What is Git?

- Git is a central repository using which we can manage our project source code
- Git is also called it a version controlling system
- It maintains all modifications happening to a specific file
    - Because of versions troubleshooting and fixing bugs is easy
    - If something goes wrong in current version we can rollback to previous version
- Records who modified, when it is modified(timestamp) and why it is modified
- Git is distributed version controlling system
- Git is fast when it is compared with other version controlling tools
- Multiple developers can easily collaborate and work on same project
- It also works as backing up our project code

Getting git server to manage our projects

- Down,Install and configure git server on our own machine
- Create account online with
    - GitHub
    - Bitbucket
    - Gitlab
    - CodeCommit(AWS)

Create account in github.com and signin

Creating new repository in github.com

What is repository in git?

- In git repository represents a project.

Steps to create repository

- New Repository → Enter repository name → select public → select README file → Create Repository.

Git client

For interaction between our local machine and remote git server we need git client.
Git supports both GUI (Graphical User Interface) & CLI (Command Line Interface)
Available git clients

- Git bash
- Source tree
- Tortoisegit
- Git extension
- Smartgit
- Atom
- Etc…

Install Git bash

Download git bash
Double click the exe file and install with all default selections
In Order to open git bash right click → open git bash here

Getting code from remote git server

From your git bash

```
git clone https://github.com/javahometech/devops-123.git
cd devops-123
```

Git clone clones the remote copy into our local machine
Note: The local copy is called as local repository

Configure git client with email and username

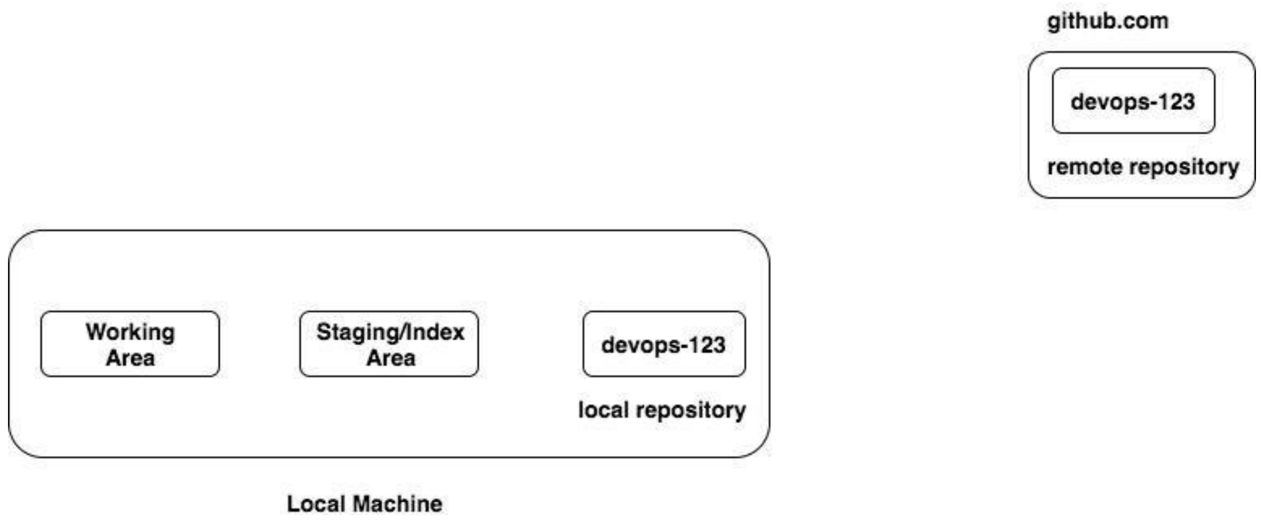
This information is used by git to record our commits

```
git config --global user.name "Hari"
git config --global user.email "hari@javahome.in"
```

Note: --global tells git to use the same information for all the repositories i manage on my local machine.

Modifying files and pushing those changes to local and remote repositories

- Open README.md file
- Add some content into this file and save it
- Staging a file
    - git add README.md



Working Area:

Any modifications we do to the local repository those modification are kept under working area.

Staging/Index Area:

This area is to stage the files we wanna commit to local & remote repository

git add README.md
*git add **
*git add *.java*
*git add *.sh*

git status
This provides information about our local repository (working area, stating area & local repository)

Commit: When we do commit, it picks the files present in staging and commits to local repository.
git commit -m 'Learning git tool'

Note: Before we perform commit git name and email must be configured

Checking commit history of this branch

git log

Checking commit history of a file

git log README.md

Pushing local commits to remote repository

git push origin master
origin → is the alias name for remote repository URL
master → is a branch to push our changes into

Git get specific version of a file

To move back to older versions of a file
git checkout <commit-hash-id><file-name>

Resolving git push conflicts

If remote contains a work which is not present in the local then git push is rejected. To solve this problem we need to pull the remote changes and merge with local changes and push it back.

Resolving git push conflicts using pull

git pull origin master
git pull pulls remote changes to local and merges with local changes by adding a new commit

Git Fetch

Gets remote changes to the local without merging.
git fetch origin master
For merging

git merge

Git Branch

Branch is used to work on a specific task (enhancement, bug, new feature)
Branch provides isolation, i.e. other work will not impact my work
Master branch: every git repository comes with a default branch which is called as master branch.

- No one should directly work on master
- Master must contain only well tested code
- In real world most of the guys will not have permissions to push to master

If there is a new task to work on, then we should create a branch and work on it.

Creating a git branch

git branch <branch-name>
This creates a new branch from current branch

Switching a branch

git checkout <branch-name>

Merging changes in our branch to main branch

We can do this in couple of ways

- By using merge command
  We want changes in 'master' so first checkout master and run merge command
  git checkout master
  git merge <branch-name>
- Create a pull request

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.

Creating pull request

Push local branch to remote
git push origin <branch-name>

From github.com create a pull request

Listing git branches

git branch
  Displays all local branches
git branch -r
  Displays all local branches in remote
git branch -a
  Displays all local plus remote branches

Deleting git branches

After merging changes to main branch we can go ahead and delete the branch used for our implementation.
git branch -d <branch-name>

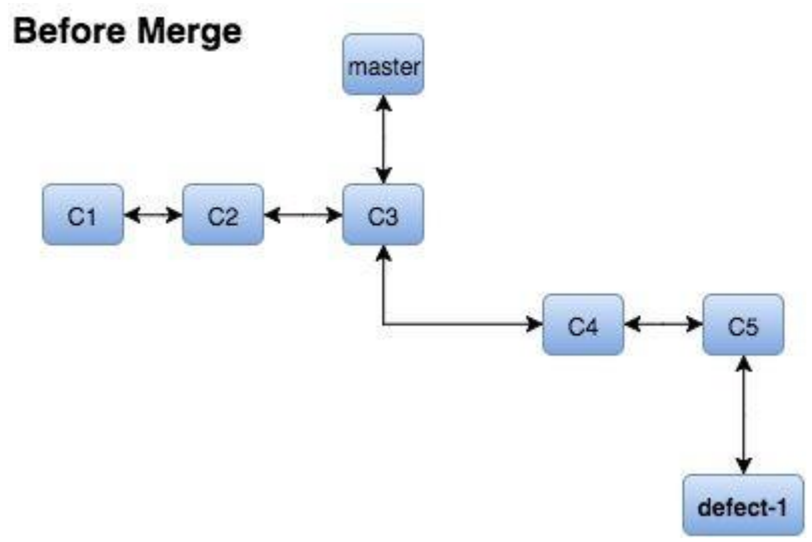
Git force delete

git branch -D <branch-name>
Note: if branch is not fully merged we git does not allow us to delete a branch, we can use force delete in such use cases.

Git merging strategies

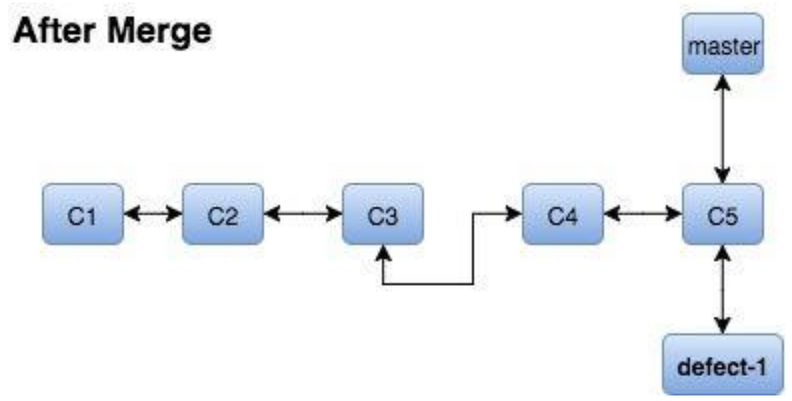
- Fast Forward merge
- Recursive/ Three way merge
- Rebase merge

Fast forward merge



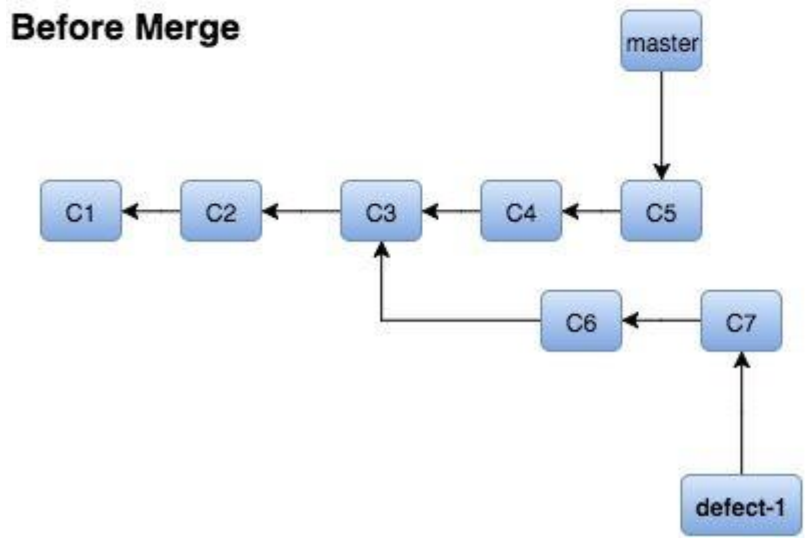
We created defect-1 from master@C3

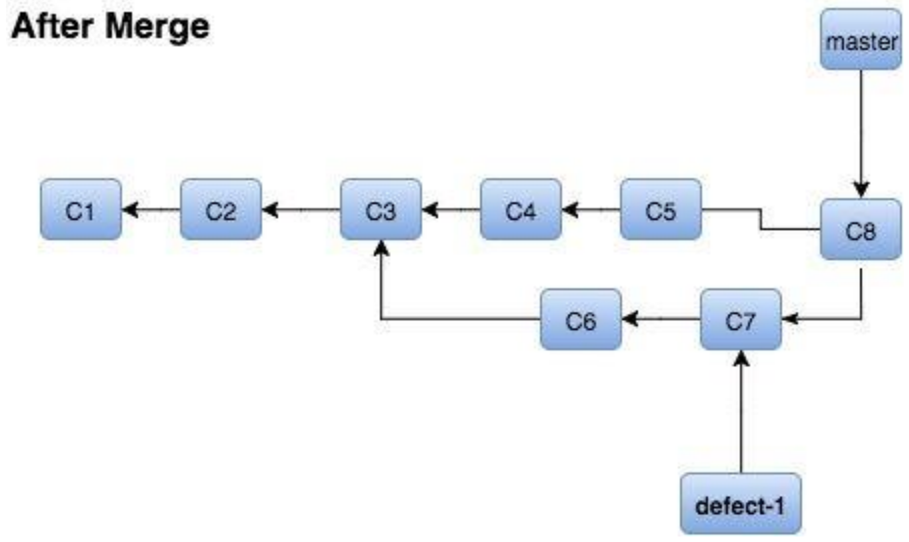
After defect-1 is created there are no commits on the master



Fast forward merge move the pointer of master from C3 to C5

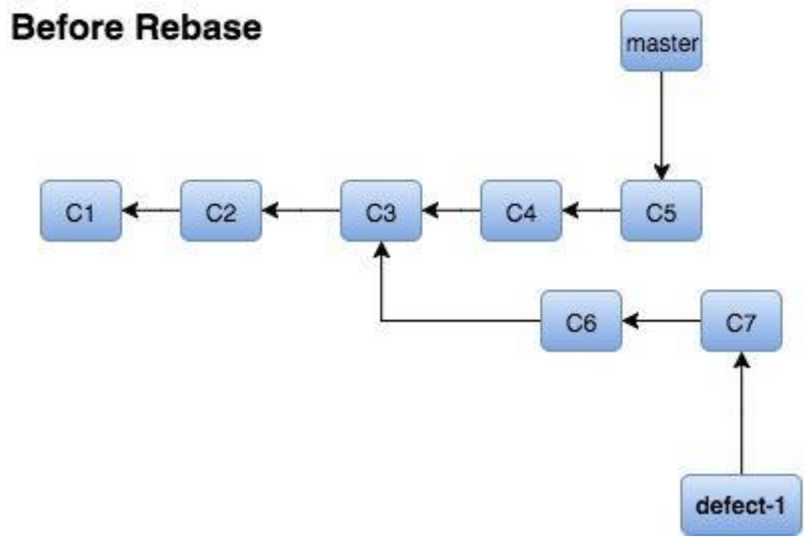
Recursive/ Three way merge

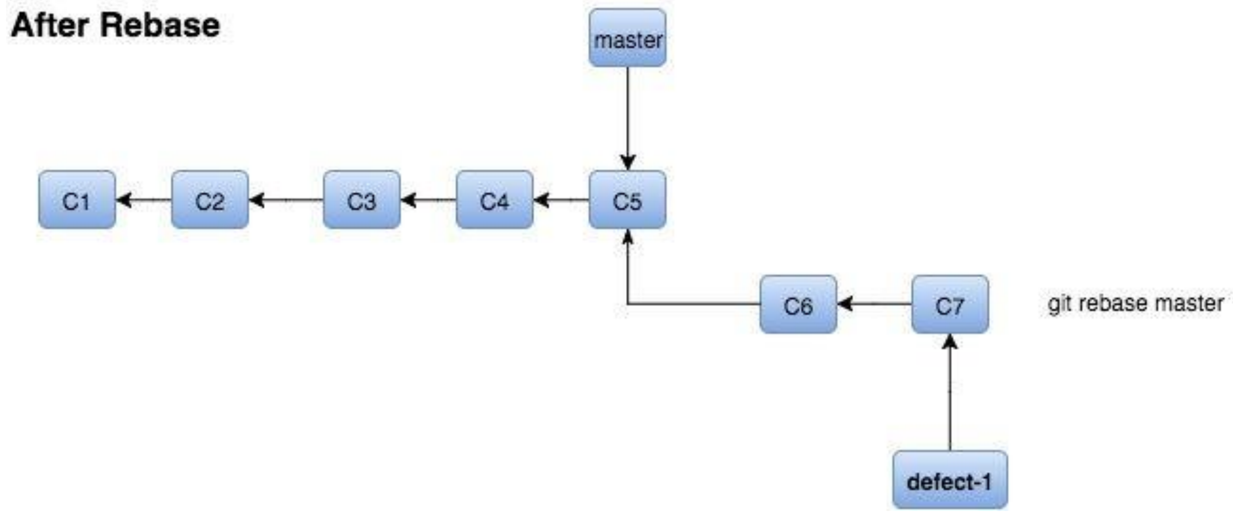




Git created a new commit(C8) for recursive merge.

Rebase Merge





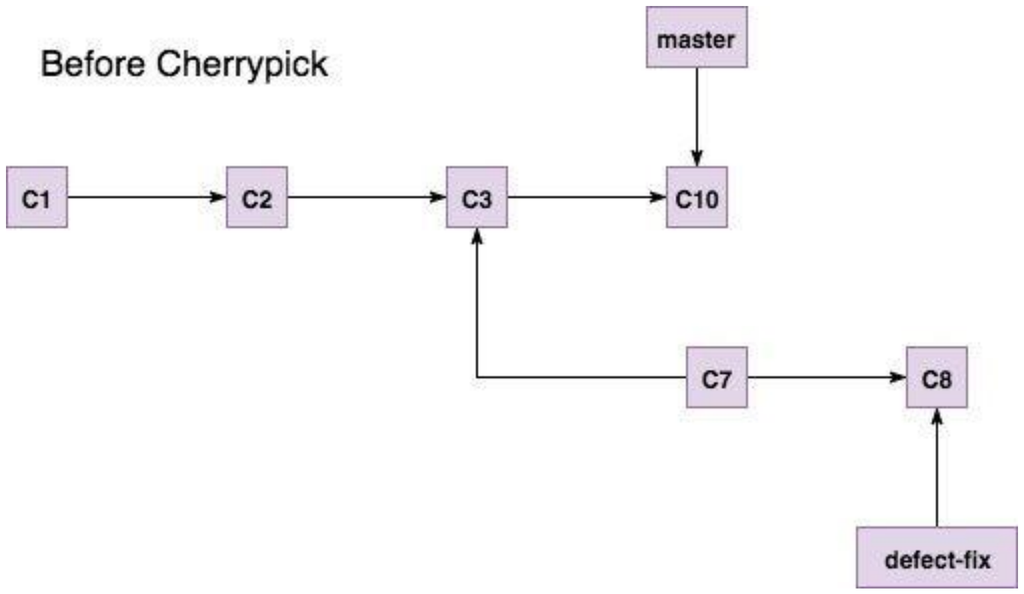
While rebasing there might be conflicts, if so we should resolve them before completing the rebase.

After rebase it enables fast forward merge

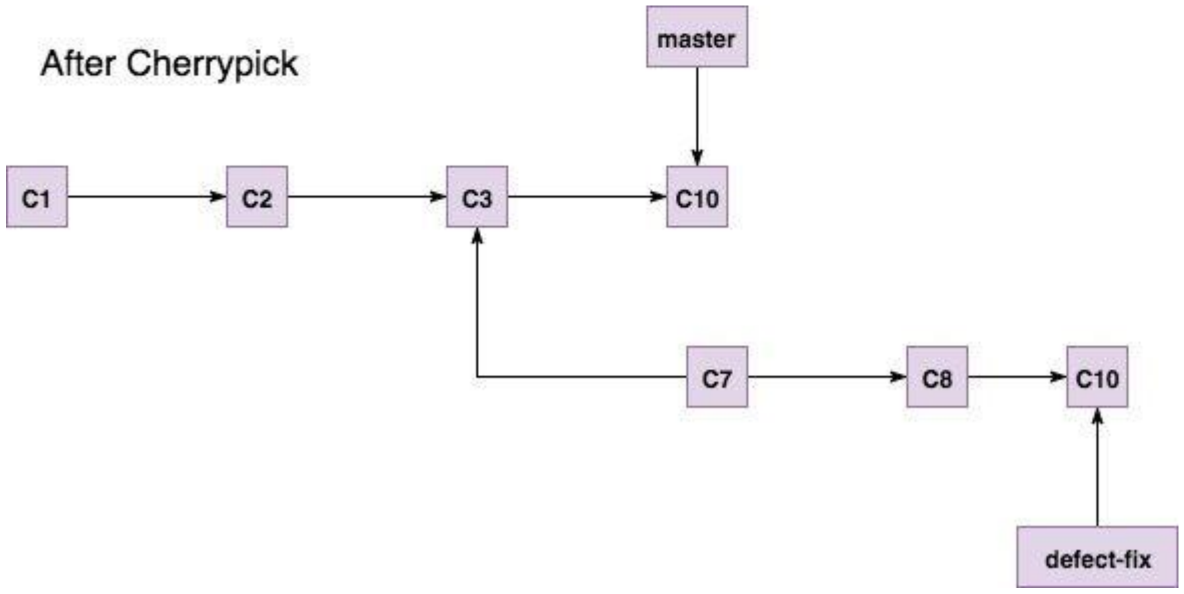
Doing rebase on the commits pushed to remote is not recommended.

Git cherry pick

It picks a commit from different branch and applies it to current branch.



I wrongly committed C10 into master, now i need changes part of C10 in to my 'defect-fix'
The solution is run **git cherry-pick <commit-id>** from 'defect-fix'



Undoing commits

- Reset
- Revert

Git reset:

- Removes the commit from the history

Don't use reset if your commit is pushed to the remote, it causes problems

git reset HEAD~1

Git revert:

It doesn't remove the commit from the history, instead it reverts changes to the files and makes a new commit.
Note: If commits are already pushed to remote we can't use reset instead we should go with revert

Checking list of files modified in a specific commit

git show <commit-id> --name-only --pretty=""

Checking difference between two commits

git diff <commit-id-1> <commit-id-1>

Display merged branches

git branch --merged

Display non merged branches

git branch --no-merged

Git stash

git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and reapply them later on

git stash save
  Stashes all the changes in working/index area and keeps working/index area clean
git stash list
  To display list of stashes
git stash pop
  It moves changes stashed to working/index area
git stash apply
  It copies changes stashed to working/index area, entry still remains in the stash
git stash apply stash@{0}
To apply a specific stash(stash@{0} is stash id)

Git branching strategy

Master branch:

Master must contain well tested code, application release happens from this branch by creating a release ‘tag’.

In real world no one directly work on master.

Hotfixes:

This branch is used for fixing production defects.

Hotfixes branch is created from master

Develop branch:

This branch belongs to a specific team, code integration of this team members are done on this branch

Develop branch is created from master

Feature branch:

Belongs to a specific developer, where his feature in implemented, after completion of a feature changes are merged into his develop branch.

Feature branch is created from develop branch

Release branch:

This branch is to integrate changes done by multiple teams under their develop branch

Git tag

In git tag is a pointer pointing to a specific commit, tag is similar to branch but, branch is used for development/bugfix, whereas tag is used for releasing software.

On a branch we can do commits, but on a tag we cannot perform commits.

Creating a tag

git tag VERSION-1.0.0

Pushing a tag to remote

git push origin VERSION-1.0.0

Delete a tag in local

git tag -d VERSION-1.0.0

Display tags

git tag -d VERSION-1.0.0

Git init

Converts loca folder as a git repository.

- Create a folder with ‘gitdemo’
- *cd gitdemo*
- *git init*
- *Add file to this repository*
- *git add **
- *git commit -m "first commit"*
- Create repository in github.com
- *git remote add origin https://github.com/javahometech/gitdemo.git*
- *git push origin master*

Getting previous version of a file

git checkout fcb83763c4a one.sh

How to check the files modified between two commits

git diff 541f3204a fcb83763c --name-only