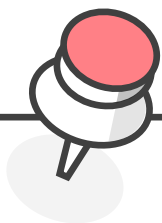




Git & GitHub

깃과 깃헙을 이용한 협업하기

강경미, 김성박 작성



Git & GitHub을 익혀야 하는 이유?

- 일단 Git이 무엇인지 알아보자.
 - 분산 버전 관리 시스템

깃 (소프트웨어)

위키백과, 우리 모두의 백과사전.

깃(Git /git/^[5])은 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템이다. 소프트웨어 개발에서 소스 코드 관리에 주로 사용되지만^[6] 어떠한 집합의 파일의 변경사항을 지속적으로 추적하기 위해 사용될 수 있다. 기하학적 불변 이론을 바탕으로 설계됐고, 분산 버전 관리 시스템으로서 빠른 수행 속도에 중점을 두고 있는 것^[7]이 특징이며 데이터 무결성,^[8] 분산, 비선형 워크플로를 지원한다.^[9]

깃은 2005년에 리눅스 커널 개발을 위해 초기 개발에 기여한 다른 커널 개발자들과 함께 2005년에 리눅스 토르발스가 처음 개발한 것이다.^[10] 2005년부터 지금까지 주니오 하마노(Junio Hamano)가 소프트웨어의 유지보수를 맡고 있다.

다른 대부분의 분산 버전 관리 시스템처럼, 또 대부분의 클라이언트-서버 시스템과 달리, 모든 노드의 모든 깃 디렉터리는 네트워크 접속이나 중앙 서버와는 독립적으로 동작하는 완전한 이력 및 완전한 버전 추적 기능을 갖춘 성숙한 저장소이다.^[11]

깃은 GNU 일반 공중 사용 허가서 v2 하에 배포되는 자유 소프트웨어이다.

목차 [숨기기]

깃

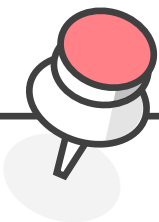


```
$ git init
Initialized empty Git repository in /tmp/tmp.IMBYSY7R8Y/.git/
$ cat > README << 'EOF'
> Git is a distributed revision control system.
> EOF
$ git add README
$ git commit
[master (root-commit) e4dccc9] You can edit locally and push
to any remote.
 1 file changed, 1 insertion(+)
 create mode 100644 README
$ git remote add origin git@github.com:cdown/thats.git
$ git push -u origin master
```

저장소 생성, 파일 추가, 원격 동기화를 표시하는 명령
줄 세션

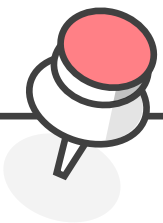
원저자

리눅스 토르발스^[1]



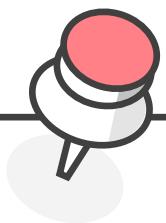
Git & GitHub을 익혀야 하는 이유?

- 버전 관리란 무엇인가?
 - 콘솔 게임(ex: 플레이스테이션)에서 게임을 하다 보면 중간 중간 Save를 한다. 중간 중간 Save를 하는 이유가 무엇인가?
 - 프로그램을 작성하다가, 과거의 어느 시점으로 되돌아 가고 싶은 적이 있는가?
- 프로젝트란 무엇인가?
 - 2인 이상이 함께 어떤 주제를 정해진 기간안에 만들어야 하는 것.
 - 다수의 사람이 프로젝트를 함께 한다면, 소스 코드를 어떻게 관리하는 것이 좋을까?
 - 아직도 특정 요일마다 각자 개발한 내용을 합친 후 빌드하고 배포하는 경우가 있다고 하더라.



Git & GitHub을 익혀야 하는 이유?

- Git
 - 가장 인기 있는 버전 관리 도구.
- GitHub
 - 가장 인기 있는 클라우드 저장소.



Git 설치하기

- 터미널 (Mac) 이나 커맨드창 (Windows)에서 git 명령을 실행해보자.

```
cmd. 명령 프롬프트
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

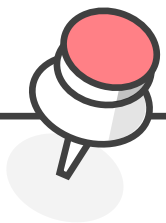
C:\Users\Wurstory>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
    clone      Clone a repository into a new directory
    init       Create an empty Git repository or reinitialize an existing one


work on the current change (see also: git help everyday)
    add        Add file contents to the index
    mv         Move or rename a file, a directory, or a symlink
    restore    Restore working tree files
    rm         Remove files from the working tree and from the index
    sparse-checkout  Initialize and modify the sparse-checkout


examine the history and state (see also: git help revisions)
    bisect     Use binary search to find the commit that introduced a bug
    diff       Show changes between commits, commit and working tree, etc
    grep       Print lines matching a pattern
    log        Show commit logs
    show       Show various types of objects
    status     Show the working tree status
```



Git 설치하기

- <https://git-scm.com/downloads> 에서 다운로드 후 설치할 수 있다.
- 윈도우 사용자는 반드시 Git Bash 옵션을 체크한 후 설치한다. 앞으로 윈도우 사용자는 Git Bash에서 Git 명령을 실행할 것이다.

The screenshot shows the 'Downloads' page of the Git website. The browser address bar displays 'git-scm.com/downloads'. The page features a sidebar with links for 'About', 'Documentation', 'Downloads' (highlighted), 'GUI Clients', 'Logos', and 'Community'. The main content area is titled 'Downloads' and includes a search bar. A red box highlights the download links for macOS, Windows, and Linux/Unix. To the right, a monitor graphic displays the 'Latest source Release 2.31.1' and a button to 'Download 2.31.1 for Windows'. Below this, there are sections for 'Older releases', 'GUI Clients', and 'Logos'. At the bottom, a section titled 'Git via Git' provides instructions on how to install Git using the command line.

git --local-branching-on-the-cheap

Search entire site...

Downloads

macOS Windows Linux/Unix

Latest source Release
2.31.1
Release Notes (2021-03-26)
Download 2.31.1 for Windows

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

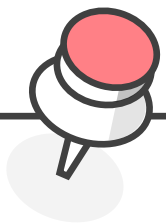
GUI Clients
Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.
[View GUI Clients →](#)

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

Git via Git
If you already have Git installed, you can get the latest development version via Git itself:
`git clone https://github.com/git/git`
You can also always browse the current contents of the git repository using the [web interface](#).

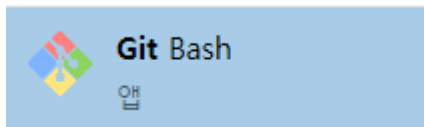
About this site
Patches, suggestions, and comments are welcome.

Git is a member of Software Freedom Conservancy

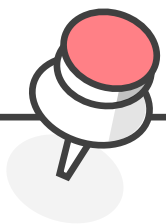


Git 설치하기

- 설치 후 Git Bash를 실행한다.
- Git Bash 안에서는 윈도우 안에서 리눅스(Linux) 명령을 실행할 수 있다.
- pwd : 현재 디렉토리(폴더) 경로(path)를 출력
- cd : 디렉토리 이동 명령

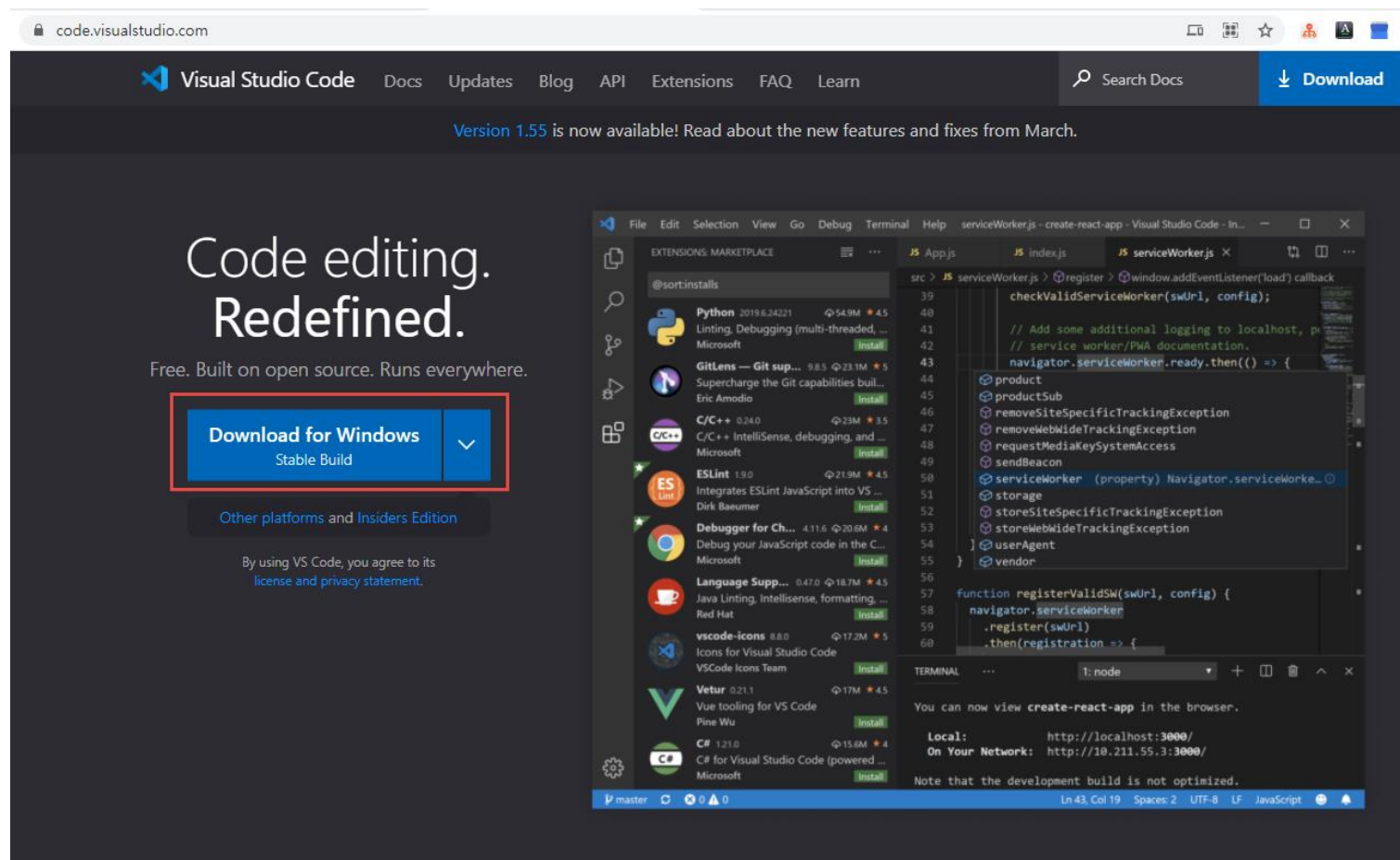


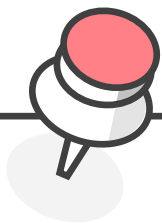
```
urstory@DESKTOP-4SM171R MINGW64 ~  
$ pwd  
/c/Users/urstory  
  
urstory@DESKTOP-4SM171R MINGW64 ~  
$ cd /d/devel  
  
urstory@DESKTOP-4SM171R MINGW64 /d/devel  
$ pwd  
/d/devel  
  
urstory@DESKTOP-4SM171R MINGW64 /d/devel  
$ |
```



Visual Studio Code 설치

- <https://code.visualstudio.com/> 에서 다운로드 후 설치한다.





SourceTree 설치하기

- <https://www.sourcetreeapp.com/> 에서 다운로드 후 설치한다.

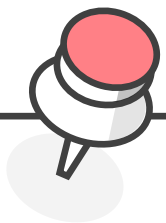
The screenshot shows the SourceTree website at <https://www.sourcetreeapp.com/>. The website features the SourceTree logo, a "Download free" button, and the text "Simplicity and power in a beautiful Git GUI". A red box highlights the "Download for Windows" button. Below this, it says "Also available for Mac OS X".

Overlaid on the website is a screenshot of the SourceTree application interface. The interface shows a sidebar with "WORKSPACE" (File status, History, Search), "BRANCHES", "BOOKMARKS", "TAGS", "REMOTES", "SHELVED", and "SUBREPOSITORIES". The main area displays a commit history table for the "sourcetree-website (Git)" repository.

Graph	Commit	Author	Description	Date
	b7358c7	Rahul Chhab...	tr master tr origin/master tr 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...	tr origin/update-google-verification Update google verificati...	Feb 11, 2016, 2:2...
	3bc3290	Tyler Tadej...	Replace outdated Atlassian logo in footer with base-64 en...	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...	tr origin/hero_images tr 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...	tr 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...	tr 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...
	0cdf96	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...

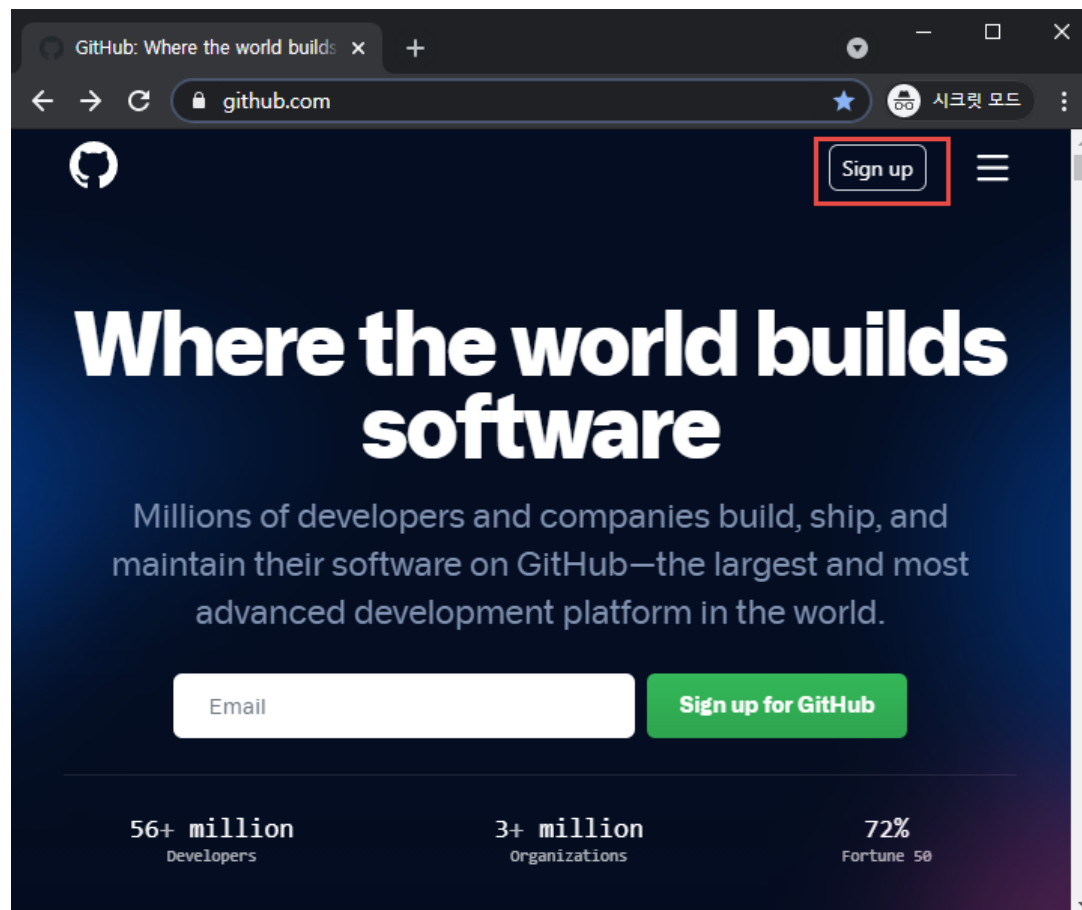
A free Git client for Windows and Mac

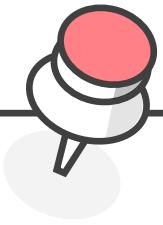
SourceTree simplifies how you interact with your Git repositories so you can focus on coding. Visualize and manage your repositories through SourceTree's simple Git GUI.



GitHub 가입하기

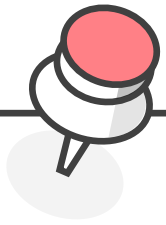
- `https://github.com/` 을 브라우저로 방문한 후 "Sign Up" 링크를 클릭하여 가입한다.





GitHub 외의 Git호스팅 사이트

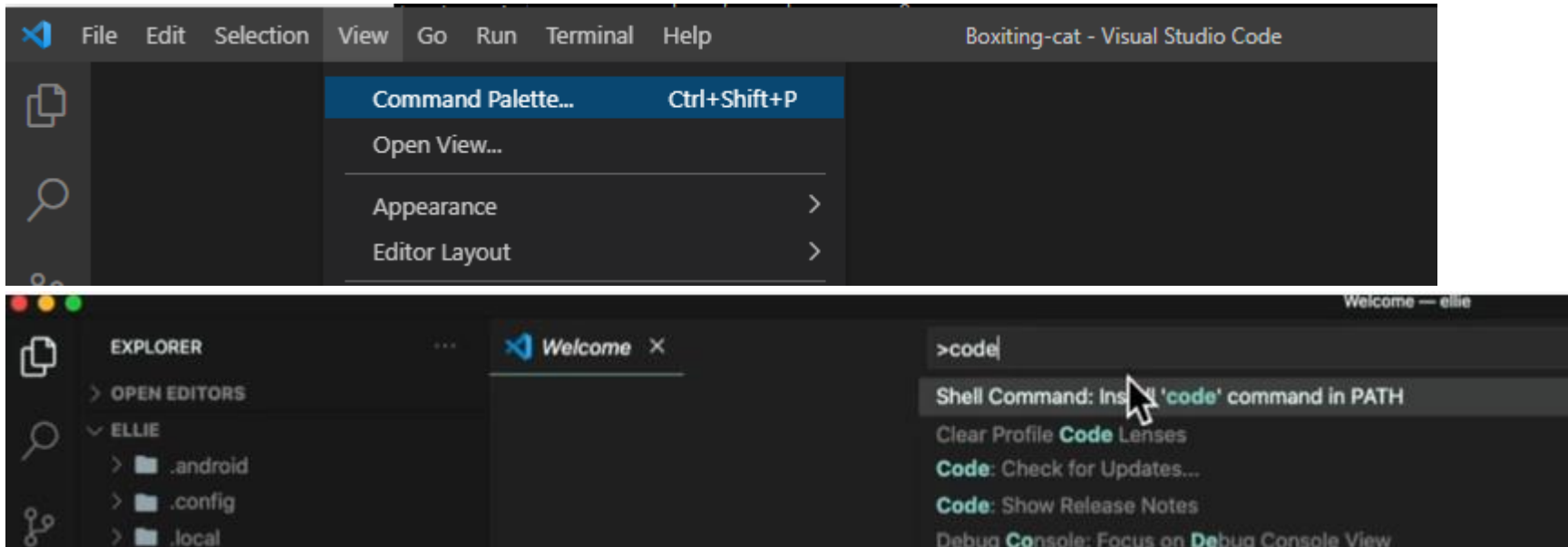
- GitLab
- BitBucket



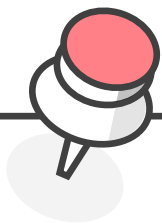
Git 워크플로우

git 설정하기

- git config --list
- Visual Studio Code를 어떤 경로에서든 code명령으로 실행할 수 있도록 설정한다.



- git config --global core.editor "code"
- git config --global core.editor "code --wait"
- git 환경설정을 에디터로 수정하겠다.
 - git config --global -e



Git 워크플로우

git사용자 이름과 이메일을 설정한다.

- `git config --global user.name "carami"`
- `git config --global user.email "carami@nate.com"`

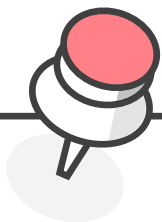
이름, 이메일을 확인한다.

- `git config user.name`
- `git config user.email`

- 윈도우 사용자 & 맥 사용자는 줄바꿈 표시가 다른 문제가 있다. 이를 해결하기 위해 설정한다.

- `git config --global core.autocrlf true`
- `git config --global core.autocrlf input`





Git 워크플로우

로컬 컴퓨터에 프로젝트를 위한 폴더를 생성한다. c:\devel\funnyblog_carami

```
MINGW64:/c/devel/funnyblog_carami
urstory@DESKTOP-4SM171R MINGW64 /c
$ mkdir /c/devel

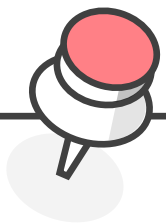
urstory@DESKTOP-4SM171R MINGW64 /c
$ cd /c/devel

urstory@DESKTOP-4SM171R MINGW64 /c/devel
$ mkdir funnyblog_carami

urstory@DESKTOP-4SM171R MINGW64 /c/devel
$ cd funnyblog_carami/

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami
$ pwd
/c/devel/funnyblog_carami

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami
$
```

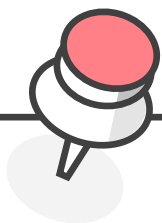


Git 워크플로우

로컬 컴퓨터에 프로젝트를 위한 폴더를 생성한다. `c:\devel\funnyblog_carami`

프로젝트 폴더(`c:\devel\funnyblog_carami`)





Git 워크플로우

git init : 현재 폴더를 Git저장소로 생성한다. .git폴더가 자동으로 생성된다.

ls : 현재 디렉토리의 파일 목록을 조회한다.

ls -la : 현재 디렉토리의 히든 파일까지 포함하여 파일 목록을 조회한다.

```
MINGW64:/c/devel/funnyblog_carami

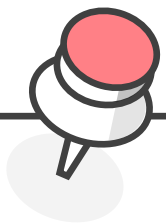
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami
$ ls

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami
$ ls -la
total 0
drwxr-xr-x 1 urstory 197121 0 Apr 18 01:17 ./
drwxr-xr-x 1 urstory 197121 0 Apr 18 01:17 ../

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami
$ git init
Initialized empty Git repository in C:/devel/funnyblog_carami/.git/

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ ls -la
total 4
drwxr-xr-x 1 urstory 197121 0 Apr 18 01:18 ./
drwxr-xr-x 1 urstory 197121 0 Apr 18 01:17 ../
drwxr-xr-x 1 urstory 197121 0 Apr 18 01:18 .git/

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ |
```

Git 워크플로우

git init 명령이 실행된 후.

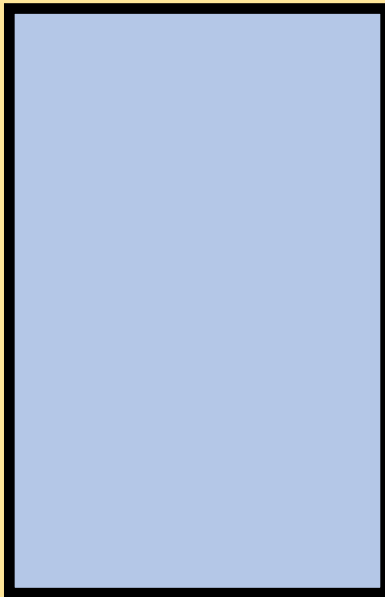
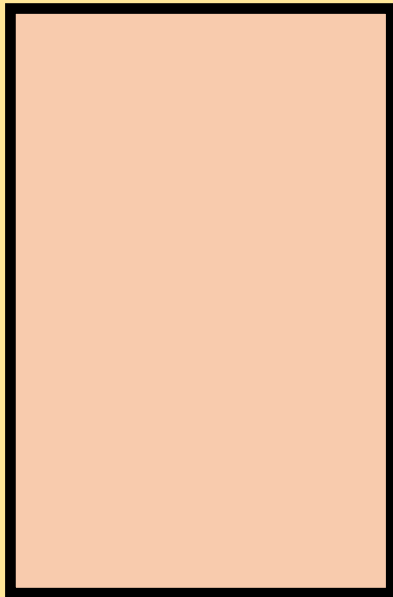
프로젝트 폴더(c:\devel\funnyblog_carami)

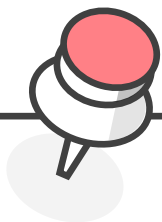
working directory

.git 폴더

staging area
(index, cache)

Local
Repository





Git 워크플로우

hello world가 저장된 3가지 파일을 생성한다.

echo 문자열 : 화면에 문자열을 출력한다.

> : 리다이렉션 기호. 출력을 입력으로 바꾼다.

echo 문자열 > a.txt : 문자열이 원래는 화면으로 출력해야 하는데, a.txt파일로 출력되게 한다.

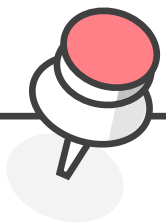
```
MINGW64:/c/devel/funnyblog_carami
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ echo hello world > a.txt

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ echo hello world > b.txt

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ echo hello world > c.txt

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ ls
a.txt  b.txt  c.txt

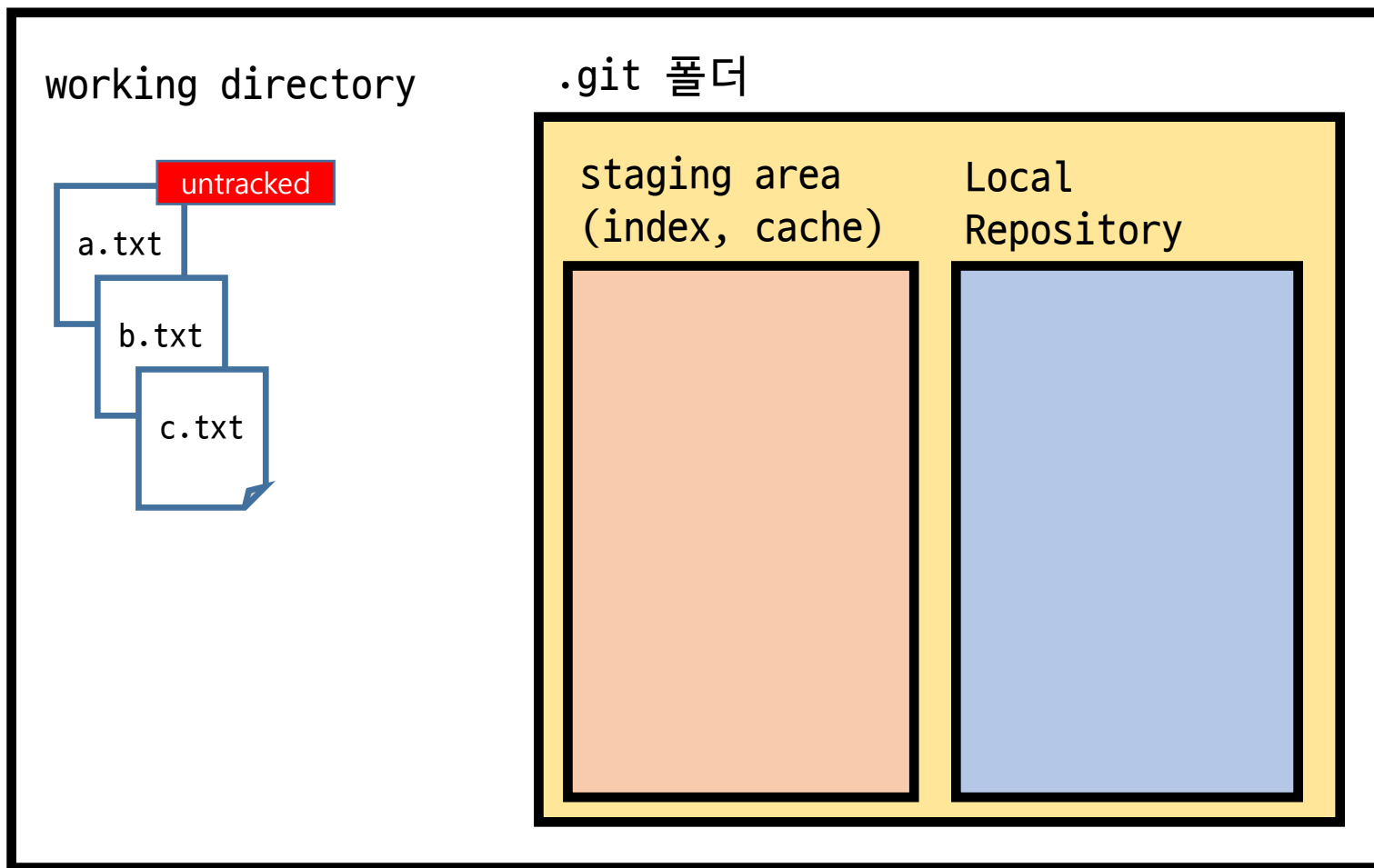
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ |
```

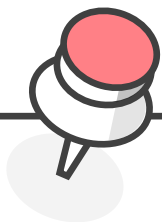


Git 워크플로우

git init 명령이 실행된 후.

프로젝트 폴더(c:\devel\funnyblog_carami)





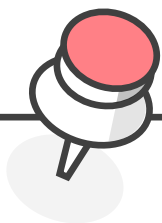
Git 워크플로우

Git이 추적하는 파일 상태

- 처음 파일을 작성하면 "untracked" 상태임
- git add 명령을 사용하면 "tracked" 상태면서, 스테이지됨.
- git commit을 사용하면 "tracked"상태면서, 스테이지에서는 사라짐.
- "tracked"인 파일을 수정하면 "수정함"인 상태를 가지게 됨.
- 수정한 파일을 다시 git add 명령을 사용하면 "tracked"상태면서, 스테이지됨

untracked

tracked



Git 워크플로우

git status 명령을 실행한다.

- On branch main : main 브랜치 사용중
- No commits yet : 아직 커밋 안됨
- Untracked files : 아직 추적되지 않는 파일들 (a.txt, b.txt, c.txt)

```
MINGW64:/c/devel/funnyblog_carami

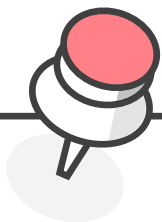
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    a.txt
    b.txt
    c.txt

nothing added to commit but untracked files present (use "git add" to track)

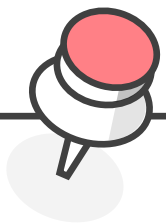
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$
```



Git 워크플로우

```
git add a.txt  
git status
```

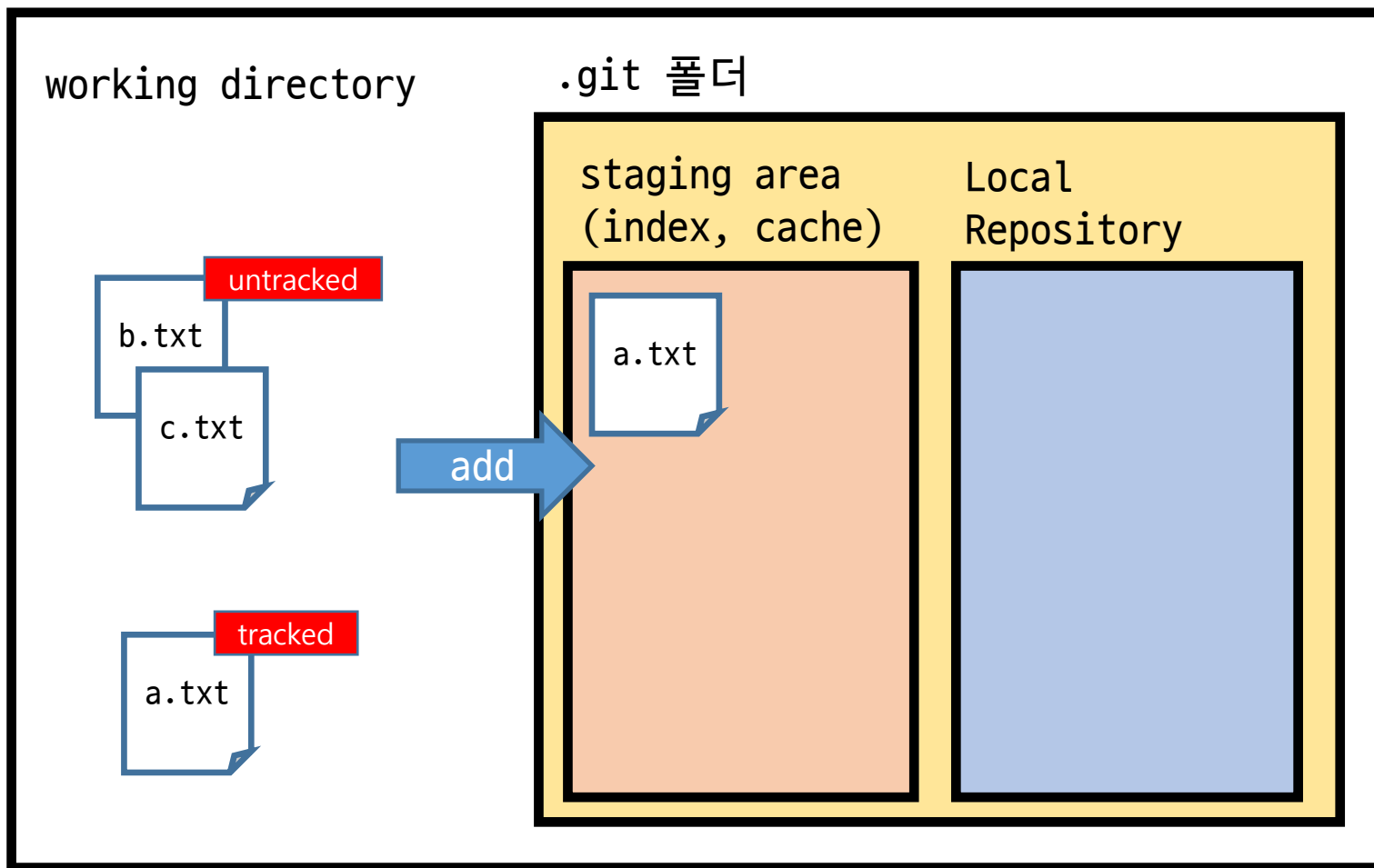
```
MINGW64:/c/devel/funnyblog_carami  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$ git add a.txt  
warning: LF will be replaced by CRLF in a.txt.  
The file will have its original line endings in your working directory  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$ git status  
On branch main  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   a.txt  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    b.txt  
    c.txt  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$
```

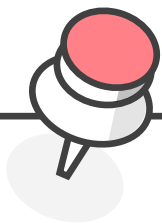


Git 워크플로우

```
git add a.txt
```

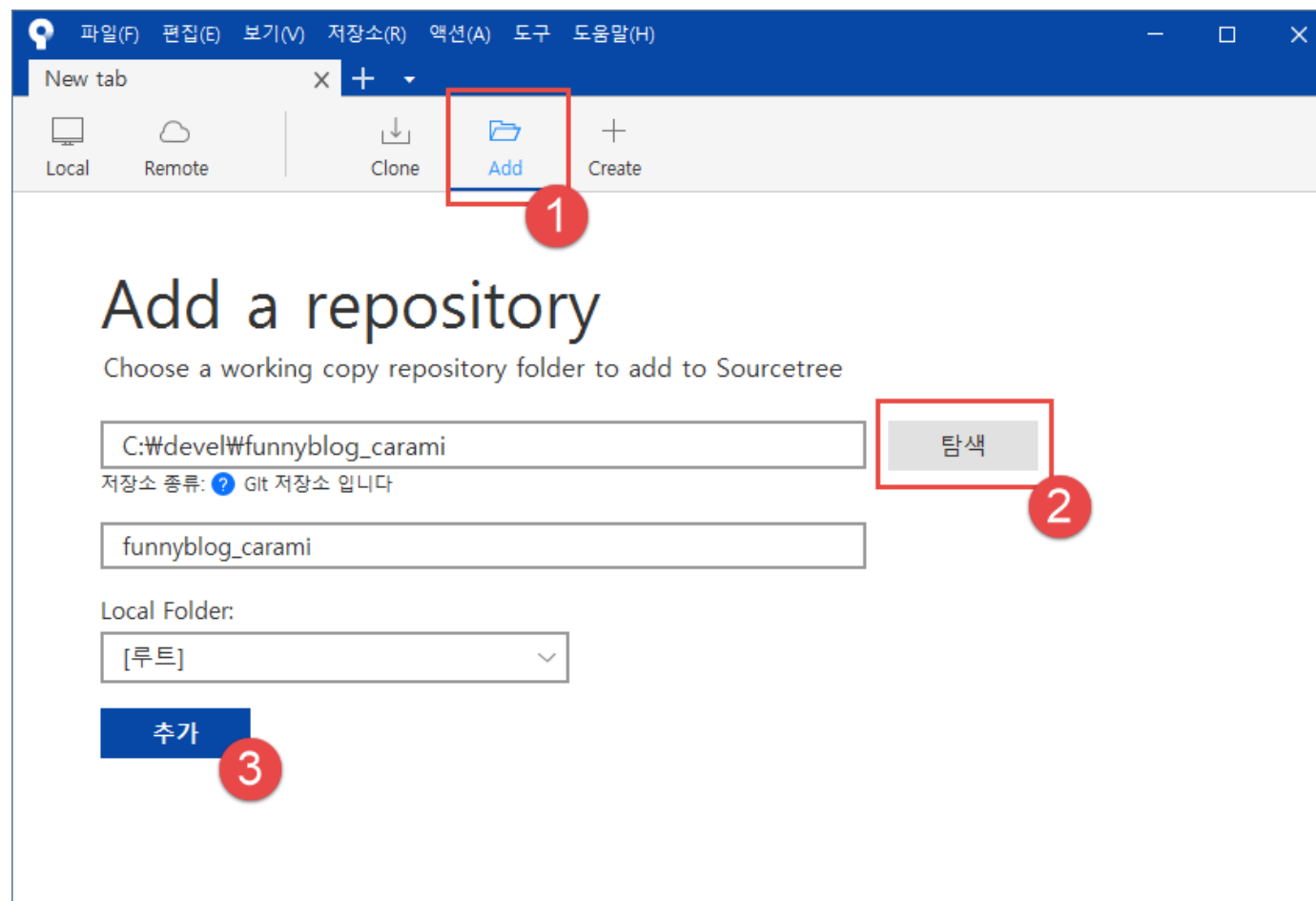
프로젝트 폴더(c:\devel\funnyblog_carami)

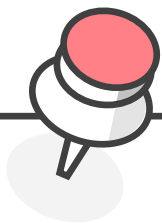




Git 워크플로우

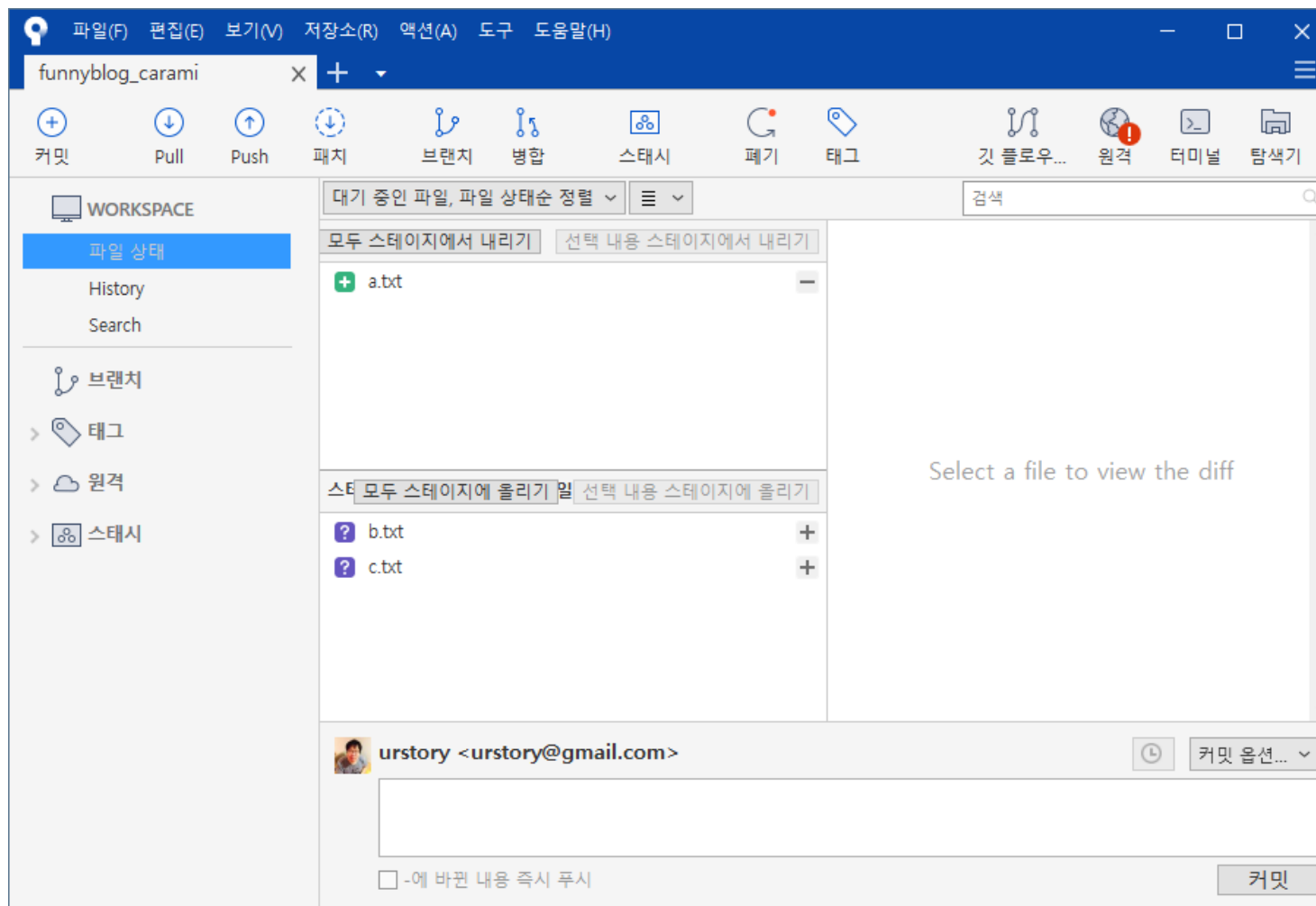
SourceTree에서 불러오기

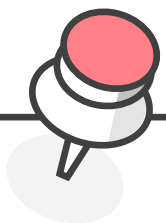




Git 워크플로우

SourceTree에서 불러오기

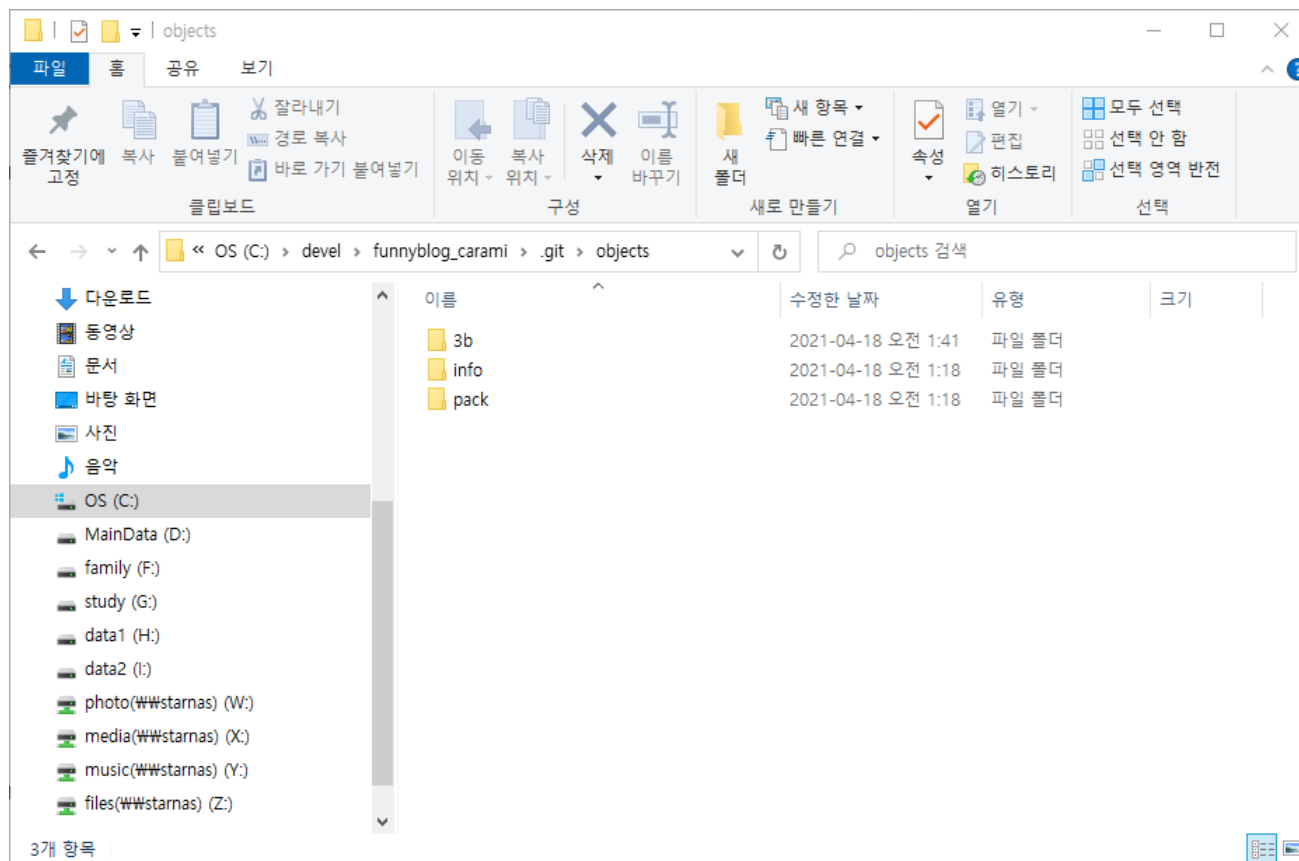


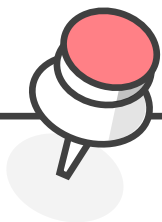


Git 워크플로우

c:\devel\funnyblog_carami\.git\objects

- staging area에 저장된다는 것은 .git\objects 폴더에 객체형식으로 저장된다는 것을 의미한다.

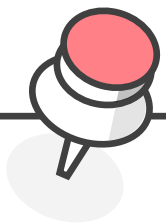




Git 워크플로우

```
git add b.txt  
git status
```

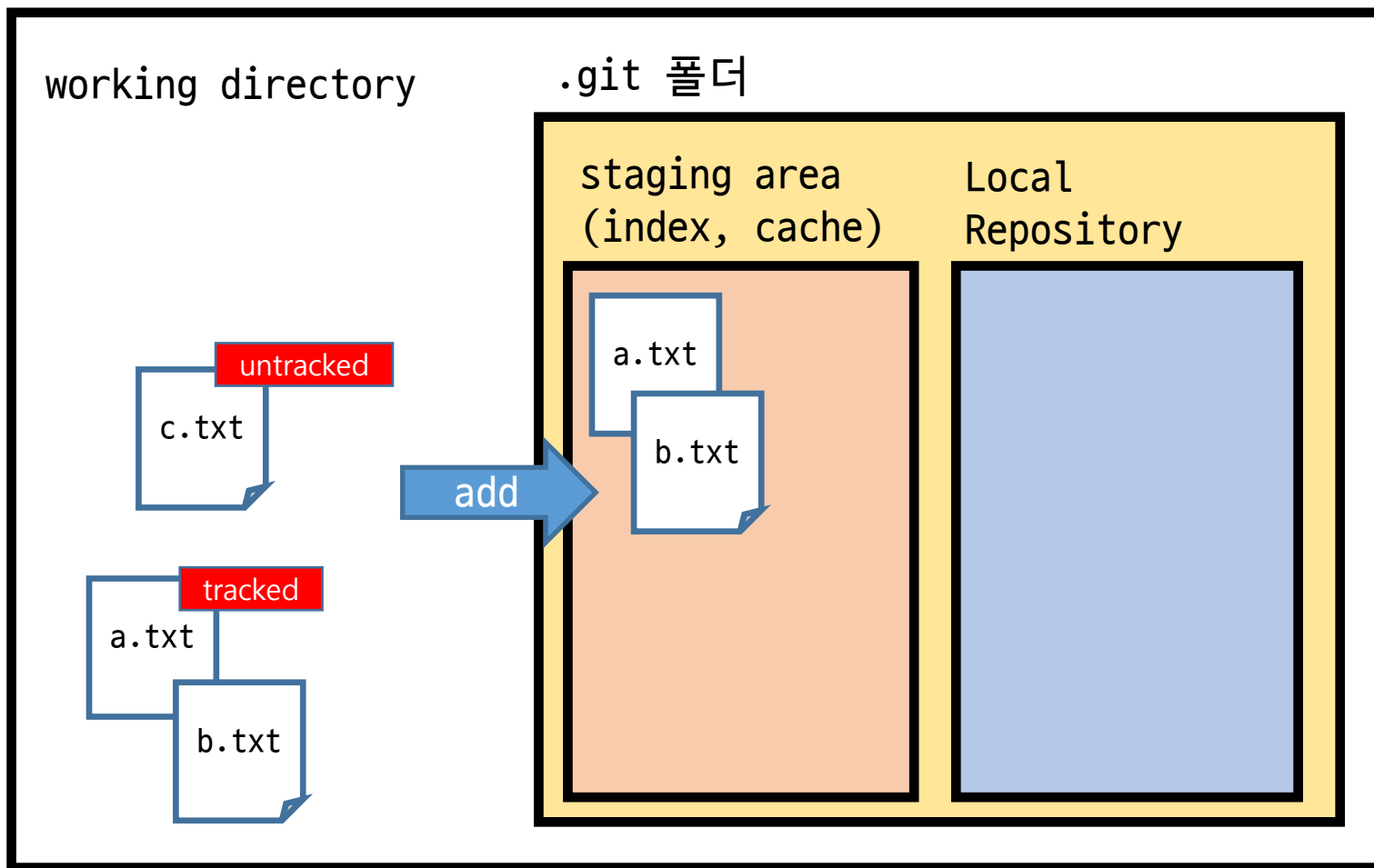
```
MINGW64:/c/devel/funnyblog_carami  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$ git add b.txt  
warning: LF will be replaced by CRLF in b.txt.  
The file will have its original line endings in your working directory  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$ git status  
On branch main  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   a.txt  
    new file:   b.txt  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    c.txt  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$
```

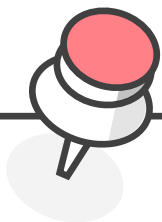


Git 워크플로우

```
git add b.txt
```

프로젝트 폴더(c:\devel\funnyblog_carami)





Git 워크플로우

echo hi >> a.txt
git status
>> : append 한다.

```
MINGW64:/c:/devel/funnyblog_carami

urstory@DESKTOP-4SM171R MINGW64 /c:/devel/funnyblog_carami (main)
$ echo hi >> a.txt

urstory@DESKTOP-4SM171R MINGW64 /c:/devel/funnyblog_carami (main)
$ git status
On branch main

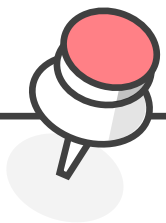
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   a.txt
    new file:   b.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   a.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    c.txt

urstory@DESKTOP-4SM171R MINGW64 /c:/devel/funnyblog_carami (main)
$ |
```

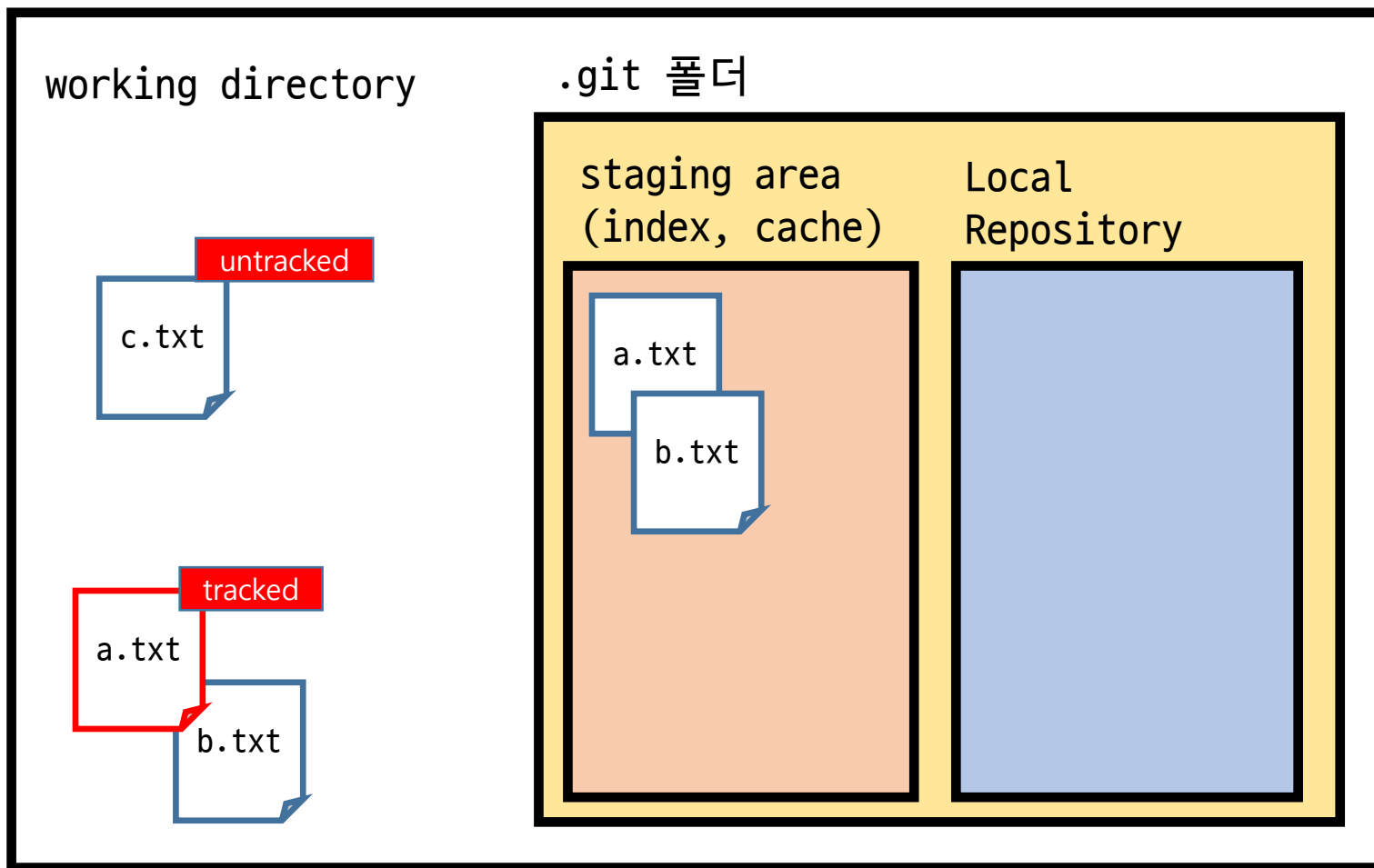


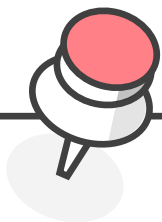
Git 워크플로우

```
echo hi >> a.txt  
git status
```

수정된 a.txt 파일은 아직 staging area에 올라가지 않았다.

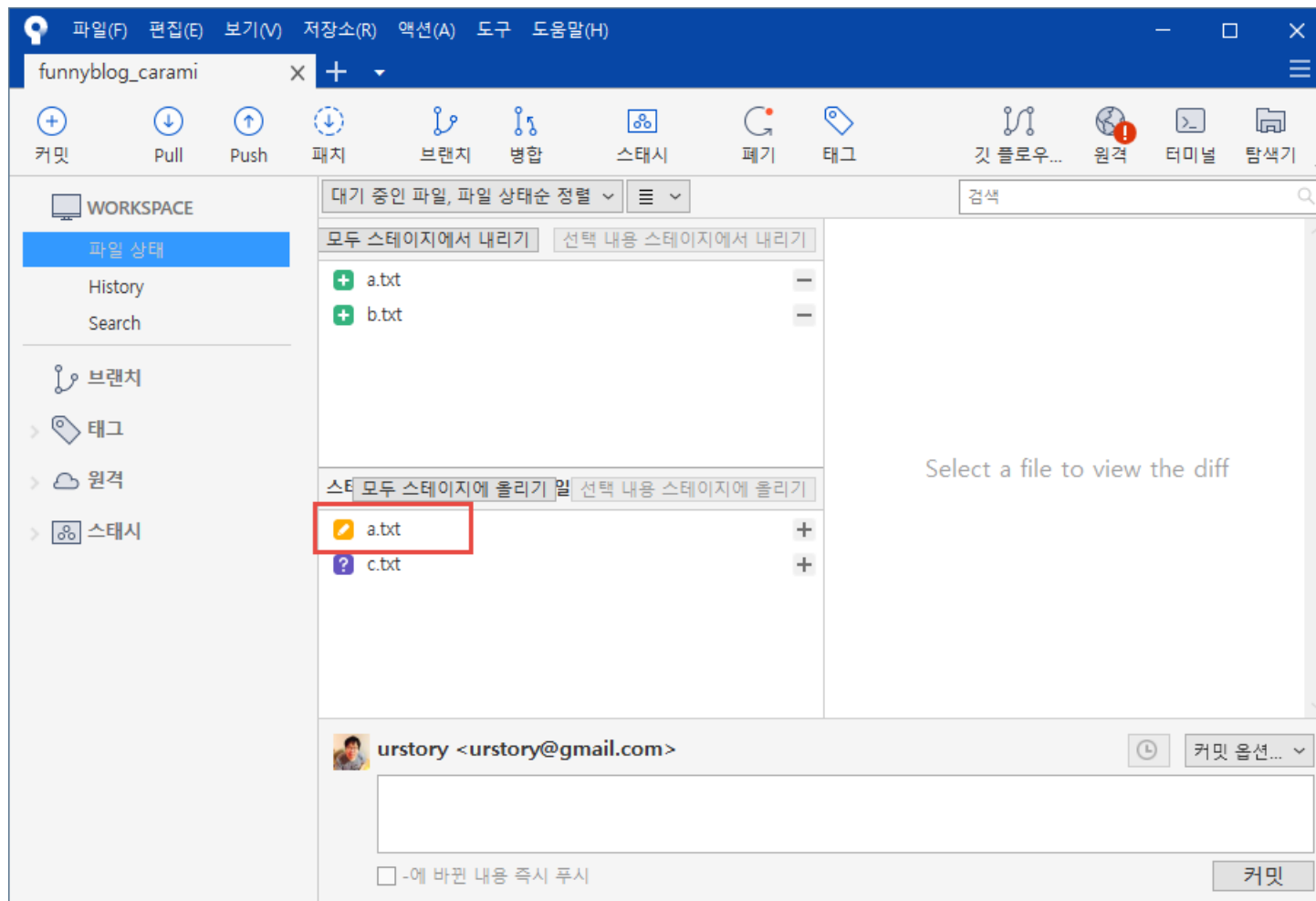
프로젝트 폴더(c:\devel\funnyblog_carami)

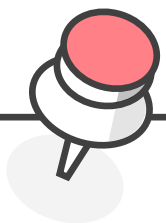




Git 워크플로우

SourceTree에서 불러오기
- 아이콘을 잘 살펴봅시다.





Git 워크플로우

git commit -m "a.txt, b.txt를 커밋합니다."

- 보통 하나의 기능 구현 단위를 commit이라고 한다.
- 파일 2개를 수정하여 기능을 구현하였다면, 파일 2개를 add하고 commi한다.

```
MINGW64:/c/devel/funnyblog_carami

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git commit -m "a.txt, b.txt를 커밋합니다."
[main (root-commit) 0796a96] a.txt, b.txt를 커밋합니다.
2 files changed, 2 insertions(+)
create mode 100644 a.txt
create mode 100644 b.txt

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ |
```

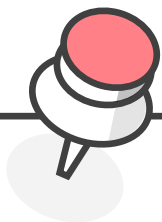
```
MINGW64:/c/devel/funnyblog_carami

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   a.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        c.txt

no changes added to commit (use "git add" and/or "git commit -a")

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ |
```

Git 워크플로우

리비전(revision) 조회하기

- 리비전이란 특정 오브젝트를 가리킬 수 있는 표현식을 말한다.

git log

- 40글자나 되는 긴 SHA-1 해시 값. 보통 줄여서 사용한다.

git log --abbrev-commit --pretty=oneline

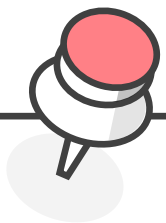
- 짧고 중복되지 않는 해시값을 보여준다.

```
MINGW64:/c/devel/funnyblog_carami

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git log
commit 0796a96a0d486f009bdd059b577c516f49f2f7ad (HEAD -> main)
Author: urstory <urstory@gmail.com>
Date:   Sun Apr 18 02:05:46 2021 +0900

    a.txt, b.txt를 커밋합니다.

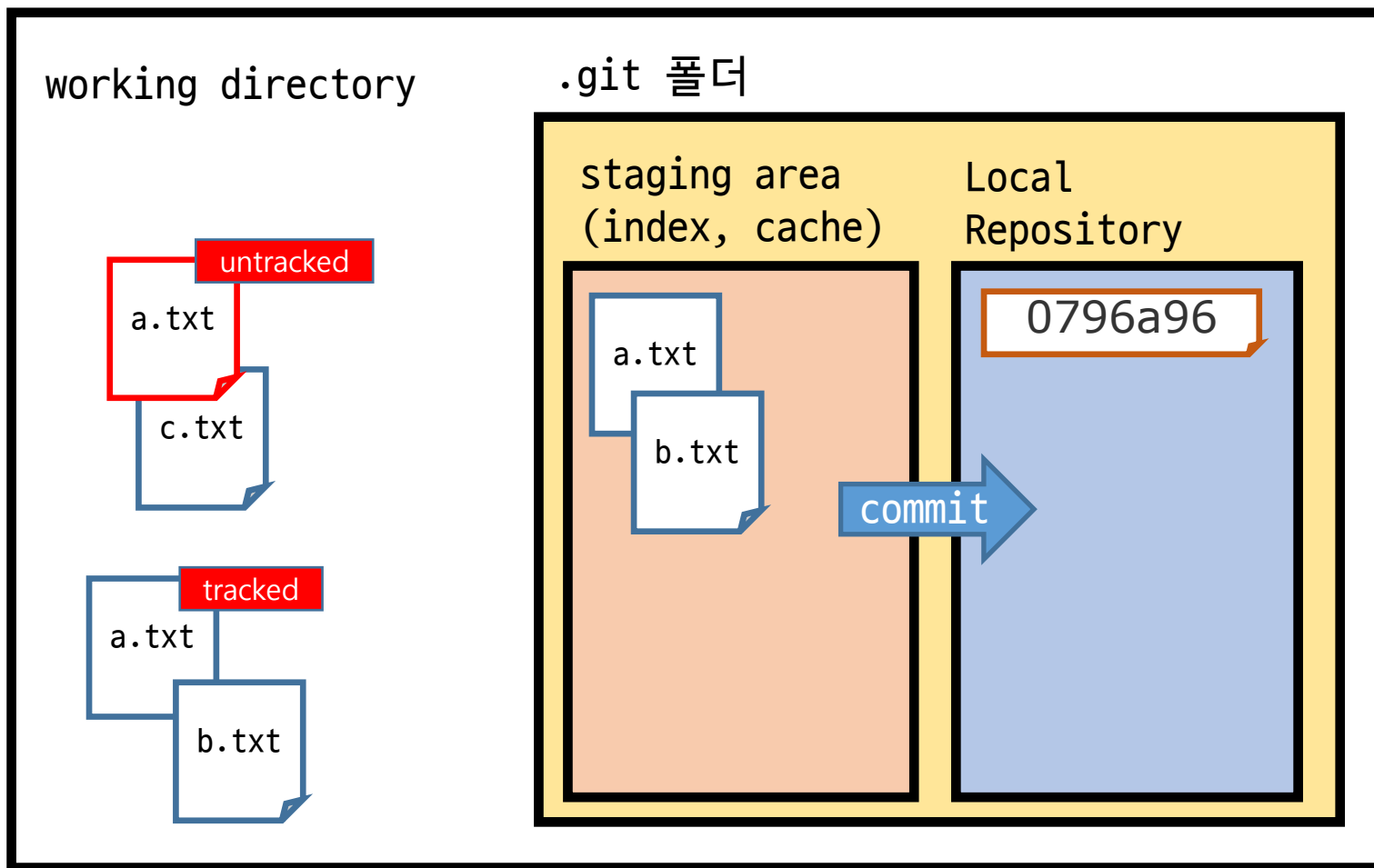
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$
```

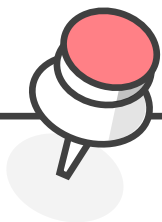


Git 워크플로우

```
git commit -m "a.txt, b.txt를 커밋합니다."
```

프로젝트 폴더(c:\devel\funnyblog_carami)





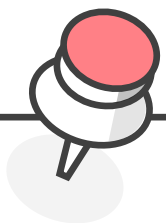
Git 워크플로우

SourceTree에서 불러오기
- 아이콘을 잘 살펴봅시다.

The screenshot shows the SourceTree application window for a repository named 'funnyblog_carami'. The interface is divided into several sections:

- Toolbar:** Contains icons for commit, pull, push, patch, branch, merge, status, revert, and tag.
- Left Panel:** Shows the 'Workspace' section with '파일 상태' (File Status) and 'History' (highlighted with a red box and a red circle with the number 1). Below it are '브랜치' (Branches) with 'main', '태그' (Tags), '원격' (Remotes), and '스태시' (Stashes).
- Commit History Table:** Displays a list of commits. The selected commit is highlighted in blue and has a red circle with the number 2 next to it. The commit details are as follows:

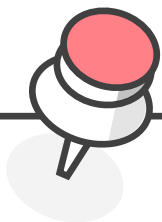
그래프	설명	날짜	작성자	커밋
○	커밋하지 않은 변경사항	18 4 2021 2:06	*	*
○	main a.txt, b.txt를 커밋합니다.	18 4 2021 2:05	urstory <urstory@gmail.com>	0796a96
- Bottom Panel:** Shows the '파일 상태' (File Status) section with a list of files: 'a.txt' and 'b.txt'. The 'a.txt' file is selected, and its details are shown in the right pane, including the commit hash '0796a96a0d486f009bdd059b577c516f49f2f7ad' and the file content 'hello world'.



Git 워크플로우

```
git add a.txt  
git status
```

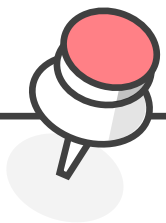
```
MINGW64:/c/devel/funnyblog_carami  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$ git add a.txt  
warning: LF will be replaced by CRLF in a.txt.  
The file will have its original line endings in your working directory  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$ git status  
On branch main  
Changes to be committed:  
  (use "git restore --staged <file>..." to unstage)  
    modified:   a.txt  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    c.txt  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$ |
```



Git 워크플로우

```
git commit -m "a파일에 hi를 추가"  
git log
```

```
MINGW64:/c/devel/funnyblog_carami  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$ git commit -m "a파일에 hi를 추가"  
[main 90a1741] a파일에 hi를 추가  
1 file changed, 1 insertion(+)  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$ git log  
commit 90a1741dd005ce9bf4af6486872cec8a9a08235e (HEAD -> main)  
Author: urstory <urstory@gmail.com>  
Date:   Sun Apr 18 02:23:05 2021 +0900  
  
    a파일에 hi를 추가  
  
commit 0796a96a0d486f009bdd059b577c516f49f2f7ad  
Author: urstory <urstory@gmail.com>  
Date:   Sun Apr 18 02:05:46 2021 +0900  
  
    a.txt, b.txt를 커밋합니다.  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)  
$ |
```

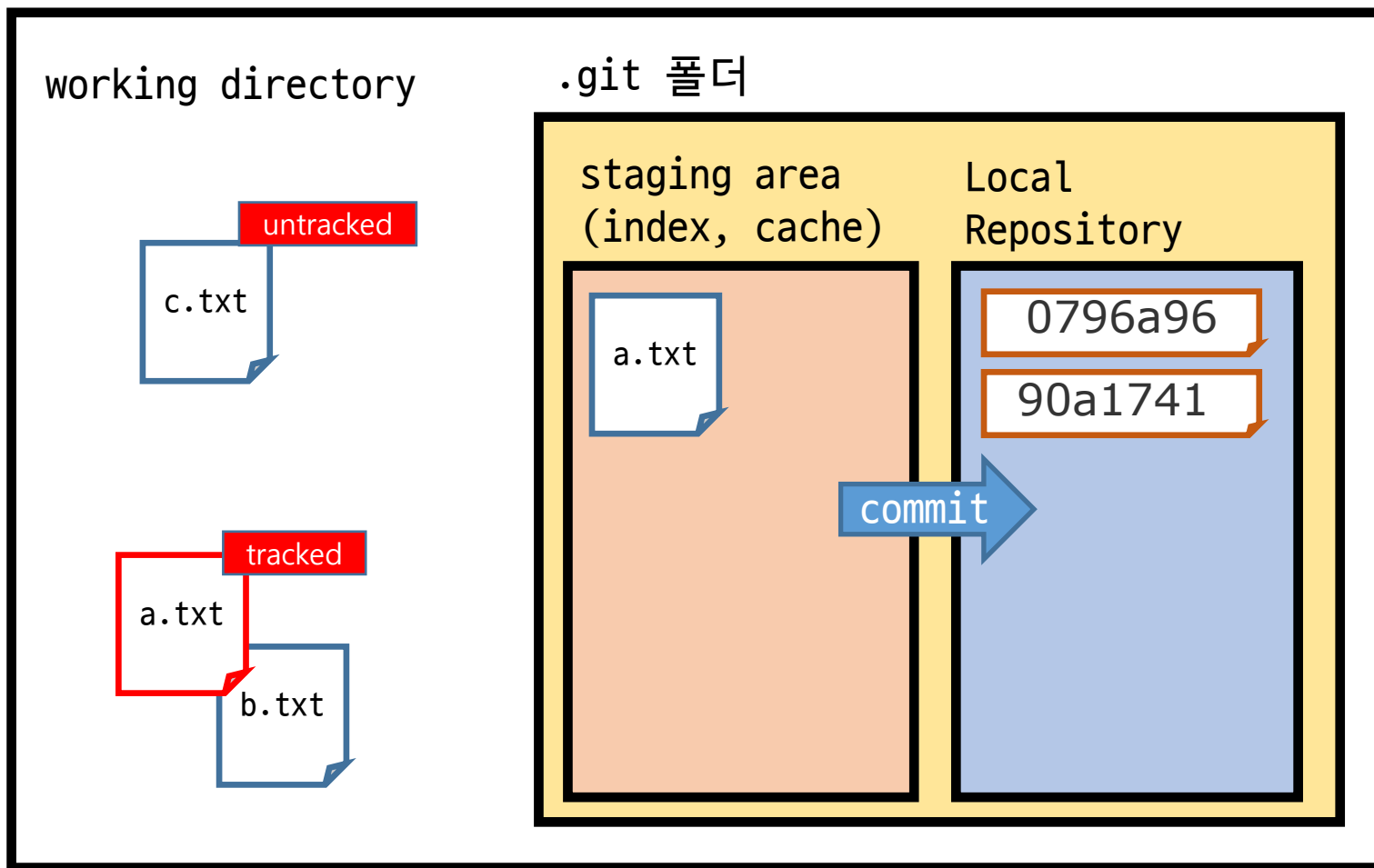


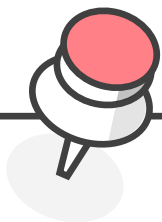
Git 워크플로우

```
git add a.txt
```

```
git commit -m "a파일에 hi를 추가"
```

프로젝트 폴더(c:\devel\funnyblog_carami)





Git 워크플로우

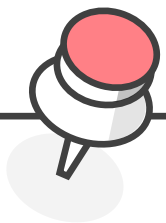
SourceTree에서 불러오기
- 아이콘을 잘 살펴봅시다.

The screenshot shows the SourceTree application window for a repository named 'funnyblog_carami'. The interface includes a menu bar, a toolbar with icons for commit, pull, push, fetch, branch, merge, stash, revert, and tag, and a sidebar with sections for Workspace, History, Search, Branches, Tags, Remotes, and Stashes.

The main area displays a commit history table with columns: 그래프 (Graph), 설명 (Description), 날짜 (Date), 작성자 (Author), and 커밋 (Commit). The table shows three commits, with the second commit selected.

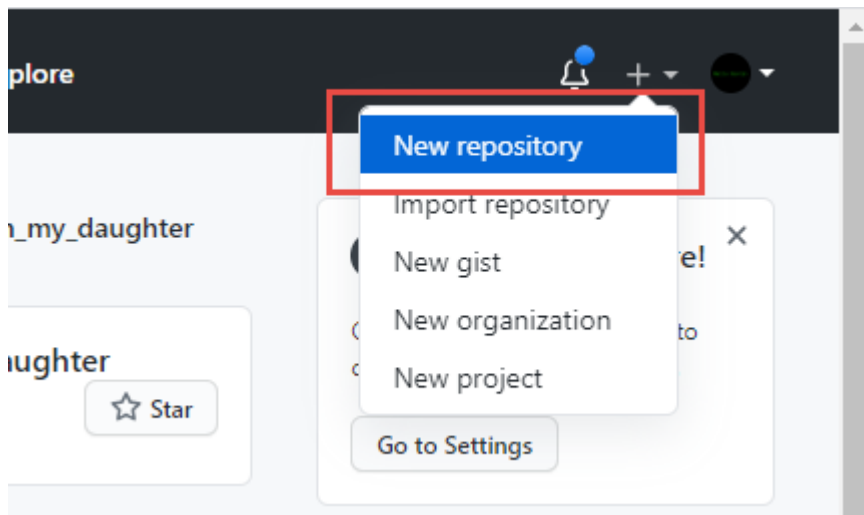
그래프	설명	날짜	작성자	커밋
○	커밋하지 않은 변경사항	18 4 2021 2:24	*	*
○	main a파일에 hi를 추가	18 4 2021 2:23	urstory <urstory@gmail.com>	90a1741
●	a.txt, b.txt를 커밋합니다.	18 4 2021 2:05	urstory <urstory@gmail.com>	0796a96


The bottom section shows the file 'a.txt' with its commit details and a diff view. The commit details include the commit hash (90a1741), the commit message (a파일에 hi를 추가), the author (urstory), and the date (2021년 4월 18일). The diff view shows the changes made in the commit, with the line 'hi' added to the file.



Git 워크플로우

Github에 funnyblog 레포지토리를 생성합니다.



 Search or jump to... Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

Repository name *

urstory / funnyblog

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

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

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

Initialize this repository with:

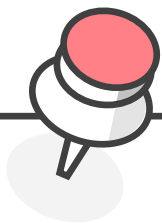
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository



Git 워크플로우

원격 저장소를 설정합니다.

```
git remote add origin https://github.com/urstory/funnyblog.git
```

원격 저장소에 현재 레포지토리의 내용을 올립니다(push).

```
git push -u origin main
```

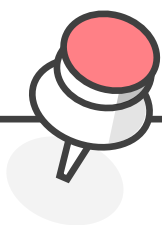
** https://github.com/urstory/funnyblog.git 주소를 origin이라는 이름으로 사용한다는 의미이다.

** main은 브랜치(branch) 이름이다.

** 원래 가장 기본이 되는 브랜치 이름은 master 였는데 main으로 바뀜.

노예제가 연상된다는 이유로 바뀌었다.

<https://www.clien.net/service/board/news/15071494>



Git 워크플로우

```
MINGW64:/c/devel/funnyblog_carami

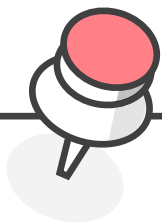
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git remote add origin https://github.com/urstory/funnyblog.git

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git push -u origin main

Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 517 bytes | 517.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/urstory/funnyblog.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ |
```



Git 워크플로우

https://github.com/urstory/funnyblog

- GitHub에 가보면 코드가 올라가 있는 것을 확인할 수 있다. 커밋 정보도 확인가능하다.

github.com/urstory/funnyblog

Search or jump to...

Pull requests Issues Marketplace Explore

urstory / funnyblog

Unwatch 1 Star 0 Fork 0

<> Code ! Issues ? Pull requests ? Actions ? Projects ? Wiki ? Security ? Insights ? Settings

main 1 branch 0 tags

Go to file Add file Code

urstory	a파일에 hi를 추가	90a1741	10 minutes ago	2 commits
a.txt	a파일에 hi를 추가		10 minutes ago	
b.txt	a.txt, b.txt를 커밋합니다.		27 minutes ago	

Help people interested in this repository understand your project by adding a README. Add a README

About

No description, website, or topics provided.

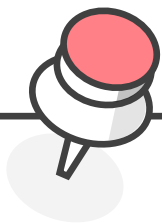
Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

© 2021 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About



Git 워크플로우

https://github.com/urstory/funnyblog

←

→

↺

github.com/urstory/funnyblog

☆

👤

🖼️

Search or jump to...

/

Pull requests

Issues

Marketplace

Explore

urstory / funnyblog

👁️ Unwatch ▾ 1

<> Code

ⓘ Issues

🔗 Pull requests

⌚ Actions

📁 Projects

📖 Wiki

🛡️ Security

📈 Insights

🔗 main ▾

👤 1 branch

🏷️ 0 tags

Go to file

Add file ▾

📄 Code ▾

🌲 Octo tree ▾

👤 urstory

a파일에 hi를 추가

90a1741 11 minutes ago ⌚ 2 commits

📄 a.txt

a파일에 hi를 추가

11 minutes ago

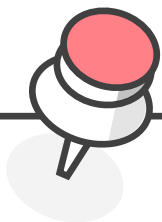
📄 b.txt

a.txt, b.txt를 커밋합니다.

29 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README



Git 워크플로우

<https://github.com/urstory/funnyblog/blob/main/a.txt>

- 커밋 리비전 번호를 확인할 수 있다.
- 추가된 내용을 확인할 수 있다.

The screenshot shows a web browser displaying a GitHub repository page for `urstory/funnyblog`. The URL in the address bar is `github.com/urstory/funnyblog/blob/main/a.txt`. The page header includes a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. Below the header, the repository name `urstory / funnyblog` is shown with 1 watch, 0 stars, and 0 forks. A navigation bar contains links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content area shows the file `main / funnyblog / a.txt`. A commit history table lists a single commit by `urstory` with the message "a파일에 hi를 추가" (Added hi to a file). The commit hash `90a1741` and the time "13 minutes ago" are highlighted with a red box. To the right of the commit is a "History" link. Below the commit table, the file content is displayed, showing 2 lines (2 sloc) and 15 Bytes. The content is:

```
1 hello world
2 hi
```

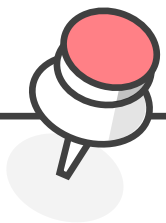
Commit	Message	Latest commit
<code>90a1741</code>	urstory a파일에 hi를 추가	Latest commit <code>90a1741</code> 13 minutes ago

1 contributor

2 lines (2 sloc) | 15 Bytes

Raw Blame

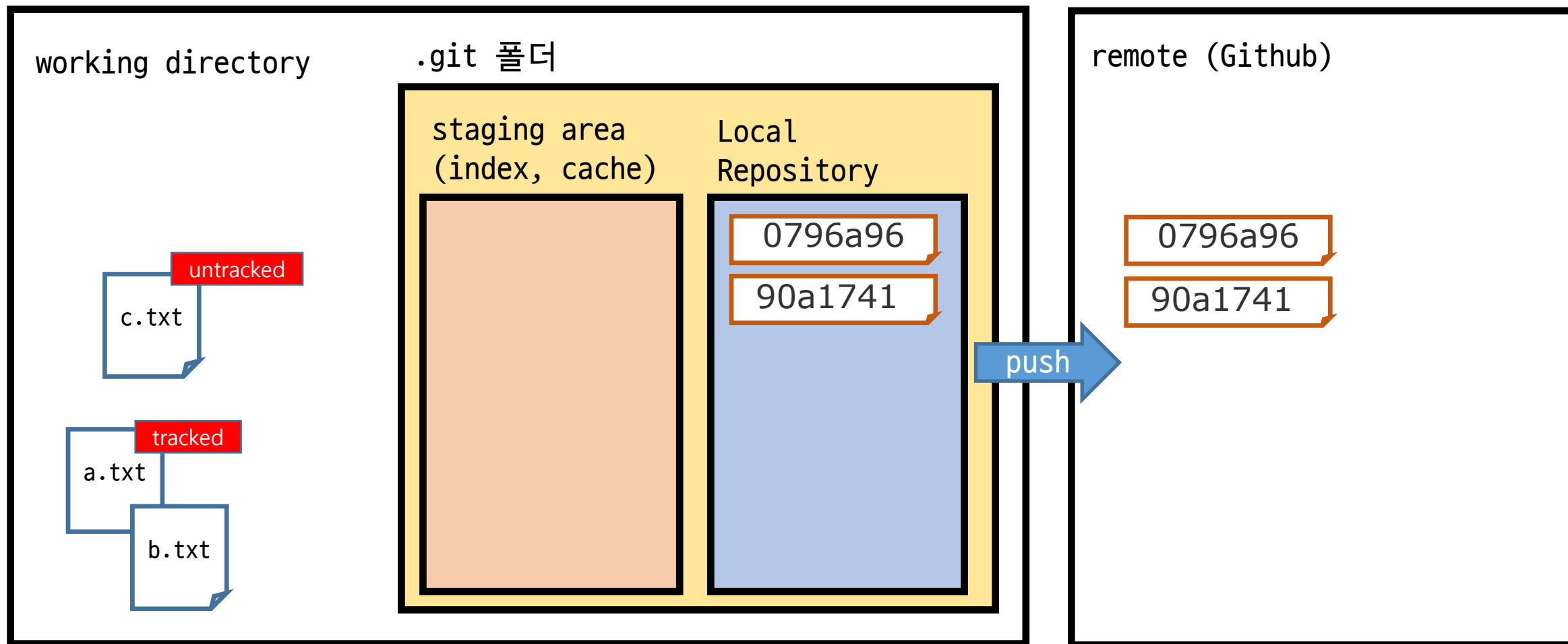
```
1 hello world
2 hi
```

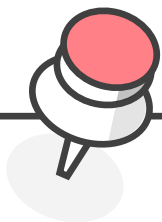


Git 워크플로우

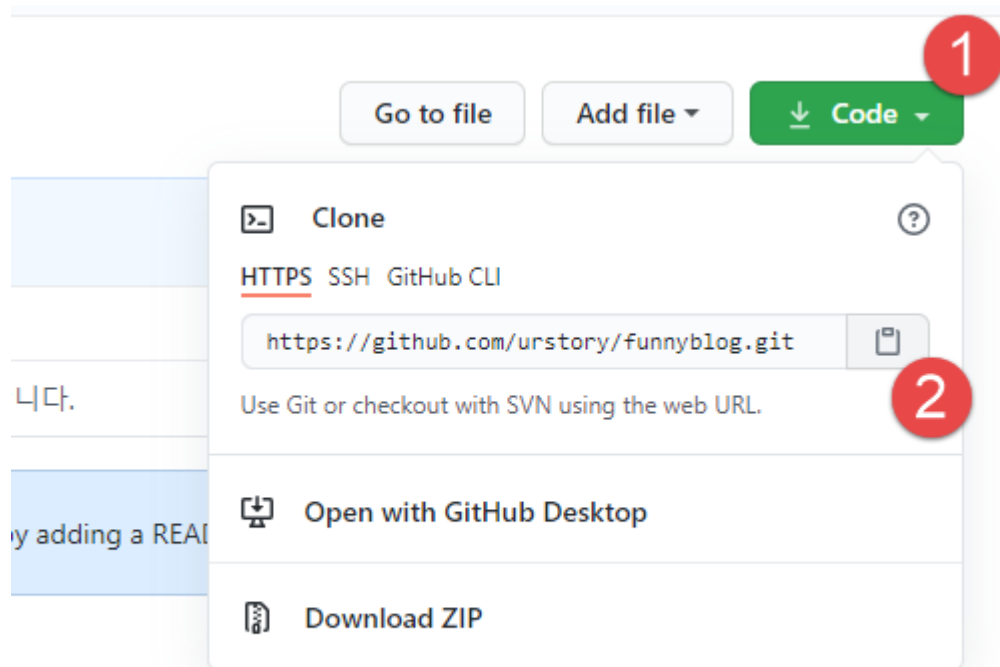
```
git push -u origin main
```

프로젝트 폴더(c:\devel\funnyblog_carami)





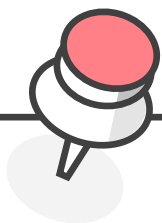
Git 워크플로우



```
mkdir /c/devel/funnyblog_esther  
cd /c/devel/funnyblog_esther
```

```
git clone https://github.com/urstory/funnyblog.git .
```

- 맨 뒤에 점을 반드시 입력해야 한다. 현재 경로 안에 소스를 가지고 온다.



Git 워크플로우

```
MINGW64:/c/devel/funnyblog_esther

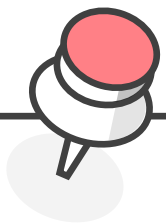
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ mkdir /c/devel/funnyblog_esther

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ cd /c/devel/funnyblog_esther

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_esther
$ ls

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_esther
$ git clone https://github.com/urstory/funnyblog.git .
Cloning into '.'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_esther (main)
$ |
```

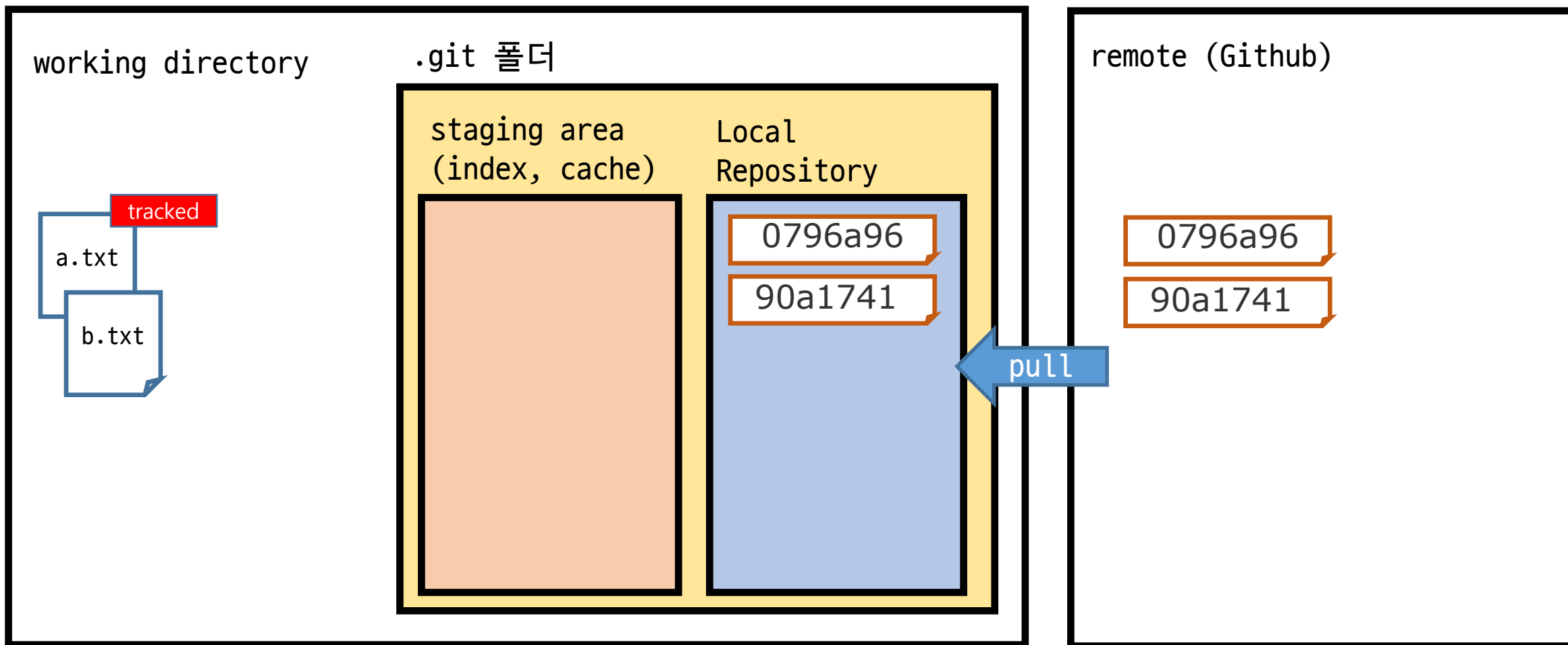



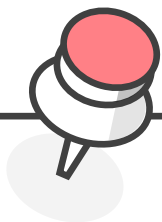
Git 워크플로우

/c/devel/funnyblog_esther 폴더에서 실행한다.

```
git clone https://github.com/urstory/funnyblog.git .
```

프로젝트 폴더(c:\devel\funnyblog_esther)

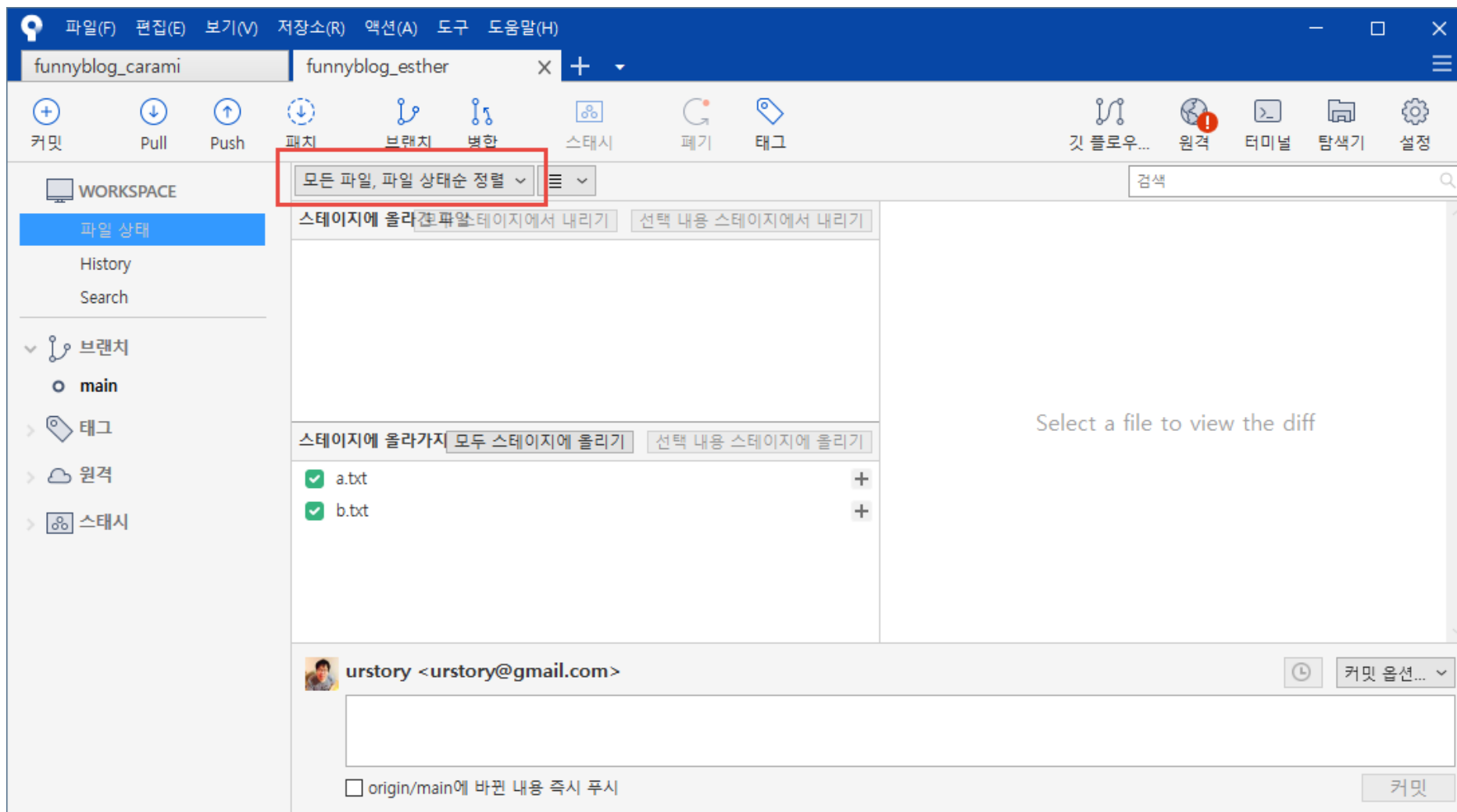


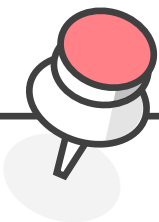


Git 워크플로우

SourceTree에서 불러오기

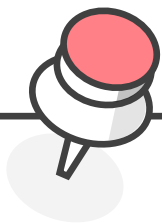
- 상단의 + 버튼을 클릭한 후, Add 아이콘을 클릭하여 추가한다.
- 모든 파일, 파일 상태순 정렬을 선택한다.





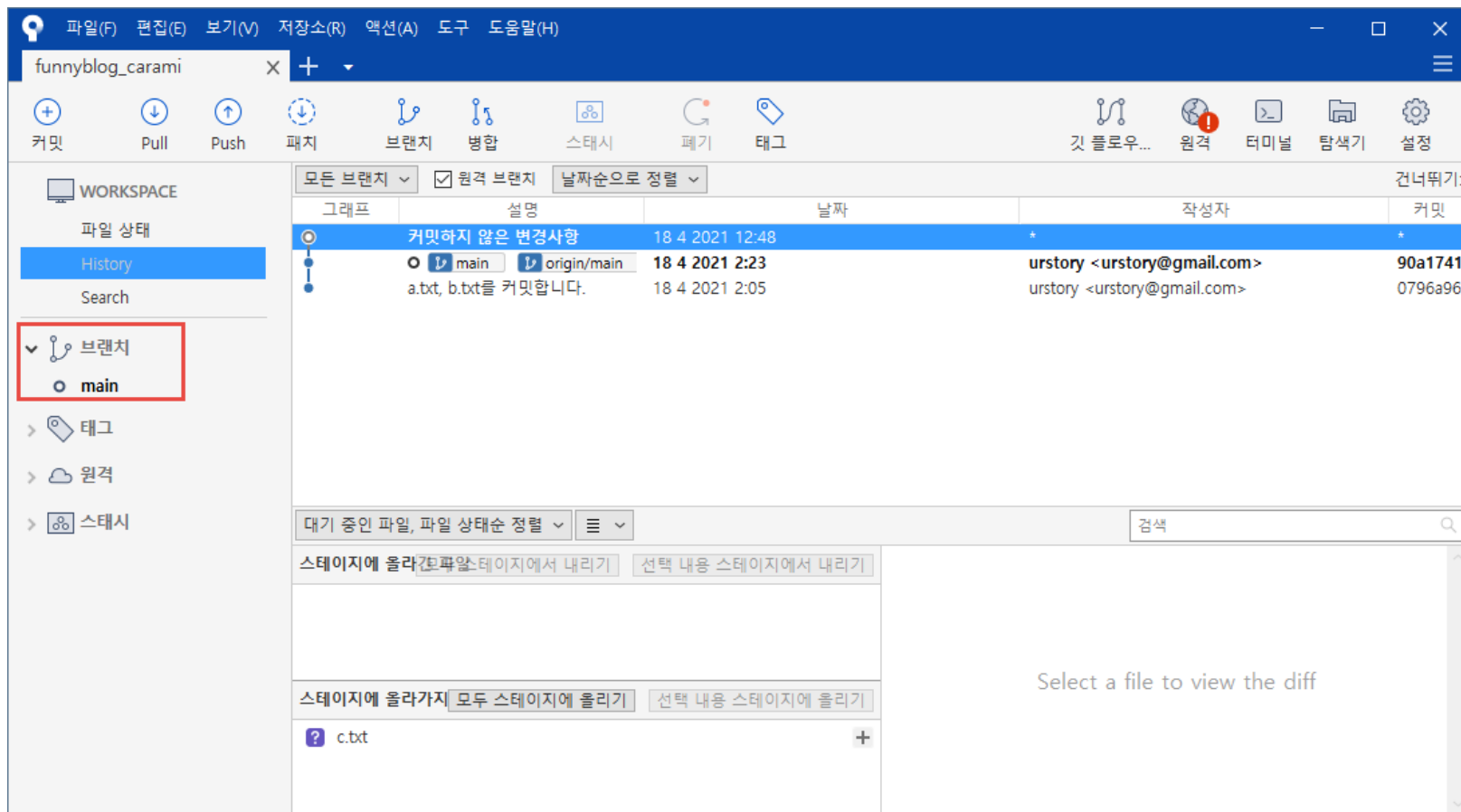
브랜치(branch)

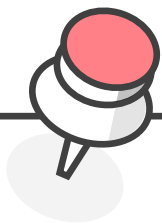
- 소프트웨어는 계속 유지 보수 하며 발전한다.
- 하나의 기능은 여러 개의 commit으로 만들어질 수 있다.
- git 저장소를 만들면 main 브랜치가 만들어진다. 해당 브랜치는 소스코드가 계속 유지보수하기 위한 큰 가지라고 생각하면 된다. 이 큰 가지의 내용이 빌드(build)되고 배포(deploy)가 된다.
- 개발을 하다보면, 새로운 기능을 만들어야 할 필요가 있다. 새로운 기능도 여러 개의 코드 뭉치(여러 개의 commit)으로 이뤄지는데, 해당 기능이 모두 만들어지기 전에는 main 브랜치에 포함이 되면 안된다고 생각할 때가 있다. 이 경우 새로운 가지를 만든 후 기능을 구현할 필요가 있게 된다. 새로운 가지에서 기능을 모두 구현한 후에는 해당 내용을 main브랜치에 적용하고 싶을 경우가 생긴다. 이를 머지(merge)라고 한다.
- 어떤 서비스냐에 따라서 브랜치를 잘 관리할 필요가 있는데, 이러한 전략들로는 Github flow, Git Flow, GitLab Flow등이 있다.



브랜치(branch)

- git 으로 레포지토리를 생성하면 "main"이라는 이름의 브랜치가 기본적으로 생성된다.
- 예전에는 "main"이 아니라 "master"라는 이름이 사용되었음.

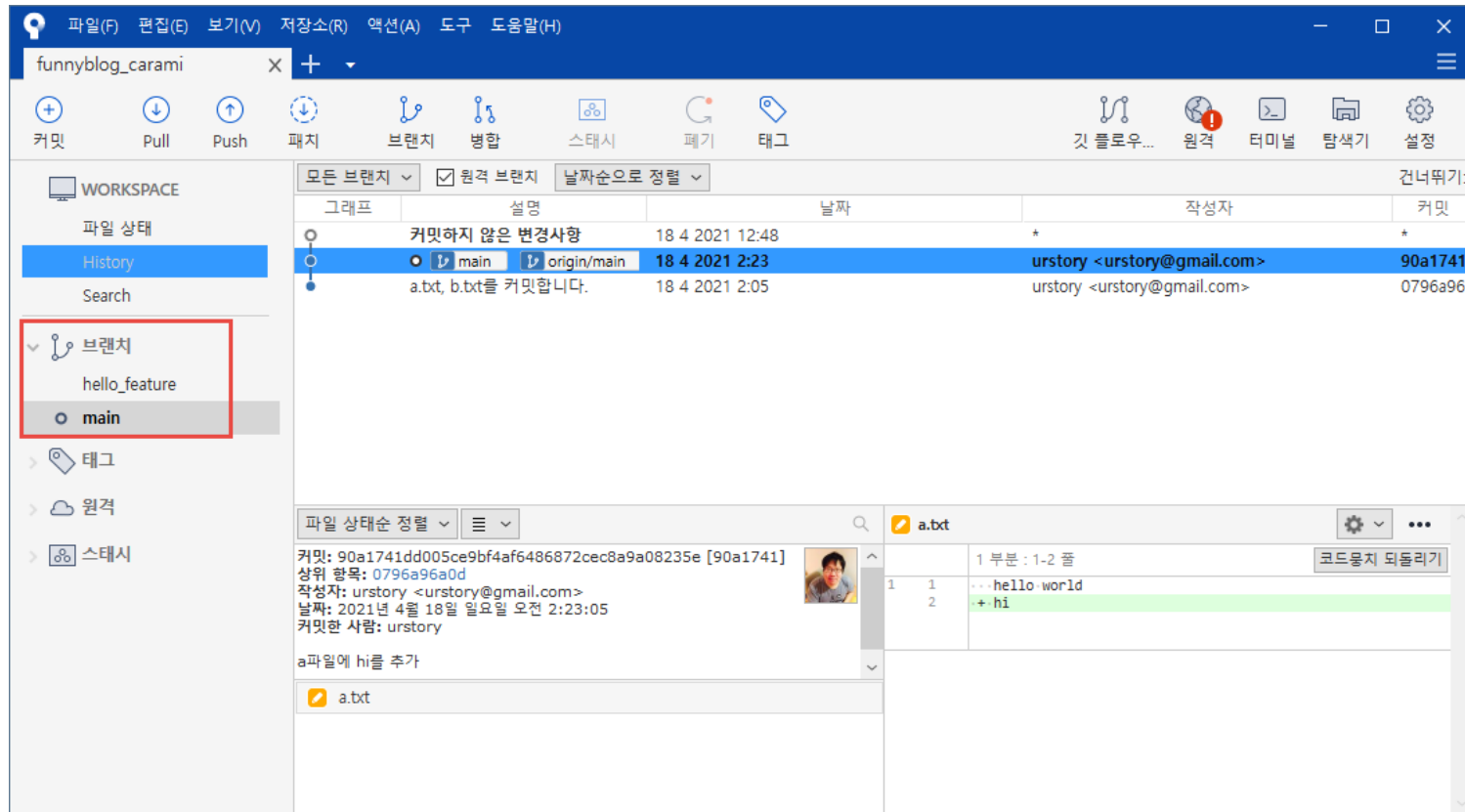


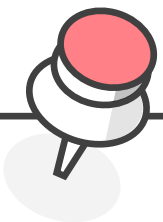


브랜치(branch)

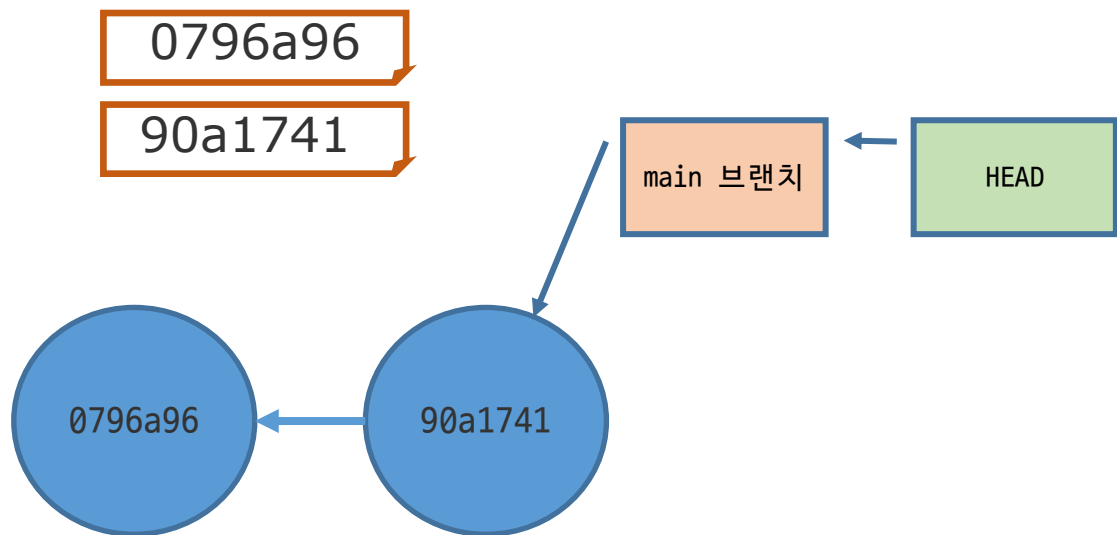
git branch hello_feature

- hello_feature 브랜치를 생성한다.
- 보통 브랜치는 기능별로 만든다.
- SourceTree로 보면 hello_feature가 생성된 것을 볼 수 있다. main앞에 점이 찍혀있는데 현재 사용중인 브랜치라는 것을 의미한다.





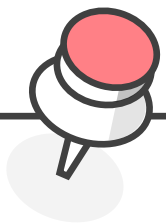
브랜치(branch)



- Git은 다른 버전 관리 시스템과는 다르게 'HEAD'라는 특수한 포인트가 있다. 이 포인트를 이용하여 현재 작업 중인 브랜치가 무엇인지 안다.
- `git log --oneline --decorate`
- `--decorate` 옵션으로 알 수 있다.

MINGW64:/c/devel/funnyblog_carami

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git log --oneline --decorate
90a1741 (HEAD -> main, origin/main, hello_feature) a파일에 hi를 추가
0796a96 a.txt, b.txt를 커밋합니다.
```



브랜치(branch)

git checkout hello_feature

- hello_feature 브랜치로 바꾼다.
- 프롬프트의 괄호 안의 브랜치 명이 바뀐 것을 알 수 있다.
- SourceTree의 점도 바뀌었다.

MINGW64:/c/devel/funnyblog_carami

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git checkout hello_feature
Switched to branch 'hello_feature'

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (hello_feature)
$ |
```

The screenshot shows the SourceTree application window for the 'funnyblog_carami' repository. The 'WORKSPACE' panel on the left shows the 'hello_feature' branch selected, with 'main' listed below it. The 'Commit History' panel shows a list of commits. The 'a.txt' file is open in the editor, showing the content 'hello world' and 'hi'.

그래프	설명	날짜	작성자	커밋
○	커밋하지 않은 변경사항	18 4 2021 12:48	*	*
●	main origin/main	18 4 2021 2:23	urstory <urstory@gmail.com>	90a1741
●	a.txt, b.txt를 커밋합니다.	18 4 2021 2:05	urstory <urstory@gmail.com>	0796a96

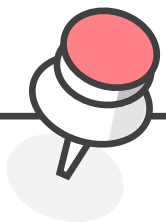
파일 상태: a.txt

커밋: 90a1741dd005ce9bf4af6486872cec8a9a08235e [90a1741]
상위 항목: 0796a96a0d
작성자: urstory <urstory@gmail.com>
날짜: 2021년 4월 18일 일요일 오전 2:23:05
커밋한 사람: urstory

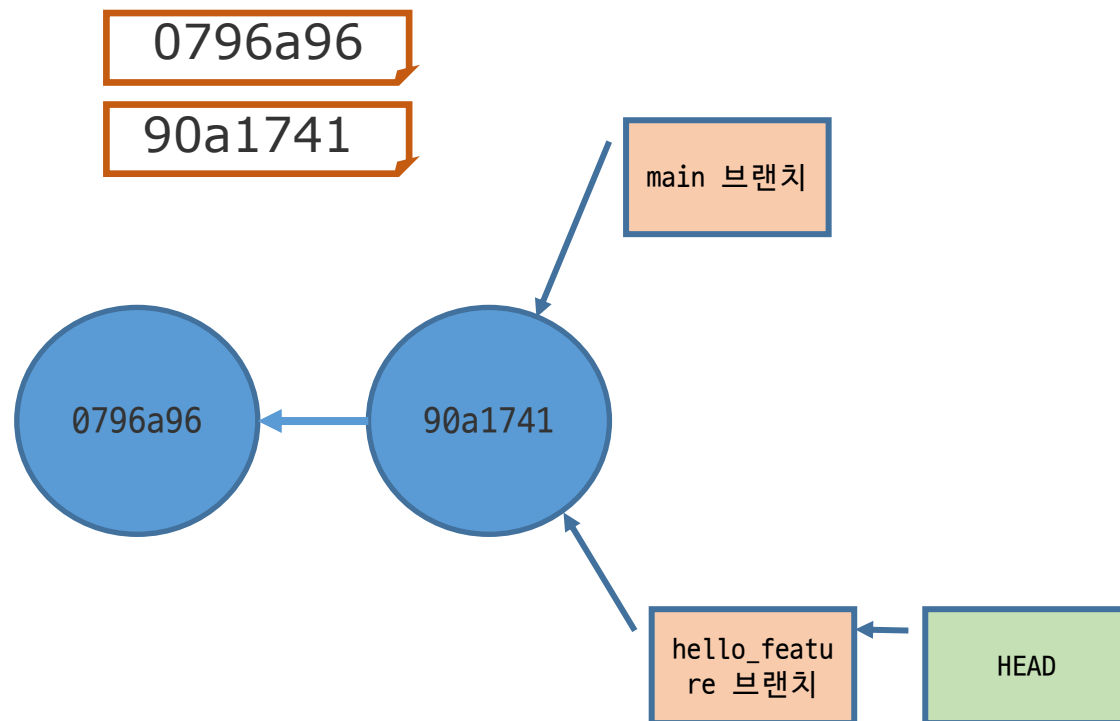
a파일에 hi를 추가

a.txt

1 부분: 1-2 줄
1 ... hello world
2 + hi

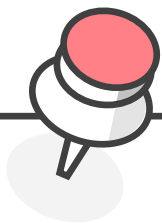


브랜치(branch)



MINGW64:/c/devel/funnyblog_carami

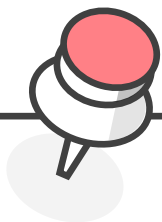
```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (hello_feature)
$ git log --oneline --decorate
90a1741 (HEAD -> hello_feature, origin/main, main) a과 일 에 hi를 추 가
0796a96 a.txt, b.txt를 커밋 합 니 다 .
```

Git 워크플로우

```
echo hello > hello.txt  
git add hello.txt  
git commit -m "hello.txt 파일 생성"
```

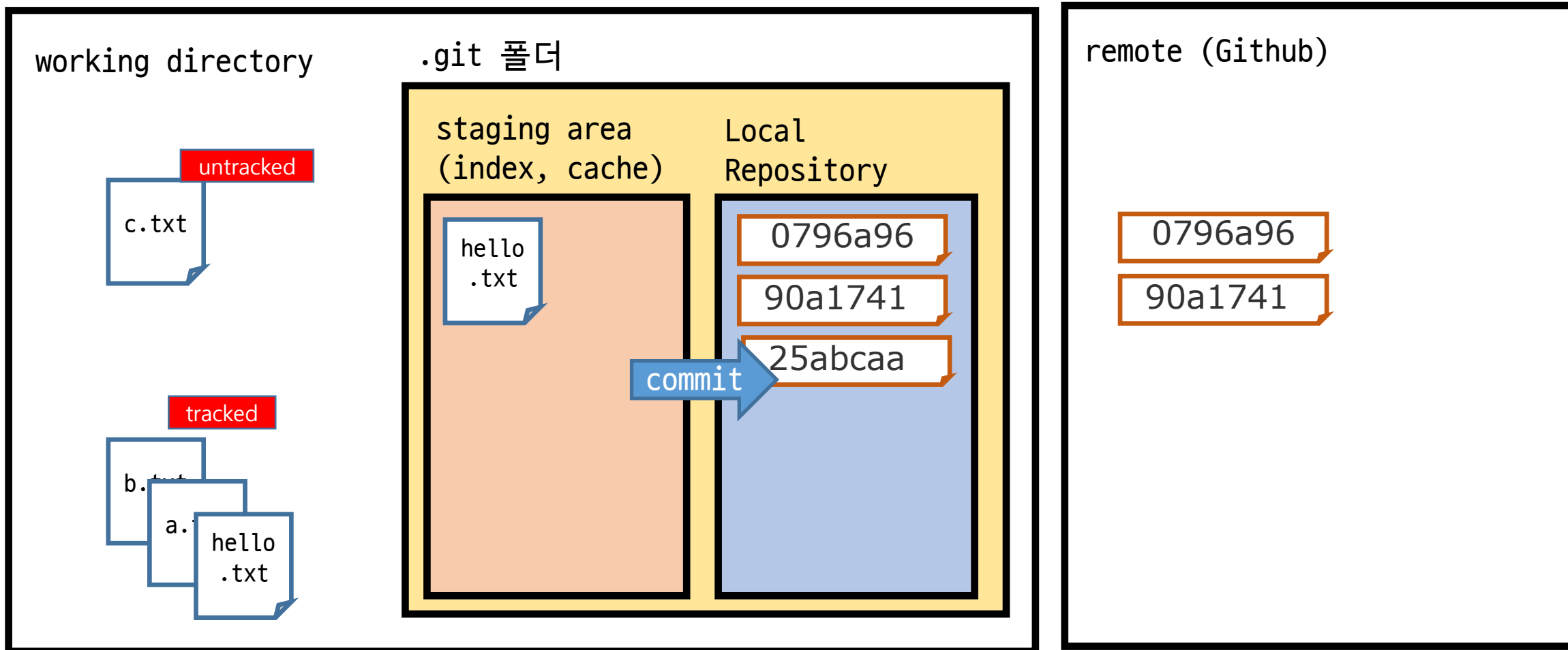
```
MINGW64:/c:/devel/funnyblog_carami  
urstory@DESKTOP-4SM171R MINGW64 /c:/devel/funnyblog_carami (hello_feature)  
$ echo hello > hello.txt  
  
urstory@DESKTOP-4SM171R MINGW64 /c:/devel/funnyblog_carami (hello_feature)  
$ git add hello.txt  
warning: LF will be replaced by CRLF in hello.txt.  
The file will have its original line endings in your working directory  
  
urstory@DESKTOP-4SM171R MINGW64 /c:/devel/funnyblog_carami (hello_feature)  
$ git commit -m "hello.txt 파일 생성"  
[hello_feature 25abcaa] hello.txt 파일 생성  
1 file changed, 1 insertion(+)  
create mode 100644 hello.txt  
  
urstory@DESKTOP-4SM171R MINGW64 /c:/devel/funnyblog_carami (hello_feature)  
$ git log  
commit 25abcaa928e07e683a1165ded02ac7340a074017 (HEAD -> hello_feature)  
Author: urstory <urstory@gmail.com>  
Date: Sun Apr 18 15:44:29 2021 +0900  
  
    hello.txt 파일 생성  
  
commit 90a1741dd005ce9bf4af6486872cec8a9a08235e (origin/main, main)  
Author: urstory <urstory@gmail.com>  
Date: Sun Apr 18 02:23:05 2021 +0900  
  
    a파일 에 hi를 추가  
  
commit 0796a96a0d486f009bdd059b577c516f49f2f7ad  
Author: urstory <urstory@gmail.com>  
Date: Sun Apr 18 02:05:46 2021 +0900  
  
    a.txt, b.txt를 커밋합니다.  
  
urstory@DESKTOP-4SM171R MINGW64 /c:/devel/funnyblog_carami (hello_feature)  
$
```

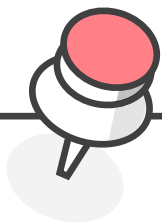


Git 워크플로우

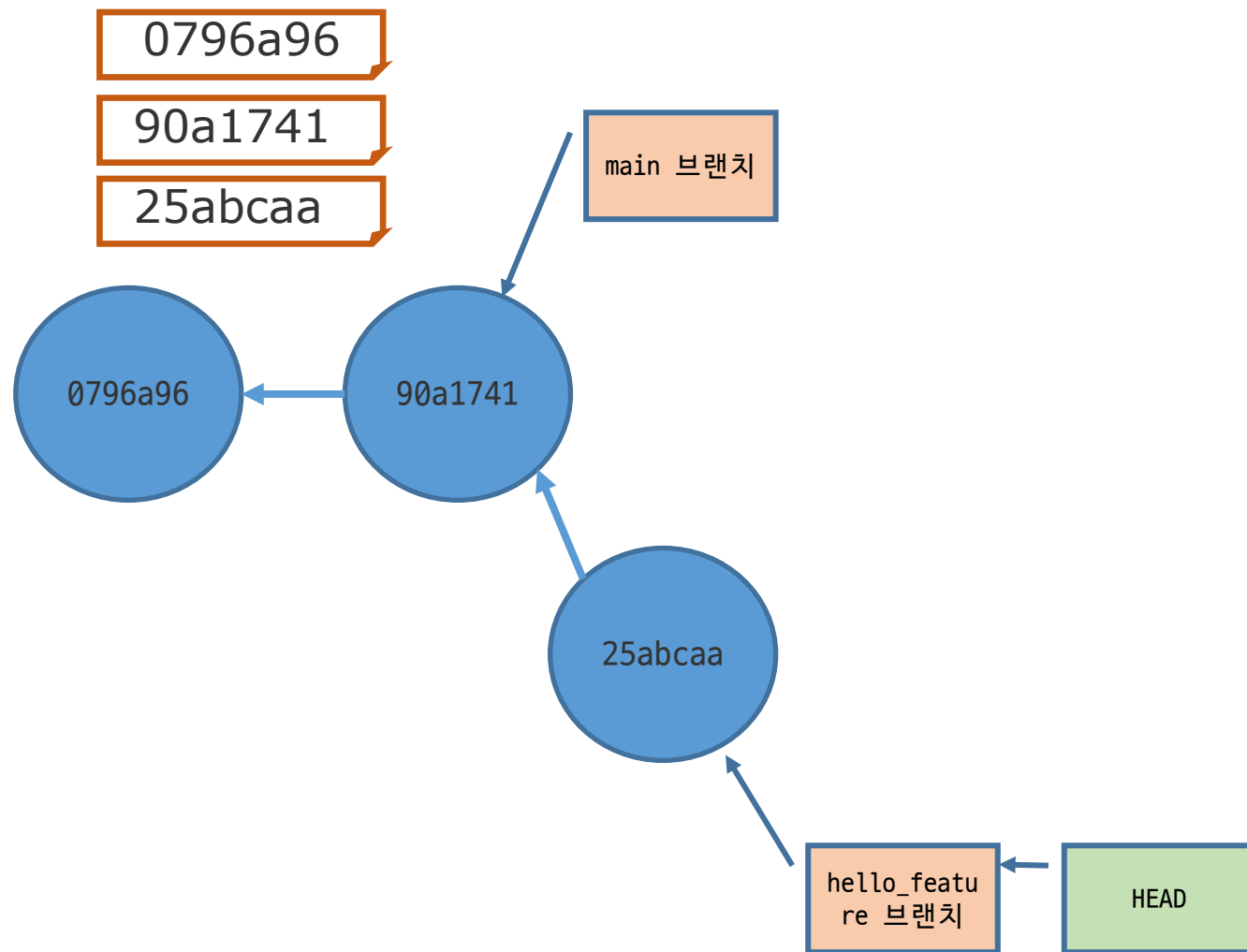
```
echo hello > hello.txt  
git add hello.txt  
git commit -m "hello.txt 파일 생성"
```

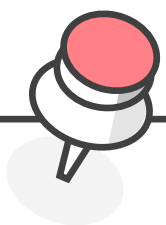
프로젝트 폴더(c:\devel\funnyblog_carami)





브랜치(branch)



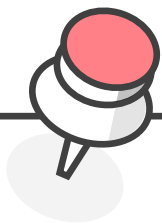


브랜치(branch)

hello_feature에는 파일이 4개 존재한다.

MINGW64:/c/devel/funnyblog_carami

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (hello_feature)
$ ls -la
total 12
drwxr-xr-x 1 urstory 197121  0 Apr 18 15:51 ./
drwxr-xr-x 1 urstory 197121  0 Apr 18 02:41 ../
drwxr-xr-x 1 urstory 197121  0 Apr 18 15:51 .git/
-rw-r--r-- 1 urstory 197121 15 Apr 18 02:01 a.txt
-rw-r--r-- 1 urstory 197121 12 Apr 18 01:36 b.txt
-rw-r--r-- 1 urstory 197121 12 Apr 18 01:36 c.txt
-rw-r--r-- 1 urstory 197121  7 Apr 18 15:51 hello.txt
```



브랜치(branch)

SourceTree에서 불러오기
- hello_feature 브랜치 모습

funnyblog_carami

커밋 Pull Push 패치 브랜치 병합 스테시 폐기 태그

WORKSPACE

- 파일 상태
- History
- Search

브랜치

- hello_feature
- main

태그

원격

스테시

그래프	설명	날짜	작성자	커밋
○	커밋하지 않은 변경사항	18 4 2021 15:47	*	*
●	hello_feature hello.txt	18 4 2021 15:44	urstory <urstory@gmail.com>	25abcaa
○	origin/main main	18 4 2021 2:23	urstory <urstory@gmail.com>	90a1741
○	a.txt, b.txt를 커밋합니다.	18 4 2021 2:05	urstory <urstory@gmail.com>	0796a96

파일 상태순 정렬

hello.txt

커밋: 25abcaa928e07e683a1165ded02ac7340a074017 [25abcaa]

상위 항목: 90a1741dd0

작성자: urstory <urstory@gmail.com>

날짜: 2021년 4월 18일 일요일 오후 3:44:29

커밋한 사람: urstory

hello.txt 파일 생성

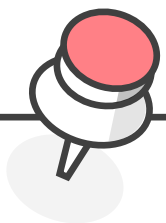
hello.txt

파일 내용

코드용지 되돌리기

0

+ hello



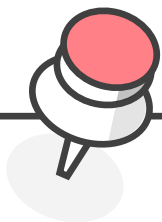
브랜치(branch)

main에는 파일이 3개 존재한다.

- hello_feature에서 작성한 파일 hello.txt는 존재하지 않는다.

MINGW64:/c/devel/funnyblog_carami

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ ls -la
total 11
drwxr-xr-x 1 urstory 197121  0 Apr 18 15:52 ./
drwxr-xr-x 1 urstory 197121  0 Apr 18 02:41 ../
drwxr-xr-x 1 urstory 197121  0 Apr 18 15:52 .git/
-rw-r--r-- 1 urstory 197121 15 Apr 18 02:01 a.txt
-rw-r--r-- 1 urstory 197121 12 Apr 18 01:36 b.txt
-rw-r--r-- 1 urstory 197121 12 Apr 18 01:36 c.txt
```



브랜치(branch)

SourceTree에서 불러오기

- main브랜치 모습. (브랜치명을 더블클릭하면 브랜치가 바뀐다. checkout명령과 같음)

The screenshot shows the SourceTree application window for a repository named 'funnyblog_carami'. The interface includes a menu bar at the top, a toolbar with icons for commit, pull, push, patch, branch, merge, stash, and tag, and a sidebar on the left with sections for 'WORKSPACE' (File Status, History, Search), 'Branches' (hello_feature, main), 'Tags', 'Remotes', and 'Stashes'. The main area displays a commit history table with columns for 'Graph', 'Description', 'Date', 'Author', and 'Commit ID'. The 'main' branch is selected and highlighted in blue. Below the table, the 'File Status' section shows the selected commit's details, including the commit ID, hash, author, date, and a list of files changed. The 'a.txt' file is highlighted, showing its content: 'hello world' and 'hi'.

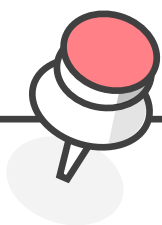
그래프	설명	날짜	작성자	커밋
	커밋하지 않은 변경사항	18 4 2021 15:49	*	*
	hello_feature hello.txt 파일	18 4 2021 15:44	urstory <urstory@gmail.com>	25abcaa
	main origin/main	18 4 2021 2:23	urstory <urstory@gmail.com>	90a1741
	a.txt, b.txt를 커밋합니다.	18 4 2021 2:05	urstory <urstory@gmail.com>	0796a96

커밋: 90a1741dd005ce9b4af6486872cec8a9a08235e [90a1741]
상위 항목: 0796a96a0d
작성자: urstory <urstory@gmail.com>
날짜: 2021년 4월 18일 일요일 오전 2:23:05
커밋한 사람: urstory

a파일에 hi를 추가

a.txt

```
1 1 ...hello world
2 2 + hi
```

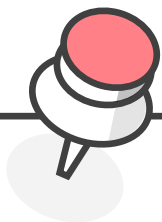


브랜치(branch)

git push origin hello_feature
- hello_feature 브랜치를 push한다.

MINGW64:/c/devel/funnyblog_carami

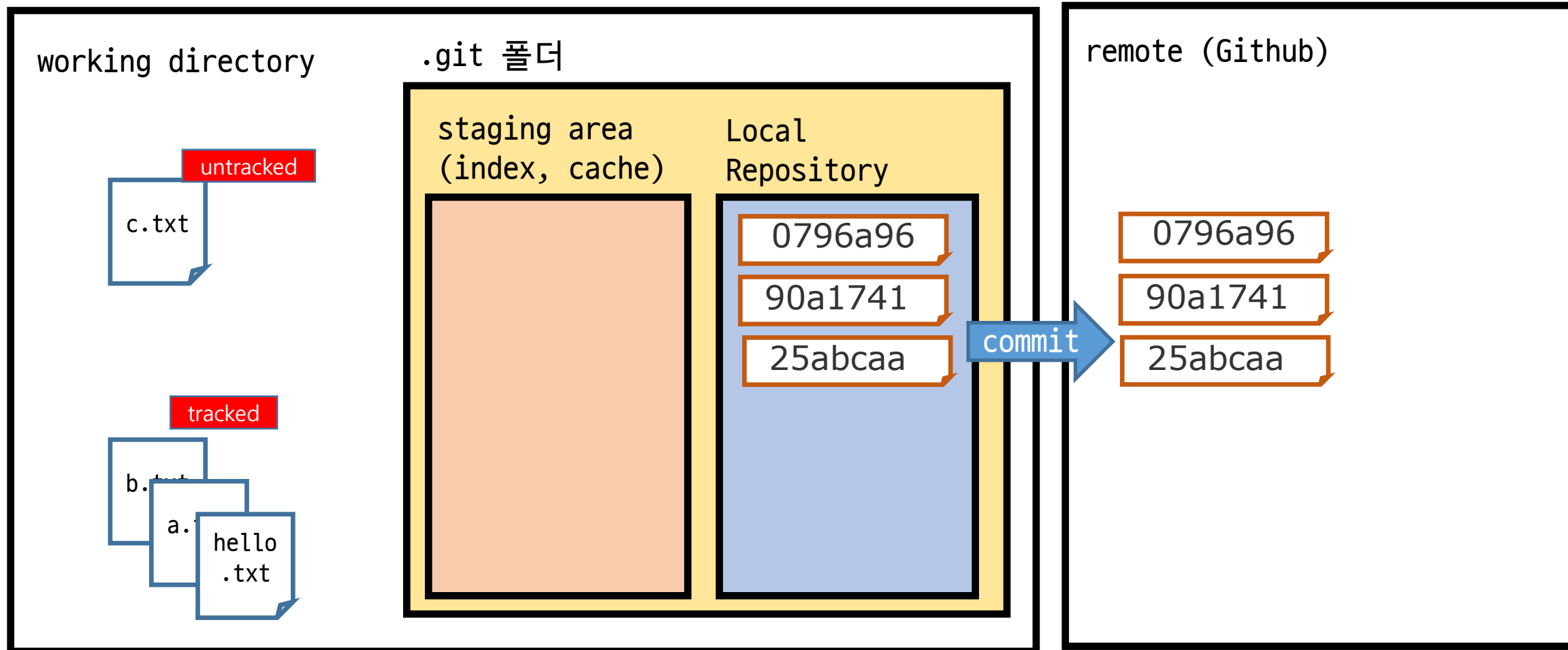
```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git push origin hello_feature
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 318 bytes | 318.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'hello_feature' on GitHub by visiting:
remote:   https://github.com/urstory/funnyblog/pull/new/hello_feature
remote:
To https://github.com/urstory/funnyblog.git
 * [new branch]      hello_feature -> hello_feature
```

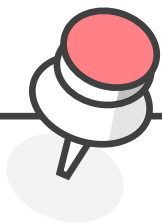



브랜치(branch)

```
echo hello > hello.txt  
git add hello.txt  
git commit -m "hello.txt 파일 생성"
```

프로젝트 폴더(c:\devel\funnyblog_carami)





브랜치(branch)

GitHub에서 생성된 브랜치를 확인한다.

- 생성된 브랜치로 바꾼 후, hello.txt파일이 존재하는지 확인한다.

github.com/urstory/funnyblog

urstory / funnyblog

hello_feature had recent pushes 1 minute ago

main 2 branches 0 tags

Switch branches/tags

Find or create a branch...

main default

hello_feature

View all branches

github.com/urstory/funnyblog/tree/hello_feature

urstory / funnyblog

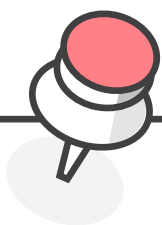
hello_feature had recent pushes 2 minutes ago

hello_feature 2 branches 0 tags

This branch is 1 commit ahead of main.

urstory hello.txt 파일 생성 25abcaa 12 minutes ago 3 commits

a.txt	a파일에 hi를 추가	14 hours ago
b.txt	a.txt, b.txt를 커밋합니다.	14 hours ago
hello.txt	hello.txt 파일 생성	12 minutes ago



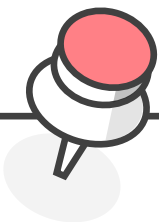
브랜치(branch)

git pull origin hello_feature

- hello_feature 브랜치를 pull한다.
- /c/devel/funnyblog_esther 에서 브랜치를 pull한다.

MINGW64:/c/devel/funnyblog_esther

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_esther (main)
$ git pull origin hello_feature
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 298 bytes | 29.00 KiB/s, done.
From https://github.com/urstory/funnyblog
 * branch          hello_feature -> FETCH_HEAD
 * [new branch]     hello_feature -> origin/hello_feature
Updating 90a1741..25abcaa
Fast-forward
 hello.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 hello.txt
```



브랜치(branch)

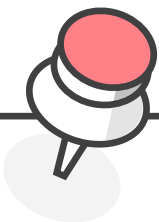
/c/devel/funnyblog_esther에서
명령

```
echo hi > hi.txt  
git add hi.txt  
git commit -m "hi.txt를 추가함"  
git push origin hello_feature
```

funnyblog_carami에는 해당 커밋
된 내용이 존재하지 않는다.

MINGW64:/c/devel/funnyblog_esther

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_esther (hello_feature)  
$ echo hi > hi.txt  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_esther (hello_feature)  
$ git add hi.txt  
warning: LF will be replaced by CRLF in hi.txt.  
The file will have its original line endings in your working directory  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_esther (hello_feature)  
$ git commit -m "hi.txt를 추가함"  
[hello_feature c1eb29d] hi.txt를 추가함  
1 file changed, 1 insertion(+)  
create mode 100644 hi.txt  
  
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_esther (hello_feature)  
$ git push origin hello_feature  
Enumerating objects: 4, done.  
Counting objects: 100% (4/4), done.  
Delta compression using up to 12 threads  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (3/3), 283 bytes | 283.00 KiB/s, done.  
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (1/1), completed with 1 local object.  
To https://github.com/urstory/funnyblog.git  
25abcaa..c1eb29d hello_feature -> hello_feature
```



브랜치(branch)

/c/devel/funnyblog_carami에서
명령

git pull origin hello_feature

수시로 git pull을 하는 것이 충돌을 발생시키지 않을 수 있다.

git pull을 하자 hi.txt가 추가되는 것을 볼 수 있다.

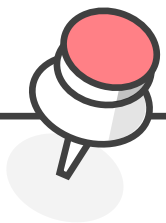
```
MINGW64:/c/devel/funnyblog_carami

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git checkout hello_feature
Switched to branch 'hello_feature'

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (hello_feature)
$ ls
a.txt  b.txt  c.txt  hello.txt

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (hello_feature)
$ git pull origin hello_feature
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), 263 bytes | 29.00 KiB/s, done.
From https://github.com/urstory/funnyblog
* branch                hello_feature -> FETCH_HEAD
   25abcaa..c1eb29d      hello_feature -> origin/hello_feature
Updating 25abcaa..c1eb29d
Fast-forward
 hi.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 hi.txt

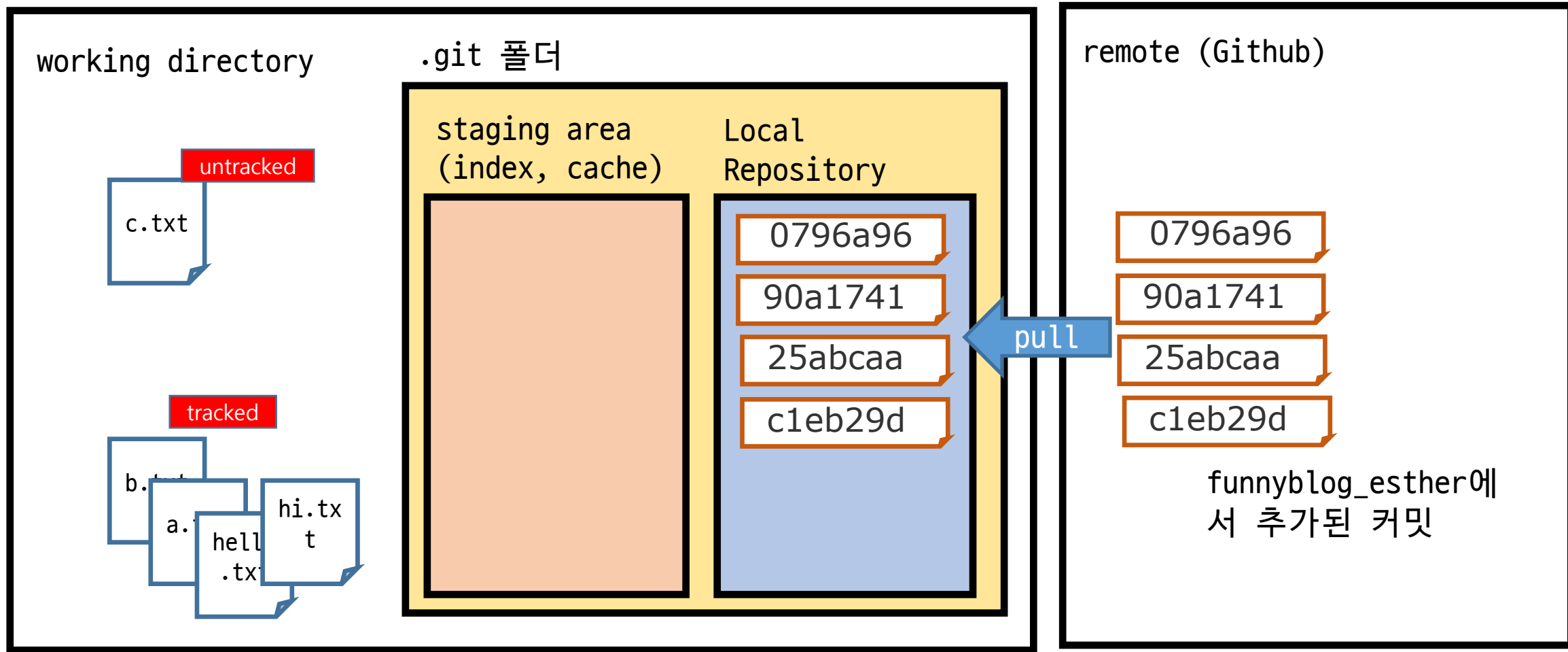
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (hello_feature)
$ ls
a.txt  b.txt  c.txt  hello.txt  hi.txt
```

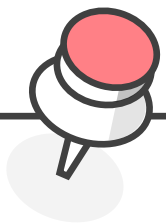


브랜치(branch)

```
git pull origin hello_feature
```

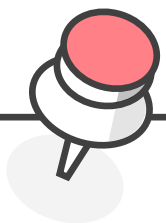
프로젝트 폴더(c:\devel\funnyblog_carami)





머지(merge)

- 브랜치를 다른 브랜치에 합치는 것을 머지(merge)라고 한다.
- hello_branch의 내용을 main에 합치려면, main브랜치안에서 hello_branch의 내용을 땡겨와서 합쳐야 한다.



머지(merge)

git merge hello_feature

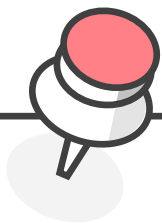
- 현재 브랜치에 hello_feature의 내용을 병합(merge)한다.
- 병합을 하게 되면 main브랜치의 내용이 변한다. 변한 것은 로컬에서만 변한 것이기 때문에 리모트로 적용하려면 푸시를 해야한다.

```
MINGW64:/c/devel/funnyblog_carami

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (hello_feature)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git merge hello_feature
Updating 90a1741..c1eb29d
Fast-forward
 hello.txt | 1 +
  hi.txt   | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 hello.txt
 create mode 100644 hi.txt

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ ls
a.txt b.txt c.txt hello.txt hi.txt
```

머지(merge)

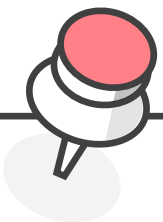
SourceTree에서 main브랜치를 보면 2개의 push되지 않은 파일이 있는 것을 알 수 있다.
해당 파일은 hello_feature에서 병합된 hello.txt, hi.txt파일이다.

The screenshot shows the SourceTree application window for a repository named 'funnyblog_carami'. The interface includes a menu bar, a toolbar with icons for commit, pull, push, fetch, branch, merge, stash, and tag, and a sidebar with sections for Workspace, History, Search, Branches, Tags, Remotes, and Stashes.

In the 'Branches' section, the 'main' branch is selected, and a red box highlights the '2' icon, indicating two uncommitted files. The 'Commit History' table shows the following entries:

그래프	설명	날짜	작성자	커밋
	커밋하지 않은 변경사항	18 4 2021 16:10	*	*
	hello_feature origin/hello_fe	18 4 2021 16:04	urstory <urstory@gmail.com>	c1eb29d
	hello.txt 파일 생성	18 4 2021 15:44	urstory <urstory@gmail.com>	25abcaa
	origin/main main a파일에	18 4 2021 2:23	urstory <urstory@gmail.com>	90a1741
	a.txt, b.txt를 커밋합니다.	18 4 2021 2:05	urstory <urstory@gmail.com>	0796a96

The bottom panel shows the file status for 'hi.txt', which is marked as 'added'. The commit details for the selected commit (c1eb29d) are displayed, including the commit message 'hi.txt를 추가함' and the author 'urstory'.



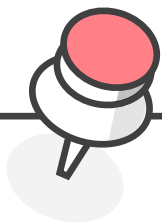
머지(merge)

git push origin main

- 병합된 내용을 서버에도 push한다.

MINGW64:/c/devel/funnyblog_carami

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/urstory/funnyblog.git
 90a1741..c1eb29d  main -> main
```

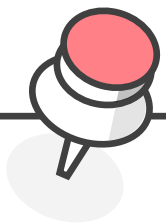


머지(merge)

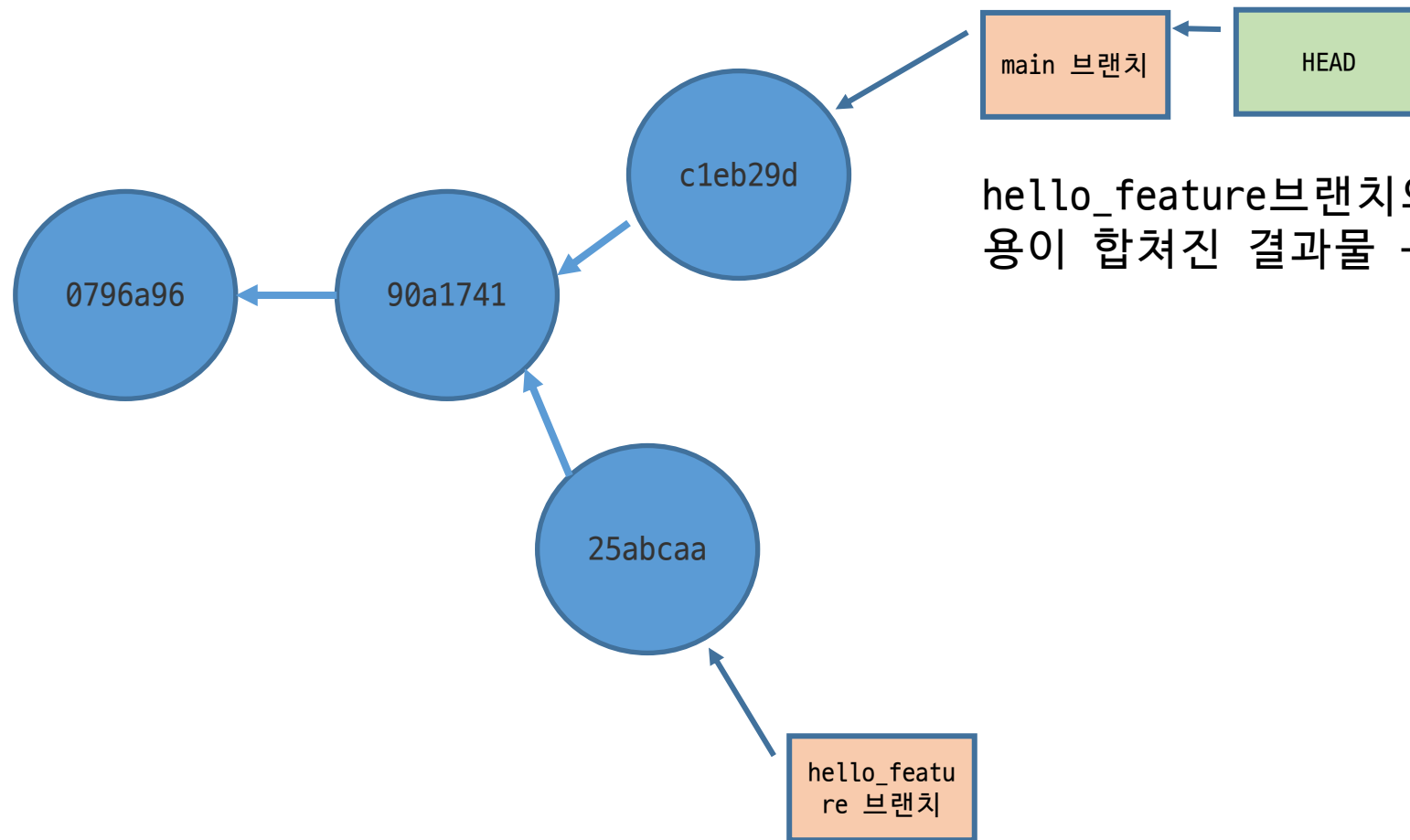
main 브랜치에 hello.txt, hi.txt가 포함되어 있는 것을 알 수 있다.

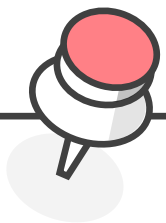
The screenshot shows the GitHub interface for the repository 'urstory/funnyblog'. The browser address bar displays 'github.com/urstory/funnyblog/tree/main'. The repository name 'urstory / funnyblog' is shown at the top. Below the repository name, there are tabs for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The 'main' branch is selected and highlighted with a red box. To the right of the branch name, it says '2 branches' and '0 tags'. There are buttons for 'Go to file', 'Add file', and 'Code'. Below this, a commit history table is visible:

Commit	Commit Message	Time
urstory	hi.txt를 추가함	c1eb29d 26 minutes ago 4 commits
a.txt	a파일에 hi를 추가	14 hours ago
b.txt	a.txt, b.txt를 커밋합니다.	14 hours ago
hello.txt	hello.txt 파일 생성	1 hour ago
hi.txt	hi.txt를 추가함	26 minutes ago



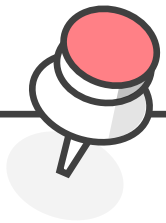
머지(merge)





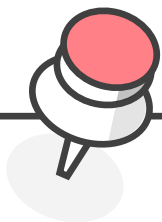
머지(merge)

- 풀 리퀘스트(Pull Request)를 이용하여 머지할 수도 있다. 이 부분은 뒤의 풀 리퀘스트 부분을 살펴보자.
- 팀원들과 소스코드를 리뷰하고, 문제가 없을 경우에만 머지한다.



충돌(conflict)

- 머지를 하다보면 충돌이 발생할 때가 있다.
- 여러 사람이 함께 작업을 하다 보면, 같은 파일을 수정하여 commit 할 경우가 있다.
- 실제 개발시에 충돌은 잘 발생하지 않는다. 같은 파일을 작성하는 경우가 많지 않고, 같은 파일을 작성하더라도, 서로 다른 부분을 작성하는 경우가 대부분이기 때문이다.
- 충돌이 잘 발생하지 않는다 하더라도, 충돌이 발생할 경우 문제를 해결할 수 있어야 한다.



충돌(conflict)

/c/devel/funnyblog_esther에서 명령

```
git checkout main  
git pull origin main
```

/c/devel/funnyblog_esther에서 명령

```
echo hi >> hello.txt  
git add hello.txt  
git commit -m "hello.txt를 수정함"  
git push -u origin main
```

/c/devel/funnyblog_carami에서 명령

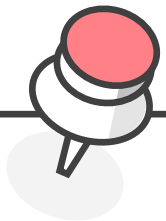
```
echo bye >> hello.txt  
git add hello.txt  
git commit -m "hello.txt를 수정함"  
git push -u origin main
```

```
urstory@DESKTOP-45M171R MINGW64 /c/devel/funnyblog_carami (main)  
$ git push -u origin main  
To https://github.com/urstory/funnyblog.git  
! [rejected]        main -> main (non-fast-forward)  
error: failed to push some refs to 'https://github.com/urstory/funnyblog.git'  
hint: Updates were rejected because the tip of your current branch is behind  
hint: its remote counterpart. Integrate the remote changes (e.g.  
hint: 'git pull ...') before pushing again.  
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

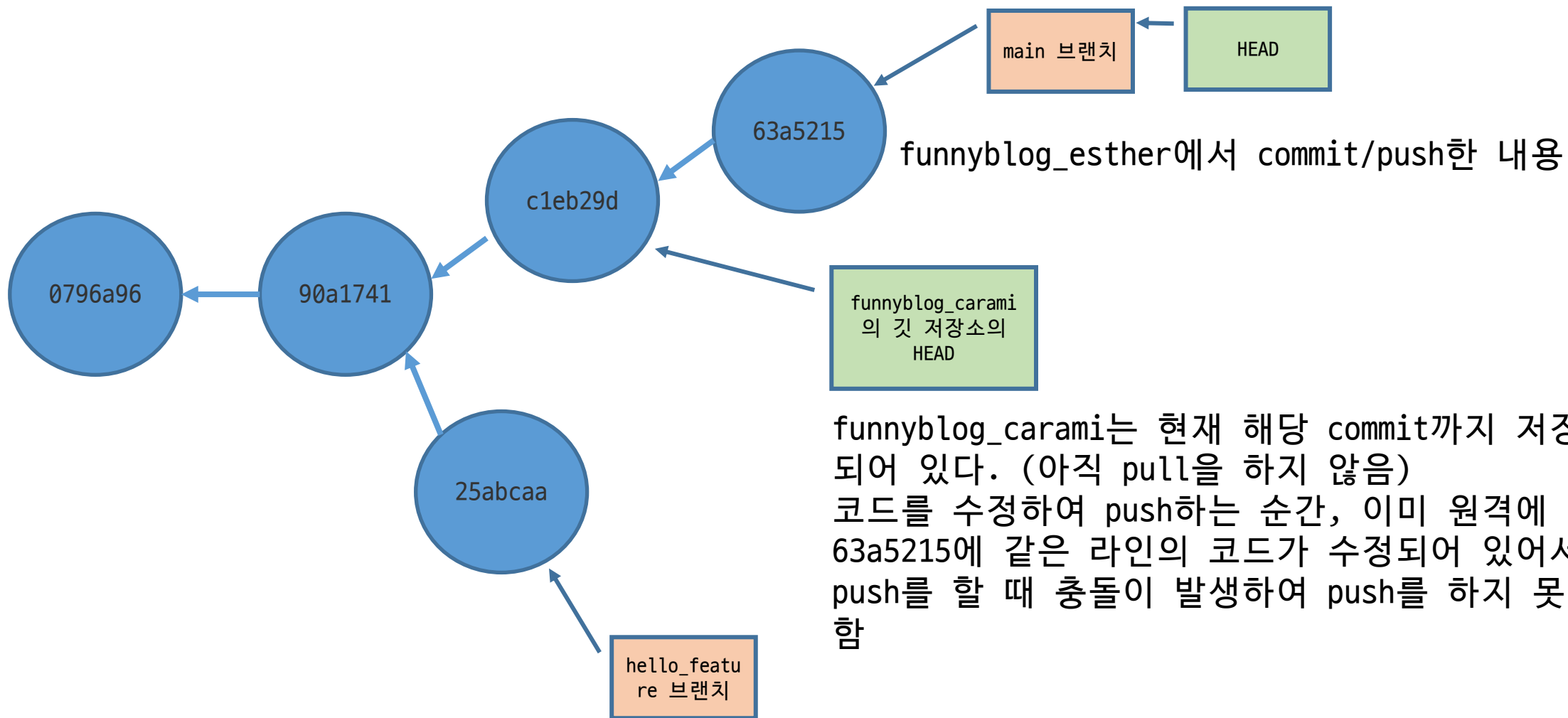
충돌 발생

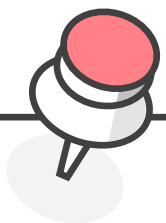


- 같은 라인이 다른 내용이라 충돌.



충돌(conflict)





충돌(conflict)

/c/devel/funnyblog_carami에서 명령

```
git pull origin main
```

hello.txt에서 충돌이 발생한 것을 알 수 있다.

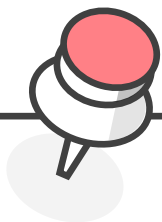
MINGW64:/c/devel/funnyblog_carami

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git pull origin main
From https://github.com/urstory/funnyblog
* branch      main      -> FETCH_HEAD
Auto-merging hello.txt
CONFLICT (content): Merge conflict in hello.txt
Automatic merge failed; fix conflicts and then commit the result.
```

MINGW64:/c/devel/funnyblog_carami

```
hello
<<<<<<< HEAD
bye
=====
hi
>>>>>>> 63a5215b781f4588e7ed53d735193e6c71c8c41e
```

hello.txt 파일의 내용을 보면, 현재 carami의 HEAD가 가리키는 부분은 bye라고 작성되어 있고, 이미 리모트에 commit/push된 내용에는 hi라고 작성되어 있다.
어떻게 맞는 코드인가?



충돌(conflict)

bye가 맞는지 hi가 맞는지는 코드를 작성한 사람과 토론하여 결정해야 한다. 만약 carami의 bye가 맞는 부분이라면 hello.txt에서 bye부분만 남기고 코드를 수정한다.

코드를 수정하였다면, 해당 내용을 commi하고 push한다.

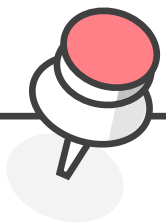
```
git add hello.txt
git commit -m "hello.txt의 코드 충돌을 해결 bye가 맞는 거였음"
git push -u origin main
```

```
$ cat hello.txt
hello
bye
```

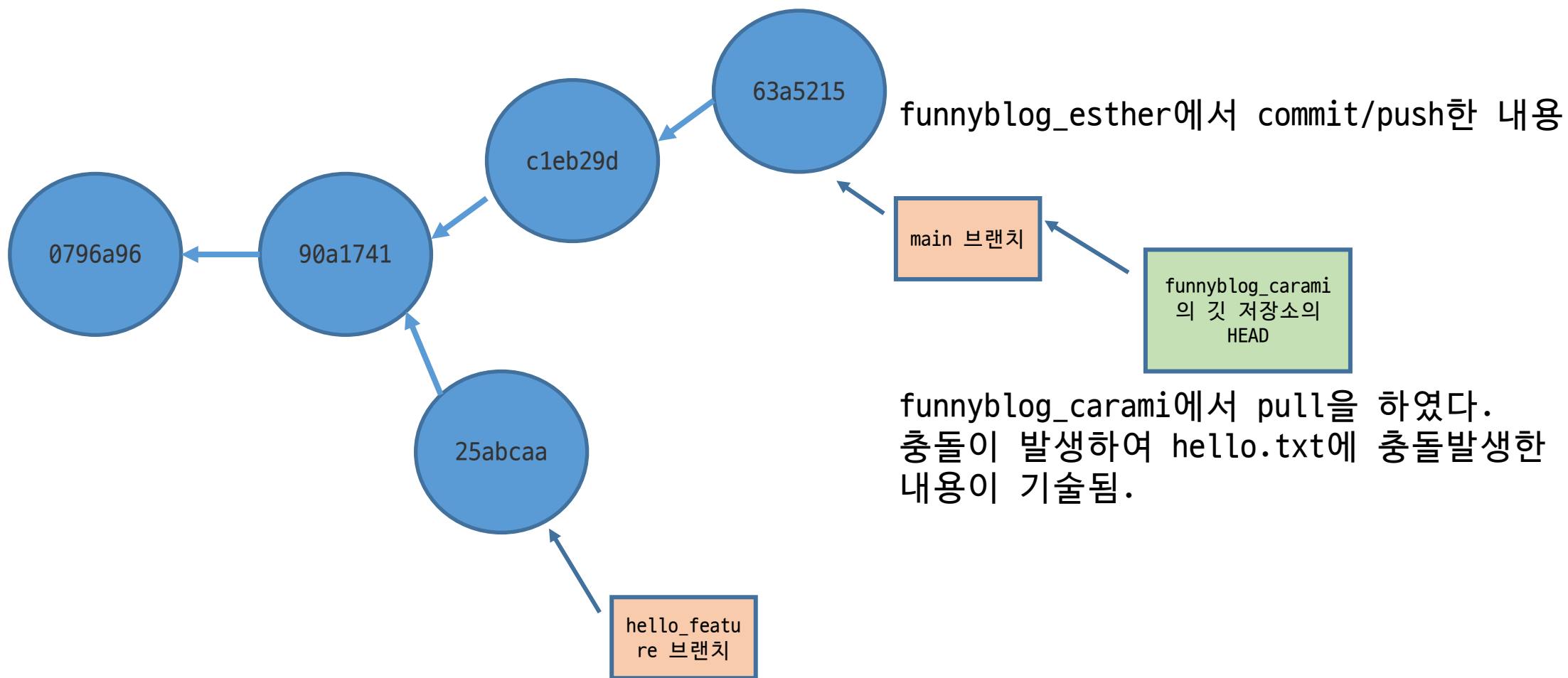
```
MINGW64:/c/devel/funnyblog_carami
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main|MERGING)
$ git add hello.txt

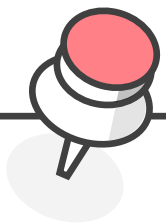
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main|MERGING)
$ git commit -m "hello.txt의 코드 충돌을 해결 bye가 맞는 거였음"
[main 172cb40] hello.txt의 코드 충돌을 해결 bye가 맞는 거였음

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git push -u origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 588 bytes | 588.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/urstory/funnyblog.git
   63a5215..172cb40  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

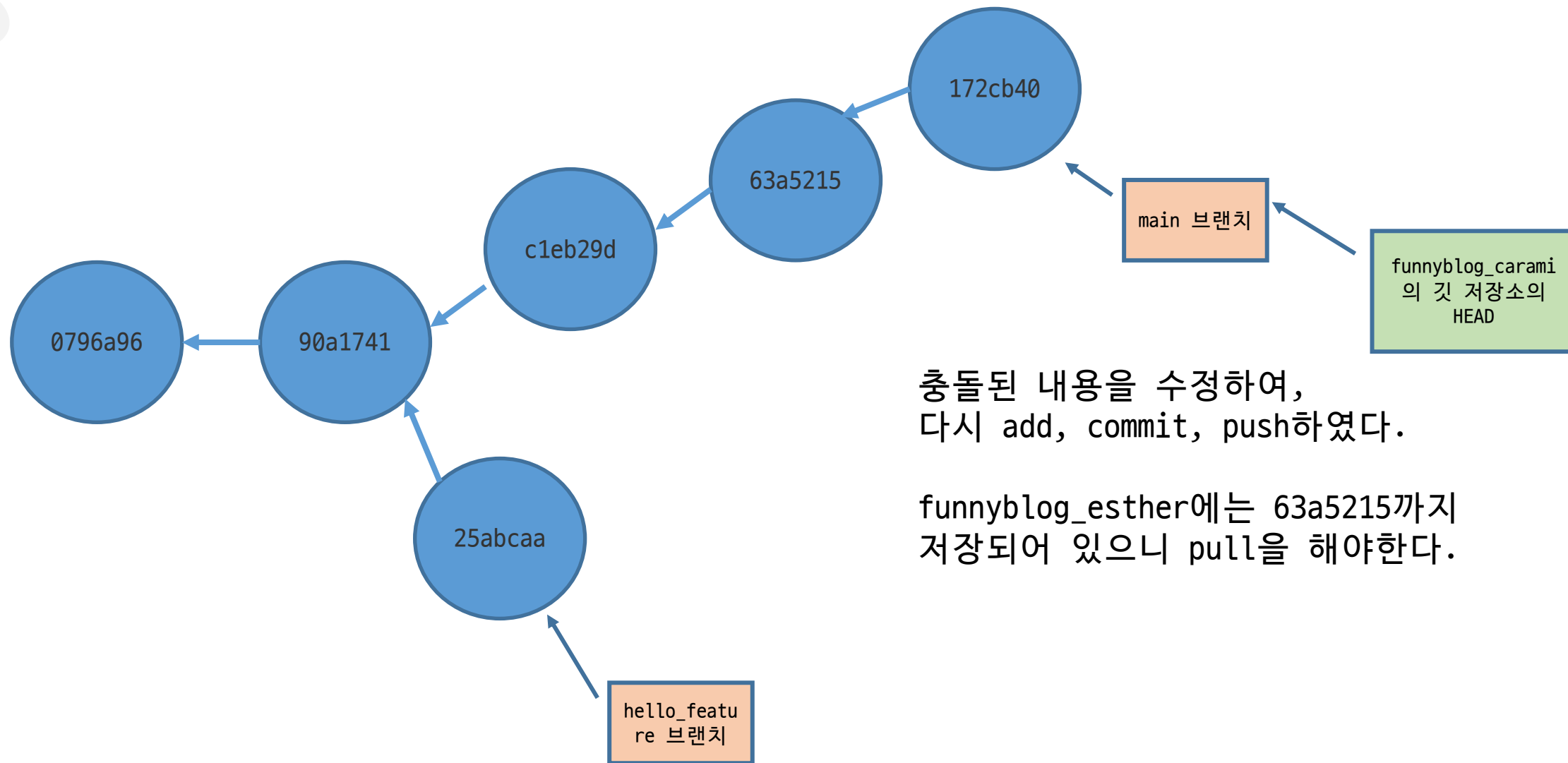


머지(merge)



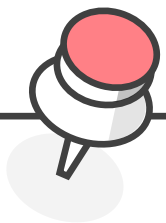


머지(merge)



충돌된 내용을 수정하여,
다시 add, commit, push하였다.

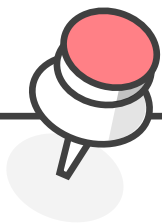
funnyblog_esther에는 63a5215까지
저장되어 있으니 pull을 해야한다.



충돌(conflict)

/c/devel/funnyblog_esther에서 명령
git pull origin main

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_esther (main)
$ git pull origin main
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 568 bytes | 51.00 KiB/s, done.
From https://github.com/urstory/funnyblog
* branch                main       -> FETCH_HEAD
   63a5215..172cb40      main       -> origin/main
Updating 63a5215..172cb40
Fast-forward
 hello.txt | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)
```



충돌(conflict)

SourceTree

funnyblog_carami

커밋 Pull Push 패치 브랜치 병합 스테시 페기 태그

WORKSPACE

파일 상태

History

Search

브랜치

hello_feature

main

태그

원격

스테시

모든 브랜치 원격 브랜치 날짜순으로 정렬

그래프	설명	날짜	작성자	커밋
	커밋하지 않은 변경사항	18 4 2021 17:18	*	*
	main origin/main hello.	18 4 2021 17:13	urstory <urstory@gmail.com>	172cb40
	hello.txt에 hi를 추가함	18 4 2021 16:48	urstory <urstory@gmail.com>	63a5215
	hello.txt를 수정함	18 4 2021 16:43	urstory <urstory@gmail.com>	0f4134a
	origin/hello_feature hello_featu	18 4 2021 16:04	urstory <urstory@gmail.com>	c1eb29d
	hello.txt 파일 생성	18 4 2021 15:44	urstory <urstory@gmail.com>	25abcaa
	a파일에 hi를 추가	18 4 2021 2:23	urstory <urstory@gmail.com>	90a1741
	a.txt, b.txt를 커밋합니다.	18 4 2021 2:05	urstory <urstory@gmail.com>	0796a96

파일 상태순 정렬

커밋: 172cb40017f884dcd89d0b435472ccfcb5feac2 [172cb40]

상위 항목: 0f4134ab3b, 63a5215b78

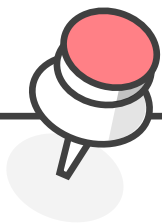
작성자: urstory <urstory@gmail.com>

날짜: 2021년 4월 18일 일요일 오후 5:13:29

커밋한 사람: urstory

hello.txt의 코드 충돌을 해결 bye가 맞는 거였음

Select a file to view the diff

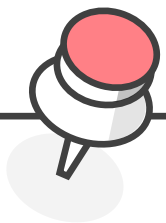


포크(fork)

GitHub에서 협업을 하는 사람은 Settings메뉴에서 Manage acces를 클릭한 후 collaborator를 추가하면 된다.

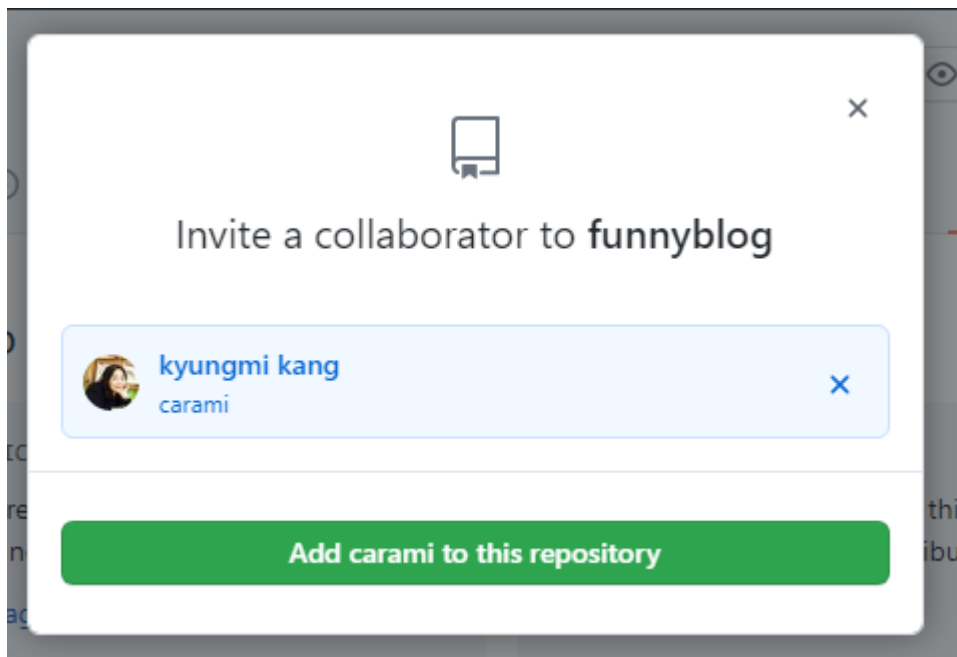
The screenshot shows the GitHub repository settings page for 'urstory/funnyblog'. The 'Settings' tab is selected and highlighted with a red circle labeled '1'. In the left sidebar, the 'Options' menu is expanded, and 'Manage access' is highlighted with a red circle labeled '2'. The main content area shows 'Who has access' with two sections: 'PUBLIC REPOSITORY' (This repository is public and visible to anyone. Manage) and 'DIRECT ACCESS' (0 collaborators have access to this repository. Only you can contribute to this repository.). Below this, the 'Manage access' section shows a message: 'You haven't invited any collaborators yet' with an 'Invite a collaborator' button highlighted with a red circle labeled '3'.

The screenshot shows the 'Invite a collaborator to funnyblog' dialog box. It features a search bar with the text 'carami'. Below the search bar, a list of users is displayed, each with a profile picture, name, and an 'Invite collaborator' button. The first user, 'kyungmi kang' (carami), is highlighted in blue. Other users listed include 'caramire', 'Caramiriel-SH', 'caramirez805', and 'CaraMia11'.



포크(fork)

멤버를 추가하면, 초대 메시지가 전달된다.



Manage access

Invite a collaborator

☐ Select all

Type ▾

Find a collaborator...

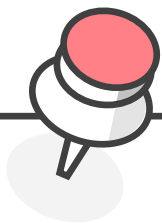


kyungmi kang

Awaiting carami's response

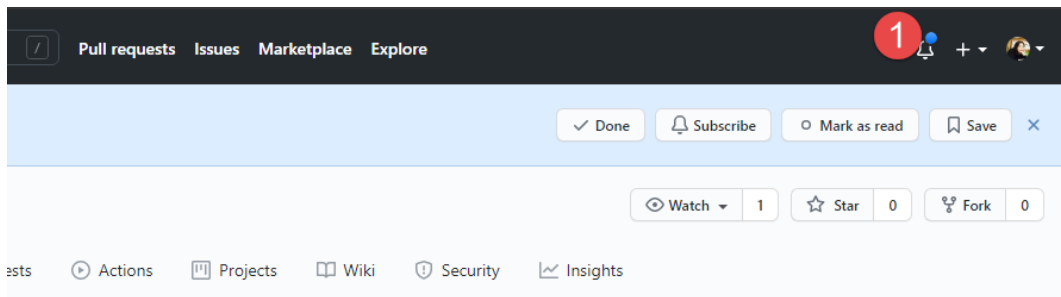
Pending Invite





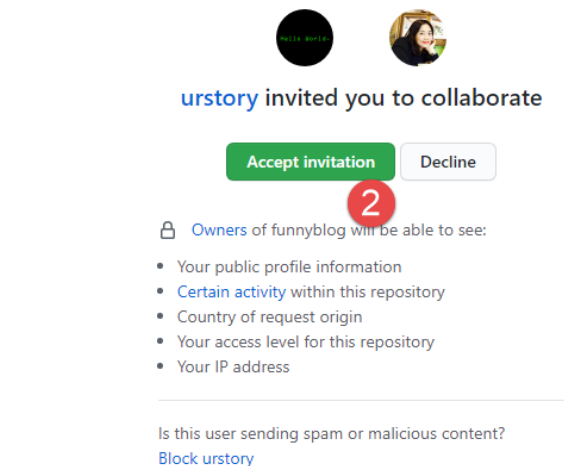
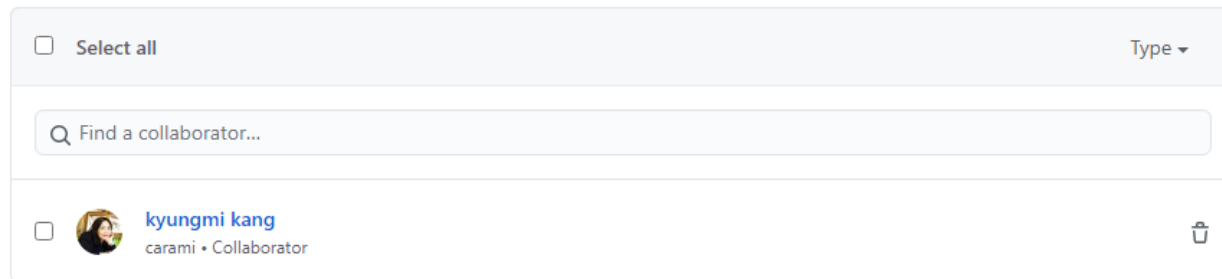
포크(fork)

초대 메시지를 받은 멤버는 알림에서 확인 후 "Accept Invitation"버튼을 클릭한다.

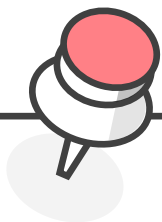


Manage access

Invite a collaborator



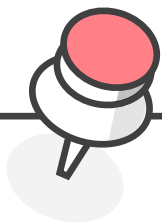
이렇게 추가된 사용자는 해당 레포지토리에 접근할 수 있다.



포크(fork)

- 내가 push할 수 있는 권한을 가진 프로젝트에 참여하고 싶은 경우가 있을 수 있다.
- 대표적인 경우가 오픈 소스에 참여하고 싶은 경우가 있다.
- 이 경우에는 GitHub에서 해당 오픈 소스 프로젝트를 간 후 fork를 수행한다.

The screenshot shows the GitHub repository page for 'carami/funnyblog'. The page layout includes a header with the GitHub logo, a search bar, and navigation links for Pulls, Issues, Marketplace, and Explore. Below the header, the repository name 'carami / funnyblog' is displayed, along with buttons for Watch (1), Star (0), and Fork (0). A red circle with the number '1' is placed over the Fork button. Below the repository name, there are tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights. The 'Code' tab is selected, showing a list of commits. The first commit is by 'carami' with the message 'a파일에 hi 추가' (Add hi to file a), dated '1 hour ago'. Below this, there are two files listed: 'a.txt' and 'b.txt', both with commit messages and dates. On the right side of the page, there is an 'About' section with the text 'No description, website, or topics provided.' and a 'Releases' section with the text 'No releases published'.

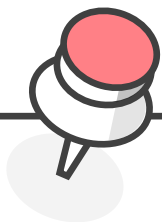


포크(fork)

- fork를 하면 사용자의 계정으로, 해당 레포지토리가 그대로 복사된다. 같은 이름의 레포지토리가 있을 경우 레포지토리명-<숫자> 형태로 복사가 된다.
- 이제 해당 프로젝트를 clone 하여 코드를 수정하고 add, commit, push를 하면 된다.
- 이렇게 하면 내 레포지토리의 내용이 변경되는 것이지 가지고 온 곳의 코드가 변경되는 것은 아니다.

The screenshot shows a web browser at the URL `github.com/urstory/funnyblog-1`. The repository is a fork of `carami/funnyblog`. The interface includes a search bar, navigation tabs for Pulls, Issues, Marketplace, and Explore, and a header with Watch (0), Star (0), and Fork (1) buttons. Below the header, there are tabs for Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content area shows the 'main' branch with a message 'This branch is even with carami:main.' and buttons for 'Go to file', 'Add file', and 'Code'. A commit history table is visible, showing two commits by 'carami' from 1 hour ago. The first commit, 'a파일에 hi 추가', added a file 'a.txt'. The second commit, 'a.txt,b.txt를 커밋합니다.', added a file 'b.txt'. On the right side, there are sections for 'About' (No description, website, or topics provided), 'Releases' (No releases published), and 'Packages' (No packages published).

Commit	Author	Message	Time
a.txt	carami	a파일에 hi 추가	1 hour ago
b.txt	carami	a.txt,b.txt를 커밋합니다.	1 hour ago



포크(fork)

// GitHub으로 부터 레포지토리를 복제한다. funnyblog-1폴더가 자동으로 생성된다.

```
git clone https://github.com/urstory/funnyblog-1.git
```

// 생성된 폴더로 이동

```
cd funnyblog-1
```

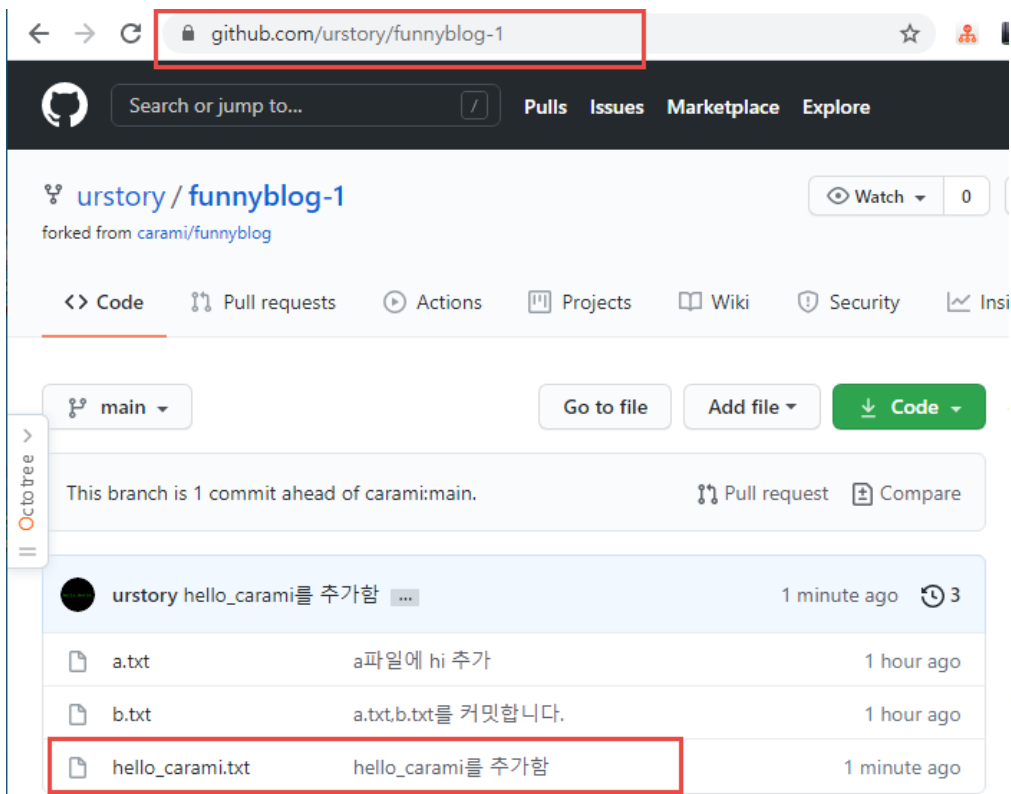
// 새로운 파일을 작성. add, commit, push진행함

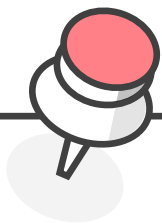
```
echo hello_carami > hello_carami.txt
```

```
git add hello_carami.txt
```

```
git commit -m "hello_carami를 추가함"
```

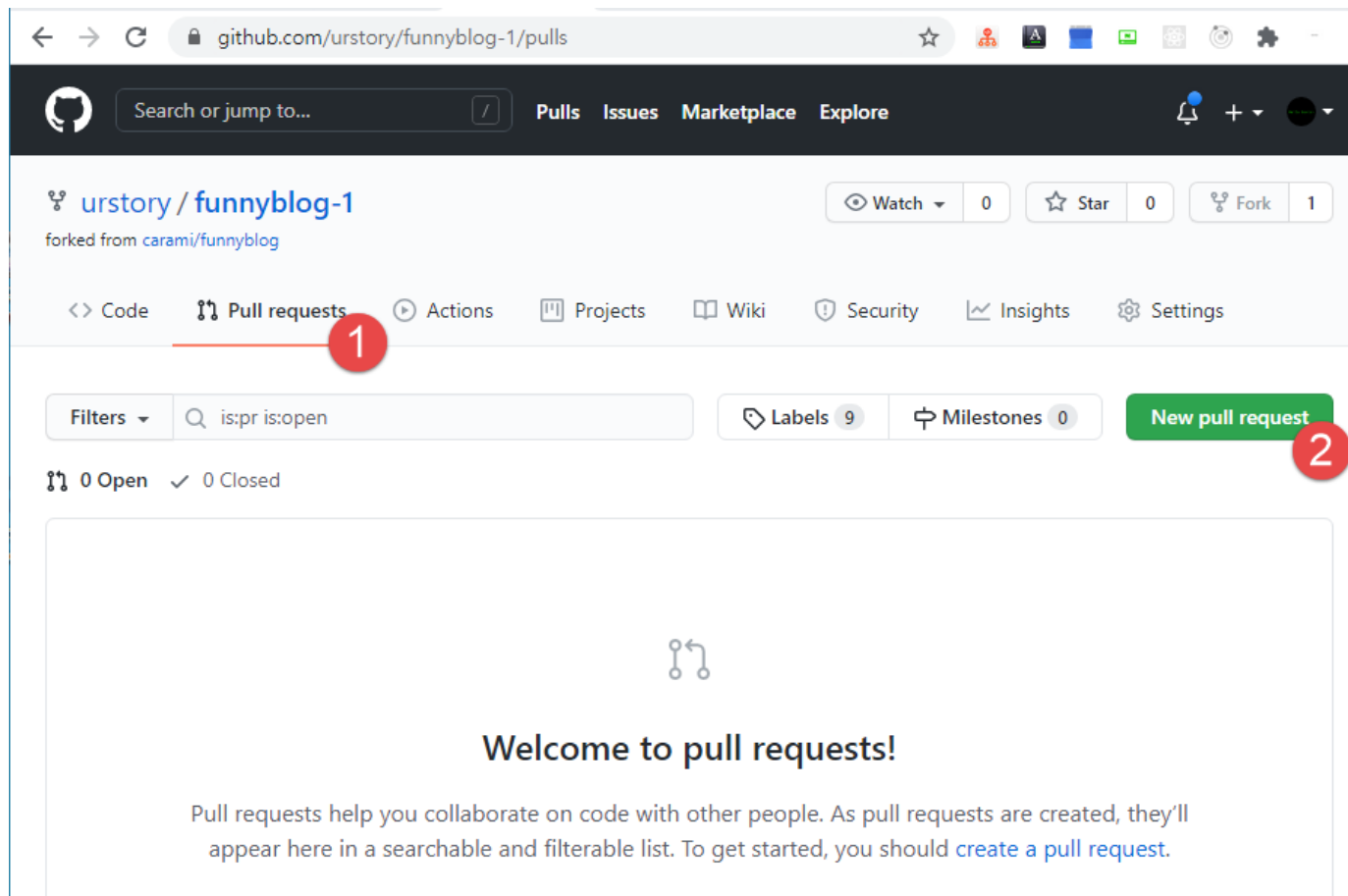
```
git push -u origin main
```

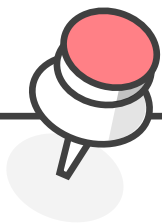




풀 리퀘스트(Pull requests)

fork된 레포지토리의 Pull requests메뉴를 선택한 후 "New pull request"버튼을 클릭한다.





풀 리퀘스트(Pull requests)

우측(urstory/funnyblog-1)의 main브랜치의 내용을 carami/funnyblog의 main브랜치에 반영해 달라고 풀 리퀘스트를 생성한다.

github.com/carami/funnyblog/compare/main...urstory:main

Search or jump to... Pulls Issues Marketplace Explore

carami / funnyblog Watch 1 Star 0 Fork 1

<> Code Issues Pull requests Actions Projects Wiki Security Insights

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base repository: carami/funnyblog base: main ← head repository: urstory/funnyblog-1 compare: main

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

1 commit 1 file changed 0 comments 1 contributor

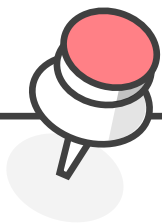
Commits on Apr 18, 2021

hello_carami를 추가함 ed11b2b

Showing 1 changed file with 1 addition and 0 deletions. Unified Split

1 hello_carami.txt

```
... @@ -0,0 +1 @@
1 + hello carami
```



풀 리퀘스트(Pull requests)

해당 풀 리퀘스트에 대한 제목과 내용을 적은 후 "Create pull request" 버튼을 클릭한다.

github.com/carami/funnyblog/compare/main...urstory:main

Search or jump to... Pulls Issues Marketplace Explore

carami / funnyblog Watch 1 Star

Code Issues Pull requests Actions Projects Wiki Security Insights

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](#).

base repository: carami/funnyblog base: main head repository: urstory/funnyblog-1 compare: main

✓ Able to merge. These branches can be automatically merged.

hello_carami를 추가함

Write Preview

H B I

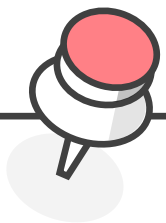
hello_carami를 추가하는게 좋을 것 같아서 추가하였어요.
리뷰해주시기 바랍니다.

1

Attach files by dragging & dropping, selecting or pasting them.


☒ Allow edits by maintainers ?

Create pull request 2






풀 리퀘스트(Pull requests)

풀 리퀘스트가 생성되면, 원 저작자에게 풀리퀘스트 알림이 간다.



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

1



 Inbox 9

 Saved

 Done

All Unread

Filter notifications

Group by: Date

☐ Select all



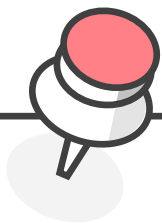
☐ carami/funnyblog #1
hello_carami를 추가함

2

subscribed



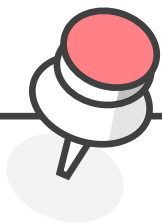
now



풀 리퀘스트(Pull requests)

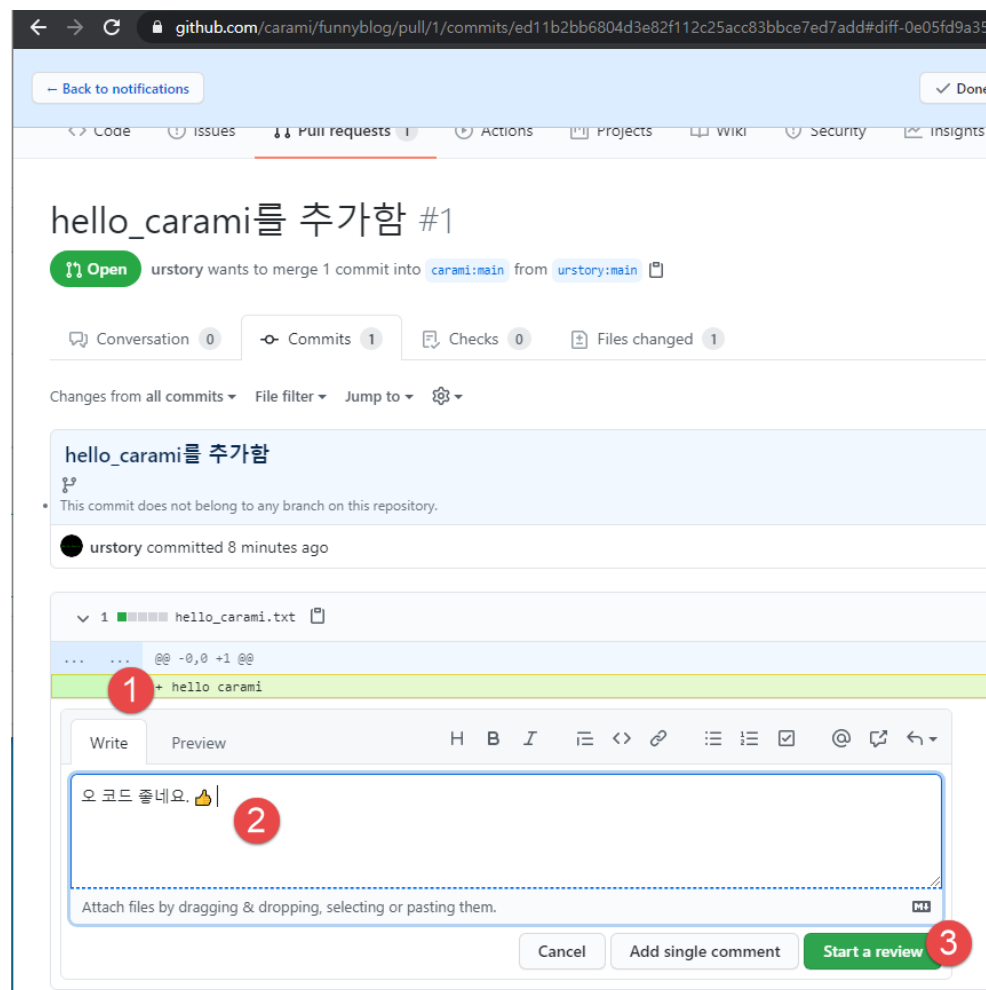
알림에 있는 제목을 클릭하면 풀리퀘스트 내용이 보여진다. 지금은 커밋이 한개지만, 여러개의 커밋이 보여질 수 있다. 해당 커밋 메시지를 클릭하여 코드를 리뷰한다.

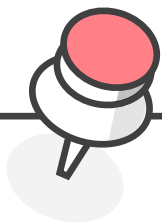
The screenshot shows a GitHub Pull Request page. The browser address bar displays 'github.com/carami/funnyblog/pull/1'. The page title is 'hello_carami를 추가함 #1'. Below the title, it says 'urstory wants to merge 1 commit into carami:main from urstory:main'. The 'Pull requests' tab is selected in the navigation bar. The main content area shows a comment from 'urstory' stating 'hello_carami를 추가하는게 좋을 것 같아서 추가하였어요. 리뷰해주시기 바랍니다.' Below the comment, there is a commit entry for 'hello_carami를 추가함' with a commit hash 'ed11b2b'. At the bottom, a green box contains status messages: 'Continuous integration has not been set up' and 'This branch has no conflicts with the base branch'. A red circle with the number '2' highlights a green button labeled 'Merge pull request'.



풀 리퀘스트(Pull requests)

커밋 메시지를 누르면 해당 커밋과 관련된 변경된 코드가 보여진다. 코드 앞의 + 를 클릭한 후 리뷰를 적는다.





풀 리퀘스트(Pull requests)

리뷰가 끝났다면 "Finish your review"를 클릭한다.

The screenshot shows a GitHub Pull Request page for the repository 'carami/funnyblog'. The pull request is titled 'hello_carami를 추가함 #1' and is in the 'Open' state. The user 'urstory' wants to merge 1 commit into 'carami:main' from 'urstory:main'. The pull request has 1 commit, 0 checks, and 1 file changed. The commit message is 'hello_carami를 추가함'. The commit does not belong to any branch on this repository. The commit was made 8 minutes ago. The diff shows a new file 'hello_carami.txt' with the content 'hello carami'. The user 'carami' (Owner) has a 'Pending' review. The review comment says '오 코드 좋네요. 👍'. There is a 'Finish your review' button with a red circle containing the number 1.

github.com/carami/funnyblog/pull/1/commits/ed11b2bb6804d3e82f112c25acc83bbce7ed7add#diff-0e05fd9a3599ed42837798e0cb3e6e7b361f546... 시크릿 모드

Back to notifications Done Unsubscribe Mark as unread Save

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

hello_carami를 추가함 #1 Edit Open with

Open urstory wants to merge 1 commit into carami:main from urstory:main Status: Open

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 -0

Changes from all commits File filter Jump to

hello_carami를 추가함

This commit does not belong to any branch on this repository.

urstory committed 8 minutes ago commit ed11b2bb6804d3e82f112c25acc83bbce7ed7add

1 hello_carami.txt

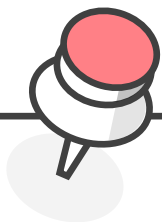
@@ -0,0 +1 @@

1 + hello carami

carami Owner Pending

오 코드 좋네요. 👍

Reply...

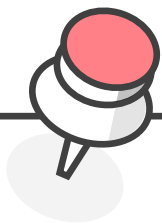


풀 리퀘스트(Pull requests)

Comment : 간단한 피드백 제출

Approve : 코드에 대한 의문점이 없다면 승인 .

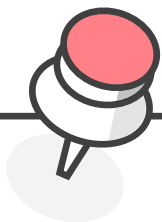
Request changes : 해당 코드에 문제가 있다고 판단되며 코드를 반드시 수정 요구 중에 하나를 선택 한 후 "Submit review" 버튼을 클릭한다.



풀 리퀘스트(Pull requests)

"풀 리퀘스트에 대한 리뷰를 한 후 최소 2명의 approval이 되어야 Merge를 하겠다." 등의 원칙을 정하는 것이 좋다. "Merge pull request"를 클릭한다.

The screenshot shows a GitHub pull request interface. At the top, the browser address bar displays 'github.com/carami/funnyblog/pull/1#pullrequestreview-638307063'. Below the address bar, there are navigation links: 'Back to notifications', 'Done', 'Unsubscribe', and 'Mark as unread'. The main content area shows a pull request titled 'hello_carami를 추가함 #1' (hello_carami를 추가함 #1) with the description 'urstory wants to merge 1 commit into carami:main from urstory:main'. The commit details show a file 'hello_carami.txt' with a diff: '... @@ -0,0 +1 @@' and a single line addition: '1 + hello carami'. Below the diff, there is a comment from 'carami now' (Owner) saying '오 코드 좋네요.' (Oh, the code is good.) with a thumbs up emoji. There is a 'Reply...' input field and a 'Resolve conversation' button. On the right side, there are sections for 'Projects' (None yet), 'Milestone' (No milestone), 'Linked issues' (Successfully merging this pull request will close these issues), and 'Notifications' (Unsubscribe). At the bottom, there is a green box with a checkmark icon and the text 'Changes approved' (1 approving review, Learn more). Below this, there is a section for '1 approval' with a dropdown arrow. Further down, there is a warning icon and the text 'Continuous integration has not been set up' (GitHub Actions and several other apps can be used to automatically catch bugs and enforce style). Below that, there is a checkmark icon and the text 'This branch has no conflicts with the base branch' (Merging can be performed automatically). At the very bottom, there is a green button labeled 'Merge pull request' with a red circle containing the number '1' next to it. To the right of the button, there is a link: 'You can also open this in GitHub Desktop or view command line instructions.'



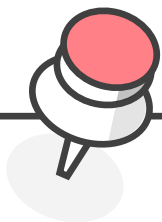
풀 리퀘스트(Pull requests)

hello_carami.txt 파일이 추가된 것을 확인할 수 있다.
이런 식으로 오픈 소스나 권한이 없는 레포지토리에 기여할 수 있다.

The screenshot shows the GitHub interface for the repository 'carami / funnyblog'. At the top, there's a navigation bar with 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below this, a blue bar contains a 'Back to notifications' link and 'Done' and 'Unsubscribed' buttons. The repository name 'carami / funnyblog' is displayed, followed by a 'Unwatch' button. A horizontal menu includes 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. A yellow notification bar states 'hello_feature had recent pushes 44 minutes ago' with a 'Compare & pull request' button. Below this, repository statistics show 'main' branch, '2 branches', and '0 tags', along with 'Go to file', 'Add file', and 'Code' buttons. The main content area shows a merge commit by 'carami' titled 'Merge pull request #1 from urstory/main', with commit hash 'a85fc7c' and '11 seconds ago', and '4 commits'. A table of file changes is shown below:

a.txt	a파일에 hi 추가	1 hour ago
b.txt	a.txt,b.txt를 커밋합니다.	1 hour ago
hello_carami.txt	hello_carami를 추가함	10 minutes ago

The row for 'hello_carami.txt' is highlighted with a red border.



되돌리기(*amend*)

코드를 작성하다보면 되돌리기(Undo)를 하고 싶은 경우가 있다.
예를 들어 커밋을 한 이후에 빼먹은 파일을 추가하고 싶거나, 메시지를 잘못 적었을 경우 다시 커밋하고 싶을 경우 파일 수정 작업을 한 후 --amend 옵션을 넣어 커밋을 재작성을 할 수 있다. 즉 기존 커밋을 덮어쓰게 된다.

```
echo aaa > a.txt
```

```
echo bbb > b.txt
```

위와 같이 기존 파일을 새로운 내용으로 수정하였다.

```
urstory@DESKTOP-45M171R MINGW64 /c/devel/funnyblog_carami (main)
$ echo aaa > a.txt

urstory@DESKTOP-45M171R MINGW64 /c/devel/funnyblog_carami (main)
$ echo bbb > b.txt

urstory@DESKTOP-45M171R MINGW64 /c/devel/funnyblog_carami (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   a.txt
        modified:   b.txt

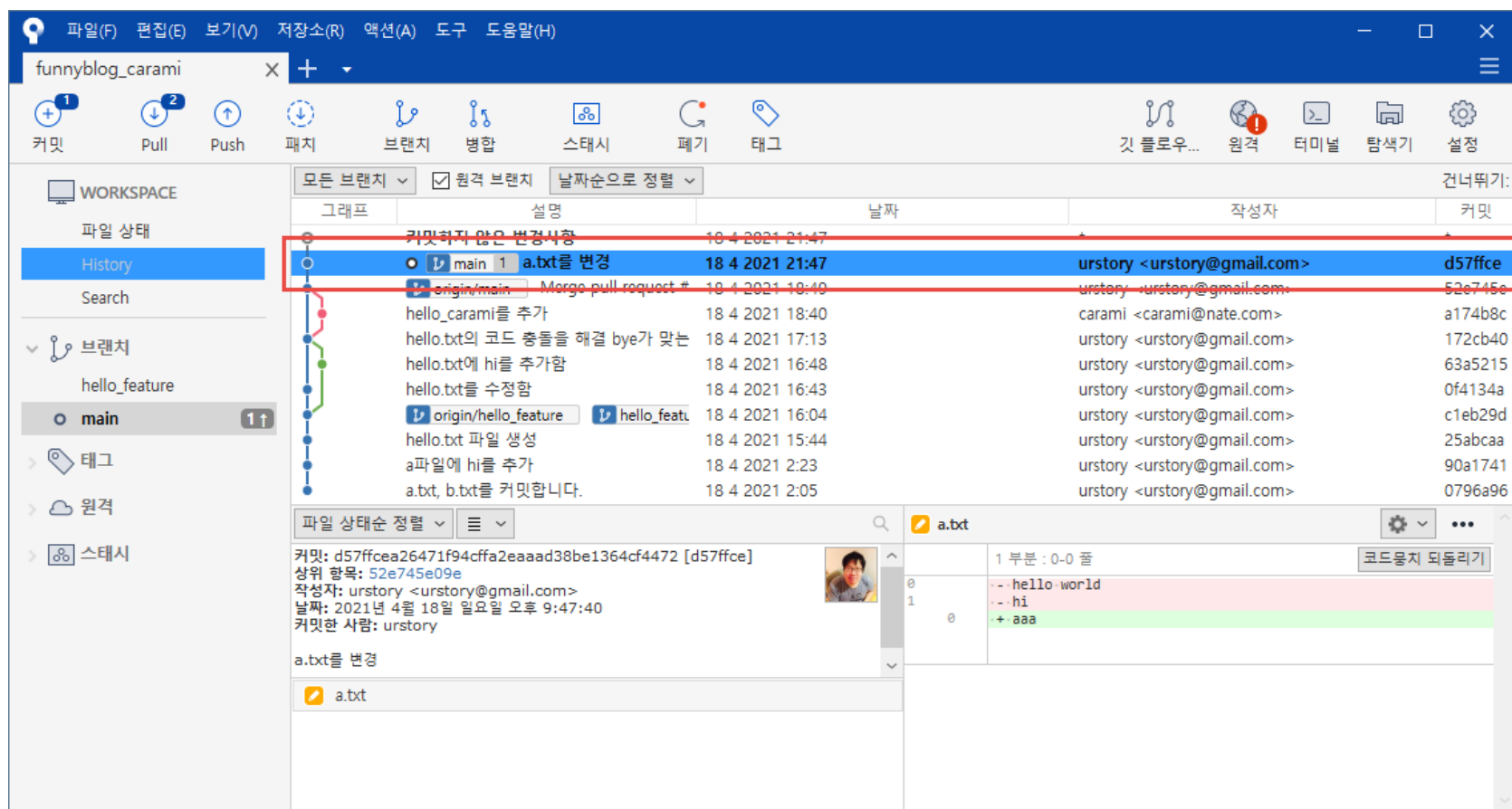
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        c.txt

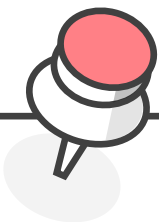
no changes added to commit (use "git add" and/or "git commit -a")
```



```
git commit -m "a.txt를 변경"
```

a.txt파일만 스테이징에 추가한 후 커밋을 하였다.





되돌리기(*amend*)

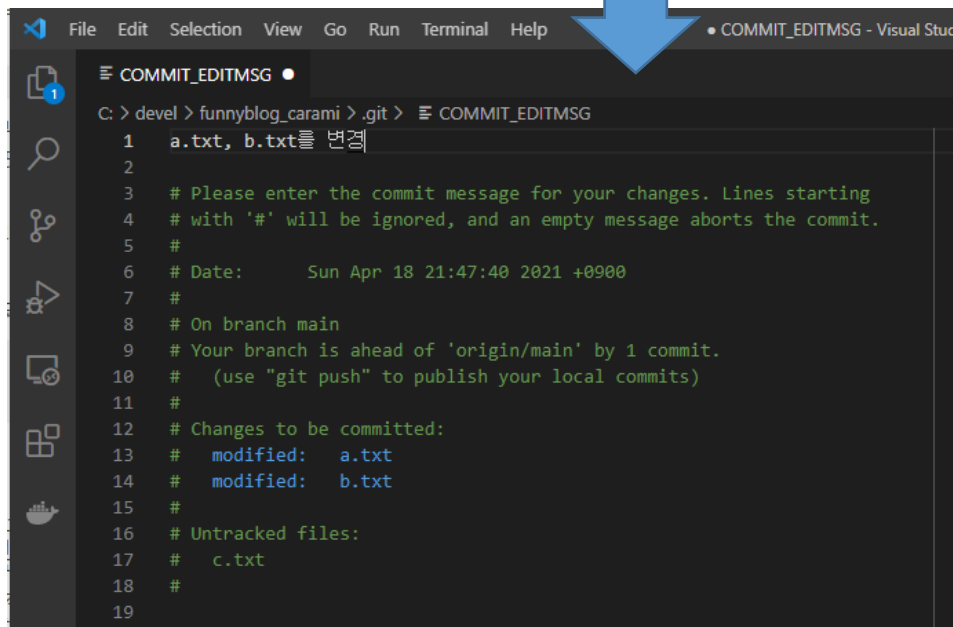

```
git add b.txt
```

```
git commit --amend
```

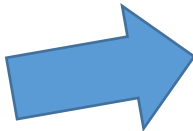
b.txt 파일을 추가한 후 되돌리기를 하였다.

위와 같이 명령을 수행하면, 등록된 에디터가 열리면서 commit 메시지를 수정할 수 있다. 수정한 후 저장한다.

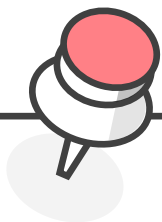
```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git commit --amend
hint: Waiting for your editor to close the file...
```



```
COMMIT_EDITMSG
C: > devel > funnyblog_carami > .git > COMMIT_EDITMSG
1 a.txt, b.txt를 변경
2
3 # Please enter the commit message for your changes. Lines starting
4 # with '#' will be ignored, and an empty message aborts the commit.
5 #
6 # Date:      Sun Apr 18 21:47:40 2021 +0900
7 #
8 # On branch main
9 # Your branch is ahead of 'origin/main' by 1 commit.
10 # (use "git push" to publish your local commits)
11 #
12 # Changes to be committed:
13 #   modified:   a.txt
14 #   modified:   b.txt
15 #
16 # Untracked files:
17 #   c.txt
18 #
19
```



```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git commit --amend
[main d4a57c4] a.txt, b.txt를 변경
Date: Sun Apr 18 21:47:40 2021 +0900
2 files changed, 2 insertions(+), 3 deletions(-)
```



되돌리기(amend)

기존 commit된 내용이 사라지고, 새로운 commit객체로 바뀐 것을 확인할 수 있다.
두번째 커밋이 기존 커밋을 덮어써버린다.

The screenshot shows the Git GUI interface for a repository named 'funnyblog_carami'. The left sidebar shows the 'History' tab selected. The main area displays a list of commits. The latest commit, 'a.txt, b.txt를 변경' by 'urstory <urstory@gmail.com>', is highlighted in blue and has its commit hash 'd4a57c4' shown. Below the commit list, the details of the selected commit are shown, including the commit message, author, and date. The file 'a.txt' is selected, and the diff view shows the changes made in the commit.

그래프	설명	날짜	작성자	커밋
○	커밋하지 않은 변경사항	18 4 2021 21:51	*	*
○	main 1 a.txt, b.txt를 변경	18 4 2021 21:48	urstory <urstory@gmail.com>	d4a57c4
○	origin/main Merge pull request #	18 4 2021 18:49	urstory <urstory@gmail.com>	52e745e
○	hello_carami를 추가	18 4 2021 18:40	carami <carami@nate.com>	a174b8c
○	hello.txt의 코드 충돌을 해결 bye가 맞는	18 4 2021 17:13	urstory <urstory@gmail.com>	172cb40
○	hello.txt에 hi를 추가함	18 4 2021 16:48	urstory <urstory@gmail.com>	63a5215
○	hello.txt를 수정함	18 4 2021 16:43	urstory <urstory@gmail.com>	0f4134a
○	origin/hello_feature hello_featu	18 4 2021 16:04	urstory <urstory@gmail.com>	c1eb29d
○	hello.txt 파일 생성	18 4 2021 15:44	urstory <urstory@gmail.com>	25abcaa
○	a파일에 hi를 추가	18 4 2021 2:23	urstory <urstory@gmail.com>	90a1741
○	a.txt, b.txt를 커밋합니다.	18 4 2021 2:05	urstory <urstory@gmail.com>	0796a96

파일 상태순 정렬

커밋: d4a57c47b415170938b0835f4e53e0e051b90dca [d4a57c4]
상위 항목: 52e745e09e
작성자: urstory <urstory@gmail.com>
날짜: 2021년 4월 18일 일요일 오후 9:47:40
커밋한 사람: urstory
커밋 날짜: 2021년 4월 18일 일요일 오후 9:48:56

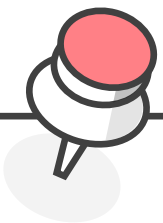
a.txt, b.txt를 변경

a.txt
b.txt

1 부분: 0-0 줄

코드돌리기

```
0 - hello world  
1 - hi  
0 + aaa
```



파일 상태를 Unstage로 변경하기

git add 명령으로 실수로 Staging Area에 추가를 하였다면 어떻게 해야할까?

git reset HEAD <파일명>

위의 명령으로 Unstage상태로 변경할 수 있다.

```
git add c.txt
```

```
git status
```

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git add c.txt
warning: LF will be replaced by CRLF in c.txt.
The file will have its original line endings in your working directory

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   c.txt
```

```
git reset HEAD c.txt
```

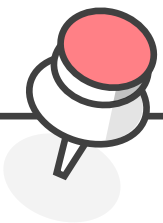
```
git status
```

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git reset HEAD c.txt

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    c.txt

nothing added to commit but untracked files present (use "git add" to track)
```



수정한 파일을 되돌리기

commit을 한 이후에 파일을 수정하였다. 수정된 파일을 원래대로 원복하려면?

```
echo a10 > a.txt
cat a.txt
git status
```

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ echo a10 > a.txt

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ cat a.txt
a10

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   a.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        c.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

```
git checkout -- a.txt
git status
cat a.txt
```

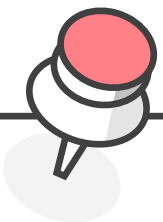
```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git checkout -- a.txt

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        c.txt

nothing added to commit but untracked files present (use "git add" to track)

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (main)
$ cat a.txt
aaa
```



태그(tag)

보통 릴리즈 할 때 사용한다. 예를 들어서 v1.0등을 지정하고 싶은 경우 사용한다. 태그를 붙여 놓으면, 나중에 해당 태그가 붙을때 당시의 소스코드를 확인할 때 편리하다. 커밋은 커밋된 파일들에 대한 정보만 가지고 있다.

```
git tag
```

태그 목록을 조회한다.

```
git tag -a v1.1 -m "설명"
```

-a 옵션을 주면 Annotated태그를 붙일 수 있다.

```
git show v1.1
```

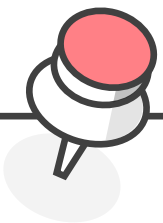
태그에 대한 정보를 확인할 수 있다.

```
git tag v1.2w
```

Lightweight태그를 작성할 땐 옵션을 주지 않는다. 파일에 커밋 체크섬을 저장한다.

```
git show v1.2w
```

별도의 태그 정보를 확인할 수 없다. 커밋정보만 확인 가능하다.



태그(tag)

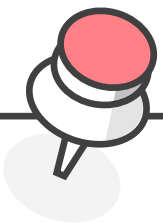
태그를 체크아웃할 수 있다.

```
git checkout <태그이름>
```

태그이름에 해당하는 내용을 체크아웃한다. 이때는 detached HEAD상태라고 하여, 여기에서 커밋을 하게 되면 해당 커밋된 결과에 접근할 수 있는 방법이 없게 된다.

```
git checkout -b <브랜치명> <태그이름>
```

태그이름에 해당하는 내용을 체크아웃하는데, 이때 새로운 브랜치로 생성하여 체크아웃한다.



스태시(stash)

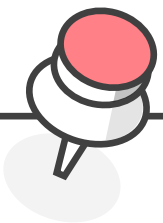
브랜치를 만들고 작업을 열심히 하고 있었는데, 브랜치를 변경할 필요가 있을 경우가 있다. 이때 완료되지 않은 작업을 commit하지 않고 브랜치를 바꾸기 위해 스택에 잠깐 작업을 저장할 수 있다. 아직 완료되지 않은 작업을 commit하지 않고 나중에 다시 꺼내와 작업을 이어할 수 있다.

```
git checkout hello_feature
echo javascript!! > javascript.txt
git add javascript.txt
git stash
git stash save 라고 명령해도 된다.
ls
```

작성한 javascript.txt파일이 워킹디렉토리에서 사라진것을 확인할 수 있다.

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (hello_feature)
$ git stash
Saved working directory and index state WIP on hello_feature: c1eb29d hi.txt를 추가함

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (hello_feature)
$ ls
a.txt b.txt c.txt hello.txt hi.txt
```



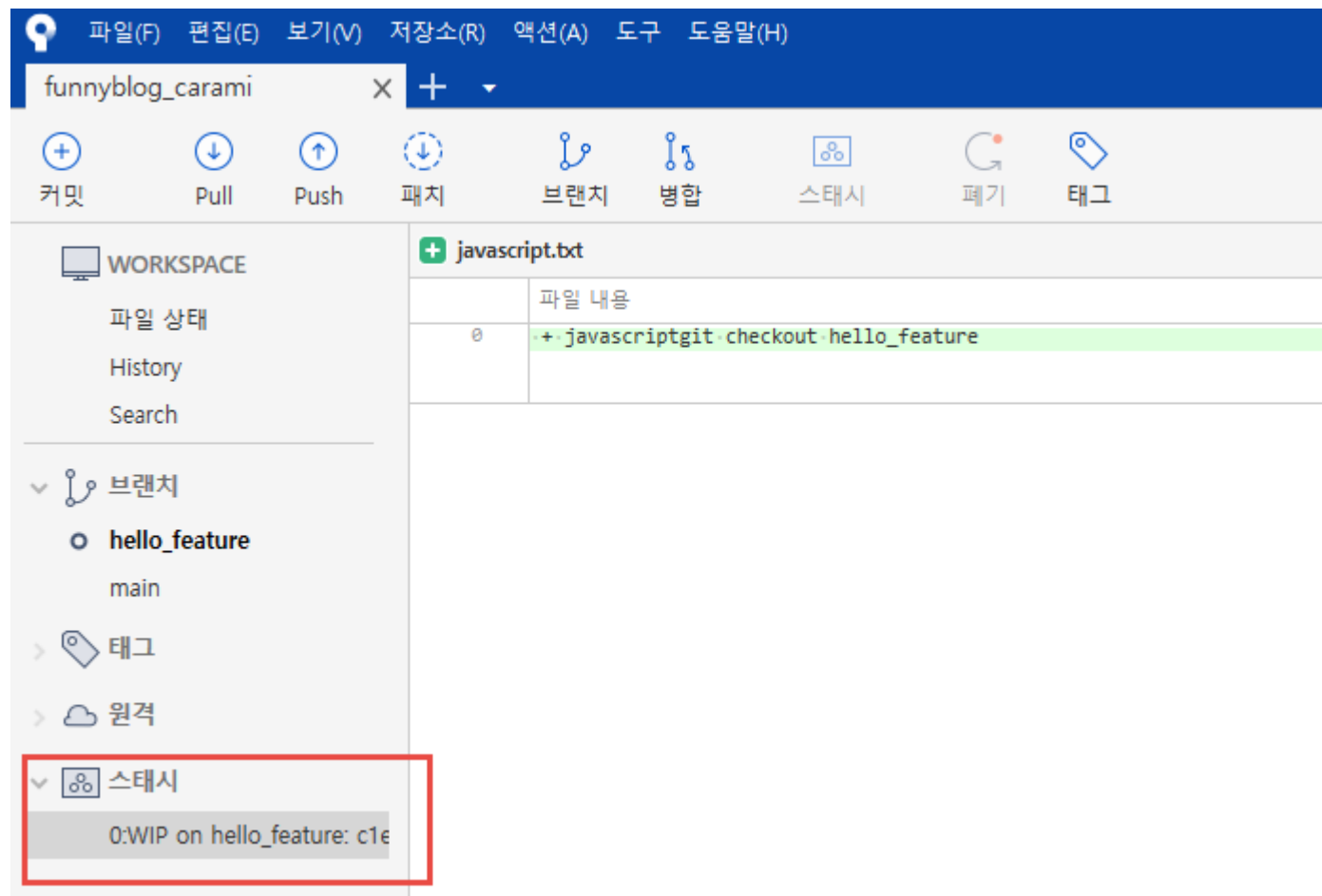
스태시(stash)

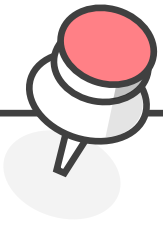
git stash list

스태시 목록 확인

SourceTree에서도 확인가능.

```
$ git stash list
stash@{0}: WIP on hello_feature: c1eb29d hi.txt를 추가함
```





스태시(stash)

`git stash apply`

스태시에 넣었던 내용을 다시 가지고 온다.

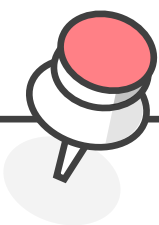
`git stash apply [stash이름]`

위와 같이 명령을 수행할 수도 있다. 스태시는 여러 개 저장될 수 있다.

```
urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (hello_feature)
$ git stash apply
On branch hello_feature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   javascript.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    c.txt

urstory@DESKTOP-4SM171R MINGW64 /c/devel/funnyblog_carami (hello_feature)
$ ls
a.txt  b.txt  c.txt  hello.txt  hi.txt  javascript.txt
```



스태시(stash)

`git stash drop`

가장 최근의 stash를 제거한다.

`git stash drop [stash이름]`

stash이름에 해당하는 stash를 제거한다.

```
$ git stash drop
Dropped refs/stash@{0} (68a252b071d288da19e71128a7ee2dcc130a1bc6)
```

funnyblog_carami

커밋 Pull Push 패치 브랜치 병합 스택시 폐기 태...

WORKSPACE

파일 상태

History

Search

브랜치

- hello_feature
- main

태그

원격

스태시

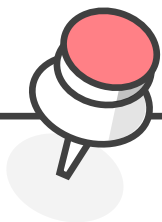
모든 브랜치 원격 브랜치 날짜순으로 정렬

그래프	설명	날짜
	커밋하지 않은 변경사항	18 4 202
	origin/main main a.txt, b.t	18 4 202
	Merge pull request #1 from carami/mai	18 4 202
	hello_carami를 추가	18 4 202
	hello.txt의 코드 충돌을 해결 bye가 맞는	18 4 202
	hello.txt에 hi를 추가함	18 4 202
	hello.txt를 수정함	18 4 202
	hello_feature origin/hello_fe	18 4 202
	hello.txt 파일 생성	18 4 202
	a파일에 hi를 추가	18 4 202
	a.txt, b.txt를 커밋합니다.	18 4 202

파일 상태순 정렬

커밋: c1eb29d09c1ed96d109cc3b0f347ea52e52cba29 [c1eb29d
상위 항목: 25abcaa928
작성자: urstory <urstory@gmail.com>
날짜: 2021년 4월 18일 일요일 오후 4:04:33
커밋한 사람: urstory

hi.txt를 추가함



그밖에.....

- reset vs revert

<https://www.devpools.kr/2017/02/05/%EC%B4%88%EB%B3%B4%EC%9A%A9-git-%EB%90%98%EB%8F%8C%EB%A6%AC%EA%B8%B0-reset-revert/>
되돌리기

- cherry-pick

<https://medium.com/react-native-seoul/git-cherry-pick-%EC%82%AC%EC%9A%A9%EB%B2%95-fe1a3346bd27>

마치 체리를 골라먹는 것처럼 특정 커밋만 특정 브랜치에 적용시킬 수 있다.

- git rebase

<https://flyingsquirrel.medium.com/git-rebase-%ED%95%98%EB%8A%94-%EB%B0%A9%EB%B2%95-ce6816fa859d>

다른 개발자와 협업하면서 깔끔하게 commit을 관리할 수 있는 rebase