# VERSION CONTROL

Git & Github

## WHAT IS GIT



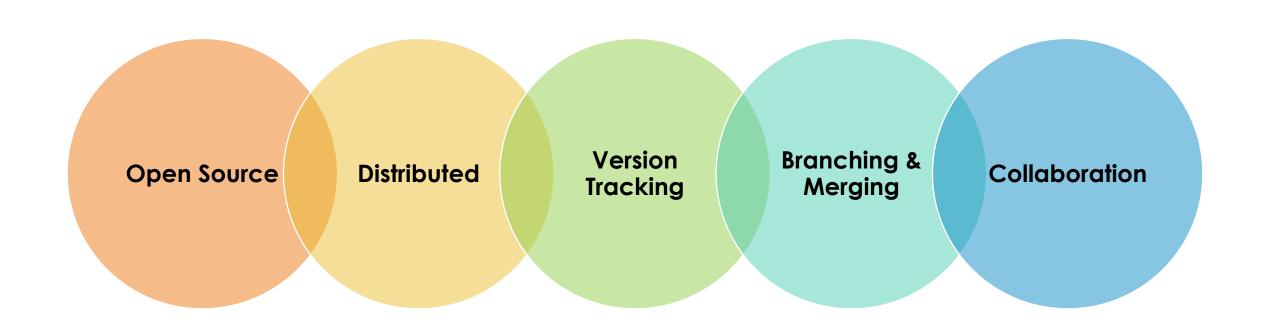
A distributed version control system (VCS).

Tracks changes in files and directories over time.



Helps manage source code for software development.

## KEY FEATURES



### MHY GITS

- Team Collaboration on a Web Development Project
- Let's Say, One team of developers is building a **React-based eCommerce** application.
- 3 developers are work on different requirement.







**Product Catalog** 

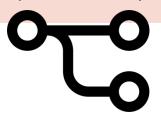


**User Authentication** 

## HOW GIT HEPLS HERE?

#### **Branching**

- Each developer can create their own branch
- feature/cart
- feature/product
- feature/authentication
- work independently.



#### Merging

- Once a feature is complete
- the changes can be merged into the main branch
- without disrupting others' work.



## Conflict Resolution

- If two developers modify the same file
- Git helps identify and resolve conflicts during merging.



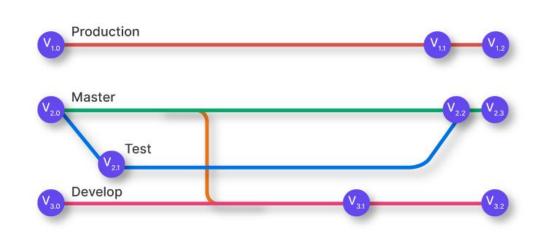
#### **Version History:**

- The team can track
- who made what changes
- and revert to earlier versions if needed.

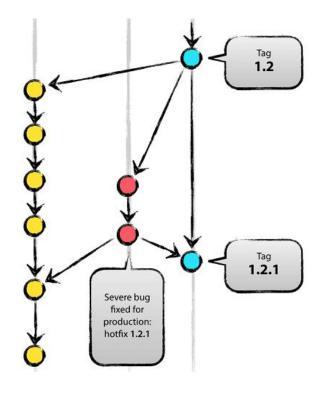


# SCENARIO: BUG FIXING AND HOTFIXES

A live production system has a critical bug, and it needs an immediate fix.



The production release is tagged (e.g., v1.0.0), ensuring the current stable version is well-documented.



Developers can create a hotfix branch from the production tag, fix the bug, and merge it back into the main branch and the development branch (dev) without disrupting ongoing development.

# SCENARIO: EXPERIMENTATION AND PROTOTYPING

A developer wants to experiment with a new feature, like adding dark mode to a website.

#### How Git helps:

**Feature Branch:** The developer can create a feature/dark-mode branch, experiment freely, and test without affecting the existing codebase.

**Reverting Changes:** If the experiment fails, the branch can simply be deleted without impacting the project.

## SCENARIO: OPEN SOURCE CONTRIBUTION

You find a bug in an open-source project on GitHub and decide to fix it.

#### **How Git Helps:**

**Fork and Clone:** You fork the repository, clone it locally, and make your changes.

**Pull Request:** After fixing the bug, you push your changes to your fork and create a pull request to suggest the changes to the main repository.

## SCENARIO: ROLLING BACK CHANGES

A recent deployment introduced a bug that broke the application

How Git helps?

Version History: Git allows you to view previous commits and identify which change introduced the bug.



Reverting Commits:
You can use git revert
or git reset to roll back
to a stable commit.

## SCENARIO: CODE REVIEWS

A senior developer reviews the code changes made by junior developers.

How Git helps?

**Pull Requests:** Developers can push their changes to a branch and create a pull request.



Review and Feedback: The senior developer can comment on specific lines of code, suggest improvements, and approve the merge.

### POPULAR PLATFORMS





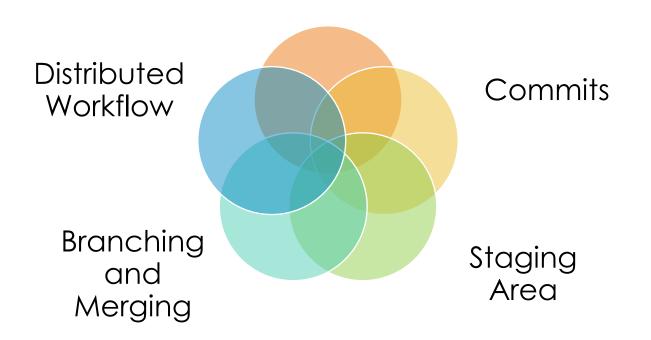


**Azure Repos** 



## GIT TERMINOLOGY

#### Repositories



## WHAT IS REPOSITORY?



A Git project is stored in a repository (repo).

Can be **local** (on your computer)

**remote** (on platforms like GitHub).

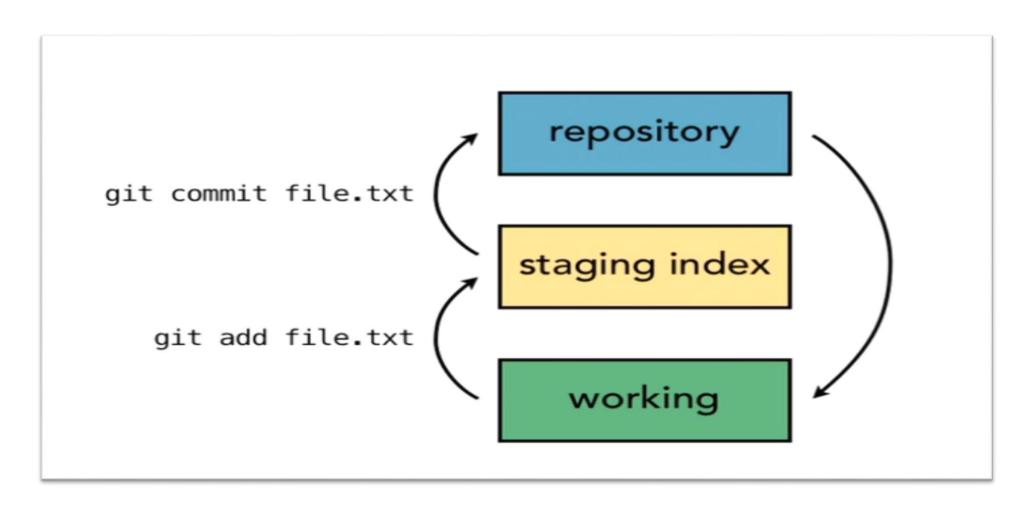
## WHAT IS COMMITS?

Git tracks changes as **snapshots** of the project at specific points in time.

#### Each commit represents a snapshot and contains

- Changes made to files.
- Metadata (author, timestamp, message).

## GIT ARCHITECTURE



## ACTIVITY

Install git on your system

Verify installation: check version using command prompt

Execute command

git -version



## GIT BUZZWORDS

- Master:
  - It is a default branch.
  - It is used by CI tools for build and deployment.
  - It is followed by the other repositories.
- Branch:
  - It is a light weight working copy.
  - It has a staging area.
  - It works without impacting the master branch.

#### • Head:

- It is a pointer to the latest commit of the working branch.
- It is present on every repository.
- It will point to the latest commit during branch switch.

#### Remote Repo:

- It is a git repository on a network outside the local machine.
- It can have more than one remote repositories pointing from the local repository.
- It can be managed and referenced with short names.

#### • Push:

- It pushes changes from the local to the remote repository.
- It is performed after committing the changes to the local repository.
- It syncs the changes with the local and remote repository.

#### • Pull:

- It transfers the updates from the local to the remote repository.
- It syncs the changes from remote to the local repository.
- It takes current code from remote repository and merges the change with the local repository.

#### • Fetch:

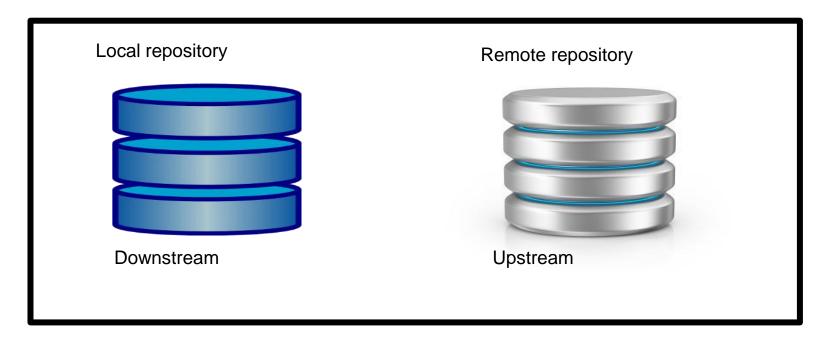
- It will not merge the changes with a local repository.
- It gives updates from remote to the local repository.
- It syncs the changes from remote to the local repository.

#### • Origin:

- It associates remote repository with names.
- It is a local name set for the default repository.
- It is useful to point the default repository when executing git commands.

#### • Clone:

- It copies the existing repository from a remote repository.
- It will get the complete repo, whereas checkout will only fetch the working copy.
- It helps to replicate the repo on the local machine.



• When the data is flowing between repository A and B, repository A is upstream and B is downstream making B pull data from repository A.

## GIT VS GITHUB

#### Git

It is installed and maintained on the local system.

It is a command line tool.

It is a tool to manage different versions of the file in a git repository.

#### GitHub

It is hosted on the web.

It is a graphical interface.

It is a space to upload a copy of the git repository.

## ACTIVITY

Open www.github.com

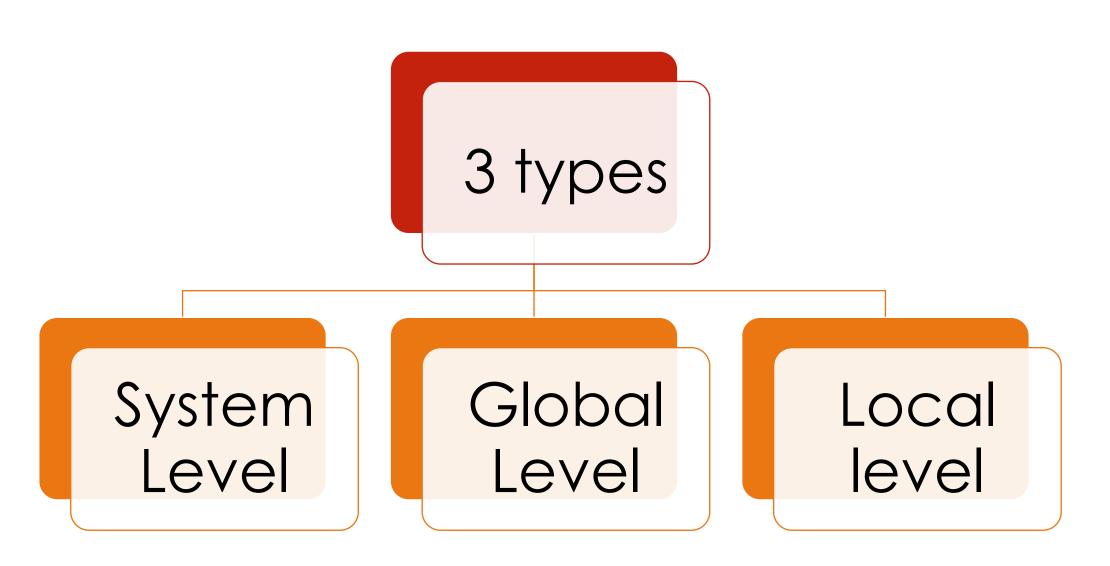
Create one account

Remember username & password

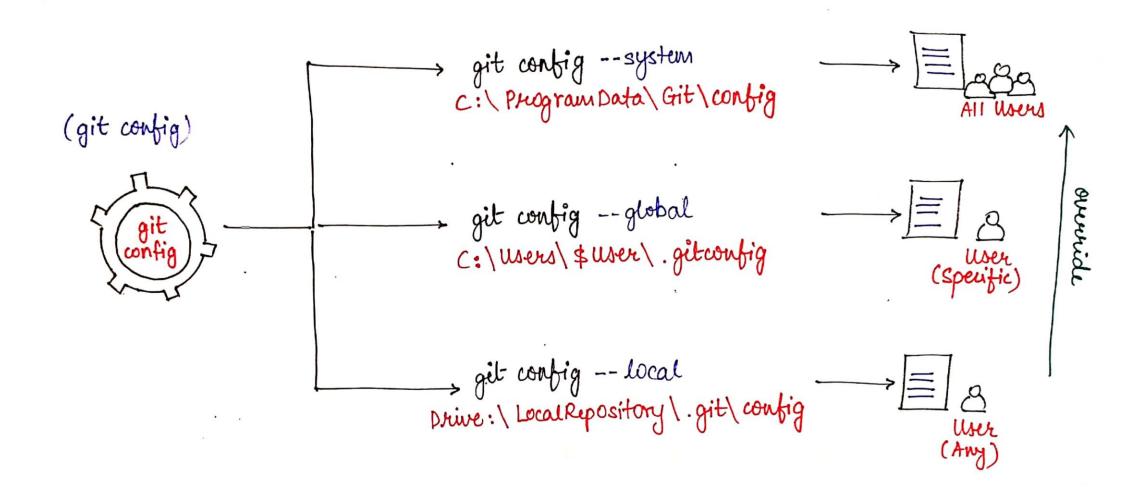
Login with the same



## GIT CONFIGURATION



## LET'S UNDERSTAND



## GLOBAL LEVEL CONFIGURATION

Global level configurations in Git are user-specific settings.

It apply to all repositories for the current user.

These settings are stored in the ~/.gitconfig file

You can usually found at C:\Users\<YourUsername>\.gitconfig on Windows OS.

## SYSTEM LEVEL CONFIGURATION

- Affects all users and all repositories on the system
- Stored in the \$(prefix)/etc/gitconfig file.
- System level username
  - git config --system user.name "System User"
- System level email
  - sudo git config --system user.email "systemuser@example.com"
- Enable System-Level Colored Output:
  - git config --system color.ui auto
- Viewing System-Level Configurations:
  - git config --system --list

## LOCAL CONFIGURATIONS

- Assume you want to set a specific username and email for a particular repository, different from your global settings, and create a few local aliases.
- Local user or email
- git config --local user.name "Your Local Name"
- git config --local user.email "local@example.com"

## GLOBAL CONFIGURATIONS

- Set Global Username:
  - git config --global user.name "Your Name"
- Set global Email:
  - git config --global user.email "you@example.com"
- Set global coloured output
  - git config --global color.ui auto
- Set global aliases:
  - git config --global alias.co checkout
  - git config --global alias.br branch
  - git config --global alias.ci commit
  - git config --global alias.st status
- View all configuration:
  - git config --global --list

## GIT COMMANDS

#### Initialize a Repository

- Command: git init
- Initializes a new Git repository in your project folder.
   Creates a .git directory to track changes.

#### Check Repository Status

- Command: git status
- Displays the state of the working directory and staging area.
   Shows untracked, modified, and staged files.

#### Add Files to Staging Area

- Command: git add <file> or git add .
- Adds files to the staging area for the next commit.
   Use . to add all changes in the current directory.

#### Commit Changes

- git commit -m "Commit message"
- Saves the staged changes as a new snapshot in the repository. Always use a meaningful commit message.

#### View Commit History

- git log
- Shows a list of commits in the repository, including messages and authors.