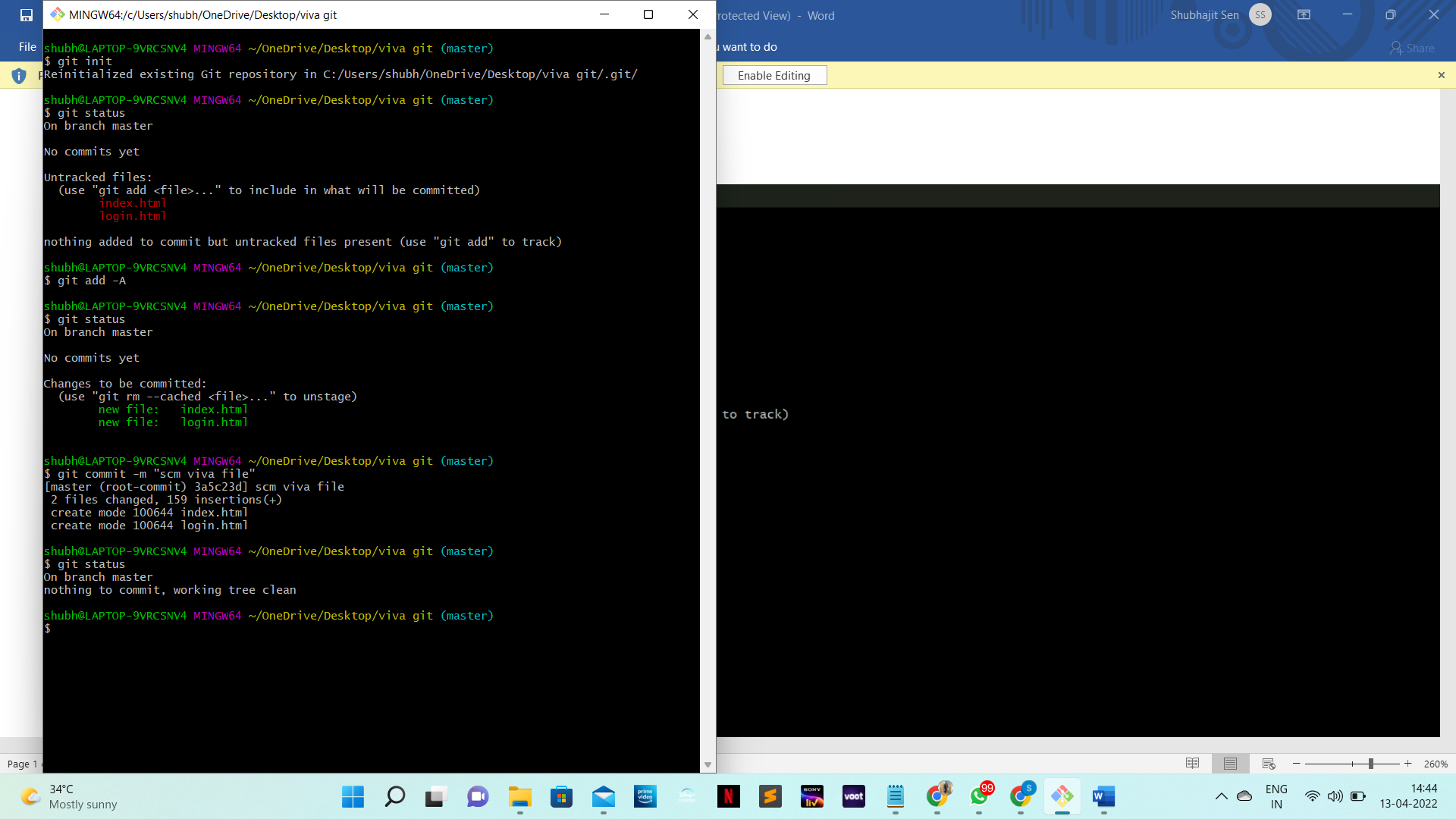
**SCM VIVA**

Q1. What do you mean by branches in git?

Sol. A branch in Git is a means to continue developing and coding a new feature or change to the programme while without impacting the main core of the project. We can also say that branches create another line of development in the project.

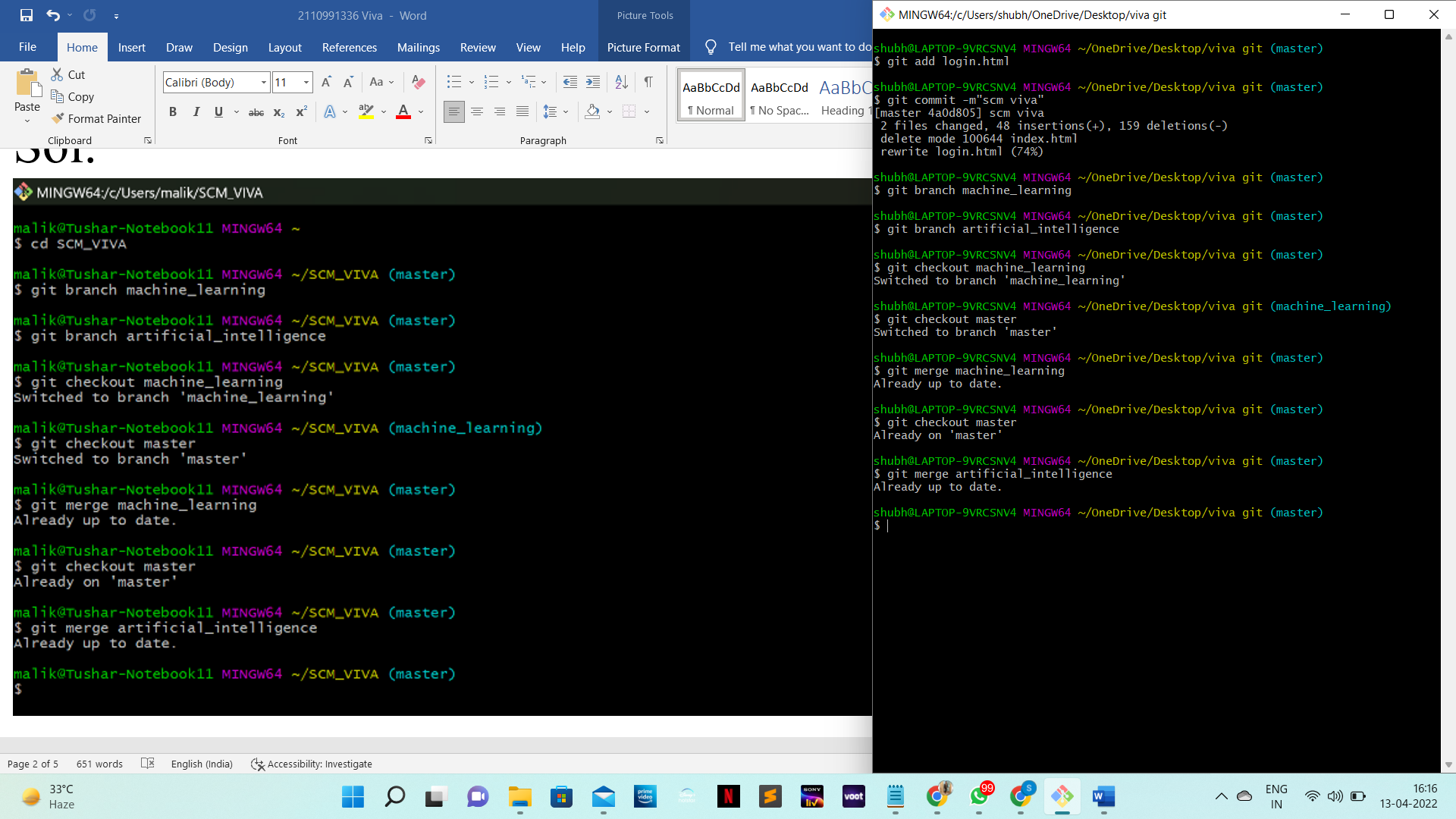
Q2. How to use commit?

Sol. 2.



Q3. Make a branch(machine learning and artificial intelligence) and merge it with some other branch.

Sol.

r

Q4. Explain the life cycle of git in detail.

## Sol. Stages in GIT Life Cycle

Files in a ***Git*** project have various stages like ***Creation, Modification, Refactoring***, and ***Deletion*** and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

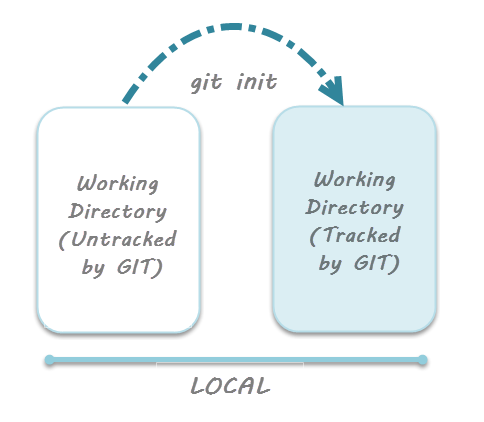
* ***Working directory***
* ***Staging area***
* ***Git directory***

These stages are the essence of Git. You get great flexibility in tracking the files due to these stages that files can reside in under Git. Let's understand each of these states one by one.

***Working Directory***

Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory.

***Working directory is the directory containing hidden .git folder.***



***Note***: ***git init*** - *Command to initialize a Git repository*

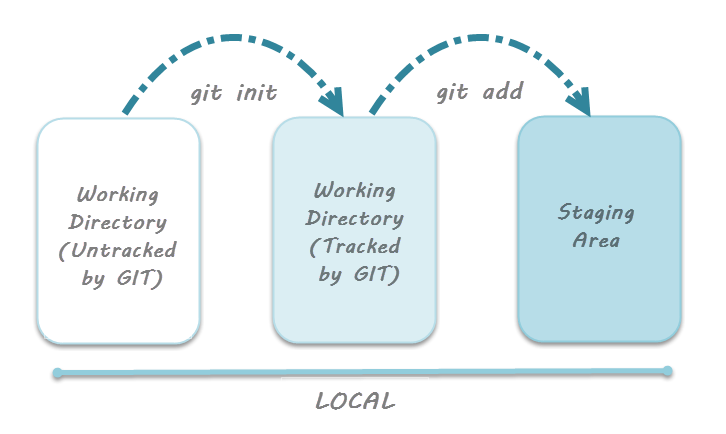
For sake of further discussion, let's assume that this directory is now tracked by Git. That is we've created a Git repository in this existing project directory. So, as discussed in the tutorial on [***Creation of Git Repository***](https://toolsqa.com/git/create-a-new-git-repository/), a hidden .git folder is initialized therein. In this state, Git is just aware of the files in the project. It doesn't track the files yet. To track the files, we've to commit these files by first adding the files to the staging area. This brings us to the next state in Git life-cycle.

***Staging Area***

While we're in the working directory, we select the files that have to be tracked by Git. ***Why do we need to this? Why don't we track everything in the project?*** That's because some files in the project like ***class files, log files, result files and temporary data files are dynamically generated***. It doesn't make sense to track the versions of these files. ***Whereas the source code files, data files, configuration files and other project artifacts contain the business logic of the application***. These files are to be tracked by Git are thus needs to be added to the staging area.

***In other words, staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.***

It's important to make a quick note of the term called ***indexing*** here. ***Indexing****is the process of adding files to the staging area. In other words, index constitutes of files added to the staging area*. This term will be explained again in the coming tutorial on ***Git terminologies***.

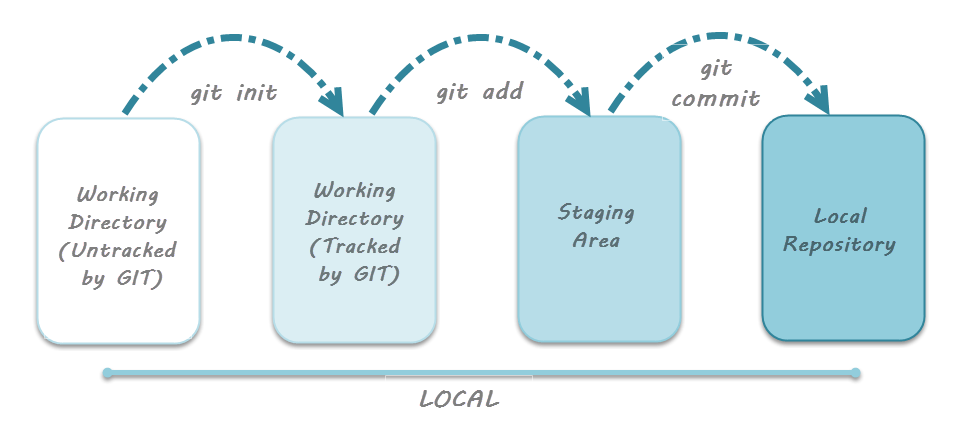


***Note***: ***git add*** - *Command to add files to staging area*.

***Git Directory***

Now that the files to be committed are grouped and ready in the staging area, we can commit these files. So, we commit this group of files along with a commit message explaining what is the commit about. Apart from commit message, this step also records the author and time of the commit. Now, a snapshot of the files in the commit is recorded by Git. The information related to this commit (*names of files committed, date and time of commit, author of commit, commit message*) is stored in the Git directory.

***Thus, Git directory is the database where metadata about project files' history will be tracked***.



***Note***: ***git commit -m"your message"*** - *Command to commit files to Git repository with message*.