**Source Control Management**

Source control is also known as version control. The source code control system (SCCS) is a complete system of commands that allows specified users to control and track changes made to an SCCS file.

SCCS files allow several versions of the same file to exist simultaneously, which can be helpful when developing a project requiring many versions of large files.

**Advantages:**

* Maintain multiple versions of code
* An ability to go back to any previous version.
* Developers can work in parallel.
* Audit traceability with clear picture of whom, which, when, where, and what are the changes.
* Synchronize the code.
* Copy/Merge/Undo the changes.
* Find out the difference between versions.
* Provides full back up without occupying much space.
* Review the history of the change.
* Capable for both small- and large-scale projects.
* Ability to share and work on the code across the globe.

**VCS: Version Control System**

VCS allows you to manage and keep track of all your source code, along with the evidence that all your changes are being stored in a repository.

* A version control system ensures that all the previous versions of all your code can be retrieved later and all changes to the code can be traced over time.
* It will help find what changes were made to which file, when why and by whom.
* It also explains what a file looked like on a specific date or at a specific release along with the ability to find the differences between any two versions of a file.
* A version control system provides the ability to work in parallel.

There are 2 types of Version Control Systems – the Centralized Version Control System (CVCS) and Distributed (DVCS).

### **Centralized Version Control:**

The concept of a centralized system is that it works on a Client-Server relationship. The repository is in one place and provides access to many clients.

The most common centralized version control systems are Concurrent Versions System (CVS), Perforce, and Subversion (SVN). There's also Microsoft Team Foundation Server (TFS), which is now known as Azure DevOps Server.

Git, the most common version control system is not a centralized VCS, but rather a distributed VCS.

Reference Link: <https://about.gitlab.com/topics/version-control/what-is-centralized-version-control-system/>

### **Distributed Version Control:**

On the contrary, in Distributed System, every user has a local copy of the repository in addition to the central repo on the server-side.

### **Benefits of Version Control Systems**

* A Version Control System works as a database of all your code and makes revisions instead of duplicating the files which helps you to save a lot of disk space.
* It keeps the history of all the files which gives you full traceability and audibility of what changes were made to which file, when, why, and by whom.
* It provides an ability to revert to the last revision or any previous stage as per requirement.
* It prevents the risk of losing functioning code or breaking test scripts by overwriting files as you can always take out the last working code at any point in time.
* It helps you to identify the difference in any set of files, compare the revisions and merge the changes as per requirements.
* It provides you with the ability to have entirely independent code versions if you prefer to keep different development code. Once you are ready, you can merge the files to create a final working version.
* It provides an ability to work as a distributed team with full collaboration across the globe and saves time and additional effort for everyone. There is no need to wait for others to complete the work.

**Git Fundamentals:**

Git Bash is an application for Microsoft Windows environments which provides an emulation layer for a Git command line experience. Bash is an acronym for Bourne Again Shell. A shell is a terminal application used to interface with an operating system through written commands. Bash is a popular default shell on Linux and macOS. Git Bash is a package that installs Bash, some common bash utilities, and Git on a Windows operating system.

### **The Three States**

Git has three main states that your files can reside in: modified, staged, and committed:

* Modified means that you have changed the file but have not committed it to your database yet.
* Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
* Committed means that the data is safely stored in your local database.

Download git from <https://git-scm.com/>

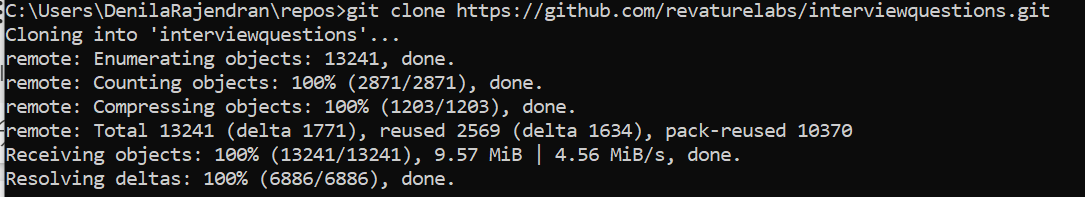
Open Command line and try the below codes

>git

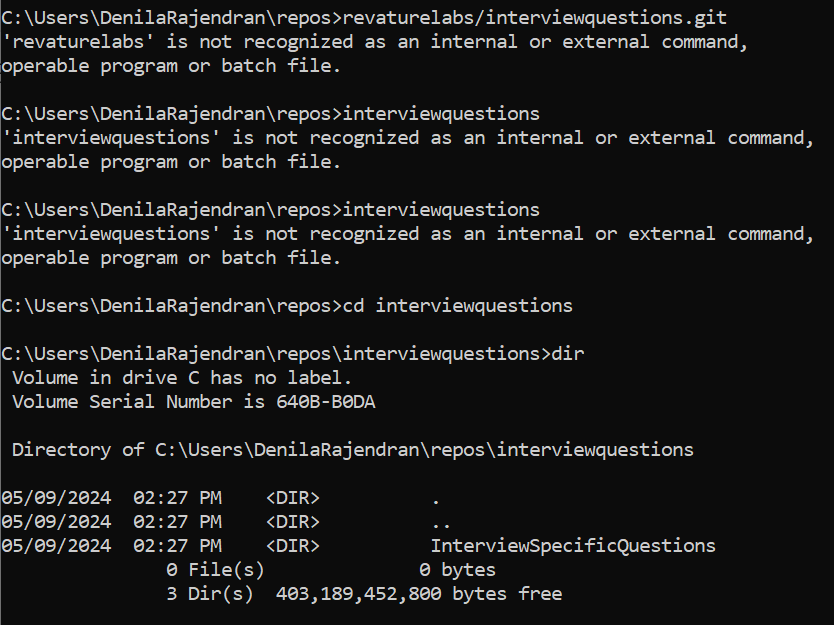
>mkdir repos

>cd repos

>git clone (HTTPS Link)



>cd (link name)



>dir

>git pull #to update

**Initializing a Repository:**

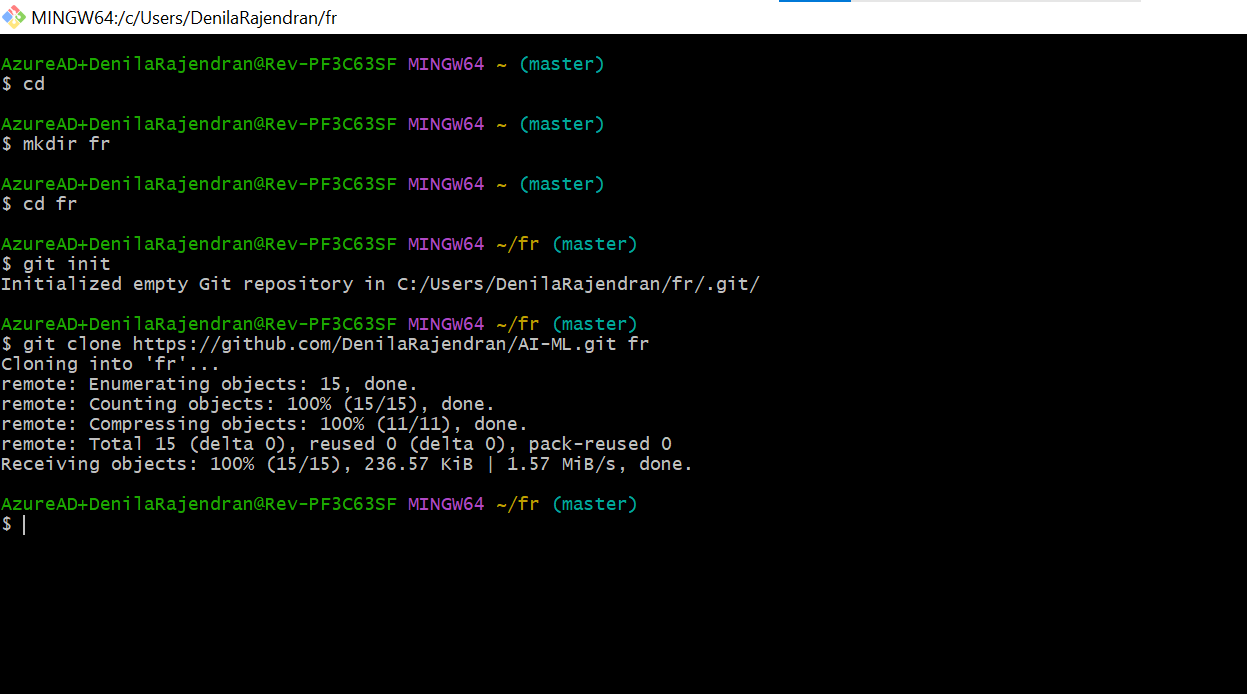
### **What is a Git Repository?**

A Git repository tracks and saves the history of all changes made to the files in a Git project. It saves this data in a directory called .git, also known as the repository folder.

Here are some reasons for using a Git repository:

* **Cloud repositories**
  + It is generally more secure.
  + It is easier to work collaboratively. Any team member can download the latest version of the repository from any machine.
  + It is cheaper than a traditional server.
* **Distributed file system**
  + Git is distributed, meaning that every local copy of the global repository is a fully working copy.
  + In case there is a problem with the server and the global repository is corrupted or lost, any local copy can recreate the full history.
  + In a centralized version control system, the global server contains all changes in the project and the local copies are just light versions of it. If the server goes down, you lose all the history.
* **Perfect to work with others**
  + Git is designed for creating projects where many contributors develop software in parallel.
* **Good documentation**
  + Git has been around for many years now and it's really easy to find good documentation.
* **Branches allows for simultaneous code versions**
  + Branches are one of the best features of a version control software and are used to develop in parallel to the main repository.
  + A branch is a fork of the main code to develop a new feature.
* **Encourages code reviews**
  + Code reviews are a good practice that every developer team should follow.
  + Git facilitates code reviews with an operation called pull request.
* **Simpler to roll back mistakes**
  + Every commit is referenced with a hash that uniquely identifies it, see this example.
  + With git, we can revert to any past commit and fix a mistake.
* **It's the current de facto open-source umbrella**
  + If you want to develop open-source code, the biggest repository is GitHub.
  + Here you can find the most popular repositories in Github, it includes bootstrap, react, d3, tensorflow, angular, etc.

**Implementation**

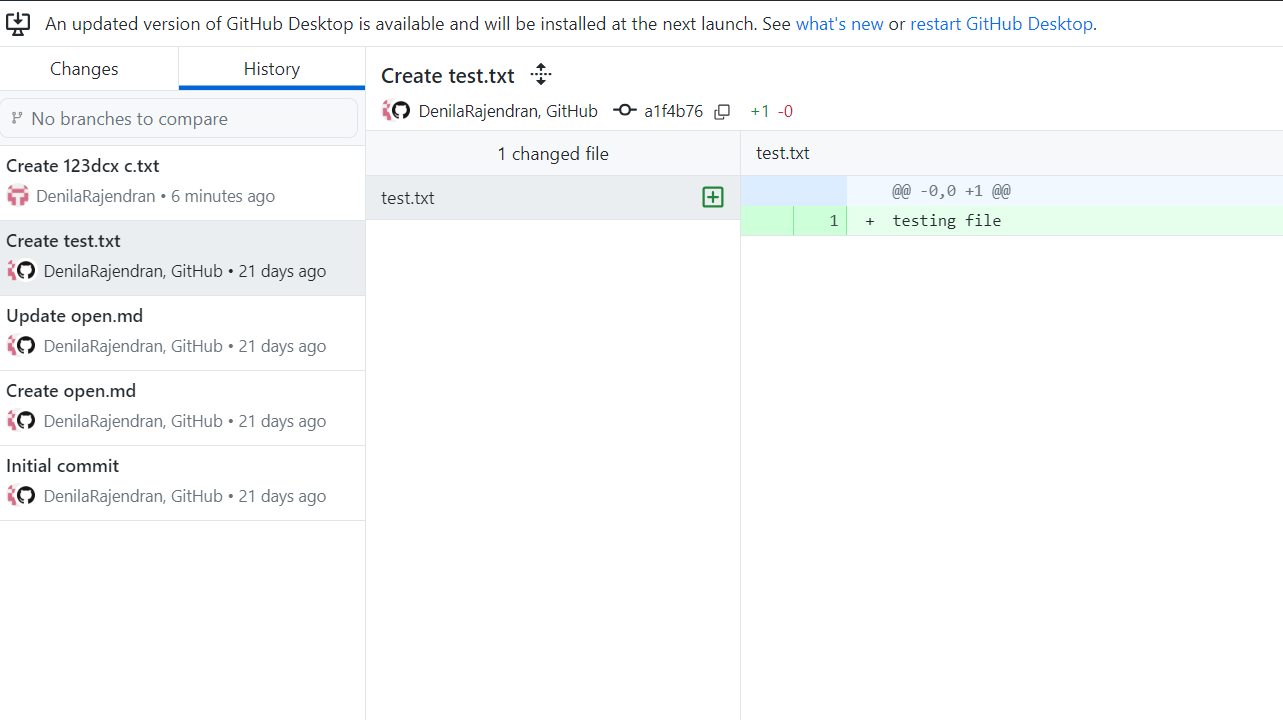


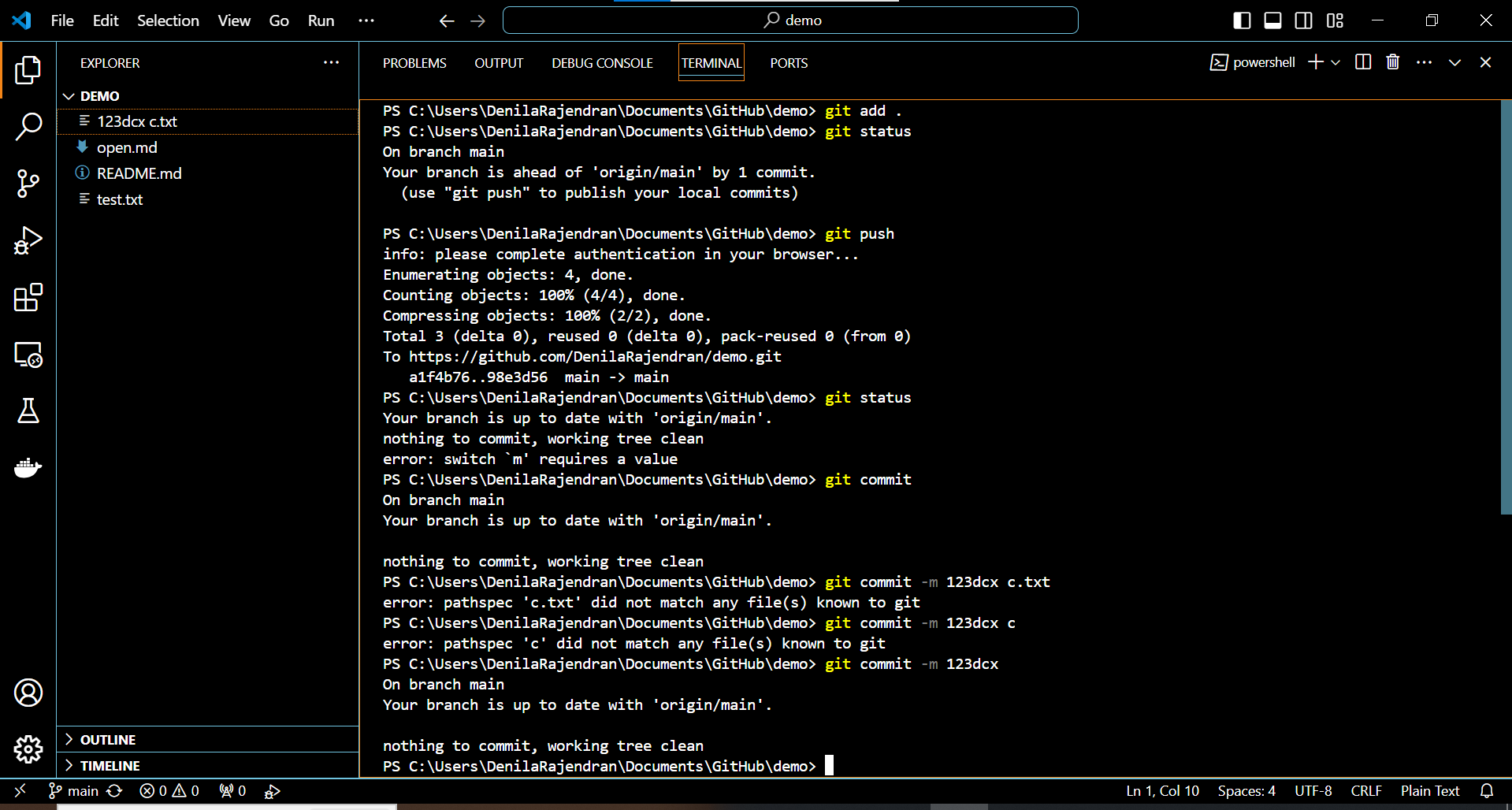
**Pushing to a remote repository**

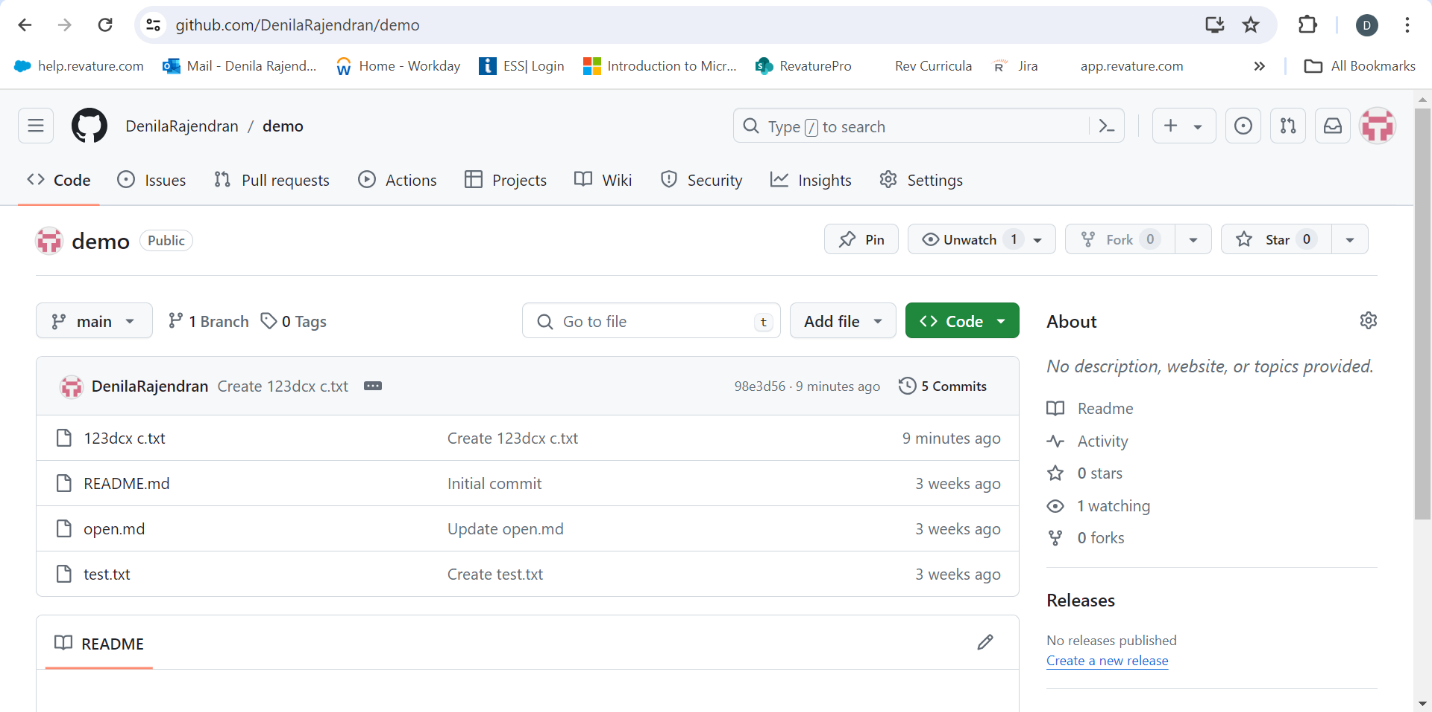
The git push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo. The fetch and push commands are counterparts; fetch imports remote commits to the local repository while push exports local commits to the remote repository.

**Push all of your local branches to the specified remote.**

Git push is one component of many used in the overall Git "syncing" process. The syncing commands operate on remote branches which are configured using the git remote command. git push can be considered and 'upload' command whereas, git fetch and git pull can be thought of as 'download' commands. Once change sets have been moved via a download or upload a git merge may be performed at the destination to integrate the changes.





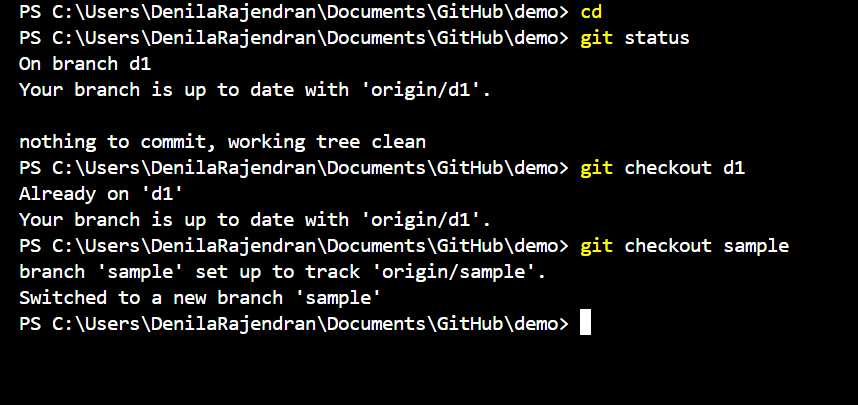


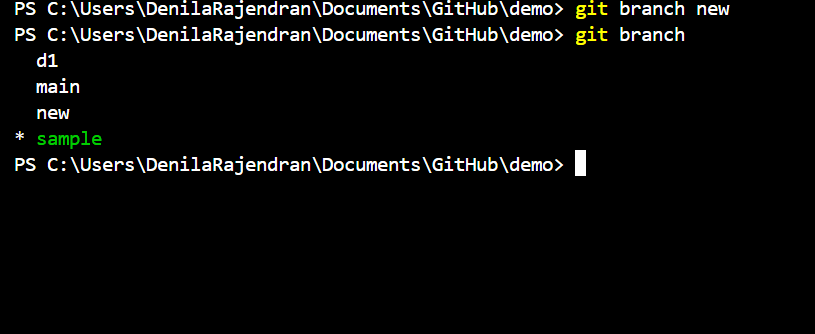
Reference link: <https://www.datacamp.com/tutorial/git-push-pull>

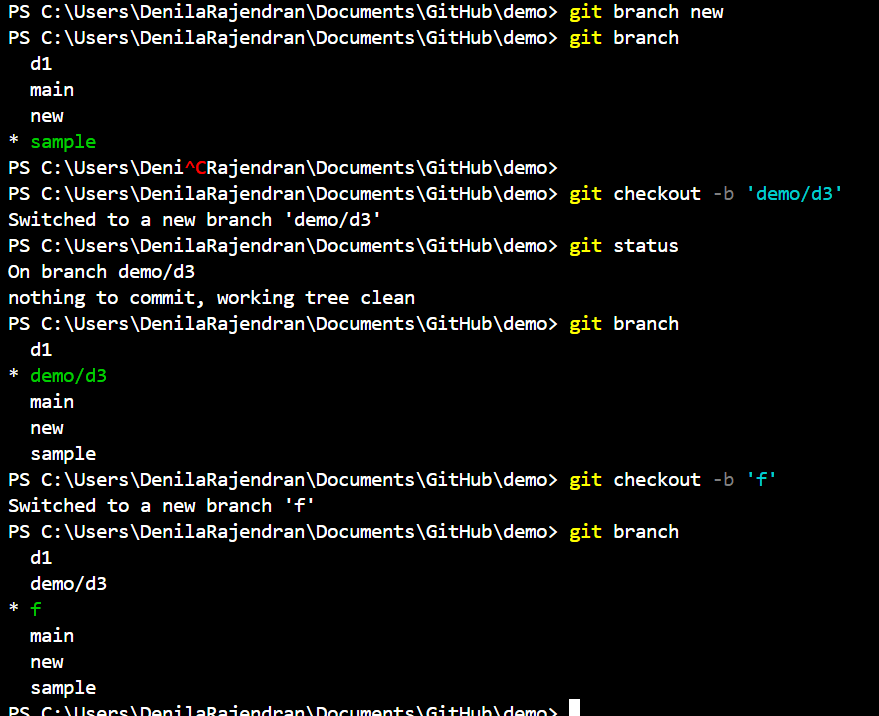
**Git commit Branch Merge Push Pull**

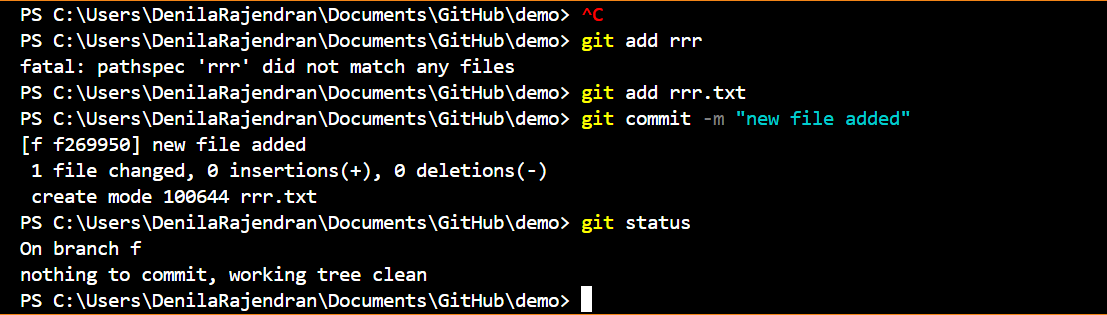
Merging is Git's way of putting a forked history back together again. The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.

Git merge will combine multiple sequences of commits into one unified history. In the most frequent use cases, git merge is used to combine two branches.









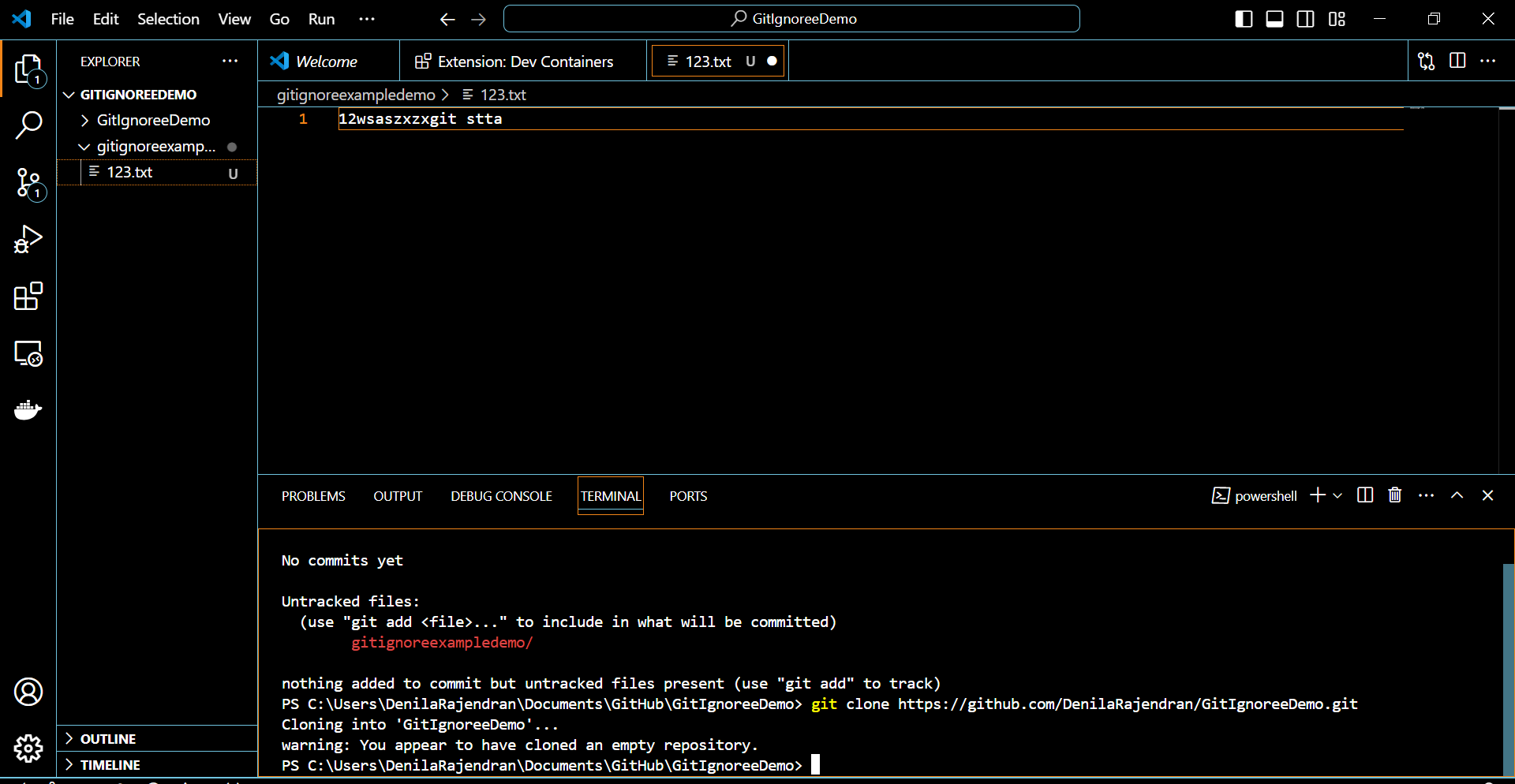
Reference link: <https://www.varonis.com/blog/git-branching#:~:text=To%20merge%20branches%20locally%2C%20use,branch%20into%20the%20main%20branch>

**Gitignore**

The .gitignore file is a text file that tells Git which files or folders to ignore in a project. Files already tracked by Git are not affected. Each line in a gitignore file specifies a pattern. When deciding whether to ignore a path, Git normally checks gitignore patterns from multiple sources, with the following order of precedence, from highest to lowest.

Whatever files and folders from any project you want to ignore those files you will place in the .gitignore file by using the following patterns.

* \* is used as a wildcard match.
* / is used to ignore pathnames relative to the .gitignore file.
* # is used to add comments to a .gitignore file.
* \*\* can be used to match any number of directories.
* ! to negate a file that would be ignored.



***Thank you***