

Git 版本控制

/ 講者: 國立高雄師範大學軟體工程與管理學系 余信陞



Outline

- 什麼是Git ?
- 安裝與設定
- 基本指令與觀念
- GitHub
- 遠端儲存庫
- 衝突與分支
- 多人協作

什麼是Git？

/ Git 版本控制

Introduction

- 由Linux核心的作者 Linus Torvalds 在開發而成，最初是為了更好地管理於核心開發所設計
- 版本控制系統 (Version Control System)
- 玩遊戲的儲存進度
- 一本記錄歷史書籍

Advantage

- 記錄各版本的差異
- 切換版本
- 利用分支同時開發不同功能
- 合併檔案

Record the Differences – no git

名稱	修改日期	類型	大小
 1211會議紀錄 - 修改版1.txt	2021/12/9 下午 01:30	文字文件	0 KB
 1211會議紀錄 - 修改版2.txt	2021/12/9 下午 01:32	文字文件	1 KB
 1211會議紀錄.txt	2021/12/9 下午 01:30	文字文件	0 KB

Record the Differences

Project

Feature A

Feature A
Feature B

Feature A
Feature B
Testing

Version

User : Shark
Date : 2002/10/22
Msg : initial project

User : Shark
Date : 2002/10/24
Msg : add Feature B

User : Shark
Date : 2002/10/24
Msg : Testing

Record the Differences

▼ 2 Database/migrations/create_users_table.php

@ -1,5 +1,5 @@

1 1 <?php

2 - require_once('../../Models/Model.php');

2 + require_once(__DIR__.'/../../Models/Model.php');

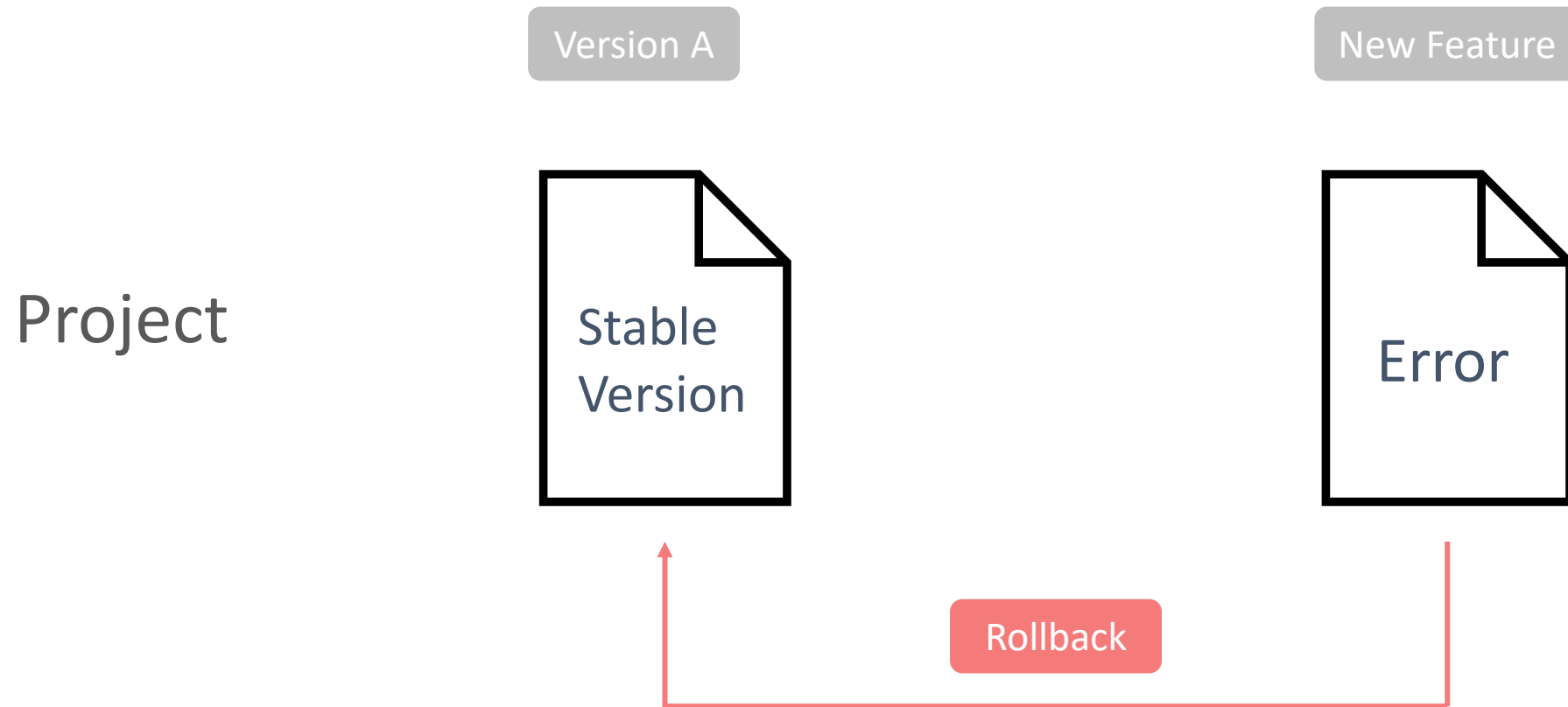
3 3

4 4 class UserMigration extends Model

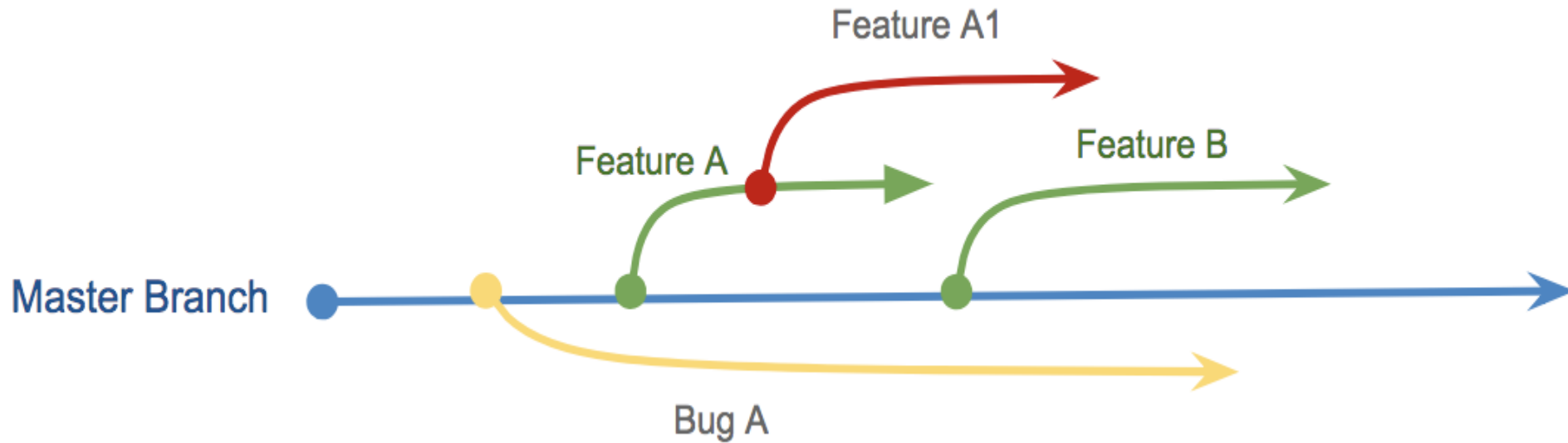
5 5 {

.....
↓

Change Version



Simultaneous Development

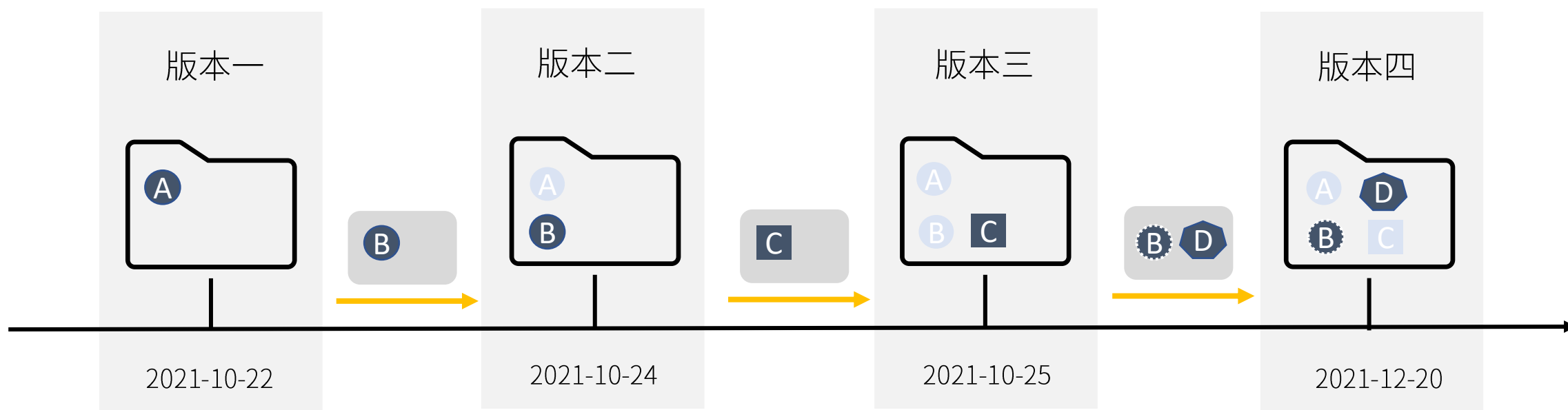


Merge - no git



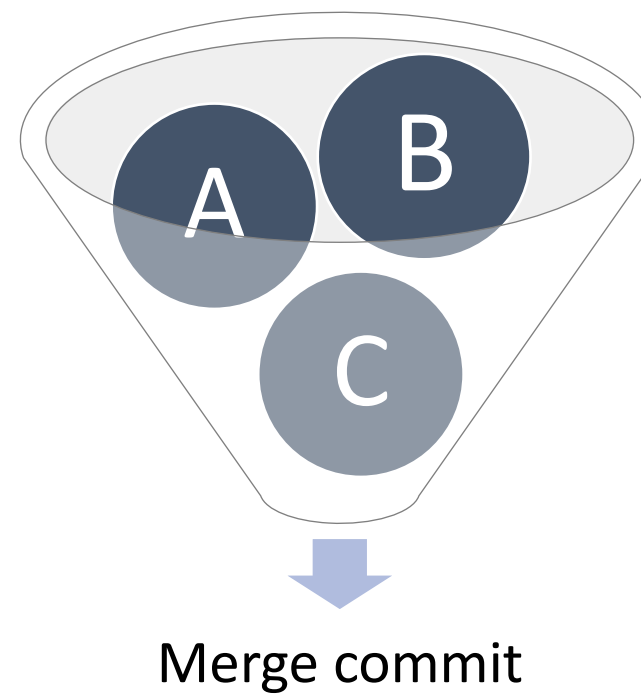
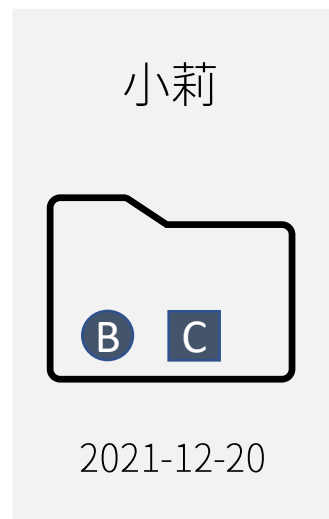
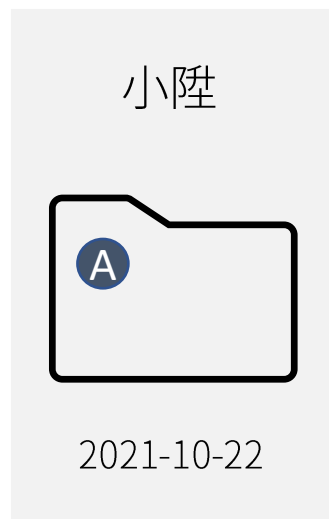
Merge

分散式系統 (Distributed Version Control)



Merge

Merge時可能會發生衝突！(詳情待X回分解)



安裝與設定

/ Git 版本控制

Install Git on Linux

Ubuntu

```
$ sudo apt-get install git
```

```
$ sudo yum install epel-release
```

```
$ sudo yum install  
https://centos7.iuscommunity.org/ius-release.rpm
```

```
$ sudo yum install git2u
```

CentOS 7

Install Git on Windows

- Git 下載連結 <https://git-scm.com/download/win>
- 打開 Git Bash

```
$ git --version  
git version 2.21.0.windows.1
```


Set up

```
$ git config --global user.name "<username>"
```

```
$ git config --global user.email "<email>"
```

```
$ git config --list
```

Alias

```
$ git config --global alias.co checkout
```

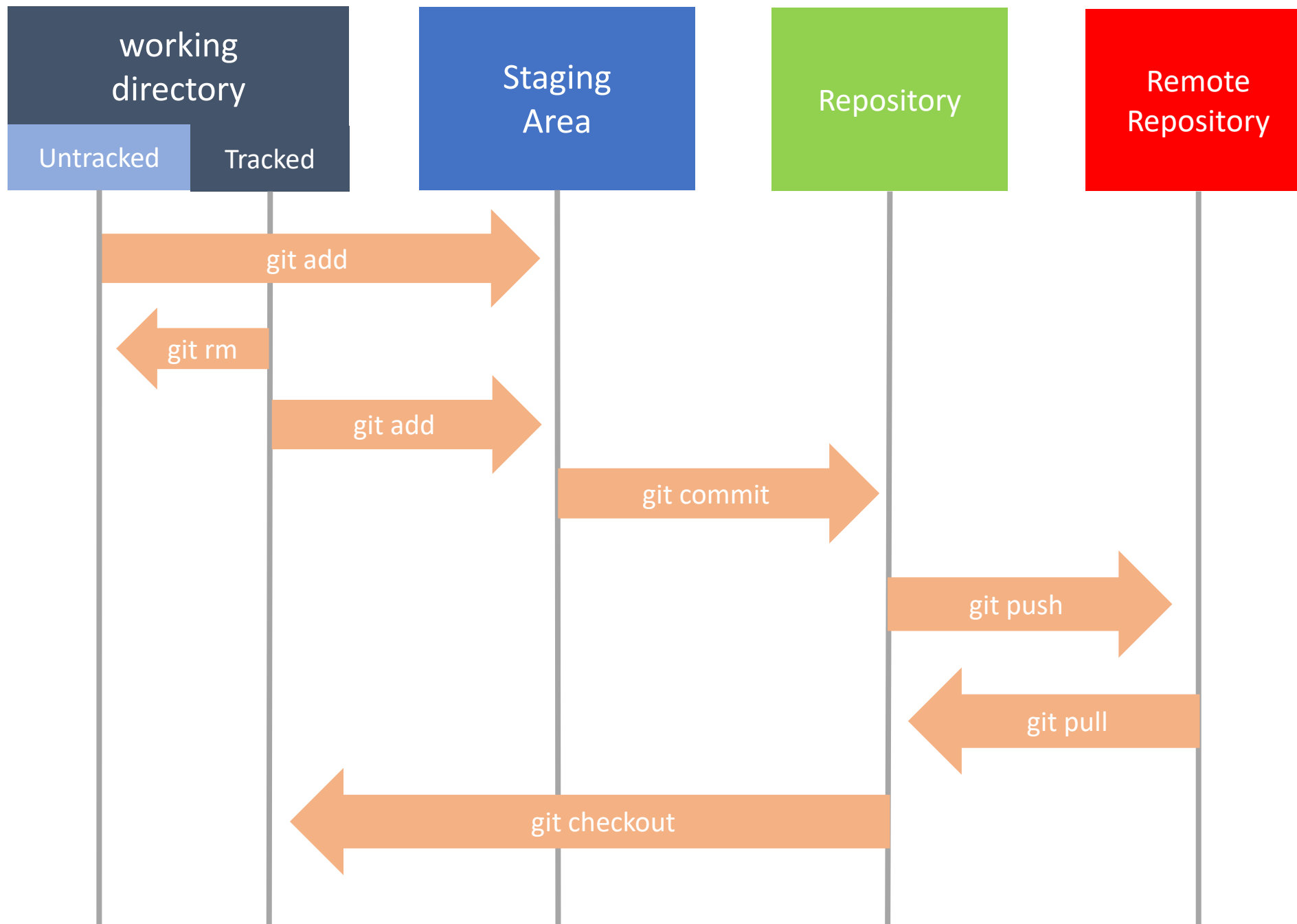
```
$ git config --global alias.br branch
```

```
$ git config --global alias.st status
```

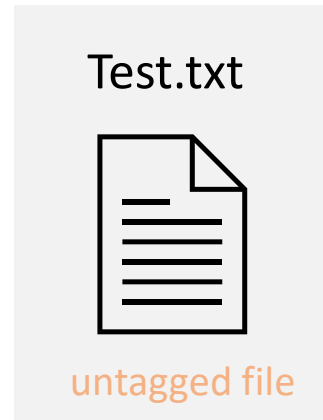
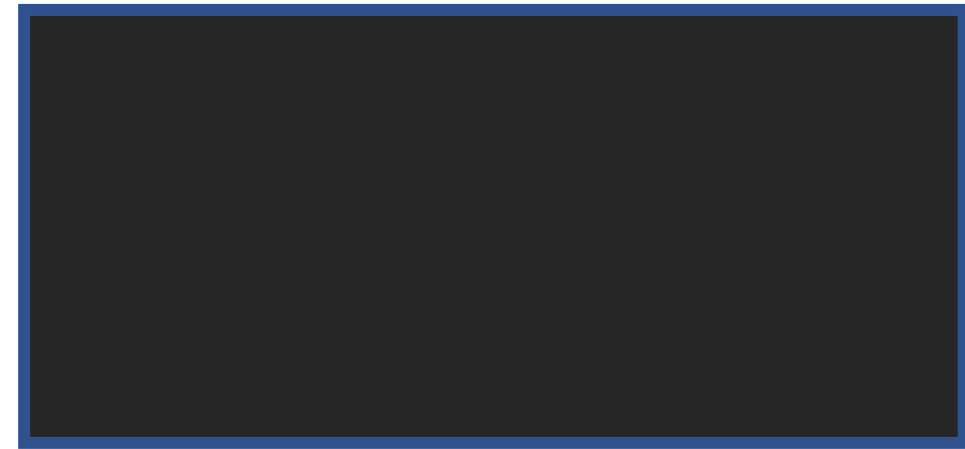
```
$ git config --global alias.l "log --oneline --graph"
```

基本指令與觀念

/ Git 版本控制



flow



working
directory

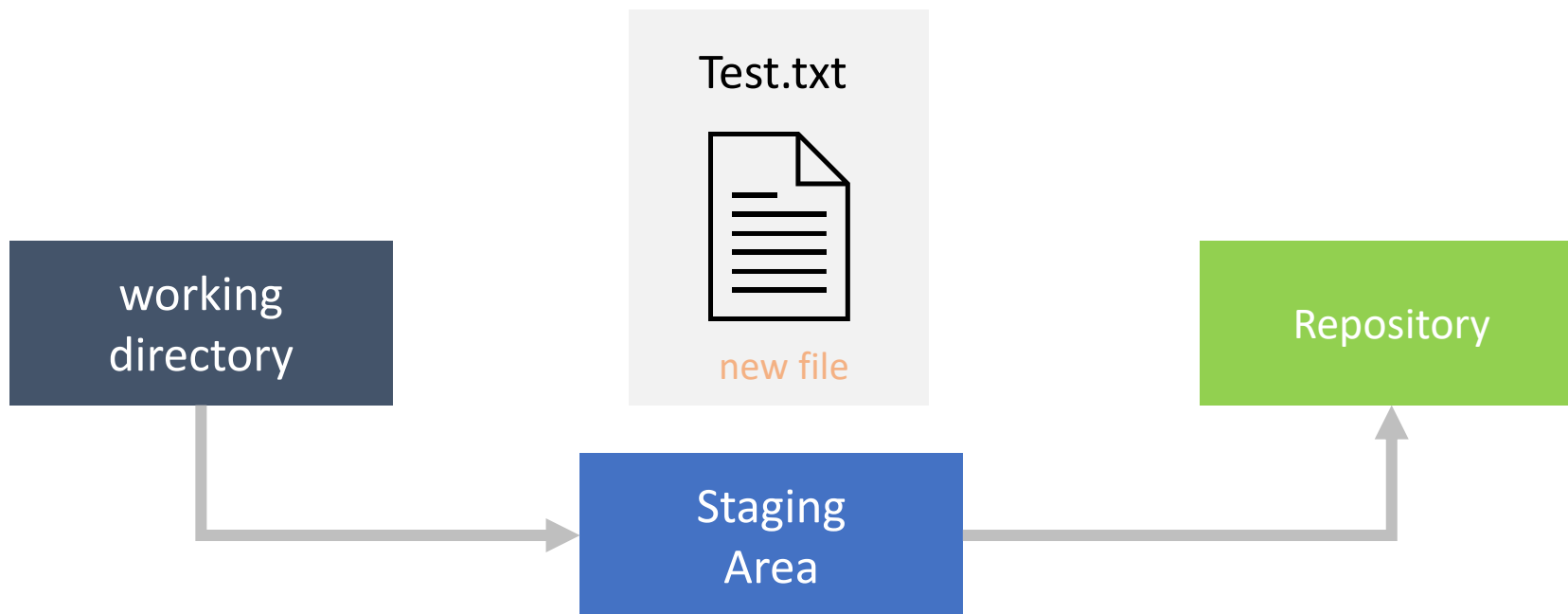
Staging
Area

Repository



flow

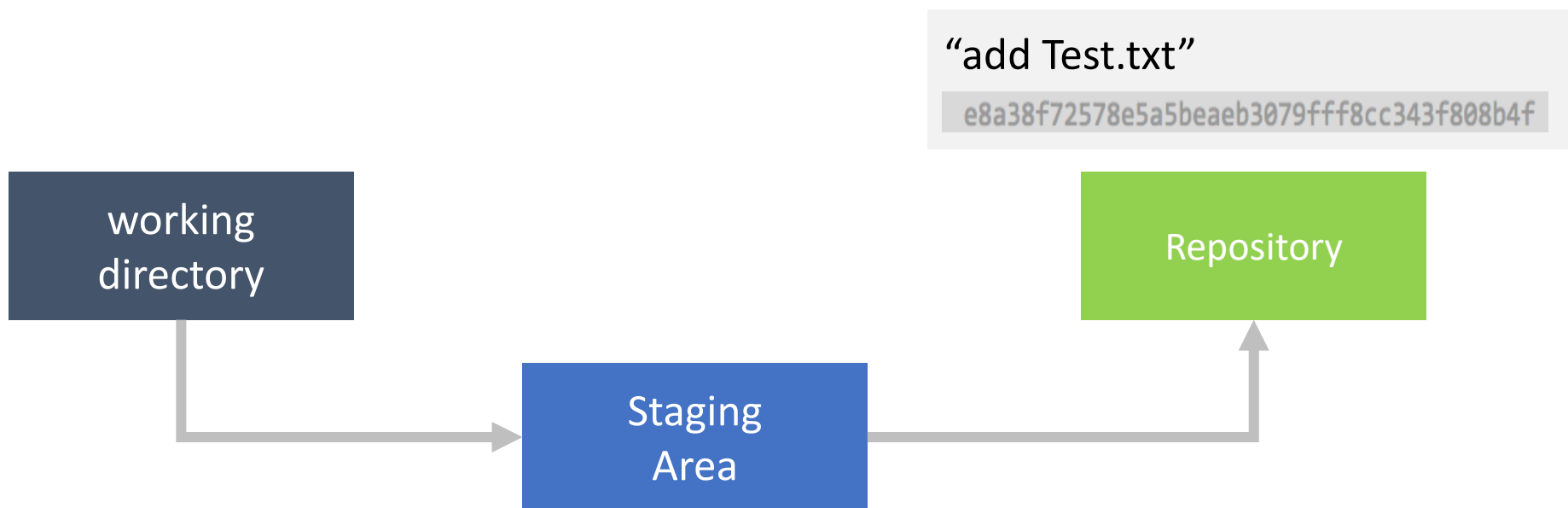
```
$ git add Test.txt
```



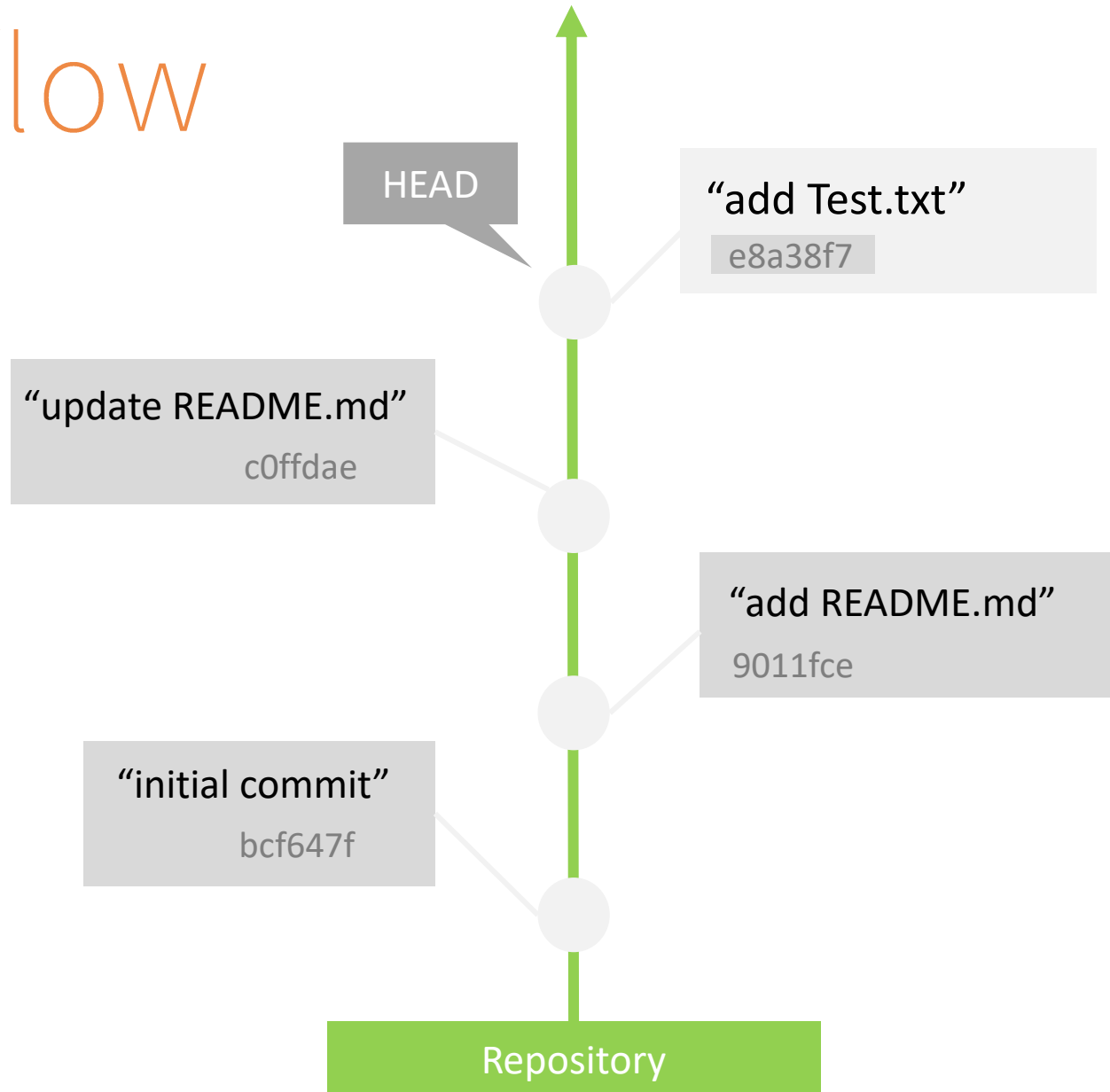
flow

```
$ git add Test.txt
```

```
$ git commit -m 'add Test.txt'
```



flow



```
$ git add Test.txt
```

```
$ git commit -m 'add Test.txt'
```

```
$ git log
```


git init

```
$ git init
```

Initialized empty Git repository in "專案路徑"/.git/

在你的資料夾建立一個 .git 資料夾，任何版本控制項目都會透過它進行監控。

git status

```
$ git status
```

```
On branch master
```

```
nothing to commit, working tree clean
```

查看工作目錄與暫存區目前的狀態

git add

```
$ git add [file]
```

```
$ git add [file] -A
```

-A: 將全部的檔案(Untracked files 或是Changes not staged for commit) 加入暫存區

```
$ git add [file] -u
```

-u: 把已被追蹤(tracked)且修改過的(Modified)檔案加入

git commit

```
$ git commit -m 'message'
```

當一個變更提交時，會計算變更的SHA1值，作為該commit的識別

git log

```
$ git log
```

```
commit b6c3c771cd8939bcd25a8c50089fdf0cd3eab98d  
(HEAD -> master)
```

```
Author: username<email>
```

```
Date: datetime
```

```
message
```

瀏覽 commit 歷史紀錄

git checkout

move HEAD pointer

```
$ git checkout [file]
```

將某個檔案的回到上一個commit的狀態

```
$ git checkout [commit SHA-1]
```

將工作目錄改到某個commit

```
$ git checkout master
```

使工作目錄檔案回到master分支的最新狀態

git reset

```
$ git reset HEAD^
```

拆掉最新一次的commit

```
$ git reset [commit SHA-1]
```

退回到某個commit

Challenge - amend

修改 Commit 紀錄：

小陞在做專案的時候，覺得心很累，於是不小心在最後一次的 commit message 中罵了客戶！後來他才想到當初客戶有要求他繳交開發紀錄，完蛋~

```
$ git commit --amend -m "message"
```


Challenge - reset

追加檔案到最近一次的 Commit

小陞經過不懈的努力，終於完成某部分，發送了一個 Commit，但發送後才發現有一個檔案居然忘了加進去！他又不想為了這個檔案重新再發一次 Commit...

```
$ git reset HEAD^
```

```
$ git add .
```

```
$ git commit -m 'message'
```

Challenge - gitignore

想隱藏一些檔案，不放入Git裡頭

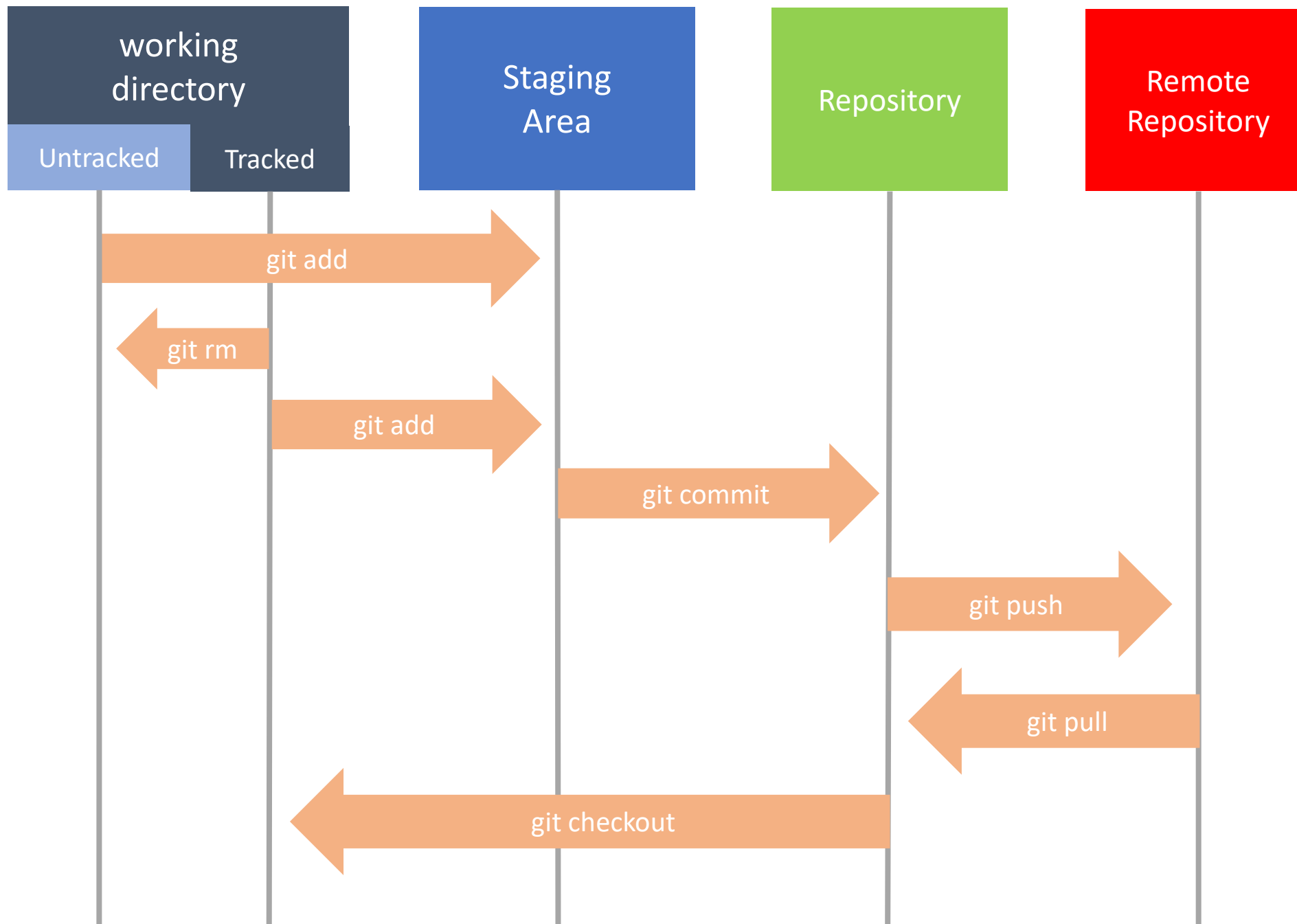
在網頁的專案裡頭通常會有一個環境檔(.env)，裡頭會存著一些設定，比如資料庫的帳號密碼。這些機密的資訊，不應該被放在Git裡頭儲存。

```
$ touch .gitignore
```

```
$ vim .gitignore
```

如果檔案在.gitignore建立之前就已經存在，那就不具備效用

```
$ git rm --cached [file]
```



GitHub

/ Git 版本控制

Sign up

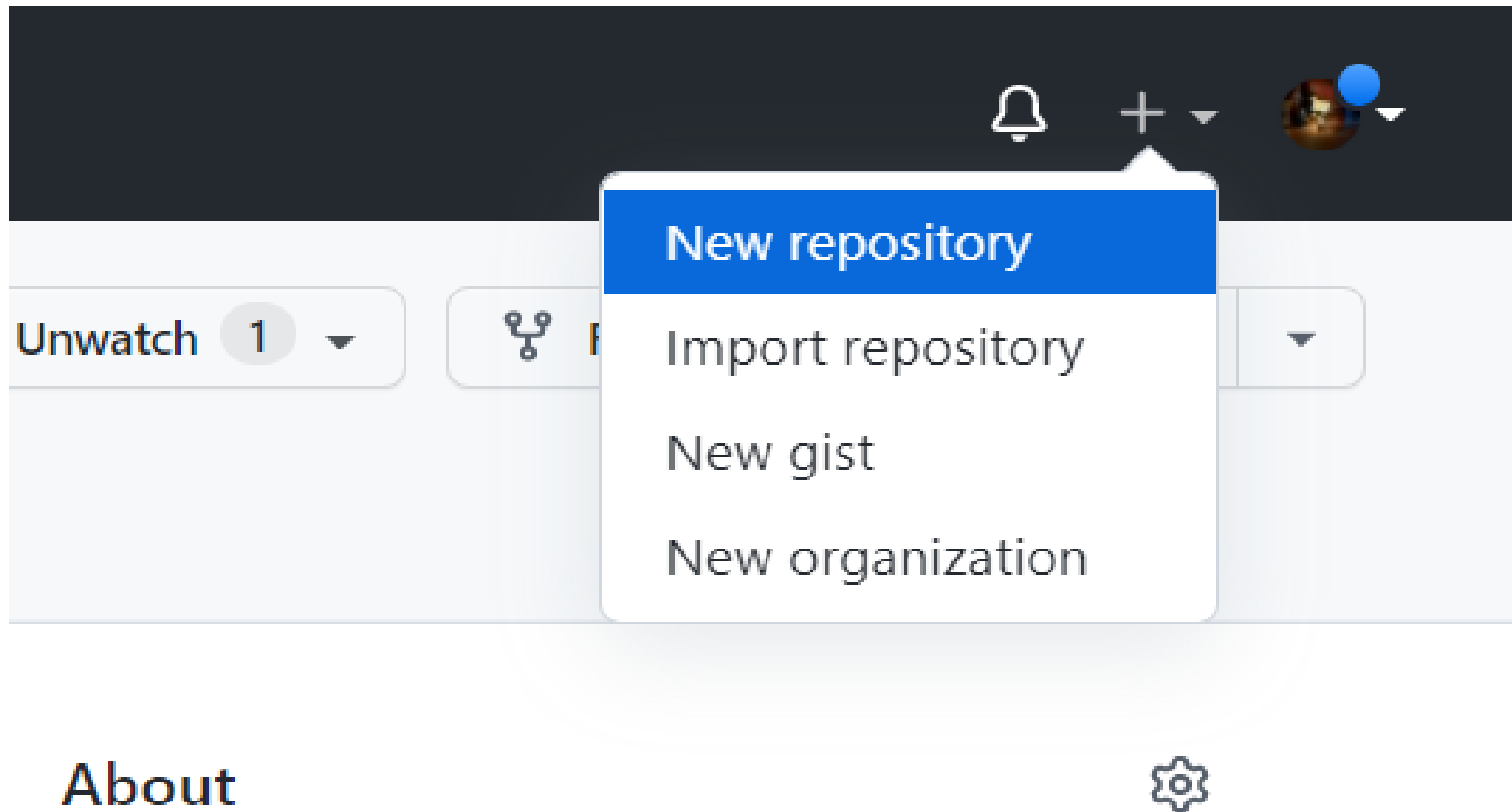
Welcome to GitHub!
Let's begin the adventure

Enter your email




Continue

Establish remote repository



Establish remote repository

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

 Set up in Desktop

or

HTTPS

SSH



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

```
echo "# gitTest" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/sharkfoolish/gitTest.git
git push -u origin main
```



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

```
git remote add origin https://github.com/sharkfoolish/gitTest.git
git branch -M main
git push -u origin main
```



Establish remote repository

```
$ git init
```

```
$ vim README.md
```

```
$ git add README.md
```

```
$ git commit -m "initial commit"
```

```
$ git remote add origin  
https://github.com/sharkfoolish/gitTest.git
```

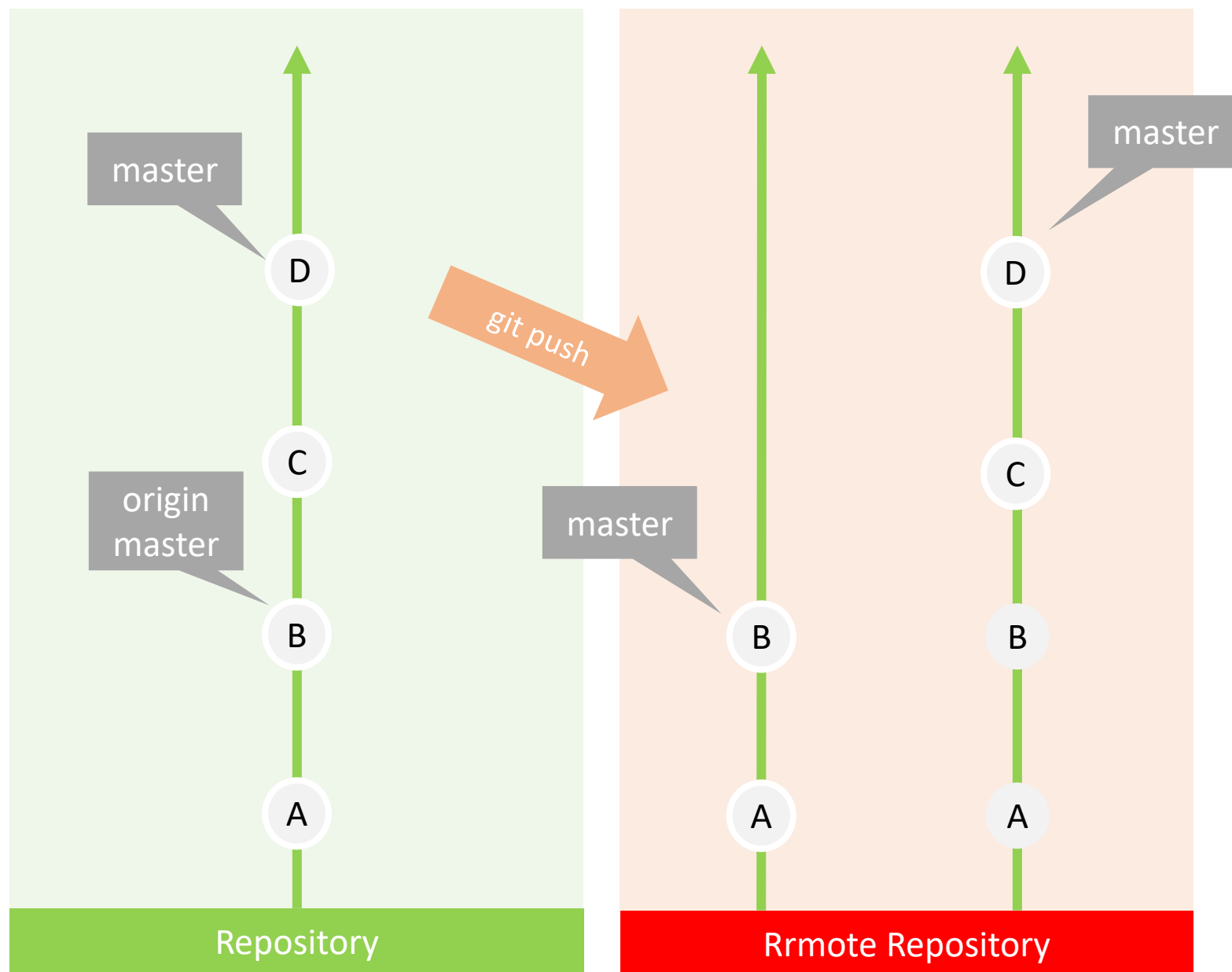
```
$ git push -u origin main
```


遠端儲存庫

/ Git 版本控制

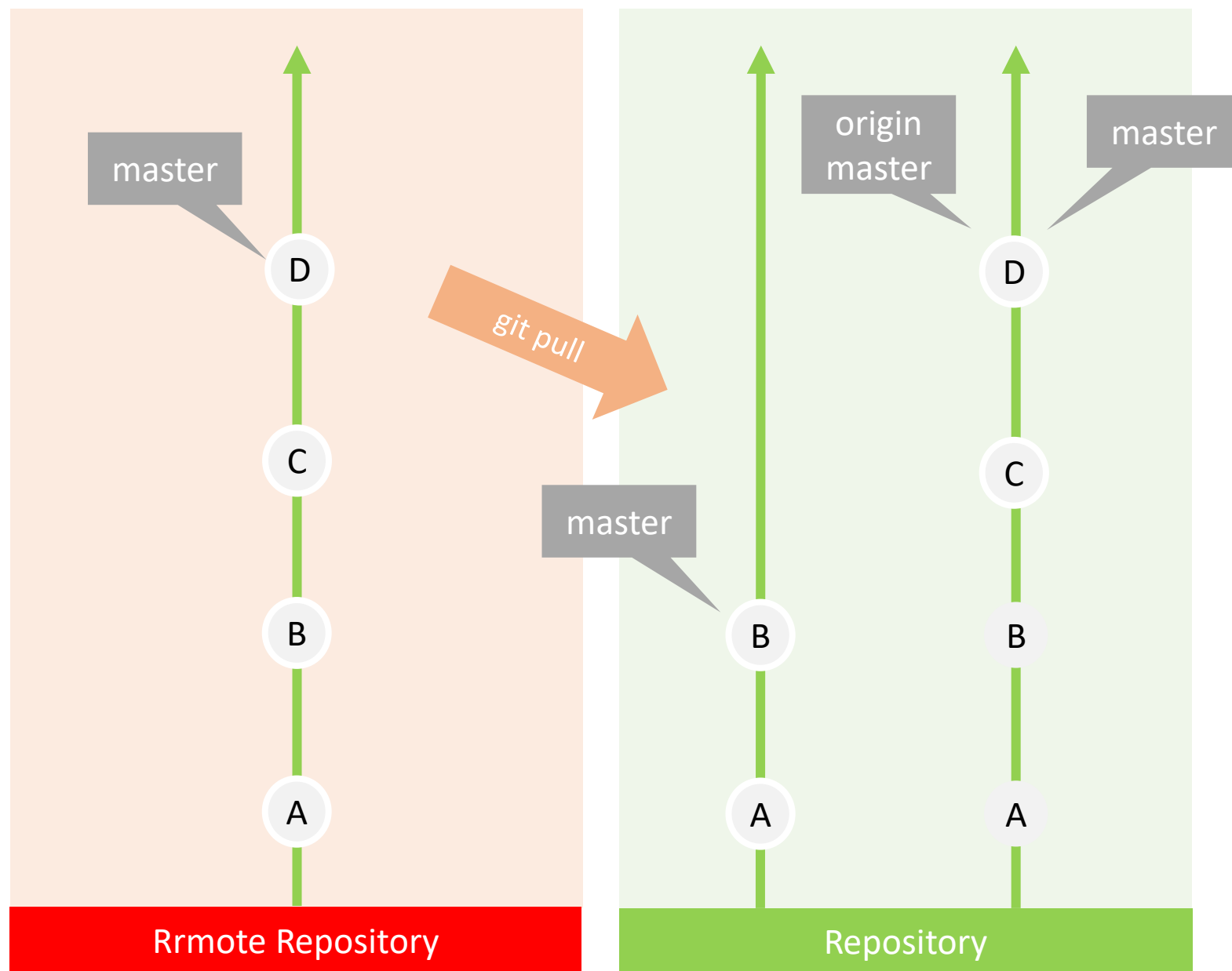
git push

可以將local repo 推上
remote repo，
將以fast-forward方式合併，
如果發生衝突則會被拒絕



git pull

可以將remote repo拉到
local repo並合併，實際
上是fetch與merge的組合。

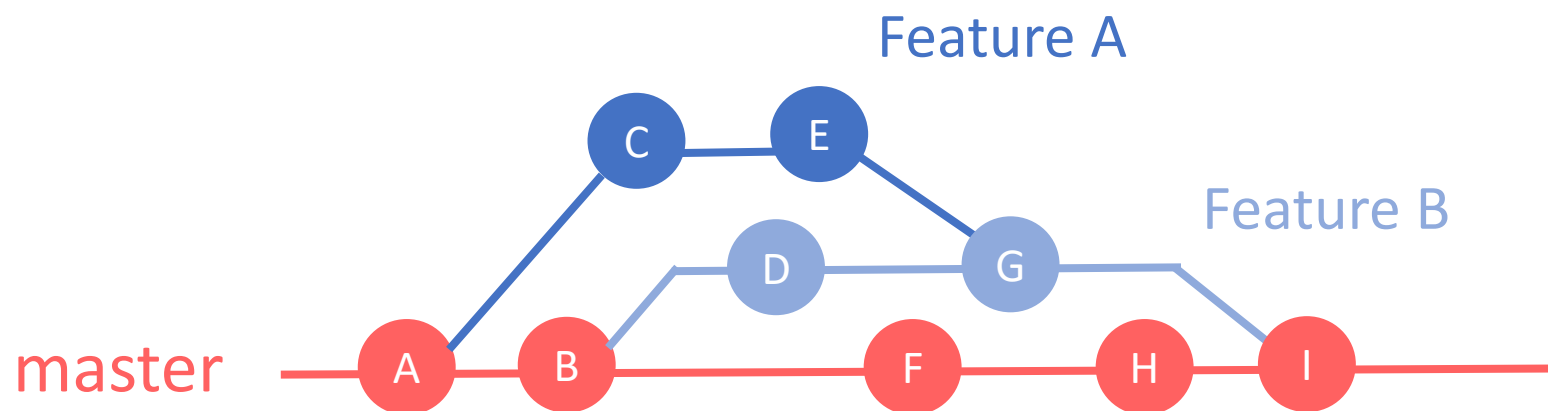


衝突與分支

/ Git 版本控制

git branch

為了多人協作互不干擾，開發人員可以各自開啟分支發展新功能，
等確定沒問題後，再合併到主要分支(穩定)



git branch

```
$ git branch
```

可以察看所有分支並指出目前在哪個分支上

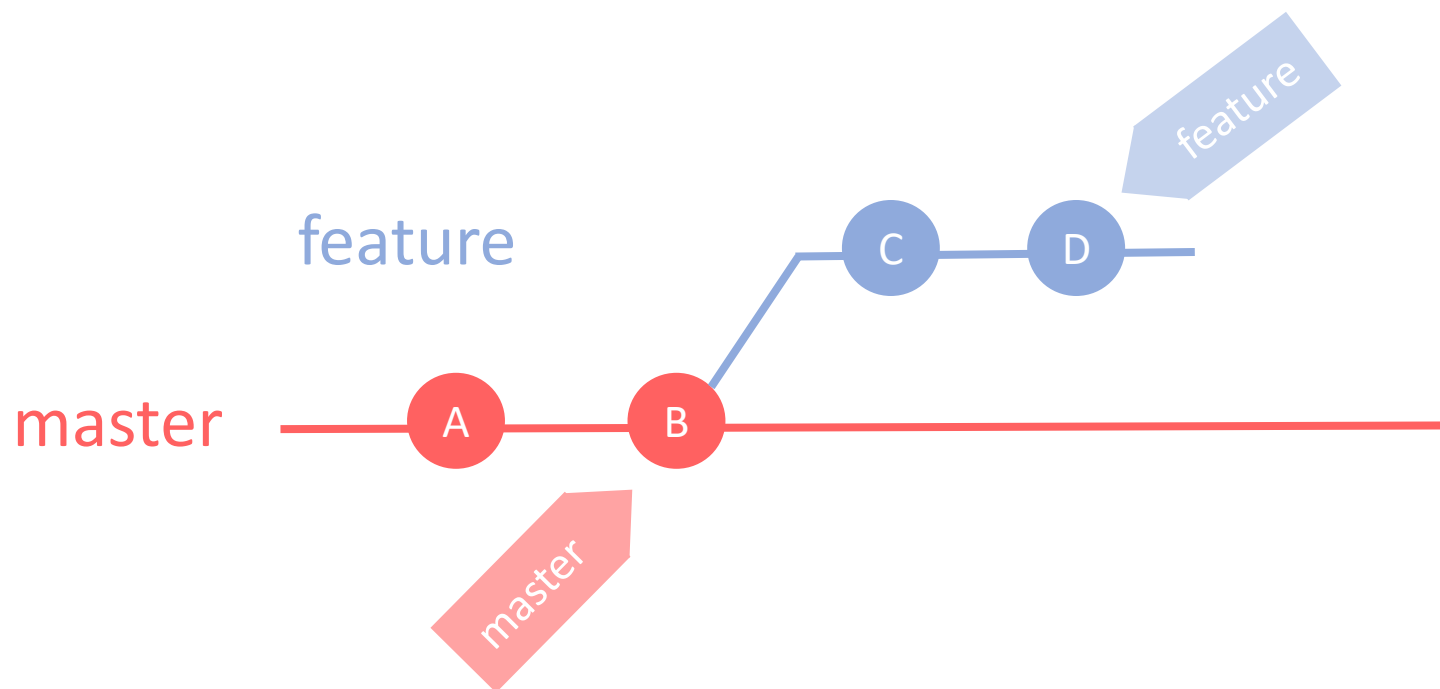
```
$ git branch [name]
```

在當前的位置上，建立一個新分支(建立完後要記得切換到新分支)

```
$ git checkout [name]
```

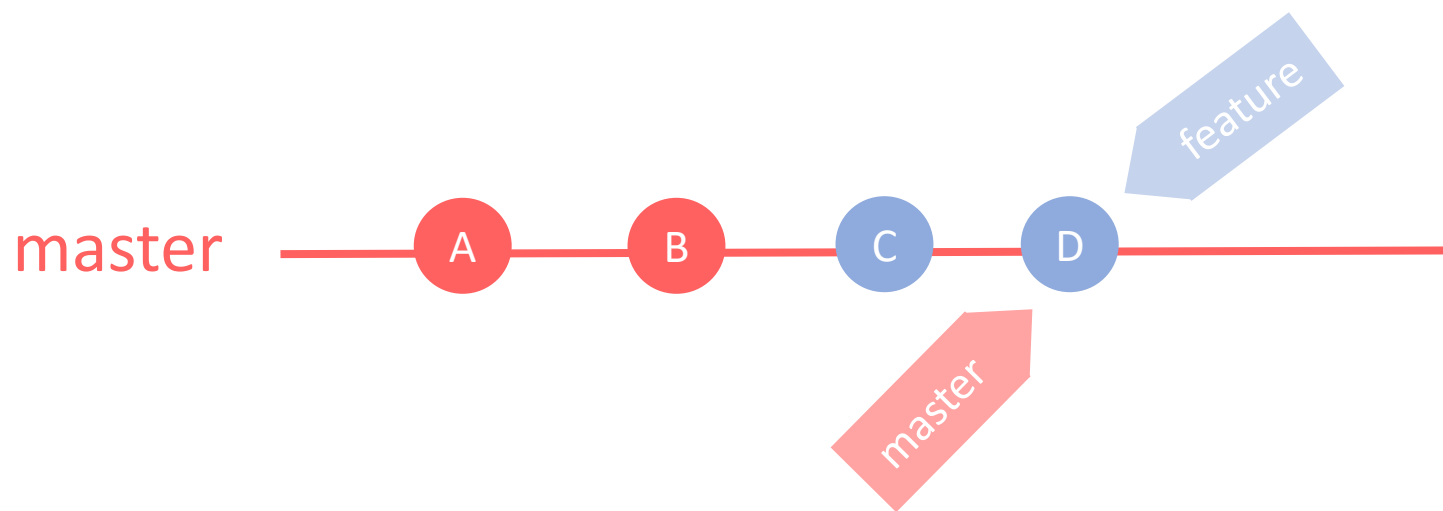
fast-forward

只是將分支的label快轉到最後的位置



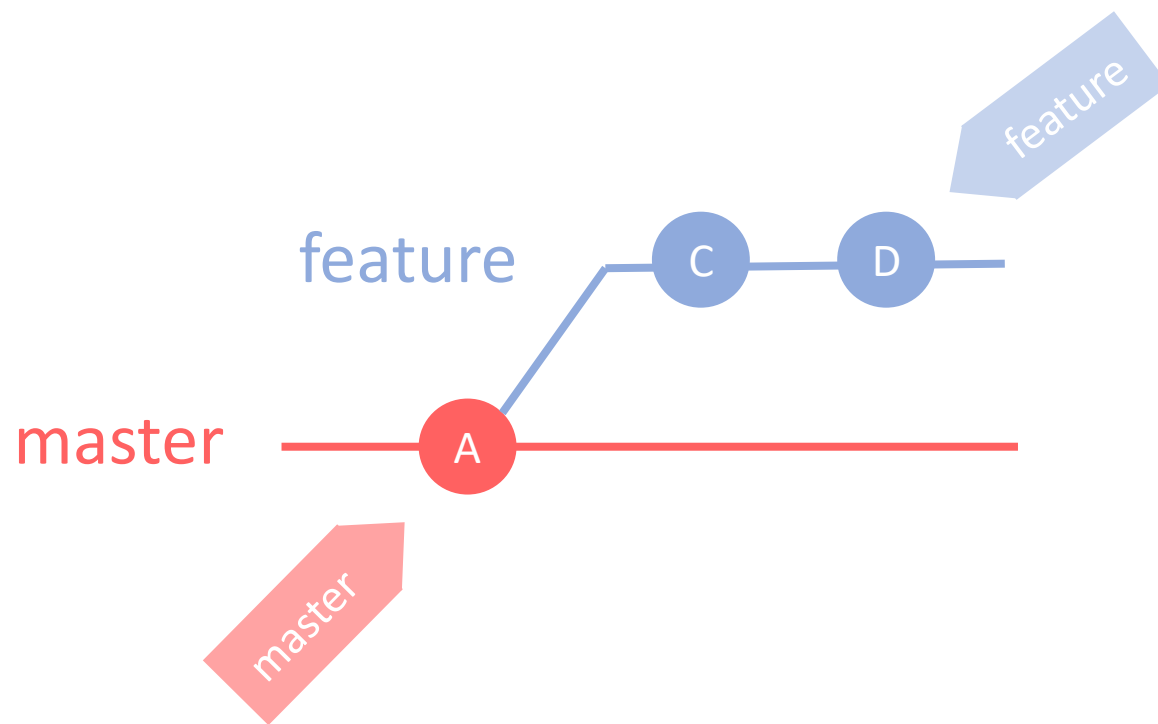
fast-forward

只是將分支的label快轉到最後的位置



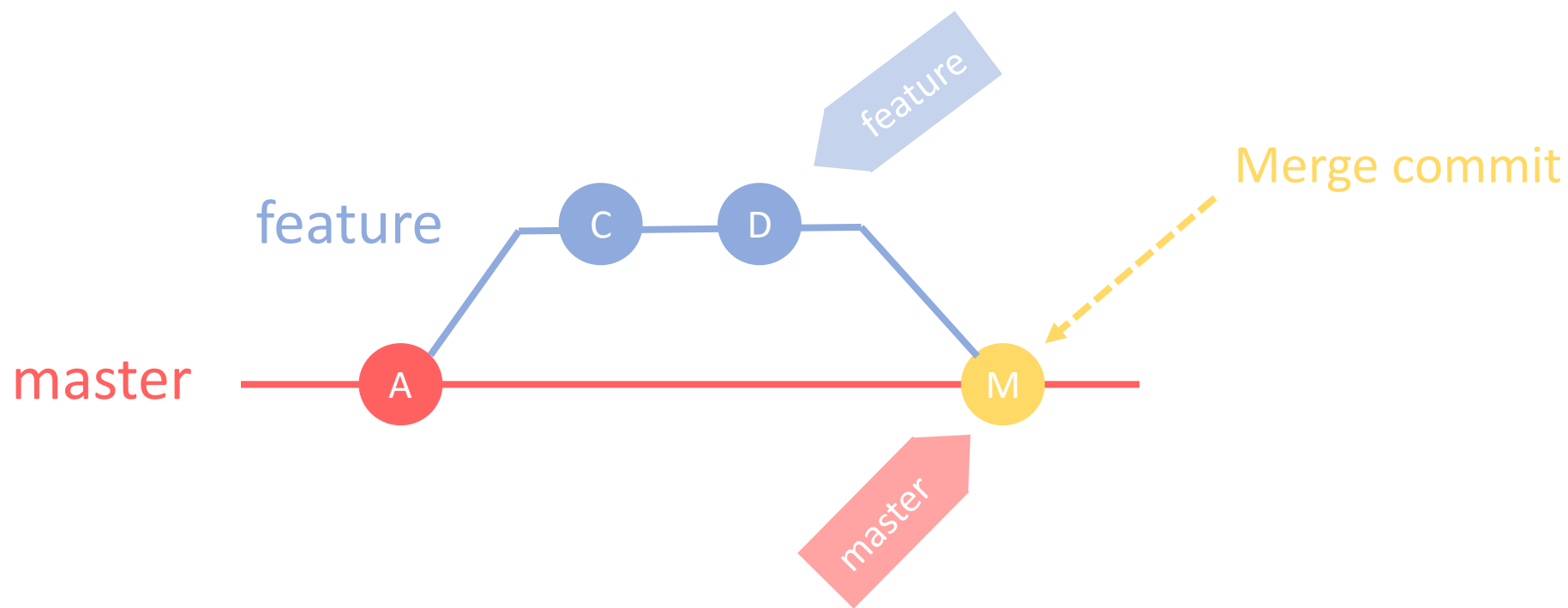
non fast-forward

建立一個merge commit來合併兩個分支



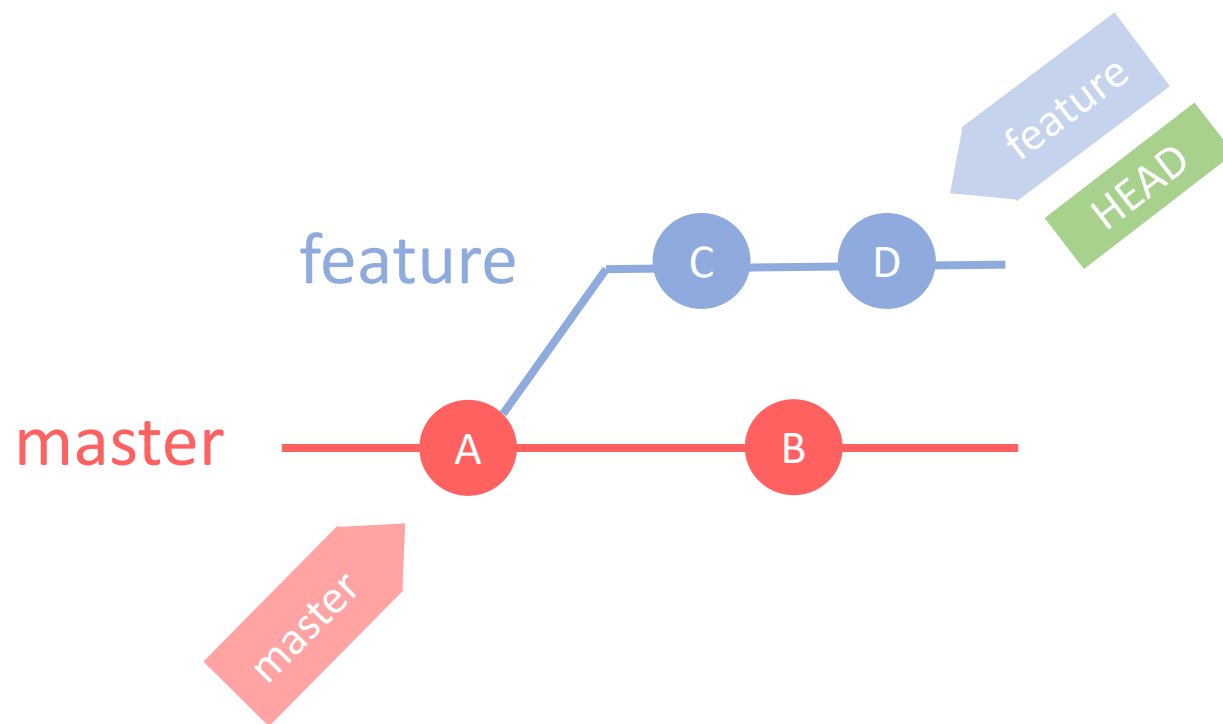
non fast-forward

建立一個merge commit來合併兩個分支



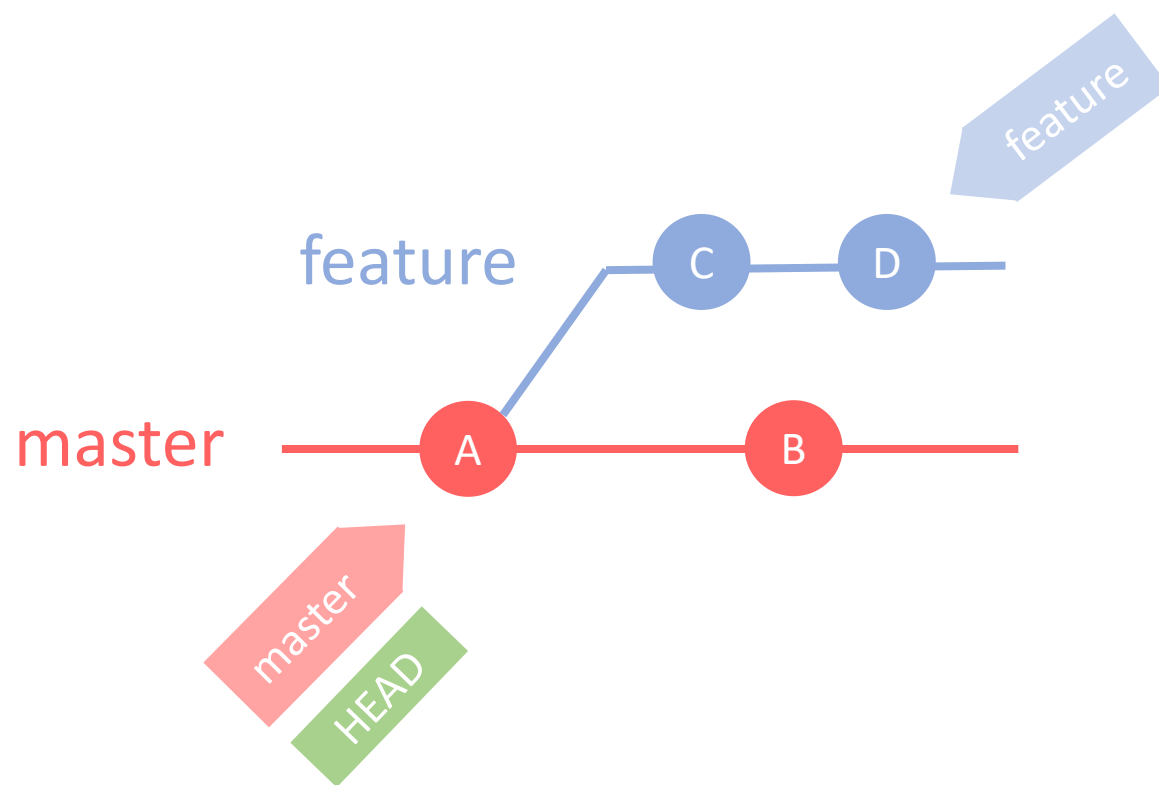
conflict

D和B改動到同一個檔案的同一個部分，所以git不知道要選哪個，所以無法fast-forward



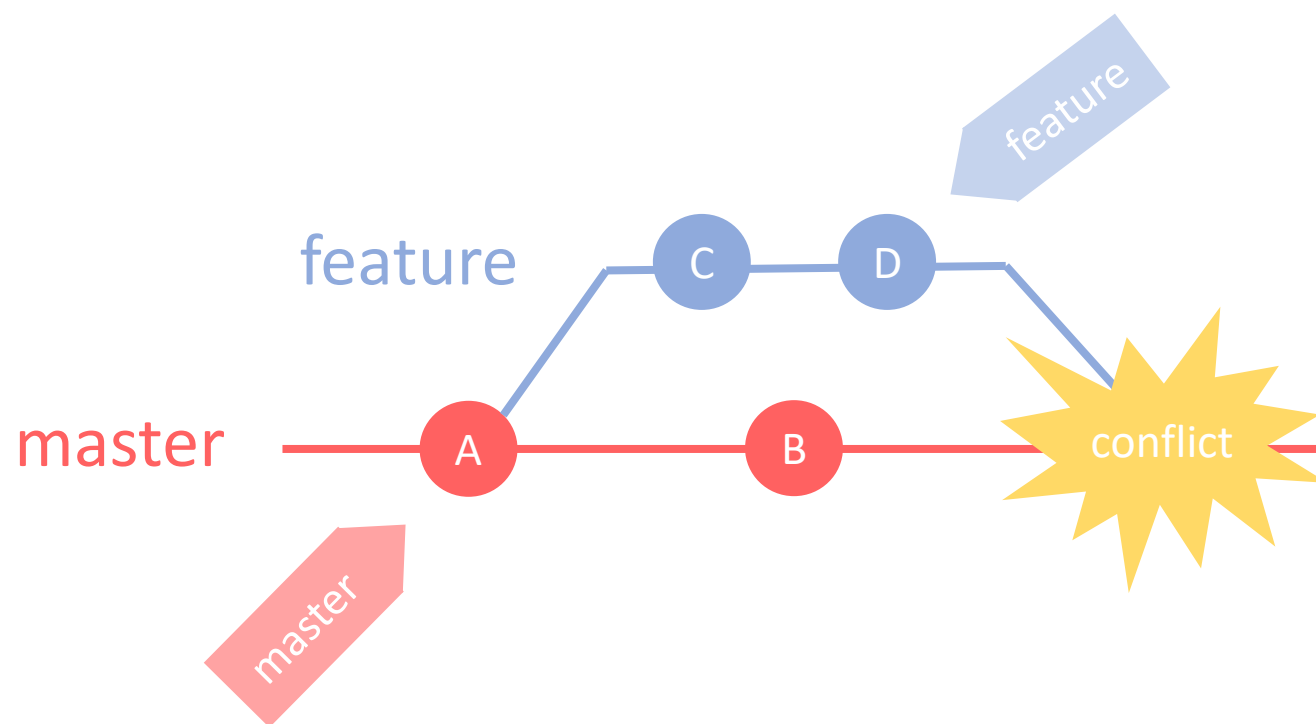
solve

先從被合併的分支(feature)切換到希望合併的目標分支(master)



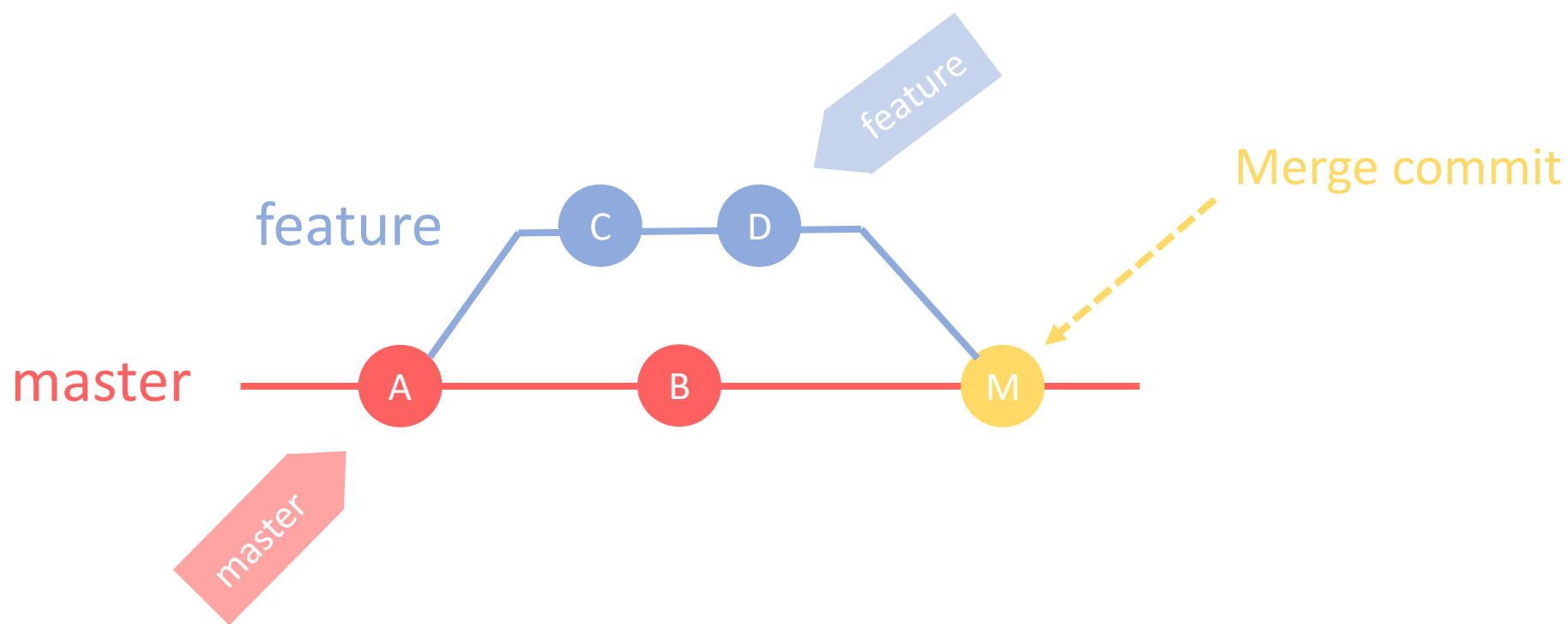
solve

執行 `$git merge`，Git偵測檔案發生衝突



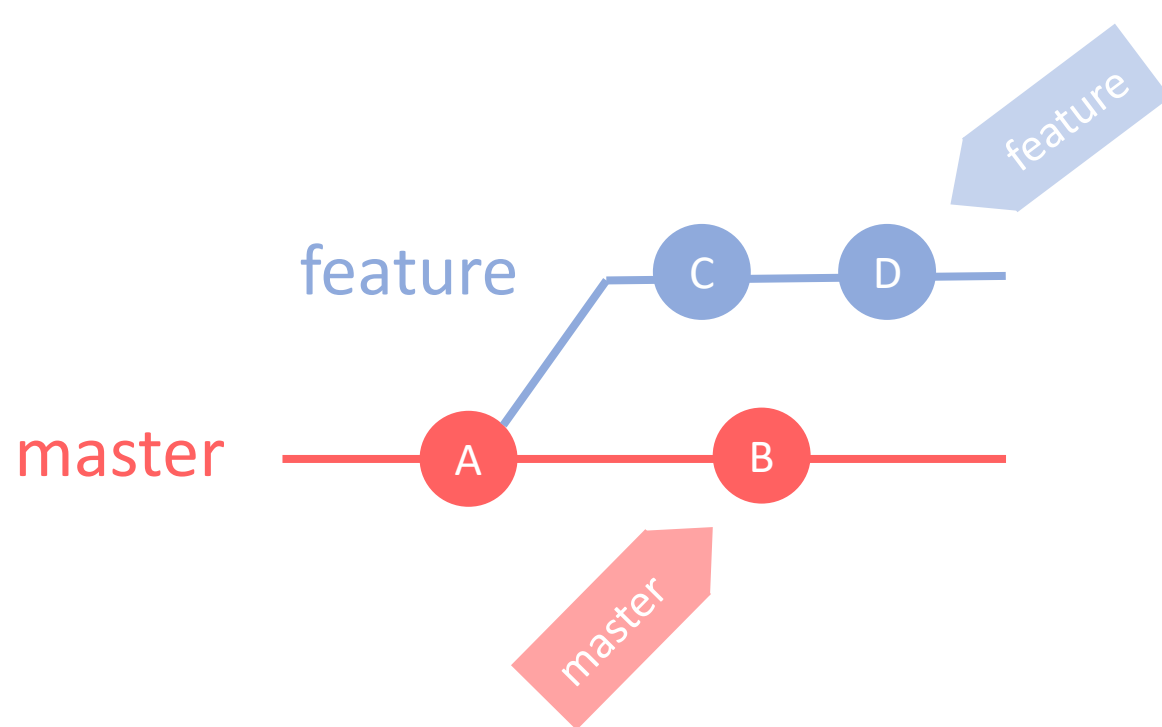
solve

修改衝突的部分後，建立merge commit



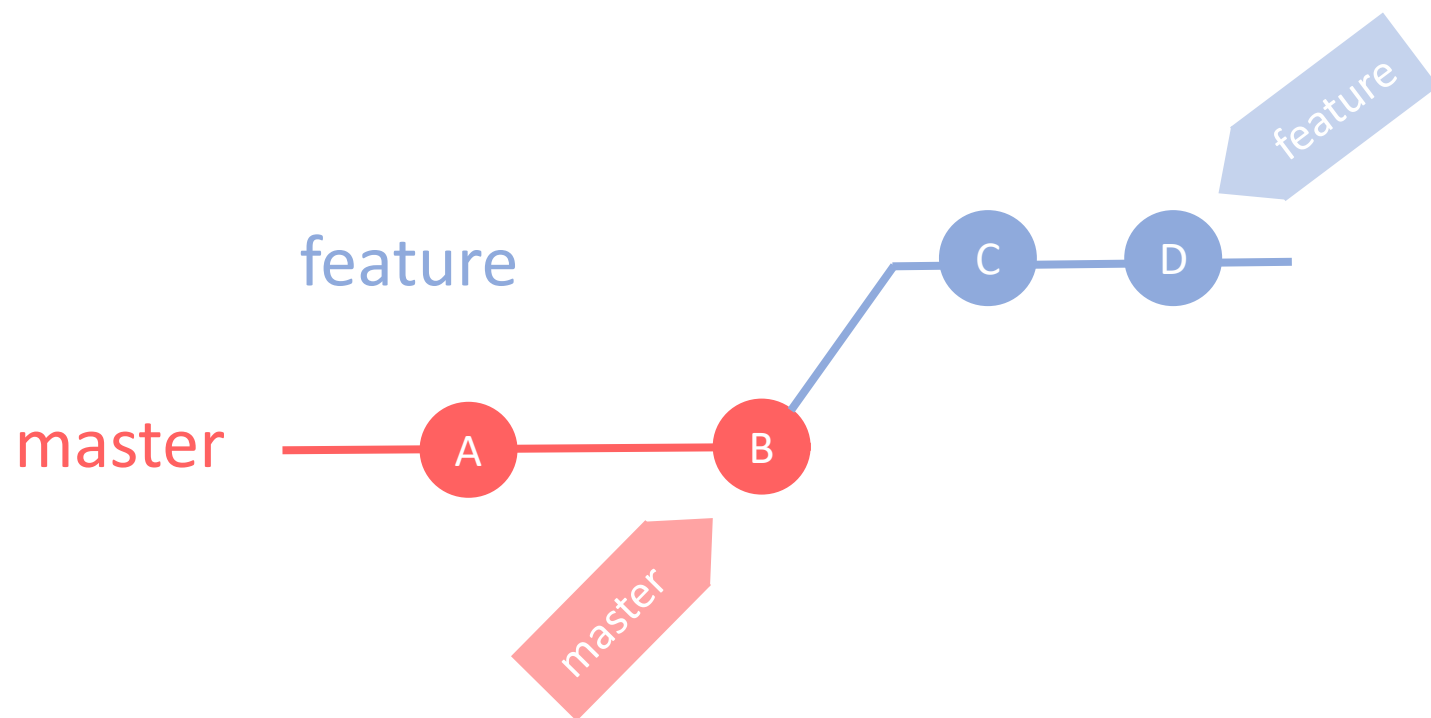
rebase

建立一個新的基底



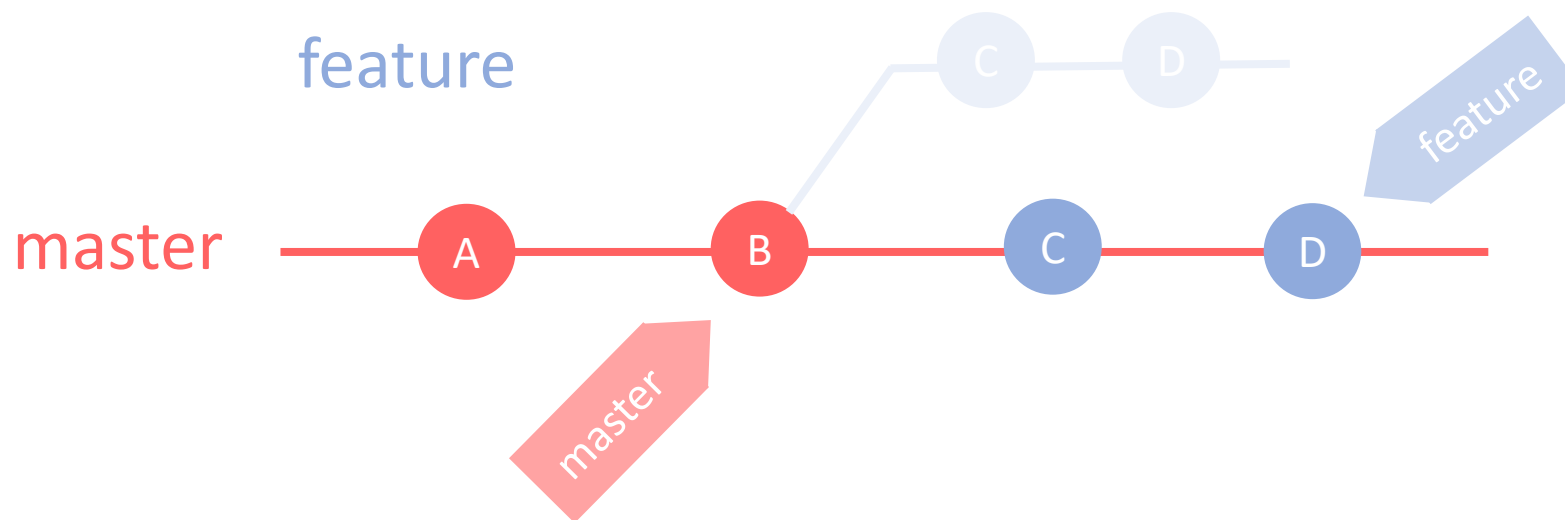
rebase

建立一個新的基底



rebase

建立一個新的基底



多人協作

/ Git 版本控制

Challenge - clone

抓取別人的repository

假設今天我在逛 GitHub 時，看到一個 repository 叫做「sharkfoolish.github.io」很不錯，要怎麼把它抓下來到我的電腦裡呢？

```
$ git clone [url]
```

```
// url : https://github.com/sharkfoolish/sharkfoolish.github.io.git
```

把 remote 的 repository 複製一份到自己的電腦上

Challenge – fork

想修改但是沒有權限

git clone 網路上別人做的「sharkfoolish.github.io」專案後，發現在本機上可以 commit，但是卻無法 git push 到 origin repository

```
$ git clone [url]
```

```
// url : https://github.com/[username]/sharkfoolish.github.io.git
```

在自己的 GitHub 建立 repository，然後再複製一份到自己的電腦上

感謝聆聽

/ 講者: 國立高雄師範大學軟體工程與管理學系 余信陞

<https://sharkfoolish.github.io/>

參考資料

- 連猴子都懂的git入門指南

<https://backlog.com/git-tutorial/tw/>

- 為你自己學git

<https://gitbook.tw/chapters/introduction/about-this-book>

- **Git與GitHub介紹，軟體版本控制基本教學**

<https://tw.alphacamp.co/blog/git-github-version-control-guide>

- **GitAndGitHubTutorial**

Written by shize 