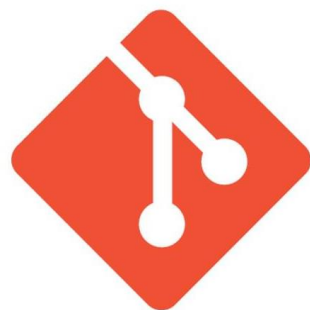


git ×



葉仲仁

steven@ispan.com.tw



# git

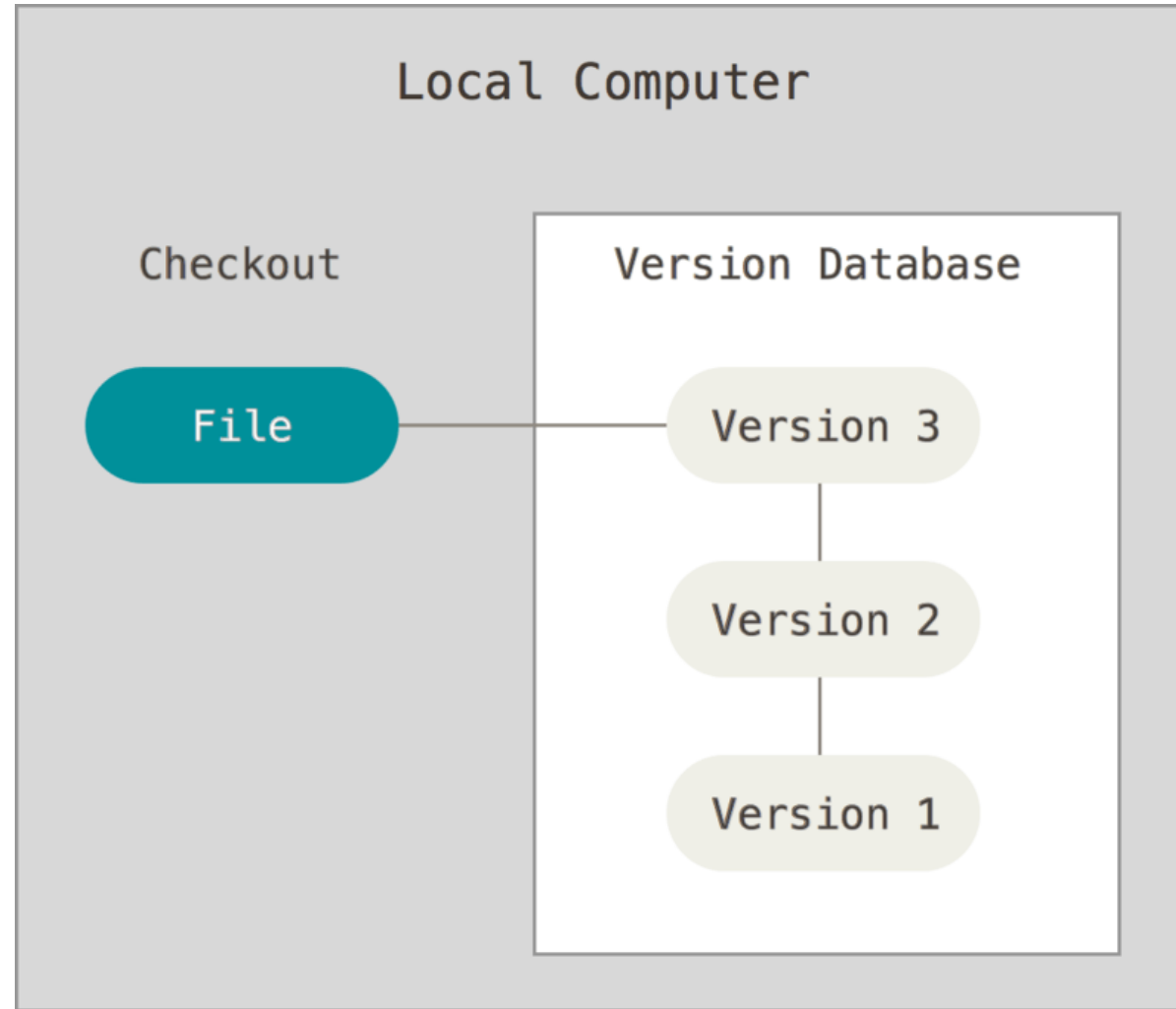
- 一種版本控制的工具。



# GitHub

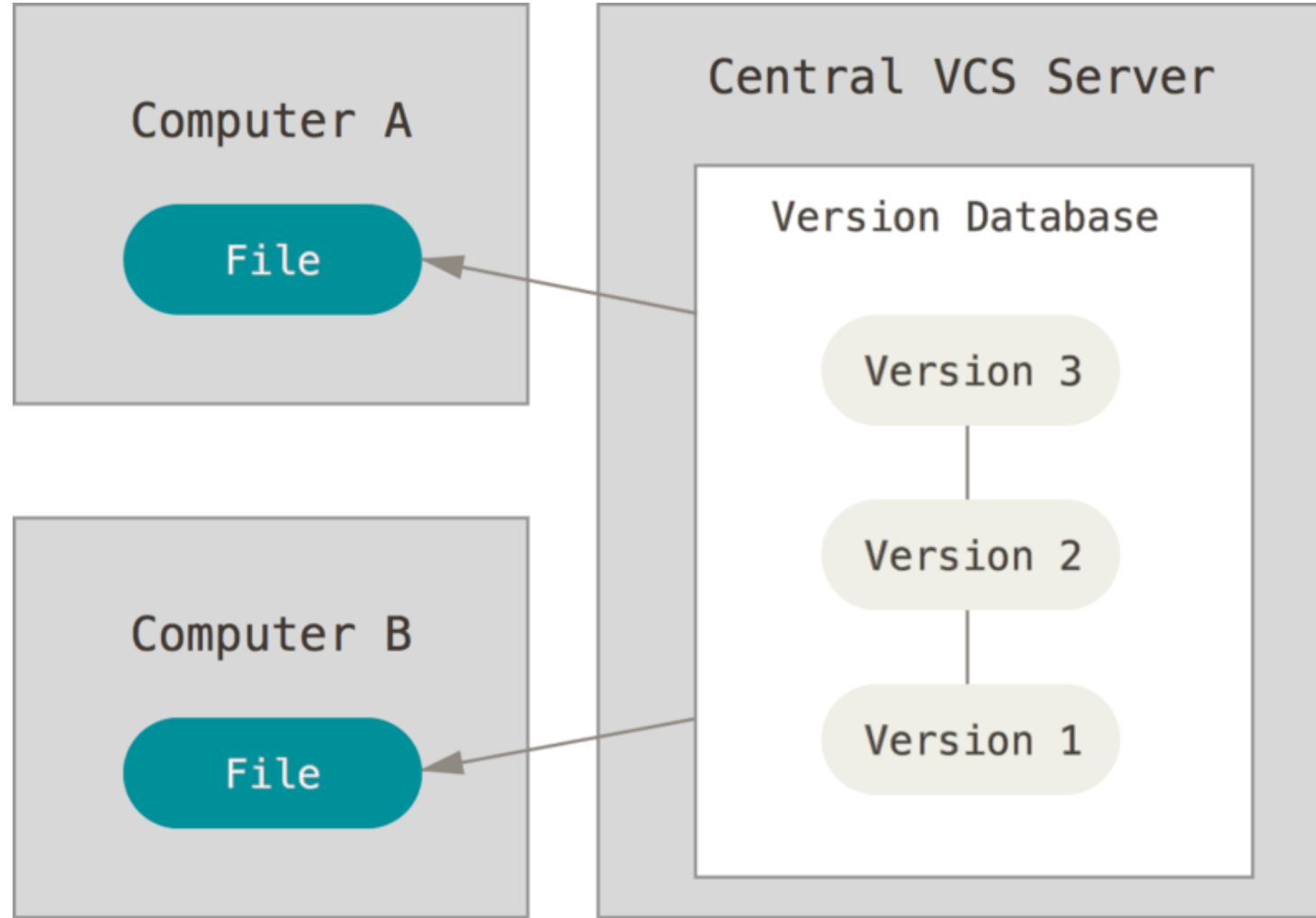
- 一個網站。存放原始碼的空間，可以想成雲端硬碟的概念。

# 本地端版本控制



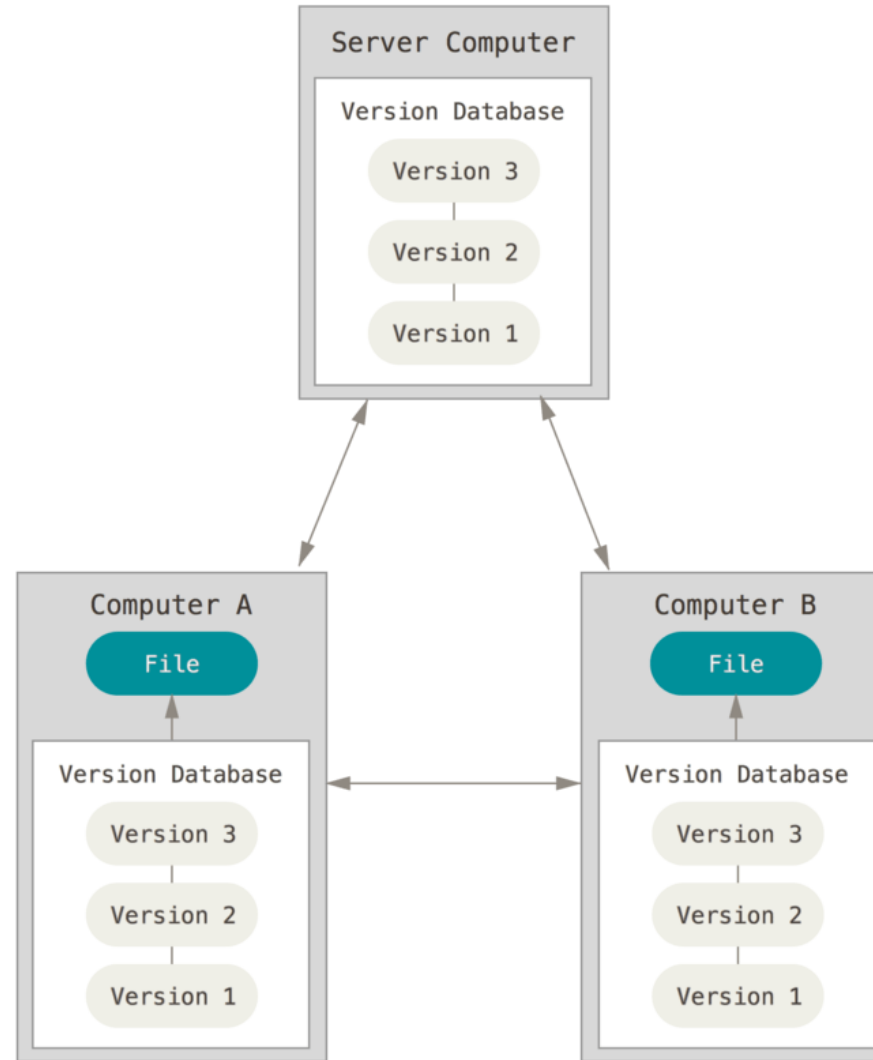
圖片來自：<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

# 集中化的版本控制系統



圖片來自：<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

# 分散式版本控制系統



圖片來自：<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

# 安裝 Git

- 下載 <https://git-scm.com/downloads> 及安裝



- 開啟命令提示字元(終端機)，輸入 `git --version`查詢安裝的版本

# 基本DOS語法(linux 指令)介紹

- cd：顯示目前目錄的名稱或變更。

回到家目錄 `cd ~`

進入某資料夾 `cd myapp`

退回至前頁 `cd ..`

- ls：顯示目錄的內容
- nano：編輯檔案

```
User@DESKTOP-T5T06CG MINGW64 ~  
$ cd ~  
  
User@DESKTOP-T5T06CG MINGW64 ~  
$ cd myapp  
  
User@DESKTOP-T5T06CG MINGW64 ~/myapp (master)  
$ cd ..  
  
User@DESKTOP-T5T06CG MINGW64 ~  
$ ls  
AppData/  
'Application Data'  
Contacts/  
Cookies@  
Documents/  
Downloads/
```

# 使用者設定

- 開啟終端機設定使用者資訊

```
git config --global user.name "steven"  
git config --global user.email "steven@ispan.com.tw"
```

- 檢視目前的設定

```
git config --list
```

- 檢視單一設定

```
git config user.name
```

- 設定的資料儲存在

- ✓ C:\Users\<使用者帳號>\.gitconfig



# 開始使用 – 建立 Git儲存庫(Repository)

```
git init
```

```
E:\myapp>git init  
Initialized empty Git repository in E:/myapp/.git/
```

- 這個動作會建立一個叫.git隱藏目錄
  - .git目錄中紀錄了儲存庫中的所有更動的紀錄
- 查看儲存庫的狀態

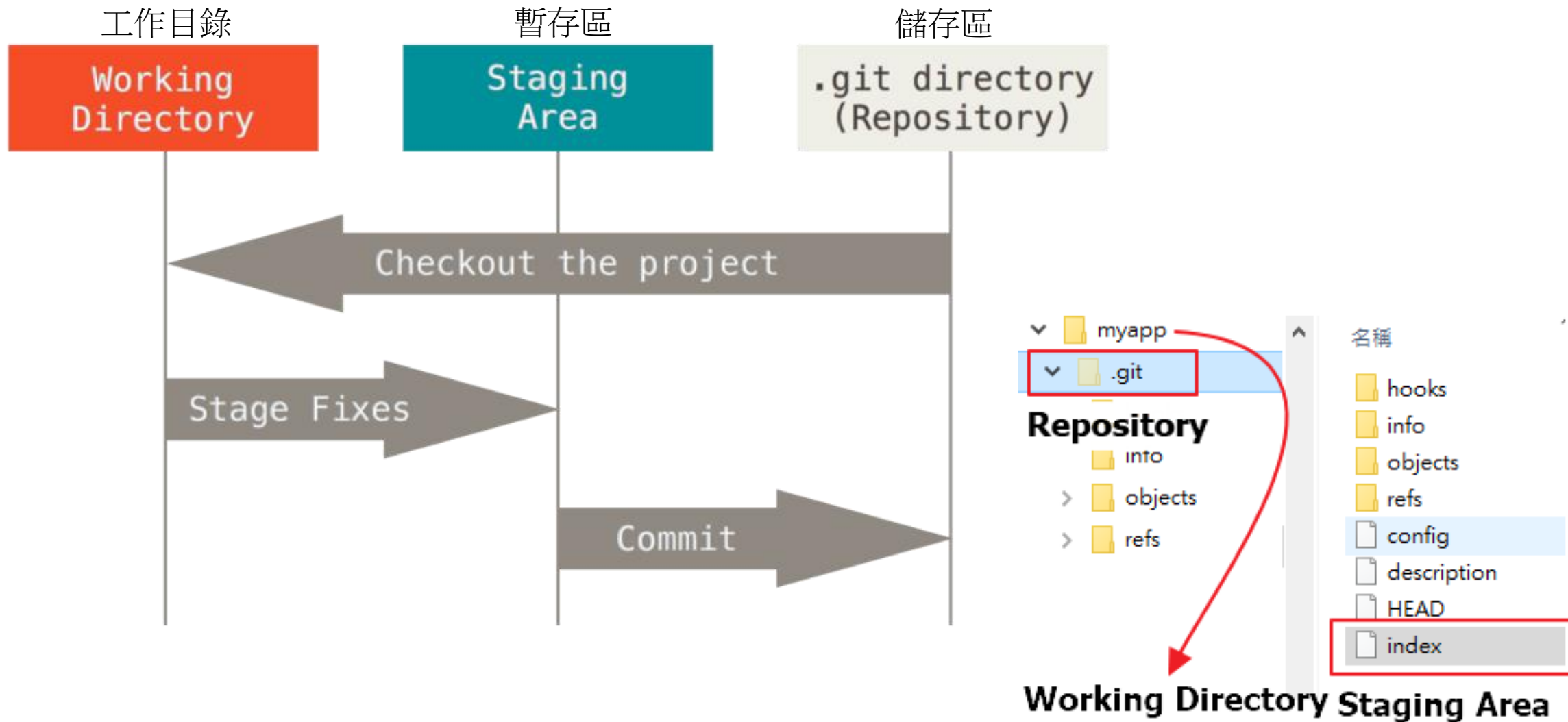
```
git status
```

```
E:\myapp>git status  
On branch master
```

```
No commits yet
```

```
nothing to commit (create/copy files and use "git add" to track)
```

# Git 主要工作區

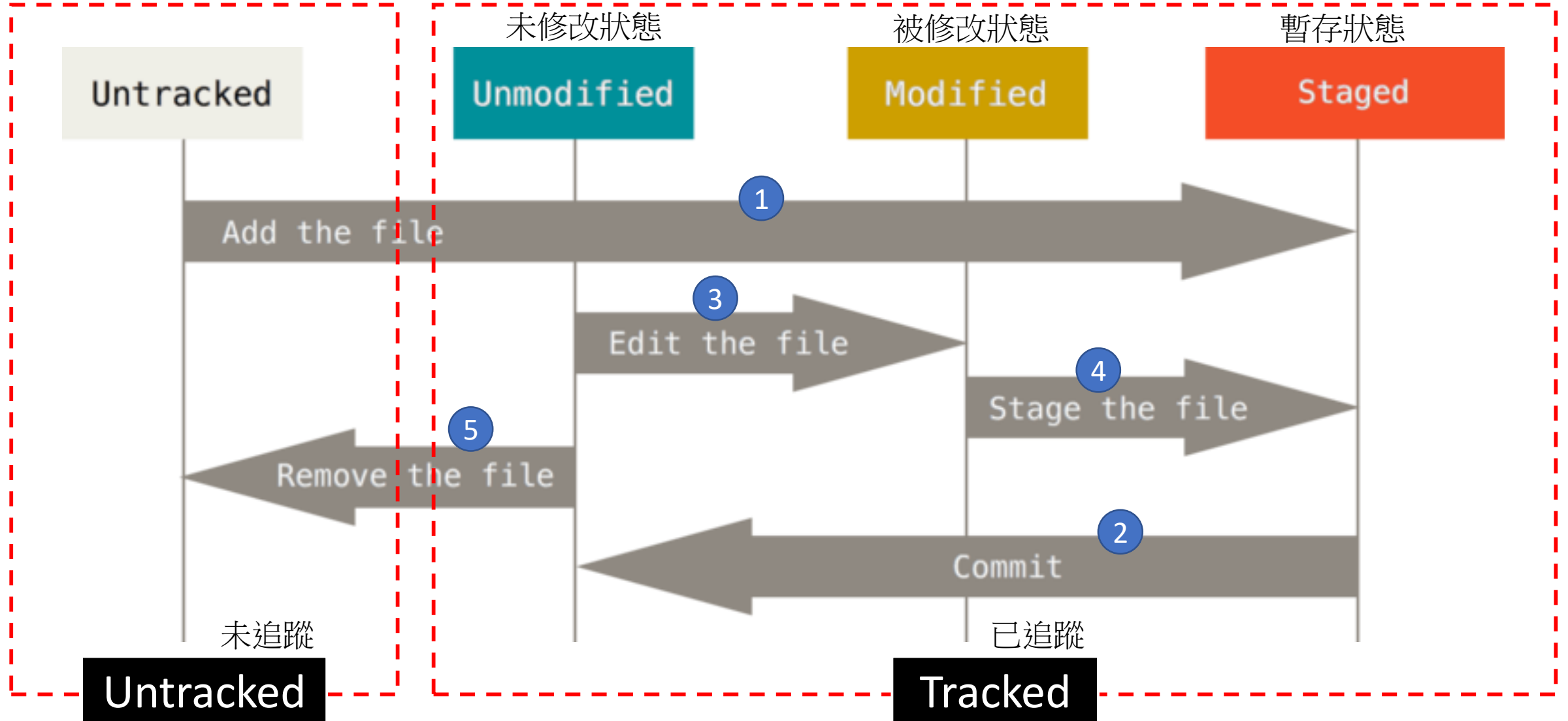


圖片來自：<https://git-scm.com/book/en/v2/Getting-Started-Git-Basics>

# 工作目錄(Working Directory)中的檔案

- 兩種基本狀態
  - tracked：已經記錄在git repository中的檔案，其中又可以分成三種狀態
    - Unmodified：commit之後，還沒修改過的檔案
    - modified：commit之後，又修改過的檔案
    - staged：加到staging area的檔案
  - untracked：還沒有納入git控管的檔案

# 檔案狀態的生命週期



圖片來自：<https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository>

# 顯示檔案狀態

- 加入一個檔案(page1.txt)到工作目錄中，然後顯示工作目錄中檔案的狀態

```
git status
```

```
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    page1.txt

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

# 將檔案交給Git管理，進入到暫存區 (Staging Area)

- untracked的檔案，可以透過git add指令，將檔案加到staging area，讓Git來管理
- 單一檔案

```
git add page1.txt
```

- 副檔名是.txt的所有檔案

```
git add *.txt
```

- 所有檔案

```
git add -all  
git add .
```

# git add

```
$ git status
On branch main

No commits yet


Untracked files:
  (use "git add <file>..." to include in what will be committed)
    page1.txt

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

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git add page1.txt
wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   page1.txt
```




# 從暫存區(Staging Area)回到工作區 (Working Directory)

```
git rm --cached page1.txt
```

```
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   page1.txt
```

```
wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)  
$ git rm --cached page1.txt  
rm 'page1.txt'
```



```
wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)  
$ git status  
On branch main
```

```
No commits yet
```

```
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    page1.txt
```

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




# 從暫存區(Staging Area)進入到備份區(Repository)

- staging area中的檔案，可以透過commit指令，將檔案確認存到透過Repository

```
git commit -m "init commit"
```

- -m "init commit" 是說明這次的commit做了甚麼事



```
$ git commit -m "init commit"  
[main (root-commit) 137750e] init commit  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 page1.txt  
  
wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)  
$ git status  
On branch main  
nothing to commit, working tree clean
```

# 將修改追加到最近一次commit

- 修改最近一次commit的message

```
git commit --amend -m "修改的message"
```

- 將要備份的檔案加到最近一次的commit

```
git commit --amend --no-edit
```

```
$ git commit --amend --no-edit  
[main 0432ca1] add some text  
Date: Tue Mar 15 23:02:08 2022 +0800  
1 file changed, 3 insertions(+), 1 deletion(-)
```

# 修改檔案

- 將剛剛備份完成的檔案拿來修改，再透過git status查詢檔案的狀態

```
$ 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:   page1.txt

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

- 將修改的檔案備份，就重新再做一次 git add、git commit，這兩個動作可以合併成一次
  - git commit -am "update content"

# 檢視commit紀錄

- 使用log指令，會由新到舊的顯示，是誰、什麼時候做了什麼事

git log

git log --oneline [--graph]

```
$ git log
commit 96916b70ac7fcf342aaa4db52f7f142c3a96e364 (HEAD -> main)
Author: hhwang <wang@ispan.com.tw>
Date: Thu Feb 17 14:49:11 2022 +0800

    add aaaaaaa

commit 137750e86bf6c0087681b465bbc55499b1976d98
Author: hhwang <wang@ispan.com.tw>
Date: Wed Feb 16 23:11:14 2022 +0800

    init commit
```

```
$ git log --oneline
96916b7 (HEAD -> main) add aaaaaaa
137750e init commit
```

- 退出 log 狀態，輸入 :q

# ignore file

- 紀錄不想被Git管理的檔案
- 在工作目錄中放一個 .gitignore 檔案

```
.gitignore x
1  #忽略log目錄下的所有檔案
2  log/
3
4  #忽略config.json檔案
5  config.json
6
7  #忽略所有附檔名是.tmp的檔案
8  *.tmp
```

- Github提供了.gitignore檔案的範例
  - <https://github.com/github/gitignore>
  - <https://www.toptal.com/developers/gitignore>

# 還原修改 git restore - 情境1

- 檔案修改後，還沒有新增到暫存區(Staging Area)，要復原檔案中的修改

```
git restore <file>
```

```
$ 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:   page1.txt

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

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git restore page1.txt

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git status
On branch main
nothing to commit, working tree clean
```

# 還原修改 git restore - 情境2

- 檔案修改後，已經新增到暫存區(Staging Area)，要退回到工作目錄(Working Directory)

```
git restore --staged <file>
```

```
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   page1.txt

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git restore --staged page1.txt

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (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:   page1.txt

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

# 切換到之前的版本 - 情境1

- 工作目錄(Working Directory )中的內容，暫存區(Staging Area)以及儲存庫(Repository)的內容回到指定的版本
- 指定版本的幾種寫法

```
git reset --hard HEAD~~~~
```

```
git reset --hard HEAD~5
```

```
git reset --hard HEAD^^
```

```
git reset --hard <commit>
```



# Reset 後的還原

- 列出所有的歷史紀錄(移動版本的所有紀錄)

```
git reflog
```

```
6179f5f HEAD@{15}: reset: moving to HEAD~  
4509c3e (HEAD, main) HEAD@{16}: reset: moving to 4509c3e  
6179f5f HEAD@{17}: reset: moving to HEAD~  
4509c3e (HEAD, main) HEAD@{18}: commit: add bbbbbb  
6179f5f HEAD@{19}: commit: add Page4.txt  
ab1602c HEAD@{20}: commit: add some content  
2c37cfb (origin/main, feature-login) HEAD@{21}: checkout:   
git to main
```

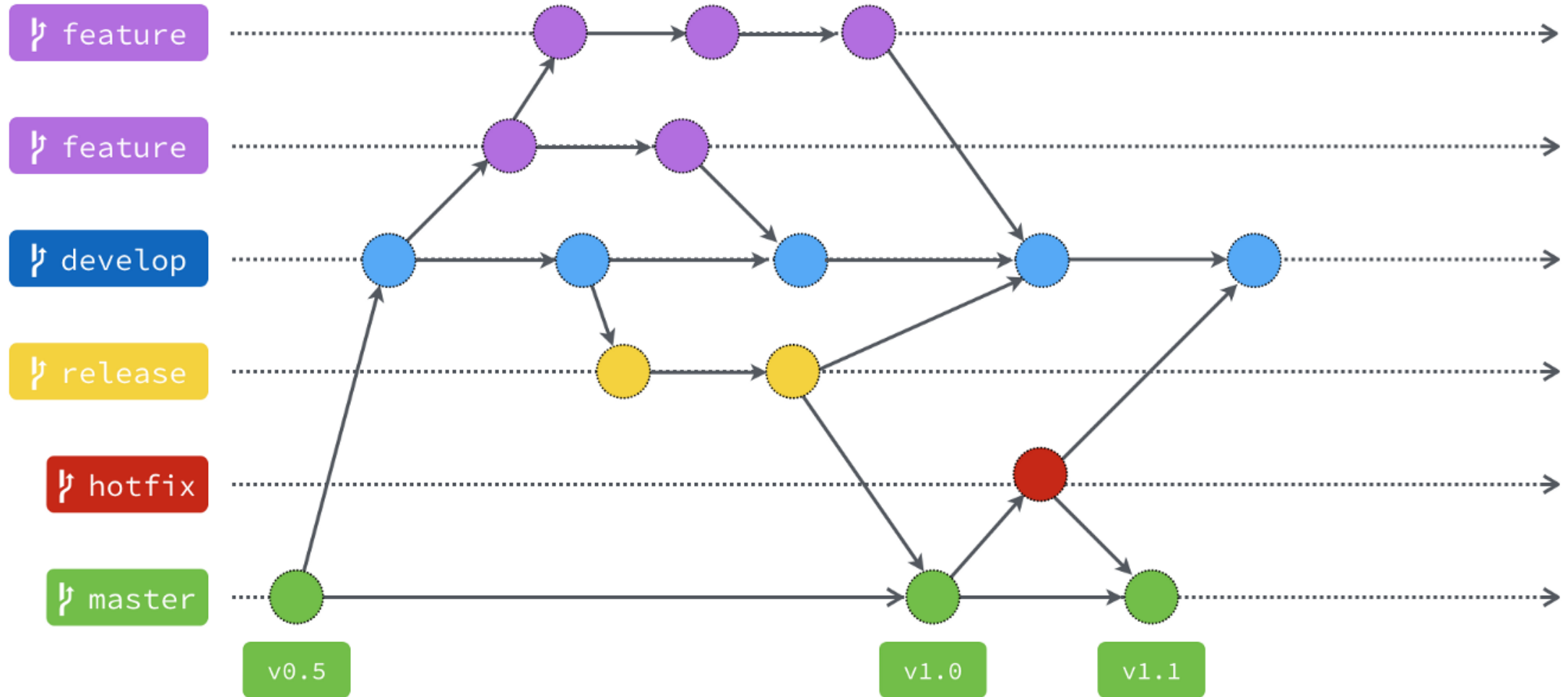
- 回歸原始版本

```
git reset --hard HEAD@{8}
```

# branch 分支

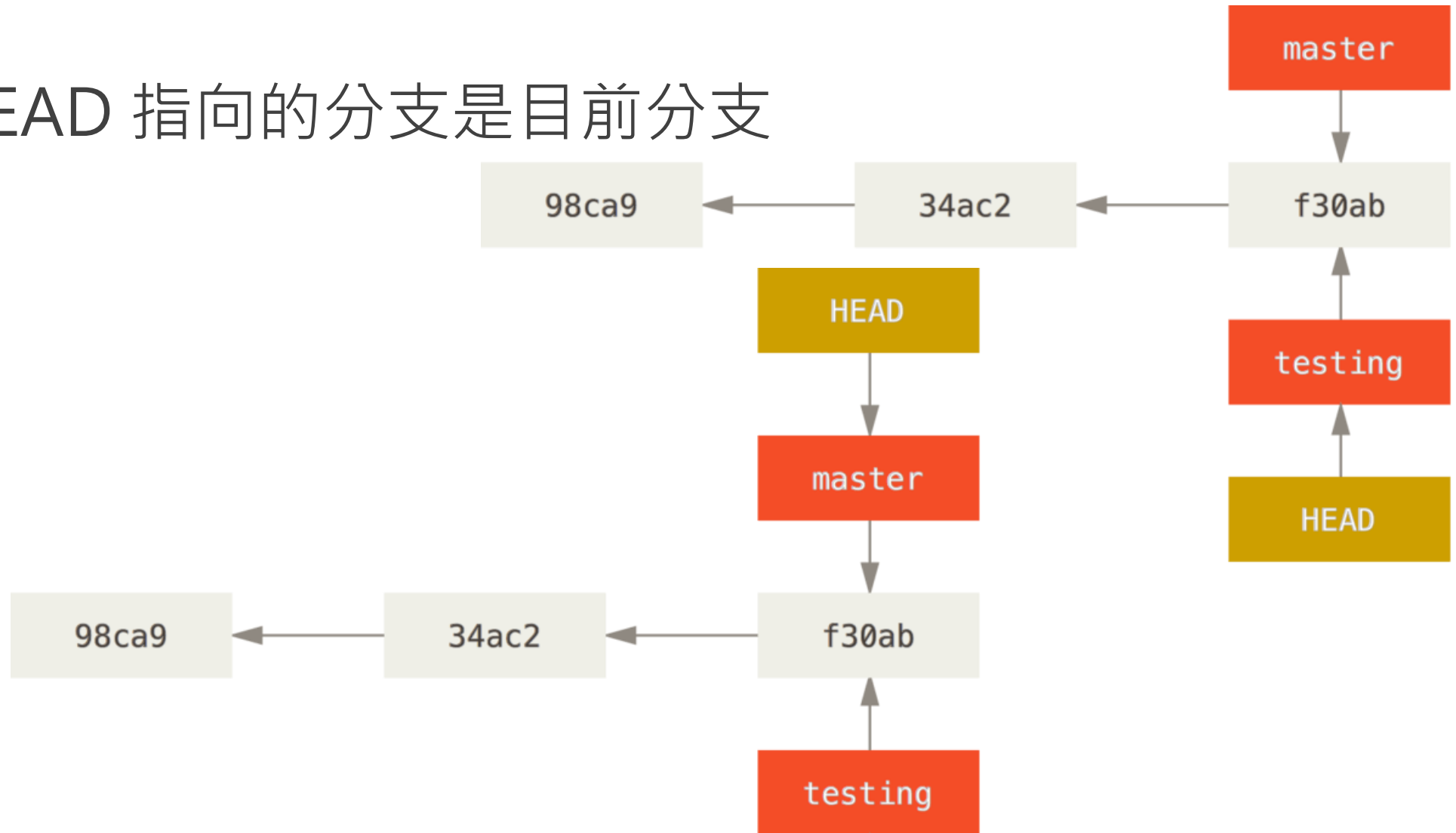
- 將程式開發從開發主線上分離開來
  - 不同的版本就使用不同的分支，ex：1.0、1.0.1、1.1、2.0、....
  - 不同的軟體版本週期使用不同的分支，ex：Alpha、Beta、RC、RTM、.....
  - 單一功能的開發使用不同的分支
  - 不同的開發人員使用不同的分支
  - 為了修復問題也可以使用不同的分支
- 預設的分支名稱叫做？以前是master，現在是main
- 透過HEAD指標，指定目前使用中的branch

# Git Flow



# 特別指標 HEAD

- 被 HEAD 指向的分支是目前分支



# 新增分支

- 查詢目前專案有哪些分支

```
git branch
```

- 新增分支

```
git branch <branch分支名稱>
```

- 切換分支

```
git checkout <branch分支名稱>
```

- 同時建立及切換分支

```
git checkout -b <branch分支名稱>
```

```
$ git branch
* main

wang0804@980517-NB3-DEI MINGW64 ~/Docu
$ git branch
* main

wang0804@980517-NB3-DEI MINGW64 ~/Docu
$ git branch dev

wang0804@980517-NB3-DEI MINGW64 ~/Docu
$ git branch
  dev
* main

wang0804@980517-NB3-DEI MINGW64 ~/Docu
$ git checkout dev
Switched to branch 'dev'

wang0804@980517-NB3-DEI MINGW64 ~/Docu
$ git branch
* dev
  main
```

# 切換分支的另一種寫法(新)

- 切換分支

`git switch <branch分支名稱>`

```
wang0804@980517-NB3-DEI MINGW64 ~/documents/workspace/gitwork (release)
$ git switch main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)
```

- 同時建立及切換分支

`git switch -c <branch分支名稱>`

```
wang0804@980517-NB3-DEI MINGW64 ~/documents/workspace/gitwork (main)
$ git switch -c feature1
Switched to a new branch 'feature1'
```

# 修改分支名稱

`git branch -m <舊的分支名稱> <新的分支名稱>`

```
$ git branch
* dev
  main

wang0804@980517-NB3-DEI MINGW64
$ git branch -m dev develop

wang0804@980517-NB3-DEI MINGW64
$ git branch
* develop
  main
```

# 刪除分支

- 不能刪除現在使用的分支，因此要先切換到別的分支

`git branch -d(-D) <分支名稱>`

```
$ git branch
develop
* main

wang0804@980517-NB3-DEI MINGW64 ~/Docume
$ git branch -d develop
Deleted branch develop (was a6a3b72).

wang0804@980517-NB3-DEI MINGW64 ~/Docume
$ git branch
* main
```



# 合併分支

- 如果要用main分支合併其它分支，就要先切換到main分支

```
git merge <分支名稱> -m "message"
```

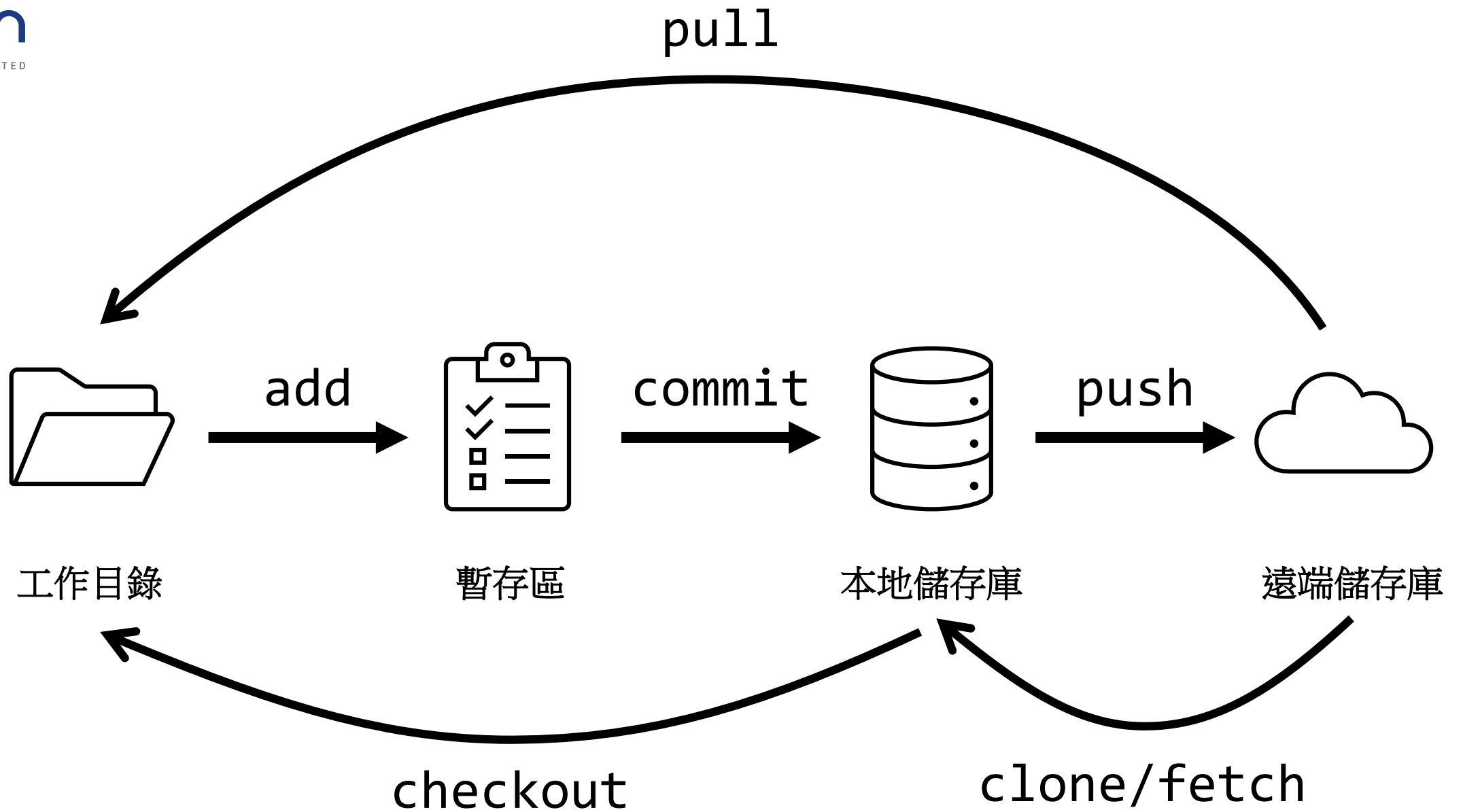
```
$ git merge feature -m "daily merge"  
Merge made by the 'ort' strategy.  
feature2.txt | 2 ++  
1 file changed, 2 insertions(+)
```

# Git remote repository

# Git Remote Repository

- Git Repository
  - Local Repository：之前的用法就是local repository
  - Remote Repository：讓開發團隊成員分享各自的local repository資料而建立
- 我們可以自行建置 Git Server或使用坊間的Git Repository託管服務
  - GitHub、GitLab、Bitbucket、Gitorious

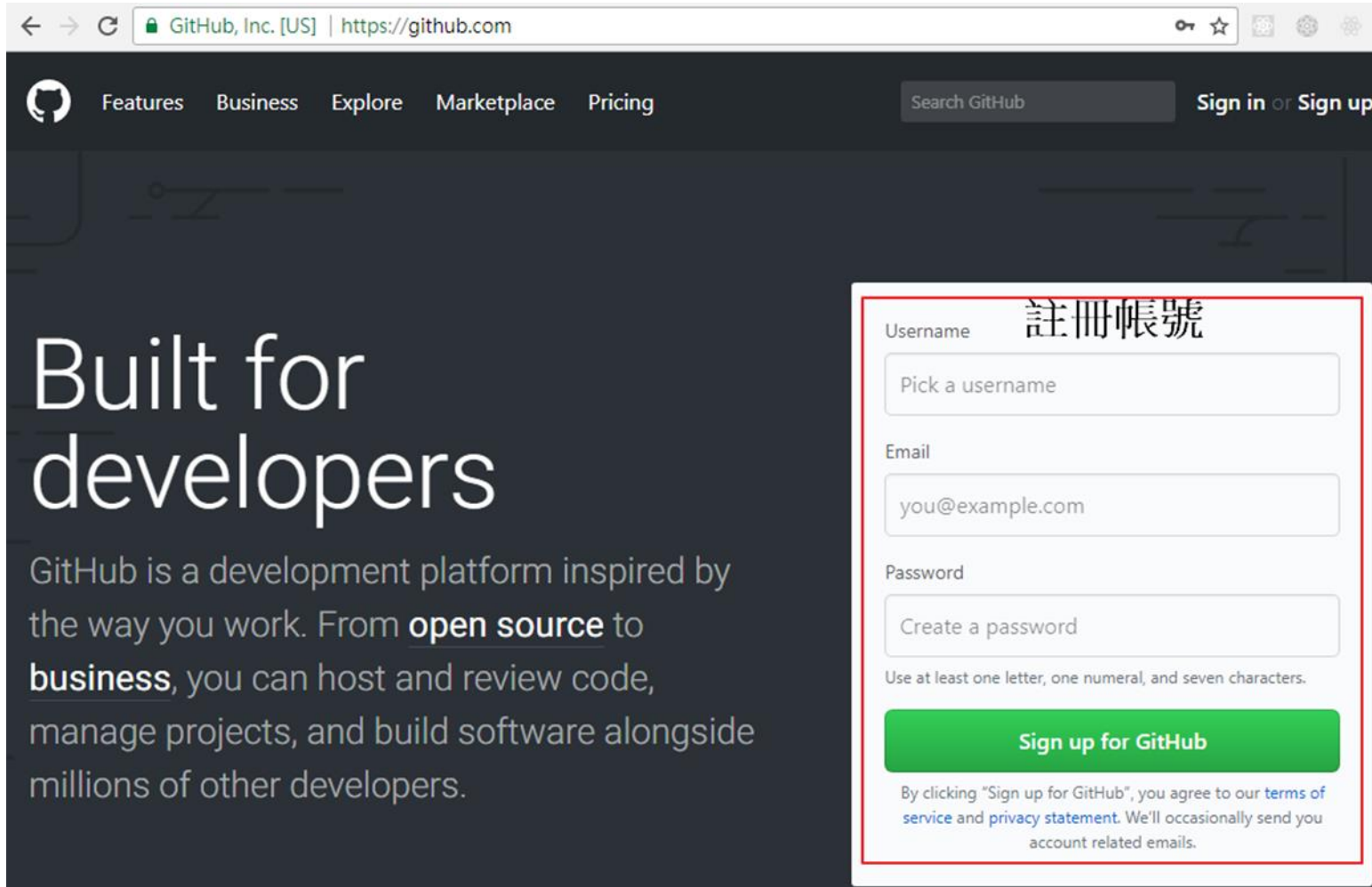




# GitHub

- 申請GitHub帳號

<https://github.com/>



The screenshot shows the GitHub homepage with a dark theme. The navigation bar includes links for Features, Business, Explore, Marketplace, and Pricing, along with a search bar and 'Sign in or Sign up' buttons. The main content area features the text 'Built for developers' and a description of GitHub as a development platform. A sign-up form is overlaid on the right side, containing fields for Username, Email, and Password, a 'Sign up for GitHub' button, and a disclaimer about terms of service and privacy.

← → ↻ GitHub, Inc. [US] | <https://github.com> 🔑 ☆ ⚙️

Features Business Explore Marketplace Pricing Search GitHub Sign in or Sign up

## Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside millions of other developers.

**註冊帳號**

Username  
Pick a username

Email  
you@example.com

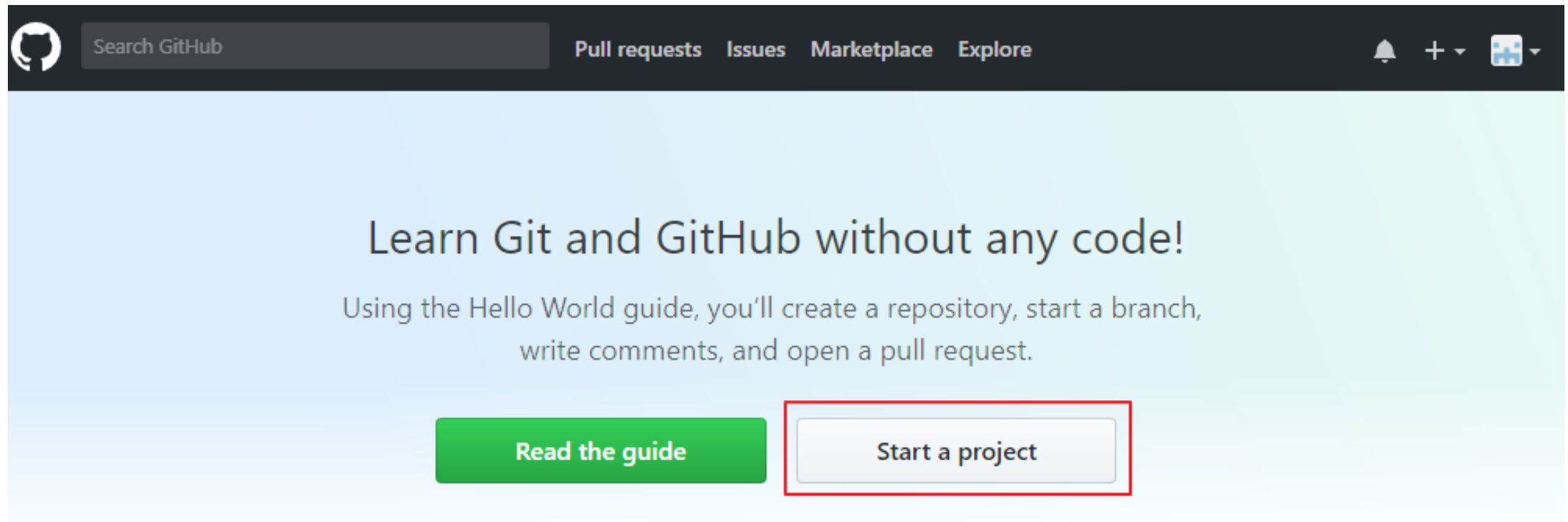
Password  
Create a password  
Use at least one letter, one numeral, and seven characters.

**Sign up for GitHub**

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.

# 註冊完後

- 按下Start a project後，還會有一個驗證email的步驟，接下來才能建立專案[Start a project]




# 建立新專案

## Create a new repository

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

Owner

 iiieduwang ▾

 /

Repository name

hello-world ✓

Description (optional)

First GitHub Repository

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

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

☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

 | 



Add a license: **None** ▾



Create repository

# 完整的使用方式說明

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

 Set up in Desktop or **HTTPS** **SSH** `https://github.com/iiieduwang/hello-world.git` 

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


## ...or create a new repository on the command line

```
echo "# hello-world" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/iiieduwang/hello-world.git
git push -u origin master
```



## ...or push an existing repository from the command line

```
git remote add origin https://github.com/iiieduwang/hello-world.git
git push -u origin master
```



## ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)



# 從local連到remote 1

- 在Git環境加入remote repository的設定

```
git remote add <自取的名稱> <remote repository 的URL>
```

- remote repository url 格式
  - https://github.com/<GitHub帳號>/<GitHub上的Repository名稱>.git
  - 檢視所有remote repository的設定

```
git remote -v
```

- 檢視某個remote repository的設定

```
git remote show <名稱>
```

# 範例參考

```
E:\myapp>git remote add origin https://github.com/iiieduwang/hello-world.git

E:\myapp>git remote -v
origin https://github.com/iiieduwang/hello-world.git (fetch)
origin https://github.com/iiieduwang/hello-world.git (push)

E:\myapp>git remote show origin
* remote origin
Fetch URL: https://github.com/iiieduwang/hello-world.git
Push URL: https://github.com/iiieduwang/hello-world.git
HEAD branch: master
Remote branch:
  master new (next fetch will store in remotes/origin)
Local ref configured for 'git push':
  master pushes to master (local out of date)
```

# 從local連到remote 2

- 改變remote repository設定的名稱
- 檢視所有remote repository的設定

```
git remote rename <舊的名稱> <新的名稱>
```

- 刪除remote repository的設定

```
git remote remove <設定的名稱>
```

# 上傳 下載資料

- local repository的資料**上傳**到remote repository

```
git push <設定的名稱> <branch的名稱>  
git push --all origin
```

- Remote repository的資料**下載**到local repository

```
git pull <設定的名稱> <branch的名稱>
```

# git push 範例參考

```
E:\fromgit>git remote add origin https://github.com/iiieduwang/fromlocal.git  
  
E:\fromgit>git push origin master  
Fatal: HttpRequestException encountered.  
Username for 'https://github.com': iiieduwang  
Password for 'https://iiieduwang@github.com':  
Counting objects: 14, done.  
Delta compression using up to 8 threads.  
Compressing objects: 100% (10/10), done.  
Writing objects: 100% (14/14), 1.21 KiB | 0 bytes/s, done.  
Total 14 (delta 1), reused 0 (delta 0)  
remote: Resolving deltas: 100% (1/1), done.  
To https://github.com/iiieduwang/fromlocal.git  
* [new branch]      master -> master
```

# git pull 範例參考

```
E:\fromgit>git remote add origin https://github.com/iiieduwang/hello-world.git
```

```
E:\fromgit>git remote -v  
origin https://github.com/iiieduwang/hello-world.git (fetch)  
origin https://github.com/iiieduwang/hello-world.git (push)
```

```
E:\fromgit>git pull origin master  
remote: Counting objects: 14, done.  
remote: Compressing objects: 100% (9/9), done.  
remote: Total 14 (delta 1), reused 14 (delta 1), pack-reused 0  
Unpacking objects: 100% (14/14), done.  
From https://github.com/iiieduwang/hello-world  
* branch                master      -> FETCH_HEAD  
* [new branch]          master      -> origin/master
```

# DOS and Linux Commands

- 完整的指令參考
  - [https://ftp.kh.edu.tw/Linux/Redhat/en\\_6.2/doc/gsg/ch-doslinux.htm](https://ftp.kh.edu.tw/Linux/Redhat/en_6.2/doc/gsg/ch-doslinux.htm)
- 常用且不同的指令

	Windows	Linux
列出目錄中的內容	dir	ls
清除畫面上的內容	cls	clear
顯示目前所在目錄的路徑	cd	pwd

# Vim 文字編輯器

- 使用說明參考
  - <https://blog.techbridge.cc/2020/04/06/how-to-use-vim-as-an-editor-tutorial/>
- 不小心進入輸入 :q 離開



## 疑難雜症

當我們輸入 `git status` 的時候，如果檔名或路徑中有中文字，而且畫面顯示亂碼

- `git config --global core.quotePath false`。

```
User@DESKTOP-LBN7FI8 MINGW64 /d/OneDrive - ispan.com.tw/桌面/test (main)
$ git st
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    新文字文件.txt
```

# GIT 近期消息

2024/5/21

Git修補5個漏洞，其中包含子模組複製儲存庫產生的RCE漏洞

<https://www.ithome.com.tw/news/163000>

iThome
新聞
產品&技術
專題
AI
Cloud
醫療IT
資安
研討會
社群
IT EXPLAINED
Q搜尋

## Git修補5個漏洞，其中包含子模組複製儲存庫產生的RCE漏洞

軟體協同開發不可或缺的工具Git，在5月中發布資安漏洞修補的改版，揭露儲存庫複製過程的問題

文/ 李宗翰 | 2024-05-21 發表

讚 115 分享

Product
Solutions
Open Source
Pricing

Search or jump to...

git / git
Public

Code
Pull requests 175
Actions
Security 26
Insights

### Recursive clones on case-insensitive filesystems that support symlinks are susceptible to Remote Code Execution

Critical dscho published GHSA-8h77-4q3w-gfgv last week

Package	Affected versions	Patched versions	Severity
git	v2.45.0 v2.44.0 <=v2.43.3 <=v2.42.1 v2.41.0 <=v2.40.1 <=v2.39.3	v2.45.1 v2.44.1 v2.43.4 v2.42.2 v2.41.1 v2.40.2 v2.39.4	Critical 9.1 / 10

#### Description

Repositories with submodules can be crafted in a way that exploits a bug in Git whereby it can be fooled into writing files not into the submodule's worktree but into a `.git/` directory. This allows writing a hook that will be executed while the clone operation is still running, giving the user no opportunity to inspect the code that is being executed.

#### Impact

CVSS base metrics
Attack vector
Attack complexity
Privileges required
User interaction
Scope
Confidentiality
Integrity
Availability

Network
High
None
None
Changed
High
High
High

廣大軟體開發者所仰賴的分散式版本控制系統Git，5月14日發布2.45.1版，之所以改版是為了修正5個資安漏洞，包含被標示為重大資安漏洞CVE-2024-32002，高風險的資安漏洞CVE-2024-32004與CVE-2024-32465，以及低度危險的資安漏洞CVE-2024-32020與CVE-2024-32021。用戶應將Git的版本升級至2.45.1、2.44.1、2.43.4、2.42.2、2.41.1、v2.40.2、v2.39.4。Git目前可安裝在多種作業

「企業混合雲實戰攻略三策」Data services & ML with Azure Hybrid Solution、  
「企業混合雲實戰攻略三策」Microsoft Azure Stack HCI 在 HPE 伺服器上的整合優勢  
IT EXPLAINED | 41 分

2021 Q4 Progress NMS 網路管理軟體  
WhatsUp Gold 進階培訓課程 (2)  
EC NETWORKER | 114 分

資安防禦零死角！勒索軟體說掰掰！  
IT EXPLAINED | 42 分

Implement Engineering  
Operation Dashboard

# markdown 語法

<https://markdown.tw/>

Col1	Col2
Item1	3000
Item2	1000

```
<style>  
  #red{color:red}  
</style>  
<h1 id="red">這是紅色的字</h1>
```

這是紅色的字

Thank you