git config --global user.email kexuan5458@gmail.com

git config --global user.name kexuan

to set your account's default identity.

Omit(忽略) --global to set the identity only in this repository.

若把--global改--local ，在這個專案進行操作的時候，就會使用特別幫這個專案設定的使用者名稱及 Email 來進行操作。

git init <資料夾位置> (例 D:/NKK/GitTutorial)

使用 git init 指令初始化這個目錄，主要目的是要讓 Git 開始對這個目錄進行版本控制。

得到->Initialized empty Git repository in D:/NKK/GitTutorial/.git/

或是->Reinitialized existing Git repository in D:/NKK/GitTutorial/.git/

如果這個目錄不想被版控，或是只想給客戶不含版控紀錄的內容的話，只要把那個 .git 目錄移除，Git 就對這個目錄失去控制權了。

什麼檔案或目錄刪了都救得回來，但 .git 目錄只要刪了就沒辦法了。

git status

使用status命令確認工作目錄與索引的狀態

注意！要先進入(使用)cd D:/NKK/GitTutorial

才使用status得到以下訊息：

--------------------------------------------------------------------------------------------------

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

GitHub\_Tutorial.txt

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

--------------------------------------------------------------------------------------------------

git add <file1 file2 file3> (沒有< >，例 add GitHub\_Tutorial.txt)

將檔案加入到索引

在工作目錄上做的任何變更並不會直接提交到數據庫。

Git在執行提交的時候，不是直接將工作目錄的狀態儲存到數據庫，

而是將索引的狀態儲存到數據庫。

因此，要提交變更，首先必需要把變更內容加入到索引中。

git add .

表示可以把當前目錄下所有的檔案加入到索引

再一次使用git status 確認GitHub\_Tutorial.txt是否已加入索引

正確應顯示：

--------------------------------------------------------------------------------------------------

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: GitHub\_Tutorial.txt

--------------------------------------------------------------------------------------------------

git commit -m "<欲提交的訊息>"

執行commit命令提交

請務必輸入提​​交訊息，因為在訊息空白的狀態下執行提交是會失敗的。

Git的標準提交訊息：

第1行：提交時修改內容的摘要

第2行：空行

第3行以後：修改的理由

(例 git commit -m "first commit on 8/8")

--------------------------------------------------------------------------------------------------

[master (root-commit) 2a024b4] first commit on 8/8

1 file changed, 4 insertions(+)

create mode 100644 GitHub\_Tutorial.txt

--------------------------------------------------------------------------------------------------

此時再試一次git status會得到：

--------------------------------------------------------------------------------------------------

On branch master

nothing to commit, working tree clean

--------------------------------------------------------------------------------------------------

git log

確認數據庫的歷史提交記錄

存檔-> add-> commit-> push 上去 一定要先add 再commit！！

若有衝突-> 先pull 下來-> 再push 上去

git push

git push <repository> <refspec>

為了將本地端數據庫的修改歷史共享到遠程數據庫

必須上​​傳本地端數據庫中儲存的修改歷史。

refspec:指定要push的分支

(例 git push -u Orgin master)(Orgin是自創的遠端數據庫別名)

git clone

git clone <repository> <directory>

執行複製後，可以把遠端數據庫裡的內容全部下載下來

修改歷史也會被複製到本地數據庫裡，如此，可以查看歷史記錄和提交

repository:指定遠端數據庫的URL

directory:指定要複製的目錄名稱(載下來後 叫做kitty的資料夾)

(例 git clone 網址 kitty)

git pull

執行pull之後，會從遠端數據庫下載最新的修改歷史

git remote add <name> <url>

name:可以為遠端數據庫取個適合的別名或暱稱，這樣以後push的時候就不需要每次輸入網址

(例 git remote add origin https://github.com/kexuan5458/Git\_Tutorial\_0821

在主控台，執行push或pull命令時想省略遠端數據庫名稱的話，會使用origin名稱作為遠端數據庫。

因此一般都會為遠端數據庫命名為origin。

git checkout -b <branch>

(例 git checkout -b issue1)

可建立issue1的分支，並且切換過去issue1

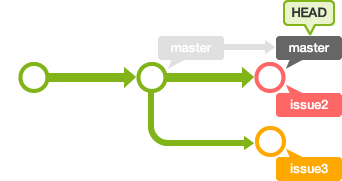
(例 git checkout issue1)

單純建立分支

合併

範例1:

有issue2和issue3，最後都要合併成master



git checkout master

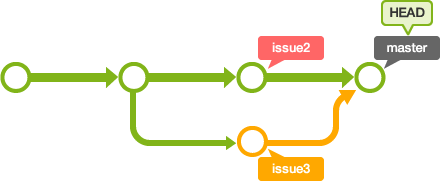
git merge issue2 (切至我要合併成的master，先與issue2合併)

git merge issue3(再與issue3合併。但是有衝突)

修改再重新add和commit

git add myfile.txt

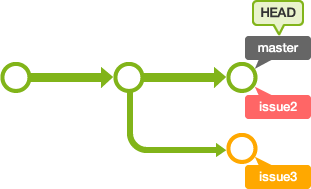
git commit -m "合併issue3" (全部都在master中做)



--------------------------------------------------------------------------------------------------

範例2:

有issue2和issue3，最後issue3成為最終master



切換到 issue3 分支後，對 master 執行 rebase

git checkout issue3

git rebase master (發現有衝突)

修改再重新add和rebase

修改好衝突後，若想要繼續 rebase的操作，在rebase後加上 --continue。

若要取消 rebase 的話，請使用 --abort。

git add myfile.txt (我在issue3之中)

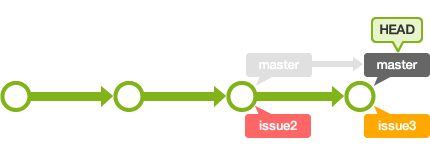
git rebase --continue

此時，在 master 分支的 issue3 分支就可以 fast-forward 合併了

切換成master後執行與issue3的合併

git checkout master

git merge issue3



pull

fetch

push

pull的內容就是fetch + merge的內容所組成的。