Learning Git for managing code and assets

knowledge hut

LEARNING OBJECTIVES

- What is Git?
- Learn how web developers work with git.
- Learn about the three-tree architecture in Git.
- List down basic difference between the CLI and GUI
- List down basic difference between the Git and GitHub
- Understand the concept of repositories and how to initialize a repository.
- Understand branches

Understanding Git

What is a GIT?

- A version control system.
- It **tracks changes** made to any kind of file, including source code and other assets.
- It enables us to **revert or revoke changes** done to a single file or a set of files, even after we modified them.
- It can **compare the changes** made to files between different versions.
- A mature and actively maintained open-source project.
- It was created by Linus Torvalds in 2005.
- It is released under GPL's open-source license.
- It is free to download and use.

What is a GIT?

- It has a **distributed architecture**. Anyone can request the latest version of the work from this server and if required, push the latest changes back to the server.

- It is fast and small.

- It **does not rely on** the **central server**.

- As most of the operations performed locally, and it gives a huge benefit in terms of speed.

- Though Git mirrors the entire repository, the size of the data on the machine is relatively small.

- It is designed to be secure and flexible, making it the de-facto choice for most software teams and enterprises.

- It is used by all kinds of companies, be it large enterprise or startups

Installation

knowledge hut

Installation

Windows: You can install Git by downloading the installer from git-scm.org

Linux: Most Linux distros have Git pre-installed. If not, it can be easily installed using the *sudo apt install git-all* command

Mac: Most likely, your Mac has Git pre-installed. If not, you can install the Xcode Command Line Tools from the App Store.

Verifying Installation

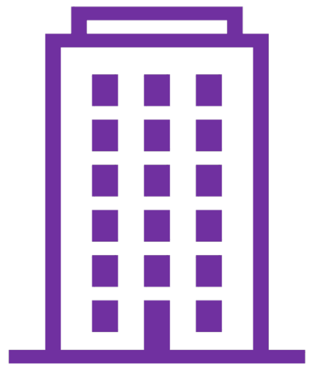
Run the following command on the terminal / command prompt to verify Git

```
$ git --version
```

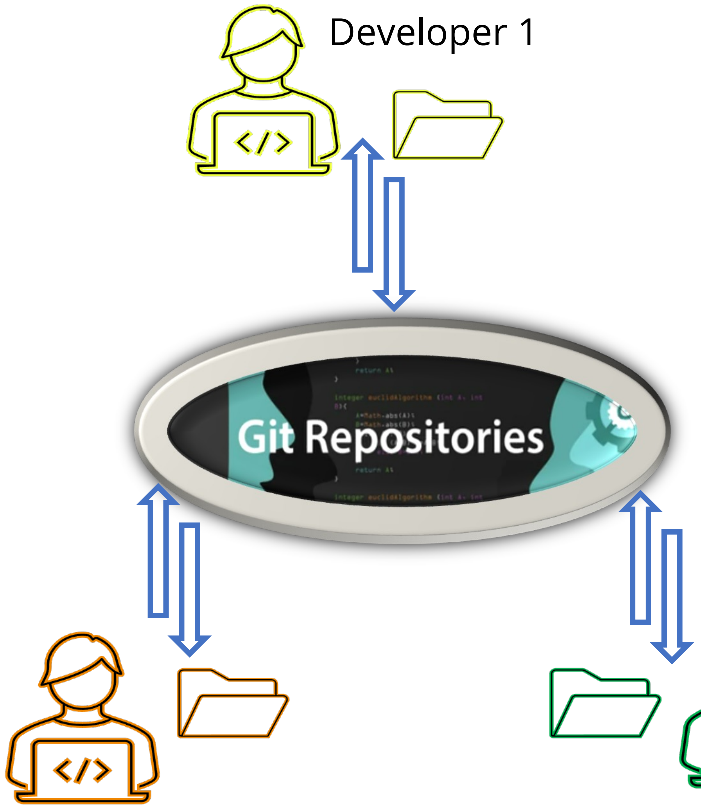
Working with Git

knowledge hut

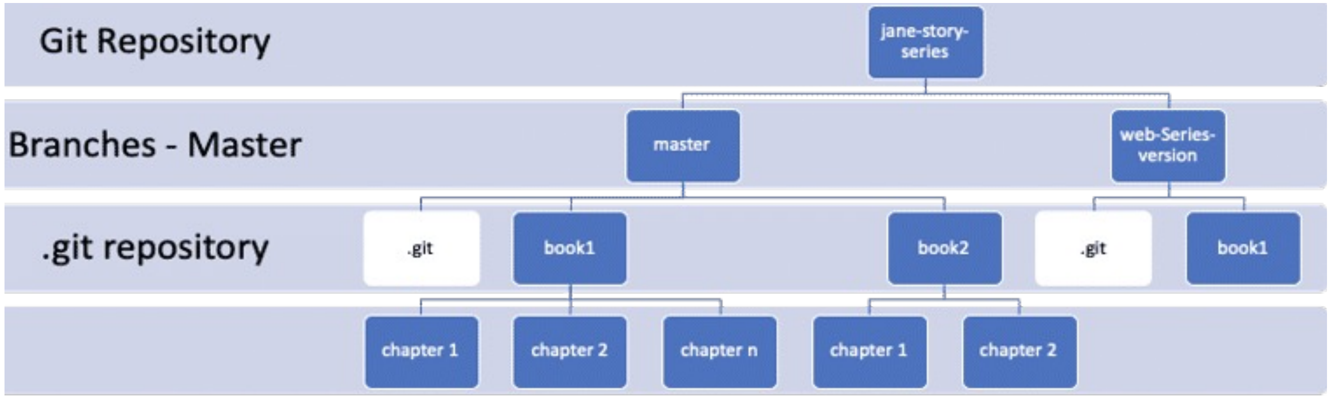
Working with GIT



Business organization



GIT Repository

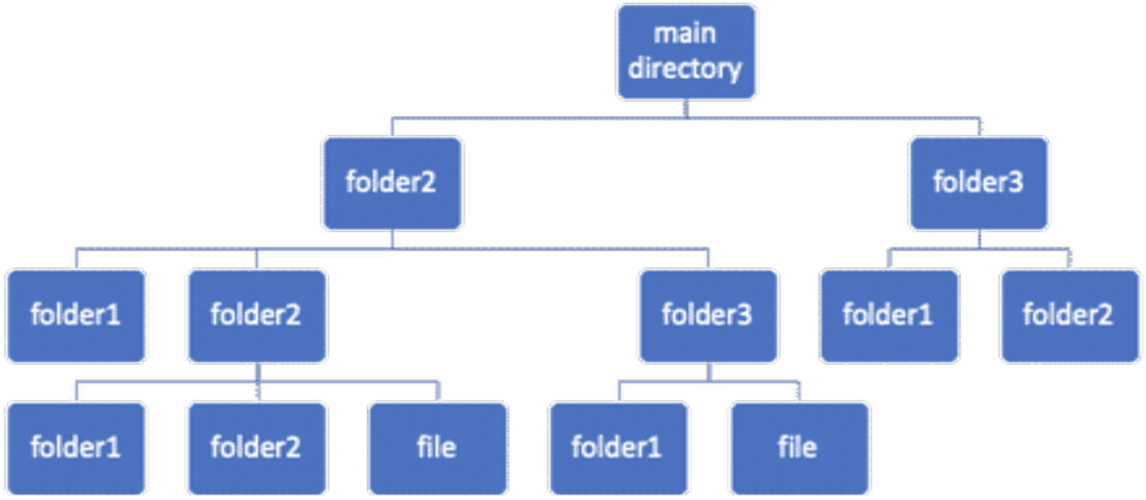


A Git Repository is a **representation of the project hierarchy with folders and files structured under it**.

- Every git project starts with a **.git** directory.
- This is the origin of the project's data, and this is where Git stores the metadata and object database for the project.
- The default branch is typically referred to as “Master”.

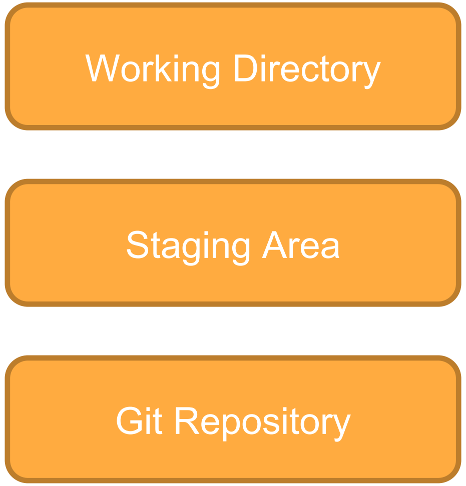
Git working directory:

- The Git working tree or Git working directory is like a workbench or a working copy of the Repository.
- It may contain the same content as a branch in the repository, but it can also be completely different, depending on the changes made to it.
- We can modify the existing files, add new files or delete a few files present in the working tree.



Git Architecture

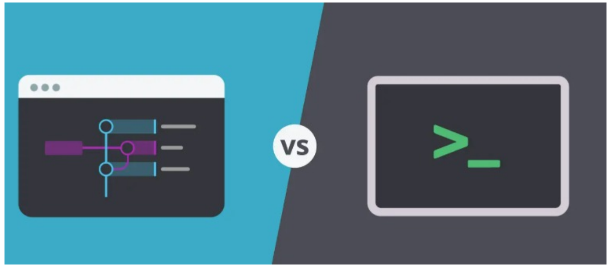
- Git uses a **three - tree architecture.** The **Staging index** is also referred to as the Staging area.
- When we want to move files between various folders of the working tree, we **"checkout"** the files into our working tree, and when we finish making our changes, we "**commit**" those changes back to the repository.
- However, in Git, a **"commit"** is a two-step process. We add our files to the **staging index**, and then from there, we commit to the **repository**.



Git's Three Trees

Command Line Vs Graphical User Interface

- Git, like many other software tools, can be operated in two ways - via a graphical user interface (GUI), or through a command-line interface (CLI).
- Git comes with built-in GUI tools like git-gui and gitk for committing and browsing.
- Graphical User Interfaces (GUI) can equally help beginners as well as the most advanced users to become effective faster.
- CLI is based on giving commands through the terminal.
- CLI is also the default choice when it comes to certain specific tasks like automation through scripts, SSH tunneling into remote servers, etc.



Git vs GitHub

knowledge hut

Git vs GitHub

Git	GitHub
Version control software for developers (VCS)	Web-based platform with git VCS features
Enables push and pull changes from another system installation	Provides project and team management features
Good for individual use	Good for collaborative work
Open source	Owned by Microsoft

Basic Workflow

knowledge hut

Steps in a Git Workflow

A Git Workflow defines or recommends how to use various Git features effectively and consistently.

1. Create a repository
2. Make your changes
3. Add the files to staging
4. Commit the files to the repository
5. Repeat steps 2 to 4

Local Vs Remote Repositories

Local Repository:

A local repository has a working copy of the file associated with it, a directory where some version of your project's files is checked out for you to work with. Local repositories are physical and locally managed by individuals.

Remote Repository:

A remote repository is like the local one in terms of operations such as branch, commit, etc. But the remote repositories are like caching proxies managed by remote links. It is a virtual storage of the current project. Hosted on a server that is more accessible across multiple devices and to multiple users - most likely on the internet or on a local network.

Initialize a Git Repository

knowledge hut

Initialize a Git Repository

The folder paths and initialization commands for Mac, Windows, and Linux are shown below.

```
$ git init
```

The **`init`** command in git opens a blank repository. It creates a .git subdirectory inside the current working directory.

Git Diff for Comparison

knowledge hut

Git diff

`git diff` is the universal diff command used for comparison.

This command helps in listing all the changes between your current working directory and the staging area.

Git diff

`git diff` is the universal diff command used for comparison.

This command helps in listing all the changes between your current working directory and the staging area.

Rolling Back

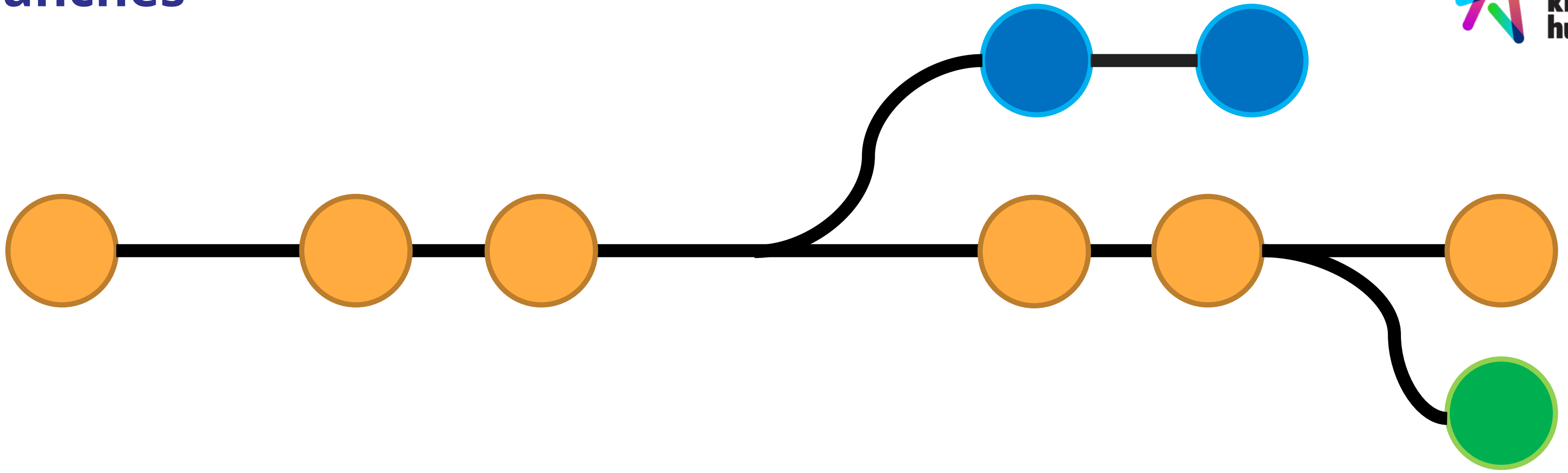
```
git revert
```

It reverses the changes made in the latest **commit** option. Which means the lines that were added will be removed, removed lines will be added, and a at the end **commit** is made again.

Understanding Branches

knowledge hut

Branches



When you need to develop a new feature, fix an issue or create an enhancement, you can create a branch and keep working on that branch without affecting the main version control track (branch)

When your fix or feature is ready, the updates from the branch can be merged into the main branch, thus releasing your feature for primary usage. This way, branches help you develop easily without affecting or damaging your application's primary line of development.

Creating and Checking Out a Branch

To create a new branch, use the git branch command. The command below will instantiate a new branch named **notifications_feature**

```
$ git branch notifications_feature
```

To start working with this new branch, use **git checkout** as shown below:

```
$ git checkout notifications_feature
```

Or you can save the effort and achieve the above in one step

```
$ git checkout -b notifications_feature
```

This will create a new branch and checkout as well in one step!

Important Commands

To list all branches, use

```
$ git branch
```

To delete a branch, use the following notation. This will not work if you have unmerged updates in the branch

```
$ git branch -d notifications_feature
```

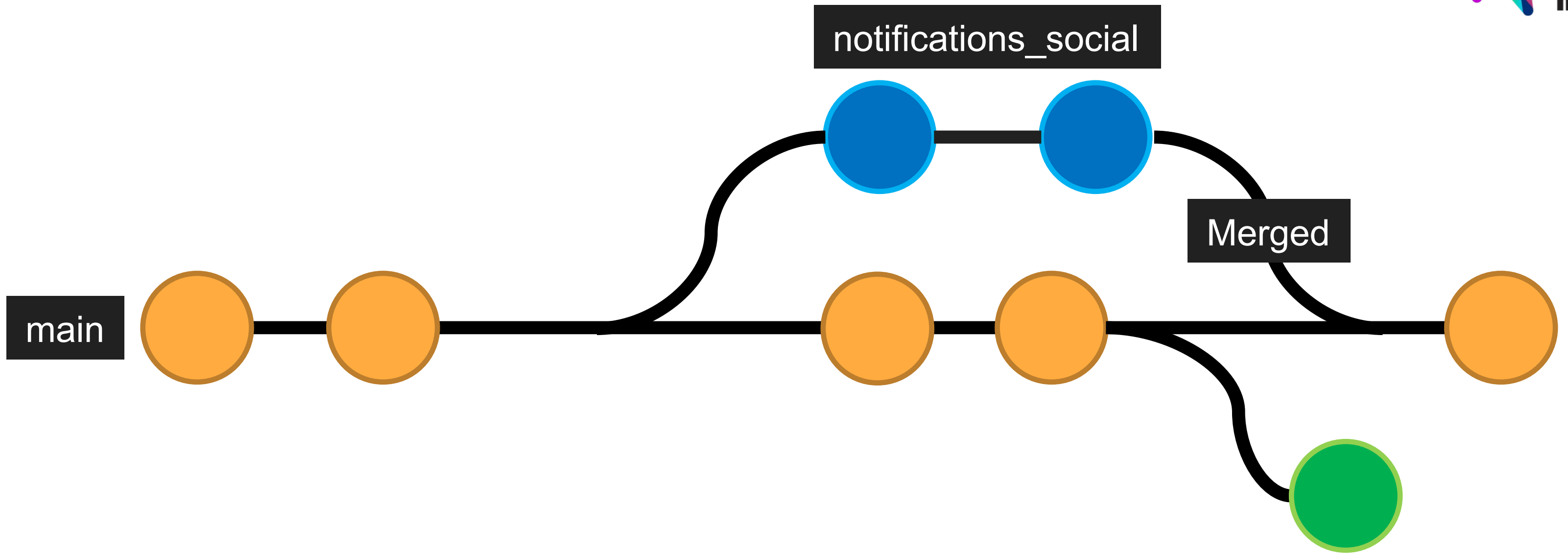
Force delete a branch even if there are unmerged changes

```
$ git branch -D notifications_feature
```

Rename the currently active branch

```
$ git branch -m notifications_social
```

Merging Branches



Use **git merge** to merge a branch into the **main** branch. If you're on the branch, checkout to main/master first and then merge

```
$ git checkout main
$ git merge notifications_social
```

Thank you