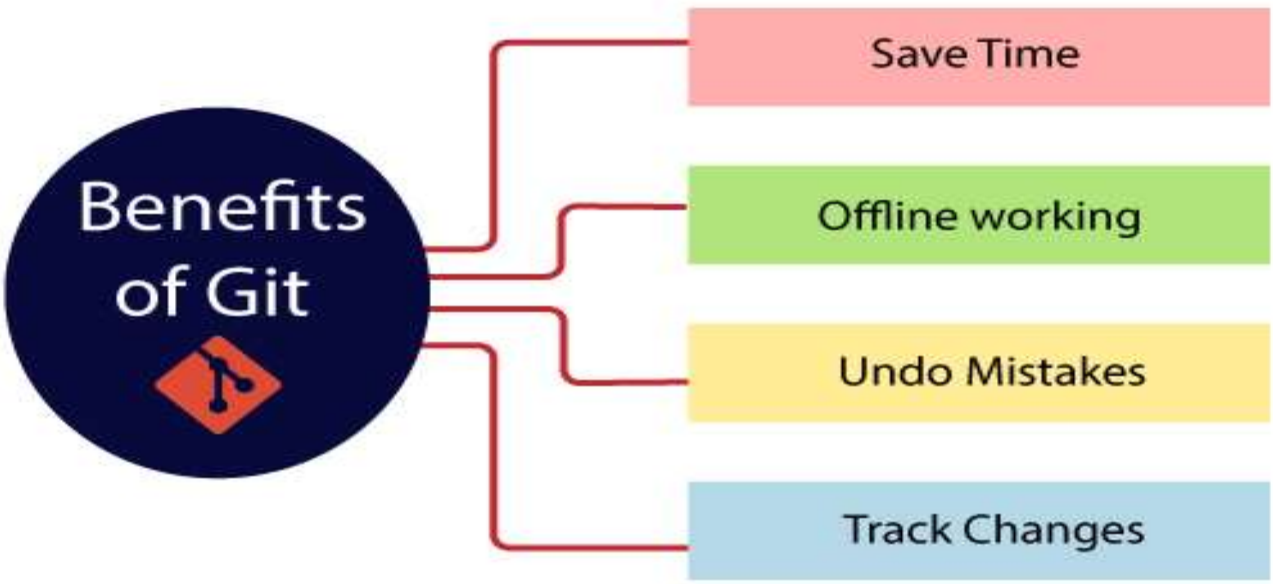
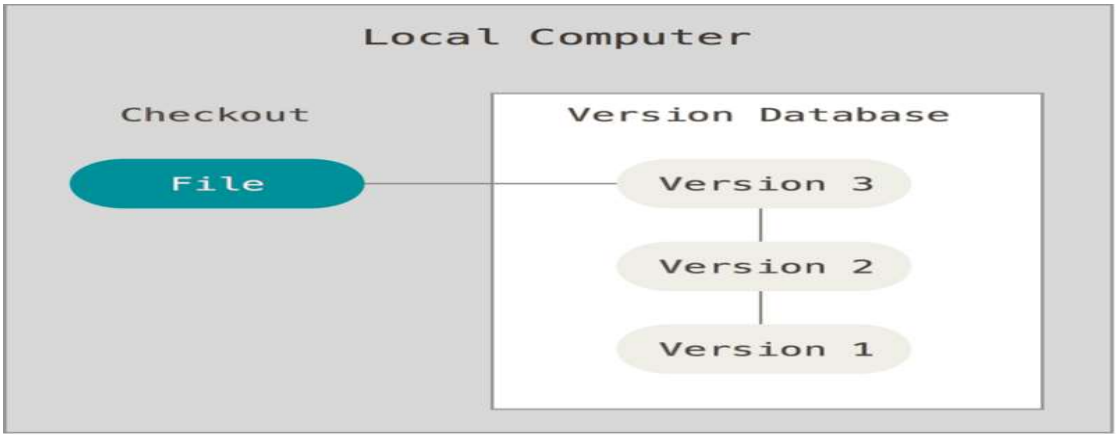

git


Benefits of Git
Save Time
Offline working
Undo Mistakes
Track Changes

Version Control

- Version control In software engineering, version control is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information.
- Version control is a component of software configuration management.
- Version control enables multiple people to simultaneously work on a single project.
- Each person edits his or her own copy of the files and chooses when to share those changes with the rest of the team. Thus, temporary or partial edits by one person do not interfere with another person's work.



Benefits of the version control system:

- Enhances the project development speed by providing efficient collaboration,
- Leverages the productivity, expedite product delivery, and skills of the employees through better communication and assistance,
- Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change,
- Employees or contributor of the project can contribute from anywhere irrespective of the different geographical locations through this **VCS,**
  For each different contributor of the project a different working copy is maintained and not merged to the main file unless the working copy is validated. A most popular example is **Git, Helix core, Microsoft TFS,**

- Helps in recovery in case of any disaster or contingent situation,
- Informs us about Who, What, When, Why changes have been made

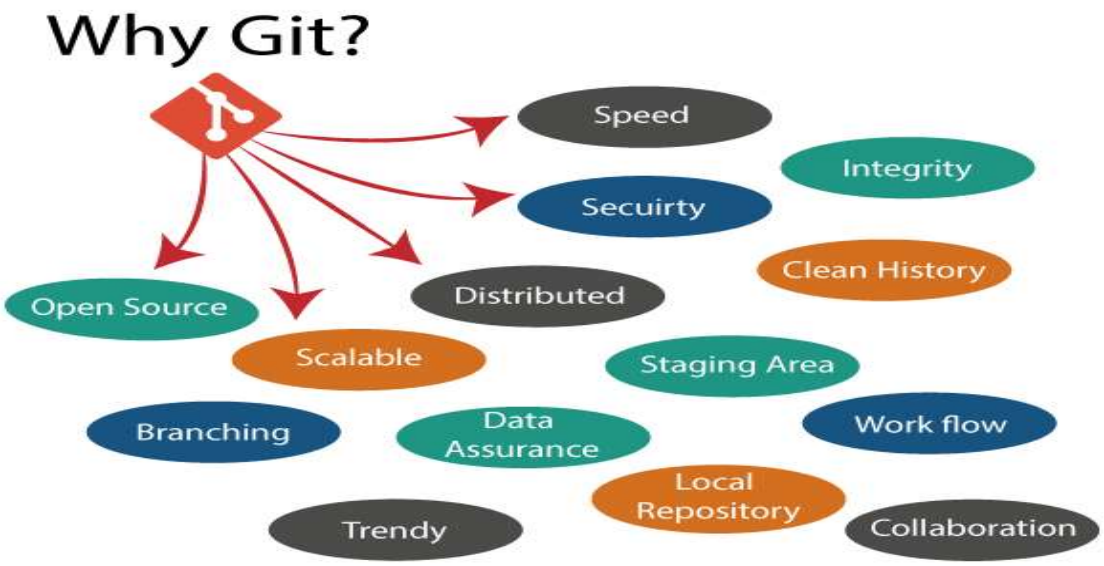
WhAt tools Are used for version control

Version control systems are a category of software tools that help a software team manage changes to source code over time.

- Git
- CVS
- SVN
- Mercurial
- Monotone
- Bazaar
- TFS

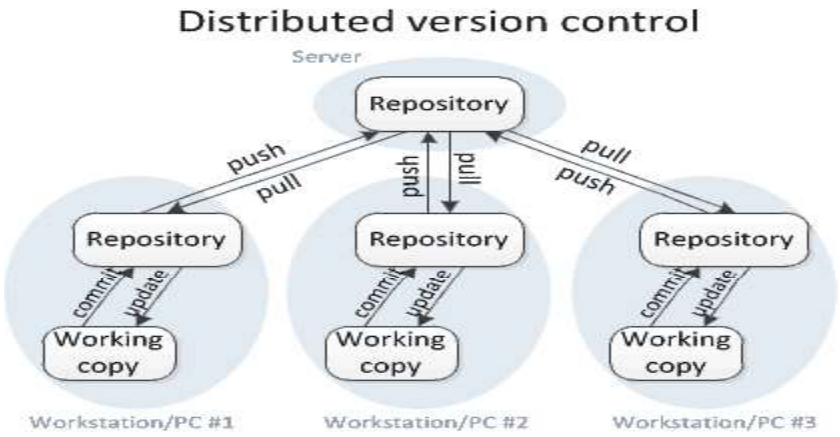
Why-Git

- Git is the most commonly used version control system.
- Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to.
- Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.



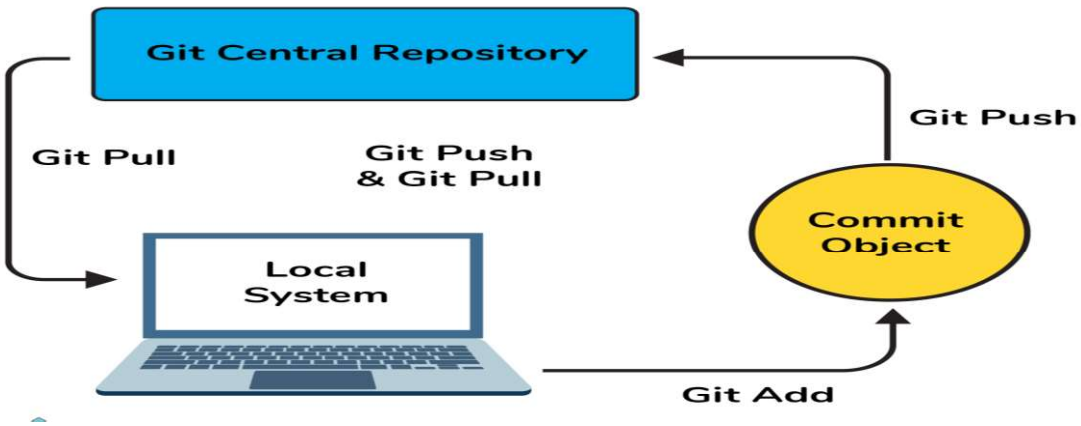
DVCS Distributed Version Control System

- A distributed version control system (DVCS) is a type of version control where the complete codebase — including its full version history — is mirrored on every developer's computer. Distributed development allows for multiple people to be working on the same project at the same time.
- Without version control you would not be able to work on the same file as someone else as one of your changes would overwrite the other's code when committing.



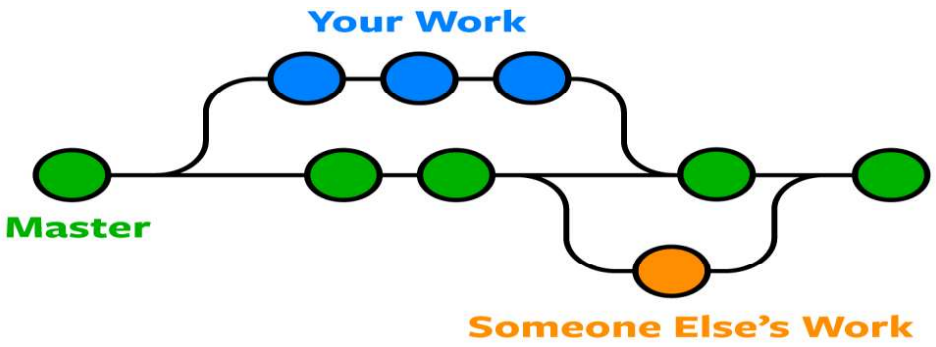
Git Components

- The working directory is a single checkout of one version of the project.
- Git add command to move those changes from the working directory to the staging area.
- Git, commit is the term used for saving changes. Git does not add changes to a commit automatically.



Git BrAnch

- Git branches are effectively a pointer to a snapshot of your changes. ... Instead of copying files from directory to directory, Git stores a branch as a reference to a commit. In this sense, a branch represents the tip of a series of commits—it's not a container for commits



WhAt Are the types of brAnches in Git

This workflow consists of five types of branches, each with different roles:

- Master
- Feature branch (aka Topic branch)
- Release branch.
- Hotfix branch.
- Develop branch (aka Integration branch

MAster BrAnch

- Git, "master" is a naming convention for a branch. After cloning (downloading) a project from a remote server, the resulting local repository has a single local branch: the so-called "master" branch. This means that "master" can be seen as a repository's "default" branch

Release Branch

- Release branches contain production ready new features The release branch helps isolate the development of an upcoming version and the current release. The release branch's lifetime ends when a particular version of a project is released

Feature Branch

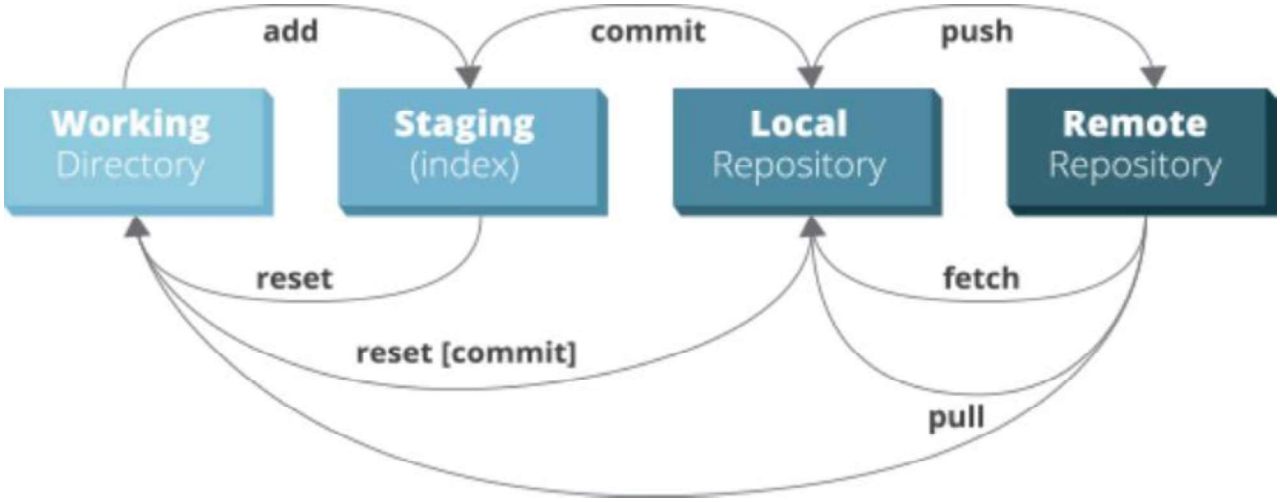
- Feature branch is a source code branching pattern where a developer opens a branch when she starts working on a new feature. She does all the work on the feature on this branch and integrates the changes with the rest of the team when the feature is done.

Hotfix Branch

- Maintenance or “hotfix” branches are used to quickly patch production releases

Git Pull vs Git fetch

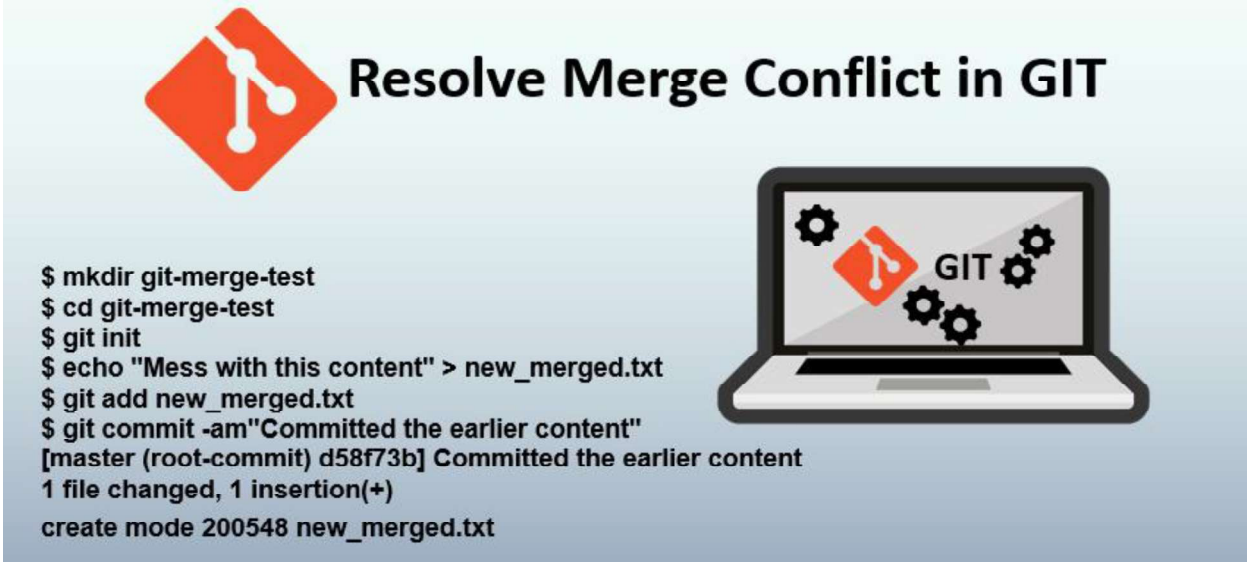
- Git fetch command downloads commits, files, and refs from a remote repository into your local repo Git pull is the more aggressive alternative; it will download the remote content for the active local branch and immediately execute git merge to create a merge commit for the new remote content
- When you use pull, Git tries to automatically do your work for you.
- pull automatically merges the commits without letting you review them first



Git conflicts

- A conflict arises when two separate branches have made edits to the same line in a file, or when a file has been deleted in one branch but edited in the other.
- Conflicts will most likely happen when working in a team environment.

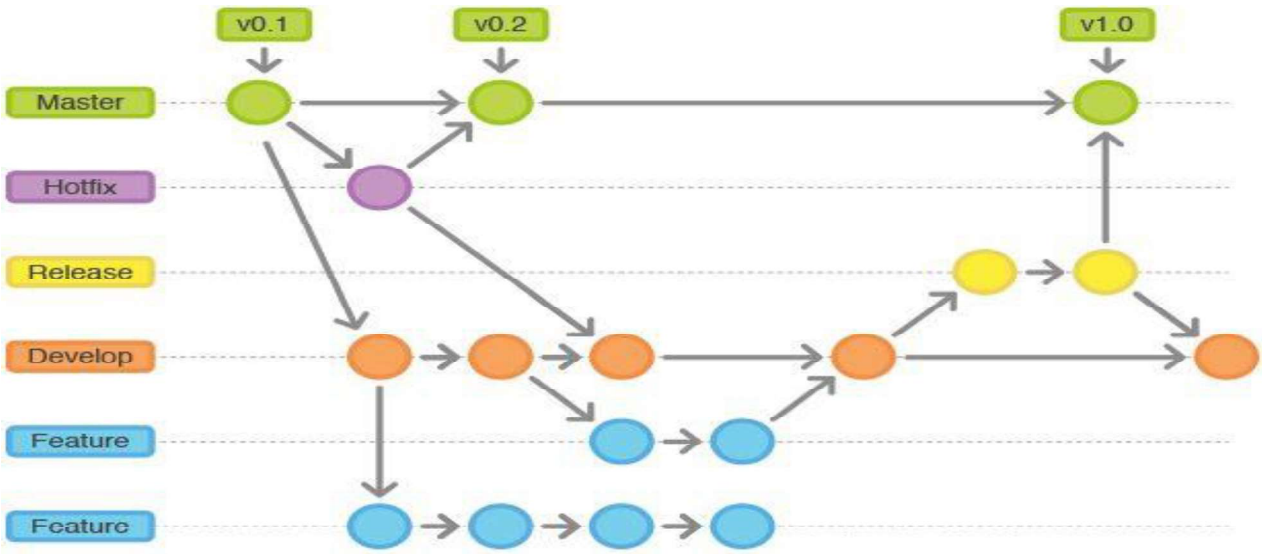
- There are many tools to help resolve merge conflicts



Git BrAnching StrAtegy

A branching strategy ensures everyone on the team is following the same process for making changes to source control.

- A develop branch is created from master. ... Feature branches are created from develop. When a feature is complete it is merged into the develop branch. When the release branch is done it is merged into develop and master. If an issue in master is detected a hotfix branch is created from master

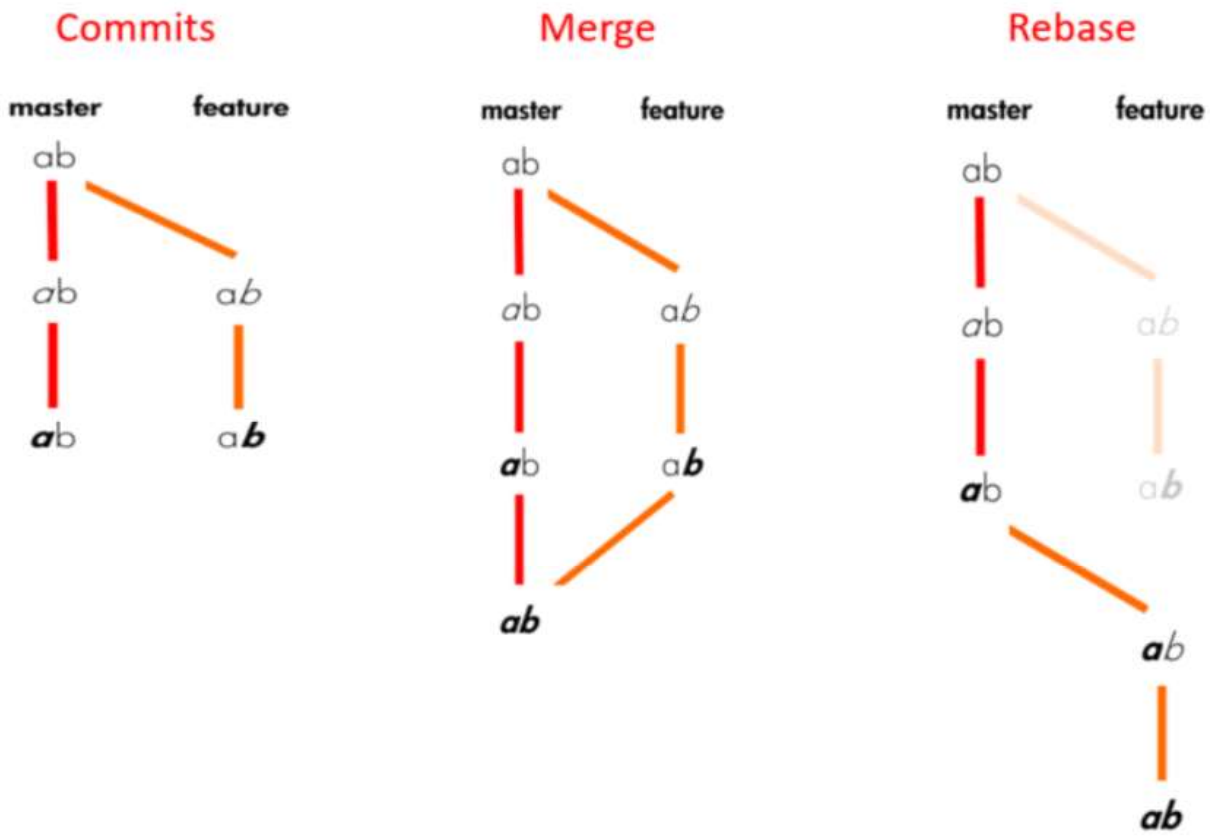


Git Vs GitHub

Git	GitHub
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the Git repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features

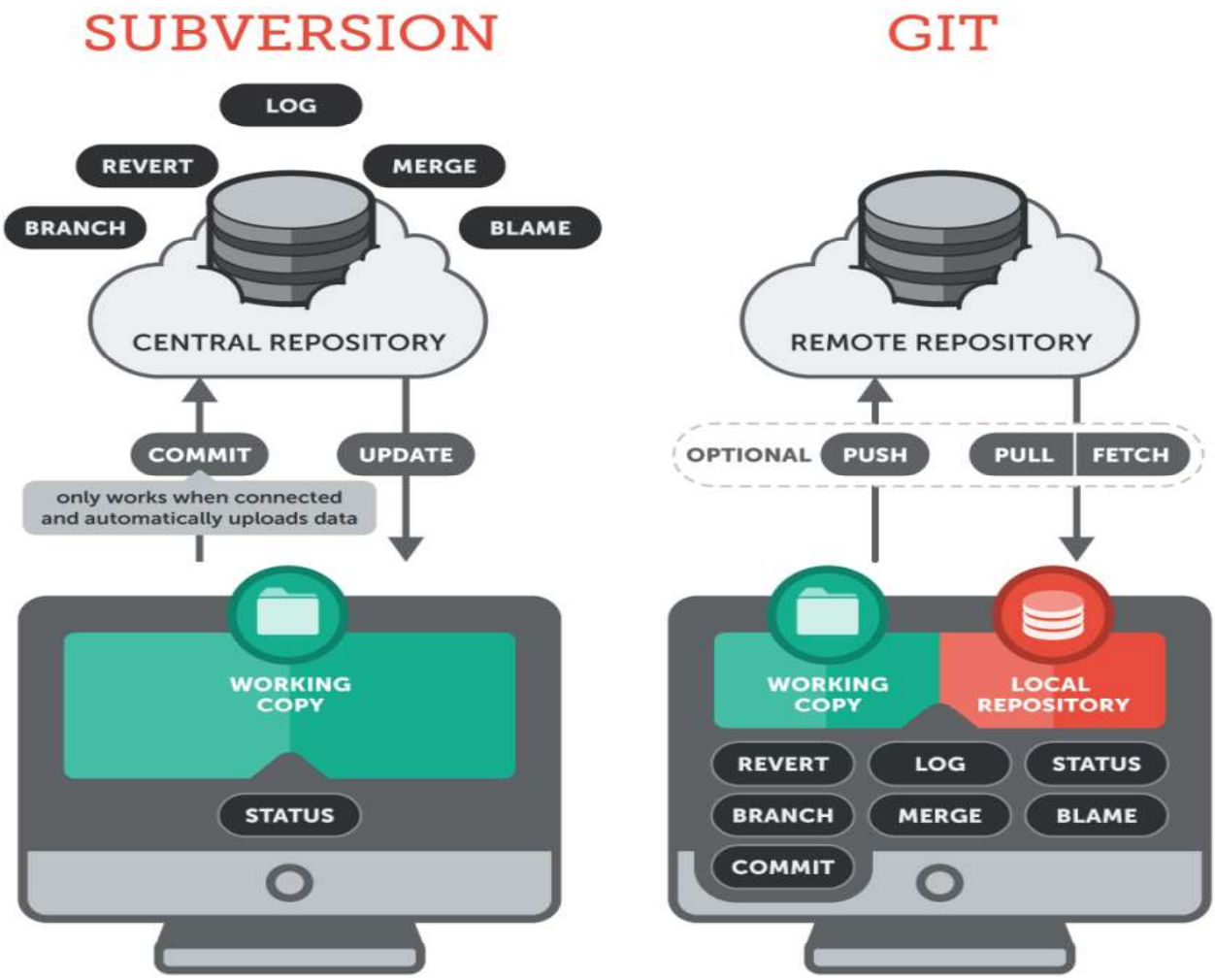
Merge vs RebAse

- Git rebase and merge both integrate changes from one branch into another. Where they d i f fe r is h ow i t's d o n e . Git rebase moves a feature branch into a master.
- Git merge adds a new commit, preserving the history.



Git Vs SVN

Git	vs	SVN
Used by ~90% of professional developers		Used by ~10% of professional developers
Distributed Version Control		Centralized Version Control
Can work locally, offline		Must be connected to commit
Each user has a copy of the full repository		Each user only has a copy of the trunk
Easy to fork, branch, and merge		Branching and merging is time-consuming

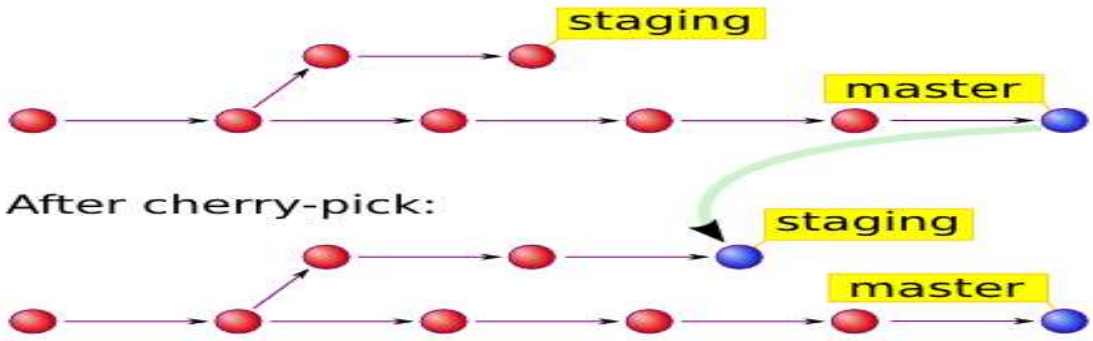


Git & Svn commAnds

Operations	Git command	Svn command
Initial checkout from existing repo	git clone <url> git checkout <origin/branch>	svn checkout <url>
Update locally from repo	git fetch git pull (does a fetch and merge)	svn update
Diff locally changed file	git diff <filename>	svn diff <filename>
Revert locally changed file	git checkout <filename>	svn revert <filename>
Revert all local changes	git reset --hard HEAD	svn revert . R
Commit changes to repo	git commit -m "message" <filename> git push	svn -ci m "message" <filename>

Git Cherry- pick

- Git cherry-pick is a powerful command that enables arbitrary Git commits to be picked by reference and appended to the current working HEAD.
- Cherry picking is the act of picking a commit from a branch and applying it to another.
- You can switch to the correct branch and cherry- pick the commit to where it should belong



Pull request

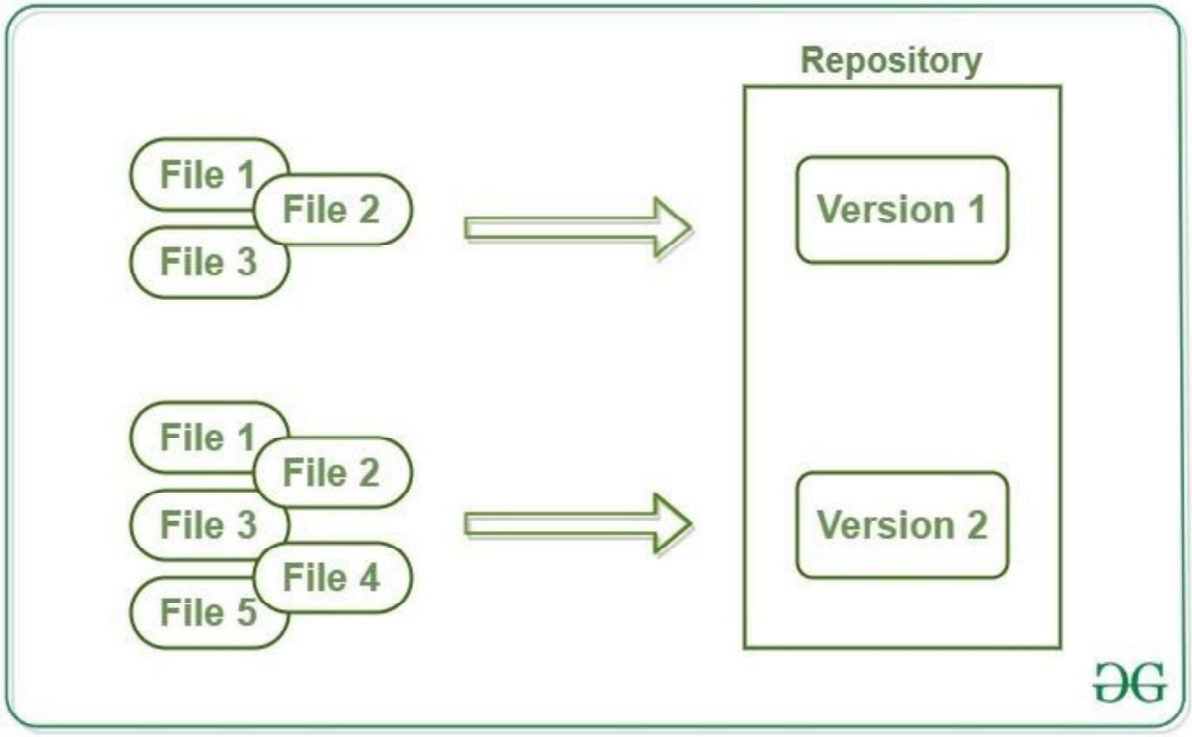
- Pull requests are a mechanism for a developer to notify team members that they have completed a feature. Once their feature branch is ready, the developer files a pull request via their Bitbucket account

Development without pull request

Development with pull request

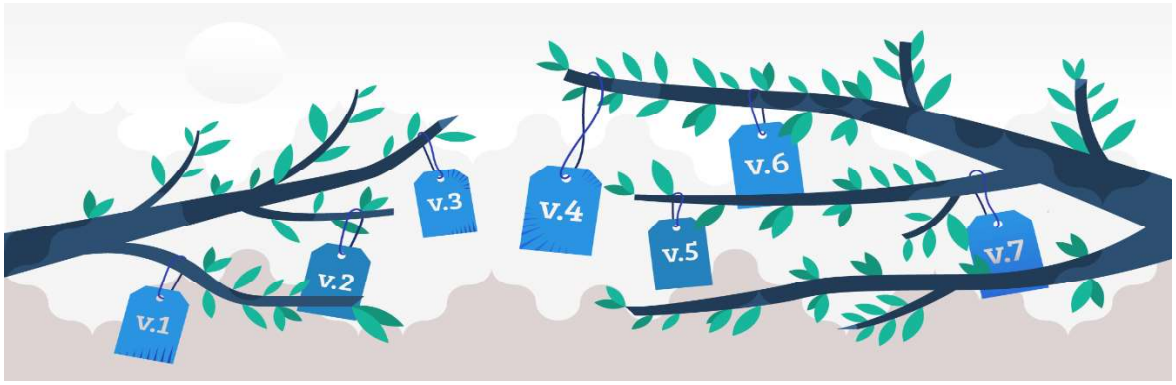
What is a GIT Repository

- Repositories in GIT contain a collection of files of various different versions of a Project.
- These files are imported from the repository into the local server of the user for further updations and modifications in the content of the file.
- A VCS or the Version Control System is used to create these versions and store them in a specific place termed as a repository



What is a tag in git

- Tags are ref's that point to specific points in Git history. Tagging is generally used to capture a point in history that is used for a marked version release (i.e. v1. 0.1).
- A tag is like a branch that doesn't change. Unlike branches, tags, after being created, have no further history of commits



Types of Version Control Systems

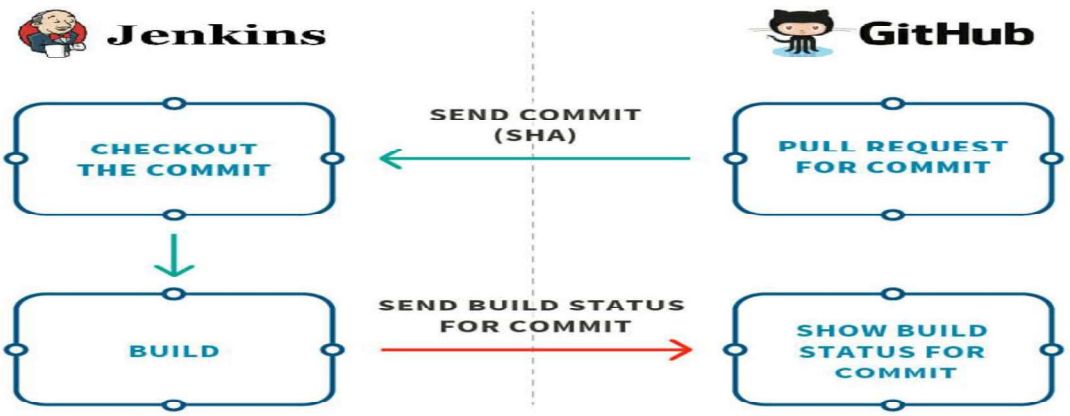
- Local Version Control Systems
- Centralized Version Control Systems
- Distributed Version Control Systems

How can you create a repository in Git

- To create a repository, create a directory for the project if it does not exist, then run the command "**git init**". By running this command .git directory will be created in the project directory.

Jenkins setup with Git

- In your browser, go to your Jenkins URL
- Click on Manage Jenkins
- Click on Manage Plugins
- Go to Available tab and search for GitHub integration plugin
- Check the checkbox on the left and click on download and install after restart



Git Commands:

Configuring user information used across all local repositories

1	Git config --global user.name "Firstname Lastname"	Configuring the name
2	Git config --global user.name "Valid-email"	Configuring the email

GIT BASICS:

Configuring user information, initializing and cloning repositories

1	git init	Initialize an existing directory as a Git repository
2	git clone [url]	Retrieve an entire repository from a hosted location via URL

STAGE & SNAPSHOT:

Working with snapshots and the Git staging area

1	git status	Show the modified files in the working directory, staged for your next commit
2	git add [file]	Add a file as it looks now to your next commit (stage)
3	git reset [file]	Unstage a file while retaining the changes in working directory
4	git diff	Diff of what is changed but not staged
5	git diff --staged	Diff of what is staged but not yet committed
6	git commit -m "[descriptive message]"	Commit your staged content as a new commit snapshot

BRANCH & MERGE :

Isolating work in branches, changing context, and integrating changes

1	git branch	List your branches. a * will appear next to the currently active branch
2	git branch [branch-name]	Create a new branch at the current commit
3	git checkout	Switch to another branch and check it out into your working directory
4	git merge [branch]	Merge the specified branch's history into the current one
5	git log	Show all commits in the current branch's history

INSPECT & COMPARE

Examining logs, diffs and object information

1	git log	Show the commit history for the currently active

		branch
2	git log branchB..branchA	Show the commits on branchA that are not on branchB
3	git log --follow [file]	Show the commits that changed file, even across renames
4	git show [SHA]	Show any object in Git in human-readable format
5	git diff branchB...branchA	Show the diff of what is in branchA that is not in branchB

SHARE & UPDATE

Retrieving updates from another repository and updating local repos

1	git remote add [alias] [url]	Add a git URL as an alias
2	git fetch [alias]	Fetch down all the branches from that Git remote
3	git merge [alias]/[branch]	Merge a remote branch into your current branch to bring it up to date
4	git push [alias] [branch]	Transmit local branch commits to the remote repository branch
5	git pull	Fetch and merge any commits from the tracking remote branch

TRACKING PATH CHANGES

Versioning file removes and path changes

1	git rm [file]	Delete the file from project and stage the removal for commit
2	git mv [existing-path] [new-path]	Change an existing file path and stage the move
3	git log --stat -M	Show all commit logs with indication of any paths that moved

REWRITE HISTORY

Rewriting branches, updating commits and clearing history

1	git rebase [branch]	Apply any commits of current branch ahead of specified one
2	git reset --hard [commit]	Clear staging area, rewrite working tree from specific commit

TEMPORARY COMMITS

Temporarily store modified, tracked files in order to change branches

1	git stash	Save modified and staged changes
2	git stash list	List stack-order of stashed file changes
3	git stash pop	Write working from top of stash stack
4	git stash drop	Discard the changes from top of stash stack

Conclusion

Git provides a way of keeping track of past versions of software and papers, making collaboration between various authors easy, and provides backup for your software. It has proven very useful to the open-source community and in academia as well.

It's not the only system with this kind of power, nor does it always employ the best interface to its concepts. What it does have, however, is a solid base to work from. In the future, I imagine many new methods will be devised to take advantage of the flexibilities Git allows. Most other systems have led me to believe they've reached their conceptual plateau — that all else from now will be only a slow refinement of what I've seen before. Git gives me the opposite impression, however. I feel we've only begun to see the potential its deceptively simple design promises

RT – Technologies

+91-9019995361