Youtube links for video for Github beginners:

1. #1Why use Git: <https://www.youtube.com/watch?v=3RjQznt-8kE>
2. #2 installing Git: <https://www.youtube.com/watch?v=MFtsLRphqDM>
3. #3 How Git works: <https://www.youtube.com/watch?v=iNP_KmOFqXs>
4. #4 Creating a Repository: <https://www.youtube.com/watch?v=v0Ch3yWQ-Zc>
5. #5 staging files: <https://www.youtube.com/watch?v=KngvG8WzYLU>
6. #6 making Commits: <https://www.youtube.com/watch?v=Fhgga2s_RmM>
7. #7 undoing things: <https://www.youtube.com/watch?v=RIYrfkZjWmA>
8. #8 branches: <https://www.youtube.com/watch?v=QV0kVNvkMxc>
9. #9 Merging branches: <https://www.youtube.com/watch?v=XX-Kct0PfFc>
10. #10 Introduction to GitHub: <https://www.youtube.com/watch?v=XX-Kct0PfFc>
11. #11 Collaborating on GitHub: <https://www.youtube.com/watch?v=MnUd31TvBoU>
12. #12 Forking(contributing on opensource): <https://www.youtube.com/watch?v=HbSjyU2vf6Y&t=329s>

* Git set up on personal computer

1. Check git version installed

git –-version

1. Config git username on your own computer

git config --global user.name imgjiang

config --global user.email [guangfei@gmail.com](mailto:guangfei@gmail.com)

2) retrieve user.name

git config user.name

1. Initialize a directory into a git repository

git init

1. Git commit process

Modify | staging | commit

Git status | git add filename; git status (check again); git rm –cached filename | git commit -m “descriptive commit message”

Git status : Check which file changed, staging etc.

Git add filename (.): add filename into staging (git add . : add all files under current folder into stage)

Git rm –- cached filename: remove filename from staging step

Git commit -m “commit message”: commit your staging file

Git log : commit history

git log –-oneline: commit history one line for each commit

1. Modify (revert commit)

Checkout commit | revert commit | reset commit

Git checkout commit(id in git log –oneline): checkback what kind of codes had been submitted; come to specific commit – can create branches at that point

Git revert commit (id in git log –oneline): revert your previous commit(id), by git log: still keep the commit history, also add 1 ‘revert commit’ history (seems to delete from my files)

Git reset commit(id in git log –oneline): The commit not exist after ‘id’, but contents still exists in your editor.

Git reset commit(id in git log –oneline) –hard: delete commit history and also contents in your computer editor. (does not change anything)

* Need to confirm about git revert; git reset; git reset --hard

1. Branches:

usage

1. Test features
2. Collaborate from different colleagues to from different branches with different features
3. Merge to master features after tests

Process of using it:

Git branch feature-1(branch name)

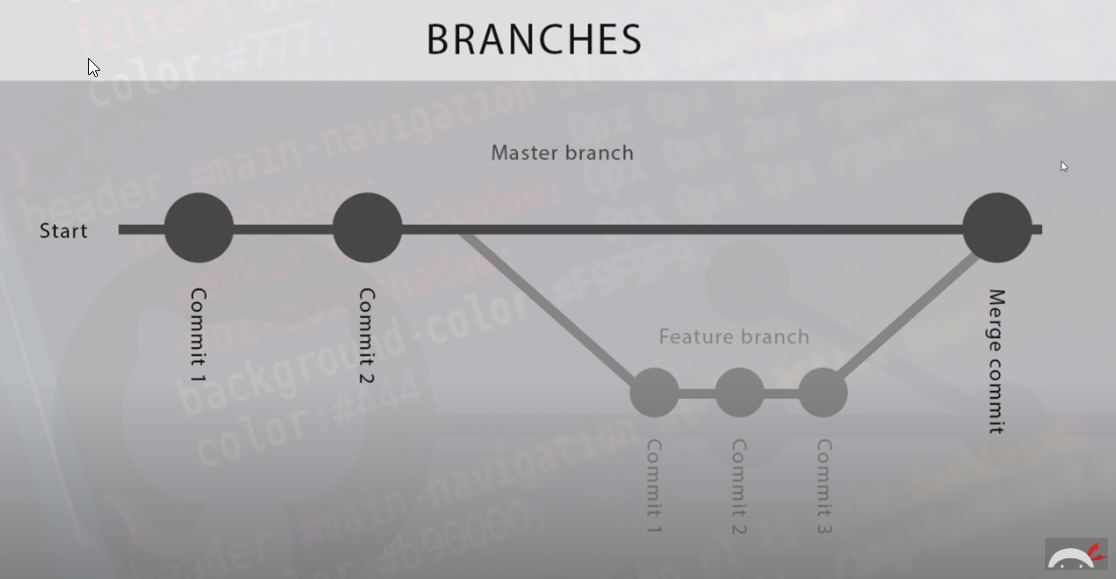
Git branch -a (list all the branch and check which branch that you are on: \*green)

Git checkout feature-1 (switch to feature-1)

Git checkout master (switch back to master branch; then delete)

Git branch -D feature-1 (delete branch feature-1)

git checkout -b feature-a (create branch feature-a and switch to feature-a)



1. Merge branches into master branch

Switch to master branch first:

Git checkout master

Git merge feature-a(branch name)

--- if there are conflicts:

1. Resolve conflicts
2. Remove conflicts mark and modify the code
3. In the master branch, git add. -> git commit
4. Push your code into github

Git push (github repository url https://xxx) master(branch name?) – might need login credentials

Git remote add <https://url> alias (origin for example)

Git push origin master

### Permanently authenticating with Git repositories

Run the following command to enable [credential caching](https://help.github.com/articles/caching-your-github-password-in-git/#platform-linux):

$ git config credential.helper store

$ git push https://github.com/owner/repo.git

Username for 'https://github.com': <USERNAME>

Password for 'https://USERNAME@github.com': <PASSWORD>

You should also specify **caching expire**,

git config --global credential.helper 'cache --timeout 7200'

After enabling credential caching, it will be cached for **7200 seconds (2 hour)**.

1. Collaborate with different colleagues:
2. Git checkout master (if going to pull down master branches)
3. Git pull url (master branch url)
4. Git checkout -b branchname (create a new branch) – can filename same as branch name?
5. Modifiy the branch contents…
6. Git add .
7. Git commit -m “descriptive commit comment”
8. Git push origin branchname

Origin: url alias (pulled down Github master location)

Branchname: branch just created and worked on

1. Go to github url
2. Click on master url, see if there are any conflicts or not – can comment on any update needed, also comment on each line of the code; can see branch and files commited
3. Everything looks good: click ‘merge pull request’ green button
4. If ‘Pull request successfully merged and closed’ 🡪 can click ‘delete branch’ button
5. Pull github repository down, modify it then upload

Cd .. (to upper folder, for another respository)

git clone <https://github.com/Guangfei0/git-two.git> (url cloned from your github https)

cd new repository location

git add .

Git commit

git remote -v (check alias)

git origin master

1. Fork 🡪 contribute to open source github location
2. Click ‘fork’ 🡪 This repository will come to my github location
3. Git clone url & do all the modifications & push the modifications back to your own github location 🡪 This step is the same as working on your own repository
4. Come to the original github page (other people’s open source)

Click ‘New pull request’ button

If ‘able to merge’, click ‘Create pull request’

But at this step, you can’t merge it to the original branch etc; the creator of this project need to review and do the merge. He/she will click ‘merge pull request’ then ‘confirm merge’

1. Delete github repository
2. Navigate to the repository main page
3. Click ‘setting’
4. Delete repository (danger zone)
5. Command line

Cd ..

Cd ./directory

Dir/ls (‘ls’ not working in windows machine)

Mkdir

Rm

Rmdir

Touch filename

Copy file1 file2 (copy file1 to file2)

Git practice before work:

Repeat the process of using GIT:

1. Clone one code from git
2. Make some changes to the code/document
3. Steps on merging data into github
4. Create branch
5. Push
6. Commit

**In command line, type ‘git’, get the following:**

usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]

[--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]

[-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]

[--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]

<command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)

clone Clone a repository into a new directory

init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)

add Add file contents to the index

mv Move or rename a file, a directory, or a symlink

restore Restore working tree files

rm Remove files from the working tree and from the index

sparse-checkout Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)

bisect Use binary search to find the commit that introduced a bug

diff Show changes between commits, commit and working tree, etc

grep Print lines matching a pattern

log Show commit logs

show Show various types of objects

status Show the working tree status

grow, mark and tweak your common history

branch List, create, or delete branches

commit Record changes to the repository

merge Join two or more development histories together

rebase Reapply commits on top of another base tip

reset Reset current HEAD to the specified state

switch Switch branches

tag Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)

fetch Download objects and refs from another repository

pull Fetch from and integrate with another repository or a local branch

push Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some

concept guides. See 'git help <command>' or 'git help <concept>'

to read about a specific subcommand or concept.

See 'git help git' for an overview of the system.