### **Basic Git Questions**

1. What is Git and why is it used?

Git is the most commonly used version control system. Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to. Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

1. Explain the difference between Git and GitHub.

Git is a version control system that lets you manage and keep track of your source code history. GitHub is a cloud-based hosting service that lets you manage Git repositories. If you have open-source projects that use Git, then GitHub is designed to help you better manage them.

1. How do you install Git on your machine?

Step 1: Navigate to the latest Git for Windows installer and download the latest version.

Step 2: Once the installer has started, follow the instructions as provided in the Git Setup wizard screen until the installation is complete.

Step 3: Open the windows command prompt (or Git Bash if you selected not to use the standard Git Windows Command Prompt during the Git installation).

Step 4: Type git version to verify Git was installed.

1. How do you configure your username and email in Git?

Username and email are configured by using below commands:

git config --global user. email "you@example.com"

git config --global user.name "your username"

1. What is a repository in Git?

The repositories of Git act as essential places for storing the files with maintaining the versions of development. By using GitHub repositories developers can organize, monitor, and save their changes of code to their projects in remote environments. The files in the GitHub repository are imported from the repository into the local server of the user for further updates and modifications in the content of the file.

1. How do you create a new Git repository?

To create a new repo, you'll use the git init command. git init is a one-time command you use during the initial setup of a new repo. Executing this command will create a new . git subdirectory in your current working directory.

1. How do you clone a repository from GitHub?

The git clone command is used to create a copy of a specific repository or branch within a repository.

When you clone a repository, you don't get one file, as you may in other centralized version control systems. By cloning with Git, you get the entire repository – all files, all branches, and all commits.

Cloning a repository is typically only done once, at the beginning of your interaction with a project. Once a repository already exists on a remote, like on GitHub, then you would clone that repository so you could interact with it locally. Once you have cloned a repository, you won't need to clone it again to do regular development.

1. What is the purpose of the .gitignore file?

gitignore file is used in a git repository to ignore the files and directories which are unnecessary to project this will be ignored by the git once the changes as been committed to the Remote repository.

1. How do you check the status of your working directory in Git?

The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show you any information regarding the committed project history.

1. How do you add files to the staging area in Git?

git add' command is used to stage files. To stage all files use 'git add . ', to stage a specific file use 'git add [filename]', and to stage a folder use 'git add [folderpath]'. If you mistakenly stage a file, you can unstage it using 'git reset HEAD [filename]'.

### **Intermediate Git Questions**

1. Explain the concept of commits in Git.

Similar to saving a file that's been edited, a commit records changes to one or more files in your branch. Git assigns each commit a unique ID, called a SHA or hash, that identifies:

* The specific changes
* When the changes were made
* Who created the changes

When you make a commit, you must include a commit message that briefly describes the changes.

If the repository you are committing to has compulsory commit signoffs enabled, and you are committing via the web interface, you will automatically sign off on the commit as part of the commit process.

1. How do you create a new commit in Git?

Using git commit -m <message>, you can create a new commit with the staged changes and the specified <message>. If you omit the -m option, Git will open the default text editor to enter the commit message.

1. What is the purpose of the git log command?

The git log command displays committed snapshots. It lets you list the project history, filter it, and search for specific changes. While git status lets you inspect the working directory and the staging area, git log only operates on the committed history.

1. How do you view the history of commits in a repository?

The most basic and powerful tool to do this is the git log command. By default, with no arguments, git log lists the commits made in that repository in reverse chronological order; that is, the most recent commits show up first.

1. How do you view the changes made in a commit?

To see the changes in a specific Git commit, you can use the git show command. This command displays the details of a commit, including the commit message, author, date, and the diff of changes introduced in that commit.

1. What is branching in Git and why is it useful?

In Git, branches are a part of your everyday development process. Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes.

1. How do you create a new branch in Git?

Git checkout works hand-in-hand with git branch. The git branch command can be used to create a new branch. When you want to start a new feature, you create a new branch off main using git branch new\_branch. Once created you can then use git checkout new\_branch to switch to that branch.

1. How do you switch between branches in Git?

The easiest way to switch branch on Git is to use the “git checkout” command and specify the name of the branch you want to switch to. If the destination branch does not exist, you have to append the “-b” option, otherwise you won't be able to switch to that branch.

1. What is the difference between git merge and git rebase?

Git Merge lets you merge different Git branches. Git Rebase allows you to integrate the changes from one branch into another. Git Merge logs show you the complete history of commit merging. Git Rebase logs are linear.

1. How do you resolve merge conflicts in Git?

Step 1: The easiest way to resolve a conflicted file is to open it and make any necessary changes.

Step 2: After editing the file, we can use the git add a command to stage the new merged content.

Step 3: The final step is to create a new commit with the help of the git commit command.

### **Advanced Git Questions**

1. What is the purpose of the git stash command?

git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on.

1. How do you apply stashed changes in Git?

To retrieve changes out of the stash and apply them to the current branch you're on, you have two options:

1. git stash apply STASH-NAME applies the changes and leaves a copy in the stash.
2. git stash pop STASH-NAME applies the changes and removes the files from the stash.
3. What is the purpose of the git tag command?

Tagging is traditionally used to create semantic version number identifier tags that correspond to software release cycles. The git tag command is the primary driver of tag: creation, modification and deletion. There are two types of tags; annotated and lightweight.

1. How do you create and push tags to a remote repository?

You can push a single tag or all tags to a remote repository using the git push command with the name of the remote and the name of the tag, or the --tags option to push all tags at once.

1. Explain the concept of remote repositories in Git.

Remote repositories are versions of your project that are hosted on the Internet or network somewhere. You can have several of them, each of which generally is either read-only or read/write for you.

A remote URL is Git's fancy way of saying "the place where your code is stored." That URL could be your repository on GitHub, or another user's fork, or even on a completely different server.

You can only push to two types of URL addresses:

* An HTTPS URL like https://github.com/user/repo.git
* An SSH URL, like git@github.com:user/repo.git

Git associates a remote URL with a name, and your default remote is usually called origin.

1. How do you add a remote repository in Git?

To add a new remote, use the git remote add command on the terminal, in the directory your repository is stored at. The git remote add command takes two arguments: A remote name, for example, origin.

1. How do you push changes to a remote repository?

To push the commit from the local repo to your remote repositories, run git push -u remote-name branch-name where remote-name is the nickname the local repo uses for the remote repositories and branch-name is the name of the branch to push to the repository.

1. How do you pull changes from a remote repository?

This is done by the use of git pull command. This command updates the local repository immediately after its execution. git pull command is a combination of two other commands which are git fetch and git merge.

1. What is the purpose of the git fetch command?

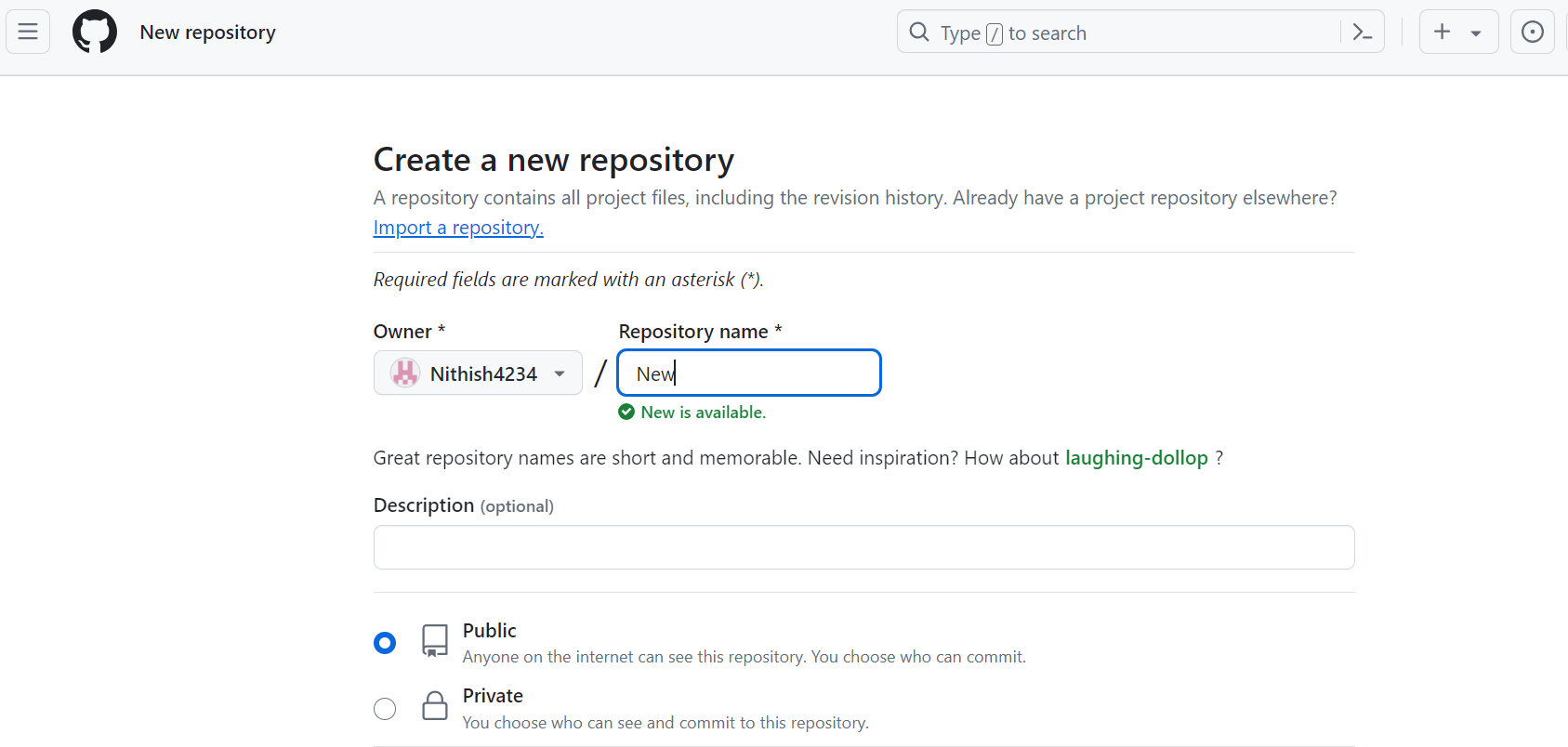
git fetch is a primary command used to download contents from a remote repository. git fetch is used in conjunction with git remote, git branch, git checkout, and git reset to update a local repository to the state of a remote.

1. How do you delete a branch in Git?

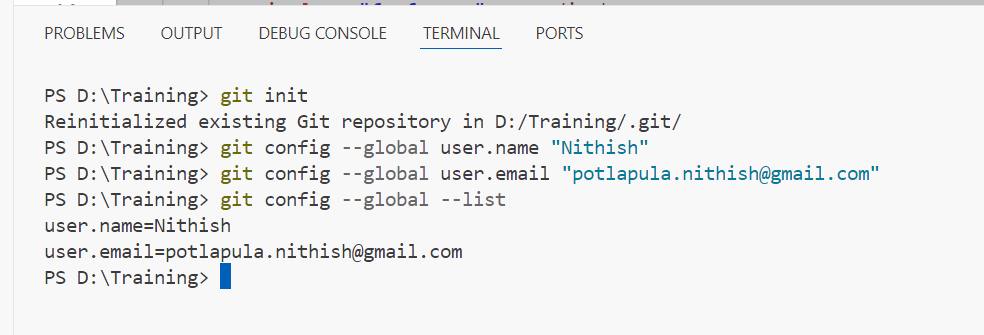
Delete a branch with git branch -d <branch>. The -d option will delete the branch only if it has already been pushed and merged with the remote branch. Use -D instead if you want to force the branch to be deleted, even if it hasn't been pushed or merged yet. The branch is now deleted locally.

### **Git Exercises**

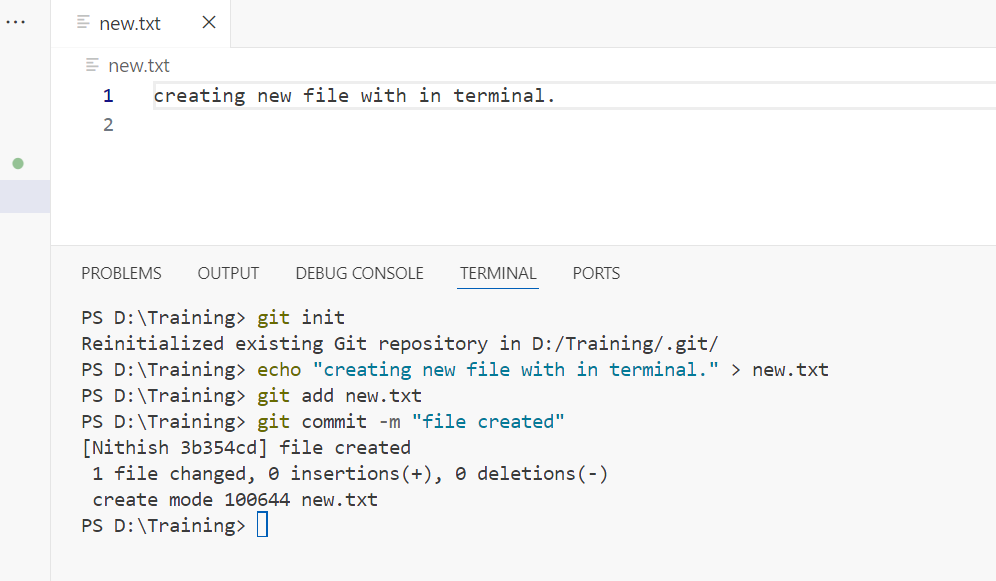
1. Create a new Git repository and configure your username and email.



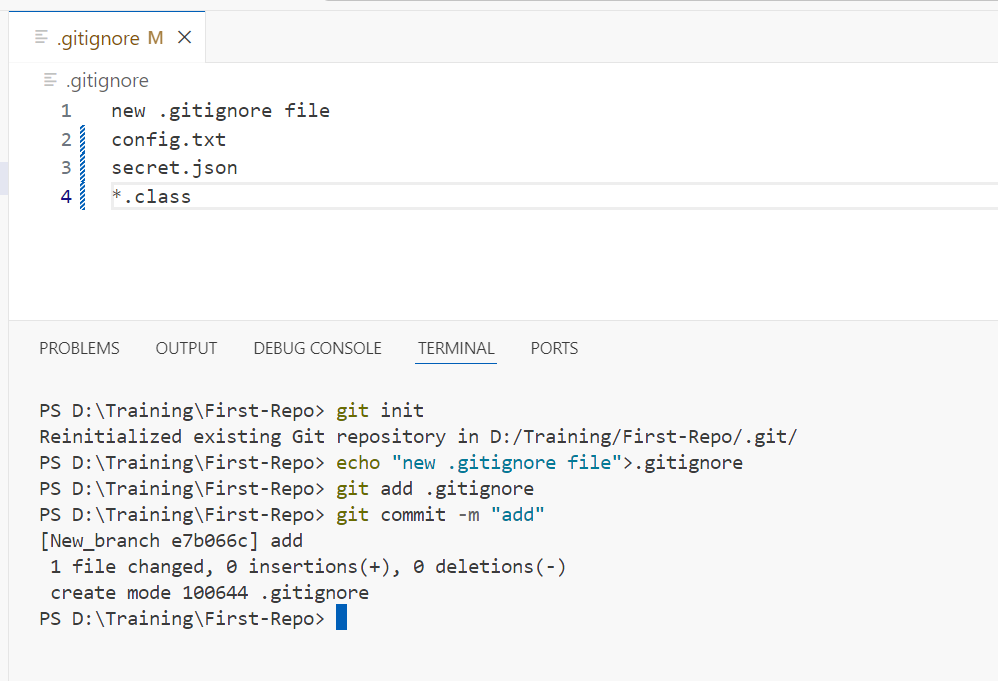




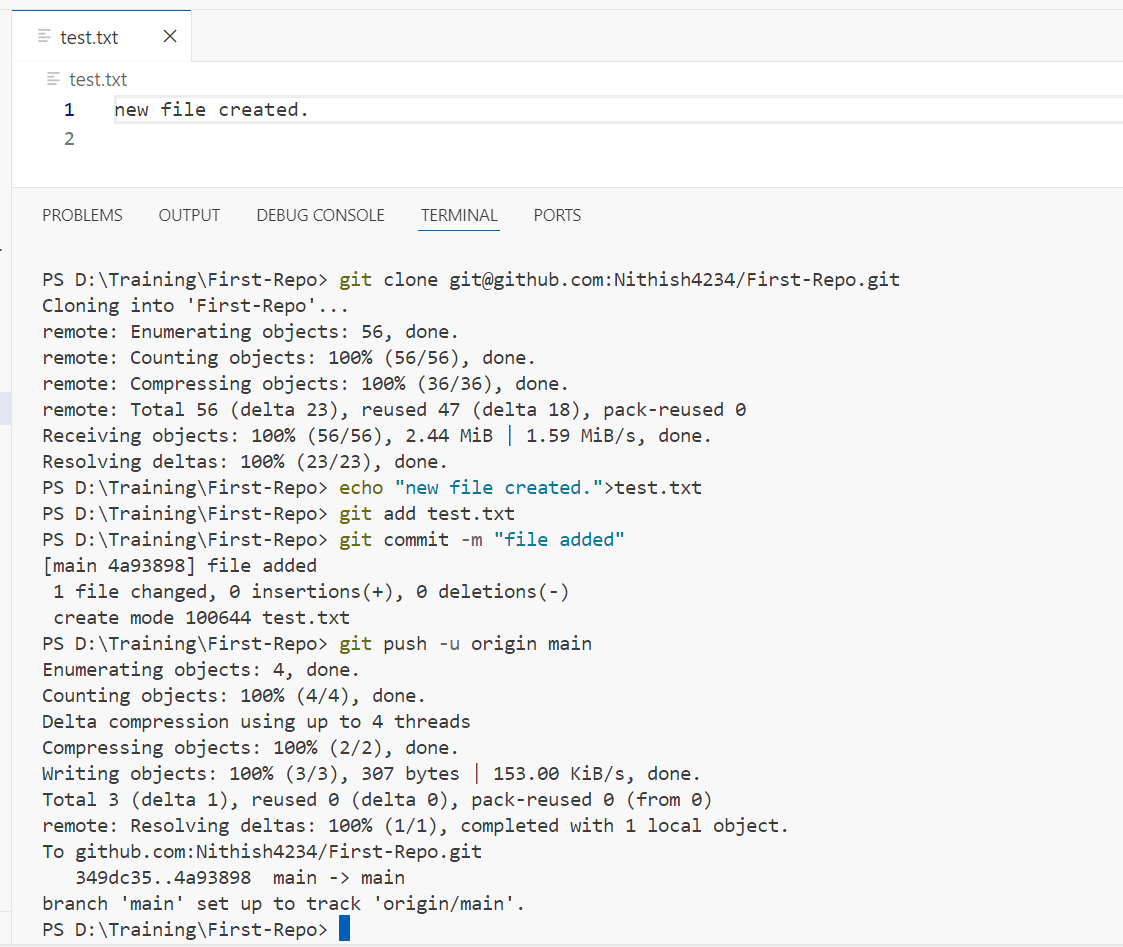
1. Create a file, add some content to it, and commit the changes.



1. Create a .gitignore file and add rules to ignore specific files and directories.

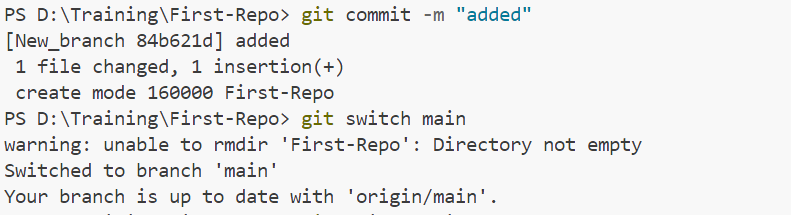


1. Clone an existing repository from GitHub and make some changes.

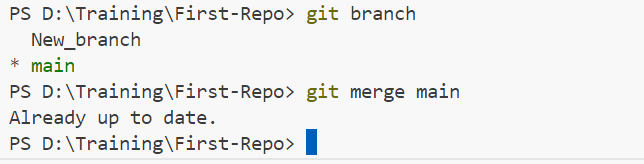


1. Create a new branch, make some changes, and switch back to the main branch.

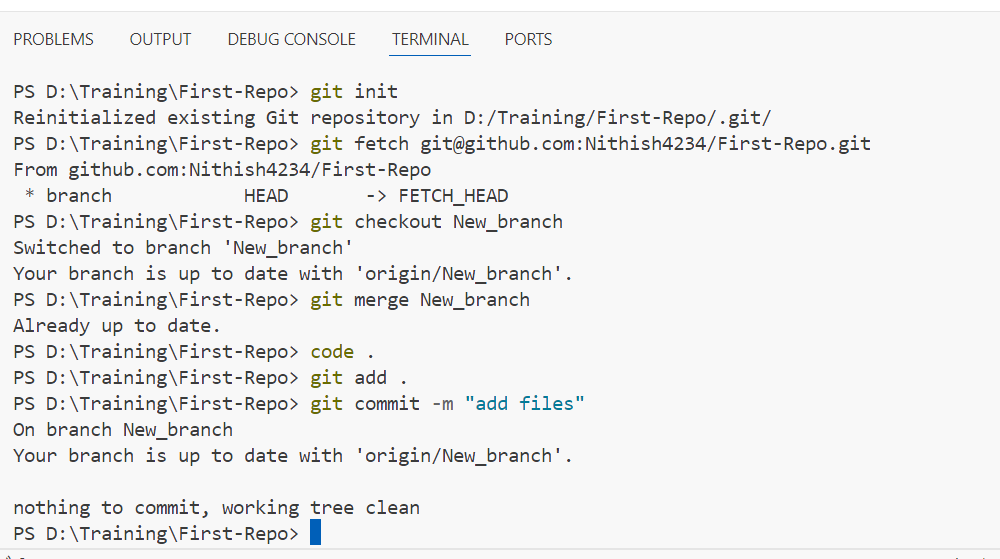




1. Merge changes from a feature branch into the main branch.



1. Resolve a merge conflict between two branches.



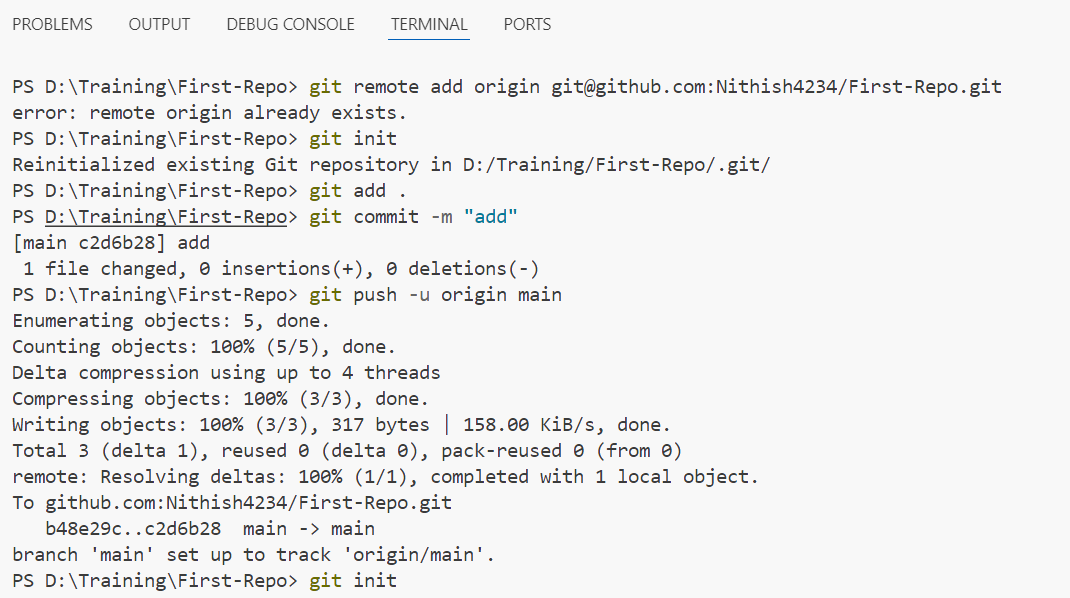
1. Use git stash to save your work and then apply the stashed changes.

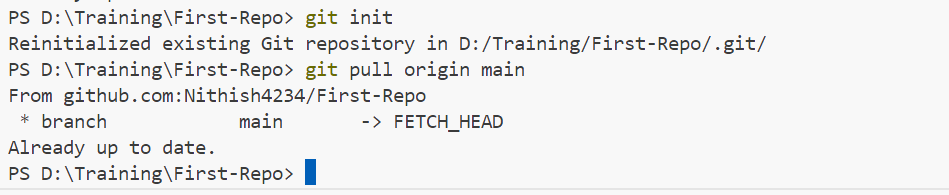


1. Create a tag for a specific commit and push the tag to the remote repository.

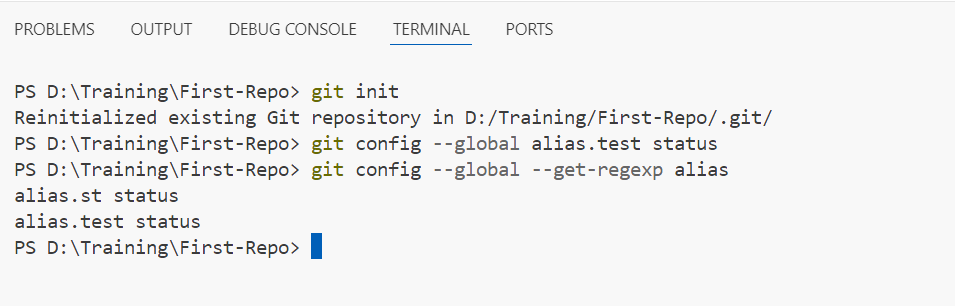


1. Add a remote repository, push your local changes, and pull changes from the remote repository.





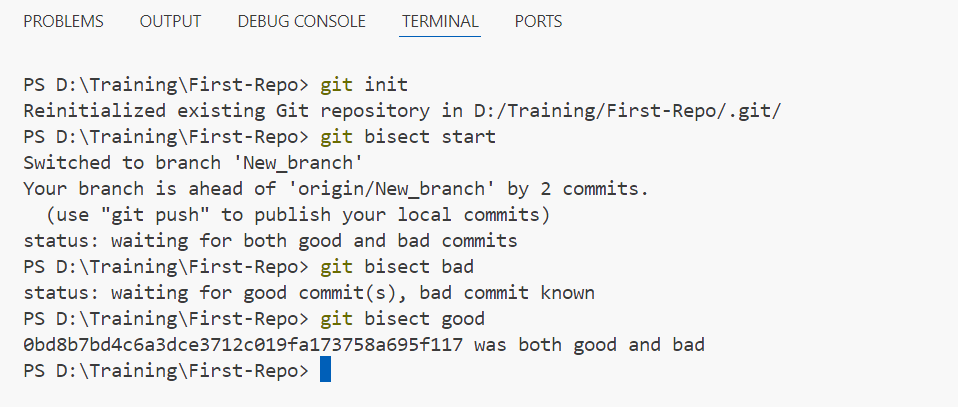
1. Create a Git alias for a commonly used command.



1. View the commit history with a graph representation.



1. Revert a commit and understand the differences between git revert and git reset.
2. Squash multiple commits into a single commit using git rebase.
3. Use git bisect to find the commit that introduced a bug.



1. Set up a Git hook to run a script before or after certain Git events.
2. Use git cherry-pick to apply a commit from one branch to another.
3. Learn how to use git blame to find out who made specific changes to a file.



1. Configure and use a Git GUI client.
2. Create a fork of a repository on GitHub, make changes, and open a pull request.