Git and GitHub

- ✓ **Introduction**
- ✓ **GitHub Administration (Org/Repo/Users/Team Creation)**
- ✓ **GIT Commands**
- ✓ **.gitignore file**
- ✓ **Create Branch**
- ✓ **Create Tag / Releases**
- ✓ **Personal Access Token Generation**
- ✓ **SSH Key Generation**
- ✓ **Git Branching Strategy**
- ✓ **Pull Request Creation and Merging**
- ✓ **Git Hooks**
- ✓ **README.md file**
- ✓ **Git Best Practices**

Introduction

GitHub Administration

Pre Requisite Software Download/ Registration :

- ❖ **Register @ https://github.com/**
- ❖ **Install git bash @ https://git-scm.com/downloads**

Git Commands

When you install Git-bash, the first thing you should be doing is setting up your user details as follows only one time.

```
#git config --global user.name "LWP Technology"
#git config --global user.email "mylwp@gmail.com"

#git config –list
# git config --global –list
```

Task 1: Create the git local repository in local machine (Laptop/Desktop), add one file (DBConnect.java) and update that file, create the github remote repository (https://github.com) and move the local code to github repository.

Go to the directory where you want to create the git repository.

```
cd ~/Desktop
```

mkdir git-code
#cd git-code
#git init : Create a local Git empty repository.
Initialized empty Git repository in /Users/SimonLegah/git/git-code/.git/
#git status : Gives the status of your untracked files.
#touch DBConnect.java
#git status
#vim DBConnect.java
#git add DBConnect.java: Add the files(here DBConnect.java) into your staging area. #git status
On branch master Initial

commit

Changes to be committed:
(use "git rm --cached <file>..." to unstage) new file:

DBConnect.java

#git status

On branch master

nothing to commit, working tree clean

- Open the file (DbConnect.java) and update with some text.

#vim DBConnect.java

#git commit -a -m "Updated DBConnect.java file" : If we use –a along with commit command, no need to execute git add command.
[master 7f795a7] Updated DbConnect.java file 1 file
changed, 1 insertion(+)

- Create the repository in github as follows. Login into github (http://github.com)

On right side top corner click on **"+"** symbol and click on **"New repository"** and give the Repository name and click on Create repository.

#git remote add origin Remote Repo URL : Adding the URL for the remote repository where your local repository code will be pushed.
 # git remote –v :
#git remote show origin : It will give the information on a particular remote (here origin is the remote name)
 # git remote remove origin : It will remove the remote origins.

#git push origin master : Push the changes in your local repository to GitHub remote repository. (Here push is the git command , origin is the remote name and master is the branch name)
Counting objects: 6, done.

Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 479 bytes | 0 bytes/s, done. Total 6
(delta 0), reused 0 (delta 0)
To git@github.com:learnwithpride/test.git
 * [new branch]   master -> master
Branch master set up to track remote branch master from origin.
#git status
 On branch master
Your branch is up-to-date with 'origin/master'. nothing to
commit, working tree clean

#git log : It will give all commit ids.
#git log -2 : It will display only 2 commit ids.

#git show --pretty="" --name-only << Commit ID >> : It will display all the files which are committed in that particular commit.

#git clean -n : It will preview the changes.

#git clean -f : If we want to remove new files from working area.

#git reset <<File Name>> : To untrack the tracked files (revert back to working area from staging area.).

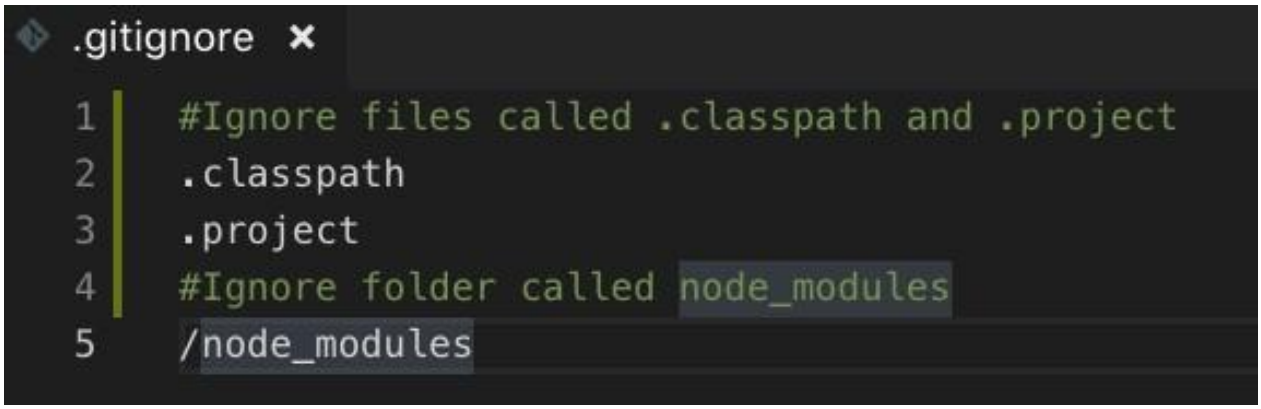
#git revert <<Commit ID>> : It will revert the changes committed in that particular commit id from local repo.
#git push origin master -f: It will revert the changes from remote repo.

What is Version Control System?

When developers are creating something (an application, for example), they are making constant changes to the code and releasing new versions, up to and after the first official (non-beta) release. Version control systems keep these revisions straight, and store the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute.

Git Ignore file (.gitignore)

Some times we don’t want to commit the files, which are generated by IDE like .roject and .classpath files or some node module folders like node_module folder into a git repository.
To ignore these files and folders to commit we will create one file called .gitignore and we will keep the file names or directory names which we don’t want to commit as follows.



https://www.atlassian.com/git/tutorials/saving-changes/gitignore

Branches

Branch

- Branches are used to create another line of development.
- By default, Git has a master branch, which is same as trunk in Subversion (SVN). Usually, a branch is created to work on a new feature.
- Once the feature is completed, it is merged back with the master branch and we delete the branch.

Tags

- Tags similar to branches, but the difference is that tags are immutable.
- It means, tag is a branch, which nobody intends to modify. Once a tag is created for a particular commit, even if you create a new commit, it will not be updated.
- Usually, developers create tags for product releases.

#git branch : It gives the branch names in current repository.
#git branch bugfix : It will create the bugfix branch in local git repository.

#git branch -v: It will display all the branches in your repo, and also tell you what branch you're currently in.

bugfix 87226db initial commit
* master 87226db initial commit
Note: Here * indicate currently in use branch.
 # git checkout bugfix : Switch to bugfix branch. Switched to branch 'bugfix'

Update the land.txt like change 2 – bugfix branch
 # git add . : Add one or more files to staging
 # git commit -m "bugfix commit"
 # git checkout master : Switch to master branch. Switched to branch 'master'

Updat the Bhaskar.txt like change 3 – master branch
 # git add .
 # git commit -m "master commit"

 # git checkout bugfix : Switch to bugfix branch. Switched to branch 'bugfix'

Check the file and see the contents in file.

#git checkout master

#git diff master bugfix

#git merge bugfix
Fix the conflicts
#git add .
#git commit –m "merging"
#git push origin --all : Push all branches code to your remote repository.

#git branch -d bugfix: Deletes the bugfix branch in local repo.
#git push origin : bugfix (OR) git push origin --delete bugfix: It will delete a remote branch in the repository.

Tags

git tag : It will displays the tags.
git tag <<Tag Name>> : It will create the tag.
git push origin tag <<Tag Name>> : It will push the tag to remote repo.
git push origin --tags: It will push all the tags to remote repo.

Note: Tags are not automatically pushed when you push a branch or use the --all option. The -- tags flag sends all of your local tags to the remote repository.

git tag -d <<Tag Name>> : It will delete the tag.

---

git stash: git stash temporarily saves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on. Stashing is handy if you need to quickly switch context and work on something else, but you're mid-way through a code change and aren't quite ready to commit.

git stash (OR) git stash save "Updated some code" :
git stash list :

git stash show : This command shows the summary of the stash diffs. The above command considers only the latest stash.

git stash show -p : It will give you the detailed list of differences. git stash show stash@{1}:
git stash apply stash@{0}:
git stash drop stash@{0} :
git stash list vim land.txt git stash
git stash list
git stash pop: It apply the latest stash and then immediately drop it from your stack.

git stash pop stash@{1} : It apply the particular stash and then immediately drop it from your stack.

---

git cherry-pick: Cherry picking in git means to choose a commit from one branch and apply it onto another.

git log
git branch
git checkout master
cat mylwp.txt
git cherry-pick <<CID>
cat mylwp.txt

git cherry-pick --abort: It will cancel the cherry-pick operation.

When you get the above error while committing the code from local repository to remote repository execute the following command in git bash.

git config --global http.sslVerify false

Steps for Code Checkout into local from Remote Repository

- Open the Gitbash
- Go to the directory where we need to commit the code/checkout the code
  cd C:\LWPTechnologies\JavaWorkspace\MTWorkSpace
- Get the code from Git Repository using below command.

  git clone <<GitHub URL>>

SSH Key

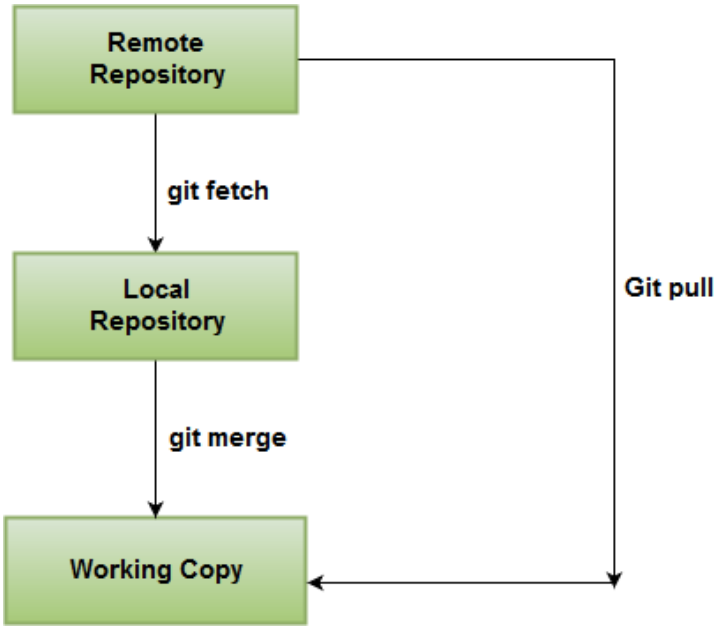
SSH keys are a way to identify trusted computers without involving passwords. You can generate an SSH key and add the public key to your GitHub account.
#ls -al ~/.ssh ---> To see if existing SSH keys are present in machine.
Generate SSH Key
 # ssh-keygen -t rsa -b 4096 -C "MySSHKey"
Here-t --> Specifies the type of key to create. The possible values are "rsa1" for protocol version 1 and "rsa" or "dsa" for protocol version 2.
-b --> Specifies the number of bits in the key to create. For RSA (Rivest, Shamir, and Adelman) keys, the

minimum size is 768 bits and the default is 2048 bits. Generally, 2048 bits is considered sufficient. DSA (Digital Signature Algorithm)keys must be exactly 1024 bits as specified by FIPS 186-2.
-C --> Provides a new comment.
cat ~/.ssh/id_rsa.pub Add SSH key to GitHub

What is the difference between git fetch and get pull?

Ans) git fetch : It will get the update from git remote repo and will update your local repo. But it will not merge with Local working copy.
git pull : It will get the update from git remote repo and will update your local repo as well it will merge with Local working copy also.

So **git pull = git fetch + git merge origin/master**



```
$ git fetch aws master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pac
k-reused 0
Unpacking objects: 100% (3/3), 691 bytes | 30.00 K
iB/s, done.
From https://github.com/landmark-ec/amazon-backend
 * branch            master     -> FETCH_HEAD
   1e8316b..acb037e  master     -> aws/master

user@mylandmark MINGW64 ~/Desktop/aws (master)
$ cat userss.java
This file contains users status

user@mylandmark MINGW64 ~/Desktop/aws (master)
$ git pull aws master
From https://github.com/landmark-ec/amazon-backend
 * branch            master     -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 userss.java | 1 +
 1 file changed, 1 insertion(+)

user@mylandmark MINGW64 ~/Desktop/aws (master)
$ cat userss.java
This file contains users status
Yahweh God of Miracle
```

Personal Access Token (PAT)

Go to **Personal settings ---> Developer settings**
---> Personal access tokens ---> Generate New Token ---> Provide some info about token in Token description input box ---> Select the appropriate the Scopes and Click on **Generate token** button.

#git commit --amend -m "an updated commit message" : Change most recent Git commit message.

 # git grep "Test()" : Search the working directory for Test()

```
bhaskars-air:gitpractice bhaskarreddyl$ git grep "Test()"
DbConnect.java:  public Test(){}
bhaskars-air:gitpractice bhaskarreddyl$
```

#git checkout <<Branch name>> : This will switch the branch. Ex: git

checkout development

#git checkout -b <<Branch name>> : It will create the branch from currently using branch and will switch to new branch.

#git checkout -b feature master: It will create a branch called feature from master branch and will switch to new branch (featur).

#git checkout -b release master: It will create a branch called release from master branch and will switch to new branch.

How to Rename a git branch name?
Ans) git branch -m <oldname> <newname> Or, if you are

already in the branch:

git branch -m <newname>

```
$ git branch
  development
* master
  stage

user@mylandmark MINGW64 ~/Desktop/aws (master)
$ git branch hotfix

user@mylandmark MINGW64 ~/Desktop/aws (master)
$ git checkout hotfix
Switched to branch 'hotfix'

user@mylandmark MINGW64 ~/Desktop/aws (hotfix)
$ git branch -m testing testing
error: refname refs/heads/testing not found
fatal: Branch rename failed

user@mylandmark MINGW64 ~/Desktop/aws (hotfix)
$ git branch -m testing

user@mylandmark MINGW64 ~/Desktop/aws (testing)
$ git branch
  development
  master
  stage
* testing

user@mylandmark MINGW64 ~/Desktop/aws (testing)
```

git branch : It will display all the local repo branches.

git branch -a : It will display all the remote and local repo branches.

git branch -r : It will display all the remote repo branches.

git config http.sslVerify false : To disable SSL verification for that singular repository

git config --global http.sslVerify false : To disable the SSL verification for Globally (For all repositories) --> Not suggested way

git clone <<Git URL>> : To get the code from repository into your local machine.

git log : It will display the commit history. s
git log -p -2 : which shows the difference introduced in each commit. You can also use -2, which limits the output to only the last two entries.

git log <FileName>: It will display the commits related to the specified file.

git log -2 StringUtilities.java: It will display 2 commit ids related to the StringUtilities.java file.

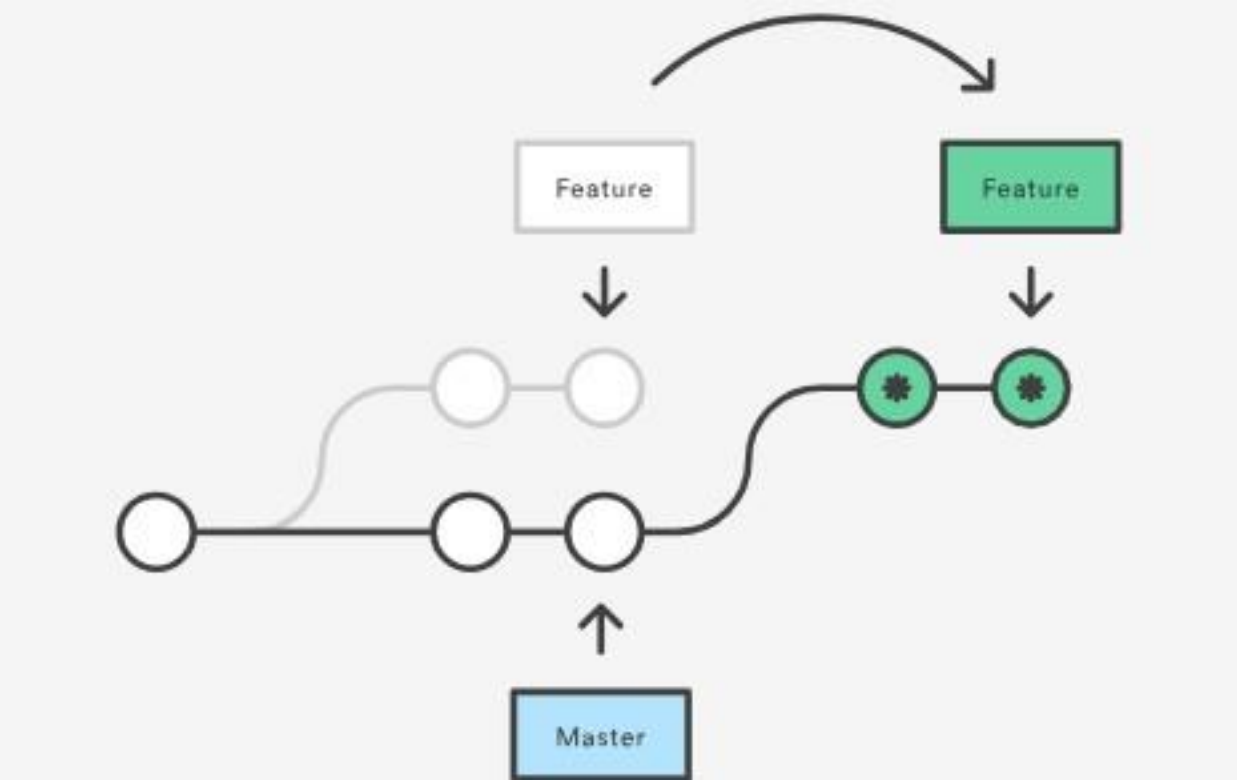
git log --stat: It will give all the files which are altered and number of lines that were added or deleted from each item.

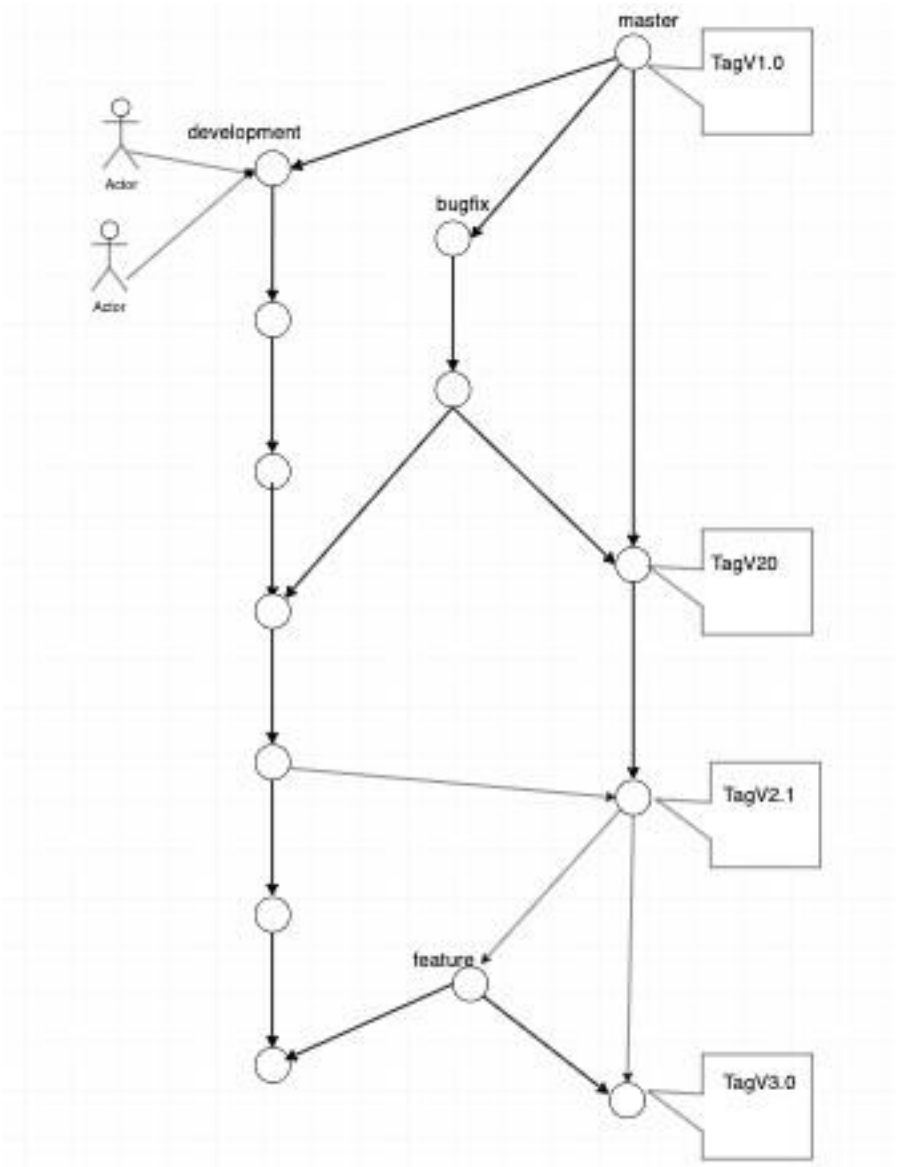
git log --graph --decorate: --graph flag draws a text-based graph of commits on left side of commit message. --decorate adds names of branches or tags of commits shown.

git log --author="mylwp@gmail.com": It will display all the commits which are **committed** by a particular author.

git rm: Removes files from your index and your working directory so they will not be tracked.

What is git rebase?





Git Hooks

Git Hooks are the scripts which trigger when you perform a specific action in Git. These are

useful to automate the tasks.
Ex: When you create a Git Hook to run commit validation every time you commit code.

There are Two types of git hooks.
1) Client-Side Hooks
2) Server-Side Hooks

Client-Side Hooks are trigged by operations such as committing and merging.
Server-Side Hooks are triggered by network operations such a receiving pushed commits.

After you are initialising the git repo, using git init command, it will create a one folder called. git and it contains some sub folders called hooks, branches, info, objects... etc

There are some predefined client-side hooks are available in hooks folder. applypatch-

msg.sample
post-update.sample
pre-push.sample
prepare-commit-msg.sample commit-
msg.sample
pre-applypatch.sample pre-
rebase.sample update.sample
fsmonitor-watchman.sample pre-
commit.sample
pre-receive.sample

To enable any hook from above list, remove the ".sample" from each hook.
Ex: To enable pre-commit.sample hook, rename this file to "pre-commit". **https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks**

README.md

You can add a README file to your repository to tell other people why and how your project is useful, what they can do with your project, and how they can use it.

A README is often the first item a visitor will see when visiting your repository. README files typically include information on:

- What the project does
- Why the project is useful
- How users can get started with the project
- Where users can get help with your project
- Who maintains and contributes to the project

Getting Started
These instructions will get you a copy of the project up and running on your local machine for development and testing purposes. See deployment for notes on how to deploy the project on a live system.

Prerequisites

What things you need to install the software and how to install them Give
examples

Installing

A step by step series of examples that tell you how to get a development env running Say what the steps will be.

Give the examples.
End with an example of getting some data out of the system or using it for a little demo

Running the tests

Explain how to run the automated tests for this system

Deployment

Add additional notes about how to deploy this on a live system

Versioning
Use versioning for each release and keep track of versions.
https://semver.org/

Authors
Add here Authors names.

License

Add the License here.

https://gist.github.com/PurpleBooth/109311bb0361f32d87a2

Git Best Practices

Use branching strategy and pull requests
Commit once you finish the task. Avoid

merge commits.

Don’t Commit Half-Done Work Test your

code before commit.

Write Good Commit Messages before you are committing Try to use

git commands rather than GUI tools.

Resources:
---
https://github.com/
https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup
https://www.atlassian.com/git/tutorials/comparing-workflows/
https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging
http://www.vogella.com/tutorials/Git/article.html
https://help.github.com/articles/duplicating-a-repository/
https://www.atlassian.com/git/tutorials/git-stash https://nathanhoad.net/tags/git
http://rogerdudler.github.io/git-guide http://nvie.com/posts/a-successful-git-branching-model/ https://www.git-tower.com/blog/git-cheat-sheet/