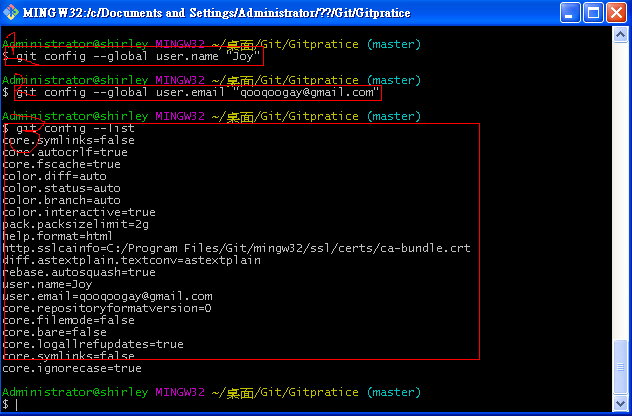
# Git and GitHub

1. 下載GIT

2、設置你的GIT

先設定你的基本資料(之後commit的時候才容易知道作者是誰)

1就是設置你的姓名

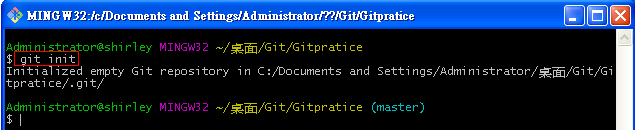
git config - -global user.name “你的姓名”

2就是設置信箱

git config - -global user.email. “你的信箱”

3檢查你的狀態，看你信箱設置多少跟姓名設置什麼

git config –list

1. 如何把自己的資料夾變成儲存庫 

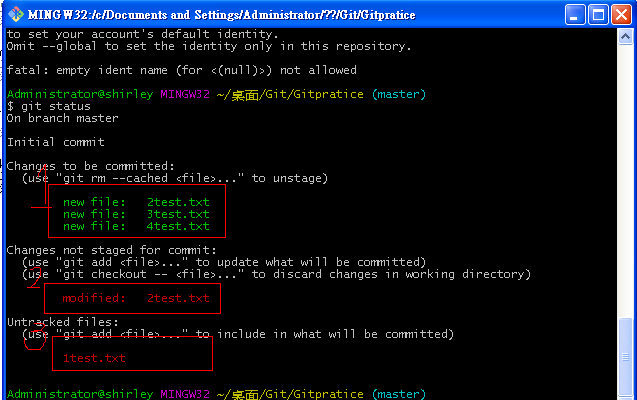
輸入git init，就可以把資料夾變成儲存庫了

首先我們來介紹Git，也就是自己的儲存庫的一些狀態，這樣才可以管理自己的儲存庫，有分四種狀態

* 未追蹤(Untracked files)：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，簡單來說就是在這個文件夾變成儲存庫後面才新加的。
* 已更改(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)，簡單來說當檔案從Changes to be committed or Committed **只要這個檔案被修改都會變成** Changes not staged for commit
* 等待提交(Changes to be committed)：在工作目錄（未追蹤Untracked files）的檔案執行git add後，會放在暫存區(Stage)。這些放在暫存區的檔案狀態便是等待提交囉！
* 已提交(Committed)：在暫存區(Stage)的檔案執行git commit後，檔案便置於儲存區(Repo)，這些放在儲存區的檔案即是已提交的狀態
* commit 是寫程式時一個很重要的動作，一個 commit 在 Git 中就是一個節點，這些 commit 的節點就是未來你可以回朔及追蹤的參考，你可以想像就像是電玩遊戲時的存檔，每一個 commit 就是一次存檔，讓我們未來在需要的時候都可以回到這些存檔時的狀態。因此我們將剛剛做的修改提交：

1. 剛剛說過儲存庫的四種形態，在資料夾中的圖形，跟如何變這四種形態

這在之前要說一個指令git status 可以看到這檔案中所有的檔案的狀態



以上圖為

* 1. 就是已經已提交(Committed) or 等待提交(Changes to be committed) 。
  2. 你看到2為什麼已提交的檔案又變成是modified 2test.txt 這表示這個檔案被修改過了 你要 git add 2test.txt 在 git commit 2test.txt 在把它變成已提交的檔案。
  3. 就是新加入的檔案,不是儲存庫裡面的檔案。

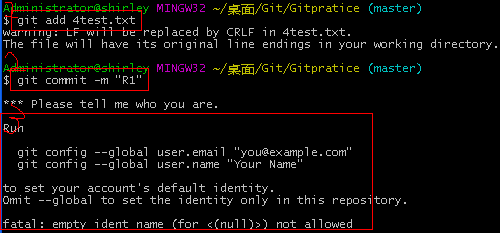
問:那如何分已提交(Committed) or 等待提交(Changes to be committed)?

3.1第一種狀態是未追蹤(Untracked files)，也就是當檔案變儲存庫，在加入的文件C:\Documents and Settings\Administrator\桌面\Git\Gitpiture\Gitstatus1.bmp

3.2先說第三種狀態是等待提交(Changes to be committed)輸入git add (檔案名稱)，就可以把未追蹤(Untracked files)－＞等待提交(Changes to be committed) 。

像下面那張圖我的檔案就叫3test.txt所以我就輸入git add 3test.txt

C:\Documents and Settings\Administrator\桌面\Git\Gitpiture\Gitstatus3.bmp

.3.3在來說第四種狀態是已提交(Committed),就是一個準備上傳的檔案了,

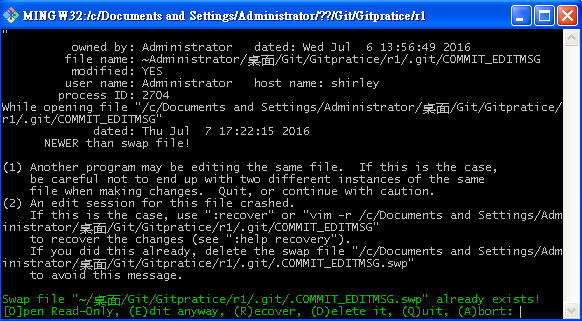
1就是把4test.txt變成等待提交(Changes to be committed),意思就是準備可以提交到自己儲存庫了。

2就是把等待提交(Changes to be committed)-> 已提交(Committed),表示這份檔案已經傳到到自己儲存庫了　–m 就是快速提交“R1”,而這裡面的””,就是註解就是提交的這個節點。

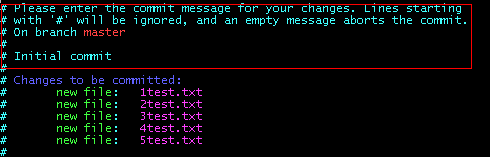
3就是告訴我們的信箱並沒有設,他不知道自己儲存庫當上傳到雲端儲存庫時要叫做什麼信箱，姓名要叫什麼，因為當遠端儲存庫要跟自己的儲存庫作push跟pull時他要知道你是誰,才能夠給你push or pull 不然,隨便一個人都可以push or pull那就誰都可以動到你的遠端儲存庫。

前面說到一個 commit 在 Git 中就是一個節點，一個很重要的概念commit是有順序的,順序可以變一張圖

假如是用　git commit 就會跑出



這時可以按q離開



這個視窗中最上面的空行是要給你寫下你這次 commit 的訊息，例如在這裡寫下 “Add test.rb file to test git function”來表示這一次提交的目的，**強烈建議在 commit 的時候要盡量清楚表達這次 commit 的內容為何**，因為這會讓你未來要回頭看 code 的時候能讓你快速的找到你想要找的內容，也能對團隊中其他成員了解你在做什麼。

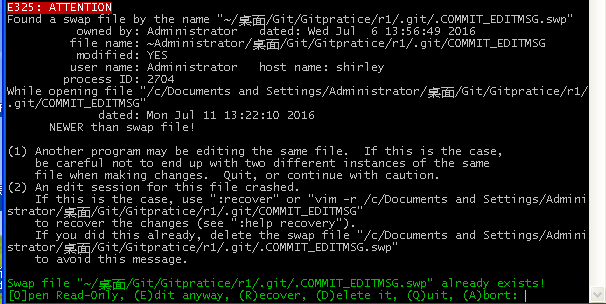
在說另一個指令git commit --amend

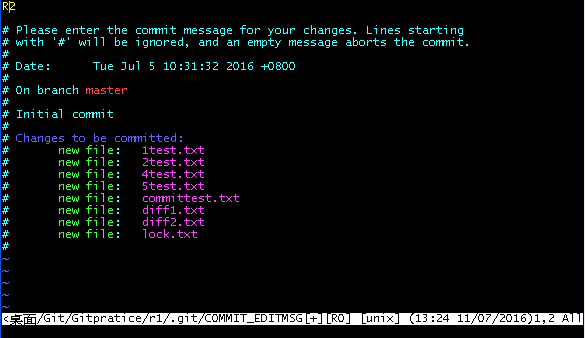
這是用來修改上一個commit註解的指令

我們來把R1改成R2

輸入git commit –amend

然後就會跑出像之前git commit 的這個畫面



按下i跑到編輯模式然後把R1改成R2 按下Esc 打上:wq!就修改歷史記錄了

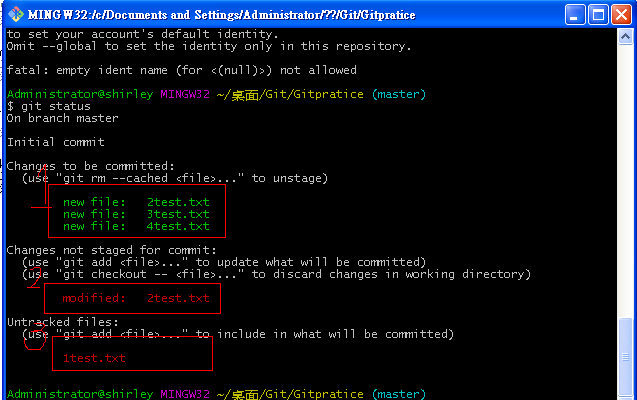
**vim 幾個簡單指令**

*i 進入編輯模式(自由打字啦)  
Esc 退出編輯模式(回到一般模式)  
dd 刪除一行(一般模式下)  
:wq! 存檔離開(一般模式下)  
:q! 不存檔直接離開 (一般模式下)*

4 在來回來說剛剛的已更改(Changes not staged for commit),

當我把2test.txt變成等待提交(Changes to be committed) or

已提交(Committed),我修改檔案時就會變已更改(Changes not staged for commit)的狀態。



* 1. 1就是已經提交的檔案。
  2. 你看到2為什麼已提交的檔案又變成是modified 2test.txt 這表示這個檔案被修改過了 你要 git add 2test.txt 在 git commit 2test.txt 在把它變成已提交的檔案。

1. 回復GIT 不同區域的修改

假設我們在工作目錄(WD)、暫存區(Stage)或儲存庫(Repo)所做的修改後悔了！想要回到之前的狀態，怎麼辦咧？

這時就要用這些指令了

git reset

git rm –cached

git checkout

git reset –hard

4.1

未追蹤(Untracked files)直接砍掉就可以了

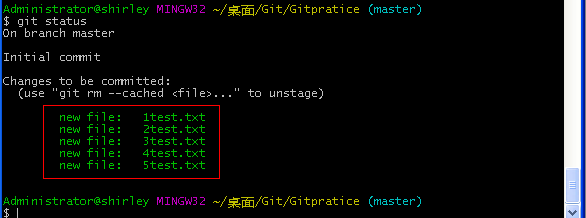
指令是

rm 1test.txt

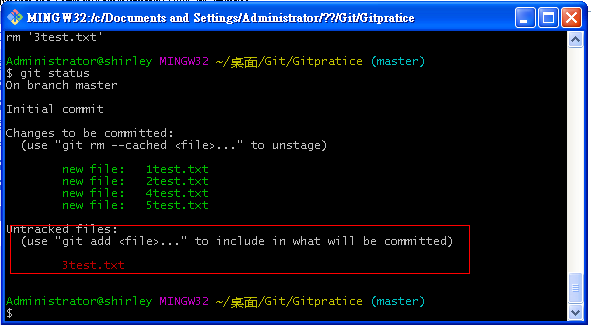


4.2

我要將狀態等待提交(Changes to be committed)變成未追蹤(Untracked files)就要打

git rm --cached 3test.txt<-這邊文件的姓名

當輸入git rm –cached 3test.txt時變成



他就變回git未追蹤(Untracked files)了

4.3

已提交(Committed)變成未追蹤(Untracked files)

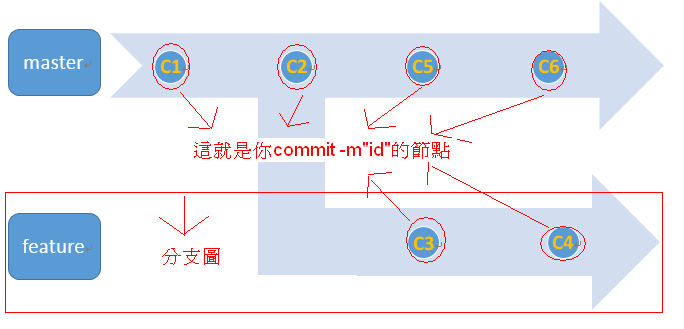
用reset

首先

git log 查看 commit 的版本

5、GIT的分支

當你在開發軟體時,假如老闆叫你去抵bug這時你不想放下手邊的工作,你就可以開一個branch(分支),後就會用到分支圖



C:\Documents and Settings\Administrator\桌面\Git\Gitpiture\Gitbranch1.bmp

Not:當你沒有commit時是不能建立分支

5.1建立分支

上面那個紅框就是你現在分支的id

當你打 git branch feature

就是創造一個分支叫做feature

轉換到分支或回去要打

git checkout feature

就會轉換到 feature 的分支

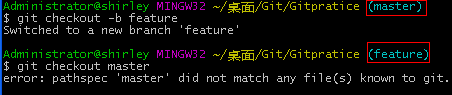
而兩個加再一起

git checkout -b feature

就可以創一個分支又轉換過去

創造分支的指令

【git checkout -b feature】 = 【git branch feature】 + 【git checkout feature】



查看分支 的指令 git branch

C:\Documents and Settings\Administrator\桌面\Git\Gitpiture\Gitbranch3.bmp

\*表示當前分支位子,而下面就是其他分支

在來要說刪除分支,假如我們創錯分支要怎麼辦?這時就可以用一個指令是

git branch –d 你要砍的分支idC:\Documents and Settings\Administrator\桌面\Git\Gitpiture\Gitdelete.bmp

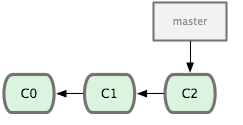
-d 就是強制刪除的意思

Note:分支砍掉commit並沒有砍掉

5.2分支的合併

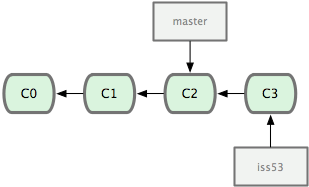
剛剛介紹過了如何創分支,在來說如何把分支合併起來,把一個分支修改整合到

假如你正在專案工作,並且提交了幾次更新

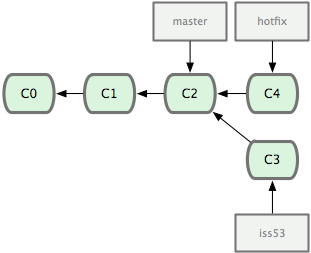


這是有個bug要做#53,你要解決,這時就必須開的分支 所以我用

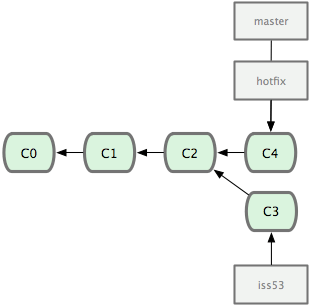
git checkout –b iss53 來開一個分支叫做 iss53 又切換過去



這時你開始修復iss53 問題 所已開了一個commit 叫做C3 但這時有個緊急問題要解決所以我們需要切回mahster 創一個新分支(不能在iss53創分支因為會變iss53的分支)所以我開個hotfix的分支,直到把問題解決

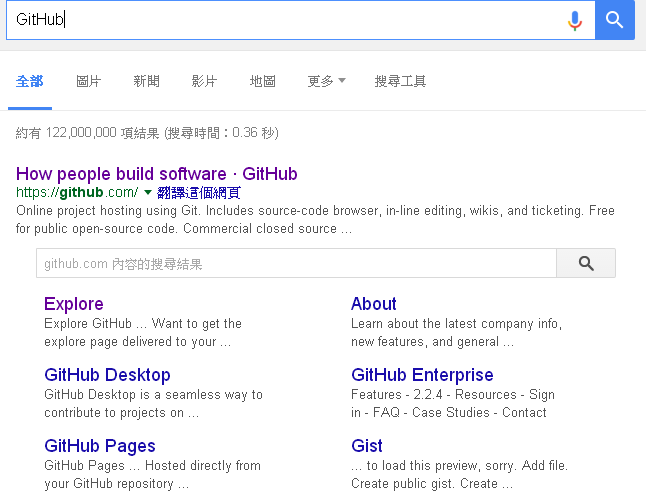


這時我們處理好hotfix的問題了,我們想把hotfix合並,這時我們可以要切回要合併的分支,然後打 git merge hotfix 就會把hotfix->合並到master就會變



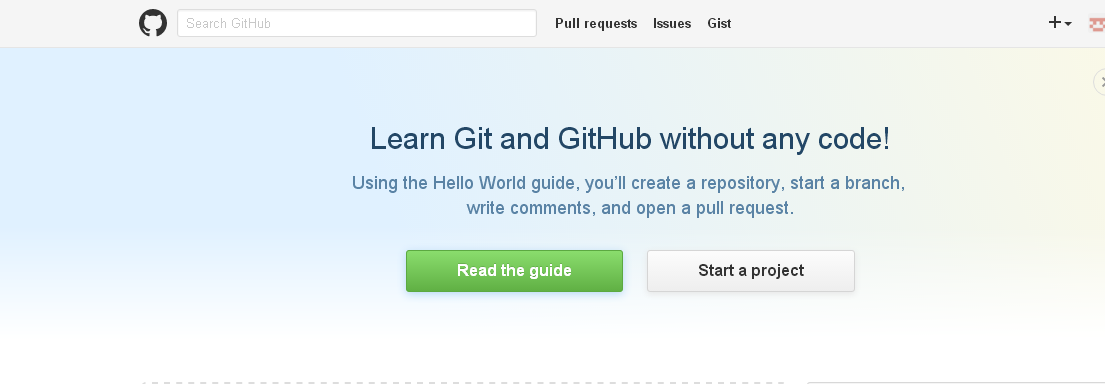
# 6、遠端儲存庫 GitHub

Git不管遠端的儲存庫的東西,所以我們需要一個東西讓我們能把自己儲存庫的東西放到遠端上,而這個東西就是GitHub,所以我們要下載一個GitHub,首先我們要去GitHub 的官網下載

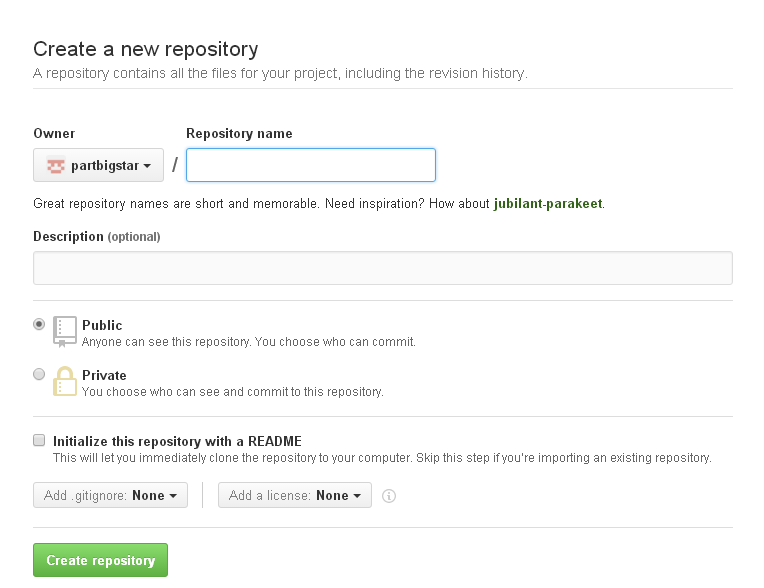




創一個帳號

你就會進入

你就點Start a poject

創一個poject

在reposltory name 輸入專案要的名子

然後public跟 private public是其他人可以下載你的東西 private是其他人不能下載你的東西private 是要付費的 而帳號是免費的都是public

而Initialize this repository with a README　打勾後,系統會幫你建一個README.md檔,github會把它當成是儲存庫的預設說明,如果你想把你的專案上傳的話,就不用勾,以免到時候要處理這個檔案遠端多人合作開發(自己一人)

先來介紹一些基本的指令

git remote –v

可以用來檢查目前的遠端儲存庫有連到哪些



Note:

當你在push時一定要 commit 不然他會push false。

Push

你必須先打

git remote add origin <https://github.com/你的id/你的檔案名.git>

例如我的id要做partbigstar

我的檔案叫r1

所以我的https://github.com/partbigstar/r1.git

C:\Documents and Settings\Administrator\桌面\Git\Gitpiture\Gitpush.bmp

這行指令意思是說我要連到的遠端儲存庫在

<https://github.com/partbigstar/r1.git>

然後我的遠端儲存庫叫做original1

上面失敗是,因為我已經建立過了

在來就是把檔案push上去

這時後就要打指令

git push –u original master

這裡就是我要把master的分支 的 commit

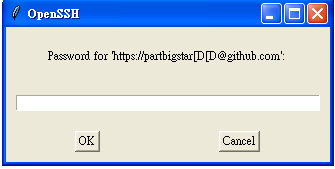
push 到original

C:\Documents and Settings\Administrator\桌面\Git\Gitpiture\Gitpush2.bmp

這時後他會叫你輸入你的傳到遠端支料庫的id

像我Id叫做partbigstar

跟密碼會顯示出這個畫面

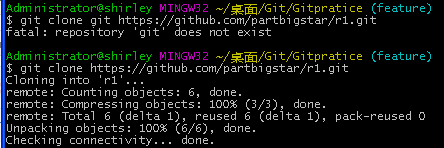


成功就會跑出



假如我的檔案都不見了,這時我要從遠端下載下來資料,這時用pull 指令把遠端儲存庫的東西下載下來, git clone 你的遠端儲存庫你就輸入

git clone <https://github.com/你的id/你的檔案名.git> 就會把 東西下載下來給你



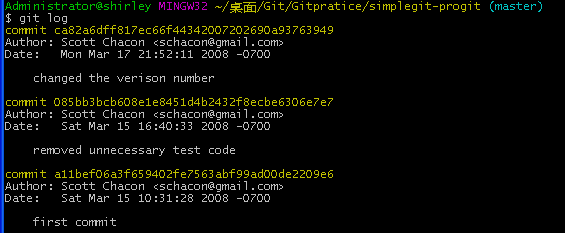
在來說個比較特殊的指令Fetch,有時後只想確定遠端儲存庫的內容,但不想真的合併,在這種情況下,請使用fetch

執行fetch時,會取得遠端儲存庫的最新歷史紀錄,取得的歷史紀錄會自動建立一個為FETCH\_HEAD的分支

# 加入或查看歷史紀錄

Note:查看git的歷史紀錄

打git log



就能看到以新到舊的順序列出儲存庫的提交的歷史紀錄，也就是說最新的更新在上面,舊的在下面,也會列出每個SHA-1查核植、跟作者大名與電子郵件地址、及提交時輸入訊息

打git log –p -2

2可以是其他數字會顯示最後兩個更新結點跟註解

SHA-1查核植是什麼?

Note:如果你遇到

fatal: Unable to create 'C:/Documents and Settings/Administrator/桌面/Git/Gitpratice/r1/.git/index.lock': File exists.

Another git process seems to be running in this repository, e.g.

an editor opened by 'git commit'. Please make sure all processes

are terminated then try again. If it still fails, a git process

may have crashed in this repository earlier:

remove the file manually to continue.

請用 rm -f ./.git/index.lock 來解決

# Git的網路協議

Git主要有四個傳輸方式,本地傳輸,SSH傳輸,Git傳輸和HTTP協定,除了HTTP協定其他都在伺服器端安裝Git運行。

1.本地協定,一個人在一台電腦中開伺服器共用一個檔案系統(NFS),所有人共用一台電腦,把遠端資料庫,丟到這台電腦中,

風險:有可能這台電腦壞了。

好處:方便。

指令是 git clone /opt/git/project.git。

Or git clone <file:///opt/git/project.git>如果是 如果你在URL(網址協議類型://伺服器地址（必要時需加上埠號）/路徑/文件名) 如果只使用路徑,Git會嘗試使用硬連結或直接複製它所需要的檔案,如果用了<file://,Git>會透過網路傳輸的方式(效率比較低),使用file://主要是你需要一個不包含無關引用或物件的乾淨倉庫副本時後(?),

要添加一個本地資料庫的指令是。

git remote add local\_proj /opt/git/project.git

‘

2.SSH協定

解決中間人攻擊(存在另一個人或者冒充真正的伺服器接收用戶傳給伺服器的資料,再冒充用戶把資料傳給伺服器)

兩種協議方式。

2.1提供兩個安全驗證,知道帳號密碼,就可以登入到遠端主機,並且所有資料加密(無法防止中間人攻擊) 。

2.2用金鑰加密,把public金鑰丟到需要存取的伺服器上,然後傳過去一把公開金鑰,用戶會像伺服器提出請求,請求用戶的公開金鑰加密,這時伺服器會對兩組金鑰一不一樣,一樣就是用用戶public金鑰加密傳給用戶,用戶再用自己的私有金鑰開啟。

你需要透過SSH複製一個Git倉庫的指令是

git clone ssh://user@server/project.git

優點:你想要倉庫的讀寫權,SSH架設比較簡單,SSH比較安全,SSH在傳輸前會壓縮所以比較快,守護排成比較容易

缺點:沒有匿名性,

3.Git協議,這是一個包含在Git軟體的一個特殊守護進城,他會監聽SSH的特定埠(9418),而且無需任何授權,若要使用Git協議就必須創造git-daemon-export-ok檔,他是提供git協定倉庫服務的必要條件-但除此之外沒有任何安全措施,要就所有人可以clone這個儲存庫,要就所有人不能,這就表示這個協議不會用來推送,可以反向思考,一但所有人知道URL檔所以人都可以進行推送許可權。

優點:Git是最快的傳輸協定,如果你在提供很大訪問的公共專案,或不需要授權的龐大項目,與SSH傳輸一要,少了加密跟解密

4HTTP/S 協議(?)

HTTP/S(是個用戶端跟伺服器端(網站)的請求合應答(TCP)標準),HTTP跟HTTPS協定方便之處在於架設上的簡便性,只需要把Git的裸倉庫放在HTTP的跟目錄上,配置,每個能訪問Git倉庫所在伺服器都可以進行clone,下面的操過可以允許HTTP對倉庫進行讀取: (?)

$ cd /var/www/htdocs/

$ git clone --bare /path/to/git\_project gitproject.git

$ cd gitproject.git

$ mv hooks/post-update.sample hooks/post-update

$ chmod a+x hooks/post-update

Git透過的 post-update 會默認適合運行的命令(git update-server-info)來確保通過HTTP 的 pull 跟 push

優點:

好架設,幾條必要的命令就可以讓全世界讀到倉庫的內容,花費時間很少

缺點:

對用戶效率很低,clone or pull 可能會花更多時間,而且http傳輸體積比其他協議還大,因為它沒有按需供應