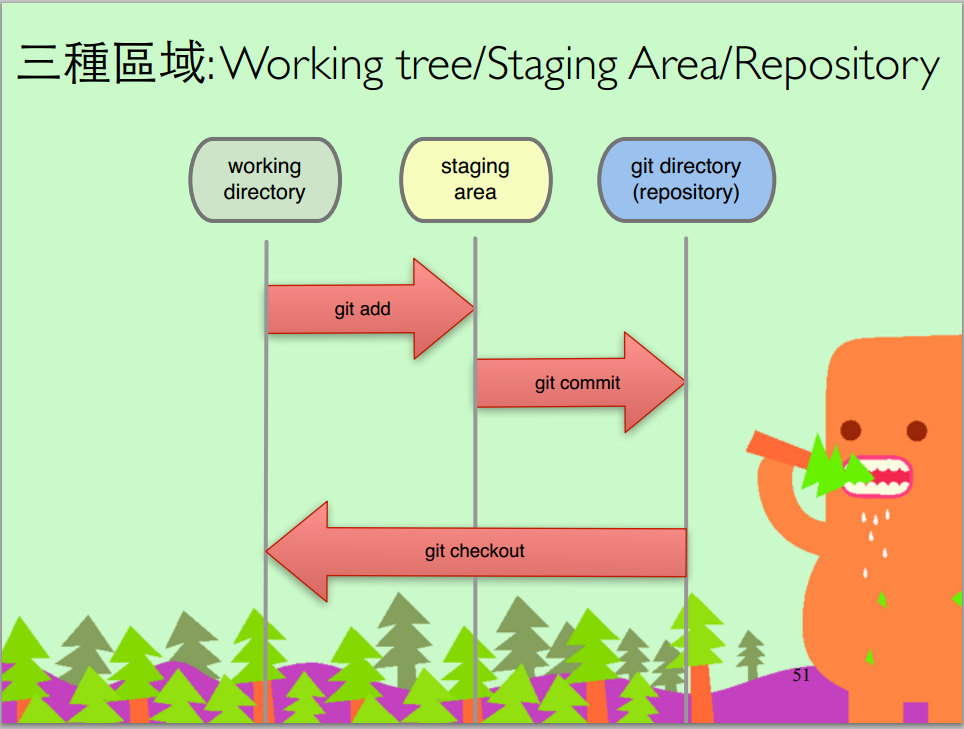
此課程以git version 2.16.2.windows.1 操作

軟體簡介

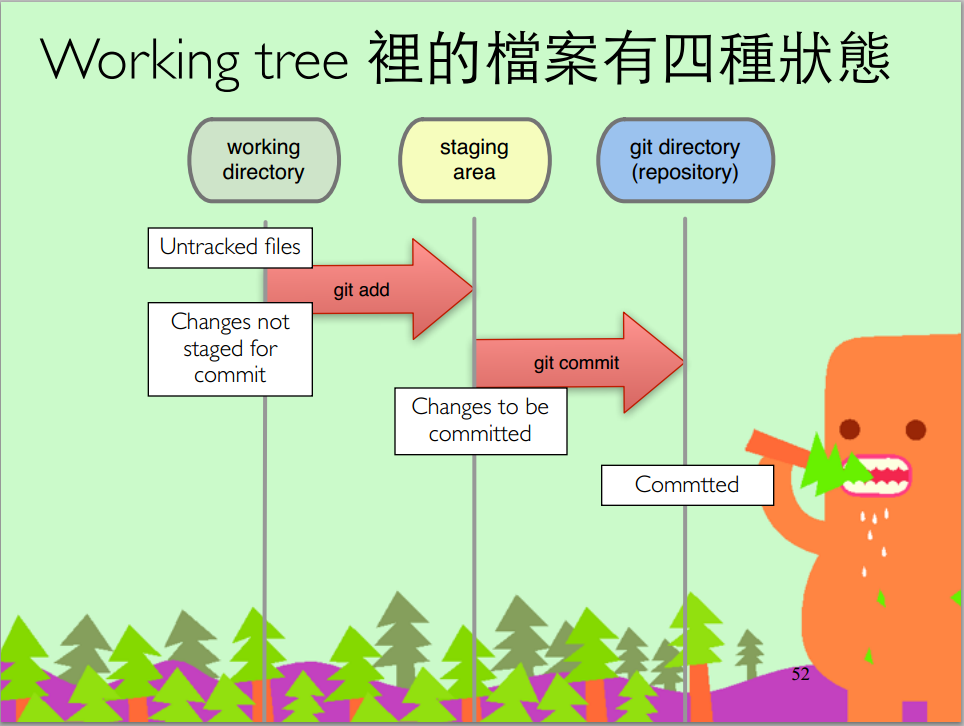
* Git for windows (https://git-scm.com/download/win)
* GitHub desktop (https://desktop.github.com/)
* TortoiseGit (https://tortoisegit.org/)
* SourceTree (https://www.sourcetreeapp.com/)

介紹GIT四個區域

* Working directory(WD), staging(index) area, local repo, remote repo

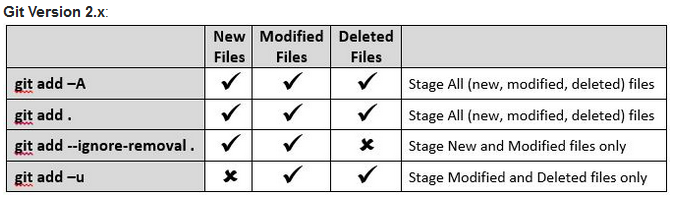


檔案四種狀態



新增local repo

* $ git
* $ git init
* $ git add . (加至staging area)
* $ git commit –m ‘註解文字’ (送出變更)
* $ git commit –am ‘註解文字’ (相當於 add + commit，且僅限已進repo的內容)
* $ git status (檢查狀態)
* $ git log
* $ git log --oneline

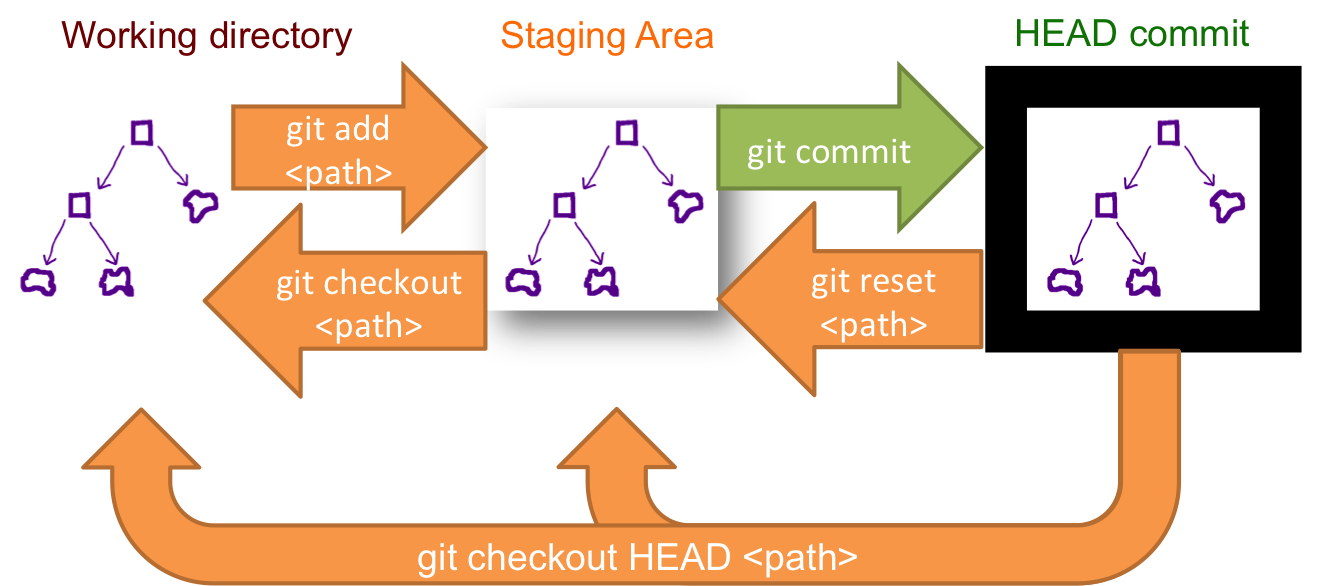


新增Remote repo

* GitHub註冊帳號
* A: GitHub建立新專案
* Create remote repo
* $ git init
* $ echo abc > abc.txt
* $ git add .
* $ git commit -m ‘註解’
* $ git remote (查詢遠端名稱)
* $ git remote -v (遠端詳細位置)
* $ git remote add origin <url> (新增remote repo)
* $ git push -u origin master (將變更push至遠端，-u順便設定遠端分支，git pull 時就不用指定)
* B: 將現存專案上傳至GitHub
* Create new remote repo without initializing a README.md file
* cd 專案目錄
* $ git init
* $ git add .
* $ git commit -m ‘註解’
* $ git remote add origin <url> (新增remote repo)
* $ git push -u origin master (將變更push至遠端，-u順便設定遠端分支，git pull 時就不用指定)
* C: clone遠端專案
* c1: by https
* $ git clone <https>
* c2: by ssh
* $ git clone <ssh> (<http://riny.net/2014/git-ssh-key/>)
* 注意，GIT\_SSH 環境變數使用的ssh client必須和TortoiseGit裡的ssh client 一致(同樣使用git提供的ssh或Tortoise提供的TortoiseGitPlink)
* 若欲變更remote url為https or ssh，可在TortoiseGit/settings/Git/Remote裡設定或使用以下指令
* $ git remote set-url origin <url>

檔案不同區域回復

* 未追蹤(Untracked files)
  + 直接刪除
* 已更改(Changes not staged for commit)
  + $ git checkout -- <file>
* 等待提交(Changes to be committed)
  + $ git reset (移除staging area內的所有變更)
* 已提交(Committed)
  + $ git reset --hard HEAD (移除WD及staging area內的所有變更)

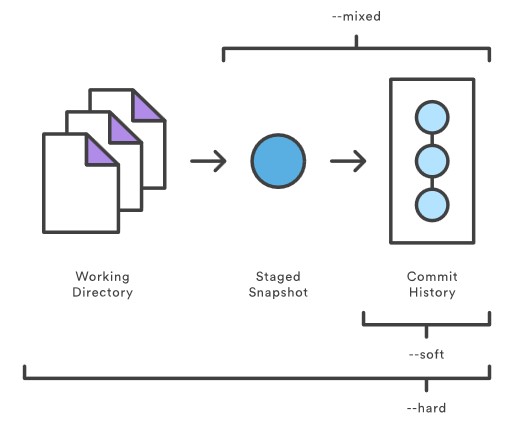


檔案刪除

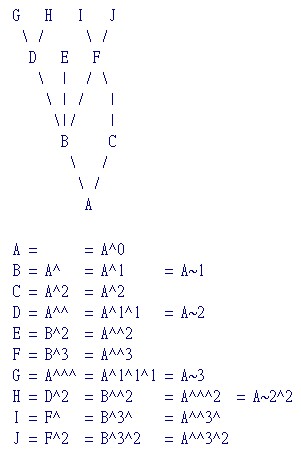
* $ git rm <file> (從local repo刪除)
* $ git rm --cached <file> (從local repo刪除，但保留在WD)
* 以上兩種方法最後仍需commit才能將變更寫入local repo

檔案復原

* $ git reset --soft <commit> (更改local repo內容為指定commit id的內容)
* $ git reset --mixed <commit> (預設值，可不加--mixed，更改local repo及staging area內容為指定commit id的內容)
* $ git reset --hard <commit> (更改WD及staging area及local repo內容為指定commit id的內容)



* $ git revert HEAD (revert目前commit)
* $ git revert HEAD~ (revert前一次第一個parent的commit)
* $ git revert HEAD~2 (revert前兩次第一個parent的commit)
* $ git revert HEAD^ (revert前一次第一個parent的commit)
* $ git revert HEAD^2 (revert前一次第二個parent的commit)



* $ git revert --abort (取消revert)

顯示所有歷程

* $ git reflog (顯示過去所有指令)
* $ git reflog --date=iso (顯示過去所有指令，加上執行日期)

顯示修改內容

* A: before added to staging area
* $ git diff (顯示全部修改內容)
* $ git diff <file> (顯示file的修改內容)
* B: after added to staging area
* $ git diff --cached
* C: after committed
* $ git show <commit> (檢視版本變更內容)

圖形化樹狀圖

* $ git log --oneline --graph --all -n (以簡潔及圖形方式列出所有分支的最新n筆log)
* $ gitk (圖形化顯示目前所在分支及合併進來分支的log歷程)
* $ gitk --all (圖形化顯示所有分支的log歷程)
* Git GUI --> repository --> visualize all branch history

分支新增、切換、合併

* $ git branch (顯示local branch)
* $ git branch --a (顯示local and remote branch)
* $ git branch <分支名稱> (新增分支)
* $ git checkout <分支名稱> (切換至指定分支)
* $ git checkout -b <分支名稱> (新增分支後並切換至新分支)
* $ git branch -d <分支名稱> (刪除分支)
* $ git branch -D <分支名稱> (強制刪除分支，分支內容尚未合併時的情境)
* $ git checkout <分支名稱> (切換至指定分支)
* $ git branch -u <upstream> (設定upstream)
* $ git branch -v (顯示細節or -vv更多細節)
* $ git merge <分支名稱>
* $ git push --all (送出所有分支的變更至remote repo)

暫存工作進度

* $ git stash
* $ git stash -u (包含untracked files)
* $ git stash save ‘註解’
* $ git stash list (列出清單)
* $ git stash apply (取出最新一次的暫存，加 --index 意為staging area也要復原)
* $ git stash pop (取出最新一次的暫存後並移除，加 --index 意為staging area也要復原)
* $ git stash drop (移除最新一次的暫存)
* $ git stash show stash@{1} (顯示指定暫存的內容)
* $ git stash clear (移除所有暫存)

checkout vs reset

* <https://makandracards.com/makandra/11485-git-basics-checkout-vs-reset>

Merge local不同分支情境

* A: fast-forward
* 於分支(iss11)修改檔案a後commit，切回master再去merge分支
* $ git checkout -b iss11
* $ echo a >> 5.txt
* $ git commit -am ‘appended a to 5.txt’
* $ git checkout master
* $ git merge iss11
* B: non fast-forward
* b1: 無衝突
* iss11修改檔案a後commit，master修改檔案b後commit，此時產生一個fork，於master merge iss11，此時產生一個新的commit
* b2: 有衝突
* iss11修改檔案a後commit，master同樣修改檔案a後commit，此時產生一個fork，於master merge iss11，此時產生衝突，衝突解決後才能commit
* b3: 使用rebase方式合併可避免分岐，且減少一次commit，但replay的SHA1值會變，意即會變更history
* $ git checkout iss11
* $ git rebase master

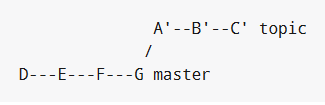
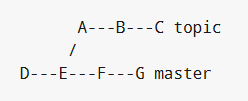
Merge remote同分支情境

* 欲push時，因遠端有更新，所以失敗，必須先pull或fetch + merge後，才能再push
* A: 無衝突
* master@office修改檔案a後commit and push
* master@home修改檔案b後commit and push，因遠端有更新，必須先pull， 自動merge時輸入註解後會commit，結束即可push
* B: 有衝突
* master@home及master@office修改相同檔案，欲push時會發生衝突，先pull，解決衝突後，手動重commit，再push即可成功
* C: 使用rebase方式合併可避免分岐，且減少一次commit，但replay的SHA1值會變，意即會變更history
* $ git pull --rebase origin master

標籤

* $ git tag (顯示local 所有tag)
* $ git tag <tag name> (加新標籤)
* $ git tag -l ‘關鍵字\*’ (列出關鍵字開頭的所有tag)
* $ git tag -a <tag name> -m ‘註解’ (tag加上註解)
* $ git show <tag name> (顯示tag資訊及所在commit的內容)
* $ git ls-remote --tags origin (顯示remote 所有tag)
* $ git push --tags (push時也送出新增的tag)

Rebase (rewind至與指定分支的共同commit後，以指定分支目前變更為底，再replay自己的變更)



* A: rebase不含衝突
* 由master建立並切換至新分支(iss11)，於新分支修改檔案a後commit
* $ git checkout -b iss11 (shorthand: git checkout master –b iss11)
* $ echo a >> 5.txt
* $ git commit -am ‘appended a to 5.txt’
* 於master修改檔案b後commit
* $ git checkout master
* $ echo a >> 6.txt
* $ git commit -am ‘appended a to 6.txt’
* 以master的變更為底後加上自己的變更
* $ git checkout iss11
* $ git rebase master (shorthand: git rebase master iss11)
* B: rebase含衝突
* 於新分支(iss11)修改檔案a後commit
* $ echo b >> 5.txt
* $ git commit -am ‘appended b to 5.txt’
* 於master同樣修改檔案a後commit
* $ git checkout master
* $ echo c >> 5.txt
* $ git commit -am ‘appended c to 5.txt’
* 以master的變更為底後加上自己的變更
* $ git checkout iss21
* $ git rebase master
* Resolve conflict
* $ git add .
* $ git rebase --continue

修改註解

* A: 修改最後一次
* $ git commit --amend (修改好註解後存檔即可)
* B: 修改前幾次
* $ git rebase -i HEAD~2
* 於editor內將欲修改的註解前改成r後存檔
* 依提示修改註解

cherry-pick (移除指定的commit，以及replay指定的commit)

* A: 無衝突
* <https://zlargon.gitbooks.io/git-tutorial/content/patch/remove.html>
* B: 有衝突
* <https://zlargon.gitbooks.io/git-tutorial/content/patch/cherry_pick_conflict.html>

reset, checkout, revert, rebase的差異

* 討論

detached HEAD state

* Head未指向分支最末端，再次checkout <branch>即可脫離此狀態
* $ git checkout <commit>
* $ git log --oneline
* $ git checkout master
* 若要保留此狀態下的commits，可建立一新分支，也可脫離此狀態
* $ git checkout -b <branch name>

強制更新遠端

* $ git push -f (以local內容強制覆蓋remote內容，小心使用)

Git Flow



GitHub Flow

* <https://guides.github.com/introduction/flow/>

GitHub Flow 及 Git Flow 流程使用時機

* <https://blog.wu-boy.com/2017/12/github-flow-vs-git-flow/>

參考文件

* <https://git-scm.com/book/zh-tw/v2>
* <https://kingofamani.gitbooks.io/git-teach/>
* <https://backlog.com/git-tutorial/tw/intro/intro1_1.html>
* <https://github.com/geeeeeeeeek/git-recipes/wiki>
* <https://zlargon.gitbooks.io/git-tutorial/>
* <https://gitbook.tw/>