GIT

Q. What is Git?

Ans: Git is a distributed version control system (DVCS). That helps to manage source code changes & manage the version of the code.

There are two types of version control system:

1. Centralised Version Control System (CVCS)
2. Distributed Version Control System (DVCS)

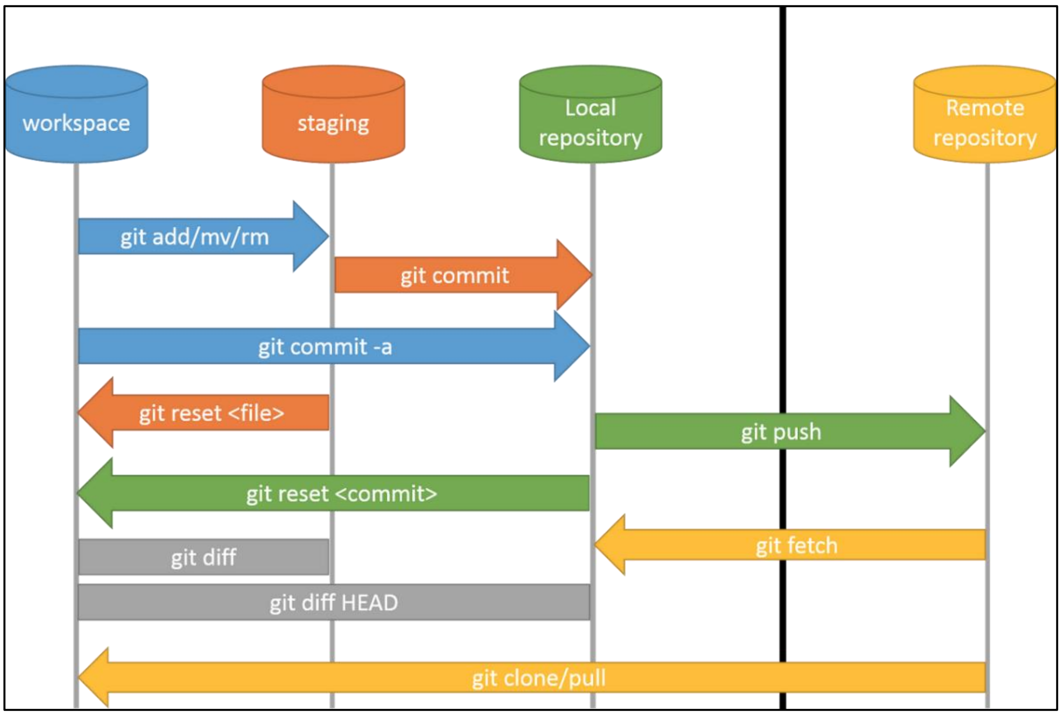
* In CVCS, a client needs to get local copy of source from server, then do the changes and commit these changes to central source of server.

* In DVCS, each client can have a local repo as well, and have a complete history on it. Client needs to push the changes to branch which will then be pushed to server repository.

There are three logical stages of Git:

1. Untracked Files: Working Directory
2. Staging Area
3. Local repo

- **Repository:** Is the place where we keep all our codes and files. It can be local or remote.
- **Working Directory:** Where we work or write the codes and keep the files initially.
- **Commit-ID/Version-ID/Version:** Reference to identity of each change. We can also identify who changed the files.
- **Commit:** To move the files or code from stagging area to local repo.
- **Branch:** In Git, a branch is a new/separate version of the main repository. Branches allow you to work on different part of a project without impacting the main branch. When the work is



complete, a branch can be merged with the main project. Even we can switch between branches and work on different project without them interfering with each other.

- **Push:** Push operations copies files/code/changes from local repo repository to a remote or central repo.
- **Pull:** Pull operation copies the code/files/changes from central repo to local machine.

- yum install -y git // To install git.
- git --version // To check version.

- 1st time we need to configure username and email.
- git config --global user.name "username"
- git config --global user.email "email id"
- git config --global --list // To check username and email.

- To make current directory as git working directory run below command and a ".git" file will auto created after run the command with some configuration.
- git init

- To check untracked and stagging files
- git status
- git status --short

- To move the file from working directory or untracked file to stagging area. We can add one file by specifying one file name or all files by using "." .
- git add <file name>
- git add .

- To move files from stagging area to untracked area.
- git rm --cached <file name>

- To move all files from stagging area to local repo.
- git commit -m "Any relevant description"
- To commit directly.
- git commit -a -m "Any relevant description"

- To see the commit status
- git log
- git log --oneline

Note: Head Commit = Latest Commit

- To show the details of the commit.
- git show <commit id>

Branches:

- To make new branch.
- git branch <branch name>
- To list all available branches.
- git branch
- To switch another branch.
- git checkout <branch name>
- Make new branch and switch directly.
- git checkout -b <branch name>
- Delete branch "-d = gracefully" & "-D = forcefully"
- git branch -d or -D <branch name>
- To merge the specific branch with master branch.
- git merge <branch name>

Points:

- Branch also helpful for parallel development and changes are personal to that particular branch.
- Git use pulling mechanism to merge branches. Only new code will copy from branch.

How to ignore some files while committing.

- 1st we need to create .gitignore file under working directory and enter file format or any specific file name which we want to ignore.
- vim .gitignore

```
  *.css
   *.java
:wq!
```

Then commit the .gitignore file.

- git add .gitignore
- git commit -m "committing .gitignore file"

Git Conflict:

- When files with same name having diff-diff content in diff-diff branches and we are doing merge then conflict will occur.
  Git not able to identify or understand which line need to wite 1st from one of the files.
  So, after merge we have to edit the file to resolve the conflict then add and commit.
  e.g.
    - git checkout branch1
    - echo "My name is Abhijeet" > file1
    - git add file1
    - git commit -m "File1 from branch1"

    - git checkout master
    - echo "I'm working in Master Branch" > file1
    - git add file1
    - git commit -m "File1 from master"

- Then merge the branch1 it will show conflict.
    - git merge branch1

So, we have to edit the file first then add & commit again.

Git Stashing:

- When we are working in some particular project code and on that time 2nd project is came and have to do 2nd one 1st by stopping current project on that time, we will move the current code in the stashing area. Once the work is completed again, we will pull the previous code from stash area to current working directory to start the stopped work.

    - git stash // To move the item into stash area.
    - git stash list // To see the stashed item list
    - git stash apply stash@{0} // To pull files from stash to WD to work again. "0" means latest.
    - git stash clear // To clear stash.

Git Reset:

- Git reset is used to remove the file from stagging area and restore in untracked file.

    - git reset filename
    - git reset --hard # To remove from stagging and untracked area.

GIT

Git Revert:

- Git revert command helps to undo the existing commit. It will revert into previous state by did one more commit. So, we can use git revert if we wrongly commit any files.

  - ➢ git revert commit-id

- How to remove untracked files.
- git clean -n   # dry run, it will prompt which files will be removed then need to run -f to delete the files.
- git clean -f # Now it will remove the files.

- How to add Tag.
- Tag Is the alias name of commit.
  - ➢ git tag -a tag-name -m “comment” commit-id   # To add tag on the commit.
  - ➢ git tag   # To list tags.
  - ➢ git tag -d tag-name # To delete tags.

GITHUB

- GitHub is the remote repository to store git files or code.
  # push the code from local repo to central repo (github).

- git remote add origin <url of git repository>   # To add the remote repo with git.

Note: When we push the code to git hub it will ask for github id & password. For password we have to generate token that we can use as pass.

setting--developer setting-- personal access token--generate new token.

- git push -u origin master   # Push the code to central repo to master branch.
- git pull origin master   # To pull the code from central repo of master branch to local repo.

- git fetch origin   # To fetch github remote repo origin details.