=+refs/heads/*:refs/remotes/test/*
branch.master.remote=test
branch.master.merge=refs/heads/master
branch.main.remote=test
branch.main.merge=refs/heads/main
```

사용자 이름, 이메일, alias(별칭), 인터페이스 옵션 등 설정

1. git 기초 명령어
2. git branch 활용 방법
3. github 활용 방법

**git branch
git checkout
git stash**

git branch = 협업 시, 충돌 방지를 위해 가지를 나눈다



git branch = 협업 시, 충돌 방지를 위해 가지를 나눈다

\$ git branch // 현재 존재하는 branch들의 리스트들을 확인

\$ git branch [branch name] // 새로운 branch를 생성

\$ git checkout [branch name] // 다른 branch로 이동

\$ git branch -D [branch name] // branch를 삭제

```
nomyeong-eun@nomyeong-eun-ui-Mac
BookAir test % git branch
* main
nomyeong-eun@nomyeong-eun-ui-Mac
BookAir test % git branch test1
nomyeong-eun@nomyeong-eun-ui-Mac
BookAir test % git branch
* main
    test1
```

git stash = 임시 저장



다른 브랜치로 이동하기 위해서는 반드시 commit이 필요
commit 할 완성도가 아니지만, 다른 브랜치로 이동해야 할 때

git stash = 임시 저장

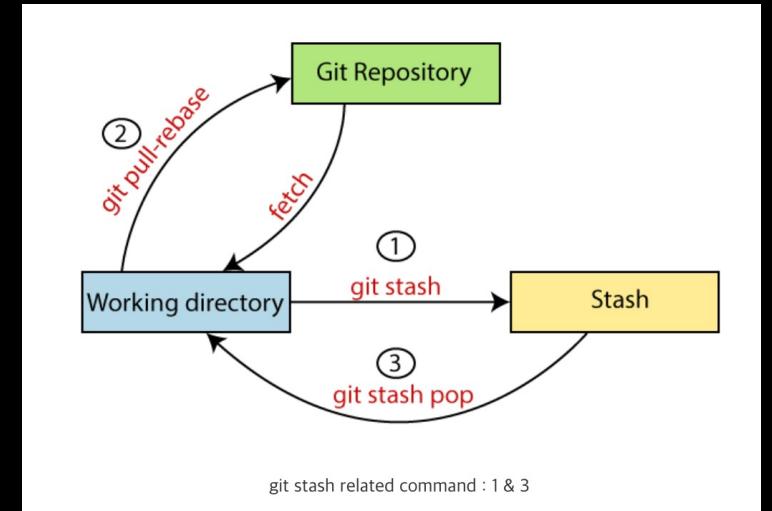
\$ git stash // 변경 내용을 임시 저장

\$ git stash list // stash 했던 내용 보기

\$ git stash apply // 가장 최근 stash 가져오기

\$ git stash apply [stash name] // 특정 stash 가져오기

\$ git stash pop // 임시 저장공간에 저장된 파일들을 현재 브랜치에 가져온다.



\$ git stash drop // 가장 최근 stash 지우기

\$ git stash drop [stash name] // 특정 stash 지우기

\$ git stash clear // 모든 stash 삭제

git stash = 임시 저장



충돌(conflict)이 일어날 경우를 대비하면,
임시 저장공간을 그대로 둔 채로 변경사항
을 가져오는 **apply**가 좋다.

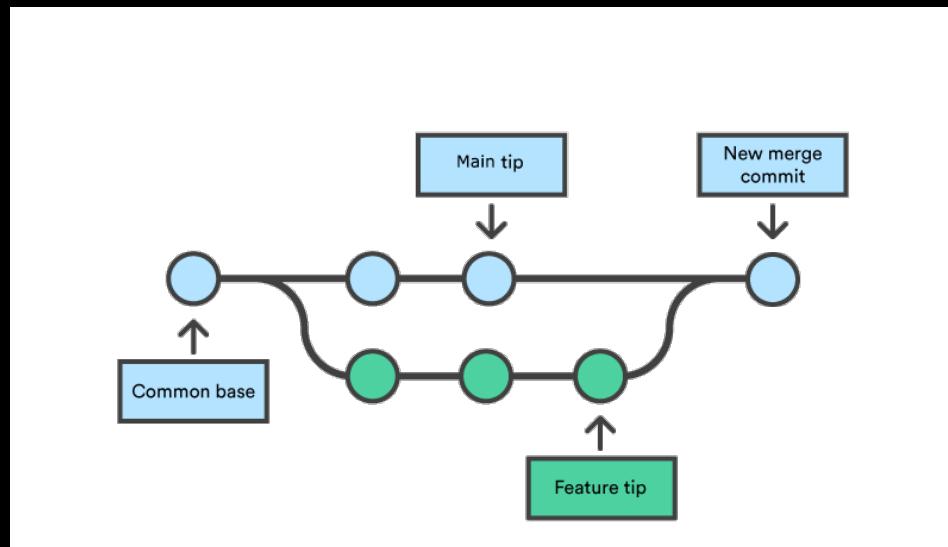
**git merge
git rebase
git cherry-pick
merge conflict**

GIT MERGE



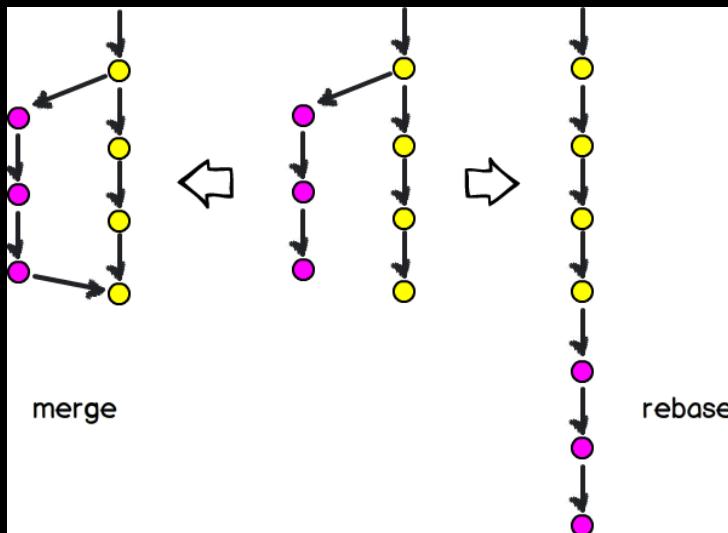
git merge = 서로 다른 브랜치, 작업내용을 합칠 때

git merge [branch name]



“현재” 브랜치에 [branch name]의 변경사항을 합친다

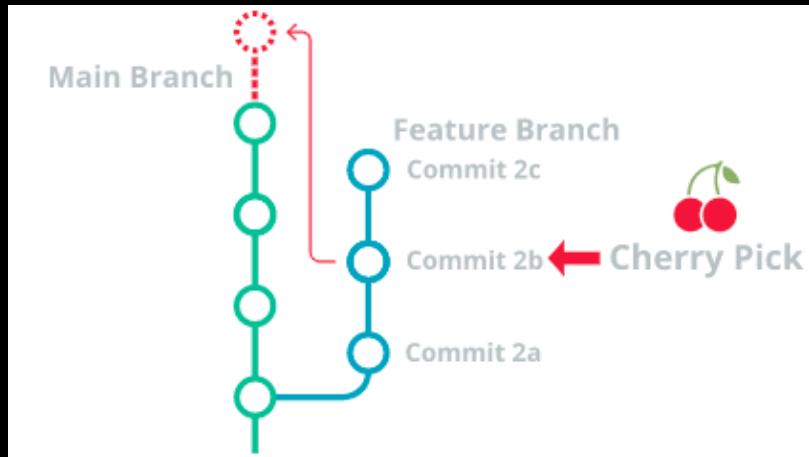
git rebase = branch의 base를 변경



`merge`와 결과적으로는 같지만, 더 깔끔한 **commit history**를 만들 수 있다.

git cherry-pick = 내가 필요한 커밋만 선택

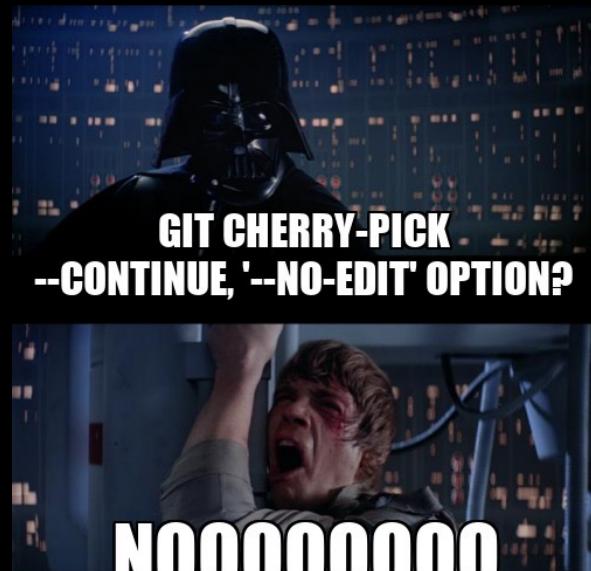
git cherry-pick [commit hash]



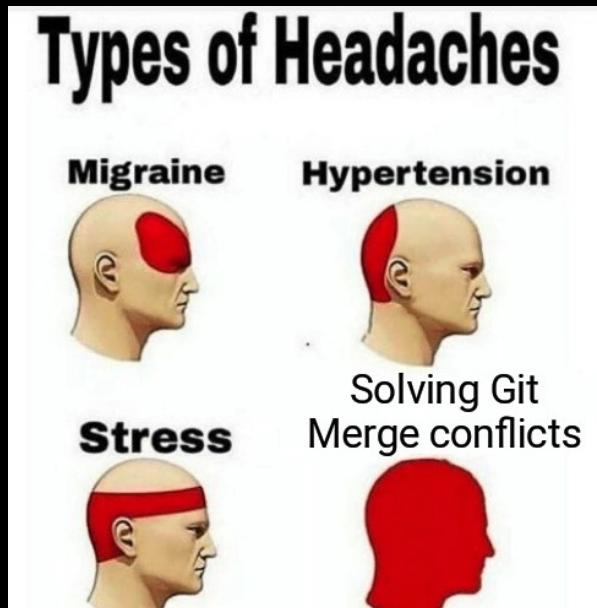
다른 branch에 있는 commit을 옮겨오는 것이 아니라,
현재 작업 브랜치에 해당 변경 내역을 새로운 commit으로
쌓는다.

merge나 rebase는 다른 브랜치의 전체 commit를 가져오는데, cherry-pick은 다른 브랜치의 원하는
commit만 가져올 수 있다.

git cherry-pick = 내가 필요한 커밋만 선택



merge conflict



merge conflict

3-way merge

원래 브랜치	내가 작업한 브랜치	동료가 작업한 브랜치	머지 예상 결과
A	A	A	A
B	B2	B	B2
C	C	C2	C2
D	D2	D3	Conflict

수동으로 수정하거나, GUI Tool을 활용하여 수정

merge conflict

```
----KHUDA 4th Member List----  
name : No MyeongEun  
id : 2021105692  
-----  
name : Park Khuda  
id : 2023100000  
-----  
~
```

```
----KHUDA 4th Member List----  
name : No MyeongEun  
id : 2021105692  
-----  
name : Song Khuda  
id : 2023100000  
-----
```

```
자동 병합 : khuda_member_list  
충돌 (내용) : khuda_member_list에 병합 충돌  
자동 병합이 실패했습니다. 충돌을 바로 잡고 결과물을 커밋하십시오.  
C:\Users\user\Documents\GitHub\khuda\src>
```

```
병합하지 않은 경로:  
(해결했다고 표시하려면 "git add <파일>..."를 사용하십시오)  
양쪽에서 수정: khuda_member_list
```

1. git 기초 명령어
2. git branch 활용 방법
3. github 활용 방법

GUI : GitKraken



The screenshot shows the GitKraken application interface. At the top, there's a navigation bar with tabs for 'repository', 'branch', 'electron', 'Boards', 'Timelines', and a '+' button. Below the navigation is a toolbar with 'Undo', 'Redo', 'Fetch', 'Push', 'Branch', 'Stash', and 'Pop' buttons.

The main area displays a timeline of commits for the 'electron' repository, specifically the 'master' branch. The commits are color-coded by author: Samuel Maddock (SM), GitHub (G), and others. Each commit has a detailed description and a list of files modified. For example, the first commit by SM is titled 'feat(extensions): expose ExtensionRegistryObserver events in Session (#25385)' and involves modifying 'docs/api/session.md', 'shell/browser/api/electron_api_session.cc', 'shell/browser/api/electron_api_session.h', and 'spec-main/extensions-spec.ts'.

On the left side, there's a sidebar with sections for 'ISSUES', 'TAGS', 'SUBMODULES', and 'GITHUB ACTIONS'. The 'ISSUES' section allows selecting an issue tracker for the repository, with options like GitHub, GitHub Enterprise, GitKraken Boards, GitLab, Jira Cloud, Jira Server, Trello, and None. The 'TAGS' section shows 1049 tags, and 'SUBMODULES' shows 2 submodules.

At the bottom right, there's a footer with icons for file operations, a search bar, a '100%' button, a 'Feedback' button, and a 'PRO 7.3.2' license information.

GUI : SourceTree



sourcetree-website (Git)

Commit Pull Push Branch Merge Shelve

All Branches Show Remote Branches Ancestor Order Jump to:

Graph	Commit	Author	Description	Date	
b7358c7	Rahul Chhab...	master	origin/master	origin/HEAD Removing ol...	Mar 3, 2016, 11:...
bdb8bef	Rahul Chhab...	Merged in update-google-verification (pull request #14)			Feb 18, 2016, 1:3...
dfe975d	Tyler Tadej...	origin/update-google-verification	Update google verificati...		Feb 11, 2016, 2:2...
3bc3290	Tyler Tadej...		Replace outdated Atlassia...		Feb 11, 2016, 2:1...
dba47f9	Tyler Tadej...	Add gitignore			Feb 11, 2016, 1:3...
ff67b45	Mike Minns...	Updated Mac min-spec to 10.10			Feb 15, 2016, 11:...
72d32a8	Michael Min...	Merged in hero_images (pull request #13)			Feb 15, 2016, 10:...
246c4ff	Joel Unger...	origin/hero_images hero_images	Used TinyPNG to c...		Feb 11, 2016, 3:3...
9d9438c	Joel Unger...	Replacing hero images with new version of SourceTree			Feb 9, 2016, 2:59...
ce75b63	Michael Min...	Merged in bug/date-https (pull request #12)			Feb 15, 2016, 10:...
85367bb	Patrick Tho...	origin/bug/date-https	fixed date and https errors		Jan 7, 2016, 12:2...
4f9b557	Joel Unger...	New Favicon			Feb 8, 2016, 3:55...
384e6d5	Rahul Chhab...	origin/search-console-access	search console google ver...		Feb 3, 2016, 2:09...
6fa47a9	Mike Minns...	updated to move supported version to OSX 10.9+			Dec 15, 2015, 2:0...
8dd87bb	Mike Minns...	remove extra , when a line is skipped due to empty server			Nov 23, 2015, 2:2...
faa195e	Mike Minns...	Skip records with empty server/user id as gas rejects them			Nov 23, 2015, 2:1...
0cdfe96	Mike Minns...	corrected paths after merge			Nov 23, 2015, 2:0...
051ab1b	Mike Minns...	corrected column counting			Nov 23, 2015, 1:5...
a723bc2	Mike Minns...	Merge branch 'au2gex'			Nov 23, 2015, 1:5...
65fd580	Mike Minns...	deal with invalid instanceids			Nov 23, 2015, 1:5...
500a892	Michael Min...	Merged in au2gex (pull request #11)			Nov 23, 2015, 1:0...

WORKSPACE
File status
History
Search
BRANCHES
BOOKMARKS
TAGS
REMOTES
SHELVED
SUBREPOSITORIES

GUI : Github Desktop



A screenshot of the GitHub Desktop application interface. The top bar shows "Current Repository" as "desktop-primary", "Current Branch" as "whitespace-no-more", and a "Publish branch" button. The main area shows a "Changes" tab with 1 changed file, "app/src/ui/history/commit-list.tsx". The code diff highlights changes in lines 283-288. The bottom panel shows a commit dialog with a user profile picture, the file path "Update commit-list.tsx", a "Description" field, and a "Commit to whitespace-no-more" button. The status bar at the bottom indicates "Committed 11 minutes ago" and "Whitespace 1".

```
@@ -280,9 +280,11 @@ export class CommitList extends React.Component<ICommitListProps, {}> {
 280   }
 281   281
 282   282   public render() {
 283     -   return (
 284     -     <div className="panel blankslate">{this.props.emptyListMessage}</div>
 285     -   )
 283+   if (this.props.commitSHAs.length === 0) {
 284+     return (
 285+       <div className="panel blankslate">{this.props.emptyListMessage}</div>
 286+     )
 287+   }
 286
 287   289
 288   290   return (
 289     <div id="commit-list">
```

KHUDA 4기 깃허브

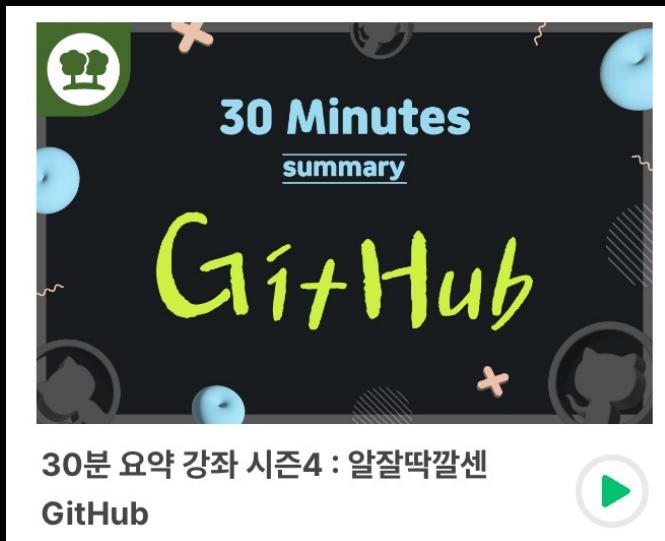
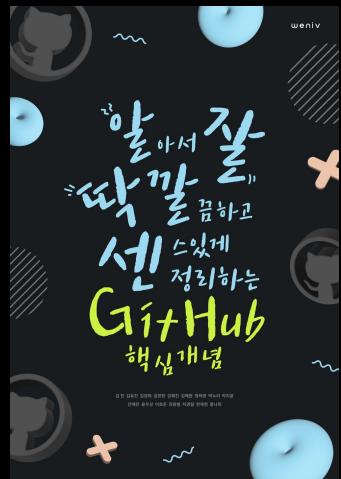
<https://github.com/khuda-4th>

The screenshot shows the GitHub repository page for 'KHUDA-4th'. At the top left is the team logo, which is a blue and red circular icon with the word 'KHUDA' below it. To the right of the logo is the repository name 'KHUDA-4th' and a brief description: 'KyungHee University Data Analysis 경희대학교 데이터분석 동아리 4기'. Below this is a 'Follow' button. The main content area features a large image with the text 'KHUDA 4TH' in bold black letters. Below the image, the repository name 'Kyung Hee University Data Analysis' is displayed. A note states 'KHUDA는 경희대학교의 데이터 분석 동아리입니다.' and 'KHUDA는 팀원과 함께 성장하며, 본인의 한계를 뛰어넘는 문화를 추구합니다.'. Below this are links for 'Instagram', 'Github', and 'Notion'. A navigation link '[> KHUDA-3rd history]' is also present. On the right side of the page, there are sections for 'View as: Public', 'Discussions', and 'Repositories'. The 'git-tutorial' repository was updated 1 hour ago, and the '.github' repository was updated 5 hours ago. There are buttons for 'Create new repository' and 'Import'.

프로젝트 진행 시 깃허브 활용

- 1. 교육팀 측에서 레포 생성**
- 2. 팀원들은 생성된 레포에서 자유롭게 프로젝트 진행**
- 3. 프로젝트 완성 이후 README.MD 작성 필수
(이후에 깃허브 관련 매뉴얼 업로드 예정)**

무료 배포 교재 공유



<https://paullabworkspace.notion.site/GitHub-435ec8074bcf4353afb947f601a030df#20c3418ef6fa49bf86353c50e02e0307>

인프런 강의(무료)

Q & A