**What is a git?**

|  |
| --- |
| **Git:** Git is a distributed version control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows. |

**What is GitHub?**

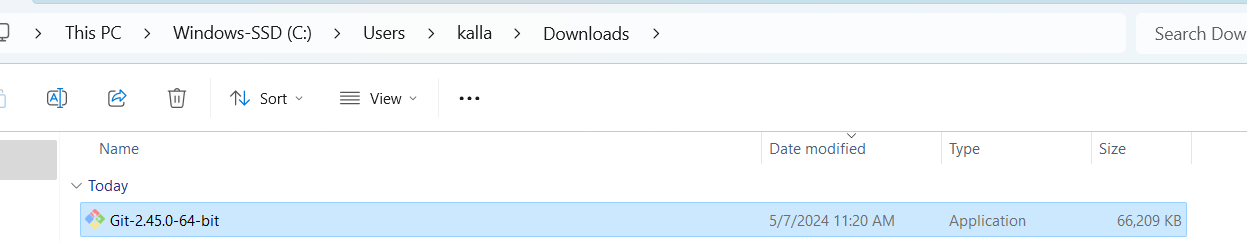
|  |
| --- |
| **GitHub:** GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. |

**Git installation**

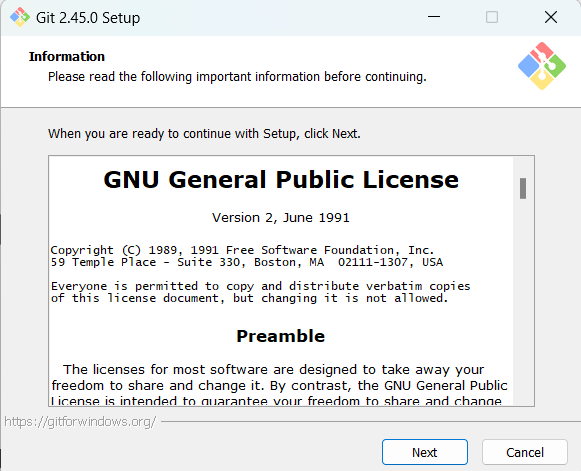
1. **Download the git from the below link**

[**https://github.com/git-for-windows/git/releases/download/v2.45.0.windows.1/Git-2.45.0-64-bit.exe**](https://github.com/git-for-windows/git/releases/download/v2.45.0.windows.1/Git-2.45.0-64-bit.exe)

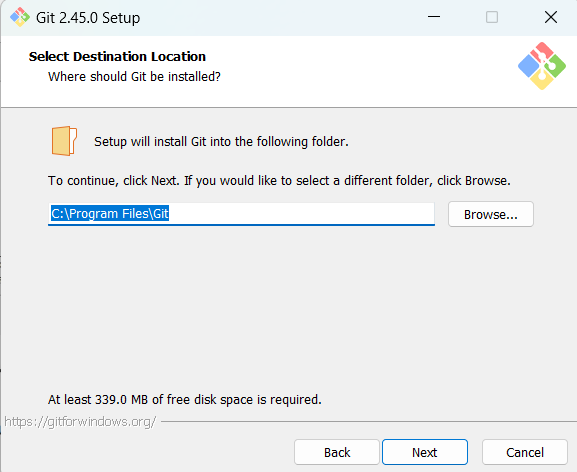
1. **Once you click the above link software will be downloaded into downloads folder like below.**

****

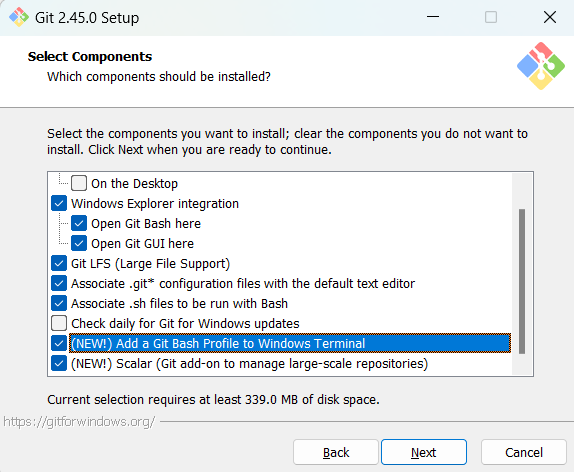
1. **Install it by clicking the above downloaded file**



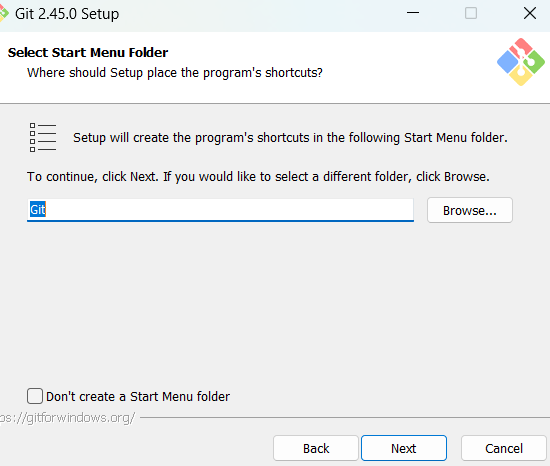
**Next**

****

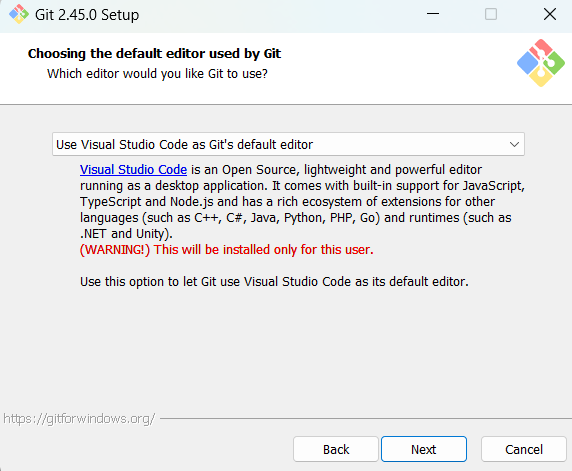
**Next (Select it Bash Profile)**

****

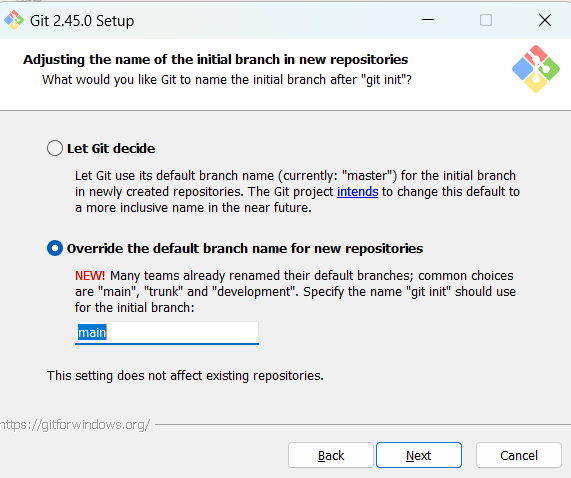
**Next**

****

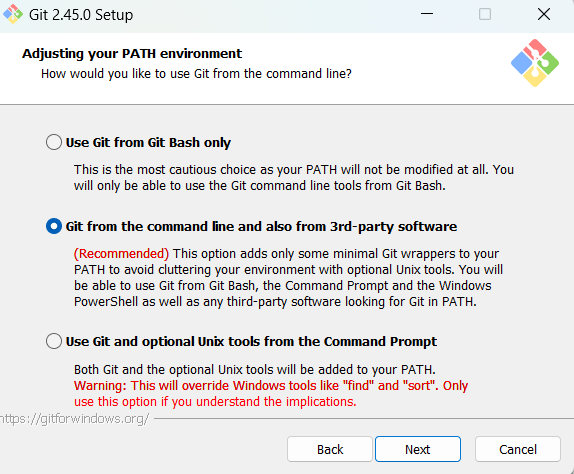
**Use Visual Studio Code as Git’s default Editor**

****

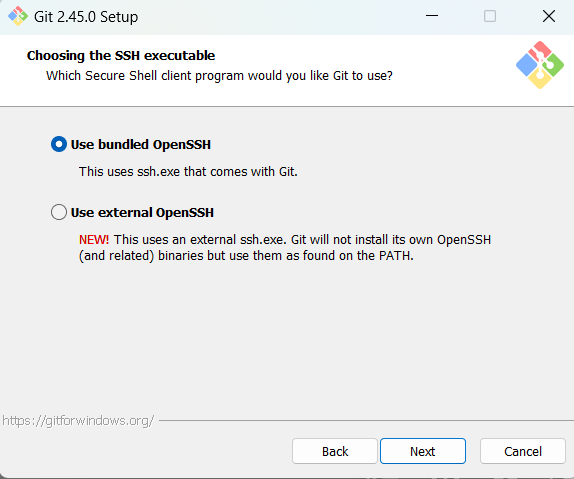
**Next(select main i.e.. ) Override the default branch name for new repositories**

****

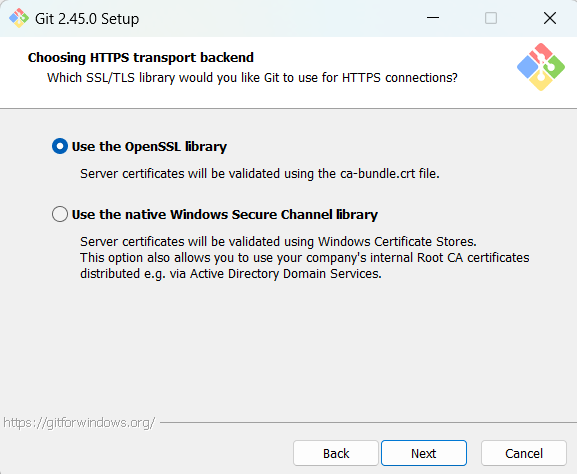
**Next**

****

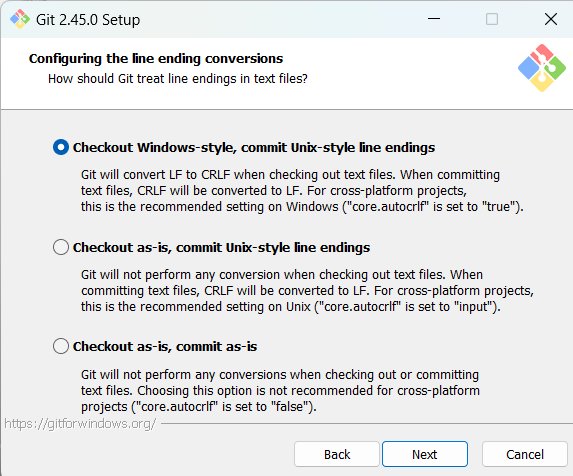
**Next**

****

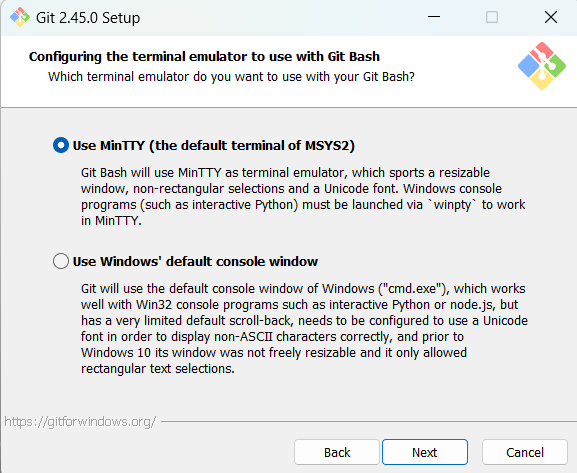
**Next:**

****

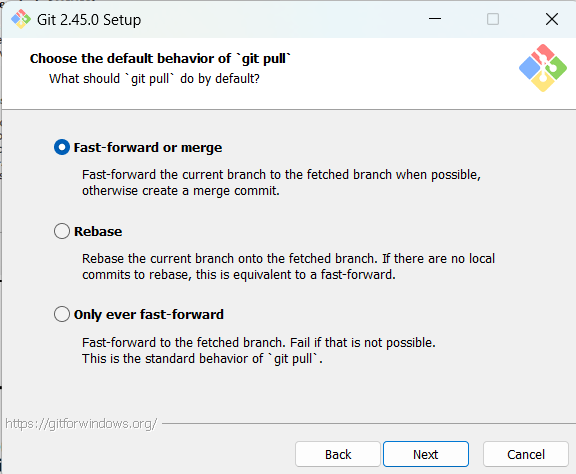
**Next**

****

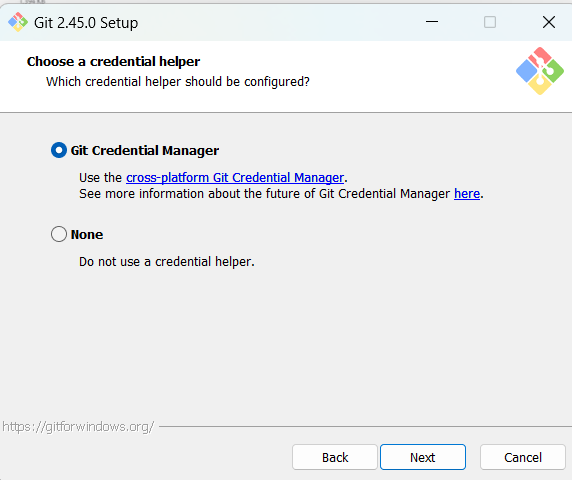
**Next**

****

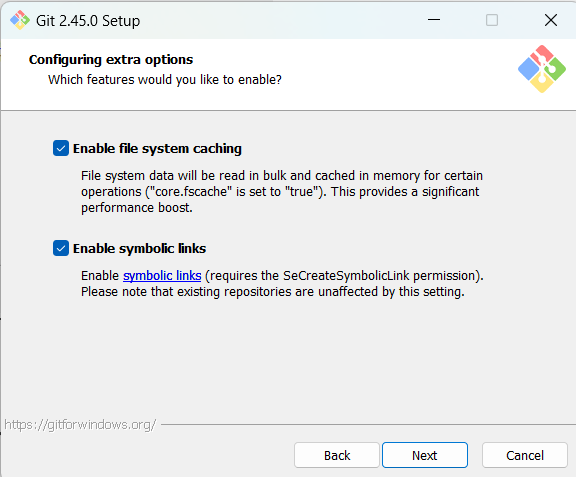
**Next**

****

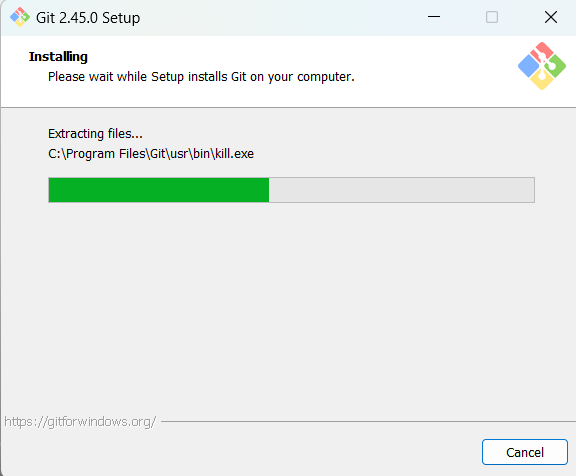
**Next**

****

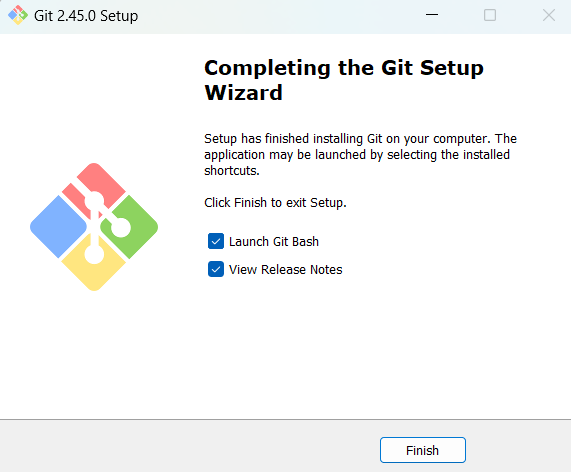
**Next(Enable symbolic Links)**

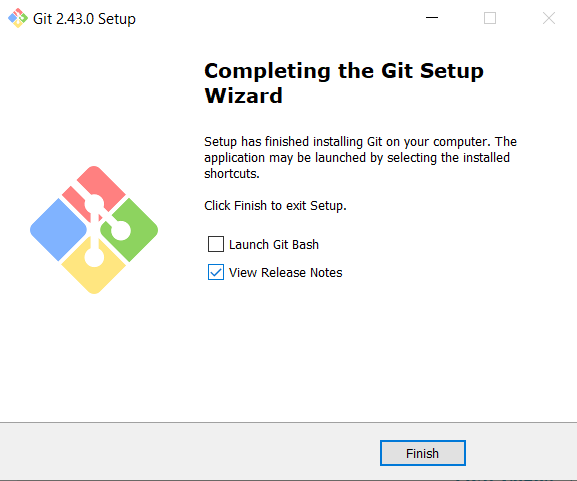
****

**Next**

****

**Next(Finish)**

****

****

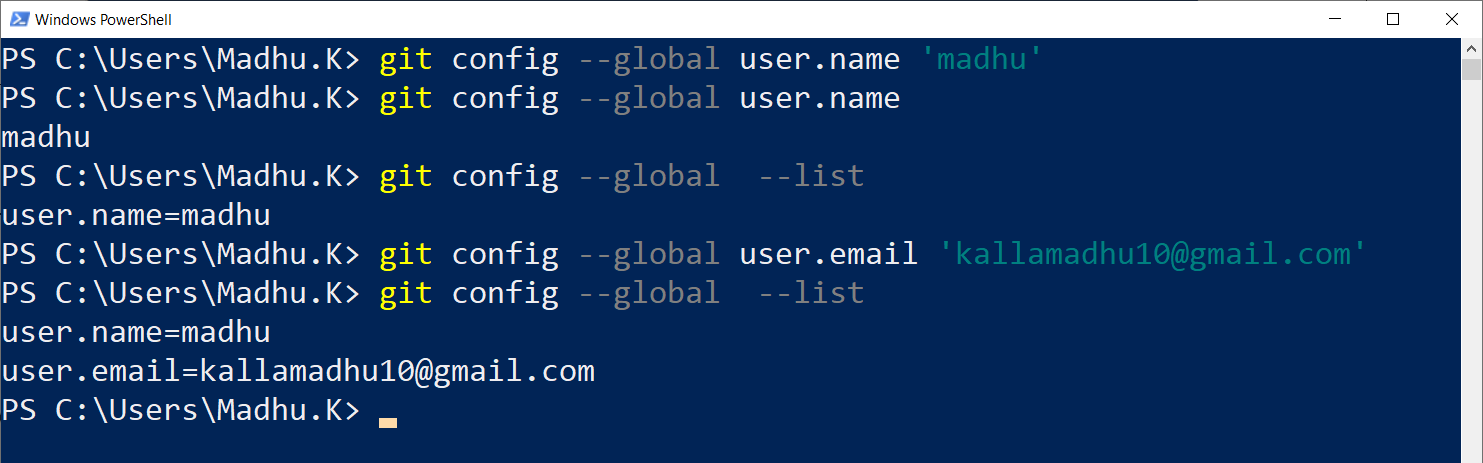
**Done**

|  |
| --- |
| Now we got the git setup. |

**Verify whether the git was installed correctly or not**

|  |
| --- |
| 1. Open the command prompt or power shell |

**When you work with git you need your own identity right? So How we can set it?**

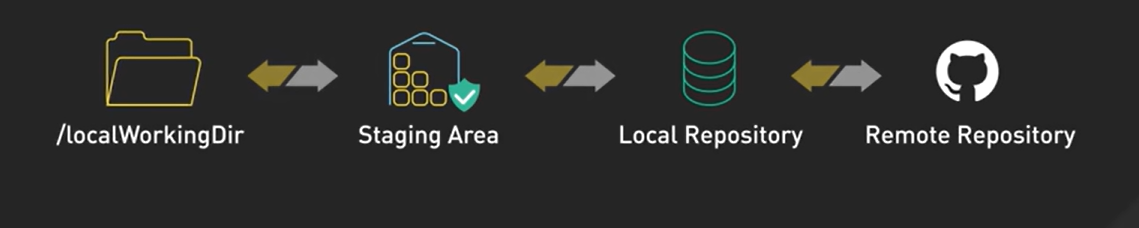
****

**To work with git better you need to have an editor( or IDE).**

|  |
| --- |
| I have installed VS-Code  Create a folder and open it in VS-Code    Now open the terminal    Terminal |

**Ter are 4 main locations where your lives in git**

1. Local Working Directory
2. Staging Area
3. Local Repository
4. Remote Repository



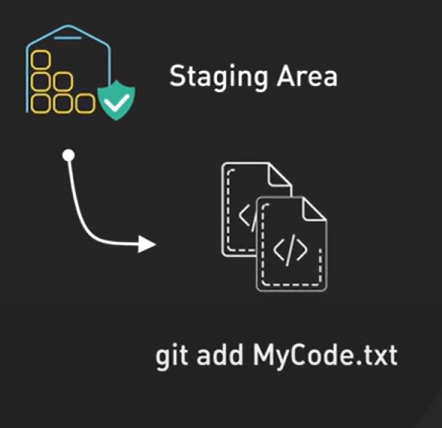
Local Working Directory

|  |
| --- |
| 1. Where we write and edit files locally 2. It is our play ground |



Staging Area

|  |
| --- |
| 1. It’s a Temporary holding spot for changes before committing |



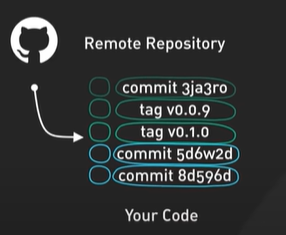
Local Repository

|  |
| --- |
| 1. This is where we store committed changes locally |

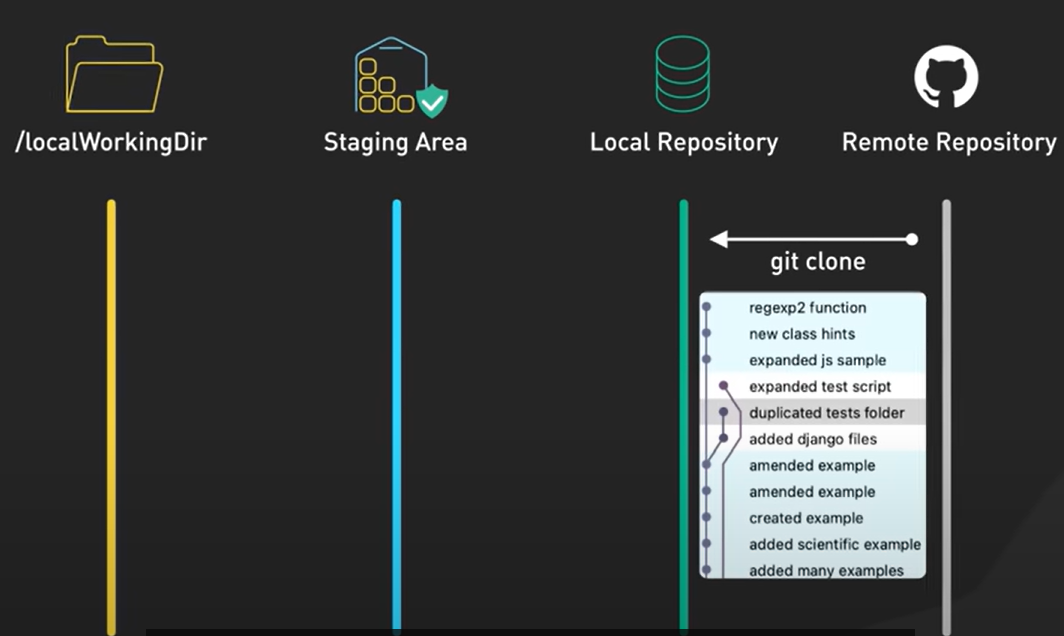


The Remote Repository

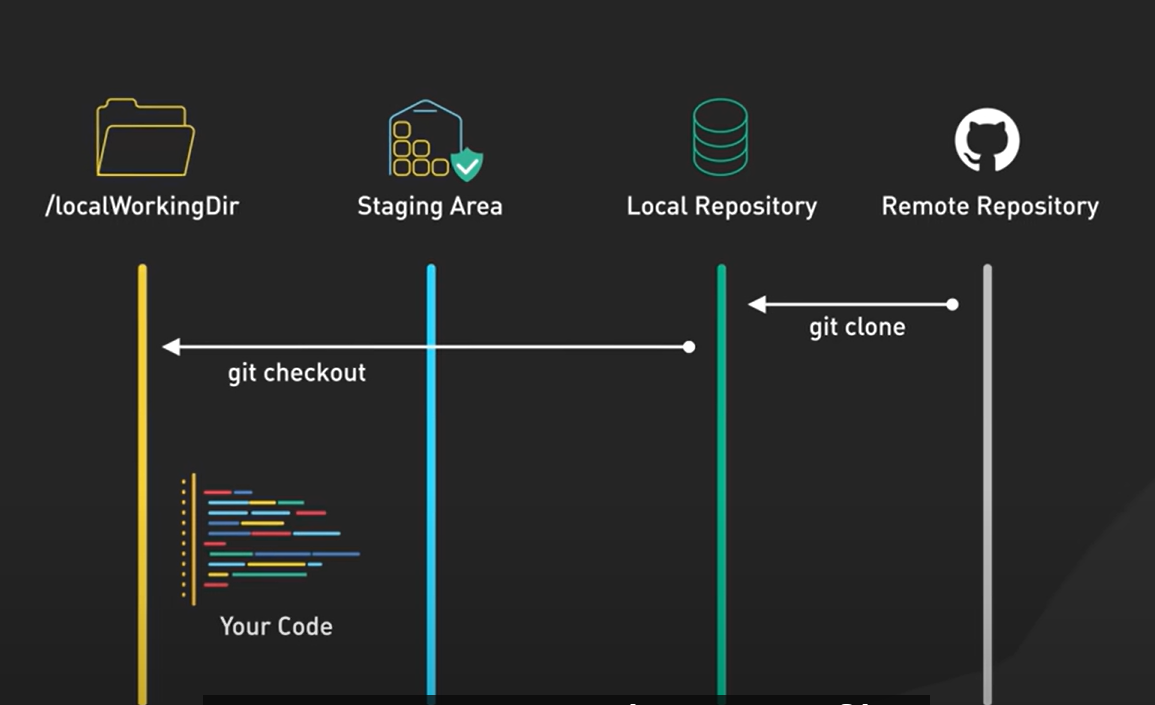
|  |
| --- |
| A server like git hub for sharing and backing up code |



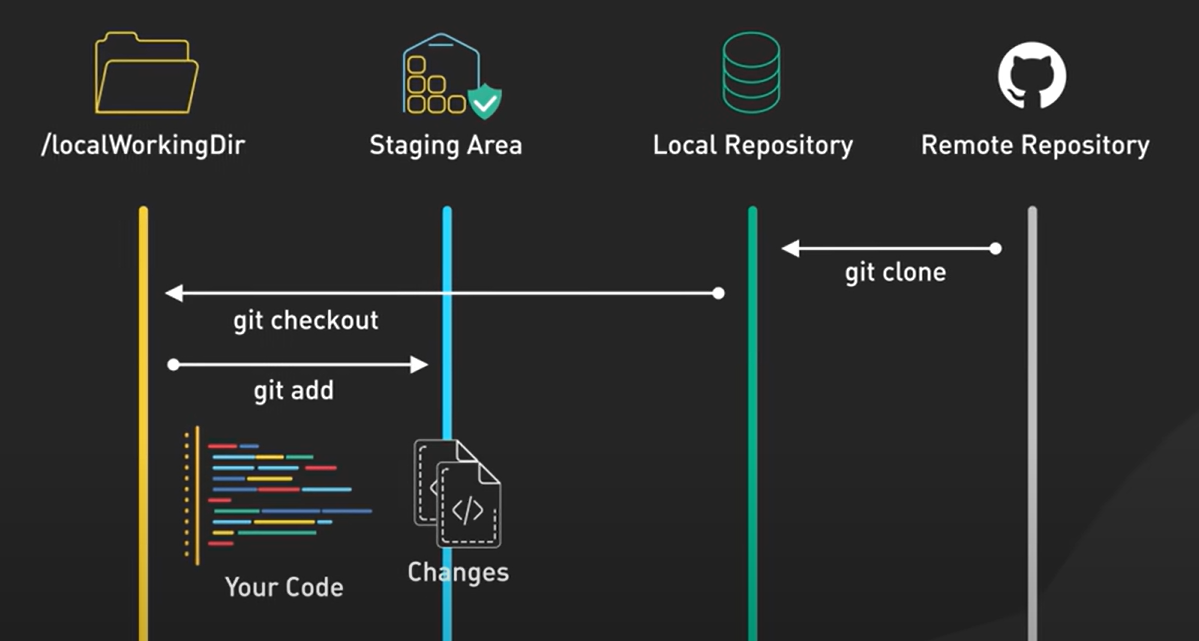
1. With these locations in mind, let’s visualize where our code travels throughout it’s journey
2. Most git commands move files between these four locations
3. First step is to git clone an existing repository so you have a local version of the project to work on, to complete with all it’s history.



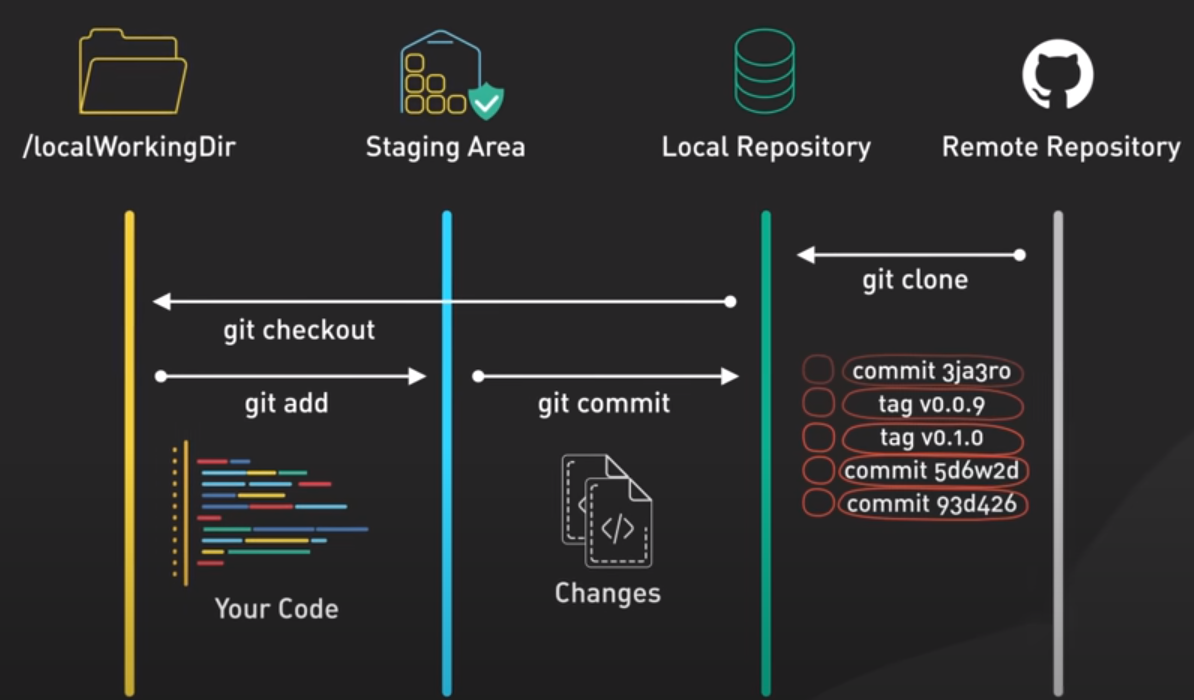
1. With the repo cloned locally, lets look at where the code lives. When you start working on file, you are in the working directory. This is your local development environment where you make changes to your code.



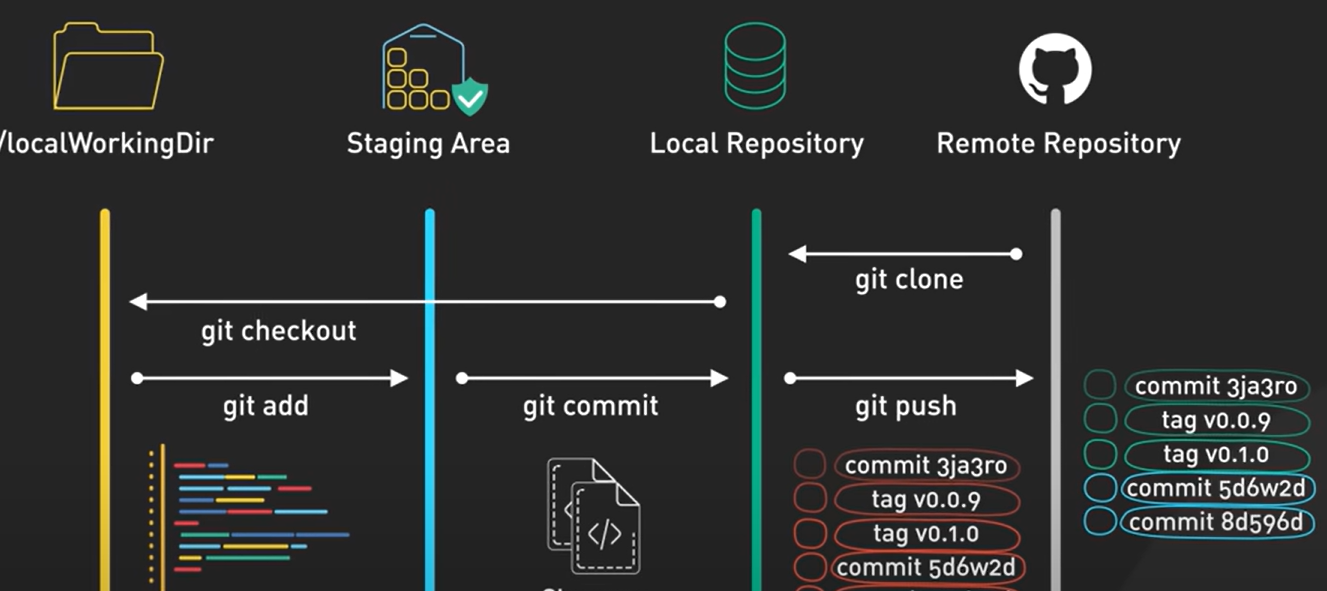
1. When you are ready to commit your changes, you will use git add to stage a snapshot of those files in the staging area. Think of this as a checkpoint, where your changes are gathered up and ready to be finalized



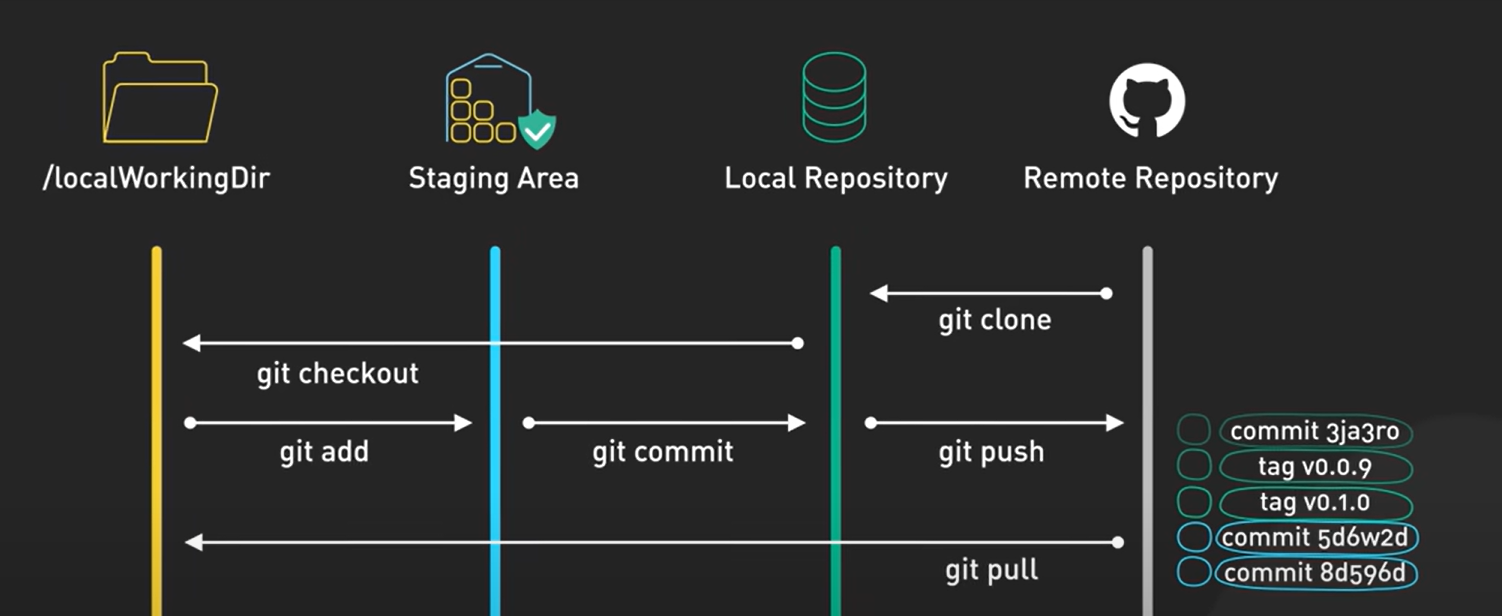
1. Next step is to use git commit which takes a snap shot of the staging area and saves it to your Local Repository. This locks in those staged changes by creating a permanent record that you can refer back to, like a snapshot in time.



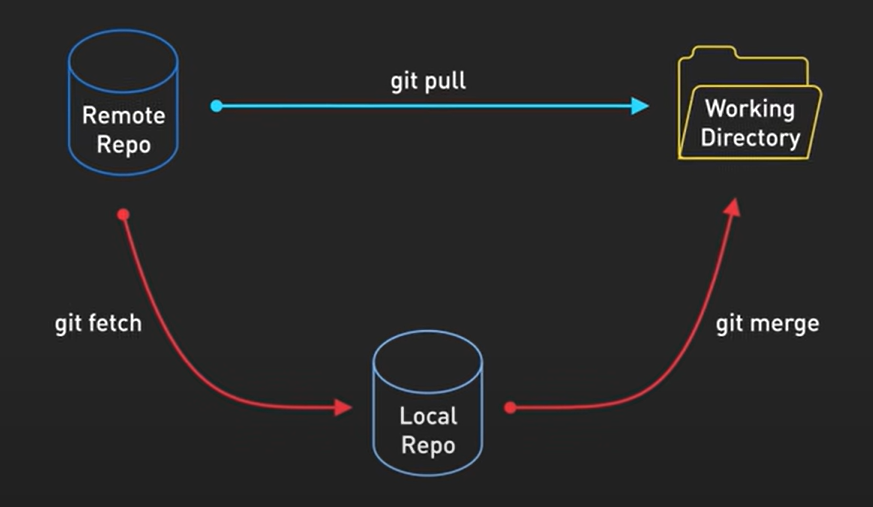
1. Your code doesn’t just stay on your local machine. When you’re ready to share your progress with the team or back up your work, you use git push to send your commits to the remote Repository



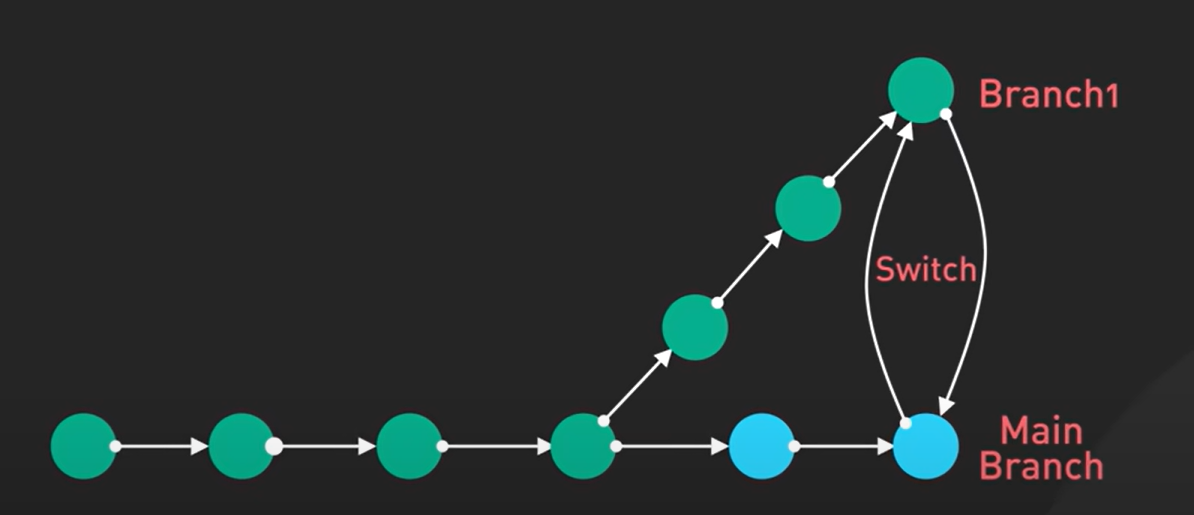
1. This is often a shared server where your team can collaborate, like GitHub or Bitbucket. Collaboration in git is a two-way exchange to integrate your teammates’ work, you use git pull, which fetches changes from the remote repository and merges them into your local repository



1. Git pull combines 2 commands: git fetch , which grabs the latest updates and git merge which integrates these updates with your work.



1. There are times when you need to switch contexts, perhaps to fix a bug on another branch, that’s where git checkout or git switch comes in. it allows to switch between different branches to work on specific features. Git branching allows you to diverge from the main codebase to develop a new feature without impacting the main code.

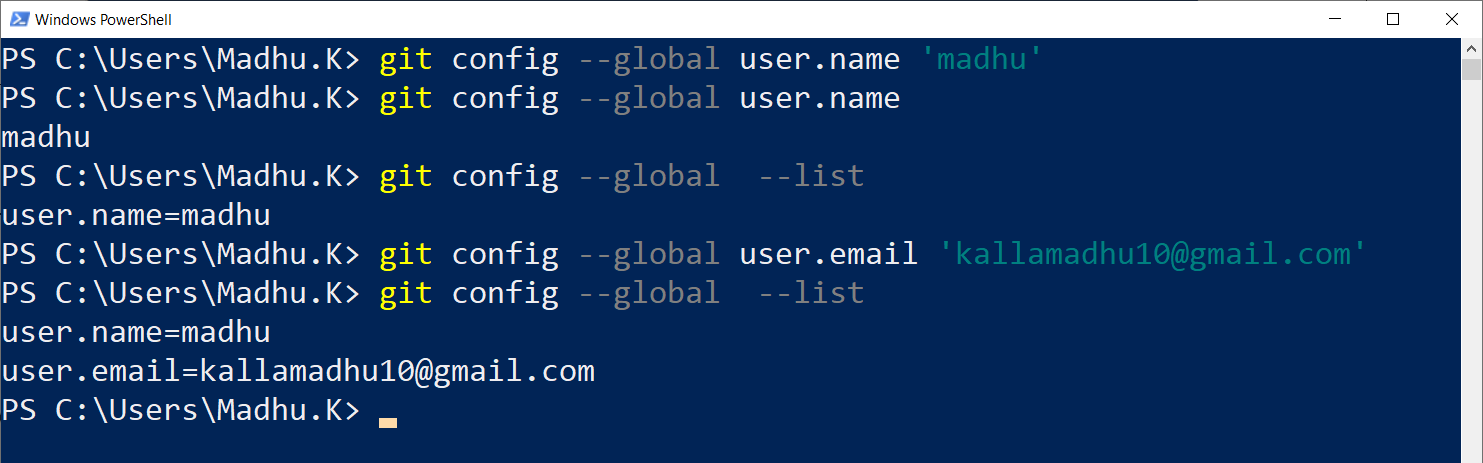


1. Some key concepts include creating a new branch with git branch, switching between branches using git switch, merging branches together with git merge. And resolving merge conflicts when changes overlap.
2. Branching enables isolated development and collaboration workflows
3. Many developers use graphical tools like GitHub Desktop and Source Tree. These tools provide visual interfaces and shortcuts for common commands

**Verify whether the git was installed correctly or not**

|  |
| --- |
| 1. Open the command prompt or power shell |

**When you work with git you need your own identity right? So How we can set it?**

****

**To work with git better you need to have an editor( or IDE).**

|  |
| --- |
| I have installed VS-Code  Create a folder and open it in VS-Code    Now open the terminal    Terminal |

**What is working directory**

|  |
| --- |
| Here First project is working directory |

**Staging Area**

|  |
| --- |
| 1. It is a place where git works 2. Whatever you are doing you have to push that into staging area. |

**Commit history**

|  |
| --- |
| Every time you commit a new version will be created in git. It helps us to go back if you want. |

**Local Repository**

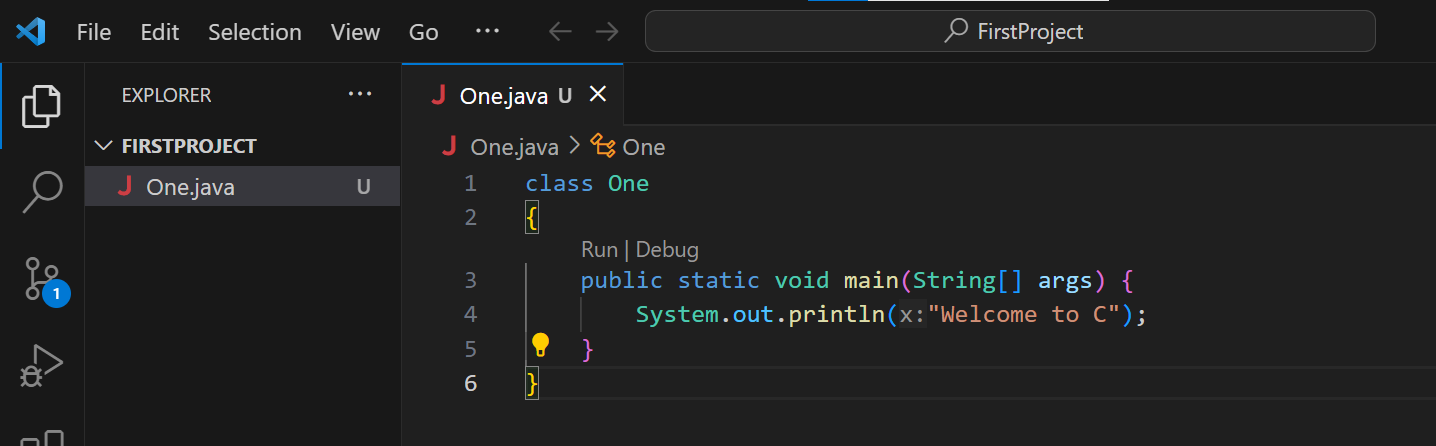
|  |
| --- |
| If you give the below command then it will give the status. If no git repository exists we will get error like below    Command to create local repository is(to do that initialize the git see the below command)    Empty git repository is created but it is a hidden folder you can’t see directloy in First Project  The staging area and commit history will be placed within the this folder called “.git”  **Note:**  If you are creating a file in working directory! Remember that, that file is part of working directory it is not part of git okay.  **Run git status command**  Now run the git status command. And you can see a new branch is created with a name called master. |

**Creating a new branch with name main**

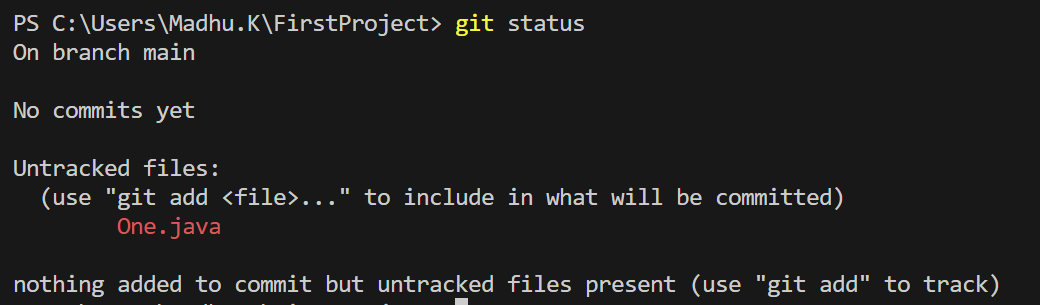
|  |
| --- |
| Better don’t work on master branch so we will create a new branch with name (our wish) main.  Remove the master branch and create a new branch using below command      Now run the git status and you can find the main branch |

**How to move a file written in working directory to staging area**

1. Creating file in working directory(see the below image)

****

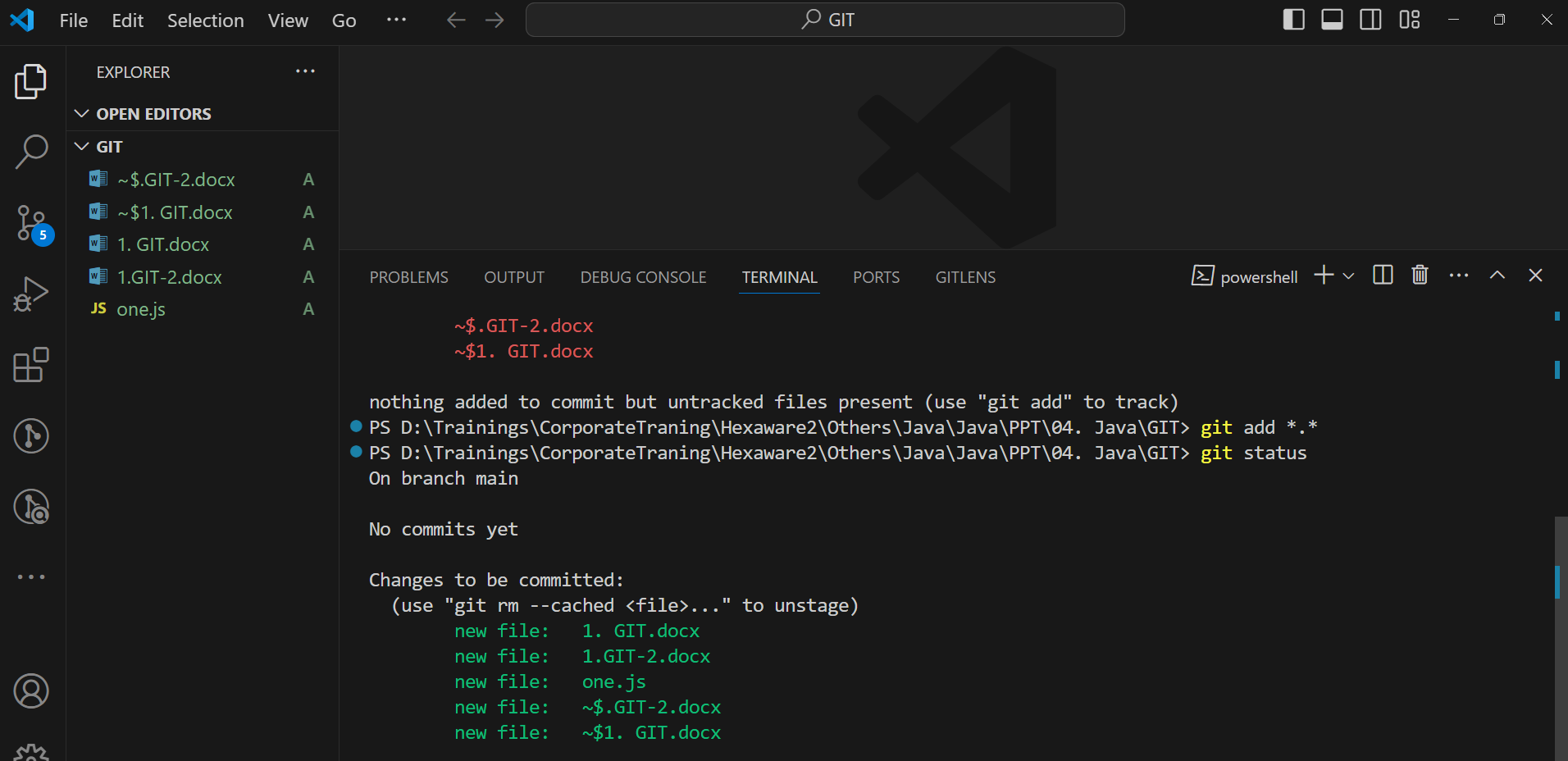
1. After creating file check the git status like below, then it will show the new file as untracked files list



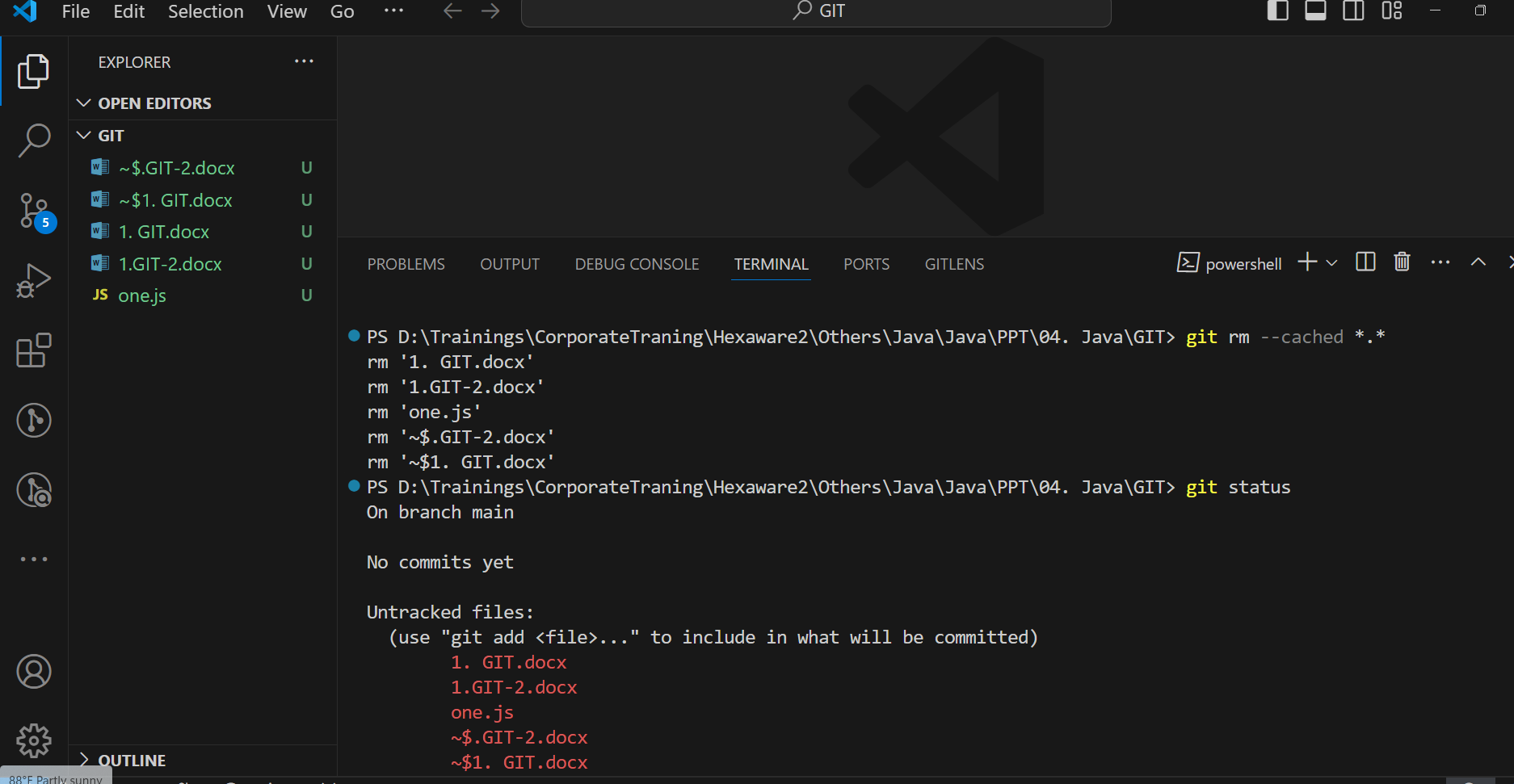
How to add One.java file from working directory to Staging area?

|  |
| --- |
| Use the git add <file-name> command to do that see the below image    One.java file is added to staging area.  Run the status |

Adding all the files at a time to staging area



To restore into working directory from staging area write the below command



How to see the no.of commits done till now?

|  |
| --- |
| This below command    i.e. no commits are done by you till now |

How we can commit?

|  |
| --- |
| If you Modified file then you have to add file to staging and then you can commit    If you see the status after coming it shows nothing to commit. |

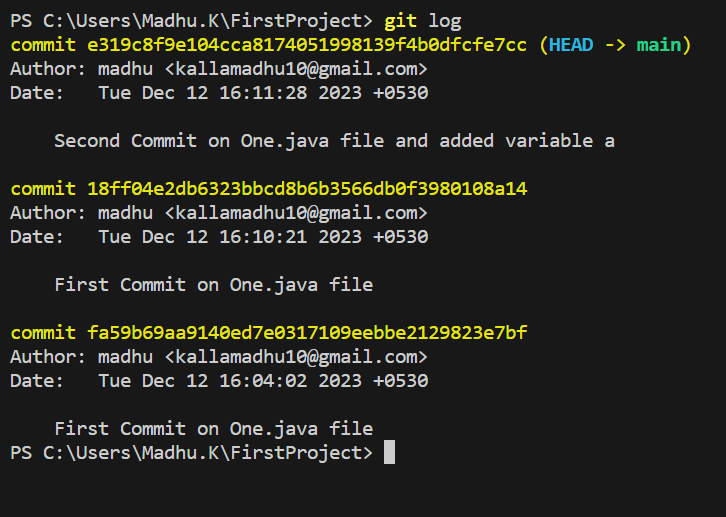
When the file will be tracked by git?

|  |
| --- |
| Note: Once commit is done you can say that the file is tracked by a git. |

What is Check sum?

|  |
| --- |
| A checksum is a value derived from the digital data of a file or set of data used to verify the integrity of the data.  It is a way of ensuring that the data has not been corrupted or tampered during transmission or storage.  Here's a simple explanation of how it works:  Generating a Checksum:  A checksum algorithm processes the data, producing a fixed-size string of characters (the checksum) based on the content of the data.  This process is designed in such a way that even a small change in the data will result in a significantly different checksum.  Verifying the Data:  The sender generates the checksum for the original data and sends both the data and the checksum to the receiver.  The receiver then recalculates the checksum based on the received data.  If the recalculated checksum matches the checksum received from the sender, it indicates that the data is likely intact.  Checksums are commonly used in computer science, telecommunications, and networking to detect errors in data transmission or storage. They provide a quick way to verify data integrity without having to compare the entire content of the data. If the checksums match, it suggests that the data is likely free from corruption. If they don't match, it signals a potential issue with the data.   1. When you commit a file or project you can see at least 7 characters that is called as checksum but it is total 40 characters long but it displays only 7 characters. 2. Every time when you commit a file in git, it will check any modifications are needed or not based on the new file checksum value and old checksum value if both checksum values are same which means no commits needed otherwise commit is needed. |

If you want to see the commit history run the below command?



U-> Untracke

A-> Added

Merging and Rebasing

<https://www.youtube.com/watch?v=e9lnsKot_SQ&t=71s>