GIT Course

By. Rohit K Singh

Version Control

- Version control, also known as source control, is the practice of tracking and managing changes to software code
- Version control software keeps track of every modification to the code in a special kind of database
- Version control tracks every developer change and helps prevent concurrent work from conflicting

Benefits of the version control system:

- ❑ Enhances the project development speed by providing efficient collaboration
- ❑ Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change
- ❑ Employees or contributors of the project can contribute from anywhere irrespective of the different geographical locations through this VCS
- ❑ For each different contributor to the project, a different working copy is maintained and not merged to the main file unless the working copy is validated. The 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, and Why changes have been made

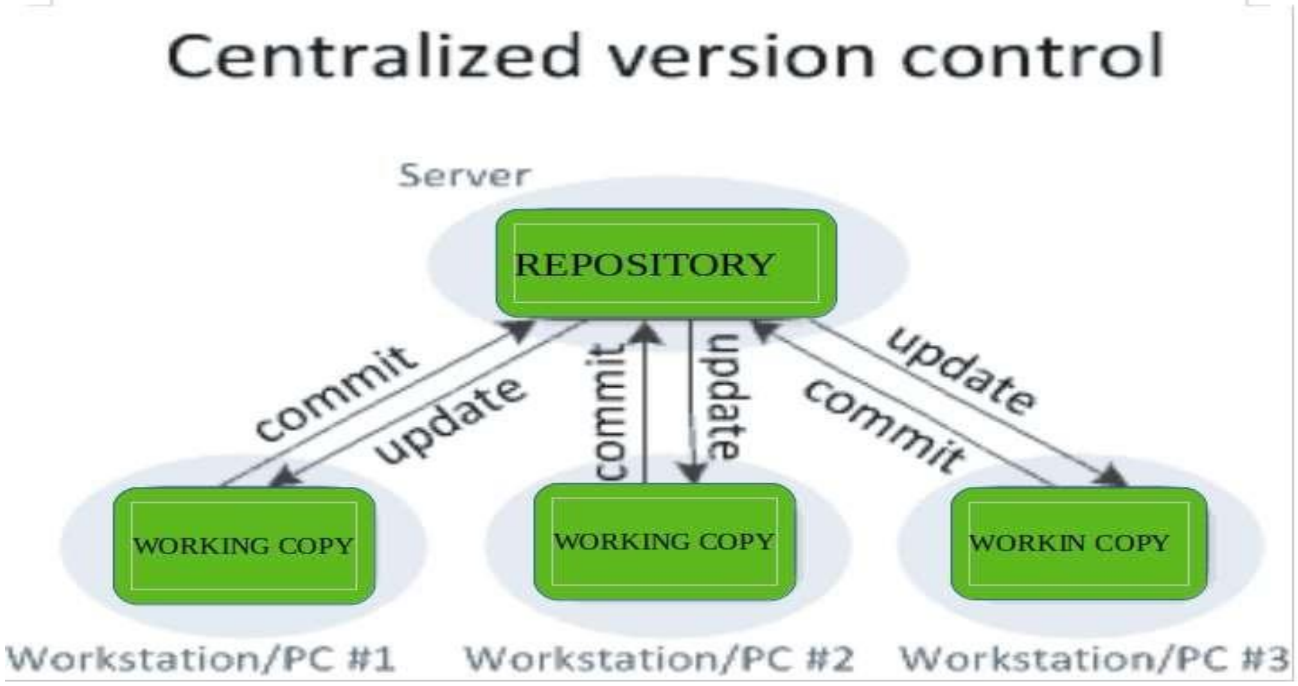
Types of Version Control Systems

Centralized Version Control Systems

Centralized version control systems contain just one repository globally and every user needs to commit to reflecting one's chances in the repository. It is possible for others to see your changes by updating.

Two things are required to make your changes visible to others which are:

- ❑ You commit
- ❑ They update



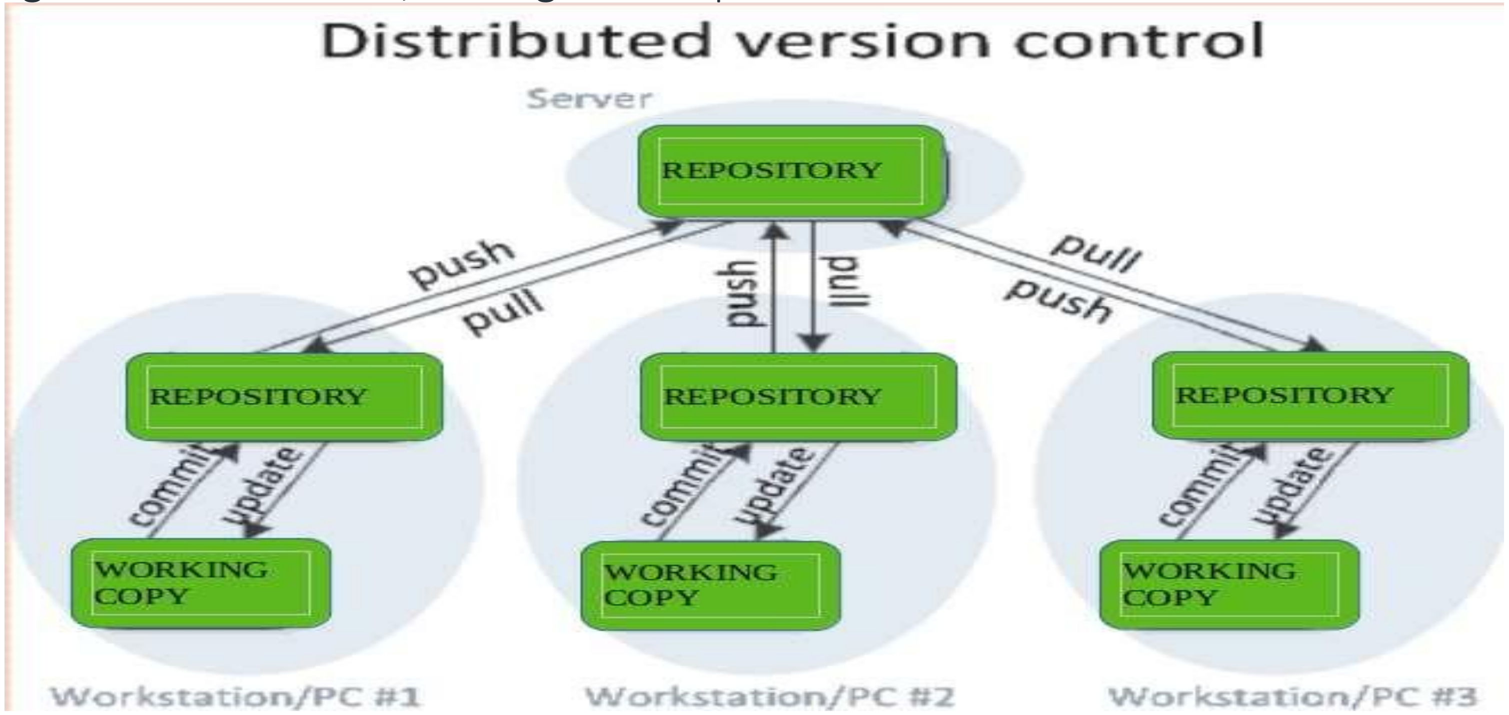
Types of Version Control Systems

Distributed Version Control Systems:

- Distributed version control systems contain multiple repositories
- Each user has their repository and working copy, users need to pull to fetch the latest changes whereas developers need to push their changes to update the remote repository

❖ To make your changes visible to others, 4 things are required:

- ❑ You commit
- ❑ You push
- ❑ They pull
- ❑ They update



Benefits of using GIT over SVN/TFS

- Git utilizes multiple repositories: a central repository and a series of local repositories. Local repositories are exact copies of the central repository complete with the entire history of changes, Unlike SVN
- The Git workflow is similar to SVN, but with an extra step: to create a new feature, you take an exact copy of the central repository to create your local repository on your local machine
- Many developers prefer GIT for the following reason:
  - Its faster to commit
  - No more single point of failure
  - If the centralized server is down, the developers can still work using a local copy
  - It's available offline
- TFS works on the concept of By Value whereas GIT works on the concept of By Reference
  - This means in TFS, the total number of branches = total number of physical folders
  - GIT, the total number of branches = Branches are logical and only the reference or pointer moves ahead or back with each branch

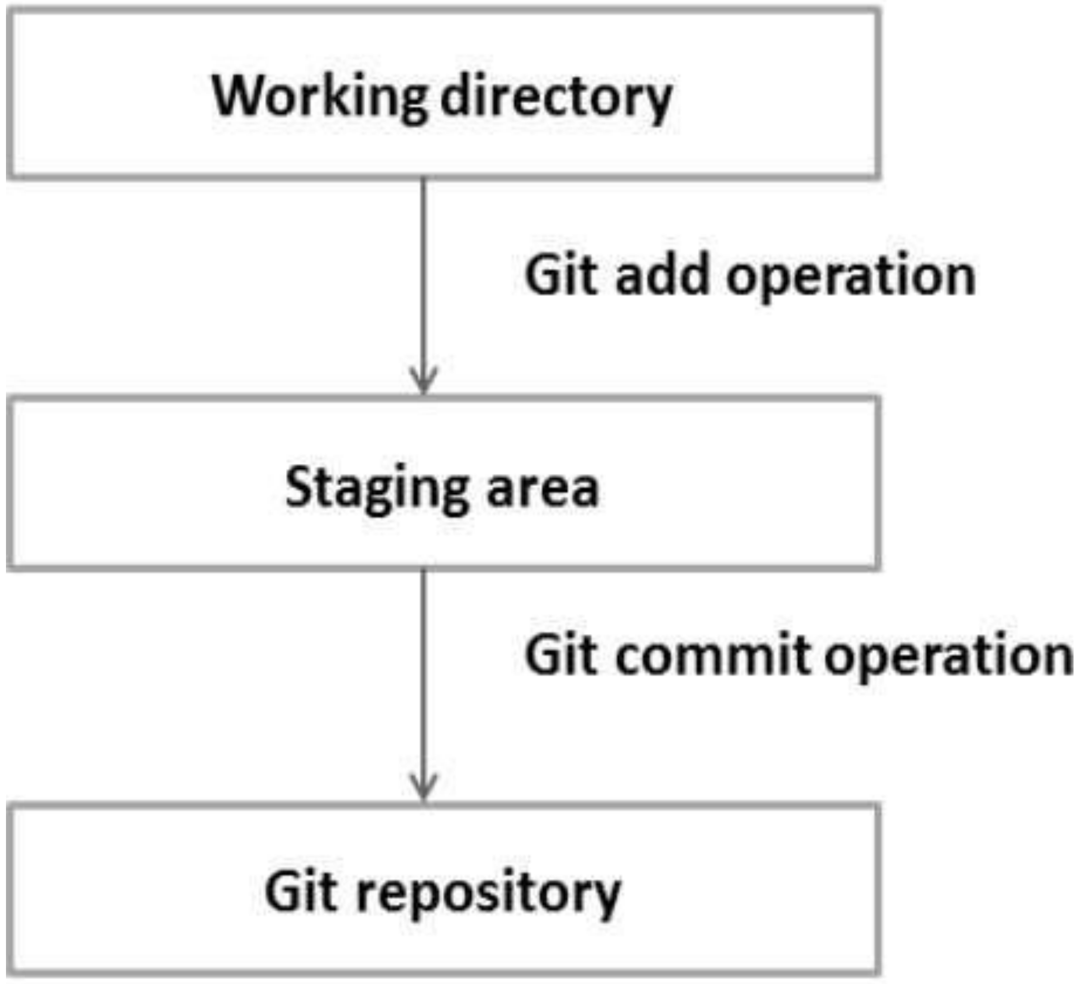
GIT Introduction

Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

Distributed VCs:

- Distributed version control (also known as distributed revision control) is a form of version control in which the complete codebase, including its full history, is mirrored on every developer's computer.
- It enables automatic management branching and merging speeds up most operations (except pushing and pulling), improves the ability to work offline, and does not rely on a single location for backups.

GIT Stages


Working directory
Git add operation
Staging area
Git commit operation
Git repository

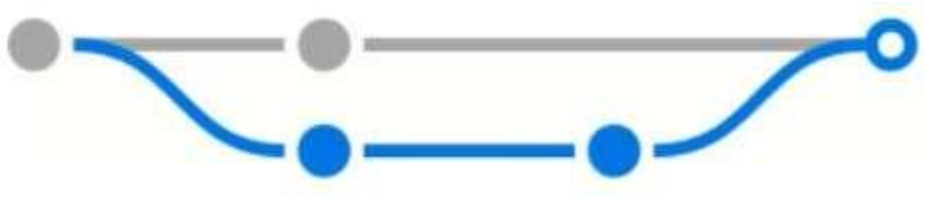
GIT commands and Demo

git init
git config
git add
git commit
git status
gitk (Graphical representation)
git log
git show
git branch
git switch
git clone
git checkout
git merge/rebase
git pull/push/fetch
git remote
git cherry-pick
git reset
git stash
git tag
git diff

Merge Types:

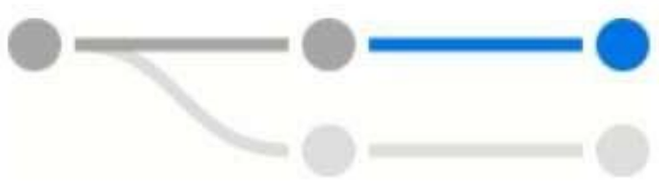
❑ **Merge (no fast-forward):**

- It is a default integration strategy in Azure Repos, GitHub, and most other Git providers
- All individual commits are preserved as-is, and a new merge commit is created
- This strategy is helpful because it illustrates exactly how a developer (or developers) worked on a pull request, including each individual, commit along the way



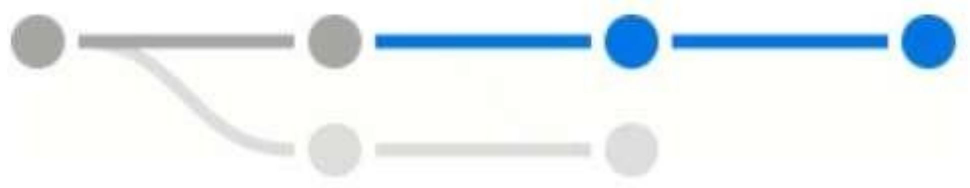
❑ **Squash Commit:**

- In Squash each pull request becomes a single commit in master, and there are no merges, just a simple, straight, linear history
- Individual commits are lost, which is best for teams that use "fix-up" commits or do not carefully craft individual commits for review before pushing them



❑ **Rebase:**

- Rebase will take each individual commit in the pull request and cherry-pick them onto the destination branch
- When this strategy is used, history is straight and linear, like it is with the "squash" option, but each individual commit is retained



❑ **Semi-linear merge:**

- It's a mix of rebasing and a merge
- First, the commits in the pull request are rebased on top of the master branch. Then those rebased pull requests are merged into the destination branch
- This strategy is best used for retaining individual commits, and for viewing how the work evolved, but instead of just being rebased, a "merge bubble" is shown so that you can immediately see the work in each pull request