GIT-Material

GIT

Introduction to GIT:

- Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- Other Words GIT is a central repository by using which we can manage our project code.
- GIT is called SCM (Source Code Management) tool,that means where we maintain all our project source code.
- GIT is also called Version Controlling System, that means GIT maintains all modifications(or versions) happening to a specific file made by different developers.
- Because of version controlling we can easily troubleshoot errors/bugs easily and also we can can roll back to previous versions from current versions if anything goes wrong.
- GIT is a distributed version controlling system that means multiple developers can easily work on the same project at same time.
- GIT also records who modified when modified and why modified, because of those features we can easily track the project source code.

Some other SCM tools:

- SVN(SubVersion)
- Clear Case
- TFS
- Perforce
- CVS
- CodeCommit(AWS)
- GIT

Setting up central git server:

- On premise --> Get your own server, install GIT, configure it and maintain it.
- On Cloud --> GitHub, Bitbucket (From Atlassian), GitLab, Codecommit (From AWS).

Creating GitHub account:

1. From https://github.com/ → Sign Up → Choose Unique username → Enter your Email → Choose alphanumeric password → Sign Up for GitHub.
2. Once you sign up git will sends to a one confirmation email to you, once you confirm it, git creates account for you.

Installing GIT client: To interact or communicate to central git server from our laptop we want mediator for that so we need git client as mediator. GIT client supports both CLI (Command Line Interface) or GUI (Graphical User Interface). Availabne GIT clients in market are

- GIT Bash
- Tortoise GIT.
- ATOM

GIT-Material

- Any Linux Flavour Etc...

Choose any one from above.

Installing GIT Bash:

- From https://git-scm.com/downloads → Choose Your OS → And download
- Once you downloaded the git executable file you can install git by choosing all default values.
- In order to open access the git bash we want to give right click choose git bash here.

To Check the Version:

git --version

Creating GIT project: There are two ways to create your project.

1. Create locally and initialize it, and upload to Central.
2. Create in central and Clone it.

First Method by using git init Command:

1. Create one empty repository on Desktop → open it → from inside the folder → right click → git bash here.

Ex: mkdir wipro
vim 1.txt

Command:

git init

When you use *git init* command, it converts our local repository to git repository.

- Next add one file to the folder with any content and save.

Creating new repository in git central hub:

Sign In to your github account → New Repository → Give unique name → Select public repository → create repository

Ex: wipro

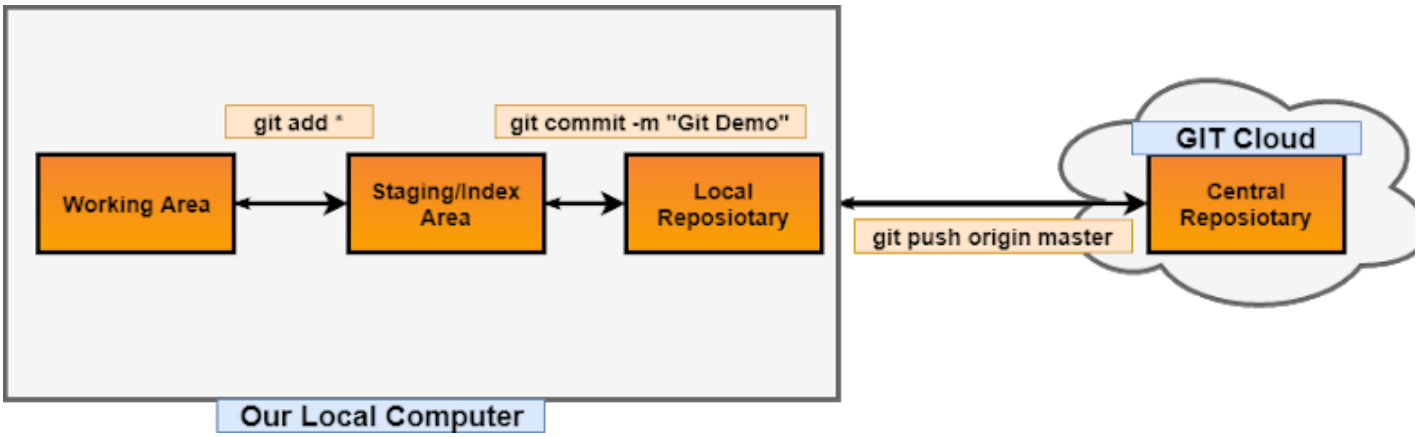
Connecting local repo with central repo:

git remote add origin https://github.com/devops-mptech/wipro.git

Command to check the remote origin details:

git remote -v

GIT Architecture:



GIT-Material

Staging/Index Area: For committing modified files first we need to stage them, this area is to stage the files we want to commit local repository.

Configuring GIT Client with user details:

Command:

git config --global user.name "john"
git config --global user.email "john123@gmail.com"
git config -l → To list the author username and email

This information is used by git to record our commits.

Note: Without configuring the user details we can't commit the files from staging area to local repository.

Commands to add the files to the staging or index area:

1. *git add one.txt*
2. *git add ** → To add all files at a time
3. *git add *.py* → To add all files with extension .py
4. *git add 1.txt 2.txt 3.txt* → To add more file by using space separated.
5. *git add p** → to add all files with file name starts with "p"

Command to Commit the files to the local repository:

git commit -m "Commit message"
Where m is "message"

Command to check the status of the git repository:

git status

This command will give the present status of the git repository.

Command to push the committed files to the central repository:

git push origin master
Where origin → alias name for remote repository url
Master → Default main branch

Command to check commit history of this branch:

git log → To check Commit history of all files
git log 1.txt → To check commit history of particular file

Second Method by using git clone Command:

"Git clone" command will use to download the project from central to local machine First create a new repository in github account → copy the url of the repository → from local machine open git bash on desktop → use git clone command to download from central

EX:

git clone https://github.com/devops-mptech/wipro1.git

Resolving GIT push conflicts:

If git remote repo contains some additional work which is not present in our local repo, when you push local to central git push will fails (or rejected). So to solve this

GIT-Material

git pull origin master

If you run this command you will get all remote changes to the local and merges with local changes by adding a new commit.

GIT fetch:

git fetch origin master

If you run this command you will get all remote changes to the local but it will not merge remote changes with local changes.

After for merging use following command

git merge

Command to discord the changes in working directory:

git checkout -- file name

Command to unstage the file:

git reset HEAD file name

Reverting Changes:

We can undo our commits in two ways. Those are

- Git reset
- Git revert

Git Reset:

git reset HEAD~1

If your changes are not pushed to the remote repo use git reset, when you use the reset it removes commit from commit history.

GIT Revert:

git revert commit id

It does not removes the commit from the history, instead of that it reverts changes to the file and makes new commit.

Note: Don't use reset command if you pushed your changes into central instead you can use "git revert" command

GIT Branching:

- Branch is used to work on a specific task.
- Branch provides isolation, that means separation of work.

Master branch : Every git repository comes with a default branch which is called as master branch. Master must contain only well tested code and no one should be directly work on master. If any work assigned to you, you can create a new branch and work on the branch and finally merge that branch to master branch.

Command to create a new branch:

git branch branch-name

Command to Switching a new branch:

git checkout branch-name

git checkout -b development → To create a new branch and switch to new branch

Merging changes in new branch to main branch:

We can do this in two ways

- By using merge command
- By creating a pull request

GIT-Material

Again switch to master branch → Next merge new branch to master branch.

git branch development → To Create new branch
git checkout development → To switch “development” branch
-----DO ALL CHANGES -----
git checkout master → To switch “Master” branch
git merge development → Finally to merge

Note: In real world all people will not had permissions to merge directly to master, so this approach is not good, Instead of we will create a ”pull request”

By creating a pull request:

Pull request enables team mates to review and comment on the changes before merging to main branch, we also can see how many file are modified, we also can compare modified file with their old version.

Procedure:

First switch to new branch → Do what all changes → stage them → commit them → push this branch to central → Next create pull request in GIT GUI.

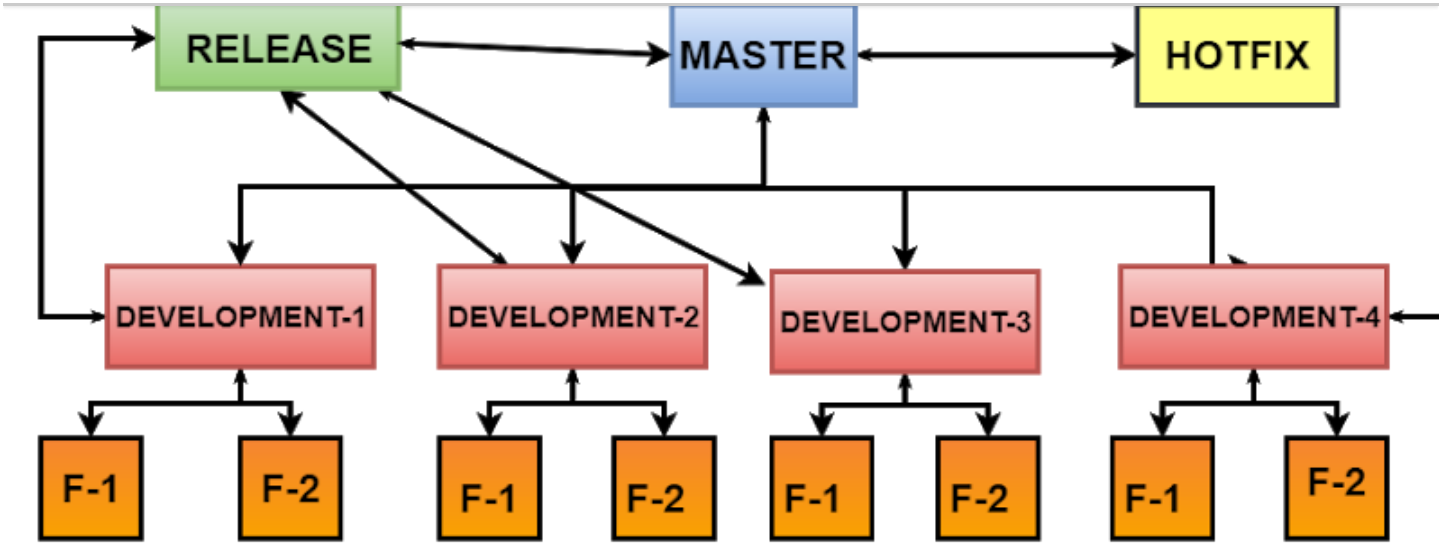
git push origin development → To push development branch to central
From github account create pull request.

Git branching commands:

1. *git branch -d development* → To delete merged branch
2. *git branch -D development* → To delete unmerged branch (force delete)
3. *git branch* → To list all local branches
4. *git branch -r* → To list all branches in remote
5. *git branch -a* → To list all local and remote branches
6. *git branch -m new-name* → Rename your local branch (use from that branch)
7. *git branch -m old-name new-name* → Rename from different branch

Git branching strategies:

GIT-Material



Text

Master Branch: Master is a default branch which contains well tested code, in real world no one will directly work on master.
Development Branch: Development branch is created from Master branch. It is specific to specific team, after completion all teams work there code will merge into “Release Branch”. All team members will integrate their code into “Development Branch”.
Feature Branch: Feature branch is specific to particular team member, it is created from “Development branch”. After completion of all team members works they will integrate their code into “Development branch”.
Release Branch: In this branch all teams changes integrate their code into in this branch. This is created from “Master Branch”.
Hotfix Branch: This branch is for to fix production defects. This is created from “Master Branch”.

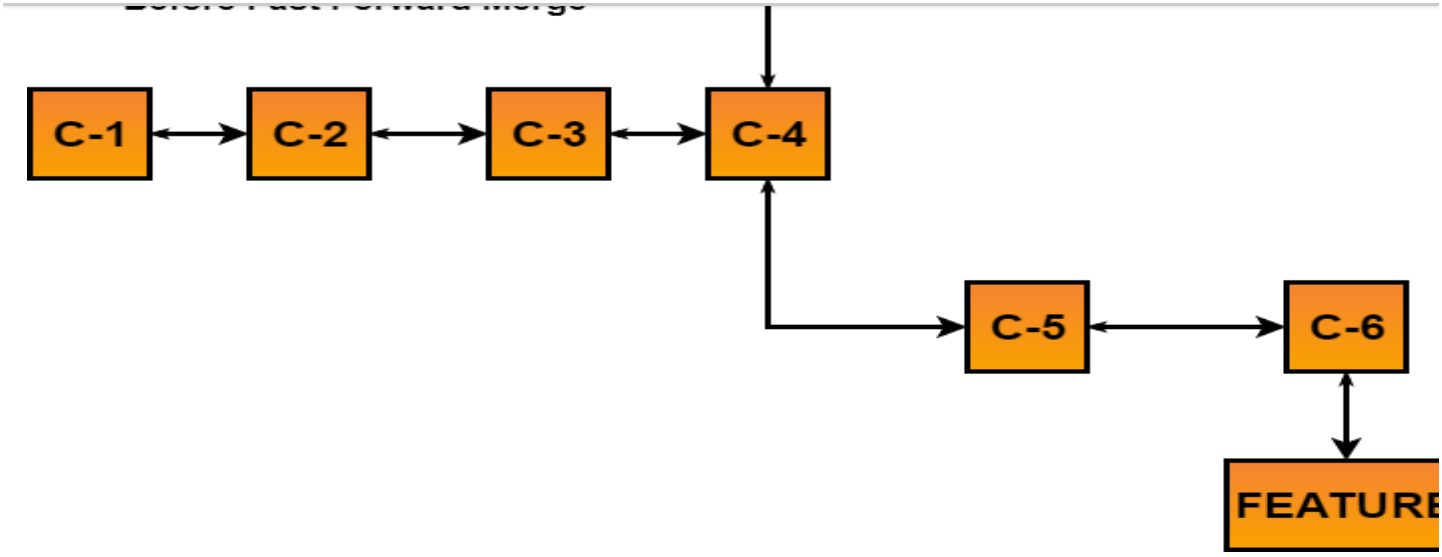
Git Merging Strategies:

There are two types of merging is there, those are

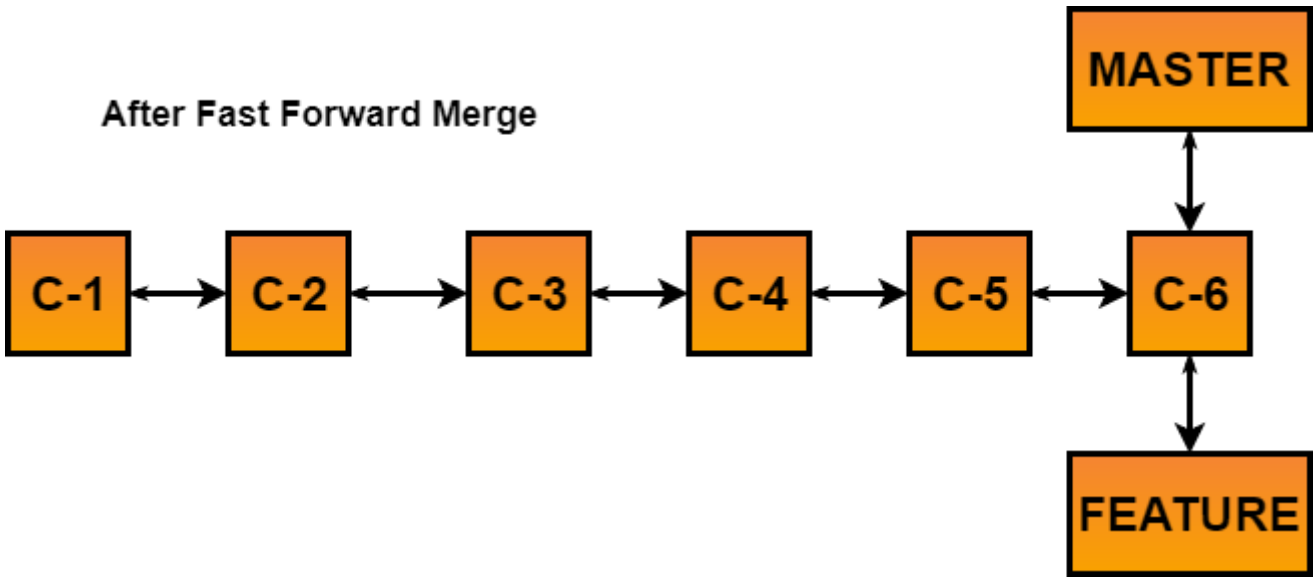
1. Fast Forward Merge.
2. Recursive/Three Way Merge

Fast Forward Merge:

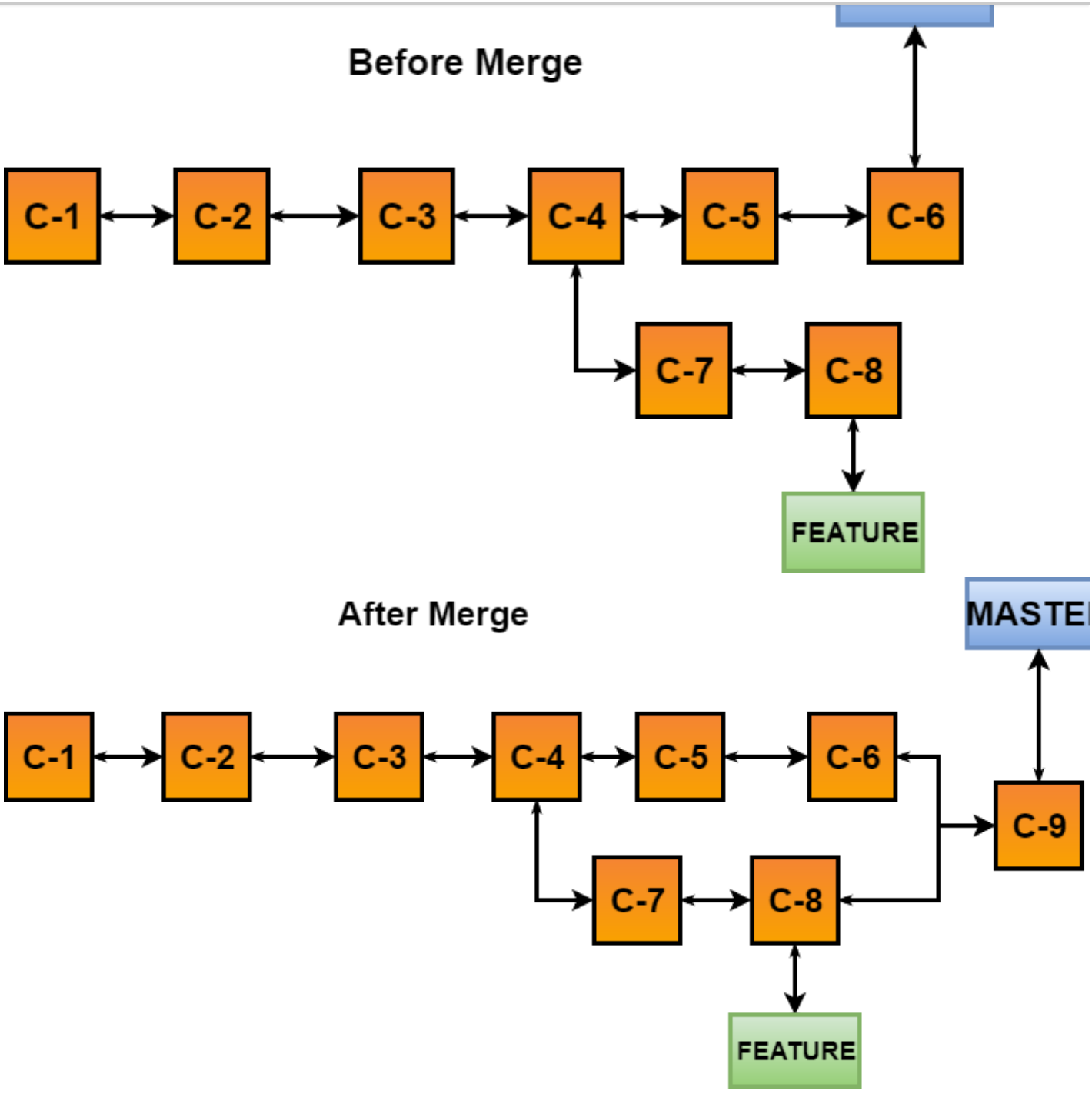
GIT-Material



After Fast Forward Merge



GIT-Material



GIT Stash:

Git Stash will saves the working area changes in temporary memory and makes working area as clean. After you can do some other work, after completion of work you can re apply your previous changes.

GIT Stash Commands:

1. *git stash save* → To save all working area changes temporarily and it keeps working area as clean.
2. *git stash pop* → Changes moves to working area, and it removes entry from list.
3. *git stash apply* → Changes moves to working area but still entry will be present in list.

GIT-Material

By using “git cherry-pick” you can copy the commit from different branch and apply to the current branch.

Command:

git cherry-pick commit id

Git tag:

From particular commit you can create a tag to release the software, tag is similar to branch. In branch you can do further commits but in tag you can’t do further commits.

Git tag Commands:

1. *git tag V-0.0.0* → To create a tag
2. *git push origin V-0.0.0* → To push tag into central.
3. *git tag -l* → To list all tags.
4. *git tag -d V-0.0.0* → To delete the tag

.gitignore file: If you want to restrict particular files to push from local to central, create “.gitignore” file and add files in “.gitignore” file which you don’t want to add.

Ex:

vim .gitignore → add files which you don’t want to push

Git fork: By using git fork you can pick the git repository from different account and saves to the your github account.

Mlscellaneous Commands:

1. *git checkout commit-id* → to check the data in particular commit
2. *git checkout commit-id 1.sh* → To check the data of particular file at that commit.
3. *git branch --merged* → To see merged branches.
4. *git branch --no-merged* → To see unmerged branches.
5. *git commit --amend* → To change the last commit message.
6. *git remote -v → To Check the remote origin details*
7. *git remote remove origin → To remove the origin details.*
8. *git remote add origin https://github.com/xyz/demo.git*