Udemy Git – GitHub Course Notes

Mortimer Sotom – [Mortimer.sotom@gmail.com](mailto:Mortimer.sotom@gmail.com)

Git is a decentralised, distributed version control system.

# Overview & Objectives

* Git basics
* States
* Commits & comments
* Edits
* Gitignore
* GitHub Vs command line
* SSH authentication
* Branching and merging
* Remotes
* Pull requests
* Resolving conflicts (P4Merge)
* Comparing differences
  + Commits
  + Pull requests
  + Branches
  + Tags
* Tags
* Issues
* Gists
* Organisations

# Basics

## Initialisation

* cd to desired path folder
* Create new empty repo: *git init RepoName*

## Git States

* Working dir
  + Contains all files and folders
* Repository
  + All commits and saved changes
* Staging area
  + Middle state

## New repo

1. Create new repo on GitHub
   1. Get remote from GH
2. Move to folder in local repo in Bash
3. Paste 'git remote add origin …'
   1. origin is the name of the remote
   2. By convention first remote is called origin
4. Local repo is now linked to GH

## Pushing changes to GH

* Command: git push -u origin master --tags
  + sets up a tracking branch relationship between 'master' branch on local and remote repos 'origin'
  + Also pushes tags to GH
* For next changes: -u not needed

# Git Commands

Machine generated alternative text:
CREATE 
Clone an existing repository 
$ git 
Create a new local repository 
$ git init 
LOCAL CHANCES 
Changed files in purworking director,' 
$ git status 
Changes to tracked files 
$ git diff 
Add all current changes to the next commit 
$ git add . 
Add changes in to the next cunmit 
$ git add -p •z-file> 
Commit all local changes in tracked files 
$ git cc—it -a 
Commit previously staged changes 
$ git 
Change the last commit 
Don •r amend published commits' 
$ git ccmit - -anend 
COMMIT HISTORY 
Shcw all commits. starting with neaeSt 
$ git log 
Shcw changes over time for a specifi file 
$ git log -p •file> 
Win changed what and when in 
$ git bl•e 
BRANCHES TAGS 
List all existing branches 
$ git branch -av 
Switch HEAD branch 
$ git checkout 
Create a new branch based 
on your current HEAD 
$ git branch enew-branch> 
Create a new tracking branch based on 
a remote branch 
$ git checkout • •track cremte/bran• 
a local branch 
$ git branch •d 
Mark current commitwith a tag 
$ git tag 
UPDATE PUBLISH 
List all currently configured remotes 
$ git re•ote •v 
Show information about a remote 
$ git re•ote sht:w 
Add remote repository, named cremote> 
$ git remote add cshortnææ• curb 
&wnload all changes from < remote>, 
but don't integrate into HEAD 
$ git fetch 
Ckmnload changes and directry 
merge/ integrate into HEAD 
$ git pull cremte> 
Publish local changes on a remote 
$ git push cremte> 
Delete a branch on the remote 
$ git branch -dr cremote/branch> 
Publish your tags 
$ git push • •tags 
MERGE G REBASE 
Merge into your current ACACI 
$ git —rge 
Rebase your cu rrent HEAD onto 
Don? rebase published commits! 
S git rebase 
Abort a rebase 
S git rebase --abort 
Continue a rebase after resaving conflicts 
S git rebase - -continue 
Use your configured merge tool to 
solve conflicts 
$ git —rgetool 
Use your editor to manual* solve conflicts 
and (after resolving) mark file as resaved 
$ git add cresolved-file> 
$ git 
UNDO 
Discard all local changes in your working 
directory 
S git reset --hard HEAD 
Discard local changes in a specific file 
S git checkout HEAD 
Revert a commit (by producing a commit 
with contrary changes) 
$ git revert ccomit> 
Reset your HEAD pointer to a previous commit 
...and discard all changes since then 
S git reset --hard 
...and preserve all changes as unstapd 
changes 
$ git reset 
..and greserve urzornmitted changes 
$ git reset - -keep 

From: <https://www.git-tower.com/blog/git-cheat-sheet>

Visual Git Cheatsheet - <http://ndpsoftware.com/git-cheatsheet.html#loc=index>

For seeing commands available on each git "layer"

# Undoing Changes

After editing a file and staging it with *git add*, *git reset HEAD fileName* to unstage it.  
To undo stages, revert to last previous version of that file which is in the Git repo

*git checkout – filename*

## Git log options

* *--oneline* simplified display of commits
* *--graph* adds a \* in a graph structured way for branching hierarchy
* *--decorate* tells which branch each commit belongs to
* *--all*  provide history for all available branches

## Transform above commands into a 1 work alias:

*git config --global alias.hist "log --oneline --graph --decorate --all"*

Global to make it at user level

Alias.<macroName>

Check it is saved in user functions:

*git config --global --list*

# Rename & Delete Files & Ignore

To rename *myFile.xxx : (git) mv myFile.xxx newFileName.xxx*

*mv* bash command renames it if doesn't change directory

Use *git* to log it in Git and possibly *commit* --> Git know the file has been renamed

Using *mv* without *git* command --> Git sees this as a deletion of original file and creation of a new one

*git add -A* to update all types of possible changes

## To ignore files:

Create a *.gitignore* file

Syntax is 1 line per file to ignore

Can target specific files: *fileToIgnore.bla*

Can target extensions: \*.bla

# SSH Authentication

* Set up on trusted personal computer
* Ease when making changes in remote repo

## SSH vs HTTPS

* HTTPS set up requires Git username and pwd for each push
* Both as secure as each other

## Setup:

1. Create a new *.ssh* directory
2. *cd* into dir
3. Machine generated alternative text:
   æ/Documents/Gi tHub Repos/Gi tpractice/. ssh (master) 
   Mortzwortz MINGW64 
   S ssh-keygen -t rsa -C "mortimer . sotom@gmail . com" 
   Generating public/private rsa key pair. 
   Enter file in which to save the key (/c/Users/MortZ/ .ssh/id_rsa): 
   Enter passphrase (empty for no passphrase): 
   Enter same passphrase again: 
   Your identification 
   Your public key has 
   The key fingerprint 
   SHA256: 
   The key's randomart 
   +---[RSA 2048] 
   - [SHA256] 
   has been saved in /c/Users/MortZ/.ssh/id_rsa. 
   been saved in /c/Users/MortZ/ .ssh/id_rsa.pub. 
   •s: 
   Image •s: 
   mortimer . sotom@gmai 1 . com 

*t* for type (rsa).  
*C* for common name.  
Passphrase is a password to access all data of the SSH key.

1. Open content of *id\_rsa.pub* key, copy contents.
2. Go to settings in GH and add SSH key by pasting.
3. Machine generated alternative text:
   æ/Documents/Gi tHub Repos/Gi tpractice (master) 
   Mortzwortz VINGW64 
   S ssh -T git@github.com 
   The authenticity of 
   host 'github.com (192.30.253.113)' can't be established. 
   RSA key fingerprint 
   is SHA256•. 
   Are you sure you want to continue connecting (yes/no)? yes 
   Warning: permanently added 'github.com,192.30.253.113' (RSA) to the list of know 
   n hosts. 
   Enter passphrase for key '/c/Users/MortZ/.ssh/id_rsa' : 
   Hi MortZx! You 've successfully authenticated, but GitHub does not provide shell 
   access . 

# Comparing Differences

To compare differences between 2 commits or 2 branches

* *git hist* to get list of commits with their refs
* *git diff <ref1> <ref2>*

Shows the edits or changes to a file!

Use *difftool* to launch associated difftool (p4merge) to show the same changes but with p4merge

Machine generated alternative text:
æ/Documents/Gi tHub Repos/Gi tpractice (master) 
Mortzwortz MINGW64 
S git difftool ff45e83 HEAD 

In GH: pull request --> new pull request:

Can compare branch on time line with the following syntax:

* *<branchName>@{<timeInterval>}*
* time interval eg: 3days or Y-M-D

Machine generated alternative text:
æ/Documents /Gi tHub 
Mortzwortz MINGW64 
S git config 
--global alias.hist "10g 
æ/Documents /Gi tHub 
Mortzwortz MINGW64 
git hist 
Repos/Gi tP racti ce 
-one] ine 
-graph 
Repos/Gi tP racti ce 
(mas ter) 
-decorate 
(mas te r) 
* 
3a22e5a 
336abb3 
44ffc3a 
266143d 
* 
e885aa6 
18a18cd 
* ccOfdae 
6b055aO 
ff45e83 
38fedd8 
(HEAD -> master, origin/master) adding ignore file 
remove aFi1e. txt 
rename and add new file 
deleting new file 
renaming new file 
adding empty txt file 
added OneNote notebook as pdf 
update REAWE 
Adding default license file For future ref 
REAWE file 
æ/Documents/Gi tHub Repos/Gi tpractice 
Mortzwortz MINGW64 
S git diff ff45e83 HEAD 
diff --git a/.gitignore b/.gitignore 
new file mode 100644 
index 0000000.. bf0824e 
/dev/nul 1 
b/.gitignore 
\ No newline at end of file 
diff --git a/LICENSE.md b/LICENSE. txt 
similarity index 100% 
rename from LICENSE .md 
rename to LICENSE. txt 
diff --git a/REAWE.md b/REAWE.md 
index e080a8e.. abe61dc 100644 
a/REAWE . md 
-1,4 +1,5 
# Git practice project README 
This is a simple readme file. 
A repo for practicing git commands and version control . 
\ No newline at end of file 
• ...skipping... 
diff --git a/.gitignore b/.gitignore 
new file mode 100644 
(mas te r) 

# Branching & Merging

A branch is a timeline of commits --> Names or labels given to timelines in git

Can be created or deleted without affecting timelines

## TYPES OF MERGES

**Fast Forward**

* Simplest case when no additional work detected on master branch
* Works as a simple chain of commits
* See an example below

**Automatic**

* When git detects non-conflicting changes with the parent branch
* Timeline of parent branch remains unchanged with a simple commit of merged branch

**Manual**

* When git can't automatically resolve conflicts
* Git enters a conflicting merge state: all conflicting mergers must be solved before committing
  + When resolved, changes saved as a merge commit

Machine generated alternative text:
æ/Documents/Gi tHub Repos/Gi tpractice (master) 
Mortzwortz MINGW64 
S git merge tempbranch 
Auto-merging README .md 
CONFLICT (content): Merge conflict in README .md 
Automatic merge failed; fix conflicts and then commit the result. 

* *cat <filename>*  will print the file and show conflicts between branches

Machine generated alternative text:
æ/Documents/Gi tHub Repos/Gi tpractice 
Mortzwortz MINGW64 
S cat REAWE.md 
# Git practice project README 
This is a simple readme file. 
A repo for practicing git commands and version control . 
HEAD 
Conflicting change on master branch. 
Conflicting edit on temp branch. 
tempbranch 
(mas ter I MERGING) 

* *git mergetool*  to bring up p4merge, choose which option to keep and accept changes
* This might create a .orig file (original) which should be add to .gitignore file

## Commands

Show all branches in repo

Machine generated alternative text:
Mortzwortz MINGW64 
S git branch 
mas ter 
æ/Documents/Gi tHub Repos/Gi tpractice (master) 

Create a new branch:

* Checkout changes to said branch
* -b creates a new branch

Machine generated alternative text:
æ/Documents/Gi tHub Repos/Gi tpractice 
Mortzwortz MINGW64 
S git checkout -b testingbranch 
Swi tched to a new branch 'testingbranch' 
REAWE . md 
(mas te r) 

Merging branches:

* Give name of branch to merge with current one

Machine generated alternative text:
æ/Documents/Gi tHub Repos/Gi tpractice 
Mortzwortz MINGW64 
(mas te r) 
S git merge testingbranch 
Updating 3a22e5a. .1cc9ba9 
Fast-forward 
REAWE.md 4 
1 file changed, 3 insertions (+) , 
1 deletion( 

Deleting branches:

* -d for delete

Machine generated alternative text:
æ/Documents/Gi tHub Repos/Gi tpractice (master) 
Mortzwortz MINGW64 
S git branch -d testingbranch 
Deleted branch testingbranch (was Icc9ba9) . 

C:\Users\MortZ\AppData\Local\Temp\msohtmlclip1\02\clip_image001.png

* -u creates a tracking branch so it is In synch with branch master
* no need to specify remote or branch when pushing from a tracking branch

Git pull pulls for current branch.

* --all pulls from all branches.

C:\Users\MortZ\AppData\Local\Temp\msohtmlclip1\02\clip_image001.png

To delete a remote branch:

* git push <remoteName> :<branchName>

Machine generated alternative text:
æ/Documents/Gi tHub Repos/Gi tpractice 
Mortzwortz MINGW64 
S git push origin :update-d 
To https://gi thub . com/MortZx/Gi t-practice . git 
[deleted] 
update-d 
(mas te r) 

After deleting a branch, git might still hold a reference to it.

* -p is the prune option: removes reference to any dead references

Machine generated alternative text:
æ/Documents/Gi tHub Repos/Gi tpractice 
Mortzwortz MINGW64 
S git fetch -p 
From https://gi thub . com/MortZx/Gi t-practice 
[deleted] 
(none) 
-> origin/another8ranch 
(mas te r) 

## Pull with Rebase

Rebase rewinds current commits on current branch to a point where the branch you're merging diverged. Playing back commits happened on branch bringing in & playing back any commits on branch currently on.

Any changes include changes made on remote branch while keeping changes on current one.

Machine generated alternative text:
'Mortzwortz MINGW64 
æ/Documents/Gi tHub Repos/Gi tpracti ce/some- 
S git pull 
-rebase 
First, rewinding head to replay your work on 
top of it... 
Applying: edit code locally before rebase 
(mas te r) 

## **Forking**

When working on a collaborative project from an existing repo, fork the repo and work on a different branch than master, merge the branches when desired. To merge your fork with the main repo, get the link of the main repo (not your forked repo) and add the remote:

C:\Users\MortZ\AppData\Local\Temp\msohtmlclip1\02\clip_image001.png

Before making the merge, make sure to pull from that original repo:

C:\Users\MortZ\AppData\Local\Temp\msohtmlclip1\02\clip_image002.png

# Tags (Special Markers)

Tags point to commits.

HEAD: last commit of current branch

HEAD can be manually moved to another location other than last commit --> see future section

## 2 types of tags:

Lightweight: no association with it.

Annotated tag:

## Commands

*git tag <tagname>* lightweight tag

*git tag -d <tagname>* delete a tag

*git tag -a <tagname> -m "commit message"* associates a commit message to a tag

*git tag --list* list all tags

*git show <tagname>*  show commit linked to an annotated tag

C:\Users\MortZ\AppData\Local\Temp\msohtmlclip1\02\clip_image001.png

Shows all info relevant to that tag: tagger, date, message, commit related info

*git push* does not push tags by default, it must be specified

C:\Users\MortZ\AppData\Local\Temp\msohtmlclip1\02\clip_image002.png

* *git push <remoteName> <tagName>*

C:\Users\MortZ\AppData\Local\Temp\msohtmlclip1\02\clip_image003.png

Deleting a tag on the local repo doesn't delete on the remote repo

C:\Users\MortZ\AppData\Local\Temp\msohtmlclip1\02\clip_image004.png

* *git push <remoteName> :<tagName>*

To move an existing tag to another commit

C:\Users\MortZ\AppData\Local\Temp\msohtmlclip1\02\clip_image005.png

* *git tag -f <tagName> <commitID>*

C:\Users\MortZ\AppData\Local\Temp\msohtmlclip1\02\clip_image006.png

* Force push to update location of tag (commitID assigned to it)

# Stashing – Saving WIP

WIP: work in progress.

Stashing saves work in progress and links it to the last commit

ortZ@VortZ V ING'•.'ÉA —'Documents/Gi tHub Repos/GitPractice (master) 
S git stash 
Saved working directory and index state WIP on master: 80eeb5c update 
to exclude . orig fi les 
ortZAMortZ VINGn64 —/Documents/Gi tHub Repos/GitPractice (master) 
S git stash list 
. giti gnore 
WIP an master: 80eeb5c update .qitiannre to exclude . oriq files 

Modified files become unstaged and back to a clean working directory.

Machine generated alternative text:
æ/Documents/Gi tHub Repos/Gi tpractice (master) 
Mortzwortz MINGW64 
S git stash pop 

 2 actions in 1: *git apply git drop*

* 1. Apply the last stash, recover changes
  2. Drops the last stash, it is no longer stashed



# Time Travel

## 3 types of resets:

* **Soft** - least destructive, changes where HEAD is pointing (pass ref of a commit)
  + Changes commit ID that HEAD is pointing to.
  + Preserves staging area and working directory

C:\Users\MortZ\AppData\Local\Temp\msohtmlclip1\02\clip_image001.png

* **Default** (mixed)
  + Changes commit ID that HEAD is pointing to
  + Unstages changes to working directory

Machine generated alternative text:
æ/Documents/Gi tHub Repos/Gi tpractice 
Mortzwortz MINGW64 
S git reset 3a22e5a --mixed 
Unstaged changes after reset: 
. giti gnore 
LICENSE . txt 
REAWE . md 
(mas te r) 

* **Hard**
  + Most destructive
  + Discards changes and any files in staging area

Machine generated alternative text:
æ/Documents/Gi tHub Repos/Gi tpractice (master) 
Mortzwortz VINGW64 
S git reset 6b055aO --hard 
HEAD is now at 6b055aO Update README 

*git log* shows all commit IDs

*git reflog* shows all actions taken in repo including any reset actions