PART NUMBER 3

Every Git repository has its own structure. To fully utilize the program, you must first grasp the structure of a Git repository. If you have any questions concerning the contents of the repository or how Git is organized. Git is a fairly complex framework. It is a free and open-source distributed version management system designed to handle projects of all sizes. It helps you to complete this task fast and efficiently. After you've used all of the commands and functions, you'll have a better grasp of the Git repository structure. The graphic below depicts the repository's organizational structure. You should now have a better idea of how it functions and how it appears as a consequence. Here is an illustration of the structure of a Git repository.

**Objects/ Folder**

Git object data is stored at this place. This contains all of the files you've ever checked in, as well as your commits, trees, and tag objects.

**Objects/[0-9a-F][0-9a-F] Folders**

Each newly produced object has its own file in which it must be saved. The objects are dispersed over 256 subdirectories using the first two characters of the SHA1 object name to make the number of directory entries in the objects themselves reasonable. Furthermore, goods found here are typically referred to as loose or unpacked objects.

**Objects/Pack Folder**

Index files are also placed in this directory to allow for random access to the files that contain a large number of compressed items.

**Objects/Info Folder**

This directory provides further information about the item that has been added.

**Refs Folder**

References are kept in subdirectories in the Refs Folder. The Git prune tool is aware that items in this directory and its subdirectories that are accessible through references should be maintained.

**Folder Refs/Heads**

Commit objects can be found here.

**Refs/Tags/Folder**

Contains the name of any object.

**Refs/Remotes/ Folder**

The commit objects from branches copied from a remote repository are saved in the Refs/Remotes/ folder.

**Packed-Refs**

The file has a lot of tags and packed heads. It is practical for getting to the repository quickly.

**HEAD FILE**

The current branch is mentioned in this file. This instructs Git to use which parent for your subsequent commit.

**Setup File**

This is the primary Git setup file. It saves your Git project settings, such as remotes, push configurations, tracking branches, and more. The configuration will be loaded first from this file. Then, if present, from a /.gitconfig file, followed by a /etc/gitconfig file.

**Branches**

This is an old way of storing URL shorthand’s for Git fetch, pull, and push. This old technique is unlikely to be found in any current repositories. Even if you don't manage a contemporary repository, you might find it useful.

**Hooks Folder:**

This directory includes shell scripts that are executed after running the corresponding Git commands. Git will try to run the post-commit script, for example, after you commit.

**File Index:**

The Git index is a staging zone located between your working directory and your repository. The index may be used to gather a bunch of changes that you want to commit all at once.

When you generate a commit, the information that is really committed is stored in the index, not in your working directory. The binary file contains an ordered series of pathnames, each with rights and the SHA-1 of a blob object.

**Info Folder**

This directory's Info Folder contains repository-related information.

**Logs Folder**

All modifications to the repository's references are recorded in the logs folder.

**Logs/Refs/Heads**

This folder, called Logs/Refs/Heads, keeps track of any alterations made to the various branch tips.

**Modules Folder:**

The Logs/Refs/Tags/folder has a record of all modifications made to the various tags.

All of the submodules' git repositories are located in the modules folder.

**Working Tree**

Folder for connected working trees containing administrative data Part of a linked working tree that is relevant to the working tree is contained in each subfolder.

The Git repository structure is seen above. It will make using Git much simpler after you study it, comprehend how the layout works, and then learn about Git commands. As a consequence, you'll be more productive at work.

1. **MECHANISM**

**These are the methods to ensure that task are done with missing any thing**

* Be clear on your mission
* Know your values
* Understand and define how you prioritise.
* Focus
* Think about what is needed next
* Always keep learning.
* Keep the plan alive.

**TOOLS**

* Network Diagrams
* Critical Path Method or CPM
* Gantt Chart

These are some tools which we can use to ensure are task is not missing some thing or on the correct path

**INTEND:**

1. **Network diagrams** are highly valued project management tools that may be particularly useful in project monitoring. They are also referred to as "arrow" diagrams since multiple arrows are utilised to represent priority or dependency between distinct project tasks. Many assumptions are made while constructing a network diagram. The original assumption is that all outstanding business was completed prior to the start of the new initiatives. The second assumption is that arrows imply logical relevance; so, project tasks must be completed in the right order based on the depiction. The network diagram must begin and end with the ultimate hypothesis, which is a single event. The Network diagram's Slack time feature, which signals supplementary information.
2. **Critical Path Method (CPM):** To track their projects, project managers typically use the critical path approach, also known as CPM. Furthermore, it allows managers to track the project's development in real time. If you use CPM, you may easily schedule follow-up meetings if any project deviations or impediments are discovered.
3. **Gantt chart**: If you work as a project manager or in management, you should be familiar with the Gantt chart. The start date of the task is indicated on the left, and the completion date is given on the right. As a manager, you may rapidly track progress using Gantt charts. Gantt charts give a visual representation of the project that may be used to track its progress.