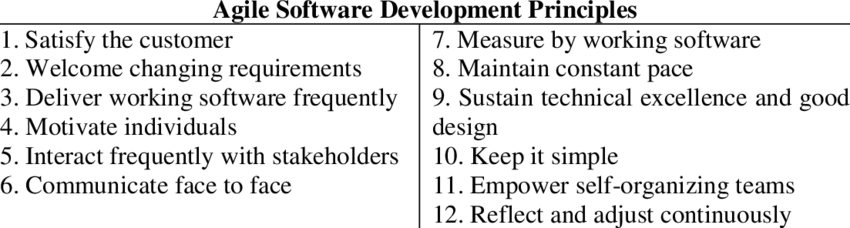
Topic-1: Introduction to Devops

* + What is Devops?
  + Water Fall Model (Plan > Design > code > test > deploy)
  + Agile Model



* + Waterfall vs Scrum
  + Devops vs Agile Models (Devops>> Implementing agile into operation)
  + Top Important points about DevOps.

Topic-2: Introduction to Version Control System

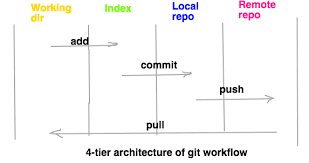
* 2.1. Need of Version Control System?
* 2.2. How version control system will work?
* 2.3. The basic terminology of version control system
* 2.4. Benefits of Version Control System
* 2.5. Types of Version Control Systems

2.5.1. Centralized Version Control System.

2.5.2. Distributed Version Control Systems.

Topic-3: Features and Architecture of GIT

* 3.1 What is GIT? >> VCS
* 3.2 Features of GIT >> Staging area, Branching and merging and Distributed
* 3.3 GIT Architecture



Topic-4: Life Cycle of File in GIT:

* 4.1 Untracked
* 4.2 Staged
* 4.3 In Repository/ Committed
* 4.4 Modified

Topic-5: Git Installation On Ubuntu(apt-get install git ; git --version)

Topic-6: Example-1 To Understand Working Directory, Staging Area(AES 128) and Local Repository

Topic-7: The 6 Git Commands With Example - init,status,add,commit,log and config

Topic-8: The complete post mortem of git log command

8.1 How to see history of all commits in local repository >> git log

8.2 How to see log information of a particular file >> git log <file-name>

* Option-1: --online option to get brief log information>> git log --oneline
* Option-2: -n option to limit the number of commits to display>> git log -n
* Option-3: --grep option to search based on given pattern in commit message >>git log --grep=”pattern”
* Option-4: Show commits more recent than a specific time.>> git log --since=”5 minutes ago”
* Option-5: Show commits older than a specific time>> --until=<date> --before=<date>
* Option-6: Show commits based on author >> git log –author=”name”
* Option-7: --decorate option to display extra information (This option will print some extra information like authername, branch and tagname) >> git log –decorate --oneline

Topic-9: The Complete Story of git diff command

Case-1: To see the difference in file content between working directory and staging area git diff file1.txt diff --git a/file1.txt b/file1.txt txt a/file1.txt means source copy which means staging area. b/file1.txt means destination copy which means working directory copy.

Case-2: To see the difference in file content between working directory and last commit git diff HEAD file1.txt

Case-3: To see the difference in file content between staged copy and last commit git diff --staged HEAD file1.tx

Case-4: To see the difference in file content between specific commit and working directory copy git diff 7chracters\_of\_specified\_commitid filename

Case-5: To see the difference in file content between specific commit and staging area copy git diff --staged e5705a6 file1.txt

Case-6: To see the difference in file content between two specified commits git diff e5705a6 6745461 file1.txt

Case-7: To see the difference in file content between last commit and last butone commit git diff HEAD HEAD^ file1.txt

Case-8: To see the differences in all files content between two specified commits git diff 6745461 be5256c

Case-9: To see the differences in content between two branches git diff master test

Case-10: To see the differences in content between local and remote repositories git diff master origin/master

Topic-10: Helix Visual Merge Tool(p4merge) For Checking Differences (Gerrit)

Topic-11: Removing files by using git rm command

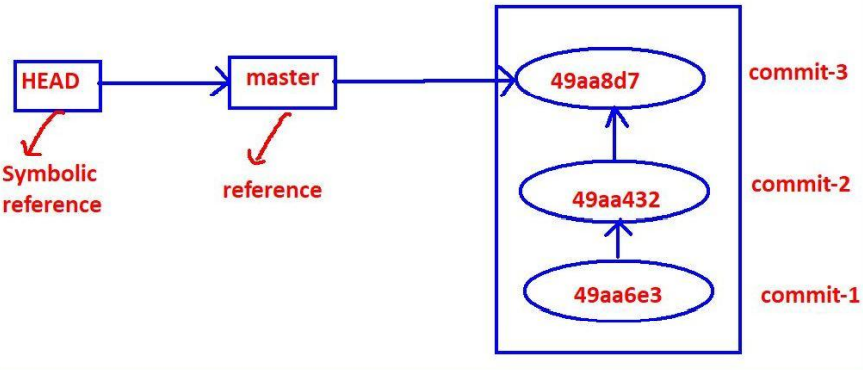
* Case-1: To remove files from working directory and staging area(git rm)
* Case-2: To remove files Only from staging area (git rm --cached)
* Case-3: To remove files Only from Working Directory (rm command)

Topic-12: Undo changes with git checkout command

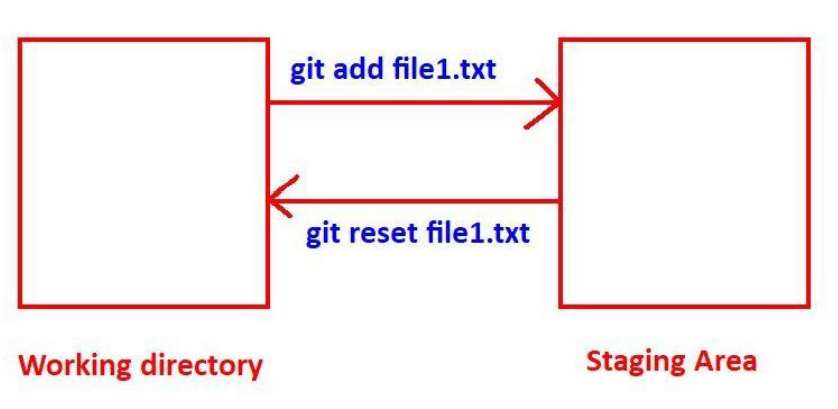
We can use checkout command to discard unstanged changes in the tracked files of working directory. Observe the 3 words:

* Only for working directory
* To discard unstaged changes(The changes which are not added to staging area)
* In the tracked files (The files which are already added to staging area/commit) It is something like undo operation. It will copy contents of the file from index area(staging area) to working directory. git checkout -- filename

Topic-13: Git References (master and HEAD )

Topic-14: Git reset command

Utility-1: To remove changes from staging area



Utility-2: To undo commits at repository level(--mixed, --soft, --hard modes)



Topic-15: Git Aliases - Providing our own convenient names to git commands

Syntax: git config --global alias.aliasname "original command without git" Eg: git config --global alias.one "log --oneline"

Topic-16: Ignoring unwanted files and directories by using .gitignore file>> This file need to create in the working directory

Topic-17: Any Special Treatment for directories by Git ??? >> No Version control would be apply to directory. But if it has file then VC would be applied. (mkdir logfile(no VC)>> touch logfile/access.log> VCS applicable.)

Topic-18: Branching and Merging

18.1. What is branching? >> creating parallel line of development

18.2. Need of creating a new branch >> without disturbing exiting setup we can parallelly work

18.3. Various Commands used in branching

* To view branches >> git branch
* To create a new branch >> git branch branch-name
* To switch from one branch to another branch >> git checkout branch-name
* Short-cut way to create a new branch and switch to that branch 1 >> git checkout -b branchname

18.4. Demo Example for branching >> create branch and work on it.

18.5. Multiple use cases where branching is required

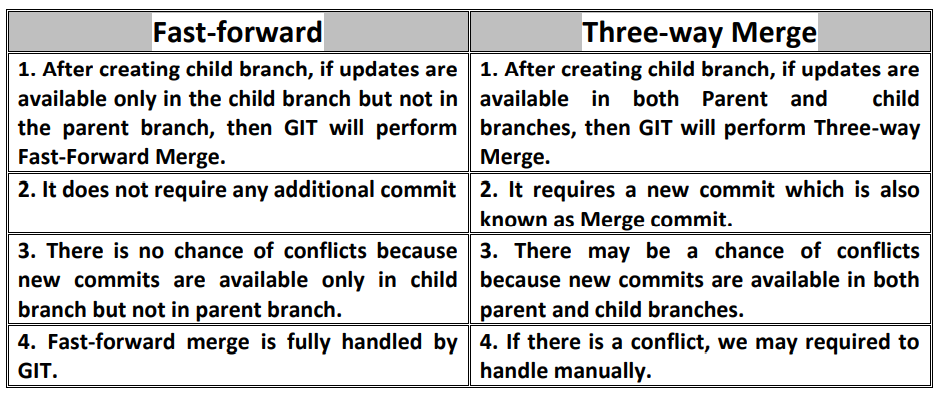
18.6. Advantages of Branching

18.7. Merging of a Branch >> git merge feature

18.8. What is Fast-forward Merge? >> No commit post child branch created.

18.9. What is Three-Way Merge? >> master branch also have changes.

18.10. Differences between Fast-forward and Three-way Merges



18.11. Merge Conflicts and Resolution Process>> by default git will do auto merge, but if it faces any issue we need to do merge manually.

18.12.How to Delete a Branch >> git branch -d branchname

Topic-19: Merging by using Rebase

19.1 Process of rebasing- Making linear history.

It is a two-step process.

Step-1: We have to rebase feature branch on top of master branch.

1. Checkout feature branch git checkout feature
2. Rebase feature branch on top of master branch git rebase master

Step-2: We have to merge feature branch into the master branch (fast-forwar merge will be happend)

A. checkout master branch git checkout master B. Merge feature branch into master branch git merge feature

19.2. Demo Example for rebasing

19.3. Advantages of rebasing

1. Rebase keeps history linear. In 3-way merge, a commit can have multiple parents. But in Rebase every commit has a single parent only. Hence history will be liner.

2. Clear work flow (Linear) will there. Hence easy to understand for the developers.

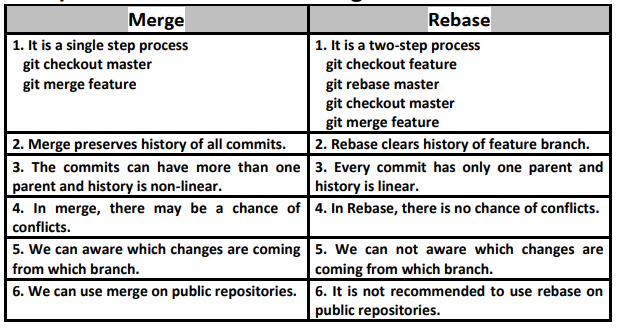
3. Internally git performs Fast-forward merge and hence there is no chance of conflicts. 4. No extra commit like merge commit.

19.4. Disadvantages of rebasing

1. It rewrites history. We cannot see history of commits what we did in feature branches

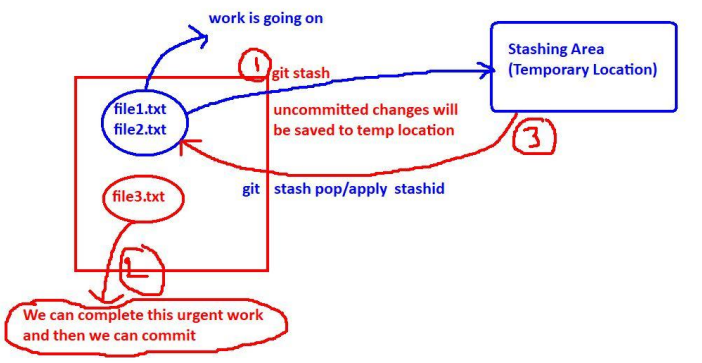
2. We does not aware which changes are coming from which branch.

19.5 Differences between Merge and Rebase



Topic-20: Stash in GIT

20.1 What is git stash? >> It’s tmp area where we can store tmp data.



20.2 Demo Example for stashing >> git stash

20.3 How to list all available stashes? >> git stash list

20.4 How to check the contents of stash?>> git show stash@{0}

20.5 How to perform unstash? We have to bring files from temporary location to our working directory. For this we have to perform unstash operation. We can perform unstashing in 2 ways:

* By using git stash pop >> Temp entry would be deleted.
* By using git stash apply>> Temp entry wouldn’t be delete.

20.6 Partial Stash >> git stash -p

20.7 How to delete the stash?

git stash clear >> To delete all stashes git stash drop stashid >>To delete a particular stash

Topic-21: Working with Remote Repositories

21.1) Need of Remote Repositories

21.2) How to work with GitHub?

21.3) How to create account in GitHub?

21.4) How to create a new reposioty in Github?

21.5) How to work with Remote Repository?

1) git remote 2) git push 3) git clone 4) git pull 5) git fetch

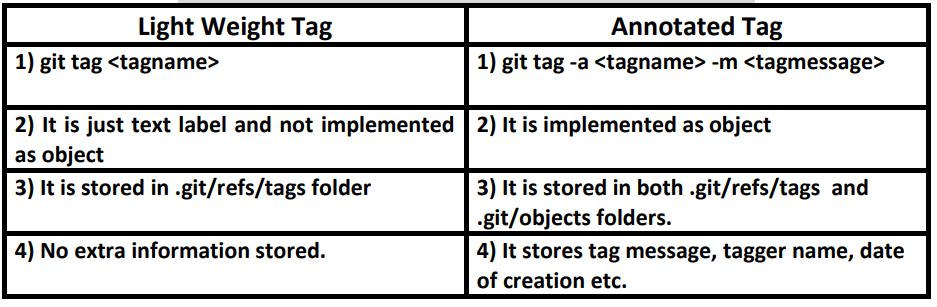
Topic-22: Git Tagging (git Tag Command)

22.1) Introduction to Tagging >> When we achieve some milestone.

22.2) Creation of a Light Weight Tag >> git tag <tag-name>

22.3) How to delete a tag? >> git tag -d tagename

22.4) Annotated Tags (Tags with Information) >> git tag -a -m “message”



22.5) How to tag a previous commit? >> git tag -a <tagname> <commitid> -m <tagmessage>

22.6) How to update an existing tag?

We can do this by using the following 2 ways.

1) Delete the tag and Recreate tag with corresponding correct commit id. $ git tag -a -f -m <tagmessage

2) By using -f or --force option to replace an existing tag without deleting.

22.7) How to compare Tags >>git diff V-1.0.0 V-1.1.0

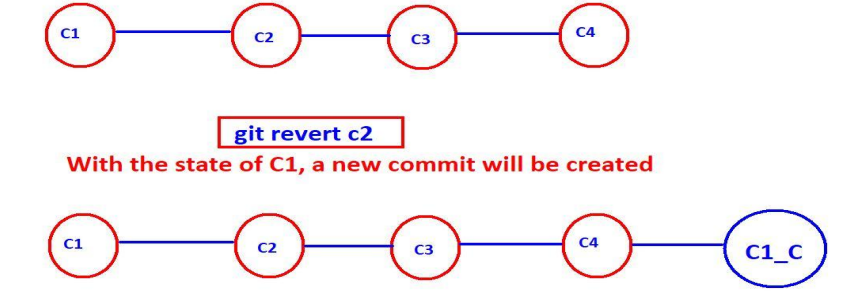
22.8) How to push tags to remote repository? By default tag would be not pushed in remote repository, we need to do it manually. git push origin <tagname>

Topic-23: git revert Command

23.1) Need of revert command >>

Suppose if this code already pushed to the remote repository and already several people pulled this code and starts working on that code, git reset command may create big problem. git reset command deletes commit history and hence it is destructive command and not recommended to use on public repositories. To overcome this problem, we have to go for git revert command. git revert command won't delete commit history. It reverts the required commit by creating a new commit. i.e it will undo a particular commit without deleting commit history.

23.2) Demo Example to revert last commit



23.3) Demo example to revert a particular commit

Topic-24: Cherry-Picking (git cherry-pick Command)

24.1) Need of Cherry-Picking : if you want to use someone’s branch commit in your branch which is not pushed for master then you can use cherry pick.

24.2) Use cases of cherry-pick 1) Sharing code between branch, and we can use in bug fixes.

24.3) Demo Example for cherry-picking >> cherry pick commit-id.

Topic-25: git reflog Command

reflog -> means reference log We can use git reflog command to display all git operations what ever we performed on local repository. We can use git reflog command to know only local repository operations but not remote repository operations.

Casestudy: suppose we have performed --hard reset so it’s very difficult for us to trace those things, but with the help of reflog we can easily do that.

Topic 26: Git fork

Forking is a git clone operation executed on a server copy of a projects repo.

Topic 27: GIT restore

enovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/project1 (master) $ git status On branch master Changes not staged for commit: (use "git add ..." to update what will be committed) (use "git restore ..." to discard changes in working directory) modified: a.txt modified: b.txt

Topic 28: Meta data and GIT fetch

Transactions on git repositories are called “commits”. Each commit is an atomic change to the files in the repository. For each commit, git maintains data for tracking what changed, and some metadata about who committed the change, when, which files were affected, etc.

Trunk-based development (TBD) is **a branching model for software development where developers merge every new feature, bug fix, or other code change to one central branch in the version control system**. This branch is called “trunk”, “mainline”, or in Git, the “master branch”