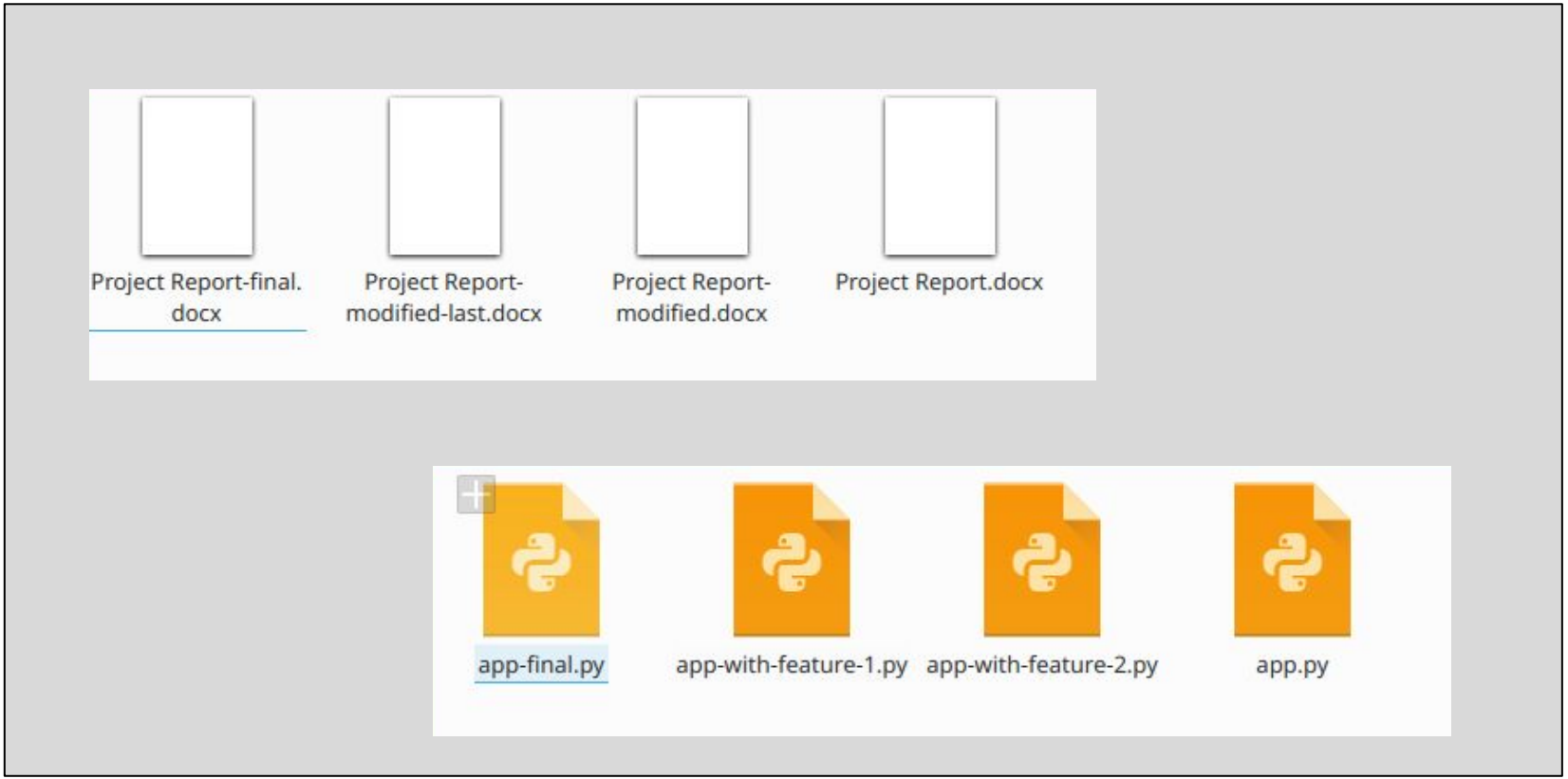
# Version Control System

Collaborative Development

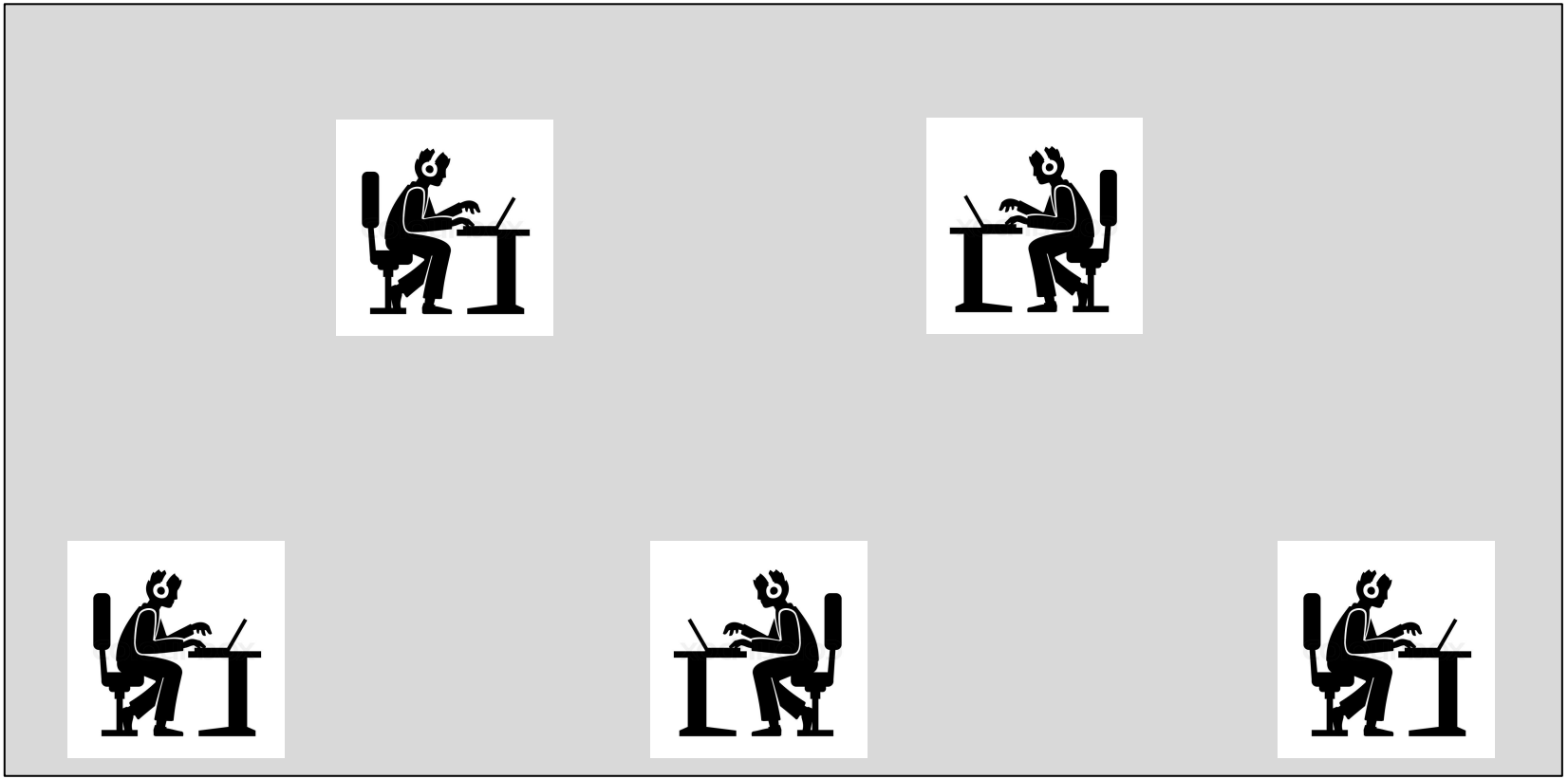
# Learning Objectives

- The Need for Version Control System
- Introduction to Version Control System
- Benefits
- Types of Version Control System
  - Centralized VCS
  - Distributed VCS
- Git
- GitHub

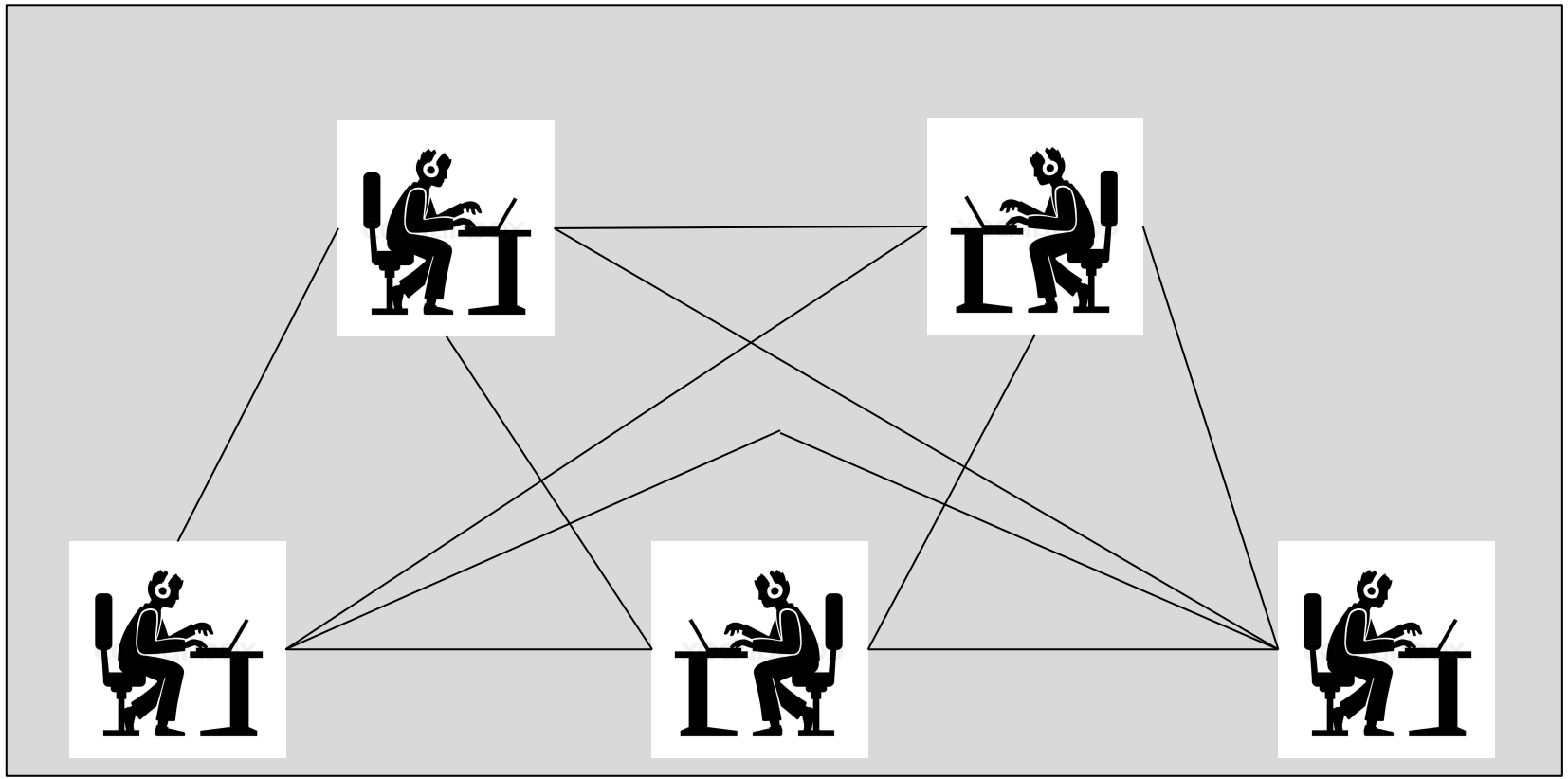
# The Need for VCS




**Tracking Multiple Versions of a Source Code, File, Document**

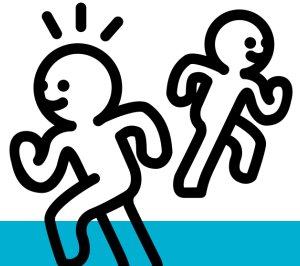


**Collaboration among multiple programmers**



Collaboration among multiple programmers

Multiplying the issue with a large number of collaborators over a large number of changes over multiple places!!



# Introduction to VCS

# 1. Version Control System

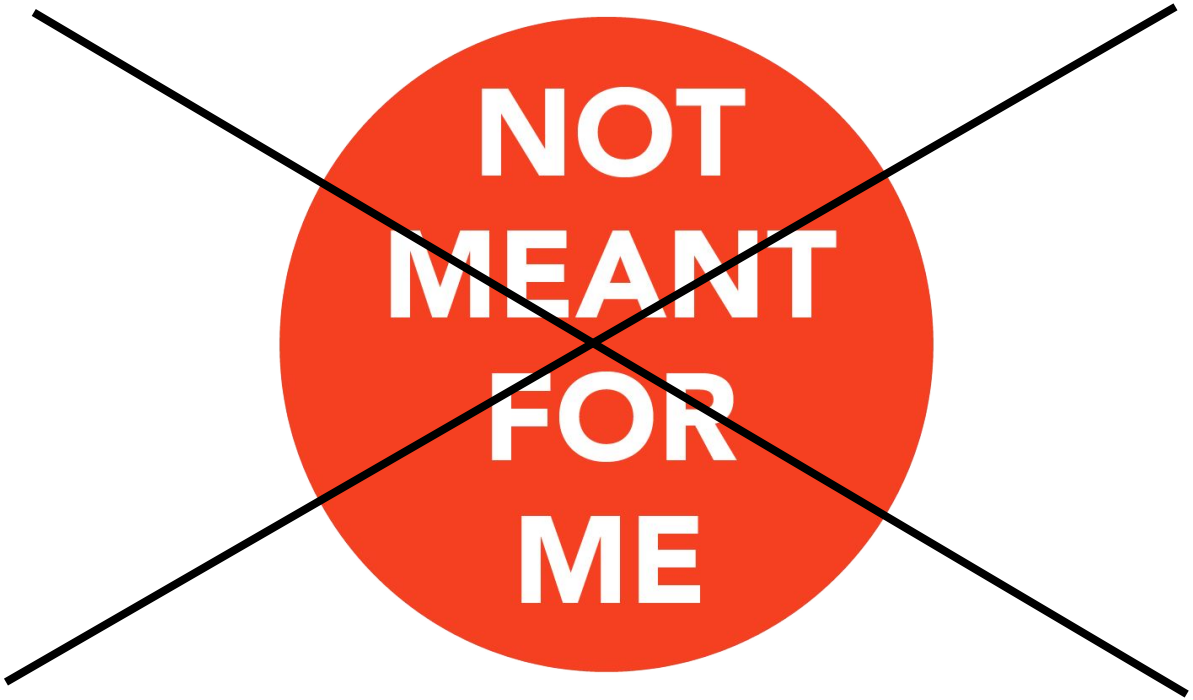
- Version control is a **system that records changes to a file or set of files over time** so that you can **recall specific versions later**.
- It allows you to
  - revert files back to a previous state,
  - revert the entire project back to a previous state,
  - review changes made over time,
  - see who last modified something that might be causing a problem,
  - who introduced an issue and when, and more

# 1. Version Control System

- It integrates work done simultaneously by different team members. In most cases, edits to different files or even the same file can be combined without losing any work.

  In cases where two people make conflicting changes to the same part of a file, then the version control system asks for manual intervention in reconciling the changes (there are called merge conflicts).

# JUST A SOLO DEVELOPER



# Benefits

1.Version Control System

**2.Benefits of VCS**

3.Types of Version Control System

# 2. Benefits of VCS

- Backup and Restore
- Synchronization
- Short-term undo
- Long-term undo
- Track Changes
- Track Ownership
- Sandboxing
- Branching and merging

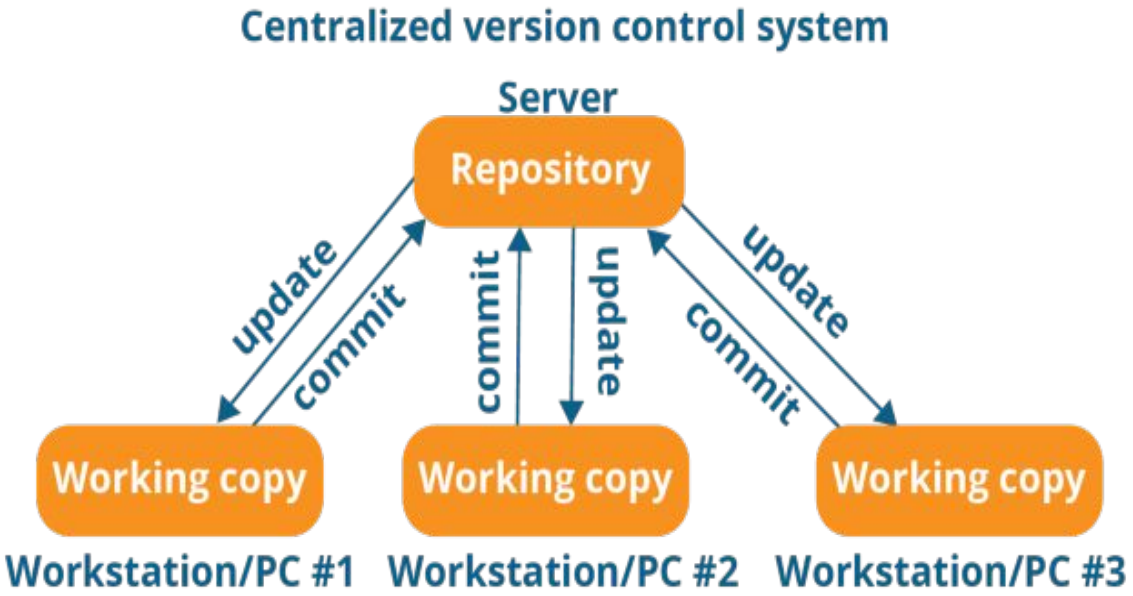
# Types of Version Control System

3.Types of Version Control System

**3.1 Centralized Version Control System**

3.2 Distributed Version Control System

# 3.1 Centralized Version Control System



3.Types of Version Control System

**3.1 Centralized Version Control System**

3.2 Distributed Version Control System

# 3.1 Centralized Version Control System

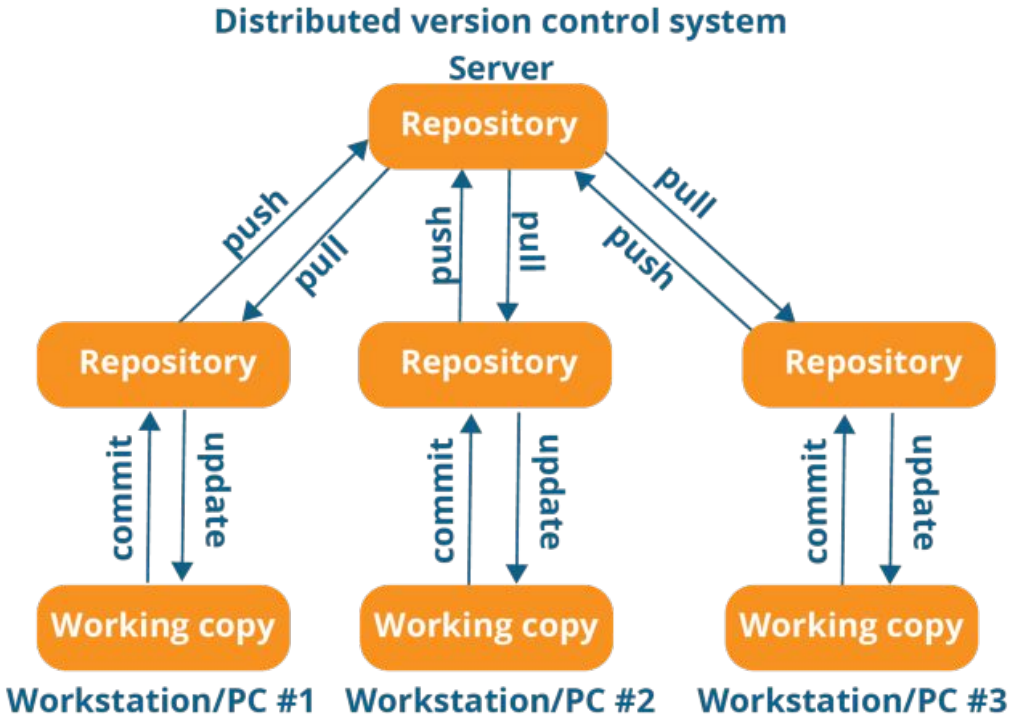
- based on the idea that there is a single “central” copy of your project somewhere (probably on a server), and programmers will “commit” their changes to this central copy.
- Some of the most common centralized version control systems you may have heard of or used are, Subversion (or SVN) and Perforce.
- Main benefits:
  - Centralized systems are typically easier to understand and use
  - You can grant access level control on directory level
  - performs better with binary files

3.Types of Version Control System

3.1 Centralized Version Control System

**3.2 Distributed Version Control System**

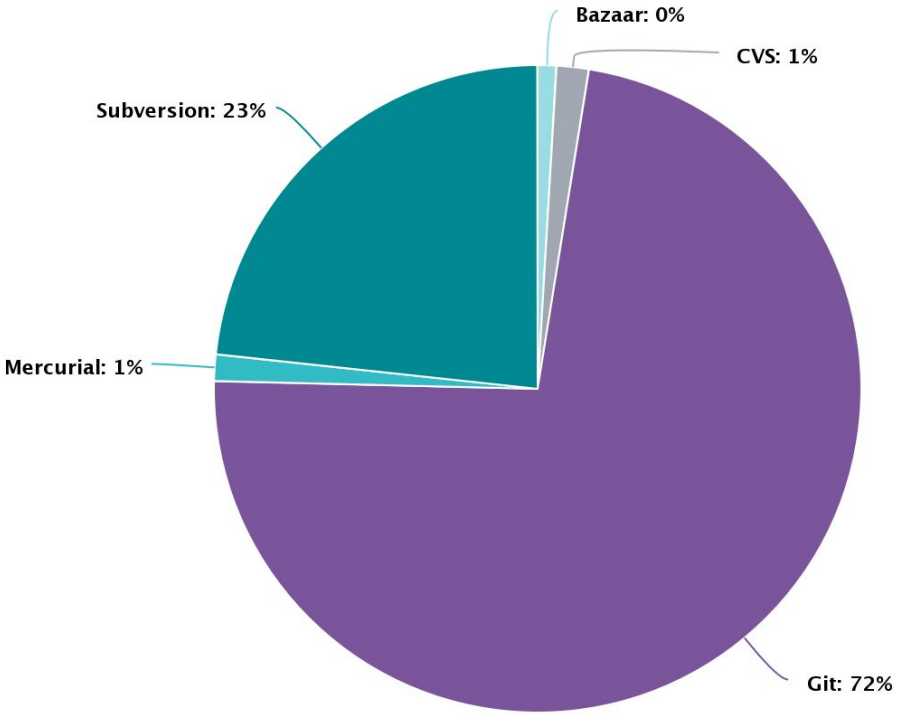
# 3.2 Distributed Version Control System



# 3.2 Distributed Version Control System

- Every developer "**clones**" a copy of a repository and has the full history of the project on their own hard drive. This copy (or "clone") has all of the metadata of the original.
- Git, Mercurial and Bazaar are popular examples.
- Main benefits:
  - Performance of distributed systems is better
  - Branching and merging is much easier
  - With a distributed system, you don't need to be connected to the network all the time (complete code repository is stored locally on PC)


git




# Usage Share of VCS

*https://www.openhub.net/repositories/compare

3.Types of Version Control System

**4.Git**

5.Essential Terminologies

# 4. Git

- Initially developed to handle the **development of Linux kernel**, ***the largest collaborative project in human history***
- A cross-platform, free and open source application
- A distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- Can work without Internet
- Lack of single point of failure

3.Types of Version Control System

**4.Git**

5.Essential Terminologies

# 4. Git

- Git is a **distributed** version control system.
  - You keep your files in a repository on your local machine.
  - You synchronize your repository with a repository on a server.
  - If you move from one machine to another, you can pick up the changes by synchronizing with the server.
  - If your partner uploads some changes to your files, you can pick those up by synchronizing with the server.
- Git is a distributed **version control** system.
  - Terminology: In git, a “version” is called a “commit.”
  - Git keeps track of the history of your commits, so you can go back and look at earlier versions, or just give up on the current version and go back some earlier version.

4.Git

**5.Essential Terminologies**

6.Git Workflow

# 5. Essential Terminologies

**Workspace**

(Working Directory)

**Index**

(Staging Area)

**Branch**

(An alternate code workspace)

**Local Repository**

(HEAD)

**Remote Repository**

(Origin)

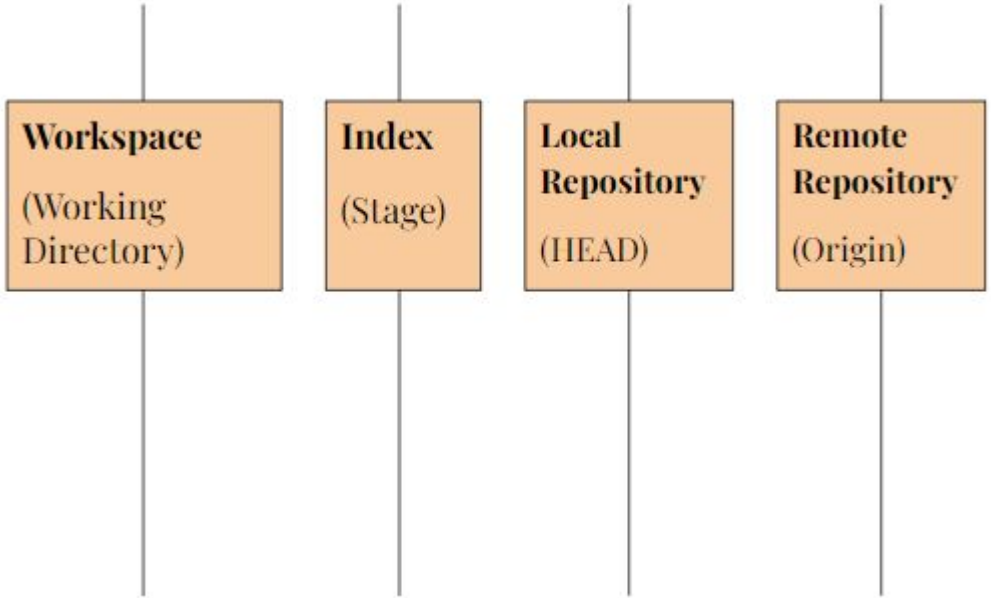
**6.Git Workflow**

6.1 Git Workflow-Step 1 - Add to Staging Area

6.2 Git Workflow-Step 2 - Commit to HEAD

6.3 Git Workflow-Step 3 -Push to Remote

# 6. Git Workflow



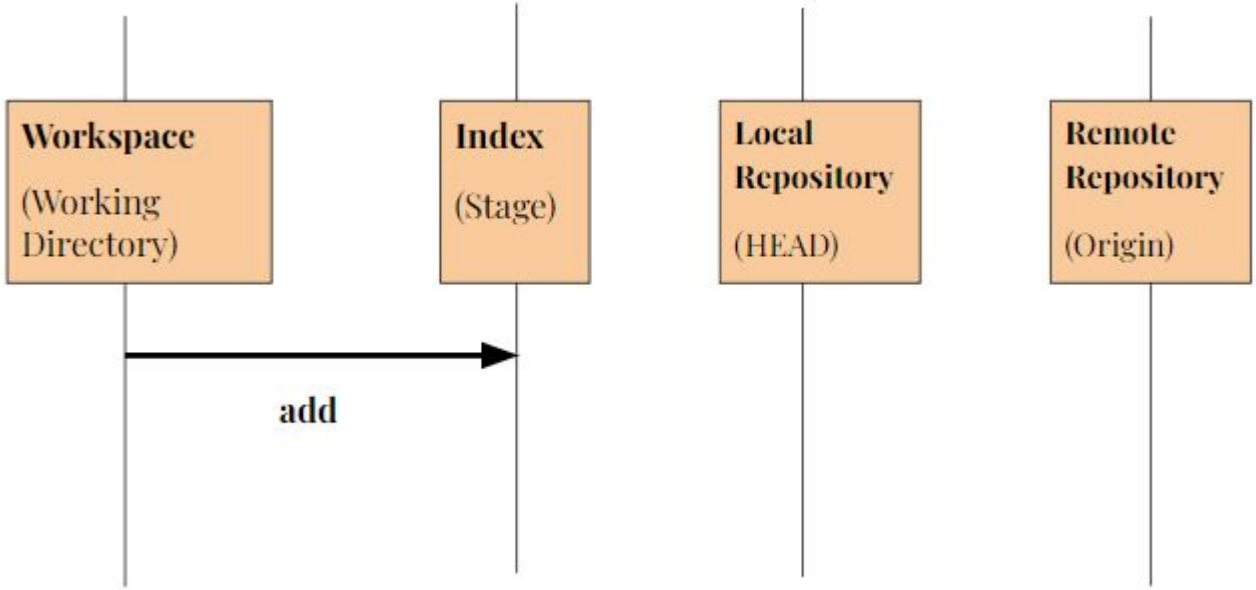
6.Git Workflow

**6.1 Git Workflow-Step 1 - Add to Staging Area**

6.2 Git Workflow-Step 2 - Commit to HEAD

6.3 Git Workflow-Step 3 -Push to Remote

# 6.1 Git Workflow - Step 1 - Add to Staging Area



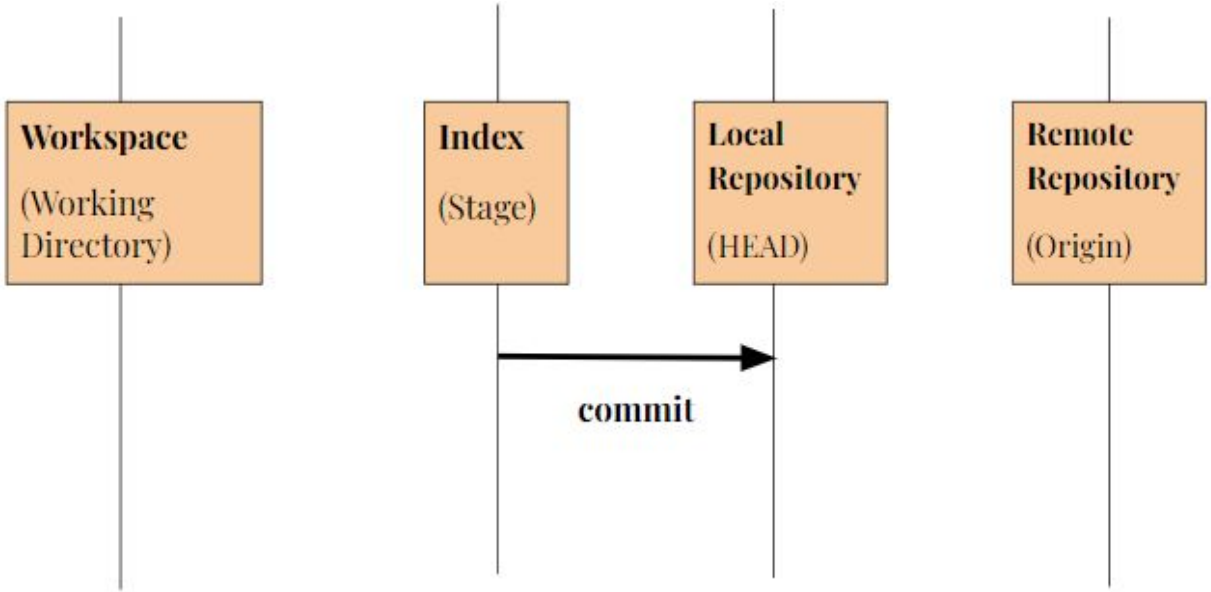
# 6.2 Git Workflow - Step 2 - Commit to HEAD

6.Git Workflow

6.1 Git Workflow-Step 1 - Add to Staging Area

**6.2 Git Workflow-Step 2 - Commit to HEAD**

6.3 Git Workflow-Step 3 -Push to Remote



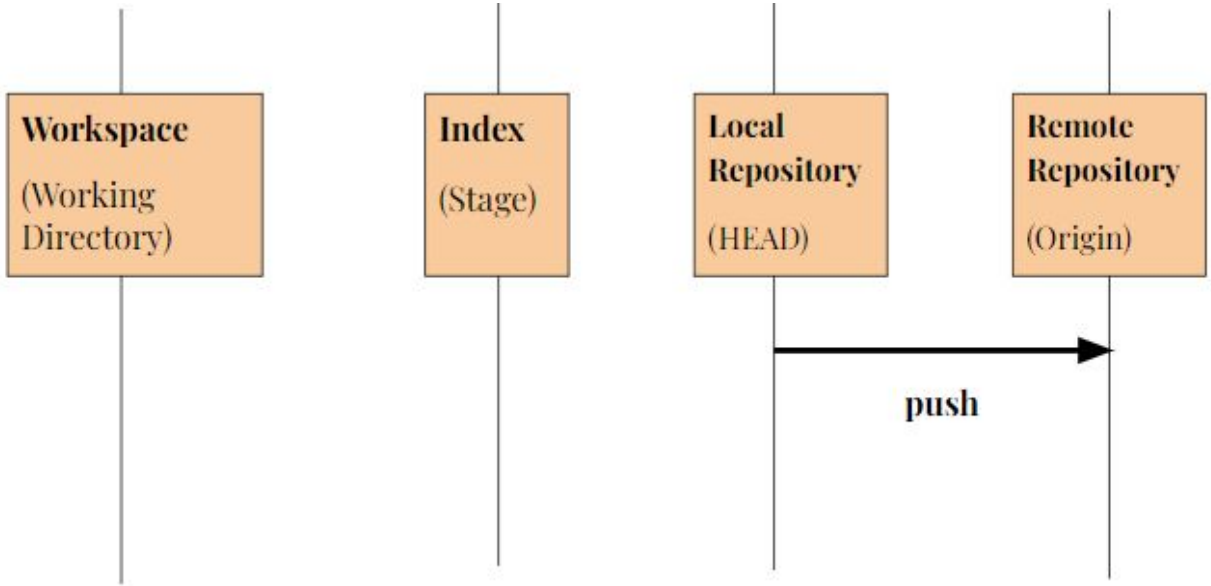
6.Git Workflow

6.1 Git Workflow-Step 1 - Add to Staging Area

6.2 Git Workflow-Step 2 - Commit to HEAD

**6.3 Git Workflow-Step 3 -Push to Remote**

# 6.3 Git Workflow - Step 3 - Push to Remote

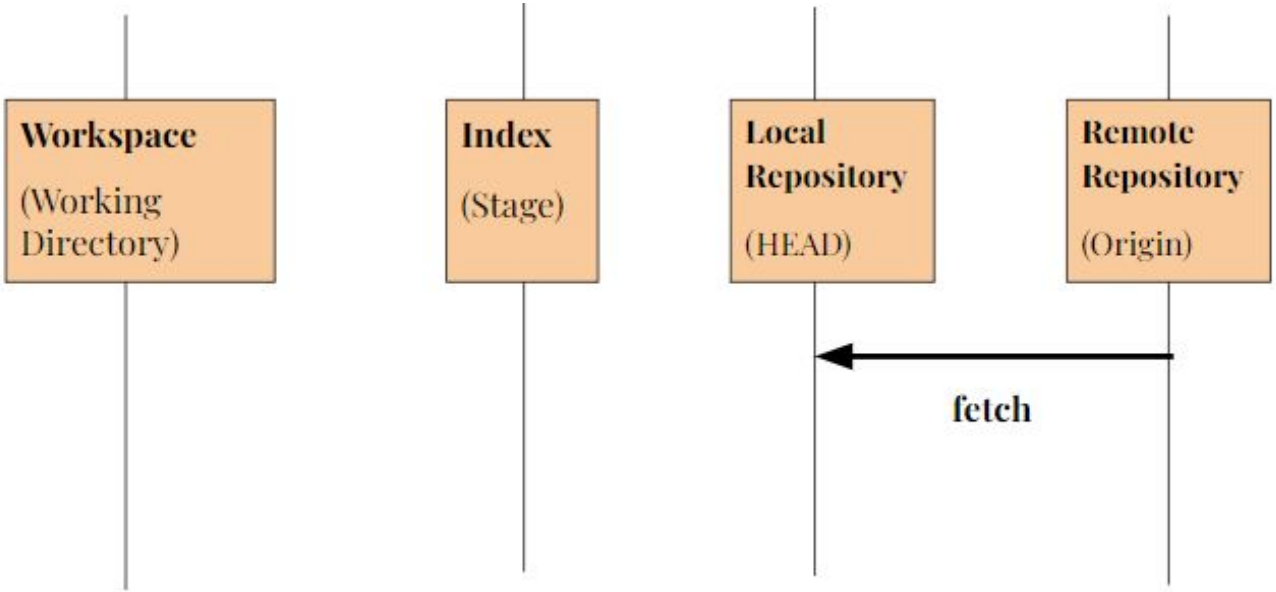


**7.1 Git Pull Workflow – Step 1-Fetch from Remote**

7.2 Git Pull Workflow – Step 2- Merge to Working Directory

7.3 Git Pull Workflow-Alternate Step-Pull from Remote

# 7.1 Git Pull Workflow - Step 1 - Fetch from Remote

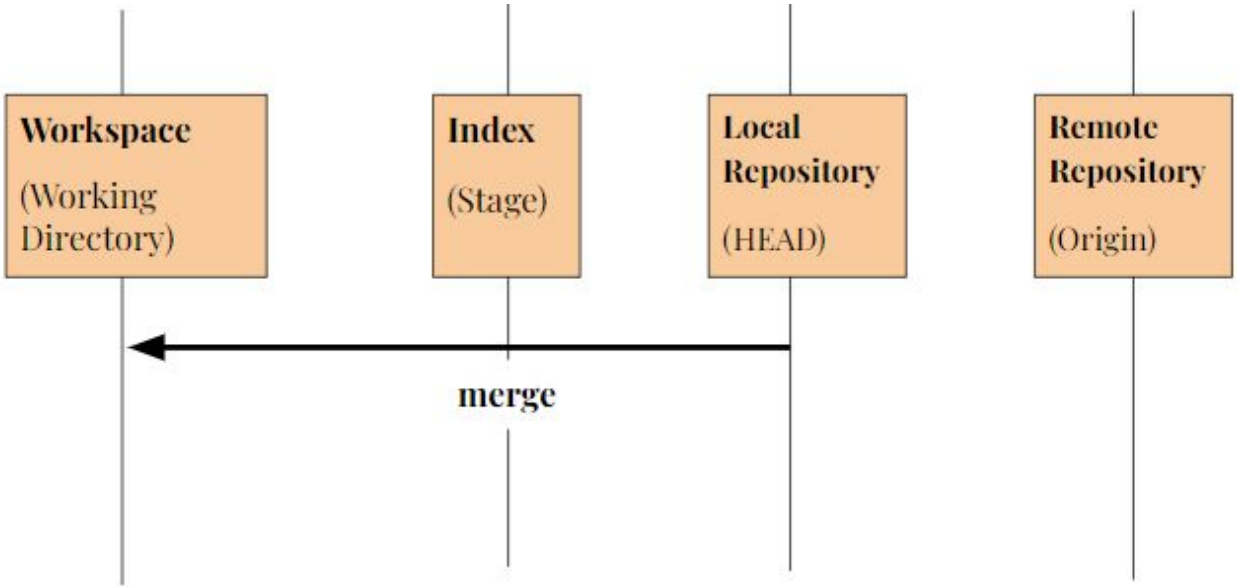


7.1 Git Pull Workflow – Step 1-Fetch from Remote

**7.2 Git Pull Workflow – Step 2- Merge to Working Directory**

7.3 Git Pull Workflow-Alternate Step-Pull from Remote

# 7.2 Git Pull Workflow - Step 2 - Merge to Working Directory

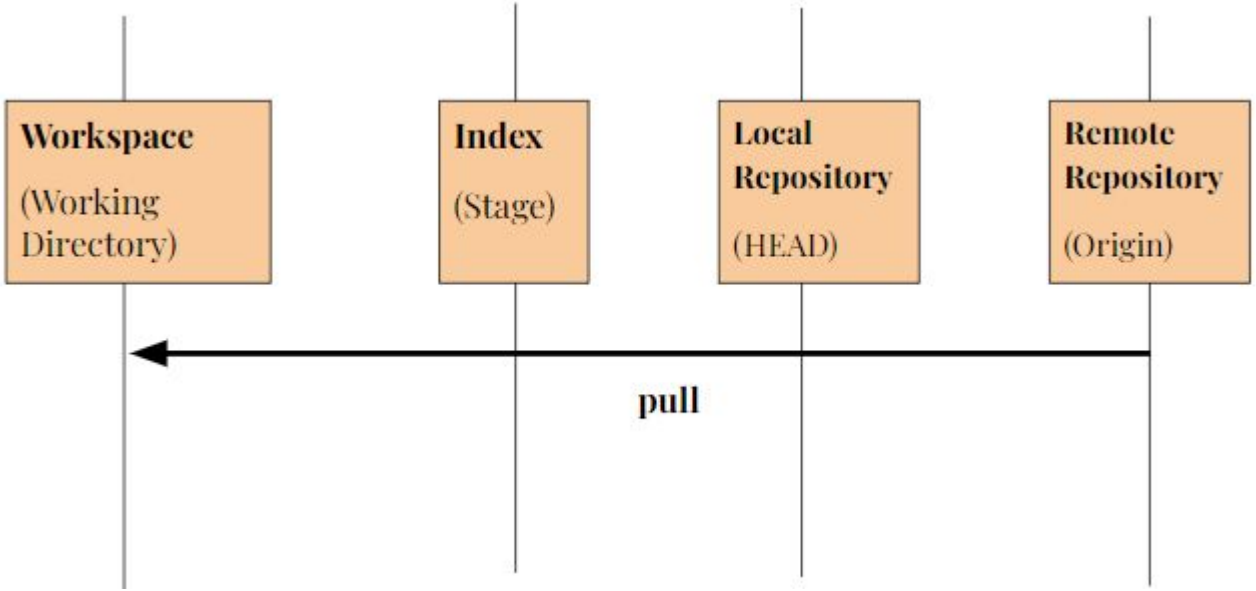


7.1 Git Pull Workflow – Step 1-Fetch from Remote

7.2 Git Pull Workflow – Step 2- Merge to Working Directory

**7.3 Git Pull Workflow-Alternate Step-Pull from Remote**

# 7.3 Git Pull Workflow - Alternate Step - Pull from Remote



# 8. Important Commands

**git init**

- Initialize a Repository

**git add <files/folder>**

- Add to Staging

**git commit**

- Commit from staging area to HEAD

**git log**

- Display Commit Log

**git status**

- Display status of the repository

**git diff**

- Show Differences between Working Directory and Stage

# 8. Important Commands

**git checkout <branch_name>**

- Bring files in branch_name to Workspace

**git checkout -b <branch_name>**

- Create a branch named branch_name and switch to it

**git branch -l**

- Show list of branches

**git fetch <remote>**

- Fetch from remote

**git merge <branch>**

- Update the current branch to merge from the mentioned branch

**git pull <remote>**

- Fetch from remote and merge to the current branch

# Github

# 9. Github

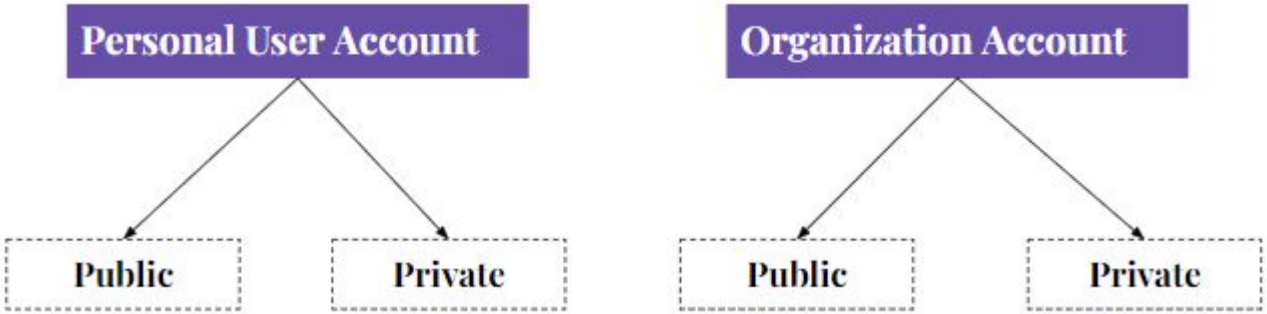
- A Platform to Host Git Code Repositories
- Can be accessed through https://github.com
- Launched in 2008
- Largest and Most Popular Source Code Repository Hosting Service
- Allows users to collaborate on projects from anywhere in the world
- Github makes development a social activity
- Adds extra functionality on top of git
    - Documentation, Bug Tracking, Feature Requests, Pull Requests and more.

9. Github

**10.Repository Offerings**

11.Summing Up

# 10. Repository Offerings



# 11. Summing Up

- Need for Version Control System
- What is Version Control System
- The Types of Version Control System
- Centralized Version Control System
- Distributed Version Control System
- Git
- Github

# End of Slides

Bring your queries and confusions to the Tutorials