Working area , Staging are, Repository

Nina: Let's talk about the three areas in Git where code lives. The first is the working area, sometimes also called the working tree. The second is the staging area, you might also see this called the cache or the index and the last is the repository.

>> Nina: The working area, in brief, is the files that are not in the staging area.

[00:00:29] They're not handled by Git. They're kind of just in your local directory. They can also be called untracked files. The working area is like your scratch space, it's where you can add new content, you can modify delete content, if the content that you modify or delete is in your repository you don't have to worry about losing your work.

[00:00:53] The staging area, that's what files are going to be a part of your next commit. It's how Git knows what is going to change between the current commit and the next one. We're gonna deep-dive into the staging area in the next section. Briefly, the repository, that's the files that Git actually knows about.

[00:01:18] The repository contains all of your commits. And the commit is a snapshot of what your working and staging area look like at the time of the commit. It's in your .git directory. And that's all of the files in your repository, they're stored away safely, you can continue to make changes as you work and you can always check out a fresh copy if you need one.

GIT Commands:

Install GIT :

Install Textmate software.

**COMMITTING SINGLE FILE INTO GITHUB**

To create git project : git init <project-name>

create file through Terminal: mate <filename.txt>

git status - is to check uncommited files

git add <filename.txt> -- to commit files

git commit ->a file would be openeded in textmate . provide commit comments & then save.File is being saved in git repo.

**COMMITTING PROJECT INTO GITHUB**

git add .

git commit -m “commit msg”

To clone Repo : git clone <http url of repo>

before commit run to update git files in local: git pull origin master

and then git push origin master

while pushing the code if you face authentication issues follow below steps:

1. generate public key for your profile by executing below command in terminal

ssh-keygen

1. this will generate public & private keys in your home folder as .ssh folder.

to view hidden files in Mac command + shift + .

1. copy public key & paste it to github profiles.

github profile -> settings -> SSH & GPG keys ->New SSH key->paste the key and provide key name then save.

1. set ssh url.

git remote add origin [git@github.com:ParvathiChinthalapati/JavaCodingQuestions.git](mailto:git@github.com:ParvathiChinthalapati/JavaCodingQuestions.git)

git remote set-url origin [git@github.com:ParvathiChinthalapati/spring-basics.git](mailto:git@github.com:ParvathiChinthalapati/spring-basics.git)

git remote set-url origin [git@github.com:ParvathiChinthalapati/JavaCodingQuestions.git](mailto:git@github.com:ParvathiChinthalapati/JavaCodingQuestions.git)

git init

git remote add origin [git@github.com:ParvathiChinthalapati/restful\_webservices.git](mailto:git@github.com:ParvathiChinthalapati/restful_webservices.git)

git remote set-url origin [git@github.com:ParvathiChinthalapati/restful\_webservices.git](mailto:git@github.com:ParvathiChinthalapati/restful_webservices.git)

git push origin master

git remote add origin <https://github.com/ParvathiChinthalapati/spring_data_jpa.git>

git remote set-url origin <https://github.com/ParvathiChinthalapati/spring_data_jpa.git>

git remote add origin <https://github.com/ParvathiChinthalapati/microservices.git>

git remote set-url origin https://github.com/ParvathiChinthalapati/microservices.git

1. Push the code

git push origin master

If we want to combine both adding files and commiting files use below command

git commit -am "<commit comment>"

command to view git tracked files

git ls-files

**Recursive Add (adding files from nested folders)**

git add .

git commit

mate will open ->provide commit comment->save

**Backout the changes from staging area:**

once changes commited if we want to roll back our changes

git reset HEAD^

git restore <filename>

**rename files:**

git mv <oldfilename> <newfilename>

**moving file to another directory:**

git mv <filename> <foldername>

**Delete file which is untracked by git:**

rm <filename>

**Delete file which is tracking by git:**

git rm <filename>

**Delete folder from git :**

rm -rf <foldername>

**To commit this change:**

git add -A “commit message”

then deleted folder commited into git.

**To view git history:**

git log

**If you want to view commit history in single line format:**

git log --oneline --graph --decorate

**If you want to view commit 3 days ago:**

git log --since=”3 days ago”

**To view specific commit id files:**

git show <commitid>

**Git Alias:**

if you want to create a alias command for another git command

git config --global alias.hist “log --all --garph --decorate --oneline”

**Configure P4merge for git:**

parvathichinthalapati@Parvathis-M2-MacBook-Pro MacOS % git config --global diff.tool p4merge

parvathichinthalapati@Parvathis-M2-MacBook-Pro MacOS % git config --global difftool.p4merge.path /Applications/p4merge.app/Contents/MacOS/p4merge

parvathichinthalapati@Parvathis-M2-MacBook-Pro MacOS % git config --global difftool.prompt false

parvathichinthalapati@Parvathis-M2-MacBook-Pro MacOS % git config --global merge.tool p4merge

parvathichinthalapati@Parvathis-M2-MacBook-Pro MacOS % git config --global mergetool.p4merge.path /Applications/p4merge.app/Contents/MacOS/p4merge

parvathichinthalapati@Parvathis-M2-MacBook-Pro MacOS % git config --global merge.prompt false

**Comparisions:**

git diff -- to view changes b/w working directory & staging are

git difftool -- to view changes in tool

git diff Head -- to view changes b/w working directory & last commit(git repository)

git difftool Head

git diff --staged HEAD -- to view changes b/w staging & git repository

git diff -- readMe.txt -- to view changes for one file

**Branching & Merging:**

git checkout -b <branchName> -- create branch and checkout that branch

git branch <branchName> -- create branch but not checkout

git branch -a -- list all git branchs

git log --oneline --decorate -- to check git commit history

git branch -m <oldBranch> <newBranch> -- for rename branch

git branch -d <branchName> -- for delete branch

git diff <firstRefBranch> <secRefBranch> -- to compare changes in branches

ex: git diff master test\_branch

git difftool <branch1> <branch2> -- to visualize differences

git merge <branchName> -- to merge changes of branch into current branch

git log --oneline --graph --decorate --all

git merge <branchName> --no-ff ----> to disable fast forward merges

git merge <branchName> -m “<commit msg>” -- to merge Branch

git mergetool -- to visualize merge conflicts

git rebase master -- adds current branch changes on top of master branch

git fetch origin master -- updates master branch references

**Git Stashing:**

git stash -- removes un comitted changes of git tracked files

git stash apply -- displays changes for stashing for both git tracking & untracked files

git stash list --list the stash entries that you currently have in

git stash drop --drops the top stash , deletes both git tracked & untracked changes

git stash -u ---removes untracked files as well

git stash pop -- this is a combination of git stash apply + git stash drop. It will display the untracked & tracked changes and drops those.

git stash save “msg” -- save the stash

git stash show stash@{1} -- displays specific entry of the stash

git stash apply stash@{1} -- display stash for first entry

git stash drop stash@{1} -- drops stash at first entry

git stash clear -- clears all the stashes

git stash branch <branchName> -- new branch would be created.added untracked,staging changes & tracked changes into the branch & then dropped the branch.

**Git Tagging:**

git tag <tagName> -- creates tag

git tag --list --displays git tags

git show <tagName> -- displays changes under that tag

git tag --delete <tagName> --deletes the tag

Annotated Tags

git tag -a v-1.0 -- allows to add tag commit msg

git show <tagName> -- git show for annotated tag displays commit msg along with the author & date modified.

git diff v1.0 v2.0 -- compares diff b/w v1.0 & v2.0 tags

git difftool v1.0 v2.0 -- visualizes the diff b/w v1.0 & v2.0 tags

Add tag to already commited msgs

git tag -a v0.9-beta 96ef75b -- adds tag to the commited git code of 96ef75b

Update the tag -- change the tag of one commit to another commit

git tag -a v0.9-beta -f bd35d46 -- adds the tag to this git commit code

pushing git tags into github

git push origin v1.0 --pushes the tag

git push origin master --tags ----- pushes all the tags into github

git push origin :v1.0 --deletes v1.0 tag from github