Git & GitHub

24 June 2020 02:09 PM

© Jithin Saji Isaac
Don Bosco Institute of Technology, Mumbai

Windows download:

https://git-scm.com/download/win

- uncheck only show new options while installing

Linux download:

https://git-scm.com/download/linux

```
sudo add-apt-repository ppa:git-core/ppa
sudo apt update
sudo apt install git
```

BEST SOURCE TO LEARN GIT codestackr
Learn Git in 30 Minutes

Git & GitHub Tutorial for Beginners #1 - Why Use Git?

Git and GitHub Beginner Tutorial 1 - Introduction

Git LOCAL

- git --version

- git init
  - .git folder will be made in the directory currently in work
  - -To remove git init, just delete .git folder

To do config

- git config --global user.name 'Jithin Isaac'

- git config --global user.email 'jithinsaji@gmail.com'

To check config

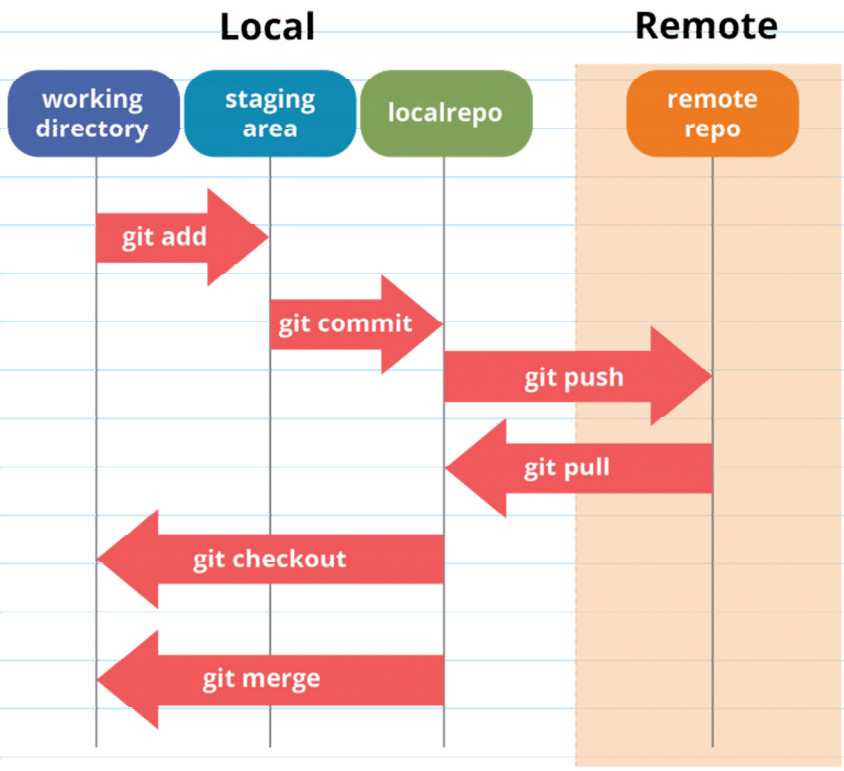
  - git config user.name
  - git config user.email

To check status

- git status

To add files to Staging

- git add filename.html ;
- git add *.html ;
- git add .



To remove files from staging

- git rm --cached filename.html
- git rm --cached *.html
- git rm --cached .

To add all files to staging

- git add . **(to add to staged)**

To commit

- git commit -m 'message_info_here'

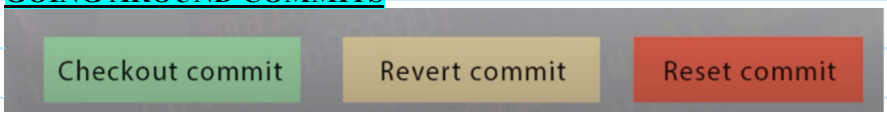
To see current log

- git log
  - **(type q to exit)**
- git log --oneline
  - **(to see all commits in one line)**

To restore files from earlier commit

- git restore filename

GOING AROUND COMMITS



To do simple checkout of other commits; HEAD at this new commit id;
Just to see how code looked like at that commit

- git checkout commit_id **(commit_id of where you want to come->below those you want to discard)**
- git checkout master **(back to master branch)**

To revert a particular commit i.e. remove that particular commit from the chain;
Like it never existed

- git revert commit_id
  - Give some revert message and go ahead; come out of page using :wq
- In git log, one will find a new commit called **Revert 'commit message'**

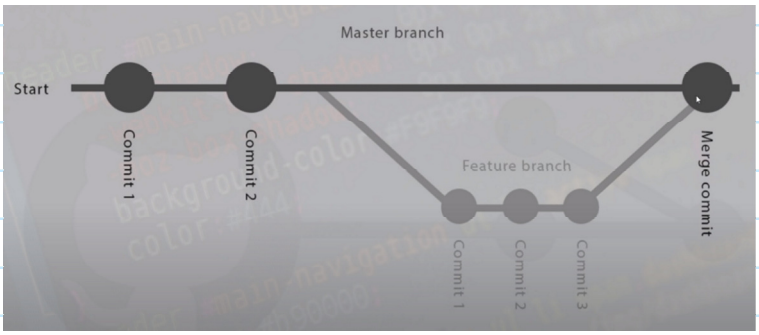
To reset a commit; permanent damage; takes you back in time to that commit;
Permanently deletes all ignored commits

- git reset commit_id
- git reset commit_id --hard
  - **Deletes all not requried commits!!**
  - **Also, changes master location to commit id**

To keep files/folders out of git

- touch .gitignore
- Inside this file, add the filenames which you want to ignore. Then git will not track those files

Branching

- 

- **To add new branch:**
  - **git checkout -b** ***example-branch-name***
    - *OR: git branch example-branch-name AND git checkout example-branch-name*
  - *Work over here, changes not reflected in master*

- **To show all the branches**
  - git branch -a

- **To switch back to master,**
  - git checkout master

- **To switch back to branch**
  - git checkout *example-branch-name*

- **How to delete branch (Delete from LOCAL)**
  - git branch -d *example-branch-name* (If branch merged)
  - git branch -D *example-branch-name* (If branch not merged)

Merge branch to master

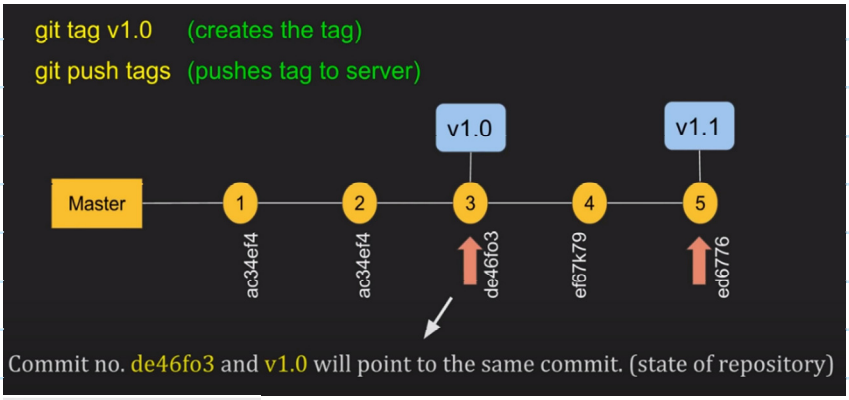
First	git checkout master
Then	git merge *example-branch-name*

- If no conflict, then merge happens automatically.

If conflict, while merging

- **This is basically when master is changed:**
  - **after branch has been created &**
  - **before it has been merged**
- If conflict, then do merging manually
- Delete comments given by git
- and then merge
  - Inside master
    - git add **.**
    - git commit *(No message, coz we are doing a merge)*
    - *In window, just do :wq*
- Check if merged:
  - git log --oneline
  - You should get a *merge branch commit* on top

Git Tags FOR GITHUB RELEASES

- 

- 1 release

- To mark release points for your project
- **git checkout master**
git tag ***tagname***	E.g. git tag v1.1
git tag -a ***tagname*** **-m** ***"tag details"***	E.g. git tag -a v1.1 -m "version 1.1 of project"

- **git tag** -> To view tag
For pushing tags to Github	git push origin v1.1	*It will then show in Releases in Github*
To push all tags to Github	git push --tags
To delete tags from local	git tag -d v1.1
To delete tags from Github	*git push origin --delete v1.1*
To checkout branch from tag	*git checkout -b branchname tagname*	***E.g.*** *git checkout -b branchname v1.0*
To create tag from some past commit	*git tag tagname commid_id*

Starting Steps

- On Github, start repository
- Copy the https link

GIT CLONE

- git clone https://github.com/jithinsisaac/Health-SpO2-Tracker.git
  - git clone creates a folder automatically for us
  - For a cloned directory,
    - Directly **git push origin master** works without stating origin location

To GitHub PUSH

- **Setup where to PUSH**
- git remote add origin https://github.com/jithinsisaac/Health-SpO2-Tracker.git
  - git remote
  - origin is an **alias**
  - it can be anything

- **Steps to PUSH**
- git push origin master
  - **OR** git push https://github.com/jithinsisaac/Health-SpO2-Tracker.git master
  - **OR** git push -u origin master
  - **OR** git push

How to find location where 'origin' is pointing to

- git remote -v
- git remote show
- git remote show origin

From GitHub PULL:
To keep files in local repo upto date with remote repo

- git pull origin master

When working in teams, after cloning and pulling,
make a new branch, work here and push THIS branch

- git checkout *example-branch-name*
- ***Work here, then;***
- git push origin *example-branch-name*
  - *Don't push your master branch*

How to delete branch (Delete from GITHUB)

- git push origin --delete *example-branch-name*

GIT DIFFERENCE TO FIND OUT DIFFERENCE

- git diff

HOW TO KEEP YOUR FORK UPDATED WITH THE ORIGINAL REPOSITORY master ALWAYS

- **Why this question?**
  - We fork a repository
  - Then we branch out, and we commit and push it to your forked repository
  - Then we raise a PR, and it gets accepted/rejected by original maintainer
- **Then what?**
  - What if after all this, the original master is updated?
  - how do you keep your fork upto date with master?

1. git remote add upstream https://github.com/jithin-isaac/upstreamtest.git *i.e. the original repository master*
2. `git remote -v` shows the paths for all remote links
3. git fetch upstream
4. git checkout master
5. git merge upstream/master
6. git add ., git commit -m 'message', git push origin master

- HELP: https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/configuring-a-remote-for-a-fork
- HELP: https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/syncing-a-fork

From GITHUB PAGE

create a new repository on the command line

- echo "# Health-SpO2-Tracker" >> README.md
  - touch README.md
- git init
- git add README.md
- git commit -m "first commit"
- git remote add origin https://github.com/jithinsisaac/Health-SpO2-Tracker.git
- git push -u origin master
- OR just git-push

push an existing repository from the command line

- git remote add origin https://github.com/jithinsisaac/Health-SpO2-Tracker.git
- git push -u origin master

HOW TO DO PULL REQUEST FROM GITHUB

- git clone https://github.com/jithinsisaac/mahabhujalDashboard.git
- Repository will be formed in your local system
- Either make a branch: git checkout -b example-branch-name
  - OR work in master
- do some change; for example edit readme file
- git add .
- git commit -m 'message'
- git push origin master/*branch-name*

- *In github, pull request comes. just pull request and wait for it to be accepted.*