### **Basic Git Questions**

1. What is Git and why is it used?

**Ans:** Git is a distributed version control system that allows programmers and developers to collaborate on projects effectively.

**Version control system:** Git helps manage source code and other files in a project. It allows you to track modifications, revert to older versions, and work collaboratively.

1. Explain the difference between Git and GitHub.

**Ans: Git:**

* Git is a distributed version control system (VCS) designed for tracking changes in source code during software development.
* It allows developers to coordinate work, track modifications, and manage code history.

**GitHub:**

* GitHub is a web-based git repository hosting service that extends Git’s functionality.
* It provides a platform for hosting git repositories, collaboration, and centralized source code management.

1. How do you install Git on your machine?

**Ans:** Download Git

* Visit the website: Git Downloads.
* Download the latest version of Git for windows
* Once download is completed, run the installation file.
* Make sure to run it with Administrator rights.
* The Git setup wizard will guide you through the installation process.
* Choose an appropriate installation location.
* Open the terminal and type git version and enter
* If Git was installed successfully, it will display the installed version

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

**Ans:**

* git config user.name "Your Name"
* git config user.email [youremail@yourdomain.com](mailto:youremail@yourdomain.com)
* git config --list

1. What is a repository in Git?

**Ans:** A repository is central location where you to store your project’s files, code, and their entire history.

* **Local repo:** The .git directory within the project folder on your local machine.
* **Remote Repo:** A copy of your local repository hosted on a remote server.
* **Ex:** GitHub, Gitlab, Bitbucket.

1. How do you create a new Git repository?

* Create Folder: Create a folder on your local machine where you want to store your local files.
* Open terminal.
* Initialize: Git init.
* Add the file and commit changes: git add.
* Git commit -m “hello”.
* Push the repo to remote server:
* Git remote add origin <repo url>
* Push your commits to the remote repo.
* Git push -u origin master/main.

1. How do you clone a repository from GitHub?

* Go to the main page of the repo on GitHub.
* Copy the URL of the repo.
* Open terminal of your local machine.
* Clone the repo: git clone <repo URL>.

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

* The .gitignore file serves crucial part in git repository.
* It specifies intentionally untracked files that git should ignore.
* The main reason for the usage of .gitignore file is:

1. Prevent Tracking.
2. Clean status.
3. Avoid accidents.
4. Pattern matching.
5. Local configuration.
6. How do you check the status of your working directory in Git?

* Open terminal or cmd.
* Use cd command to change the directory contain your repo.
* Run the status command: git status

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

* Open terminal or cmd.
* Use cd command to change the directory contain your repo.
* To add a specific file to the staging area,
* Git add <filename>
* Git add.
* Git add --all

### **Intermediate Git Questions**

1. Explain the concept of commits in Git.

**Ans:** A commit represents a overview of projects working directory at a specific point of time.

* A commit captures the state of your project at the moment of the commit. It includes all the files and changes that have been staged for commit.
* Each commit is identified by a unique SHA-1 has, that ensures the integrity and uniqueness of the commit.
* Once committed, they are considered “safe” versions of your project.
* Commits are always made to the local repo first, which allow you to work independently of others and doesn’t require interaction with a remote repo until you are ready.

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

* Open terminal or cmd.
* Use command cd to change the dir containing your repo.
* Git add <file name>
* Git commit -m “commit message”

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

* Git Log command is used to view the commit history within a git repo.
* It is a powerful tool that allows you to see a list of all the commits made to the repo, in reverse chronological order.

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

* Open terminal or cmd.
* Use cd command.
* Run the log command: git log.
* Git log will display the commit history, listing each commit .
* Git log --graph –oneline –decorate will show you a graph of commits, one commit per line, with decorations such as branch names and tags.

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

* Git show <commit hash> : will display the commit details.
* Git diff commit ~commit: will show the difference between the commit’s ancestor and the commit itself.
* Git show –name—only <commit-id>:

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

* Branching in Git is used to allow users to diverge from the main lines of development and work on new features, bugfixes, or experiments in parallel without affecting the stable version of the project.
* A branch in git is essentially a pinter to snapshot of your changes. When you create a branch, you are make a copy of the code at that point in time.
* The main branch in git is usally called as main or master, which represents the offical working version of the project.
* It is useful because of several reasons:

1. Isolation.
2. Collaboration.
3. Integration.
4. Flexibility.
5. How do you create a new branch in Git?

* Branch can be create in four ways.

1. Create a new branch on the current head: git checkout -b branchname.
2. Create a new branch on a existing branch:

git checkout existing-branch

git checkout -b branchname.

1. Create a branch from a specific commit: git checkout -b branchname

commit-hash

1. Create a new branch from a specific tag: git checkout -b branchname

Tag-name

1. How do you switch between branches in Git?

**Ans:** ToSwitch to an existing branch:

**Git checkout branch-**name. Or

**Git switch branch-**name

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

**Ans:**

* **Git merge:**
* Combines the histories of two branches together.
* Creates a new “merge commit” that has two parent commits, one from each branch.
* Preserves the original history and chronological order of commits.
* Can result in a non-linear history with “merge commits” that might complicate the project history.
* **Git Rebase:**
* Reapplies commits on top of another base tip.
* Moves the entire branch to begin on the tip of the other branch.
* Creates a linear history by rewriting the commit history.
* Can simplify the project history but requires force pushing if the branch is already pushed to a remote repo.

1. How do you resolve merge conflicts in Git?

**Ans:**

* Identify the conflicted files: git status
* Edit the conflicted files in terminal.
* Resolve the conflicts by integrating the changes from both the branches.
* After editing save the file and mark as resolved. Git add <file>
* Finalize the merge : git commit

### **Advanced Git Questions**

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

**Ans:** The git stash command is used for temporarily Strong all the modified tracked files and staged changes in your working directory so you can have a clean working directory.

* The purpose of this command is to save changes, switch tasks and work on multiple branches.

1. How do you apply stashed changes in Git?

* This command apply changes from the most recent stash to your current working directory but keeps the stash in your list for future use: git stash apply.
* If you have multiple stashes use: git stash apply stash@{n}
* Add stash and additionally removes the applied stash: git stash pop
* Apply stash and remove it from the list: git stash pop stash@{n}.

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

**Ans: git tag:** is used to mark specific points in repo history as being important. Typically, tags are used to mark release points like v1.0, v2.0, v3.0.

* Git tags contains the key points :

1. Mark release points.
2. Reference specific commits.
3. Non changing reference.
4. Annotated tags.
5. Lightweight tags.
6. How do you create and push tags to a remote repository?

* Create a tag: git tag <tag name>.
* For an annotated tag: git tag -a <tag name> -m “message”.
* Push tag to repo: git push origin <tag name>
* To push all tags: git push origin –tags.

1. Explain the concept of remote repositories in Git.

**Ans:** A remote repo is a common database of your project that is hosted on the network or internet, rather than on your local machine.

* It acts as the central hub.
* It allows multiple developers to push and pull from the remote repo, enabling collaborative work on projects.
* It serves as a backup of your project.
* Developers can synchronize their local repositories help in managing branches.
* They facilitate the use of pull and push requests.

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

* Open the terminal or cmd.
* Use git remote add origin <URL>.
* Can verify: git remote -v.

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

* Open the terminal or cmd
* Navigate to the local repo.
* Stage the changes: git add . or git add <filename>
* Commit: git commit -m “message”.
* Push the changes to the remote repo: git push origin <branch name>.

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

* Open terminal.
* Navigate to the local repo.
* Fetch the changes from current repo to local repo: git fetch origin.
* Pull the changes to update: git pull origin <branch name>

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

**Ans: git fetch:** git fetch is to update your local copy of a repo with changes from remote repo.

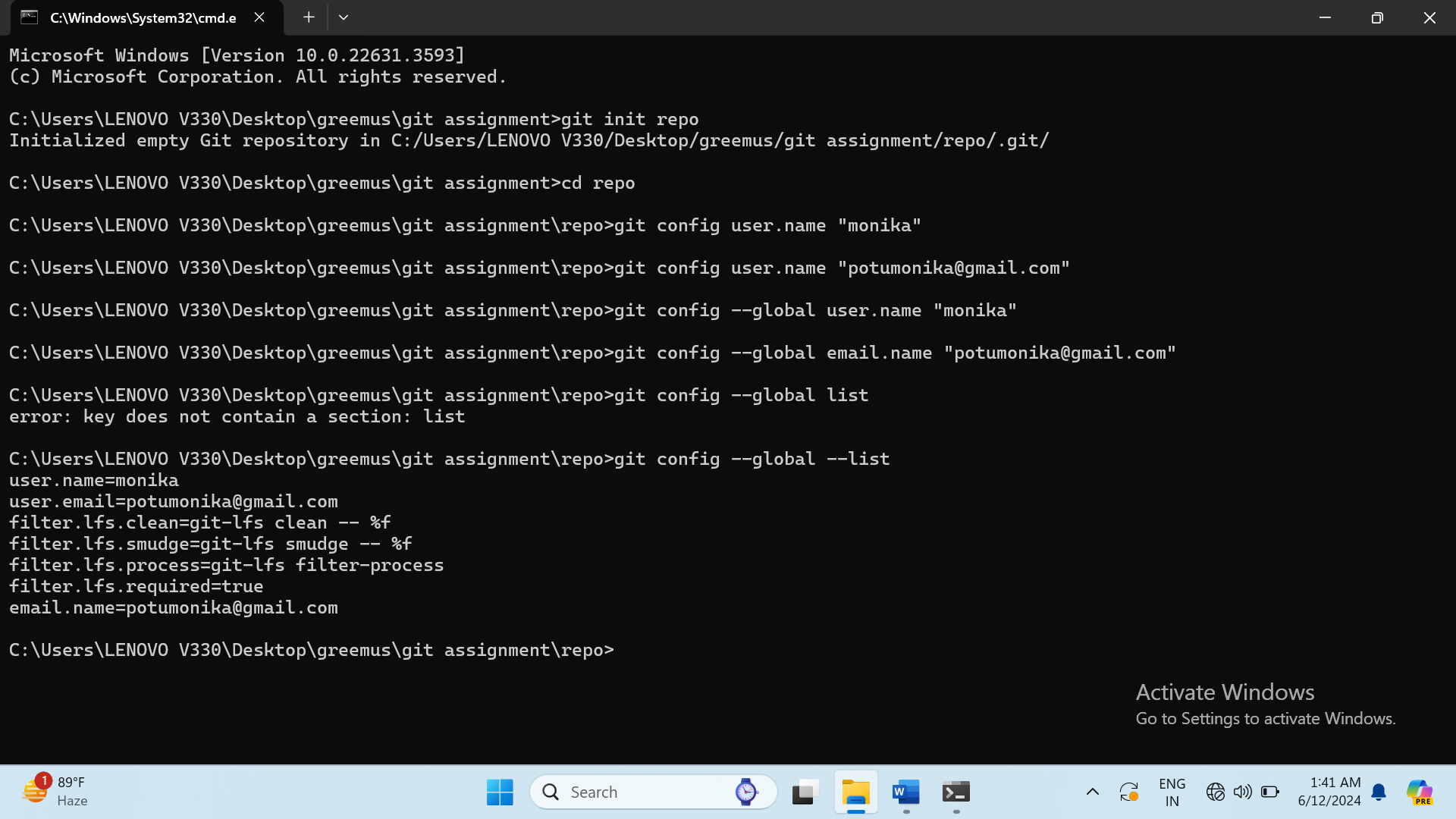
* It retrieves, updates and does not change your working directory or modify your current directory.

1. How do you delete a branch in Git?

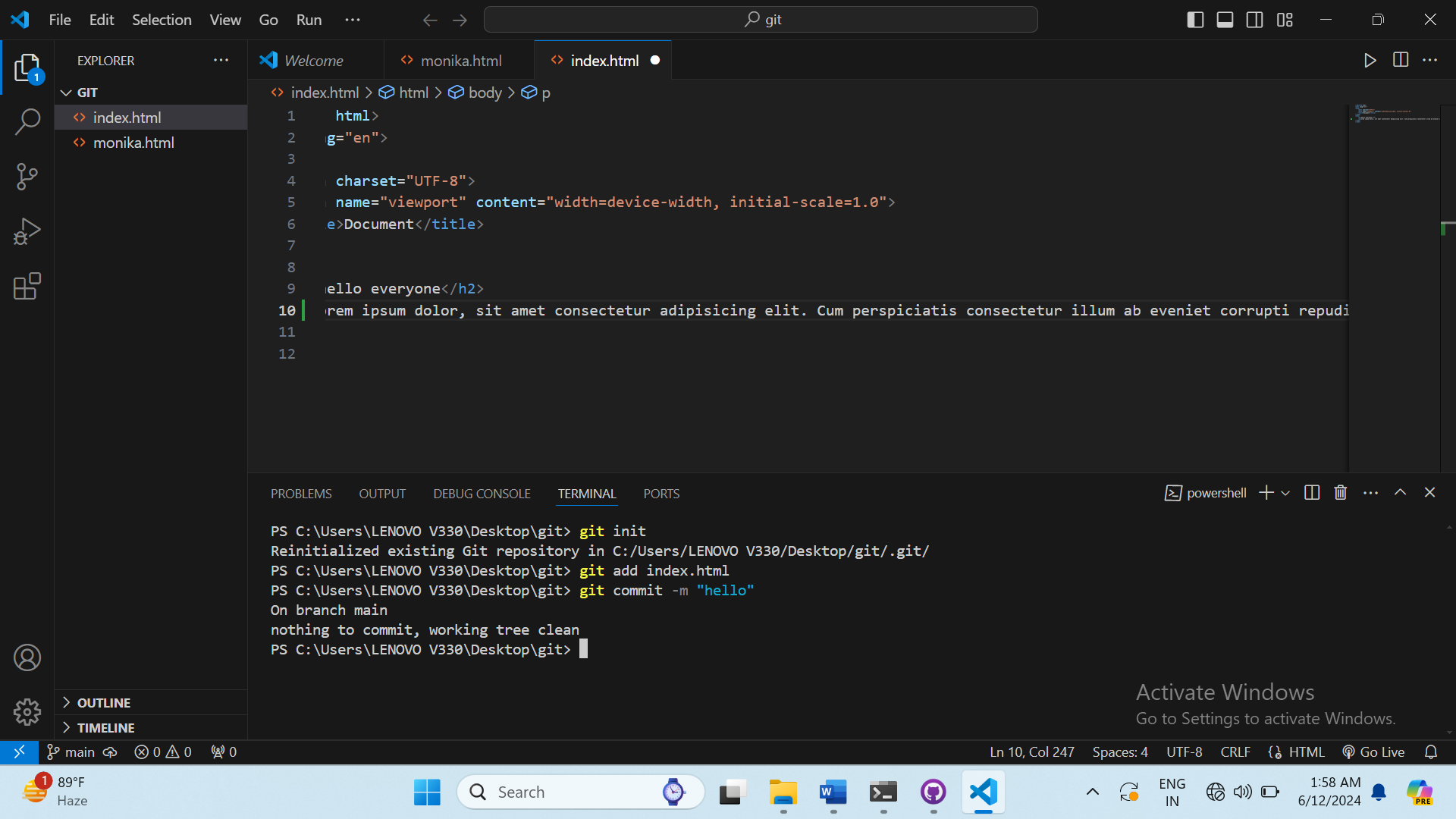
* Delete a local branch: git branch -d <branch name>
* Delete a remote branch: git push origin –delete <branch name>.

### **Git Exercises**

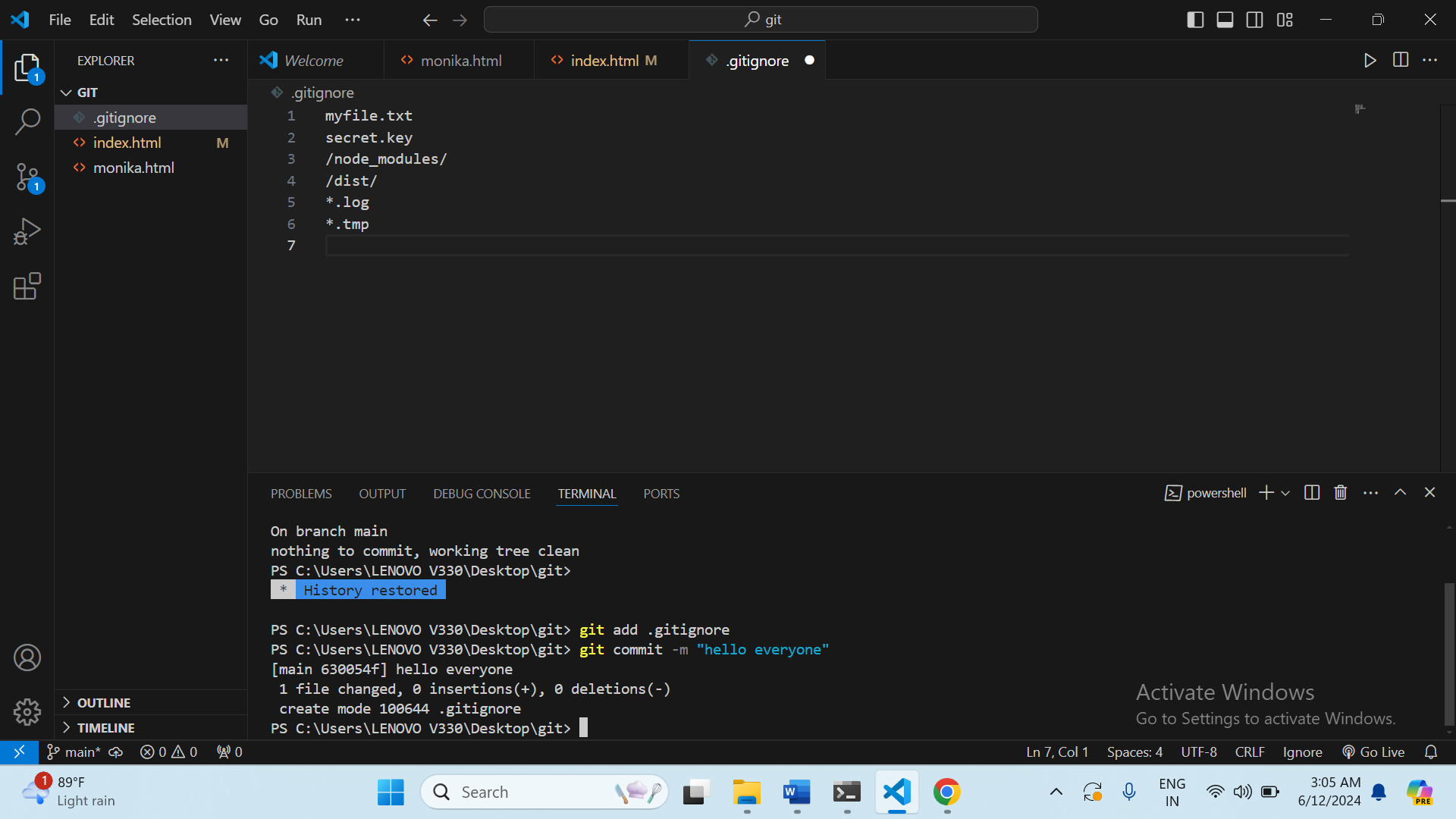
1. Create a new Git repository and configure your username and email.

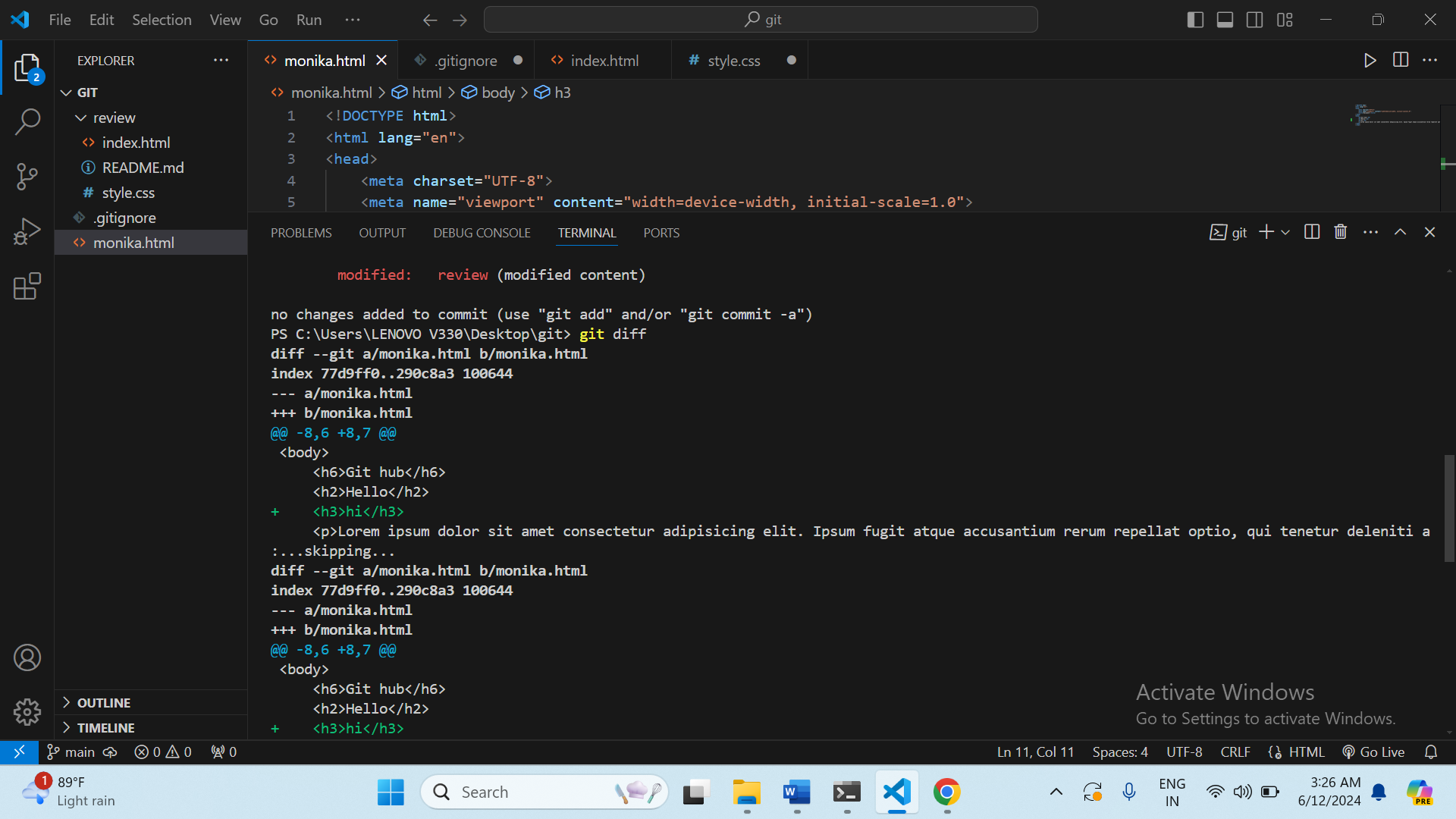
1. Create a file, add some content to it, and commit the changes.



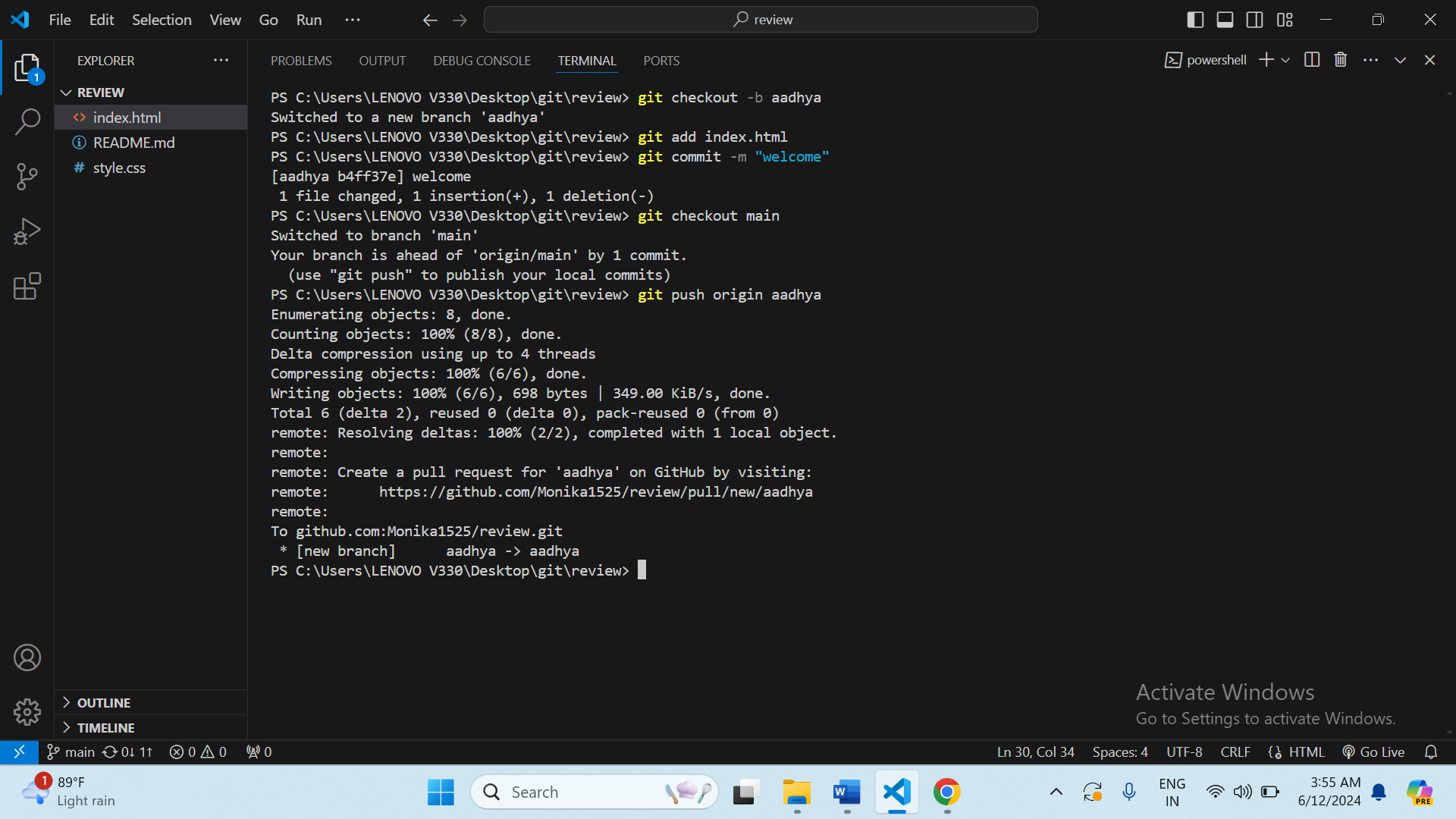
1. Create a .gitignore file and add rules to ignore specific files and directories.



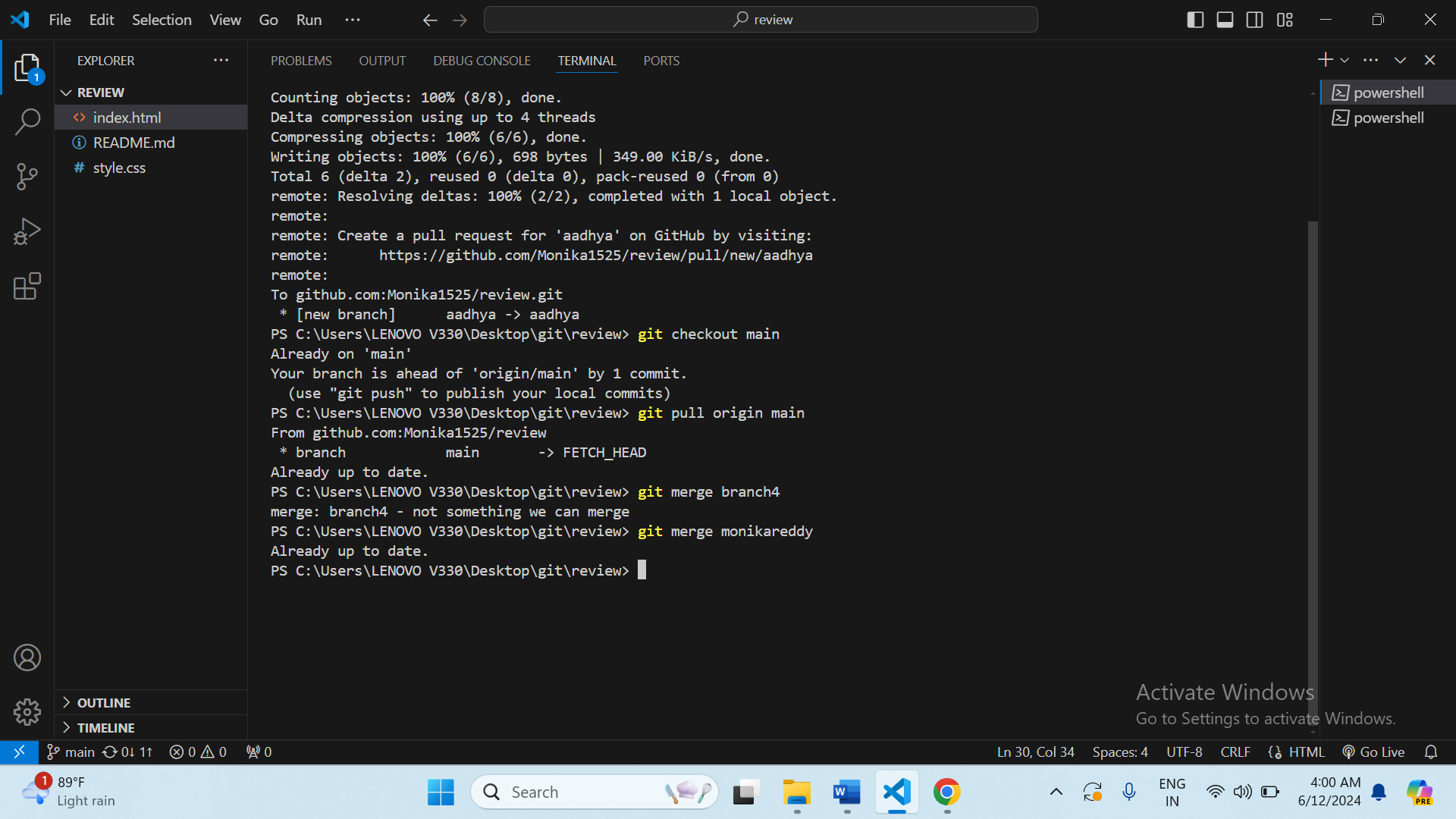
1. Clone an existing repository from GitHub and make some changes.



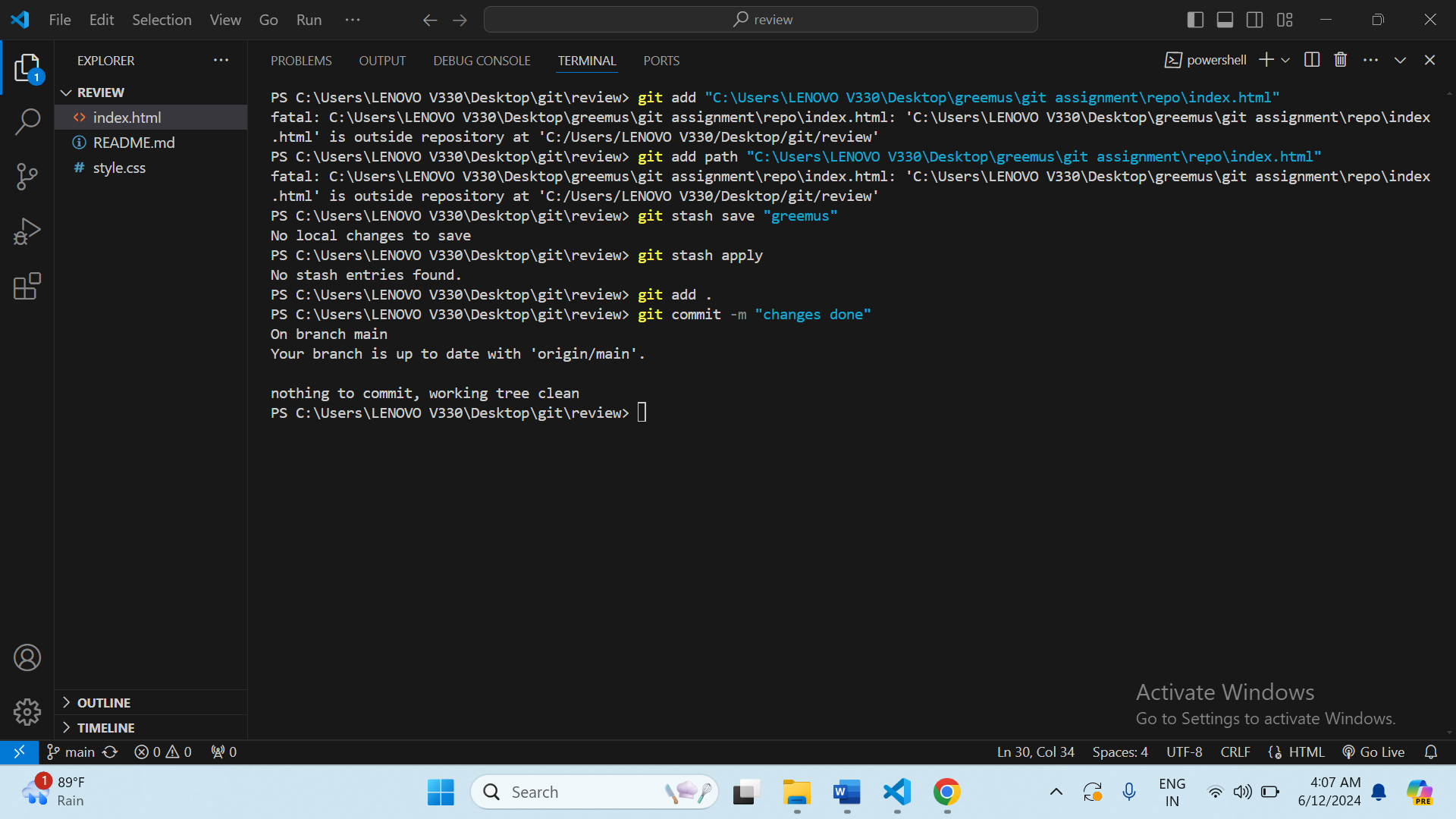
1. Create a new branch, make some changes, and switch back to the main branch.



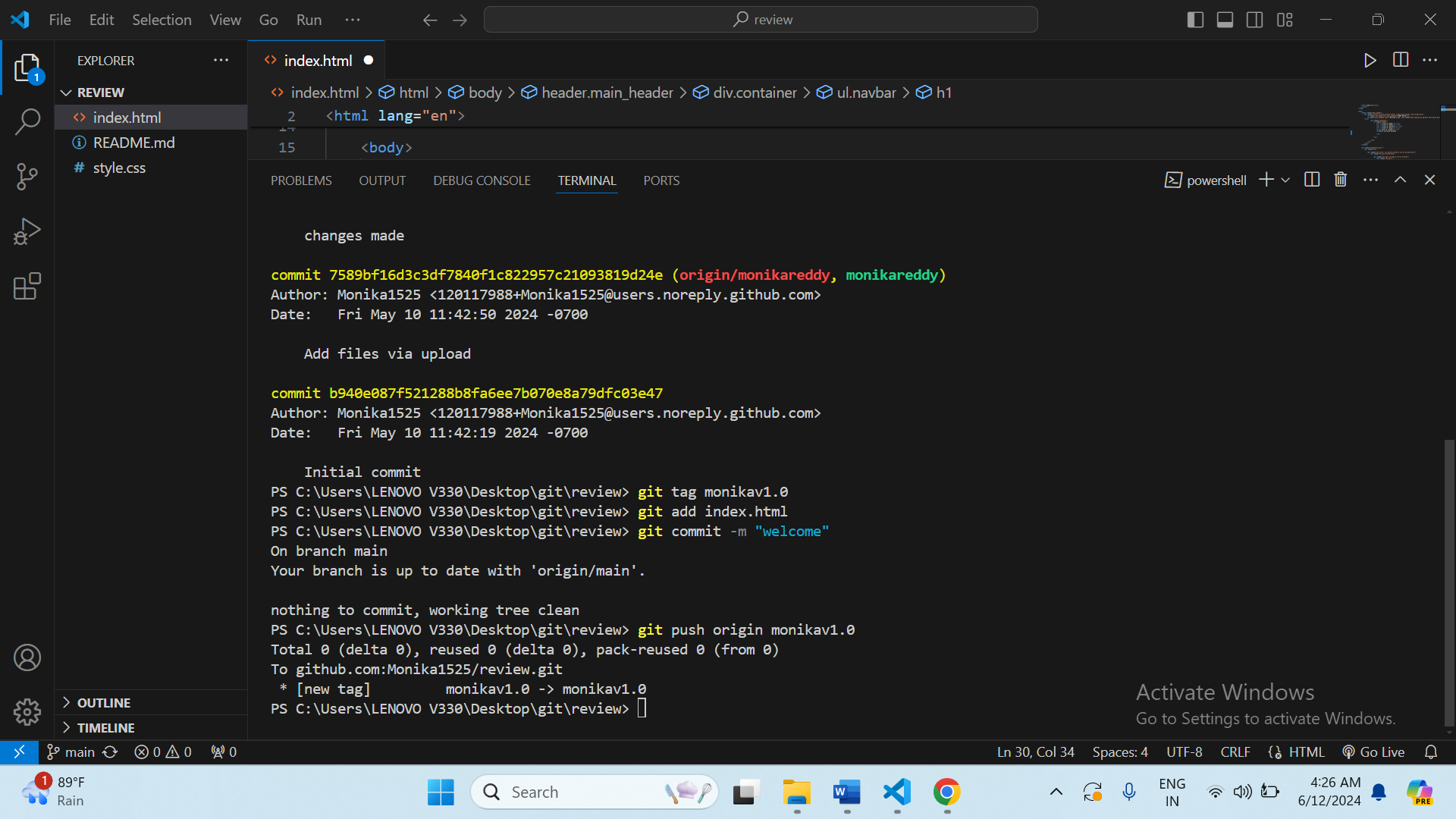
1. Merge changes from a feature branch into the main branch.



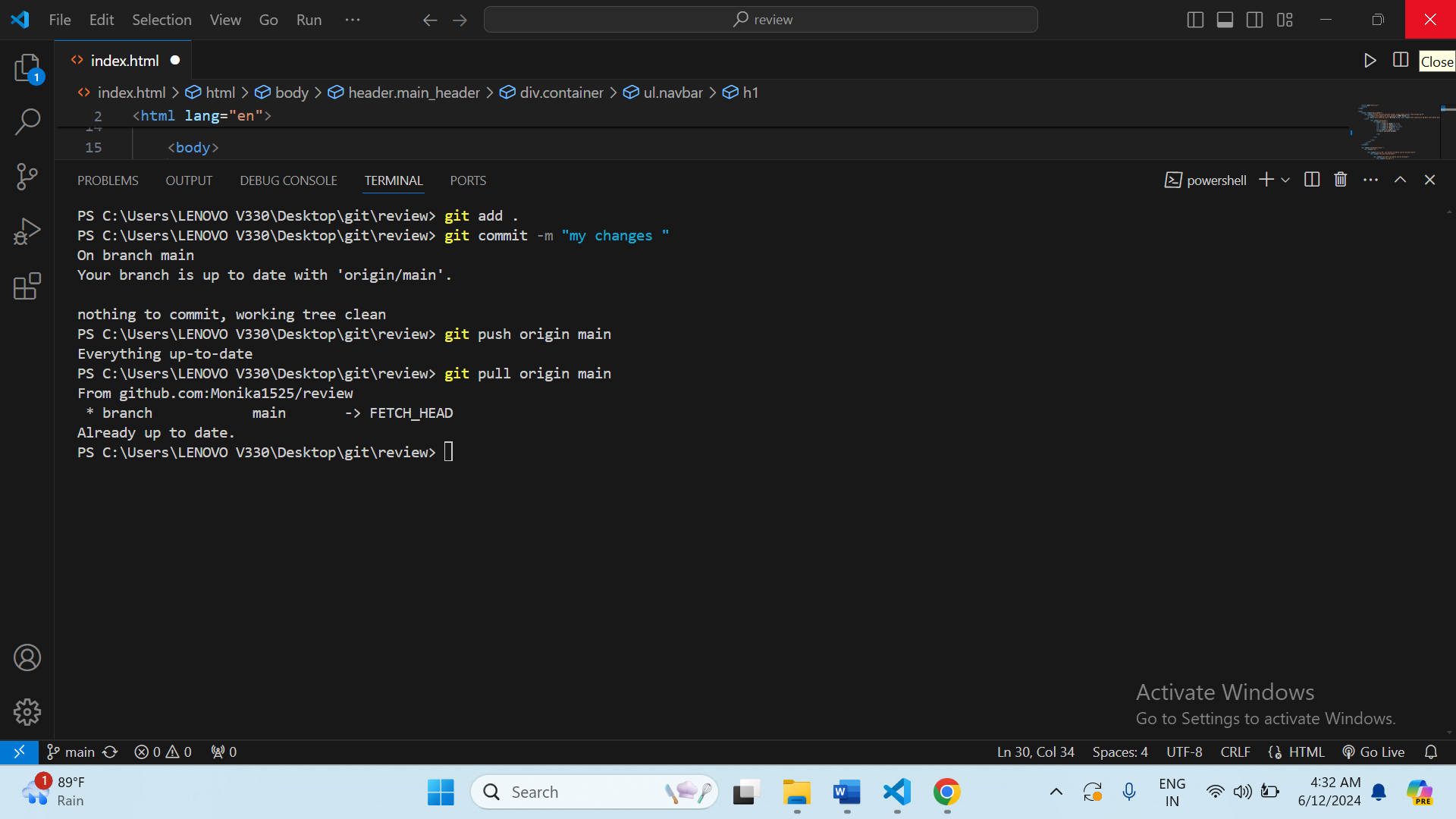
1. Resolve a merge conflict between two branches.
2. Use git stash to save your work and then apply the stashed changes.



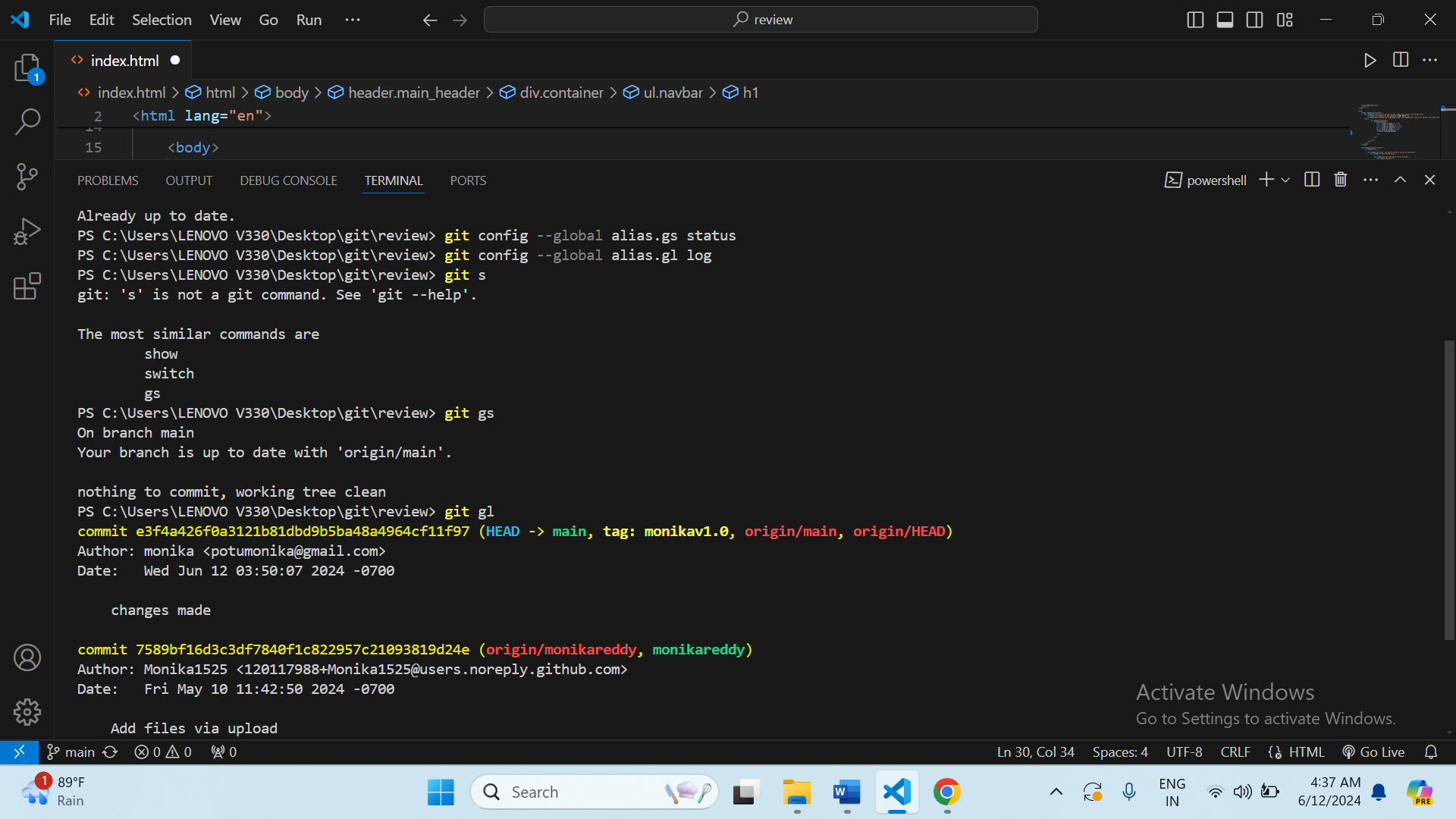
1. Create a tag for a specific commit and push the tag to the remote repository.



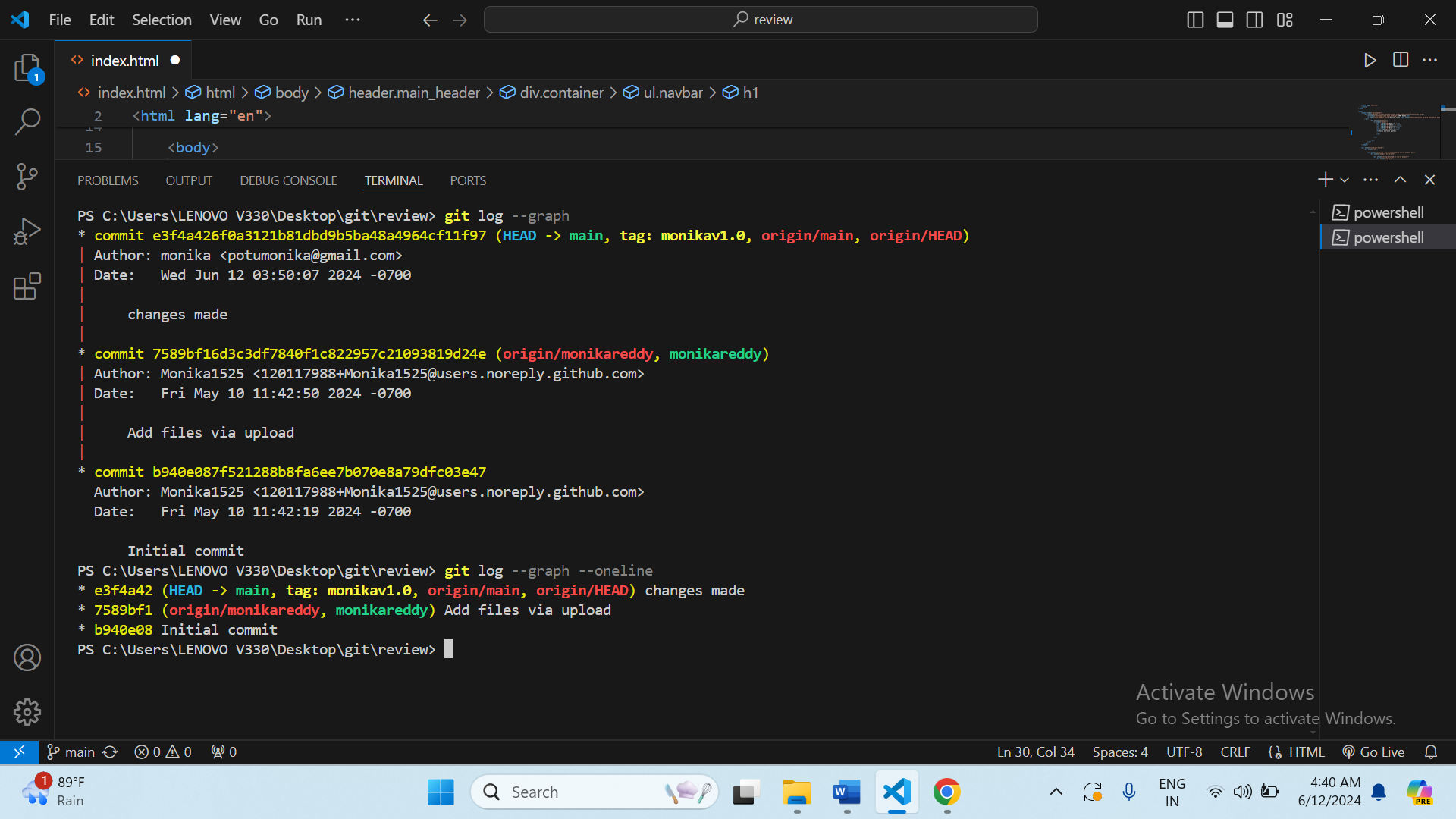
1. Add a remote repository, push your local changes, and pull changes from the remote repository.



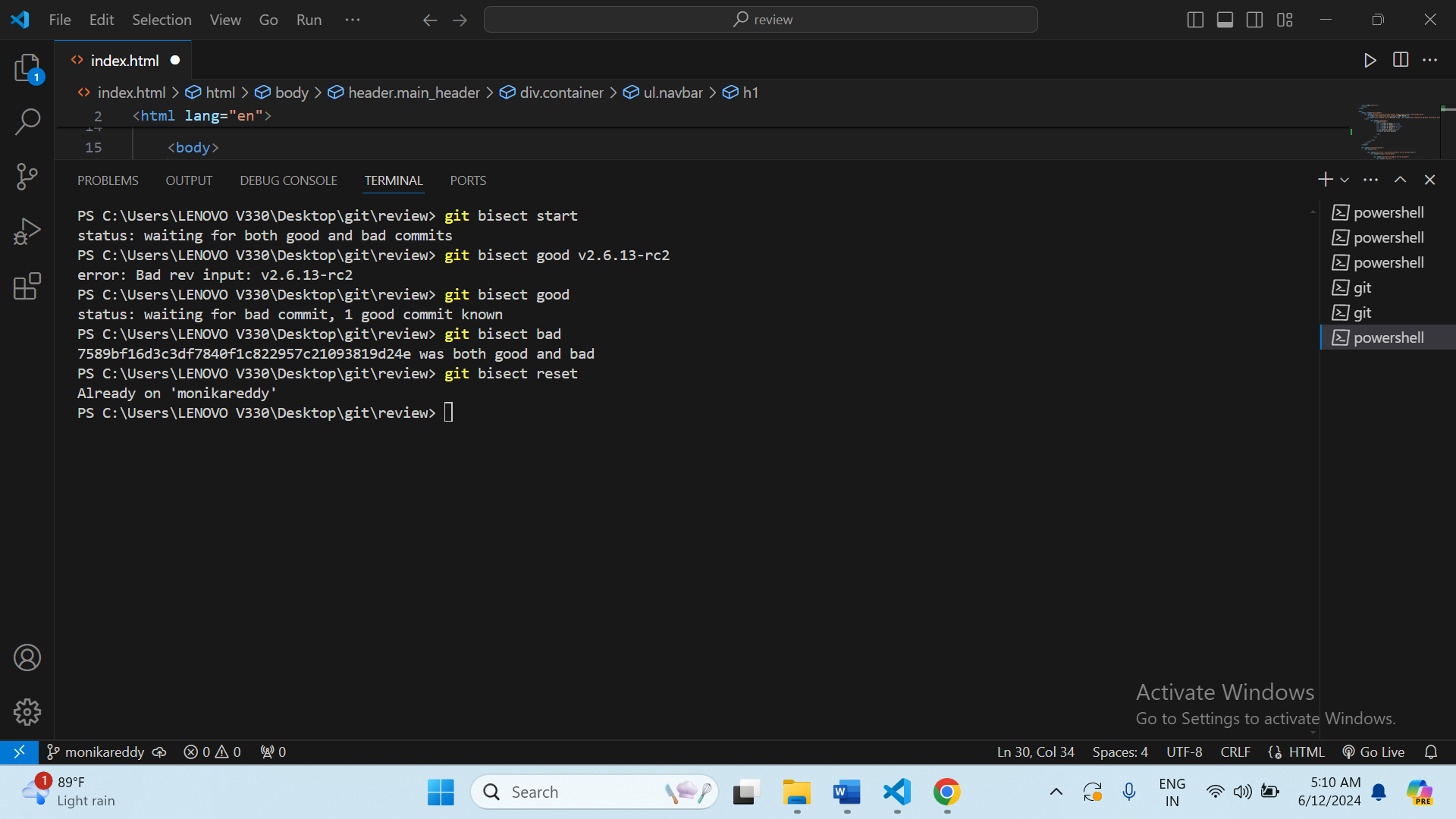
1. Create a Git alias for a commonly used command.



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

