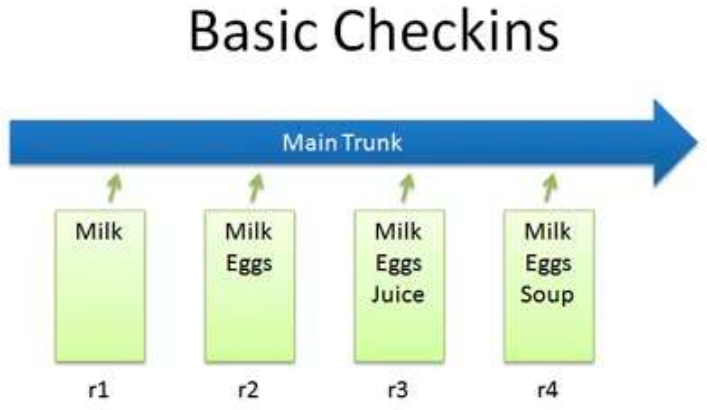
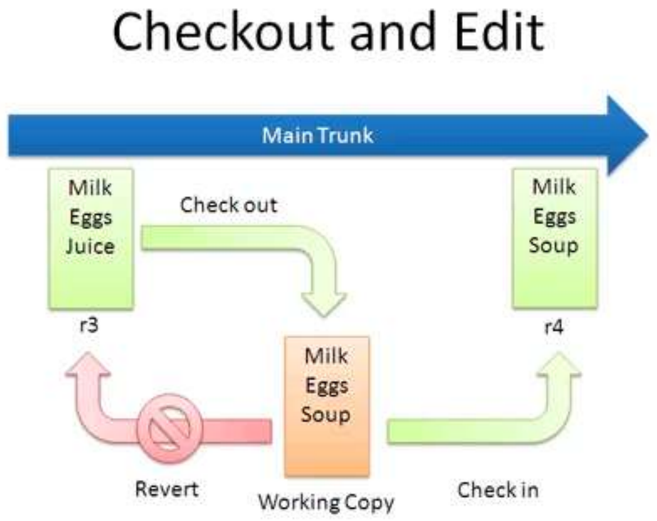
Git and gitHub

What is Version Control?

Version control provides several significant benefits:

1. **Collaboration:** It enables multiple people to work on the same project without stepping on each other's toes. Each developer can work on their own part of the project separately and then merge their changes with the rest of the project.
2. **History:** It provides a detailed history of all changes to a project, including who made the changes, when the changes were made, and why the changes were necessary. This not only provides context but also makes it easy to find and fix bugs.
3. **Backup:** Because every change is tracked, version control provides a robust backup system. If something goes wrong, you can always roll back to a previous version.
4. **Branching and Merging:** Version control systems allow for branching, where developers can create a separate line of development for bug fixes or new features. Once completed, these branches can then be merged back into the main project.



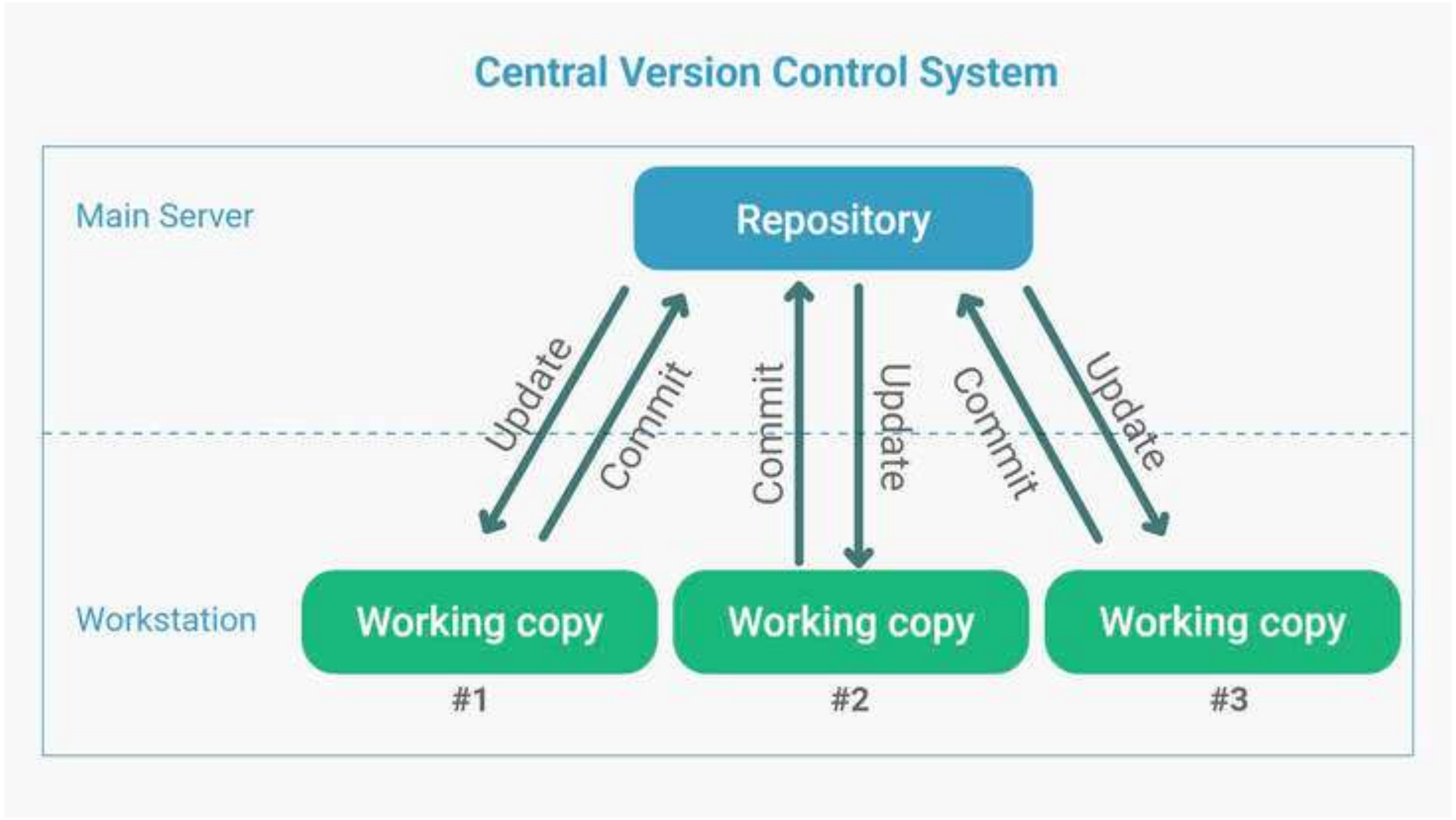


Types of Version Control?

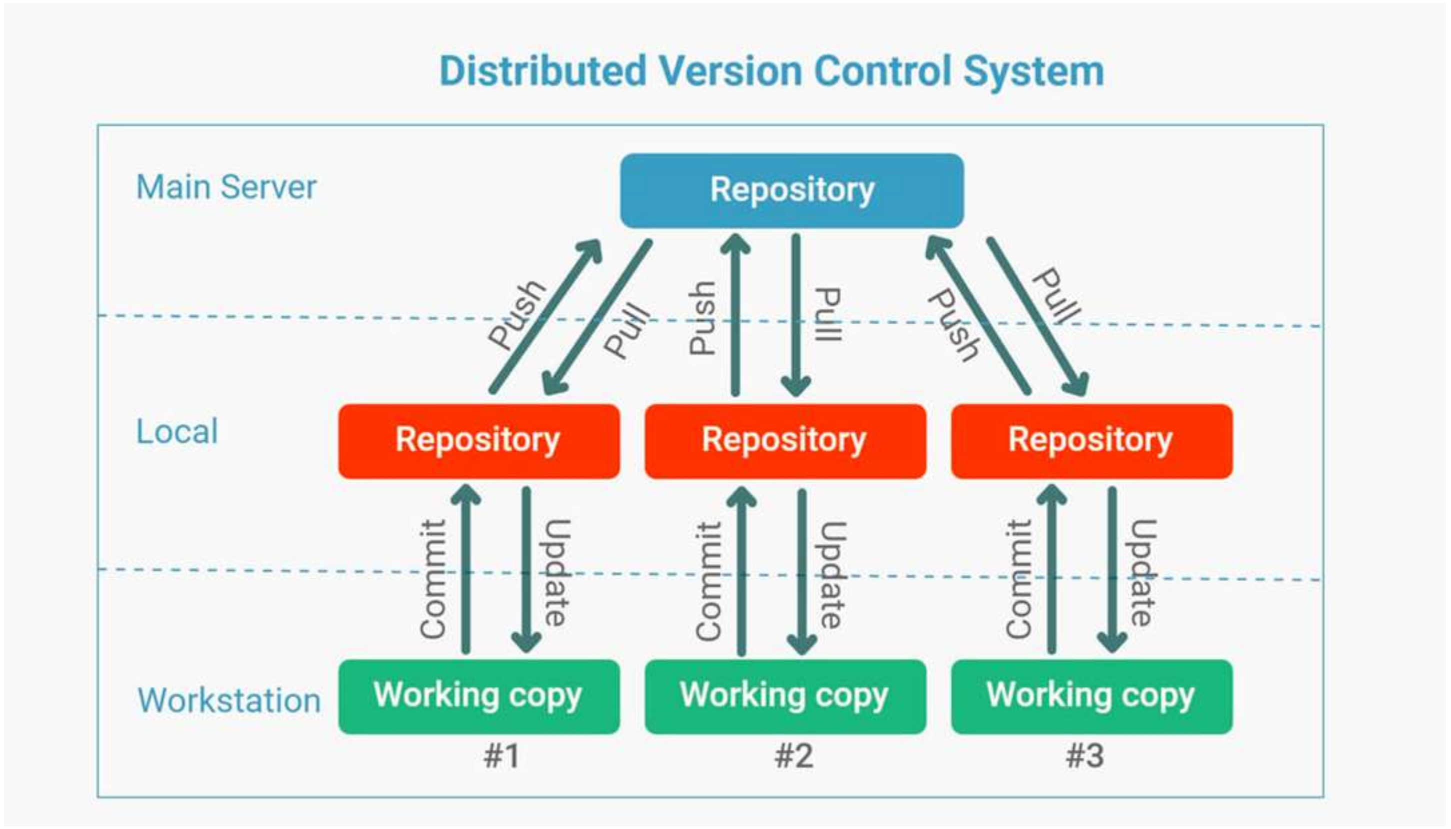
Version Control Systems:

- Centralized
- Distributed.

Centralized (CVCS)



Distributed (DVCS)



What is Git?

Git: A Brief Overview

Efficient and Fast

- Optimized performance irrespective of project size.
- Local repositories limit server communication, enhancing speed.

Flexibility

- Allows multiple developers on a single codebase.
- Developers can work on isolated branches for conflict-free development.

Data Integrity

- Checksums for every file and commit ensure data security.
- Complete retrievability of any lost information.

Scalability

- Can seamlessly handle small to large projects with consistent efficiency.

Created by Linus Torvalds, the mastermind behind Linux.
Free and Open-Source Distributed Version Control System.

How git works

Working Directory
Place where all the work is done: create, edit, delete, and organize files.

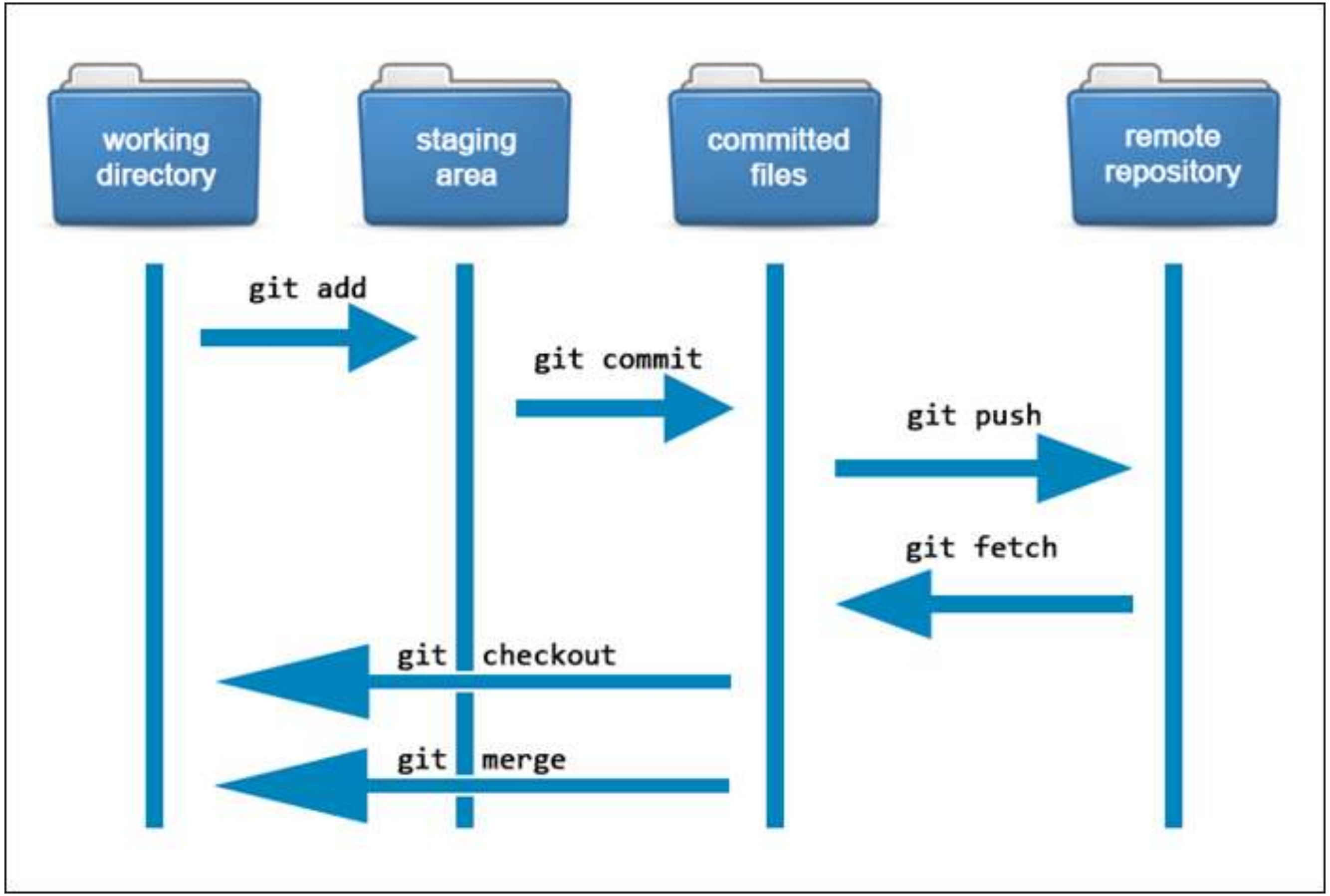
Staging Area
Preparatory space where changes are lined up. Git takes a snapshot of the changes for the commit.

Git Directory (Local Repository)
Contains all the project metadata and versions of every file.
Upon commit, the current project snapshot is saved here.

Git's Workflow Cycle

1. Modify files in the Working Directory.
2. Stage the modifications, preparing for a commit in the Staging Area.
3. Commit the changes, storing a project snapshot in the Git Directory.

How git works



git commands

Git commands:

'git init': This is the command you'll use to start a new repository. If you have a project directory that you'd like to turn into a Git repository, you navigate to that directory in your terminal and type 'git init'. This creates a new subdirectory named .git that contains all of your necessary repository files — a Git repository skeleton.

'**git add'**: Once you've made changes in your working directory, the 'git add' command allows you to take those changes and stage them for a commit. It's like saying, "Okay, Git, I want you to keep track of these changes." You can add individual files, or, if you want to add everything in one go, you can use 'git add .'

'git commit': After staging your changes with 'git add', you use 'git commit' to save those changes to your local repository. Think of it as a checkpoint in your development process. This command is typically used with a message flag (-m) to provide a description of what was changed, like so: 'git commit -m "Your message here"'

'git clone': If you want to get a copy of an existing Git repository — for example, a project you'd like to contribute to — the command you'll need is 'git clone'. If you're cloning a repository from GitHub, for instance, you can use the URL of that project with 'git clone'. This command creates a copy of the project on your local machine, complete with the entire history of the project.

'git add -A' or 'git add --all': These commands stage all changes in the entire working directory, including new, modified, and deleted files. They're useful when you've made multiple changes and want to commit all of them.

'git commit -am "Message"': This is a combination command that stages all changes to tracked files (those that Git has recorded in previous snapshots) and commits them in one step, with a message describing the commit.

'git status': This command shows the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git.

'git log': This command shows a list of all previous commits, including commit hash, author, date, and commit message.

'git checkout': This command is used to switch between different versions of a Git repository. For example, 'git checkout [branch-name]' switches to the specified branch.

'git pull': This command fetches changes from a remote repository and merges them into the current branch of your local repository.

'git push': This command pushes commits from your local repository up to a remote repository.

git

+

GitHub

Practice

Installing Git: Download and install Git on your system and configure your username and email.

Initializing a Repository: Create a new directory, open it, and use the git init command to start a new Git repository.

Adding & Committing: Make changes to a file in your working directory and use the git add command to stage the changes. Use the git commit command to commit the changes to the repository.

Checking Status & Logging: Use the git status command to check the status of your repository. Use the git log command to check the commit history.

Branching: Use the git branch command to create a new branch. Switch between branches using the git checkout command. Merge branches using the git merge command.

Stashing: Make changes to your working directory but don't commit them. Use the git stash command to temporarily save your changes without committing.

Cloning a Repository: Find a repository on GitHub and use the git clone command to create a copy of the repository on your local machine.

Forking a Repository on GitHub: Navigate to a GitHub repository and click the "Fork" button to create a copy of the repository in your own GitHub account.

Making a Pull Request: After making changes in your forked repository, navigate to the original repository and click the "New pull request" button. Compare the changes and submit the pull request.

Handling Merge Conflicts: *Intentionally create a merge conflict and learn how to resolve it.*

Reverting Commits & Resetting: Make a few commits and use the git revert command to undo a specific commit. Use the git reset command to unstage files.