第一次 必須好好跟config溝通

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

git commit --amend -m "add cat 2"

如果要修改最後一次的 Commit 訊息，只要直接在 Commit 指令後面加上 --amend 參數即可

如果要修改更早的commit紀錄，就要使用rebase

[rebase教學](https://gitbook.tw/chapters/rewrite-history/change-commit-message)

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

準備把東西推上遠端的 Git 伺服器了。首先，需要設定(加入)一個端節的節點

git remote add origin [git@github.com:kexuan5458/Git\_Turtorial\_0821.git](mailto:git@github.com:kexuan5458/Git_Turtorial_0821.git)

在慣例上，遠端的節點預設會使用 origin 這個名字。如果是從 Server 上 clone 下來的話，它的預設的遠端節點就會叫 origin。也可以改叫buffalo

git push -u origin master

這個簡單的 Push 指令其實做了幾件事：

1. 把 master 這個分支的內容，推向 origin 這個位置。
2. 在 origin 那個遠端 Server 上，如果 master 不存在，就建立一個叫做 master 的同名分支。
3. 但如果本來 Server 上就存在 master 分支，便會移動 Server 上 master 分支的位置，使它指到目前最新的進度上。
4. 設定 upstream，就是那個 -u 參數做的好事，這個稍候說明。

關於upstream

在 Git 裡，每個分支可以設定一個「上游」(最多一個) ，它會指向並追蹤（track）某個分支。

舉例：git push -u origin master

就會把 origin/master 設定為本地 master 分支的 upstream。

當下回執行 git push 指令而不加任何參數的時候，它就會猜你是要推往 origin 這個遠端節點，並且把 master 這個分支推上去。

這個git push 指令執行時，好像是用 HTTPS 的方式，所以需要輸入 GitHub 的使用者帳號及密碼。

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。

Repository:數據庫的別名

refspec:指定要push的分支(我目前操作的其中一個分支叫master)

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

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、分支、標籤和提交

repository:指定遠端數據庫的URL

directory:指定要複製的目錄名稱(例如 載下來後變成叫做hello-kitty的資料夾)

(例 git clone [git@github.com:kexuan5458/Git\_Turtorial\_0821.git](mailto:git@github.com:kexuan5458/Git_Turtorial_0821.git) hello-kitty)

如果你在 GitHub 上看到某個專案很有趣，想要下載回來看看，只要使用 Clone 指令就可以把整個專案複製一份回來了。

git pull

執行pull之後，會從遠端數據庫下載最新的修改歷史。其實就是git fetch + git merge的功能

git pull --rebase

加上 --rebase 參數，它在 Fetch 完成之後，便會使用 rebase 方式進行合併

(前面內容的補充)

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。

怎麼有時候推不上去…

1. Sherly 跟 Eddie 兩個人在差不多的時間都從 Git Server 上拉了一份資料下來準備進行開發。
2. Sherly 手腳比較快，先完成了，於是先把做好的成果推一份上去。
3. Eddie 不久後也完成了，但當他要推上去的時候發現推不上去了…

第一招：先拉再推:

git pull --rebase「內容抓下來之後請使用 rebase 方式合併」

合併如果沒發生衝突，接著應該就可以順利往上推了。

第二招：無視規則，總之就是聽我的！

git push -f

只要加上了 --force 或是 -f 參數，它就會強迫硬推上去，把 Sherly 之前的內容蓋掉

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 branch issue1)

單純建立分支

git branch -m issue1 tiger1

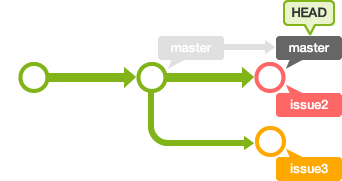
分支改名

合併

[複雜的合併](https://gitbook.tw/chapters/branch/merge-branch)

範例1:

有issue2和issue3，我想要 master 分支來合併其他分支的話，我會先切回 master (分支最後都要合併成master)



git checkout master

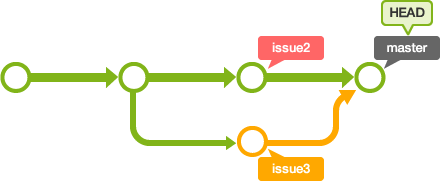
git merge issue2 (切至我要合併成的master，先與issue2合併)

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，都是master在一直往前走)



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

範例2:

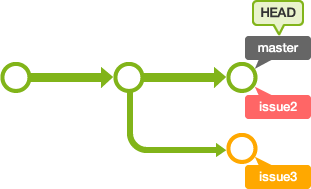
[複雜的rebase以及挽救方法](https://gitbook.tw/chapters/branch/merge-with-rebase)

「rebase」是「re」加上「base」，翻成中文大概是「重新定義分支的參考基準」

所謂「base」就是指「你這分支是從哪裡生出來的」

所以git rebase dog 大概就是「我，就是 cat 分支，我現在要重新定義我的參考基準，並且將使用 dog 分支當做我新的參考基準」的意思

現在有issue2和issue3，最後要讓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)

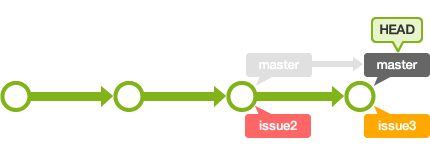
git rebase --continue

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 (是master要往前走)

git merge issue3



如果後悔了，想取消rebase，git reset ORIG\_HEAD--hard，就會回到rebase之前。

刪除

rm index.html

git add index.html

兩個指令可以合成同一個：

git rm index.html

「我不是真的想把這個檔案刪掉，只是不想讓這個檔案再被 Git 控管了」的話，可以加上 --cached 參數

(例 git rm welcome.html --cached)

誤刪的時候:

git checkout index.html (或 git checkout . 找回誤刪的所有檔案)

會把暫存區（Staging Area）裡的內容或檔案，拿來覆蓋工作目錄（Working Directory）的內容或檔案。它會回復到上一次 Commit 的狀態。

git checkout HEAD~2 .

意思就是「拿距離現在兩個版本以前的檔案來覆蓋現在工作目錄的檔案，同時也更新暫存區裡的狀態」。

改名

mv index.html hello.html

git add hello.html

兩個指令可以合成同一個：

git mv index.html hello.html

雖然只是改檔名，但對 Git 來說會被認為是兩個動作，一個是刪除index.html 檔案，一個是新增 hello.html 檔案（變成 Untracked 狀態）。