**Version Control System**

A version control system allows the users to keep track of the changes in the system development projects. Using it the developers and work together on any code.

**Purpose of Using Version Control System**

* Keep track on the changes to a set of files (code).
* Tracks every individual change by each contributor.
* Facilitates a smooth & continuous flow of changes to the code.

**Different ways in which a Version Control System streamline your projects.**

* Allows easy issue tracking.
* Helps in tracing each change made to the software.
* Provides log term change history of every file.
* Easy branching & Merging.
* Tracks the status of each step in the pipeline.
* Allows developers to tell each other about different changes pushed to a GitHub Repository.
* Gives Realtime insights into the codebase.

**Benefits of version Control System.**

* Collaborate
* Store versions
* Restore previous version
* Understand what happened
* Back up

**Types of Version Control System**

* Local Version Control System: maintain tracks of files within the local system; but chances of accidentally writing to the wrong file is higher.
* Centralized Version Control System: All the files & changes in the files are tracked under a centralized server.
* Distributed Version Control System**:** It comes into the pictureto overcome the drawbacks of Centralized VCS. Here the clients completely clone the repository, including its full history. If any server dies, any of the repository can be copied onto the differ server, which helps in restoring.

**Git**

* Type of version control system (VCS)that makes it easier to track the changes.
* When you edit a file, git can help you determining: ‘What changed’, ‘who changed it’, and ‘Why’.
* Git allows multiple developer to work together and supports non-linear developments because of parallel branches.
* Git is a software that run locally. Files and their history are stored on your computer.
* Git can merge the branches so that different people can work on different parts of the same file and later merge them.

**Purpose of using Git**

* Track changes in the source code.
* Use distributed version control tool for better source code management.
* Allows multiple developers to work together.
* Supports non-linear developments because of parallel branches.

**Benefits of using Git**

* Has flexible environment
* Keeps the files in secured server.
* Create & manage remoter repository.
* Facilitates branching & merging.

**Git Repository**

* It contains all the project files and their entire version history.

Timeline

Description automatically generated

**Life Cycle of Git**

* User Clones the Git Repository as a working copy.
* After cloning, user modify the working copy by adding or editing files.
* Review changes before committing.
* Commit the changes.
* If everything is fine, Push the changes to the repository.
* After committing, if something went wrong, then correct the last commit and push the changes to the repository.

Diagram, timeline

Description automatically generated

**Git Workflow**

The 3 different file states of Git.

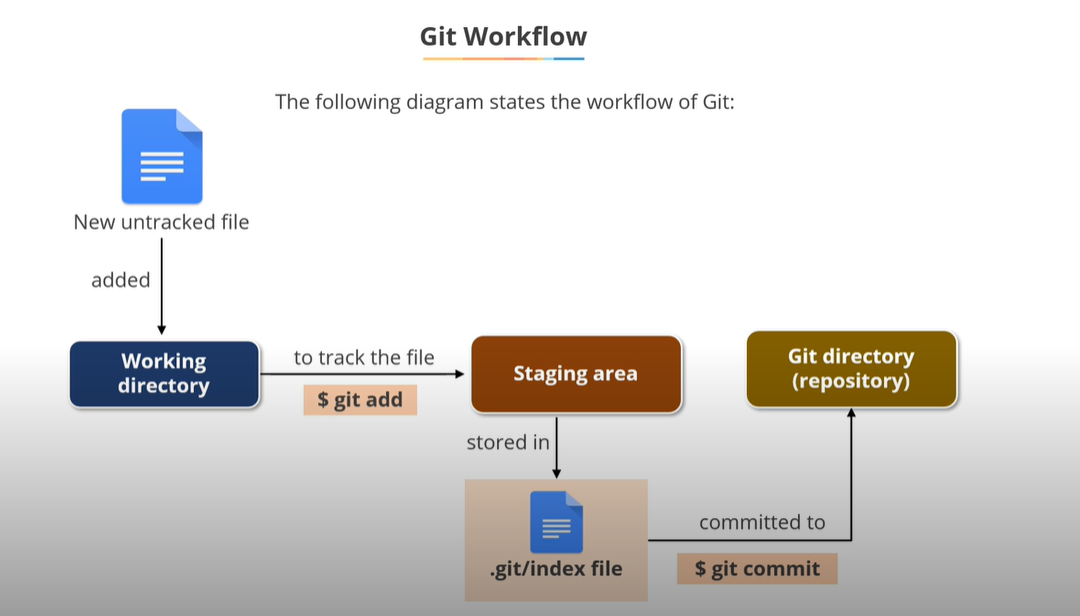
* Committed
* Modified
* Staged

Each flie states related to a specific Git section

* Working directory: Local System where we build and modify our files.
* Staging Area: It’s a place where Git keeps a reference of all the modifications to sent in the next commit.
* Git Directory (Repository): It’s a repository which stores files in compressed manner.

Git Maintains 3 snapshots of a file, which are maintied in separate Git Section.

1. The 1st section is called **Working Directory**. It refers to the directory where the command “**git init**” was executed. When a new File is added to this directory, the file is not tracked by Git.
2. For Git to track this file, we should **Stage** it. This done by **$ git add** command. Once a file is staged, it is stored in a **.git/index file** called the **Staging Area** or **Index**.
3. The index file is commited to the **Git repository** using **$ git commit** command.



**GitHub**

* It is a web-based Git repository hosting service which provides a web interface to upload files.
* GitHub supports transparency, helps to collaborate, publishes projects.

**Git vs GitHub**

|  |  |
| --- | --- |
| **Git** | **GitHub** |
| Installed & maintained in the local system. | Hosted on the web. |
| It is a command-line tool. | It is a graphical interface. |
| It is a tool to manager different versions of a file in a git repository. | It provides web-based Git repository to upload files. |

**Git Commands**

User configuration

* $ git config --global user.name "Somyakanta Pradhan"
* $ git config --global user.email somyakanta.pradhan@gmail.com (Note! Enter the e-mail address in lower case)

Create a new Repository

* $ git init

Clone an existing repository

* $ git clone [URL] -> $ git clone <https://github.com/akshu20791/batch64-simplilearn>

Check the status of file (Make changes, add files, and delete files)

* $ git status

Add changes/ Add Files to the staging area

1. Add Individual files.

* $ git add <file\_name>

1. Add Individual directories

* $ git add <directory-name>

1. Add all files

* $ git add -A

1. Add all the directories

* $ git add

Remove file from Git Repository

* $ git rm <file\_name> e.g., $ git rm file1.txt

Commit changes to the local repository

* $ git commit -m “commit\_message”

Create a branch

* $ git branch -M main

Pulling changes from a remote repository to the local system

* $ git pull <branch\_name> <remote\_url>

Push changes to a remote repository

* $ git push -u origin <branch\_name> [Push single branch]
* $ git push –all [Push all branch to remote repository]

**Deploy File to GitHub via Git**

* Install Git
* Create a GitHub account.
* Open Git Bash terminal
* Create a repository with command-> **mkdir** **<Repository\_Name>**
* Change the location to new local repository with command-> **cd <repository>**
* echo "# NewProject" >> README.md
* git init
* git add README.md
* git commit -m "first commit"
* git branch -M main
* git remote add origin https://github.com/Somya2504/NewProject.git
* git push -u origin main

**Importance of Linux in DevOps**

* Faster Development on Linux
* Open Source
* Flexible: It can be installed on any device
* Scalable: Linux is highly scalable as it can process large amounts of data

**Components of Linus OS**

* Bootloader
* Kernel
* Init System
* Daemons
* Desktop Environment
* Applications

**What is DevOps?**

DevOps is a set of practices & tools designed to reduce the lifecycle of software development process between the Dev & Operational teams.

**DevOps consists of five phases**.

* Continuous Development
* Continuous Testing
* Continuous Integration
* Continuous Deployment
* Continuous Monitoring

**CI/CD**

A CI/CD pipeline automates software delivery process. Automated pipelines remove manual errors & provide standardized feedback loops to developers.

**Continuous Integration (CI)-** Automating the integration of code changes from multiple contributors into a single software project. Key points of Continuous Integration->

* Developers needs to commit the changes in regular intervals.
* Requires regular builds.
* A general practice of CI is whenever a code commit occurs, a build should be triggered so that we can identify the errors in the early stage of the project.

**Advantages of Continuous Integration (CI)**

* Monitors the health of code coverage metrics.
* Reduce technical debts & maintenance costs.
* Automate E2E acceptance test.
* Reduce risk by providing faster feedbacks.
* CI tools quickly identify & fix the integration and regression issues.
* By automating the deployment, CI tools simplify and accelerate the delivery.

**Continuous Delivery/Deployment (CD)**- It is the extension of CI. Any successful build potentially be deployed into production.

* It is the business team who decides the best time to deliver the lates changes, not the IT team.
* CD deploys all the changes to the sources code to production automatically without the developer’s approval.
* It is software release strategy in which each code that passes the automated testing is automatically sent to production.