

What is a Version?

What is VCS also called?

what is Version control system?

what happens if we dont have an VCS?

What is the mechanism used to save the files in the Remote Repository (vtz.GIT)?

Benefits of VCS?

Types of VCS?

Difference between the Centralized, Distributed VCS?

What is GIT?

Advantages of the GIT?

Ways to use the GIT?

Configuration of GIT?

Line Ending in CRLF?

Difference between git and other version control tools?

Git Workflow:

Does the git consumes a lot of space for storing every snapshot?

what are the stages in the GIT?

What are GIT objects?

Staging in GIT?

Comman mistake every developer things about git?

What details we have in the commit?

Best committing practices:

Ignoring Files:

What is diff in the git

How to use the VScode to compare the files using diff?

Git Fork:

Branching

Merge conflicts

Merge vs Rebase

Timeline of VCS:

Command-line text editors in Linux:

Linux Commands:

What does .git folder contains after initializing git init?

What are the different repository in git?

How to discard the local changes of the file that we made and restore to the fresh file before the changes?

Restoring the Versions of the files?

GIT commands

To enter the more commit message

What is a Version?

It is a snapshot or state of the files and folders in a codebase at a given time can be called a "version."

What is VCS also called?

A version control system can also be referred to as a

source control system

source code management

version control software

version control tools

what is Version control system?

in a nutshell with a version control system we can track our project history. version control system records the changes made to our code over time in a special database called repository. we can look at our project history and see who has made what changes when and why and if we screw something up we can easily revert our project back to an earlier state

what happens if we dont have an VCS?

without a version control system we'll have to constantly store copies of the entire project in various folders this is very slow and doesn't scale at all especially if multiple people have to work on the same project you would have to constantly toss around the latest code via email or some other mechanisms and then manually merge the changes so

What is the mechanism used to save the files in the Remote Repository (vz.GIT)?

There is a metadata used to store all the information

Benefits of VCS?

Track changes - Track changes of who are working and editing the files

Backup - It also works as Backup, we can go to the previos version when needed

Collaboration - multiple people can work at the same time without overriding each others code

Flexibility - we can experiment new ideas without effecting the old code that already saved in a main branch

security - since the repository is central it can be backed up and protected to help the integrity and security of the project.

Saves the history of the file changes

Simultaneously working

Branching and Merging

Traceability

Types of VCS?

Localized Version Control System - First Generation

Ex: SCCS (Source Code Control System)

RCS (Revision Control System)

Centralized Version Control System - Second Generation

Ex: CVS (Concurrent Versions System)

SVN (Apache Subversion)

Team Foundation Server(Microsoft)

Perforce Helix Core

Distributed Version Control System - Third Generation

Ex: Git

Mercurial

BitKeeper

Darcs (Darcs Advanced Revision Control System)

Monotone

Bazaar

Fossil

Pijul

Difference between the Centralized, Distributed VCS?

In a centralized system all team members connect to a central server to get the latest copy of the code and to share their changes with others the problem with the centralized architecture is the single point of failure if the server goes offline we cannot collaborate or save snapshots of our project. so we have to wait until the server comes back online. Sin

In distributed systems we don't have these problems every team member has a copy of the project with its history on their machine so we can save snapshots of our project locally on our machine. if the central server is offline we can synchronize our work directly with others

What is GIT?

Git is a stream of snapshots. which is free an open source VCS.

It keeps track of all the changes of files, who have created, updated, deleted.

Ex: Lets assume there are 2 developers A & B. If for suppose developer A worked on a file and pushed to the Remote repository, due to some enhancements developer B worked on the same file and pushed the code to the remote repository, there might be a break of production as there are editing same file, so then we should track back or checkout to the previous version of the File to keep the applicaton up and runnig. This is the main resaon version control is used.

Advantages of the GIT?

Free

Open source

Super fast

Scalable

operations like branching and merging are slow and painful in other version control systems like subversion or tfs but they're very fast and git

Ways to use the GIT?

Command Line: Git Bash(Bourne again shell)

GUI Ex: GUI client, GitKraken, SOurce Tree

COde Editors

Ex: Git lens extension in visual studio code

Configuration of GIT?

Before using the git we should specify some configuration settings. They are:

Name

EMail

Default Editor

Line Ending

We can specify these configuration settings in 3 levels

System - sets to all levels in current computer

Global - sets to all users and respositories of current user

Local - sets to the current repository

Line Ending in CRLF?

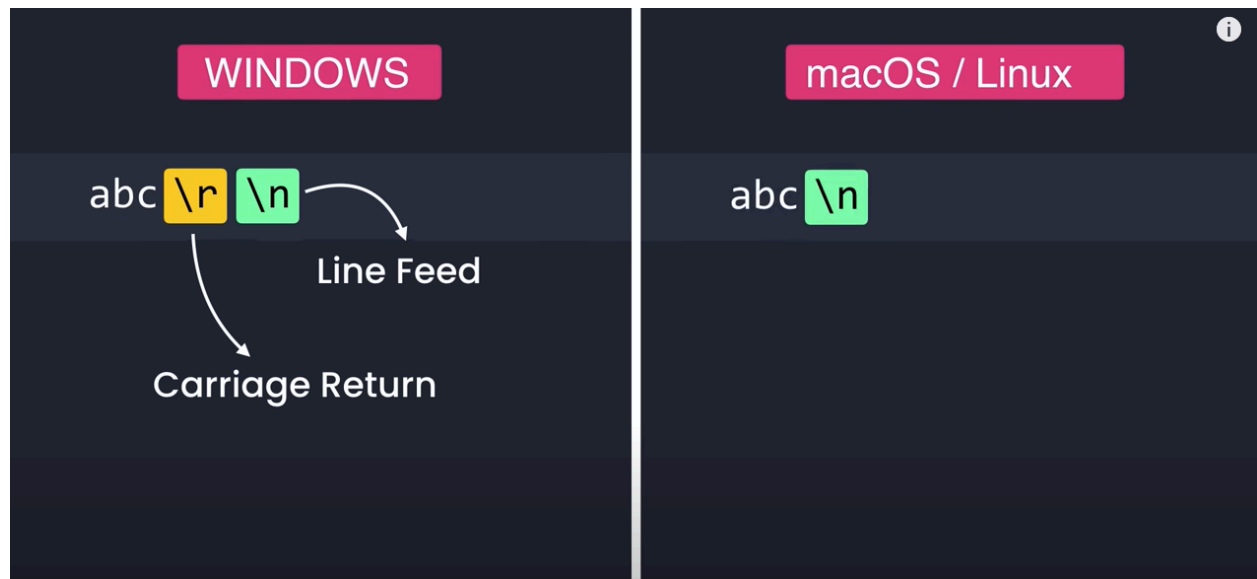
CRLF is known as Carriage Return Line Fit. This is an important setting need to be done while configuring the GIT, else we encounter warnings in future. To prevent this we have to configure core.autocrlf.

Windows:

End of lines are marked with the CRLF, when check in his code into the repository git removes the carriage return character from end of lines similarly when he checks out his code from the repository git should update end of lines and add the carriage return character

Mac/linux:

when checks out the code we don't want the carrier's return character so git shouldn't touch end of lines however if carrie's return is accidentally added to end of lines perhaps because of the editor, git should remove it when storing the code in the repository



Difference between git and other version control tools?

Git has a special area called staging area or INDEX, that's an intermediate area where we keep the modified files or the next commit i.e add the modified files to this area and review the changes and then create the next snapshot for the next commit.

Unlike the other VCS git doesn't store the deltas of what its changed, it stores the full content, so its easy to quickly restore the earlier snapshot of our project without having to compute the change.

Git Workflow:

We have project directory in the Local and a Git repository, which is actually a hidden repository in the project directory.

Creating a commit is like taking a snapshot of the project.

Does the git consumes a lot of space for storing every snapshot?

No, because git is very efficient in data storage, it compresses file contents and doesn't store the duplicate content.

what are the stages in the GIT?

There are 3 stages in the git

Unstaged area/working directory/untracked state

staged area/ Tracked state

commit: For each commit there will a unique identifier created, SO in gits database we have an object created with the ID.

Files are represented using blobs

Folders are represented as trees

What are GIT objects?

The objects in the git can be

commits

blobs (files)

Trees (Directories)

Tags

Staging in GIT?

This is a place where we add our modified files and review our changes, if everything is good we commit our code to the repository so it creates a snapshot. If we want some files to be changed or modify them we can review and unstage them.

Common mistake every developer things about git?

Once we commit the code we think the staging area is empty but it doesn't, we have the same snapshot of the recent committed code in the repository.

What details we have in the commit?

ID: unique number (a revision number) for every commit.

Message:

Date/Time:

Author

Complete snapshot of the project at the time its created.

Best committing practices:

It should be too small or too big

Ignoring Files:

If we want to ignore some files to not add to the git repository, for example log files that are used by the each developer but not need in the Git repository. So here we add all the files which we want the git to ignore them and don't want the git to track those files. It only applies to the files which are newly created and added to the .gitignore file. It doesn't ignore the files that's already in the git repository.

Echo logs/ > .gitignore // for 1<sup>st</sup> time

Echo logs/ >> .gitignore // from next time

What is diff in the git

git-diff -Diffing Shows the changes between commits, commit and working tree. The Famous diff tools are (comparing the files)

kDiff3

P4Merge

WinMerge

VScode

How to use the VScode to compare the files using diff?

First we should let the git to know that we use the VScode to compare the files using the diff tool.

Git config --global diff.tool vscode

Git config --global difftool.vscode.cmd "code -wait --diff \$LOCAL \$REMOTE"

Git config --global -e



Git Fork:

If someone not in our team and want to contribute to our project.

Branching:

Git allows us to create branches but it doesn't tell us how to use them.

Different strategies:

    Mainline Development

        Always we have to be integrating our code with the Repository

        Few branches

        Relatively small commits

        High quality testing & QA standards

    State, Release and Feature Branches

        Feature branch

        Develop branch

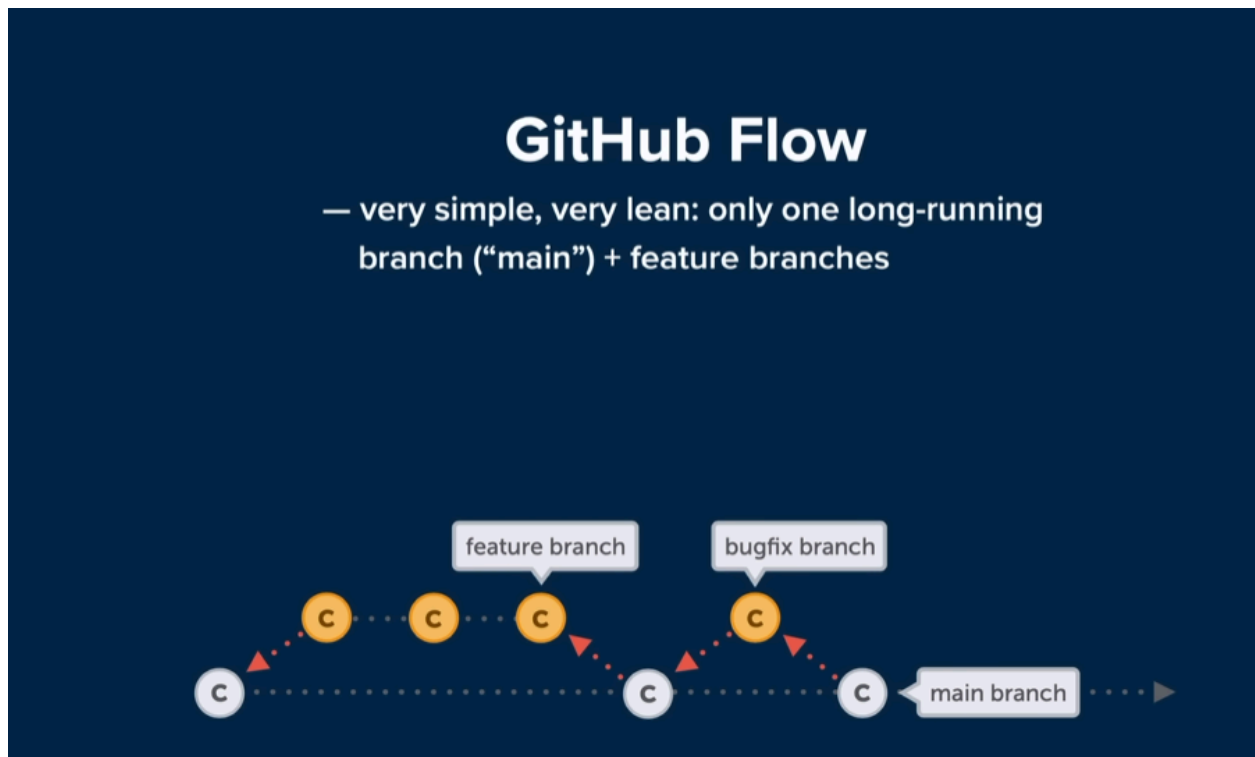
        Main branch

    Long-Running & short lived branches

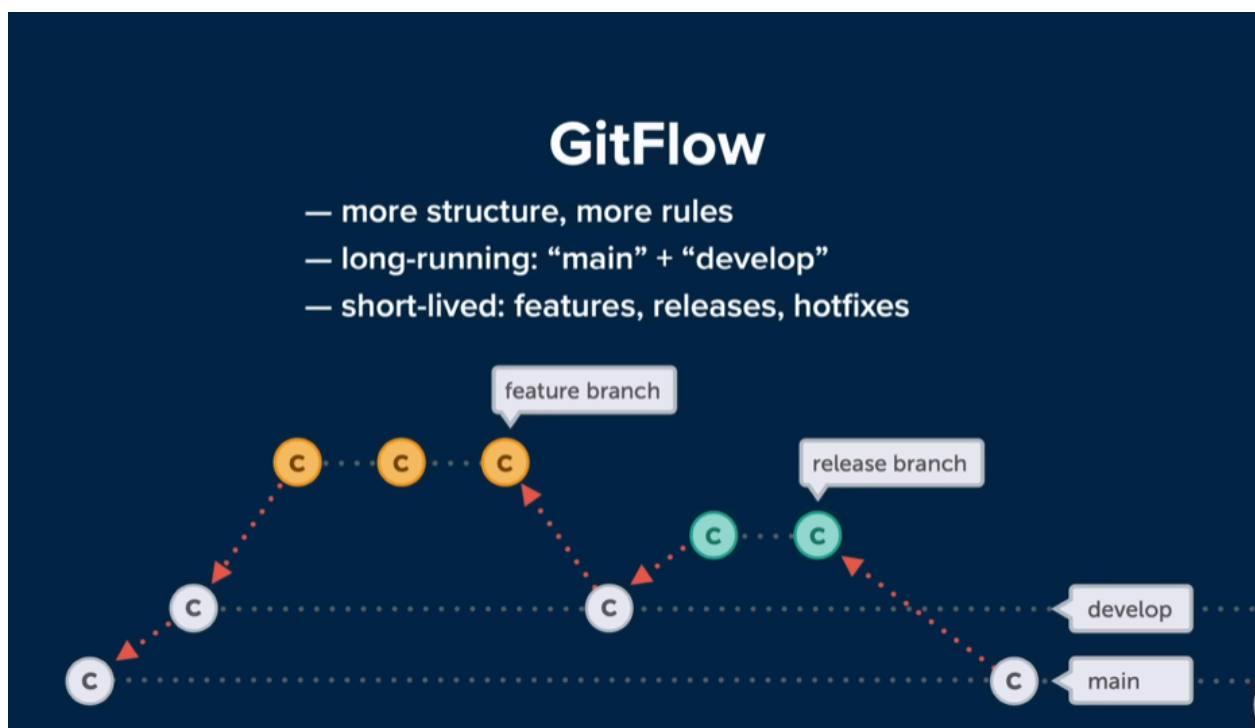
        Dev, staging, pre production, production, integration branch(dev-staging)

        Short lived branches- feature, bug fix

Git Hub flow branching strategy:



GitFlow:



Merge conflicts:



Merge vs Rebase:

- fast-forward Merge

- Git rebase branch-B

  - We can use the rebase on commits before

Timeline of VCS:

- <https://initialcommit.com/img/initialcommit/vcs-timeline.png>

Command-line text editors in Linux:

- There are two command-line text editors in Linux

- VIM

- NANO <https://docs.rackspace.com/support/how-to/command-line-text-editors-in-linux/>

Linux Commands:

- [https://github.com/chanduujjina/GitPractice\\_2023\\_B2/blob/main/basic\\_linux\\_commands.adoc](https://github.com/chanduujjina/GitPractice_2023_B2/blob/main/basic_linux_commands.adoc)

What does .git folder contains after initializing git init?

What are the different repository in git?

Local

How to discard the local changes of the file that we made and restore to the fresh file before the changes?

Git restore filename

Git takes the version of the file from the staging area and replaces it with the file in the working directory.

For new file that just got created git doesn't know where to get the previous changes and restore the file, because there is no previous version of the file in staging area or the repository.

To remove the changes of the untracked files (newly created file)

Git clean -f or -d for directories

Restoring the Versions of the files?

If we have deleted the file unexpectedly and now we want to undo or revert the last commit or restore.

Need to learn the revert and undo the commit

Restore the commit

Git restore the commit before the last commit, the last commit will be the deleted commit. This restores a file to a previous version before the once that's already deleted

Git restore --source=HEAD~1 filename

Some of the GIT commands:

Git help

git config --help

git config -h

To Know the Git version

```
git --version
```

configure the user name

```
git config --global user.name "Bhargav"
```

configure the user email

```
git config --global user.email abc@gmail.com
```

configure default editor, as wait flag is set it tells the editor to wait until the default editor is closed

```
git config --global core.editor "code --wait"
```

opens the default editor and opens the file called .gitconfig to show all the global settings

```
git config --global -e
```

setting core CRLF

In windows

```
git config --global core.autocrlf true
```

In Mac/Linux

```
git config --global core.autocrlf input
```

To Initialize empty repository

```
git init
```

Create file in git

```
cat > fileName
```

```
touch filename
```

Move single file from unstaged to staged

```
git add fileName
```

Move all file from unstaged to staged

```
git add . (. Is called period)
```

```
git add *.txt (* is called the patterns)
```

TO add chunk of data from the particular file to staging

```
Git add -p filename
```

list the files and folders

```
ls
```

list the hidden files and folders

```
ls -a
```

To check the files in the staging any time

```
git ls-files
```

colorful git terminal

Mac: install zsh

Windows: posh-git

Move file from staged area to commit area

```
git commit -m"commit message"
```

undo merge conflicts:

```
git merge abort
```

```
git rebase --abort
```

To enter the more commit message

Git commit //It opens the code editor and in that it opens the file COMMIT\_EDITMSG, to add more commit message rather than a single line.



Move file from staged area to unstaged area

git restore --staged filename

How the restore works ?

We use the restore or reset (in previous versions), to undo the file from staging area to the unstaged area i.e working directory.

For an existing files as we know git maintains an snapshot of all the files in the repository, it takes an snapshot from the repository and undo the file or folder from the staging to the untracked area .

For a new file that is just added to the staging area, it takes the file back to the untracked as the file doesn't have any snapshot from the repository


```
~/Projects/Moon git p master + !
> ✓ git status -s
M file1.js
A file2.js

~/Projects/Moon git p master + !
> ✓ git restore --staged file2.js

~/Projects/Moon git p master ! ?
> ✓ git status -s
M file1.js
?? file2.js
└─┘

~/Projects/Moon git p master ! ?
> ✓
```

we have two question marks



To check the file status in git(wether in untracked/tracked/commit area)

```
git status
```

```
git status -s // gives the short form of the status
```

How to see the exact lines changed for the particular file after adding to the staging area.

```
Git diff --staged
```

Diff is a utility

TO check the changes in the files in the working directory

```
Git diff
```

To delete single file permanently from untracked state

```
git clean -f filename
```

To delete all files permanently from untracked state

```
git clean -f .
```



commit the code

```
git commit -m "commit message"
```

commit the snapshot without the staging the code

```
git commit -am "commit message"
```

To see the committed files

```
Git log
```

to see committed files in one line

```
git log --oneline
```

```
git log --oneline --reverse
```

To see the contents of the commit

```
Git show unique identifier ex: git show d640008q
```

```
Git show HEAD~1 // ~ tilde it used to see the changes of the files in a 1st commit i.e
```

Head

```
Git show HEAD~1:filename // to see the exact version of the commit, the final file contents after changed of the committed file
```

TO see all the files in a commit

```
Git ls-tree //as we discussed earlier the git staging area saves all the files (snapshot) of the working directory
```

```
Git ls-tree
```

What is a tree?

Tree is a data structure for representing hierarchical information so in a tree we can have nodes and these nodes can have children now a directory on the file system can be represented using a tree because each directory can have children these children can be files and other subdirectories

UNDO commands:

To delete repository or folder

```
rm -rf .git
```

To delete file in repository

```
git rm fileName
```

move files from commit area to staged area

```
git reset --soft HEAD~1
```

move single file from staged to unstaged area

```
git restore --staged filename
```

move all file from staged to unstaged state

```
git rm -r --cached .
```

move single file from staged to unstages state

```
git rm --cached filename
```

Move file from commit area to staged area

```
git reset --soft head~1
```

Delete file from commit area Permently

```
git reset --hard head~1
```

How to convert normal desktop folder to git repository?

create an Normal folder on desktop

```
Java_practice
```

create an git repository in the Github withe the same name as the local folder

Java\_practice

Now open the local folder in the GitBash and initialize the git command

```
git init
```

```
git add .
```

```
git clone -m "commit message"
```

```
git remote add origin gitlink
```

```
git remote -v (To check origin is set or not)
```

```
git branch -M branchname
```

for the first time push we should set the upstream

```
git push --set-upstream origin master
```

Difference between upstream and the origin?

<https://stackoverflow.com/questions/9257533/what-is-the-difference-between-origin-and-upstream-on-github>

Issues:

GIT:

On a Windows machine, I added some files using git add. I got warnings saying?

warning: in the working copy of 'Test1.txt', LF will be replaced by CRLF the next time Git touches it

<https://stackoverflow.com/a/20653073>

Reference Links:

Version Control System - <https://initialcommit.com/blog/Technical-Guide-VCS-Internals>

commonly used Git commands:

<https://training.github.com/downloads/github-git-cheat-sheet.pdf>