

```
child: Column(  
  crossAxisAlignment: CrossAxisAlignment.  
  children: [  
    /*2*/  
    Conta  
    pad  
    chi  
    '  
    s  
    )  
  ),  
  Text(  
    'Ka  
    sty  
    c  
  ),  
),
```

# Git基礎教學

使用 Sourcetree + Github 進行版本控制

```
children: [  
  con(icon, color: color  
  ontainer(  
margin: const EdgeIns  
child:  
  label  
style
```



**周華瑋**

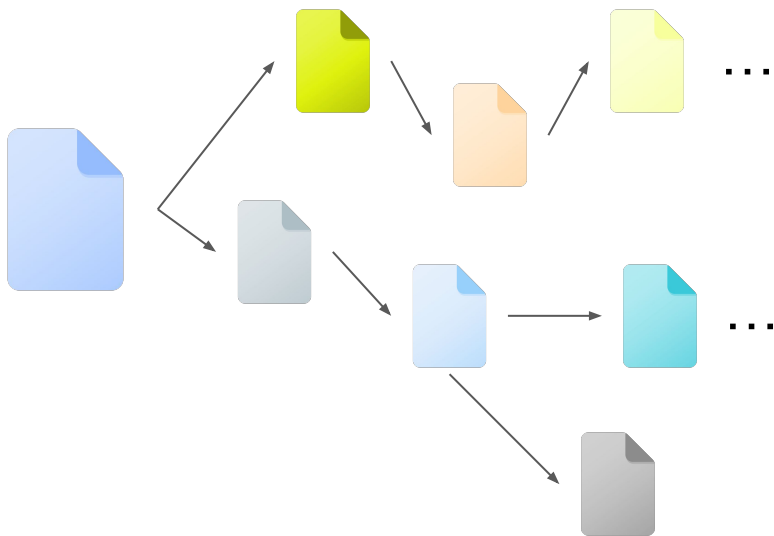
工科系113

GDSC NCKU Unity組長



wei8656

# 版本控制？



**Ctrl+C 、 Ctrl+V**

- 占空間
- 內容差異未知
- 版本資訊未知
- 忘記複製QQ



- **版本儲存**

保留對於檔案新增、修改或是刪除等操作的歷史紀錄

- **共同編輯**

不會因為多人同時進行編輯而導致 內容會被覆蓋

- **儲存空間**

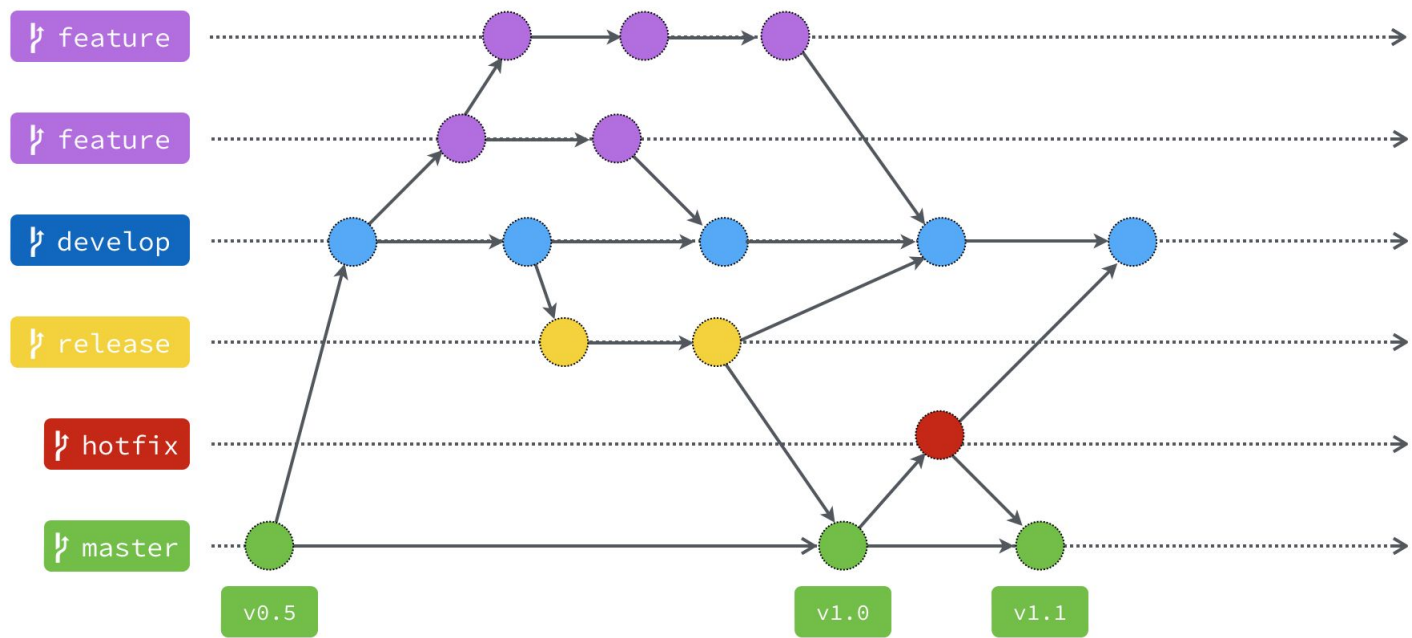
儲存許多資料的同時占用較少的空間

- **Open Source**

免費 👍

- **分散式系統**

可以在沒有伺服器與網路的環境下進行



Wayne@DESKTOP-T6R6178 MINGW64 ~

\$ help

GNU bash, version 4.4.23(1)-release (x86\_64-pc-msys)

These shell commands are defined internally. Type 'help' to see this list.

Type 'help name' to find out more about the function 'name'.

Use 'info bash' to find out more about the shell in general.

Use 'man -k' or 'info' to find out more about commands not in this list.

A star (\*) next to a name means that the command is disabled.

```

job_spec [&]
(( expression ))
. filename [arguments]
:
[ arg... ]
[[ expression ]]
alias [-p] [name=value] ... ]
bg [job_spec ...]
bind [-lpsvPSVX] [-m keymap] [-f file]
break [n]
builtin [shell-builtin [arg ...]]
caller [expr]
case WORD in [PATTERN [| PATTERN]...)>
cd [-L|[-P [-e]] [-@]] [dir]
command [-pVv] command [arg ...]
compgen [-abcdefgjkuv] [-o option] [>
complete [-abcdefgjkuv] [-pr] [-DE] >
compgot [-o|+o option] [-DE] [name ...]
continue [n]
coproc [NAME] command [redirections]
declare [-aAfFgIlNrtux] [-p] [name=v>
dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [jobspec ... | pid >
echo [-neE] [arg ...]
enable [-a] [-dnps] [-f filename] [na>
eval [arg ...]
exec [-cl] [-a name] [command [argume>
exit [n]
export [-fn] [name=value] ...] or ex>
false
fc [-e ename] [-lnr] [first] [last] o>
fg [job_spec]
for NAME [in WORDS ... ] ; do COMMAND>
for (( exp1; exp2; exp3 )); do COMMAN>
function name { COMMANDS ; } or name >
getopts optstring name [arg]
hash [-lr] [-p pathname] [-dt] [name >
help [-dms] [pattern ...]
history [-c] [-d offset] [n] or hist>
if COMMANDS; then COMMANDS; [ elif C>
jobs [-lnprs] [jobspec ...] or jobs >
kill [-s sigspec | -n signal | -sigs>
let arg [arg ...]
local [option] name[=value] ...
logout [n]
mapfile [-d delim] [-n count] [-O or>
popd [-n] [+N | -N]
printf [-v var] format [arguments]
pushd [-n] [+N | -N | dir]
pwd [-LPW]
read [-ers] [-a array] [-d delim] [->
readarray [-n count] [-O origin] [-s>
readonly [-aF] [name[=value] ...] o>
return [n]
select NAME [in WORDS ... ;] do COMM>
set [-abefhkmnptuxvBCHP] [-o option->
shift [n]
shopt [-pqsu] [-o] [optname ...]
source filename [arguments]
suspend [-f]
test [expr]
time [-p] pipeline
times
trap [-lp] [[arg] signal_spec ...]
true
type [-afptP] name [name ...]
typeset [-aAfFgIlNrtux] [-p] name[=v>
ulimit [-SHabcdefiklmnpqrstuvxPT] [l>
umask [-p] [-S] [mode]
unalias [-a] name [name ...]
unset [-f] [-v] [-n] [name ...]
until COMMANDS; do COMMANDS; done
variables - Names and meanings of so>
wait [-n] [id ...]
while COMMANDS; do COMMANDS; done
{ COMMANDS ; }

```

intellipaat@DESKTOP-SPC6JQB MINGW64 ~/ProjectGit1 (master)

\$ git log -3

commit be73fc3e802f8afece5a9f12cea4415665e36bf4 (HEAD -&gt; master, origin/master)

Author: prabhpreetk &lt;prabhpreet.intellipaat@gmail.com&gt;

Date: Wed Jun 19 14:56:33 2019 +0530

Committing 234master.txt in master

intellipaat@DESKTOP-SPC6JQB MINGW64 ~/ProjectGit1 (master)

\$ git checkout branch1

Switched to branch 'branch1'

intellipaat@DESKTOP-SPC6JQB MINGW64 ~/ProjectGit1 (branch1)

\$ git log -3

commit deae5df00b52e75abe175f9f5bdcfde84feb6dd8 (HEAD -&gt; branch1, origin/branch1)

Author: prabhpreetk &lt;prabhpreet.intellipaat@gmail.com&gt;

Date: Wed Jun 19 15:43:54 2019 +0530

123master.txt file modified from feature branch

commit bbf434bc2eceaca5d1742664638a9bd05630636d

Author: prabhpreetk &lt;prabhpreet.intellipaat@gmail.com&gt;

Date: Wed Jun 19 15:41:09 2019 +0530

123branch1.txt file in feature branch; 1st commit in feature branch

commit be73fc3e802f8afece5a9f12cea4415665e36bf4 (origin/master, master)

Author: prabhpreetk &lt;prabhpreet.intellipaat@gmail.com&gt;

Date: Wed Jun 19 14:56:33 2019 +0530

Committing 234master.txt in master

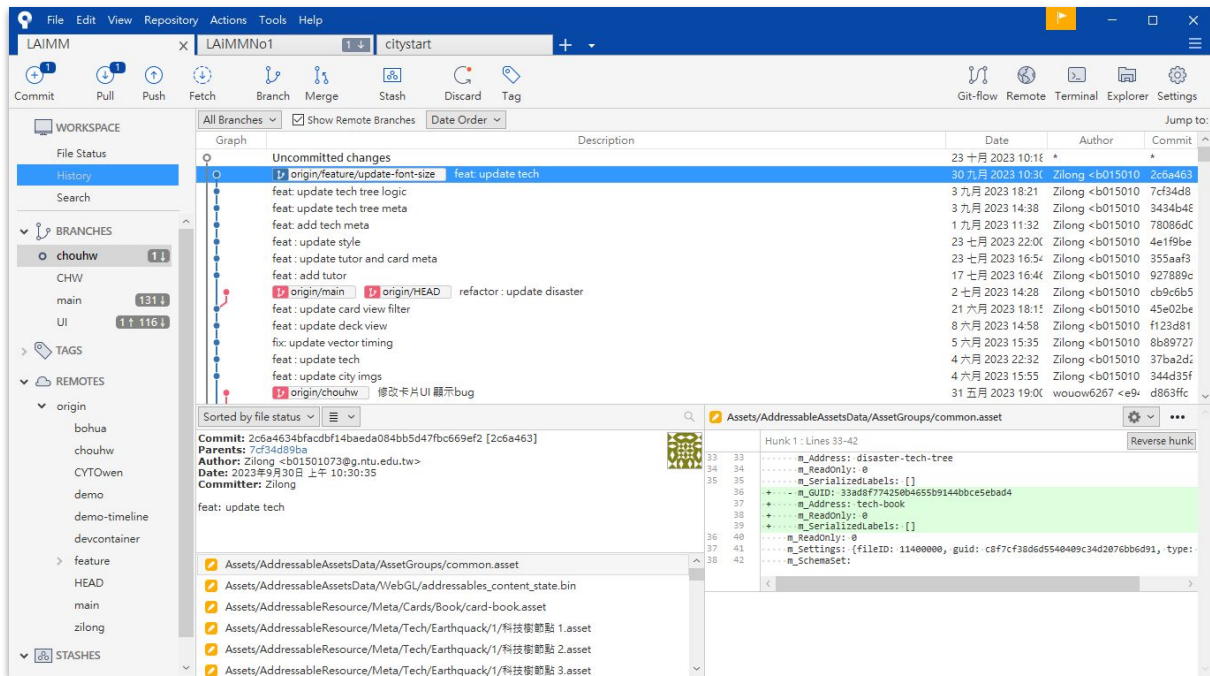


# Sourcetree

## 圖形化介面

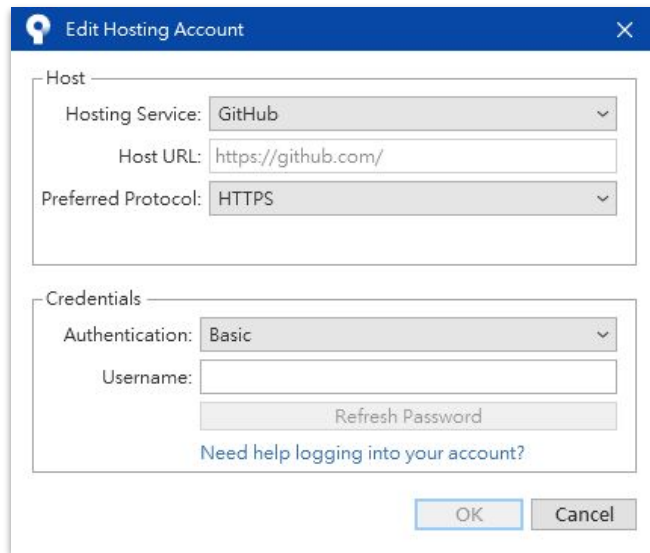
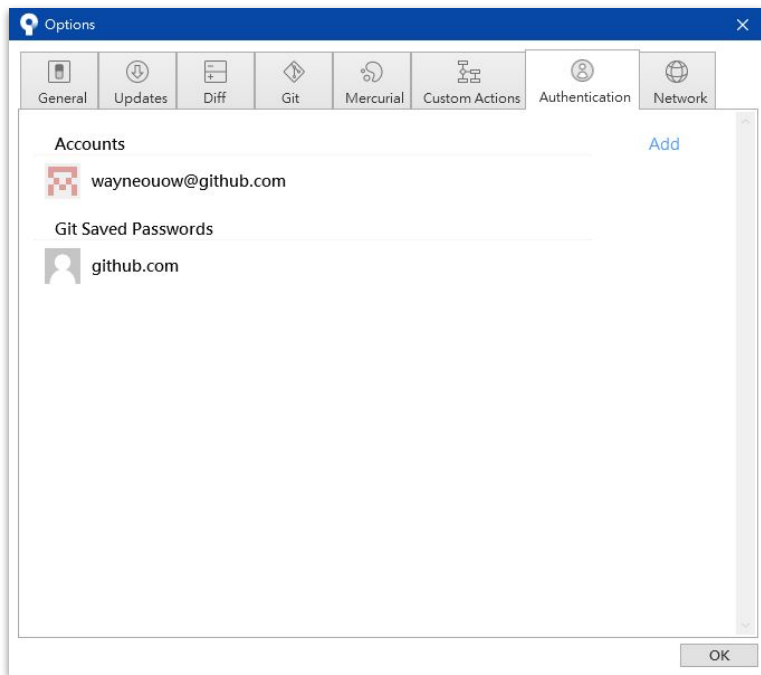
### 將指令轉為視覺化圖形

- 不用打指令
- 一目了然



# 授權設定

Tools -> Options -> Authentication -> Add





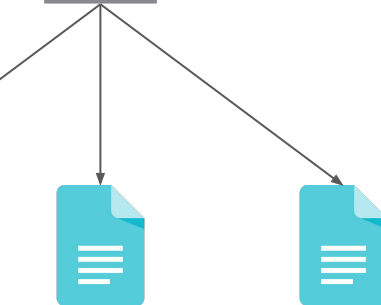
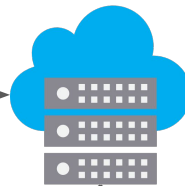


## Git雲端倉庫:一個存放Git的空間

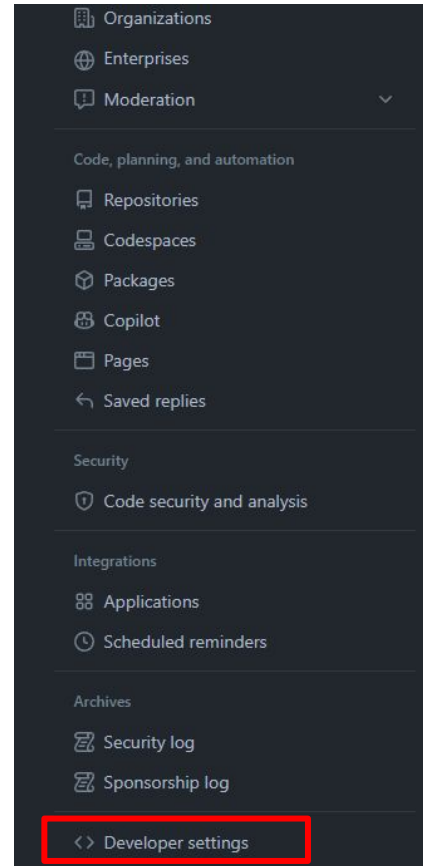
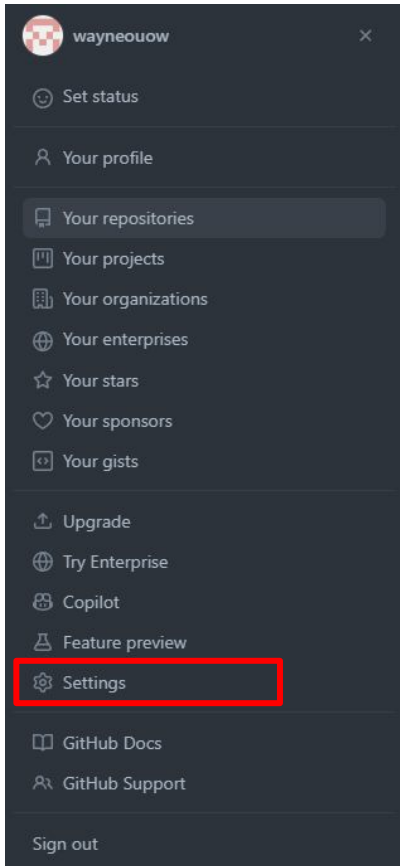
Local



Remote



# Personal Access Tokens



# Personal Access Tokens

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens Beta

**Tokens (classic)**

## Personal access tokens (classic)

**Generate new token** **Revoke all**

Tokens you have generated that can be used to access the [GitHub API](#).

**Sourcetree** — `admin:enterprise`, `admin:pgp_key`, `admin:org`, `admin:org_hook`, `admin:public_key`, `admin:repo_hook`, `admin:ssh_signing_key`, `audit_log`, `codespace`, `delete:packages`, `delete_repo`, `gist`, `notifications`, `project`, `repo`, `user`, `workflow`, `write:discussion`, `write:packages` Last used within the last week **Delete**

**⚠ This token has no expiration date.**

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

# Personal Access Tokens

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

What's this token for?

Expiration \*

30 days

 The token will expire on Wed, Nov 22 2023

Note 及 Expiration  
自行設定

Scopes 選項全打勾 



# Personal Access Tokens

## Personal access tokens

Generate new tokenRevoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

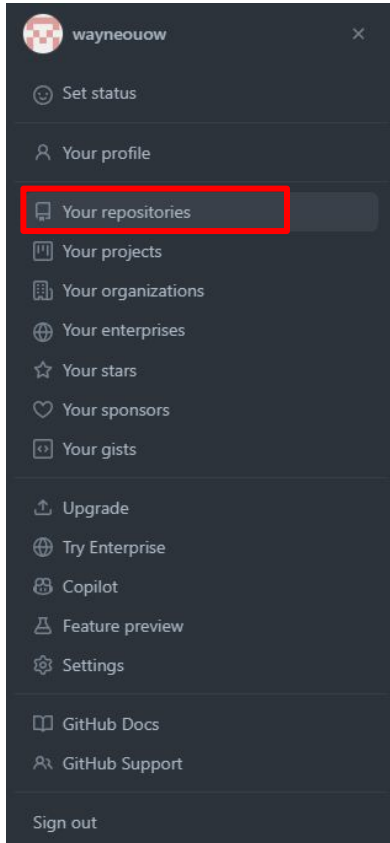
Your tokens

Delete

將這份 token 存起來, 只能在這裡看到一次

如果忘記了就要重新產生一次

# GitHub: Create Repository



## Create a new repository

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

*Required fields are marked with an asterisk (\*).*

Owner \* wayneouow / Repository name \*

Great repository names are short and memorable. Need inspiration? How about [bug-free-barnacle](#)?

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:

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

Add .gitignore

.gitignore template: Unity

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

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


This will set `main` as the default branch. Change the default name in your [settings](#).

① You are creating a public repository in your personal account.

[Create repository](#)

# GitHub: Create Repository

Owner \*      Repository name \*

 wayneouow /

1. `purchaserestservice`
2. `purchase-rest-service`
3. `purchase_rest_service`



153



The problem with camel case is that there are often different interpretations of words - for example, `checkinService` vs `checkInService`. Going along with Aaron's answer, it is difficult with auto-completion if you have many similarly named repos to have to constantly check if the person who created the repo you care about used a certain breakdown of the upper and lower cases. avoid upper case.

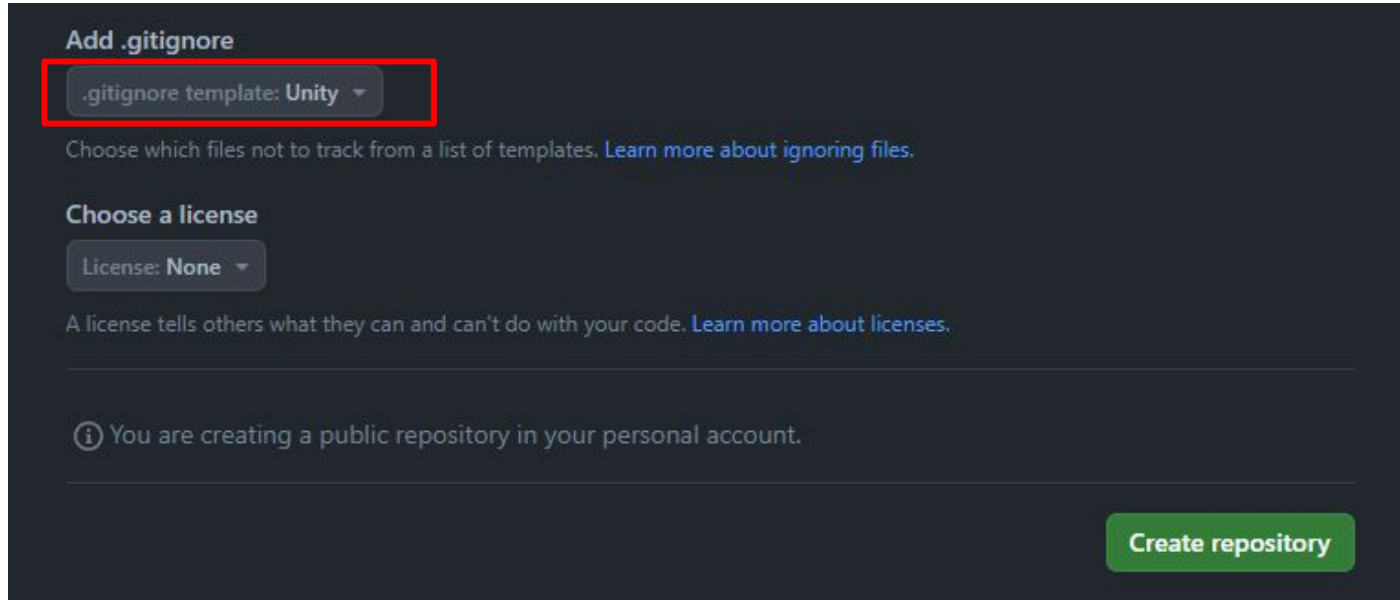


His point about dashes is also well-advised.

1. use lower case.
2. use dashes.
3. be specific. you may find you have to differentiate between similar ideas later - ie use `purchase-rest-service` instead of `service` or `rest-service`.
4. be consistent. consider usage from the various GIT vendors - how do you want your repositories to be sorted/grouped?

[repo naming convention](#)

# GitHub: Create Repository



**Add .gitignore**

.gitignore template: **Unity** ▾


Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

License: **None** ▾

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

---

 You are creating a public repository in your personal account.

---

**Create repository**

[.gitignore?](#)



# GitHub + Sourcetree Clone

The screenshot shows the GitHub interface for a repository named 'test' by user 'wayneouow'. The repository is public and has 1 branch (main) and 0 tags. The 'Code' dropdown menu is open, showing options to clone the repository using HTTPS, SSH, or GitHub CLI. The HTTPS URL, `https://github.com/wayneouow/test.git`, is highlighted with a red rectangle. Below the URL, there is a button to copy the URL. Other options in the menu include 'Open with GitHub Desktop', 'Open with Visual Studio', 'Download ZIP', and a promotional message about AI pair programming. The repository's README is visible in the background, showing the title 'gdsc-test' and a description 'My description'.

wayneouow / test

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

test Public

main 1 branch 0 tags

Go to file Add file <> Code

Local Codespaces

Clone

HTTPS SSH GitHub CLI

`https://github.com/wayneouow/test.git`

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Open with Visual Studio

Download ZIP

Code 55% faster with AI pair programming.

Start my free trial Don't show again

About

My description

Readme Activity 0 stars 1 watching 0 forks

Releases

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

Packages

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

# GitHub + Sourcetree Clone

Local Remote Clone Add Create

## Clone

Cloning is even easier if you set up a remote account

Source Path / URL: Browse

Repository Type: ? No path / URL supplied

Destination Path: Browse

Name:

Local Folder:  
[Root]

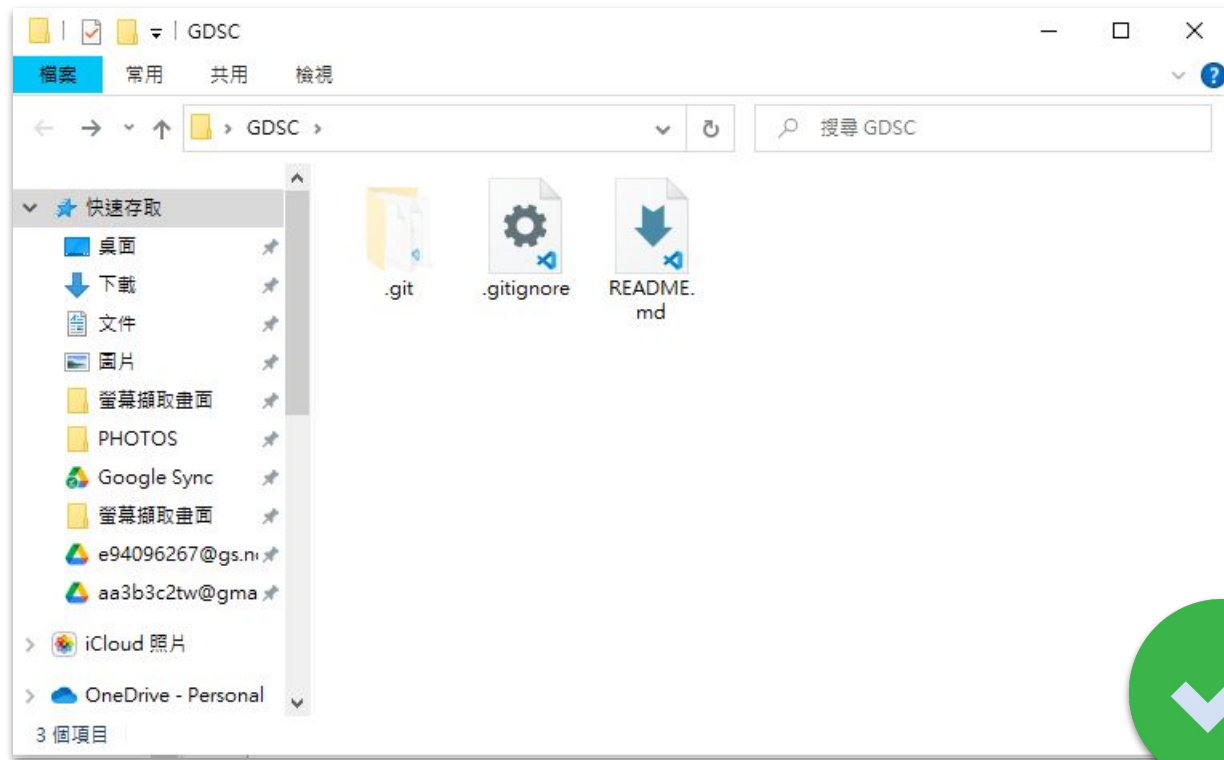
> Advanced Options

Clone

貼上 GitHub Repo之URL

選擇存放位置

# GitHub + Sourcetree Clone



# Sourcetree

The image shows the Sourcetree application interface. The top toolbar includes icons for Commit, Pull, Push, Fetch, Branch, Merge, Stash, Discard, and Tag. On the right, there are icons for Git-flow, Remote, Terminal, Explorer, and Settings.

The left sidebar contains a 'WORKSPACE' section with 'File Status', 'History' (selected), and 'Search'. Below this are 'BRANCHES' (main), 'TAGS', 'REMOTES', and 'STASHES'.

The main area displays a commit history table with columns: Graph, Description, Date, Author, and Commit. The selected commit is 'Initial commit' with description 'Initial commit', dated '23 十月 2023 11:34', by 'wayneouow <82 84aecce'.

Below the commit history, the 'Sorted by file status' section shows a list of files: '.gitignore' and 'README.md'. The 'File Contents' pane on the right shows the content of the selected file, '.gitignore', which includes instructions on how to use the file and lists various file types to ignore.

Graph	Description	Date	Author	Commit
	Initial commit	23 十月 2023 11:34	wayneouow <82 84aecce	84aecce

Sorted by file status

File Contents
<pre>+ # This .gitignore file should be placed at the root of your Unity project direc + # + # Get latest from https://github.com/github/gitignore/blob/main/Unity.gitignore + # + /[Ll]ibrary/ + /[Tt]emp/ + /[Oo]bj/ + /[Bb]uild/ + /[Bb]uilds/ + /[Ll]ogs/ + /[Uu]ser[ss]ettings/ + + # MemoryCaptures can get excessive in size. + # They also could contain extremely sensitive data + /[Mm]emoryCaptures/ + + # Recordings can get excessive in size + /[Rr]ecordings/ + + # Uncomment this line if you wish to ignore the asset store tools plugin</pre>

```
child: Column(  
  crossAxisAlignment: CrossAxisAlignment.  
  children: [  
    /*2*/  
    Conta  
    pad  
    chi  
    '  
    s  
    )  
  ),  
),  
Text(  
  'Ka  
  sty  
  c  
  ),  
),  
),
```

# Git Basic

## File Status

```
graph LR; A[File Status] --- B[Untracked]; A --- C[Changes not staged for commit]; A --- D[Changes to committed]; A --- E[Committed];
```

### Untracked

在版本提交後才又加進來的檔案  
這些檔案並沒有被GIT所追蹤控管

### Changes not staged for commit

已提交版本後, 卻又再次修改  
這些檔案會被丟回工作目錄(WD)

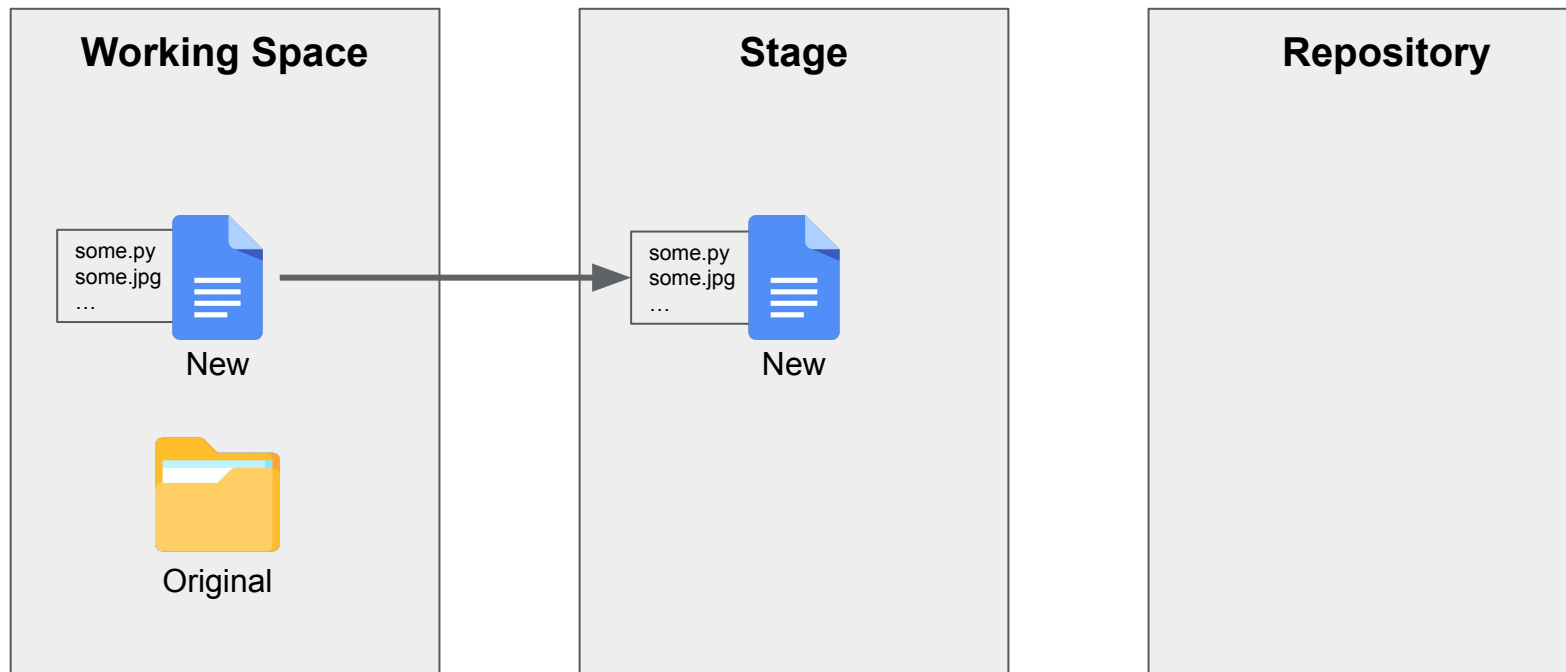
### Changes to committed

在工作目錄檔案執行git add後  
會放在暫存區(Stage)等待提交

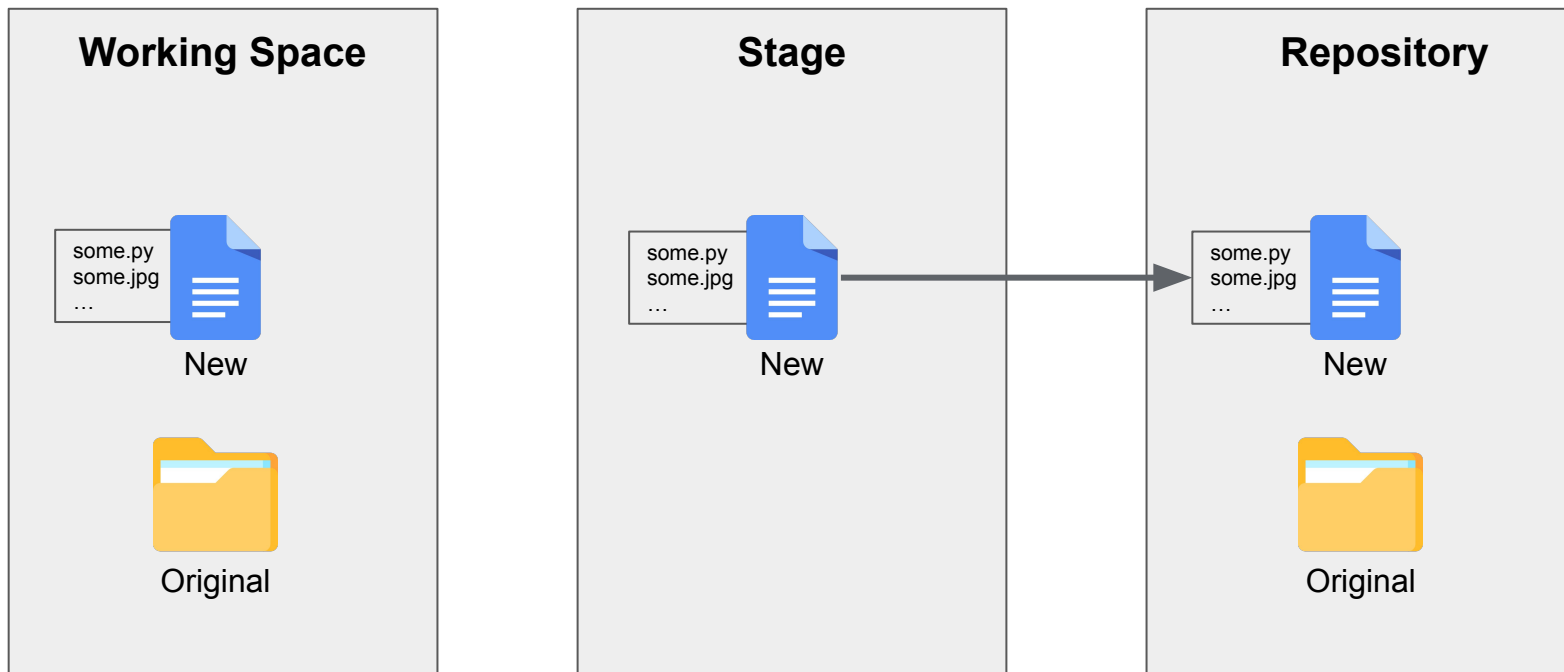
### Committed

在暫存區檔案執行git commit後  
置於儲存區(Repo), 即是已提交的狀態

# Add

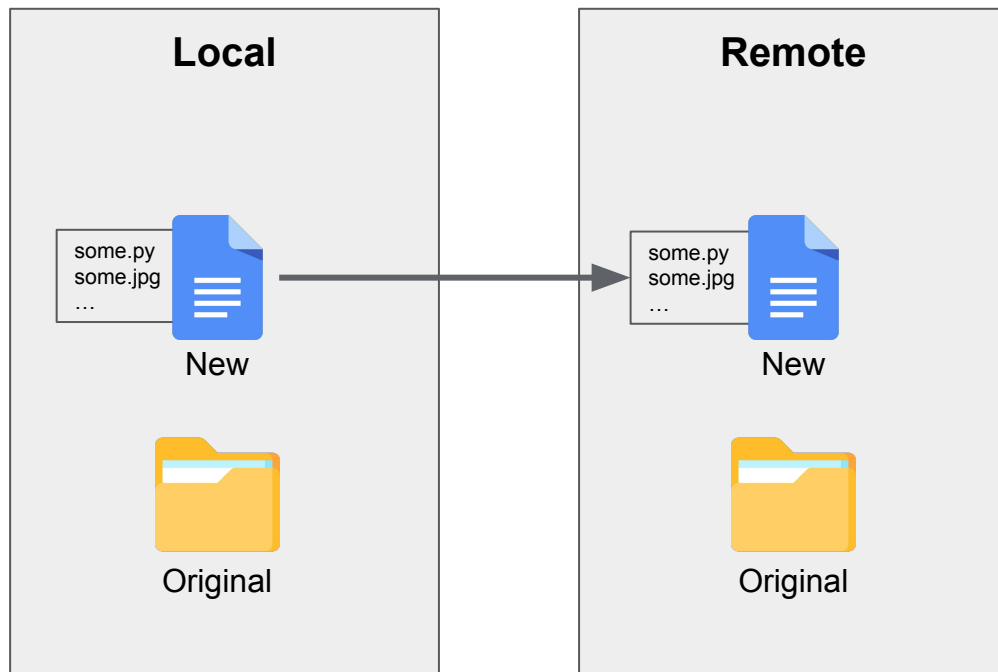


# Commit

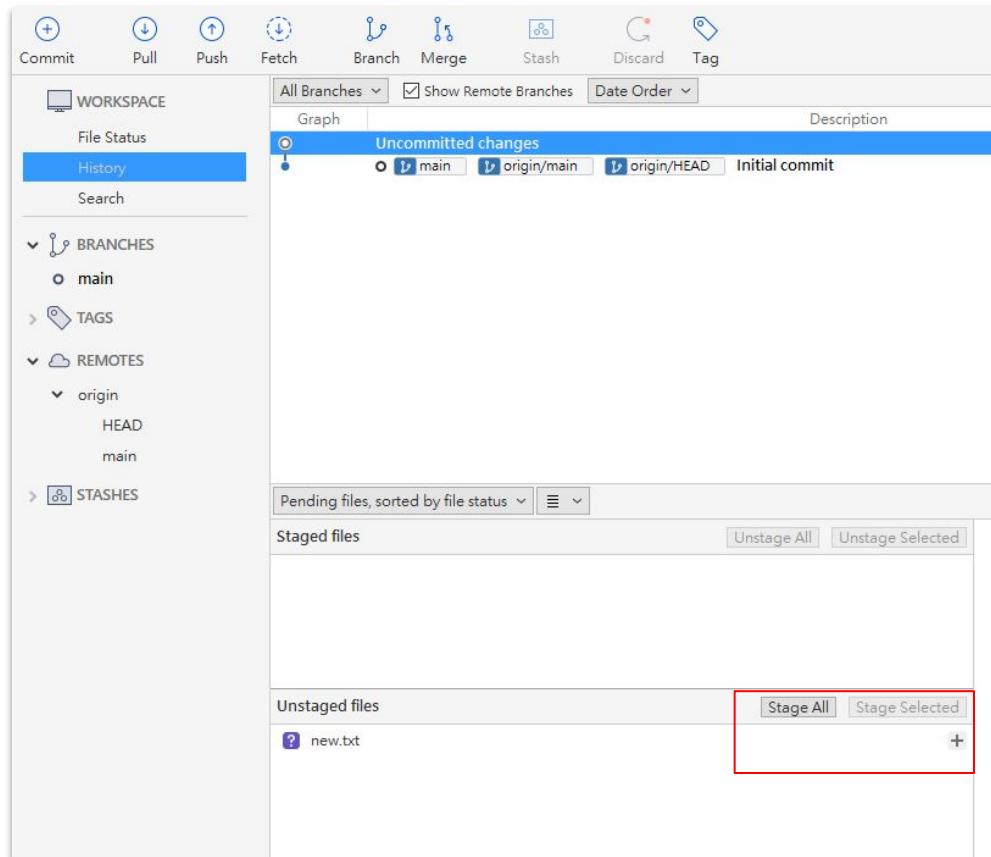




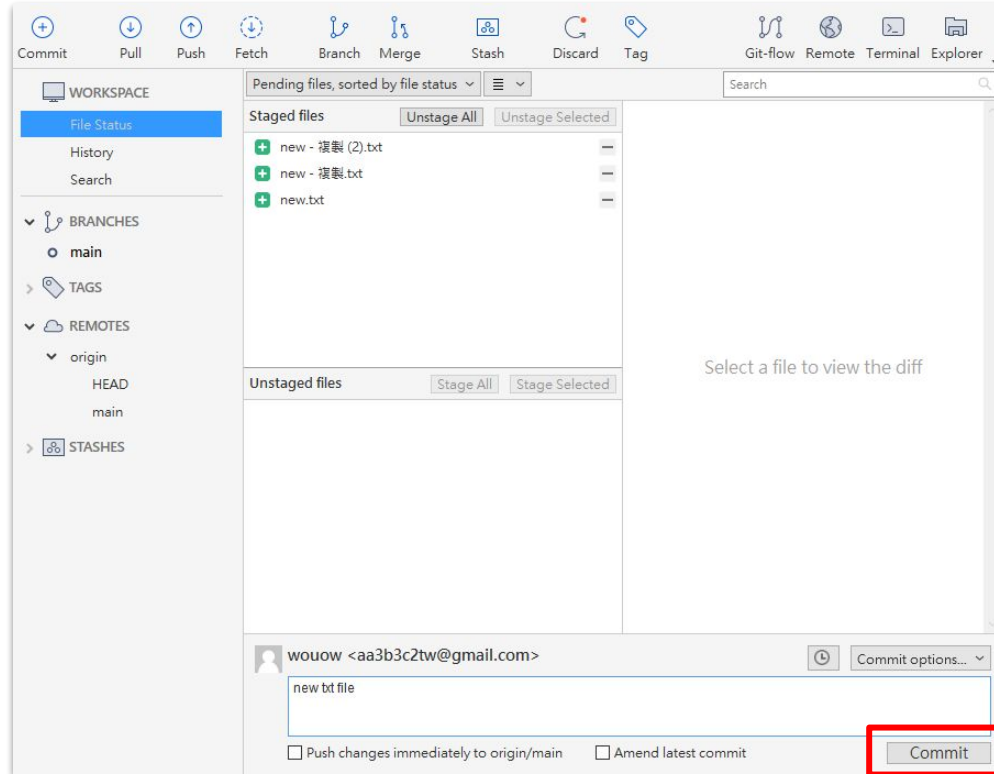
# Push



# File Status







# File Status

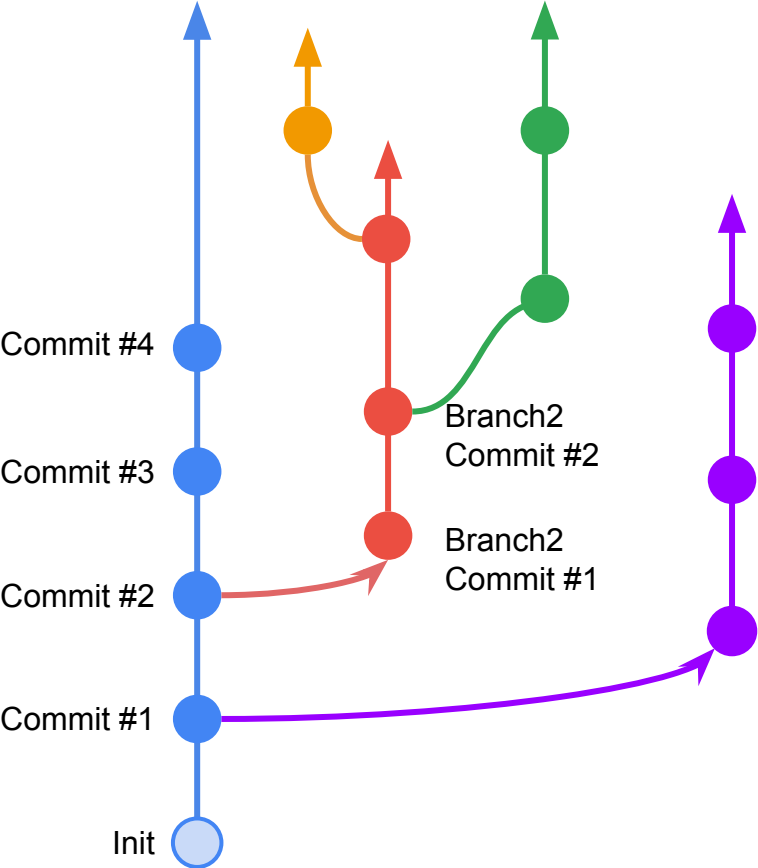


# Committed

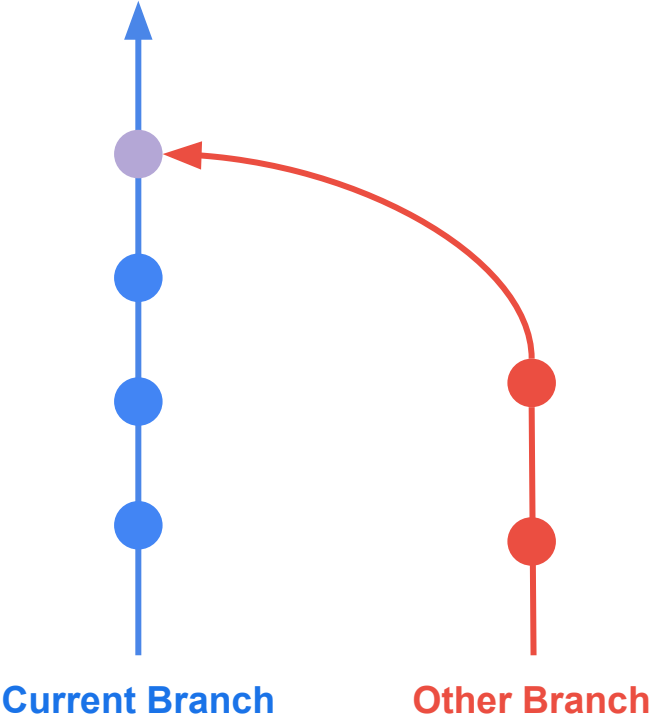
The screenshot displays a Git GUI interface. On the left, a sidebar is divided into two sections: 'WORKSPACE' and 'BRANCHES'. Under 'WORKSPACE', 'History' is selected. Under 'BRANCHES', the 'main' branch is listed with a '1 ↑' indicator. The main panel shows a commit history graph. At the top, there are controls: 'All Branches' (dropdown), 'Show Remote Branches' (checked checkbox), and 'Date Order' (dropdown). The graph has two columns: 'Graph' and 'Description'. A single commit is shown on the 'main' branch, represented by a blue circle with a dot below it. The commit is labeled 'new' in the 'Description' column, with 'Initial commit' below it. Remote branches are shown as 'origin/main' and 'origin/HEAD' with their respective icons.

Graph	Description
	 main 1 ↑ new
	origin/main
	origin/HEAD
	Initial commit

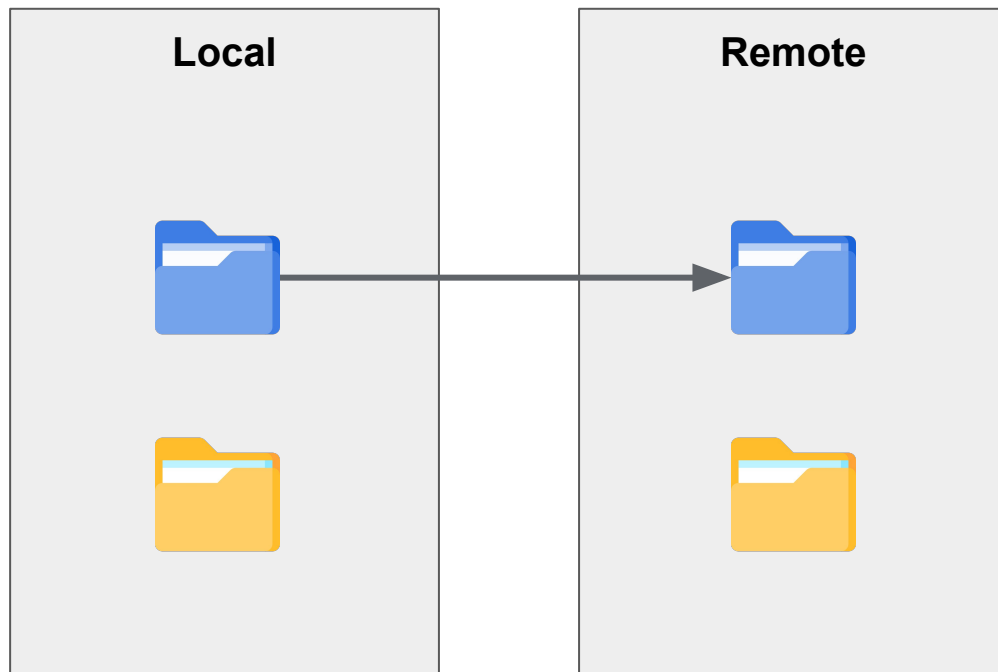
# Branch



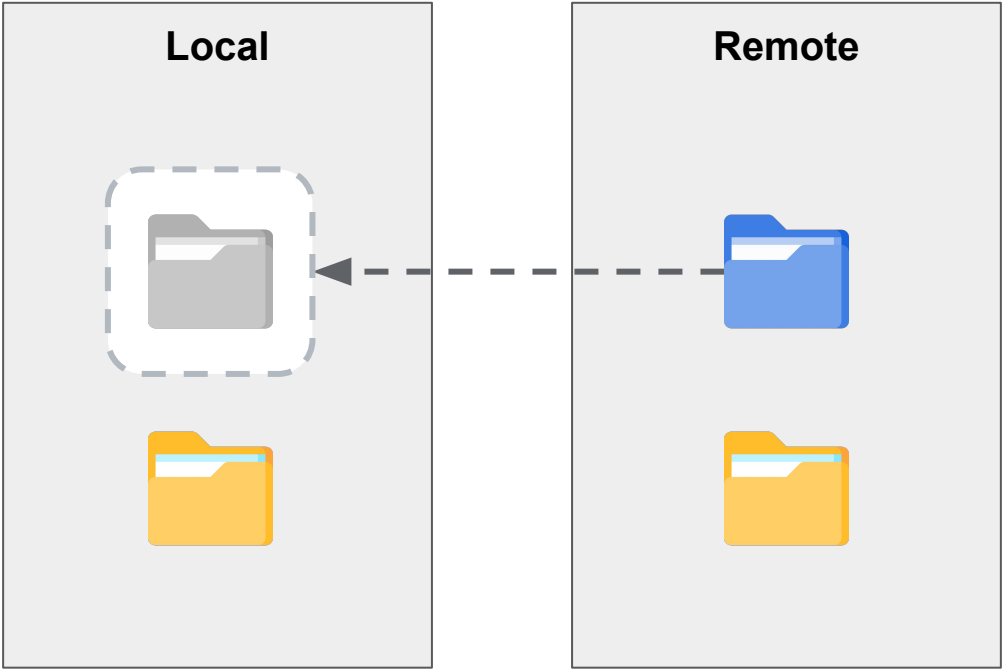
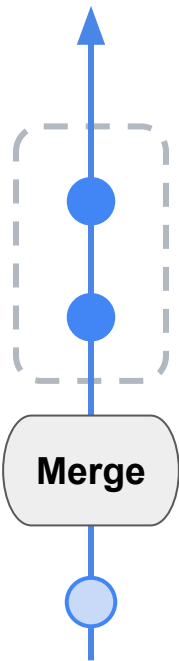
# Merge



# Push

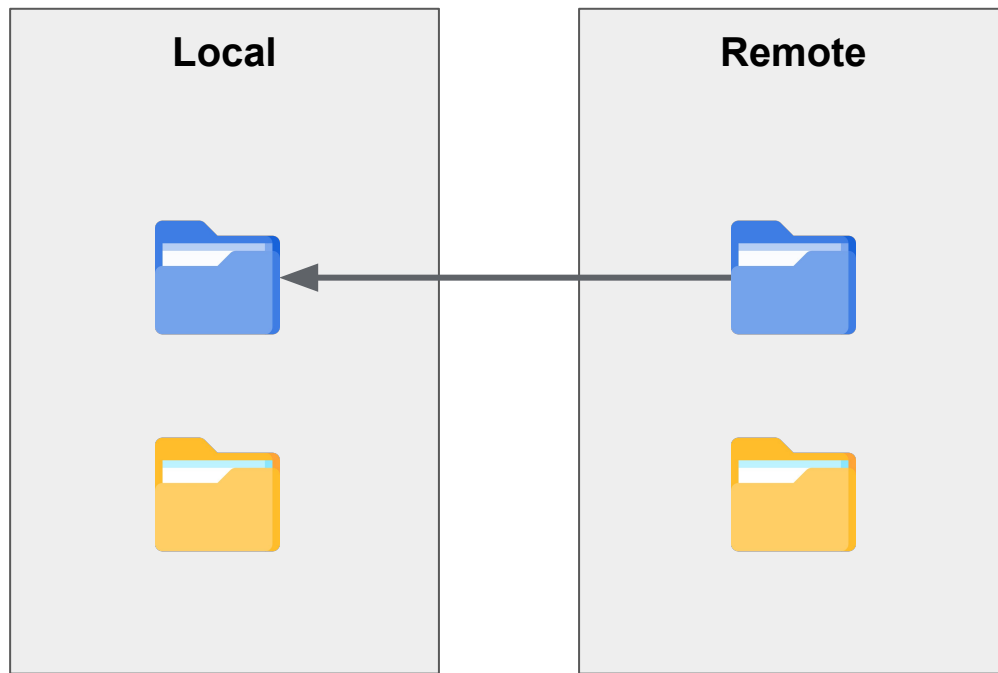
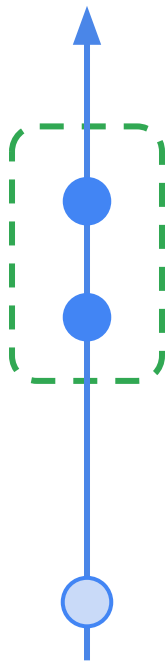


# Fetch





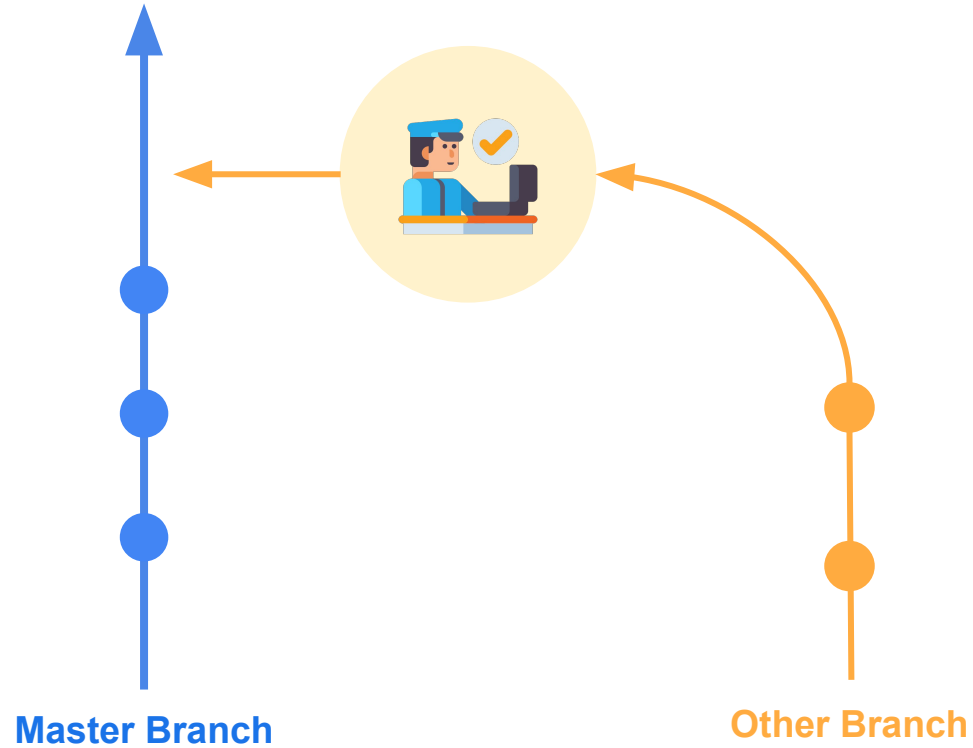
# Pull



取得遠端新檔案 (不合併): `git fetch`

取得遠端新檔案並合併: `git pull = git fetch + git merge origin/master`

## Github: Pull Request (PR)

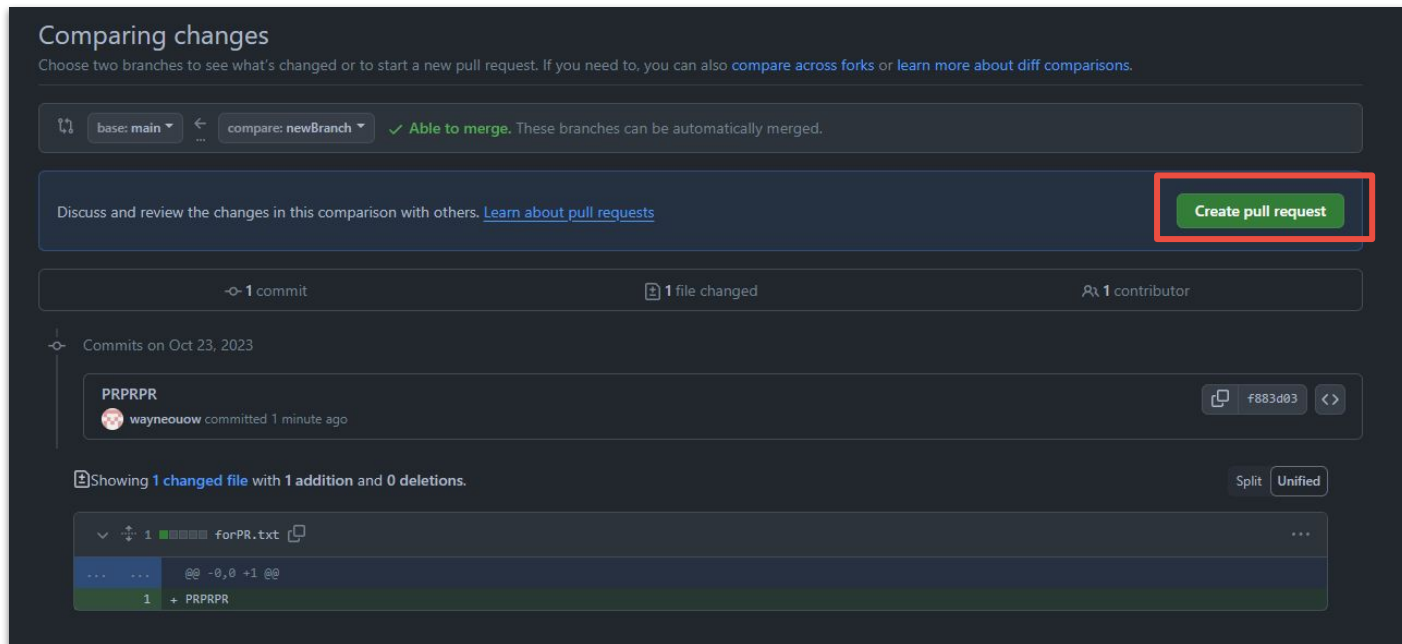


# Github: Pull Request (PR)

The screenshot shows a GitHub repository interface for a repository named "test" (Public). The main branch is "main", and there are 2 branches and 0 tags. A dropdown menu for "Switch branches/tags" is open, showing a search bar "Find or create a branch..." and a list of branches: "main" (checked) and "newBranch". The commit history on the right shows a sequence of commits:

Commit Message	Commit Hash	Time	Commits
Initial commit	352c7ad	16 hours ago	17
Merge branch 'main' of <a href="https://github.com/wayneouow/test">https://github.com/wayneouow/test</a>		16 hours ago	
for PR		16 hours ago	
add newBranch.txt		18 hours ago	

# Github: Pull Request (PR)



The screenshot shows the GitHub 'Comparing changes' interface for a Pull Request. At the top, the title 'Comparing changes' is followed by a subtitle: '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](#) or [learn more about diff comparisons](#).' Below this, a comparison bar shows 'base: main' and 'compare: newBranch' with a green checkmark and the text 'Able to merge. These branches can be automatically merged.' A large blue box contains the text 'Discuss and review the changes in this comparison with others. [Learn about pull requests](#)' and a prominent green 'Create pull request' button, which is highlighted with a red rectangle. Below the discussion box, a summary bar indicates '1 commit', '1 file changed', and '1 contributor'. A section titled 'Commits on Oct 23, 2023' shows a commit by 'wayneuoow' with the message 'PRPRPR' and a commit hash 'f883d03'. Below the commit, a message states 'Showing 1 changed file with 1 addition and 0 deletions.' and a 'Unified' view tab is selected. The diff for 'forPR.txt' is shown, with a green line indicating a single addition: '1 + PRPRPR'.

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](#) or [learn more about diff comparisons](#).

base: main ← compare: newBranch ✓ 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 1 contributor

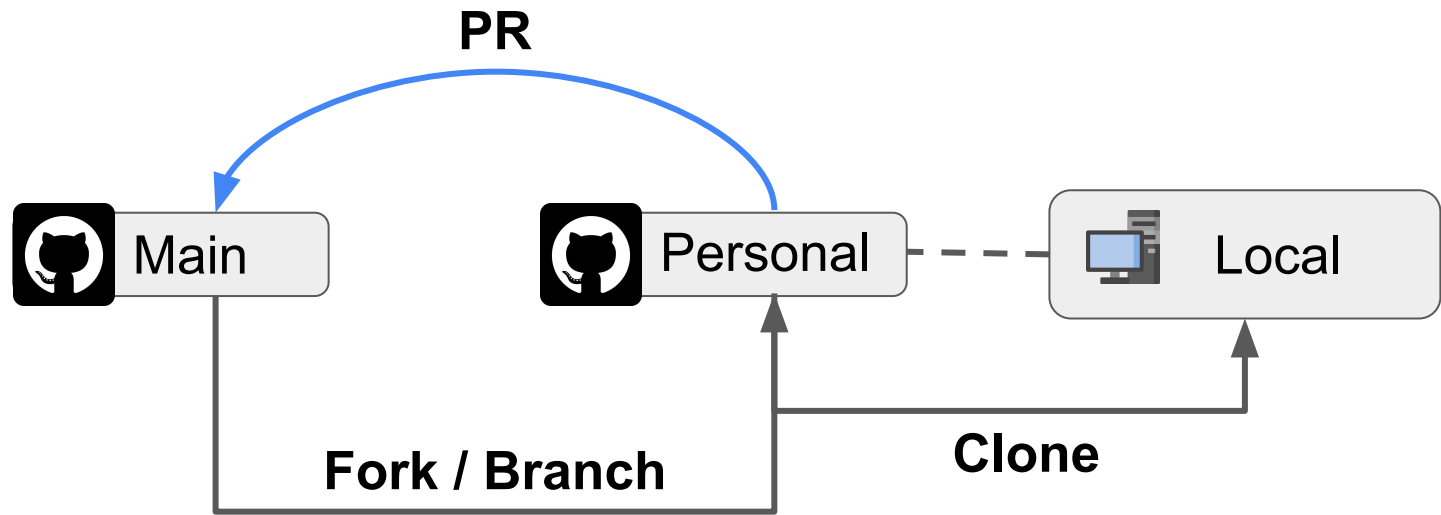
Commits on Oct 23, 2023

PRPRPR  
wayneuoow committed 1 minute ago

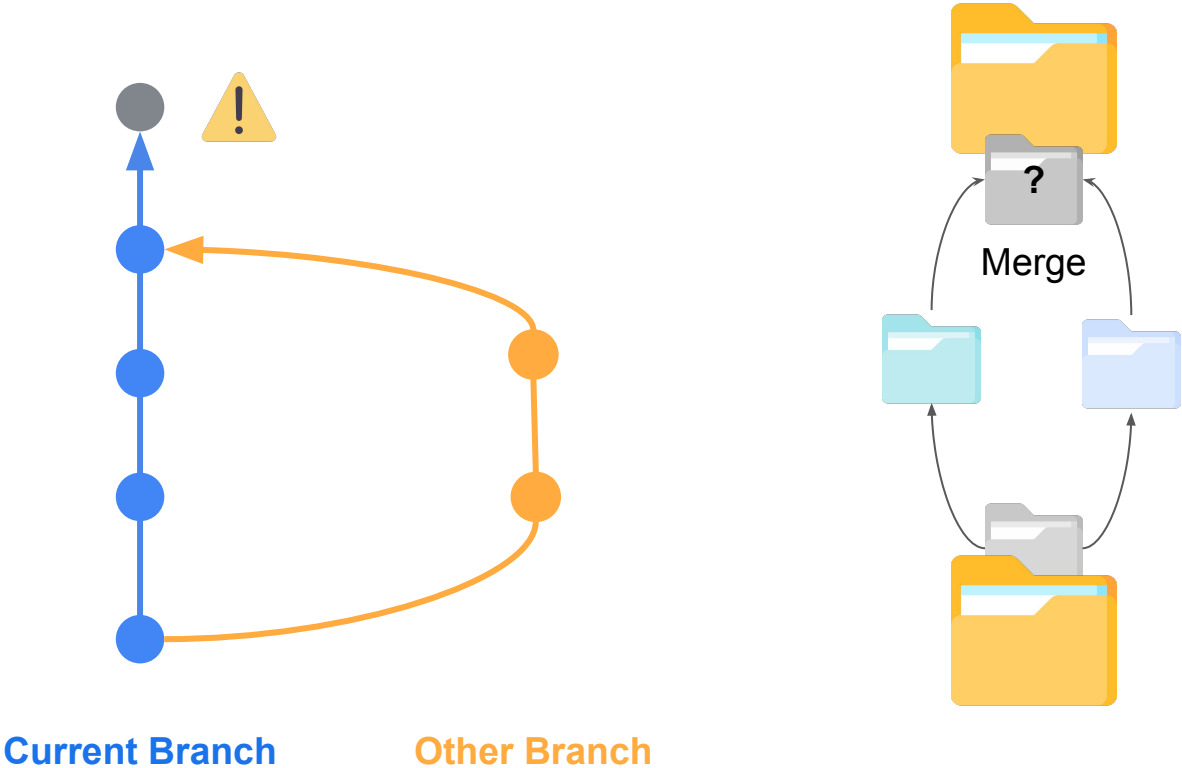
Showing 1 changed file with 1 addition and 0 deletions. Split Unified

1 + PRPRPR

# Github 流程

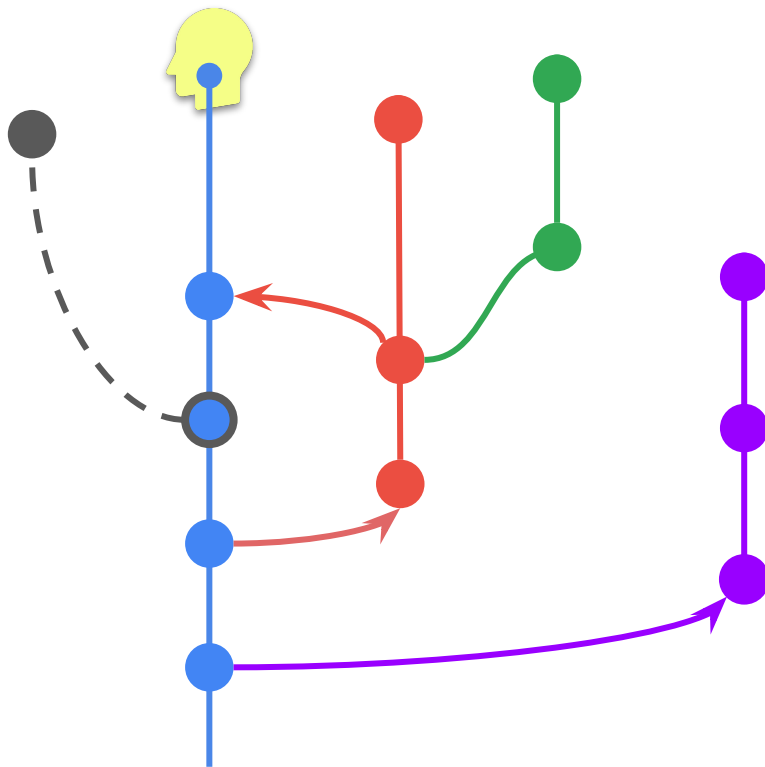


# Conflict

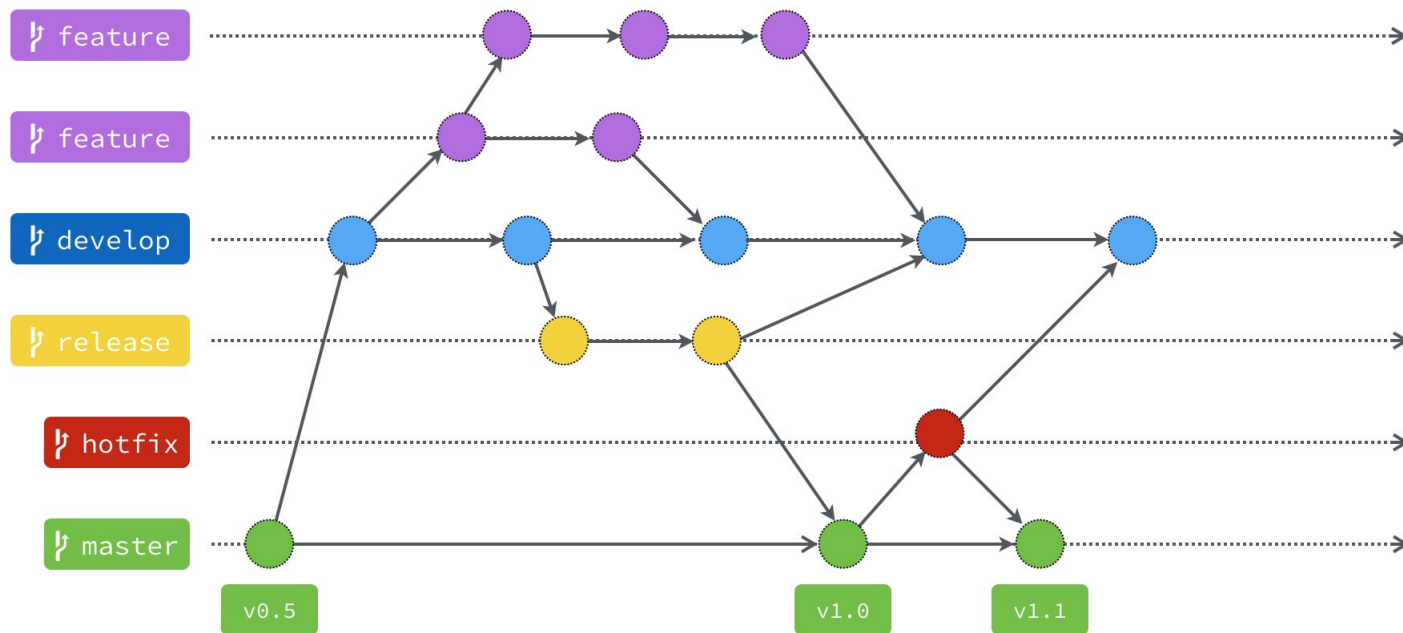


## detached HEAD

HEAD: 指向某一個分支的指標



# Git Flow

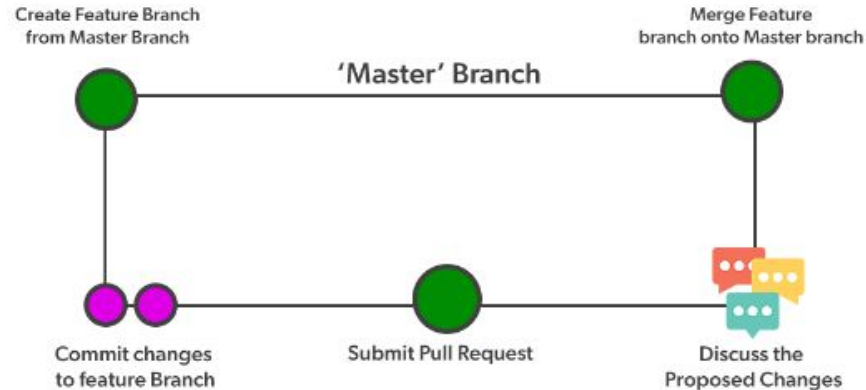


<https://gitbook.tw/chapters/gitflow/why-need-git-flow>



## GitHub Flow

**GitHub flow** is really very simple to adapt as this flow doesn't have releases because your deployment to production happens every day. There are two types of branches involved in this flow, the *master* branch, and the *feature* branch.



<https://www.geeksforgeeks.org/git-flow-vs-github-flow/>

# Sourcetree

The image shows the Sourcetree application interface. The top toolbar includes buttons for Commit, Pull, Push, Fetch, Branch, Merge, Stash, Discard, and Tag. On the right, there are icons for Git-flow, Remote, Terminal, Explorer, and Settings.

The left sidebar contains the following sections:

- WORKSPACE
  - File Status
  - History (selected)
  - Search
- BRANCHES
  - chouhw (selected, 1 ↑ 116 ↓)
  - CHW
  - main (131 ↓)
  - UI (1 ↑ 116 ↓)
- TAGS
- REMOTES
  - origin
    - bohua
    - chouhw
    - CYTOWen
    - demo
    - demo-timeline
    - devcontainer
    - feature
    - HEAD
    - main
    - ziliong
- STASHES

The main area displays a commit history table with columns: Graph, Description, Date, Author, and Commit. The selected commit is 401a017 by wouow, titled "關卡頁面更新".

Graph	Description	Date	Author	Commit
feat: update tax, main title, game logic	30 五月 2023 16:27	Zilong <b015010	ef07903	
origin/CYTOWen feat:完成花費科技點數解鎖科技的功能。 docs-新增MainSceneTechView.cs和MainSceneTechController.cs, 用於實	30 五月 2023 6:09	DESKTOP-7RKMT	3400631	
chouhw 1 ↑ 116 ↓ 關卡頁面更新	30 五月 2023 3:44	wouow <aa3b3c	401a017	
build: update build settings	29 五月 2023 14:51	Rainforest <rainfc	616ce54	
feat:完成在頁面上方顯示玩家資料的功能 docs-新增MainSceneTechModel.cs, 開始著手實作TechPage的功能	29 五月 2023 6:10	DESKTOP-7RKMT	e670075	
chore: build log	28 五月 2023 23:16	Zilong <b015010	d56b521	
feat: update row and meta	28 五月 2023 23:03	Zilong <b015010	360775c	
Merge branch 'main' of https://github.com/LAiMM-Studio/LAiMMNo1	28 五月 2023 18:46	Zilong <b015010	a49df4a	
feat: update card meta and game result	28 五月 2023 18:46	Zilong <b015010	62fc551	
build: update project settings	28 五月 2023 12:56	Rainforest <rainfc	d78a8fd	
docs-新增player.cs和Tech.cs兩個ScriptableObject style-粗略的製作科技頁面的UI	27 五月 2023 7:27	DESKTOP-7RKMT	05d683c	
feat:修改CardRightView.cs	26 五月 2023 7:11	DESKTOP-7RKMT	065520b	
feat: update demo scene state UI	25 五月 2023 14:02	Zilong <b015010	7677cdf	
feat: add ui plugin and update demo scene	25 五月 2023 13:27	Zilong <b015010	693abf3	
feat: update card view	23 五月 2023 16:17	Zilong <b015010	b6eb1f4	

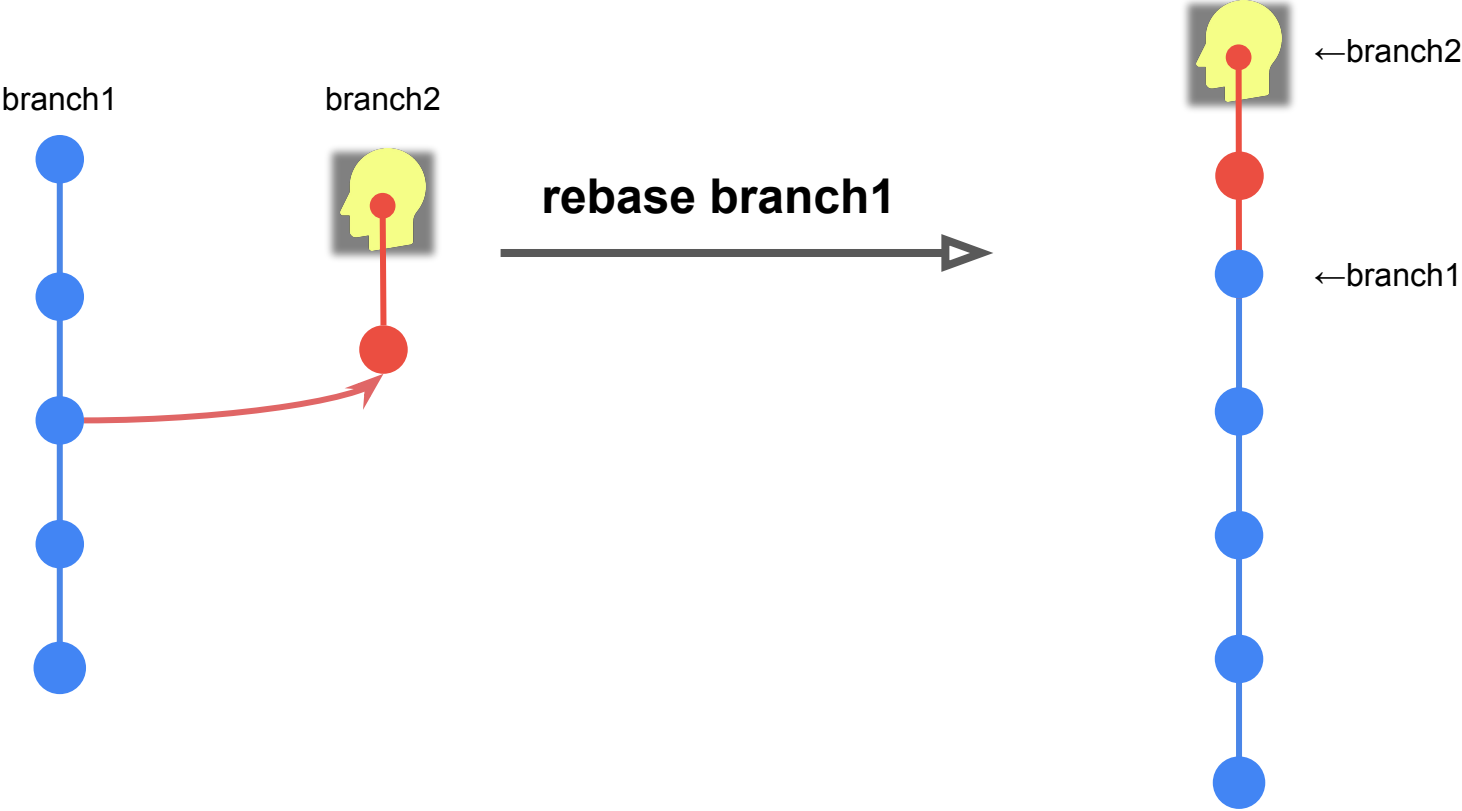
Below the table, the commit details for 401a017 are shown:

Commit: 401a017c0c178cd4e1eed03dfc2ee3b476638711 [401a017]  
Parents: b217ed7950  
Author: wouow <aa3b3c2tw@gmail.com>  
Date: 2023年5月30日 上午 03:44:25  
Committer: wouow

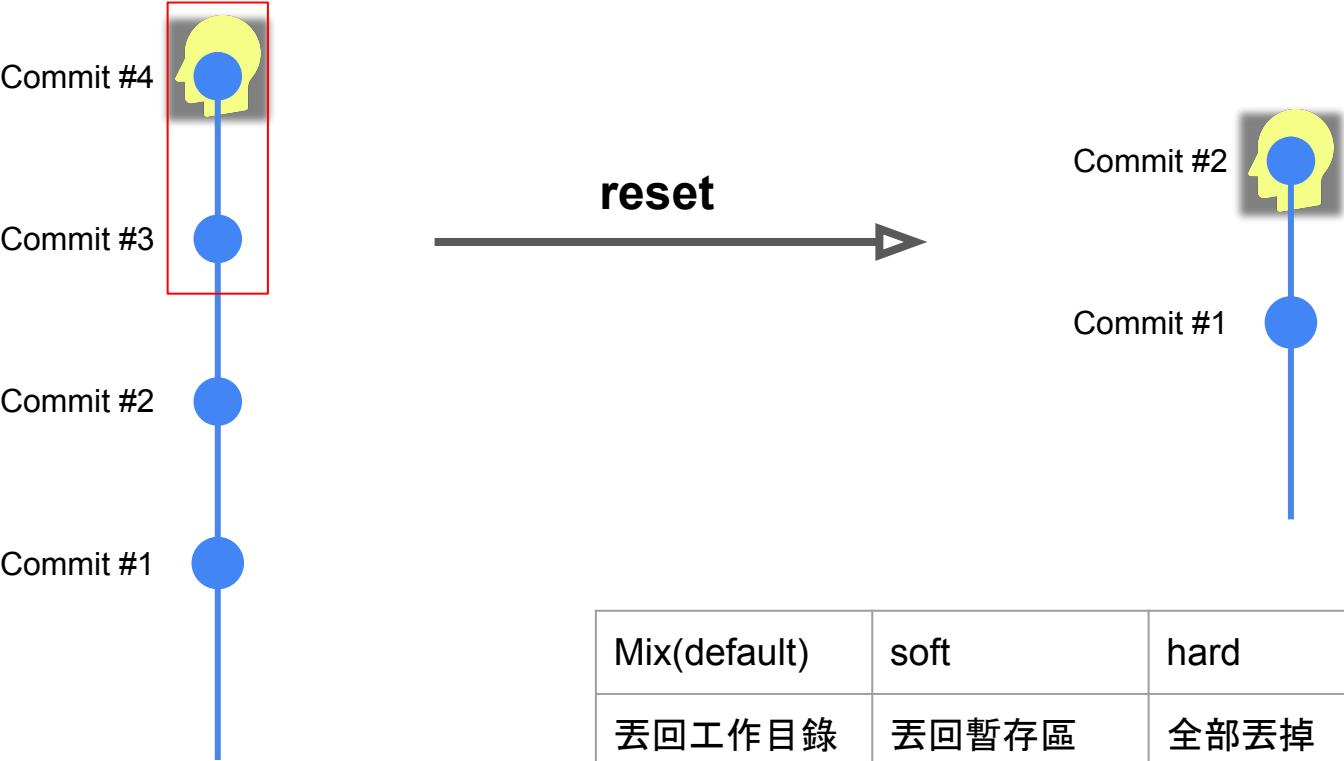
The diff view shows changes to the following files:

- Assets/AddressableResource/Meta/Level/level-list-meta.asset
- Assets/AddressableResource/Meta/Level/level-meta 1.asset
- Assets/AddressableResource/Meta/Level/level-meta 2.asset
- Assets/AddressableResource/Meta/Level/level-meta.asset
- Assets/AddressableResource/Prefab/Main/Report/level-btn.prefab
- Assets/AddressableResource/Prefab/Main/Report/report-level.prefab

# Rebase



# Reset



Mix(default)	soft	hard
丟回工作目錄	丟回暫存區	全部丟掉

# Reverse

