**Git Mini Project: Simple Website**

**Project Overview**

Create a simple website project to learn the basics of Git. The website will consist of an HTML file, a CSS file, and a JavaScript file.

**Steps**

1. **Set Up Your Project**
   * Create a directory for your project.
   * Initialize a Git repository in that directory.

mkdir my-website

cd my-website

git init

1. **Create Initial Files**
   * Create the initial HTML, CSS, and JavaScript files.
   * Make your first commit.

echo "<!DOCTYPE html><html><head><title>My Website</title></head><body><h1>Welcome to My Website</h1></body></html>" > index.html

echo "body { font-family: Arial, sans-serif; }" > styles.css

echo "console.log('Hello, world!');" > script.js

git add index.html styles.css script.js

git commit -m "Initial commit with HTML, CSS, and JavaScript files"

1. **Branching and Feature Development**
   * Create a new branch for adding a new feature.
   * Modify the files in this branch.
   * Commit the changes.

git checkout -b add-navbar

echo "<nav><ul><li>Home</li><li>About</li><li>Contact</li></ul></nav>" > navbar.html

sed -i '/<body>/r navbar.html' index.html

git add index.html navbar.html

git commit -m "Add navbar to the website"

1. **Merging Branches**
   * Switch back to the main branch.
   * Merge the feature branch into the main branch.

git checkout main

git merge add-navbar

1. **Collaborating with Others**
   * Simulate collaboration by creating a new branch, making changes, and merging back into the main branch.
   * Handle potential merge conflicts.

git checkout -b update-styles

echo "nav { background-color: #333; color: white; }" >> styles.css

git add styles.css

git commit -m "Update navbar styles"

git checkout main

git merge update-styles

If there are conflicts, Git will indicate the files with conflicts. Open the conflicted files, resolve the conflicts manually, add the resolved files, and commit the merge.

# Resolve conflicts in your editor

git add styles.css

git commit -m "Resolve merge conflict"

1. **Using Remote Repositories**
   * Create a remote repository on a platform like GitHub.
   * Push your local repository to the remote.

git remote add origin <remote-repository-URL>

git push -u origin main

From this point, you can pull changes from the remote repository, collaborate with others, and push updates to the remote repository.

**Final Thoughts**

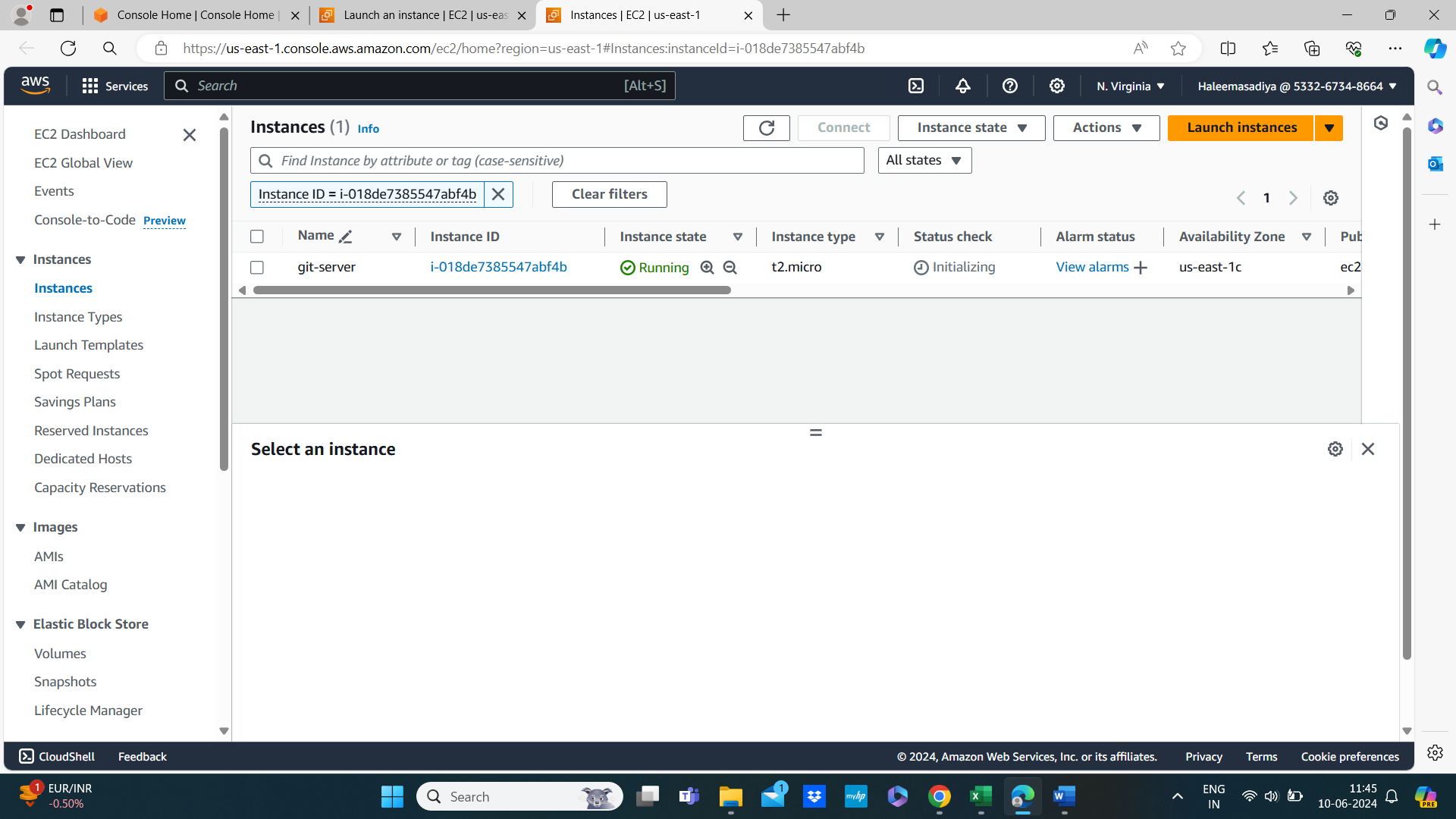
This mini project should give you a hands-on introduction to using Git for version control. Here are a few additional tasks you could try to further solidify your understanding:

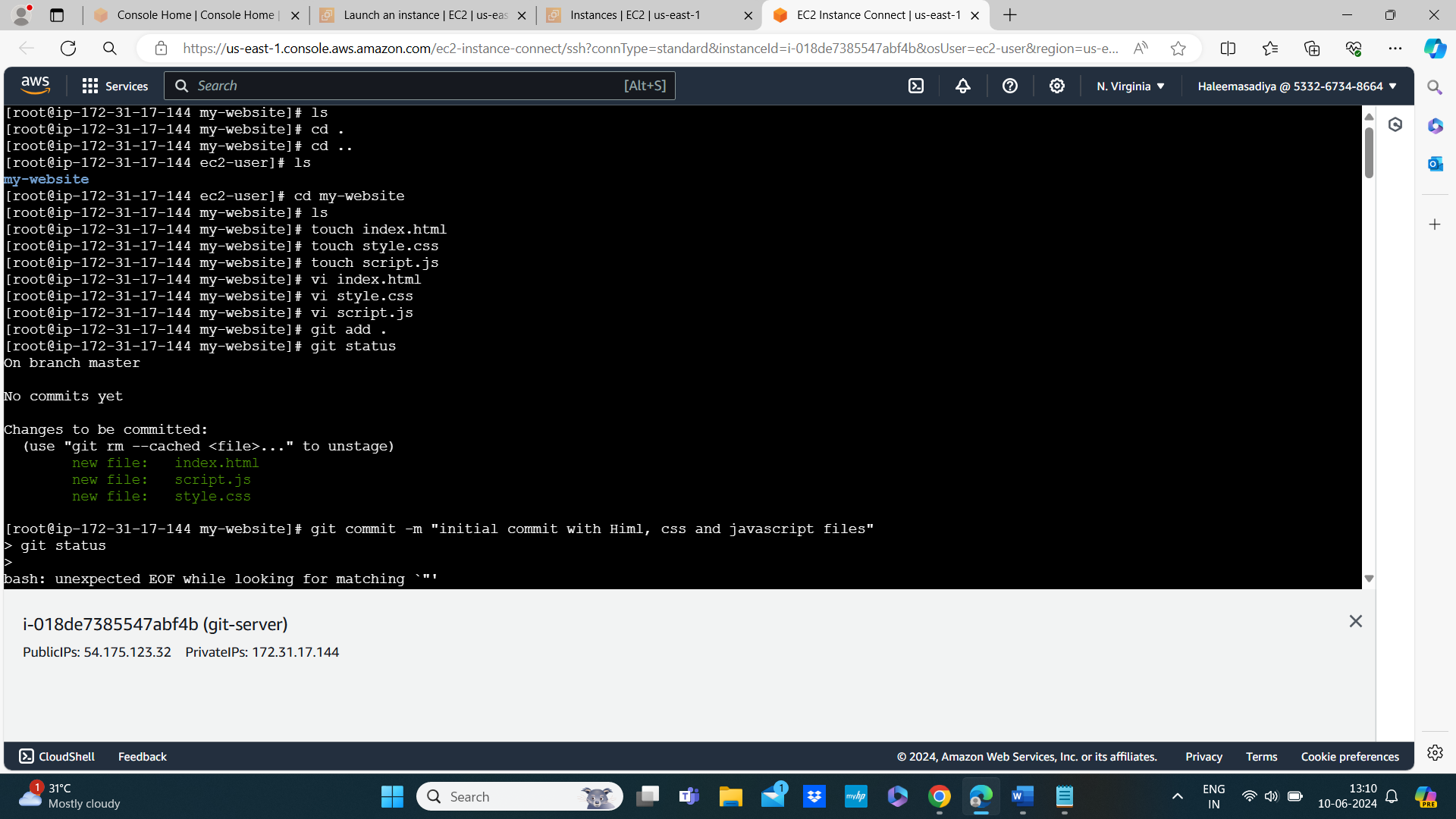
* **Create and use tags**: Mark important points in your project history (e.g., v1.0, v2.0).
* **Stash changes**: Temporarily store changes that are not ready to be committed.
* **Rebase**: Learn how to rebase branches for a cleaner project history.
* **Explore advanced collaboration**: Use pull requests and code reviews if you're working with others.

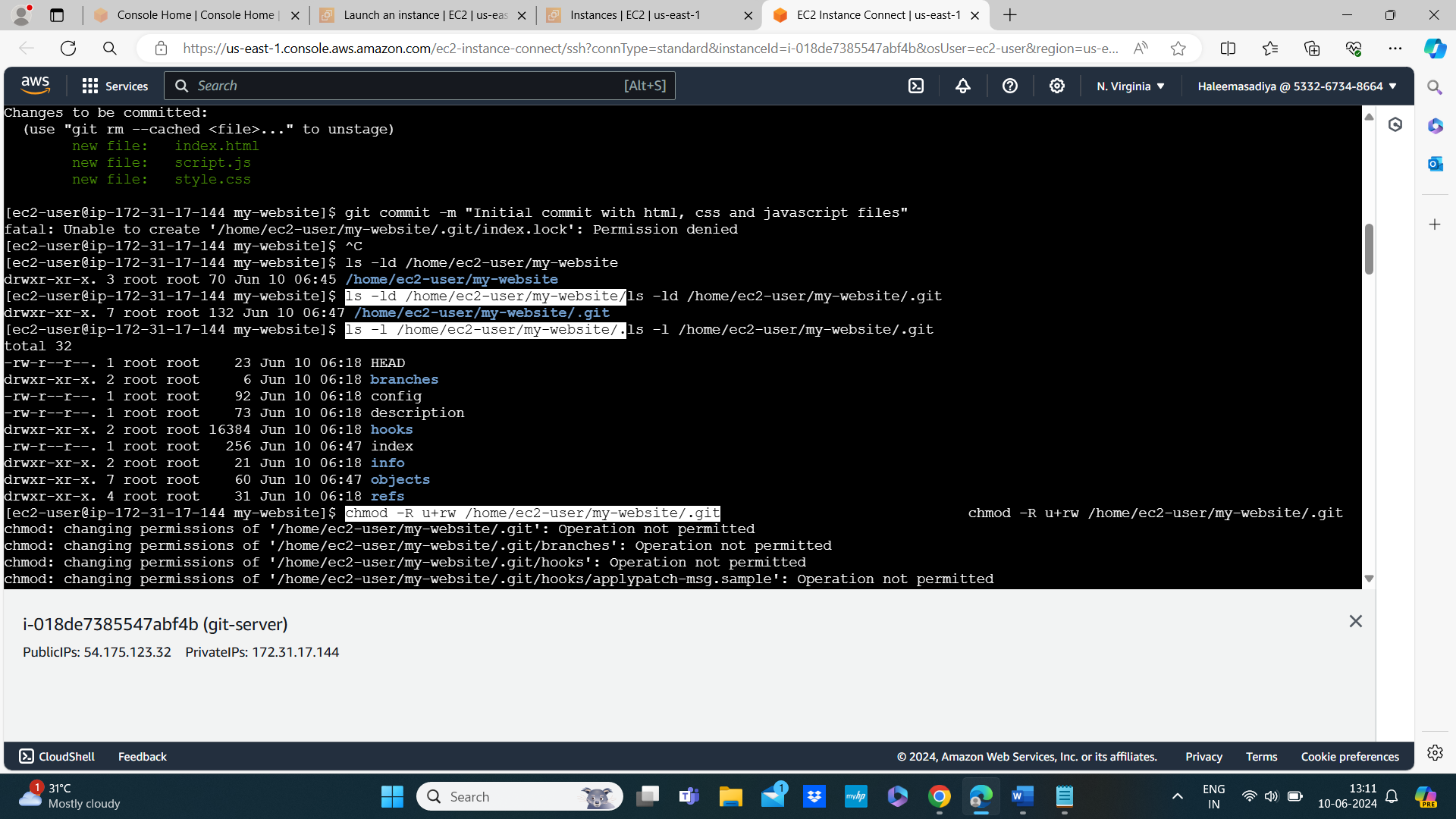
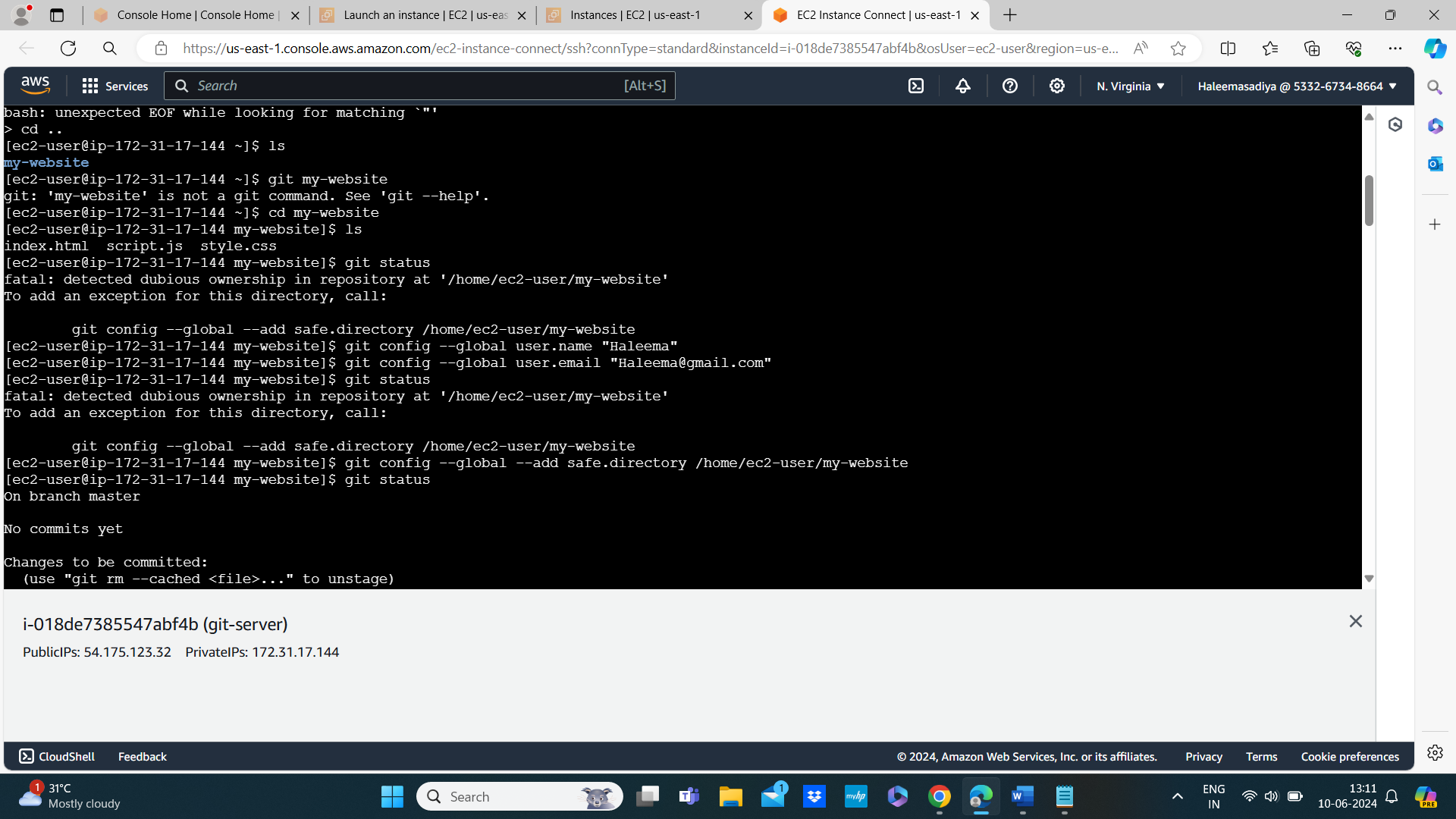
By following these steps and exploring additional features, you'll develop a strong foundation in using Git for your development projects.

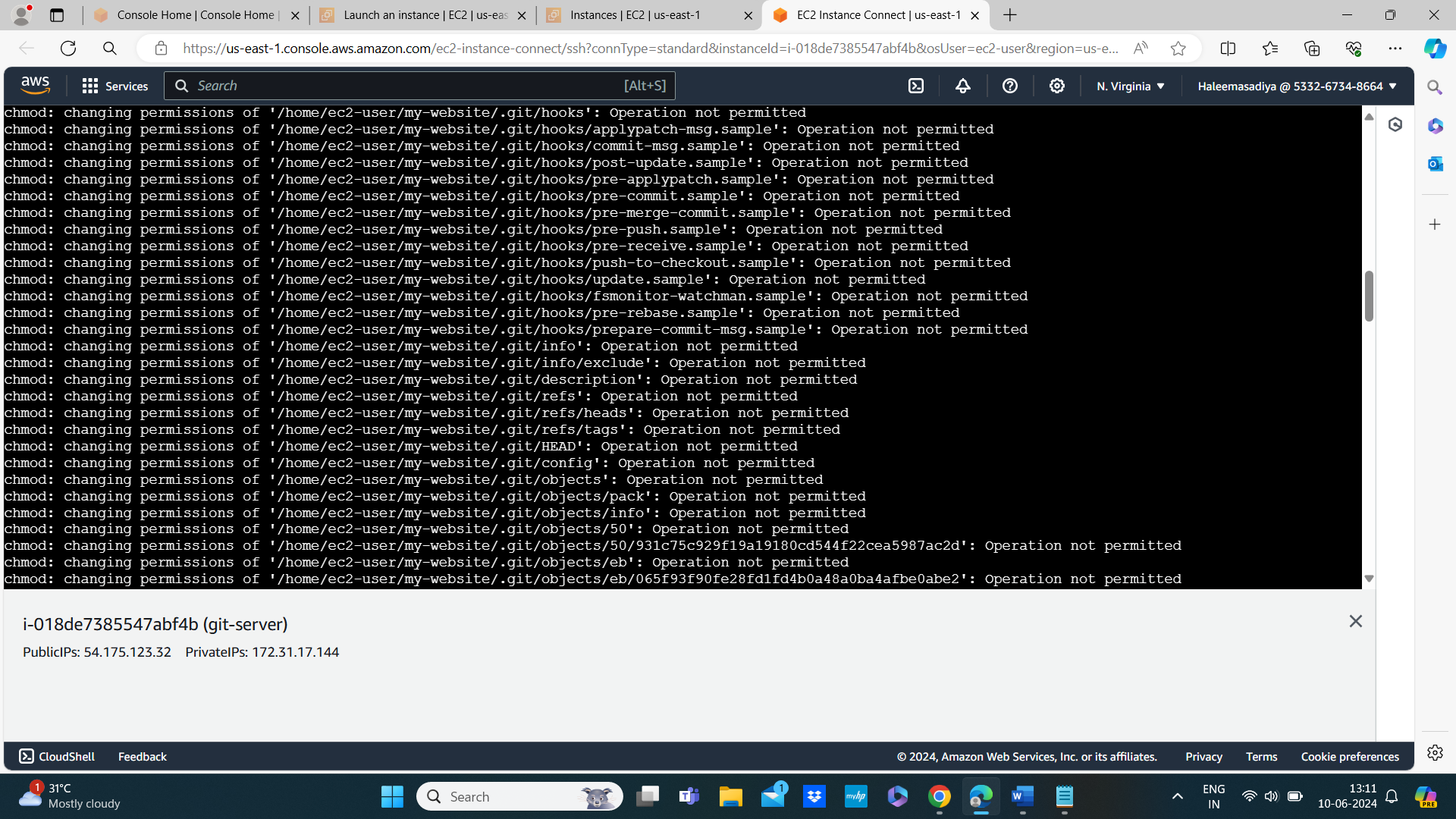
Top of Form

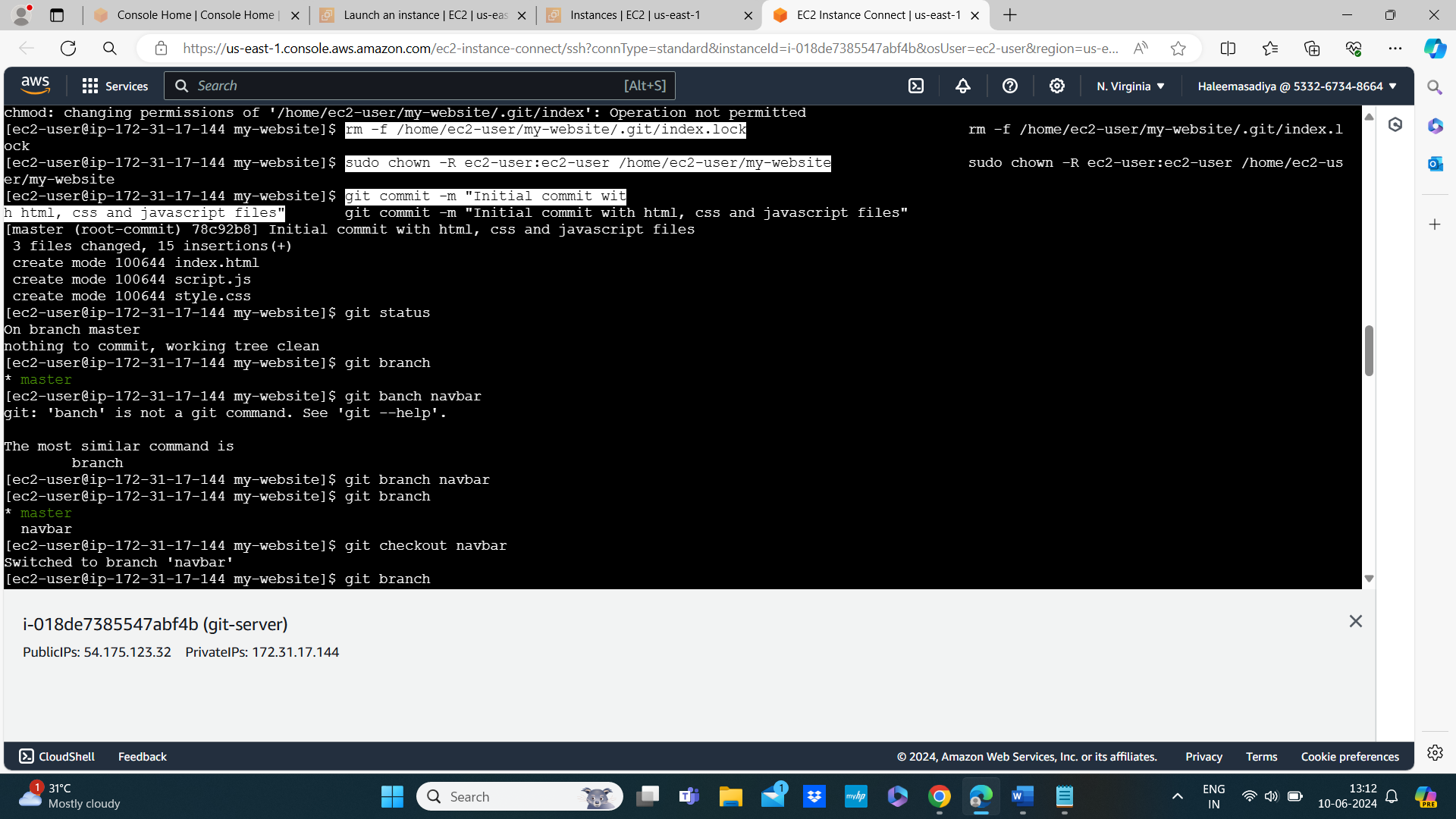
Bottom of Form

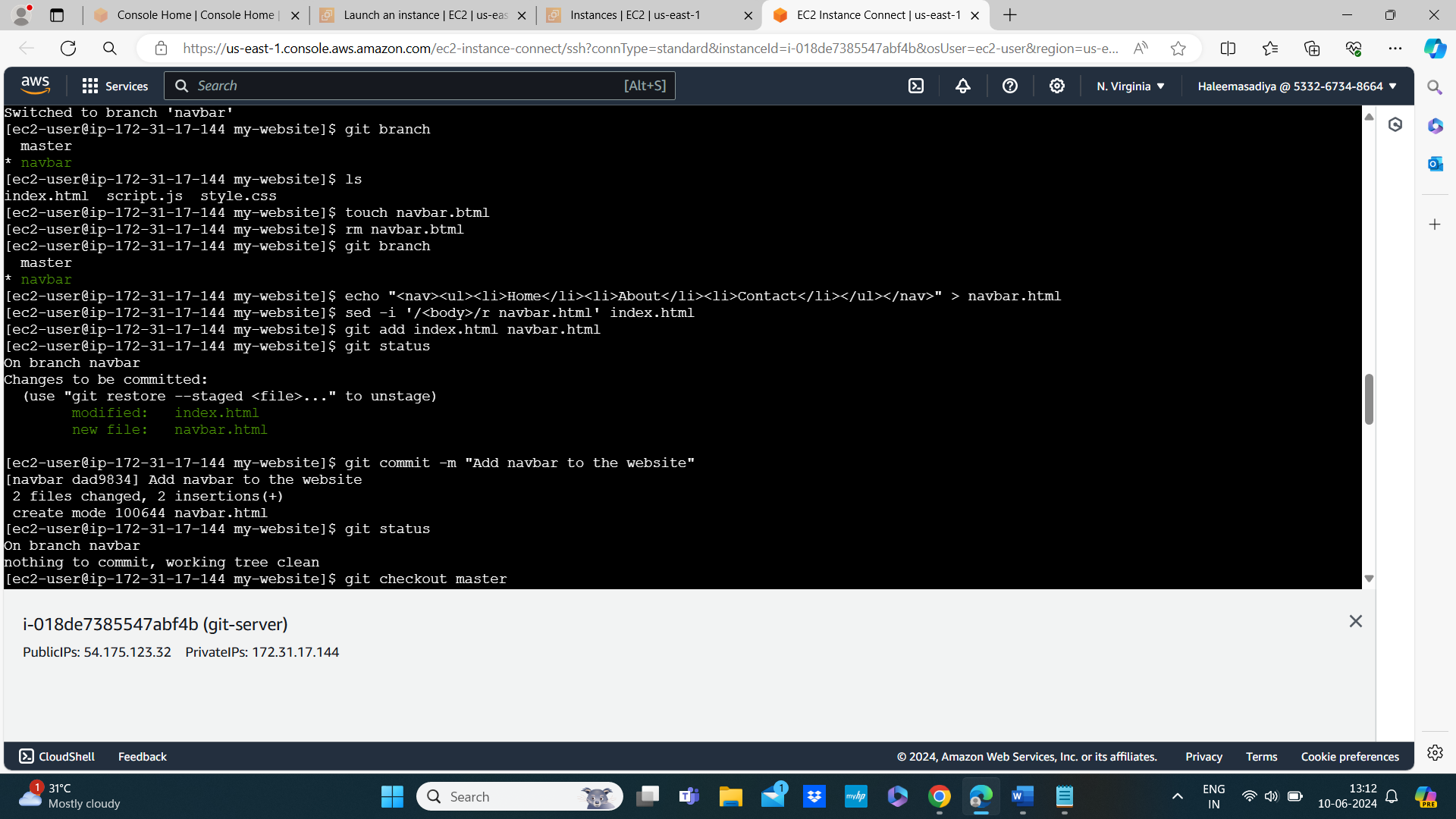


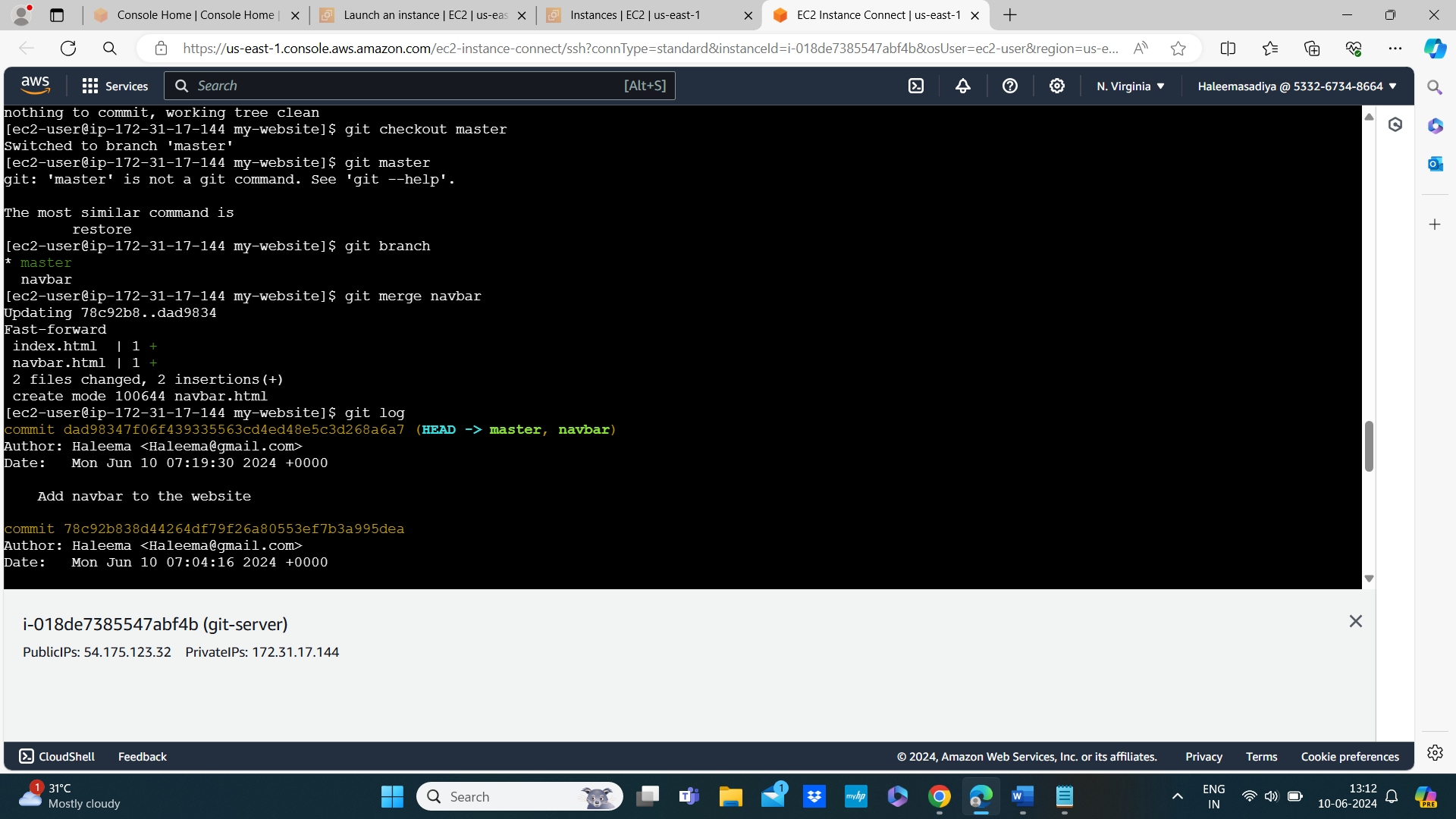


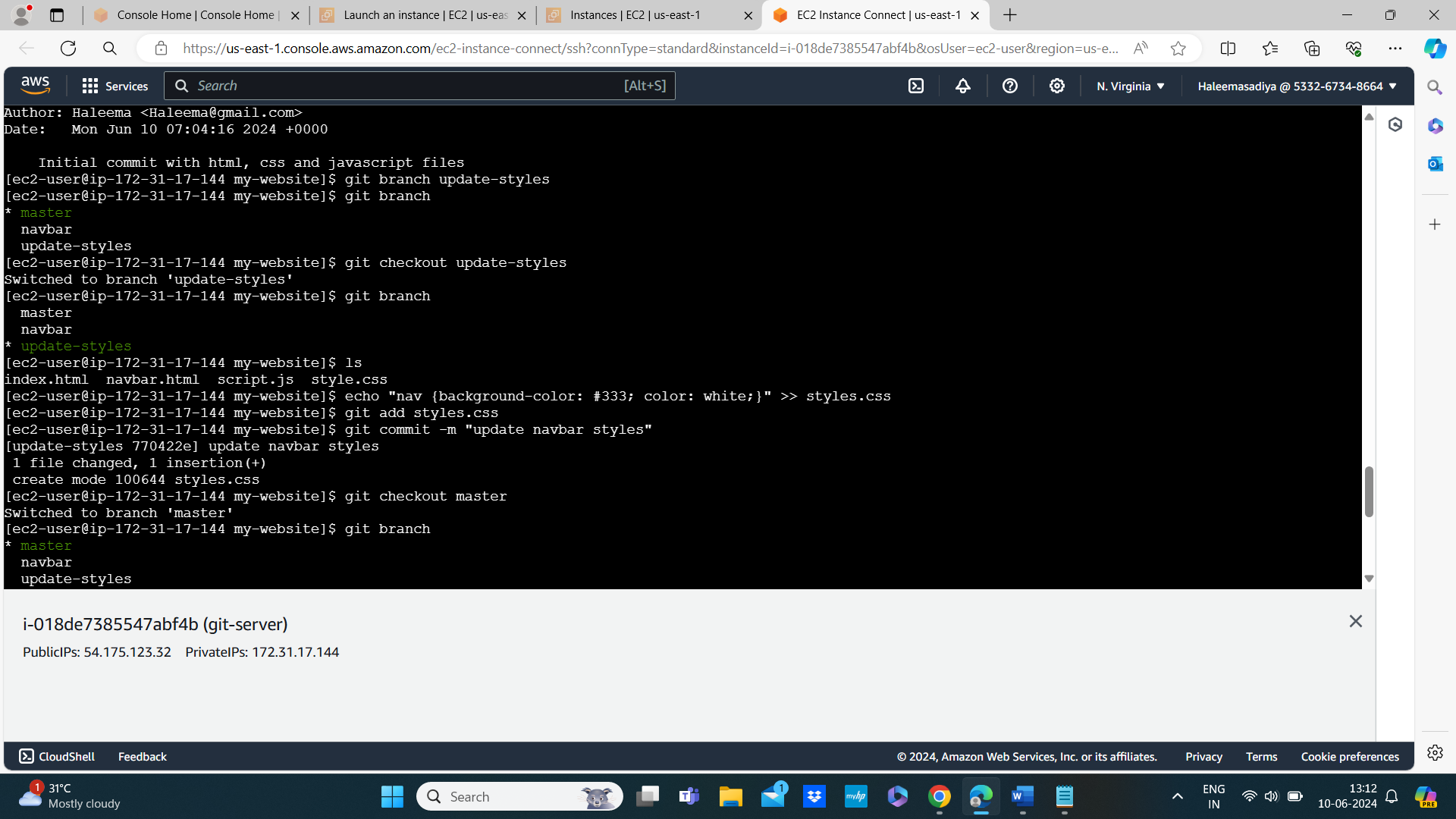


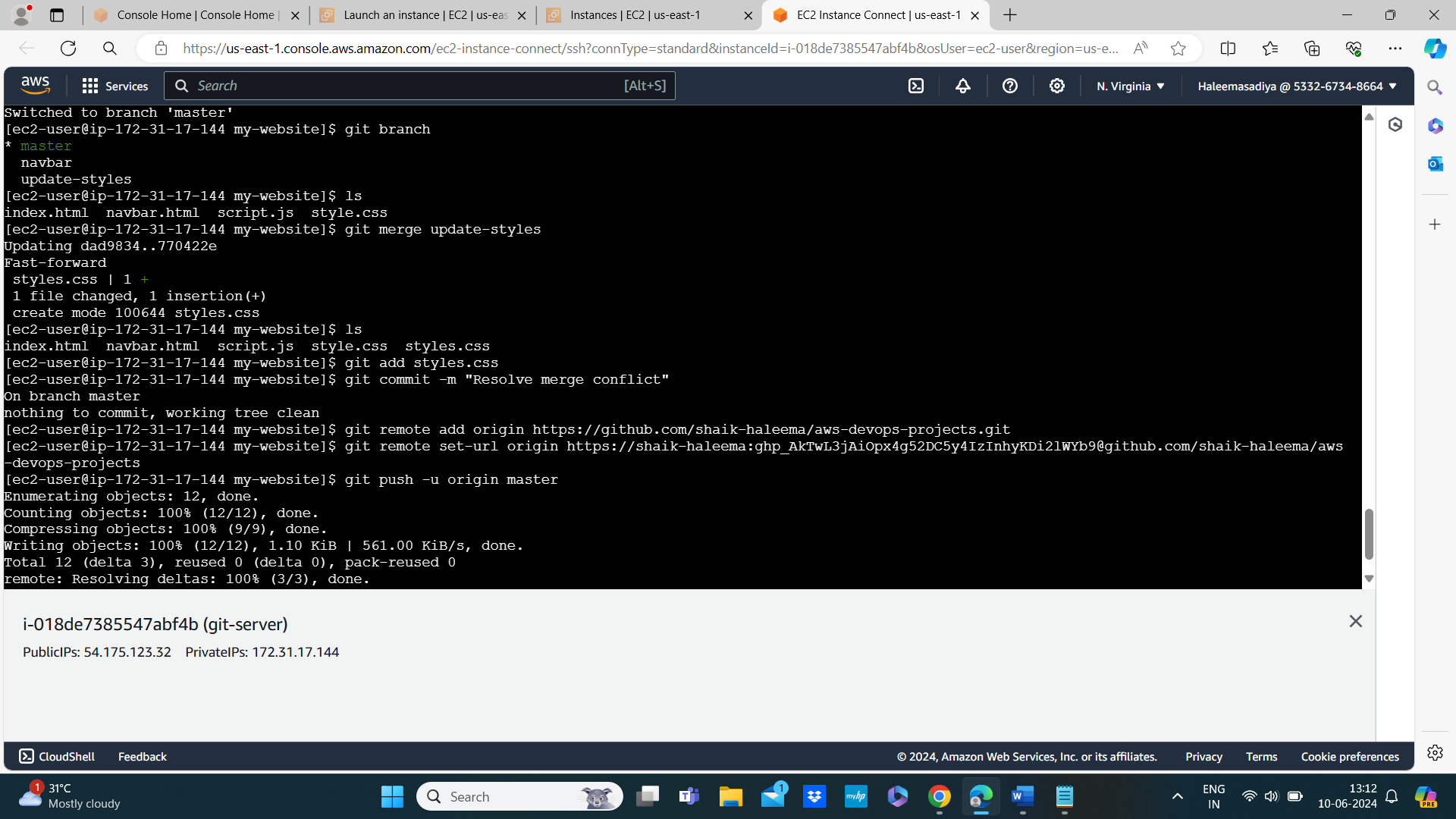


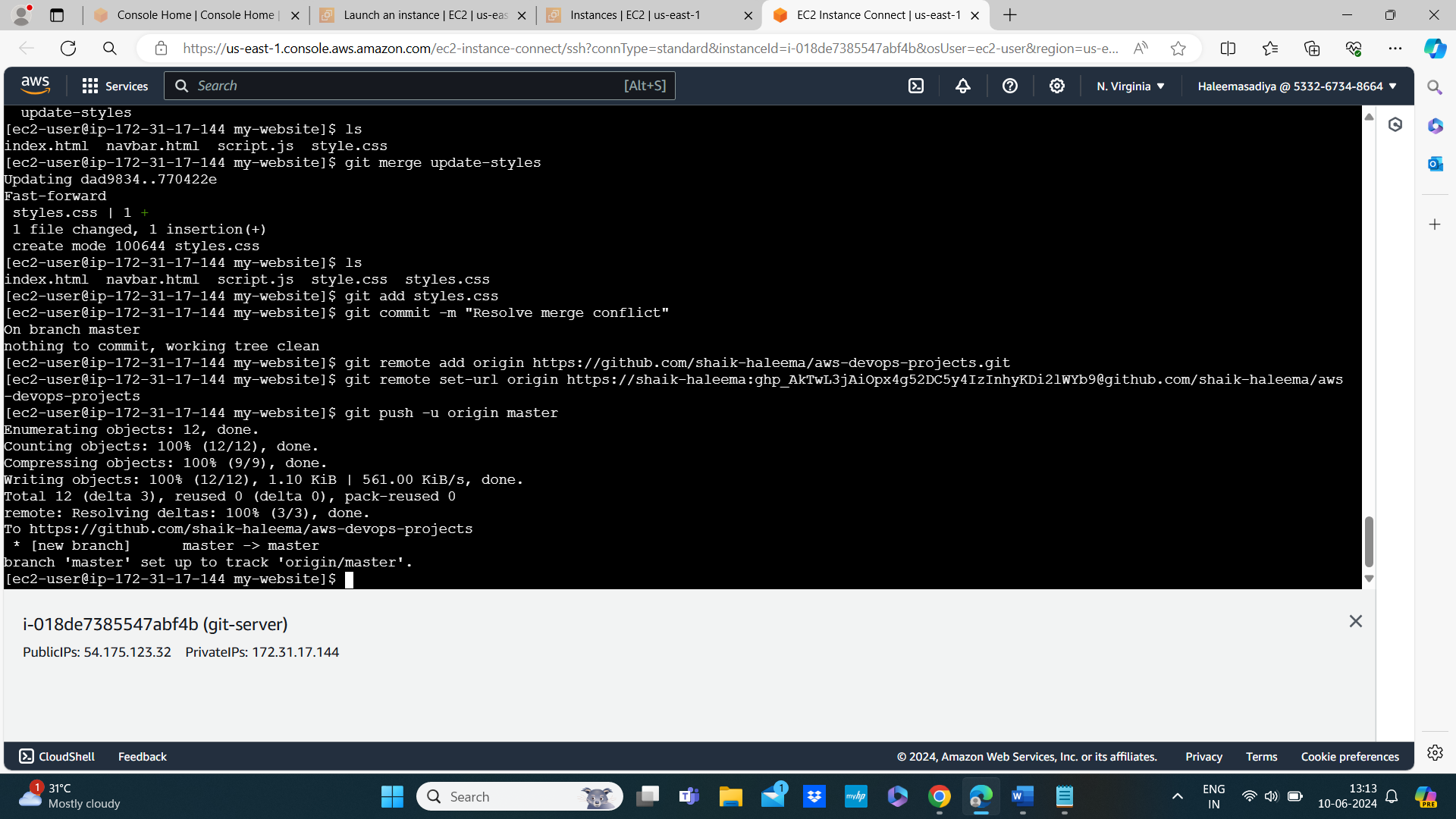


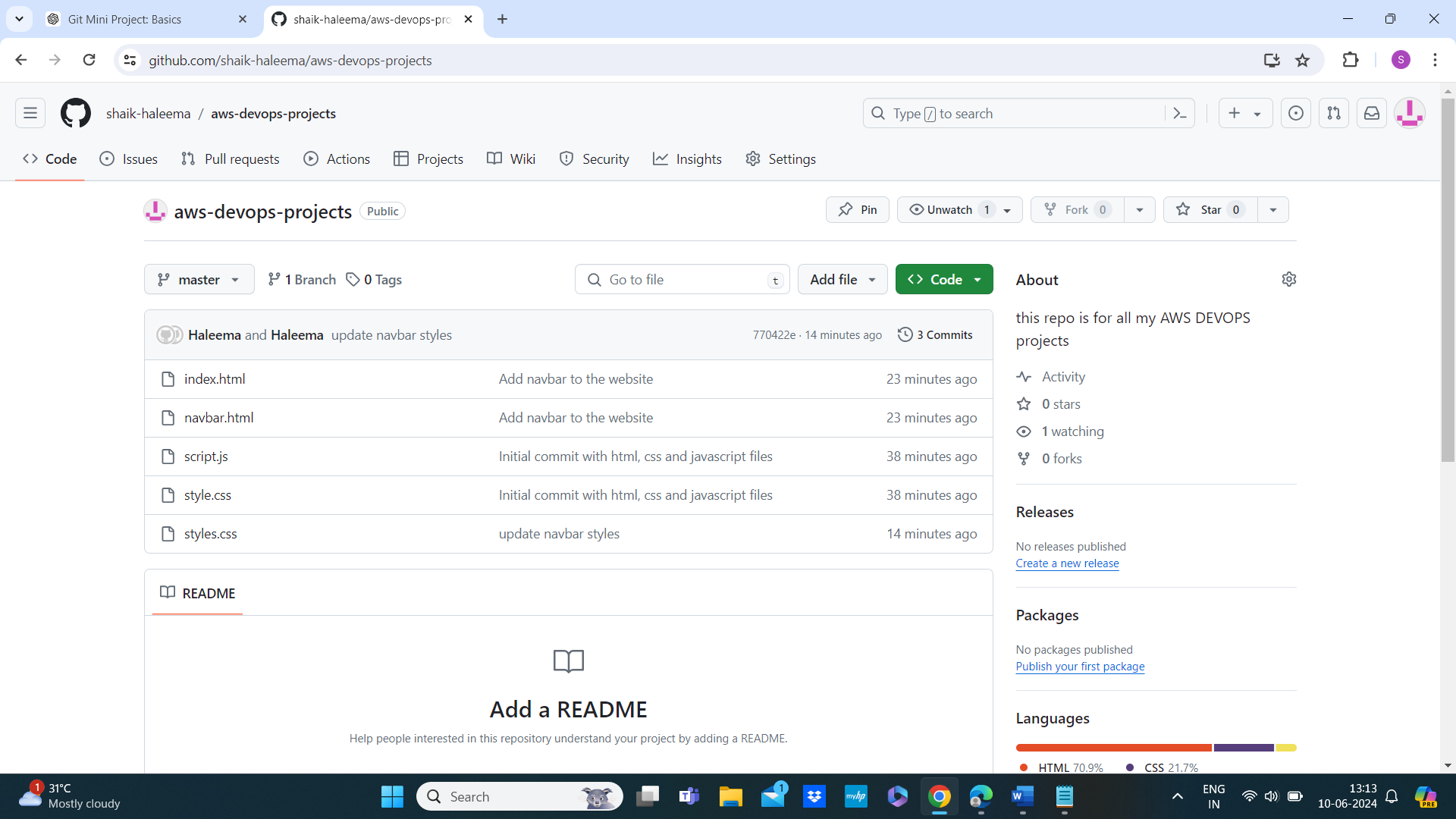


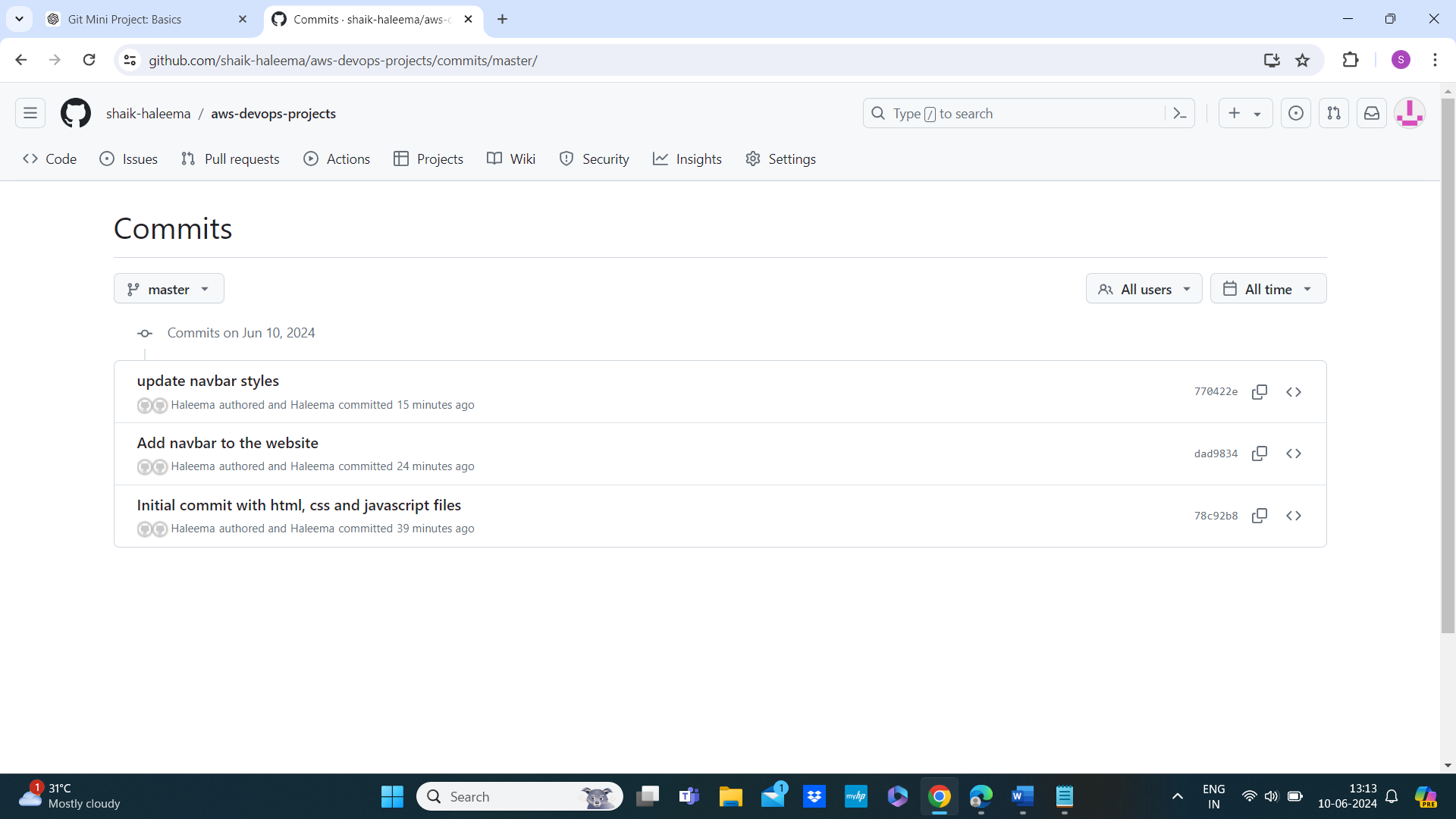


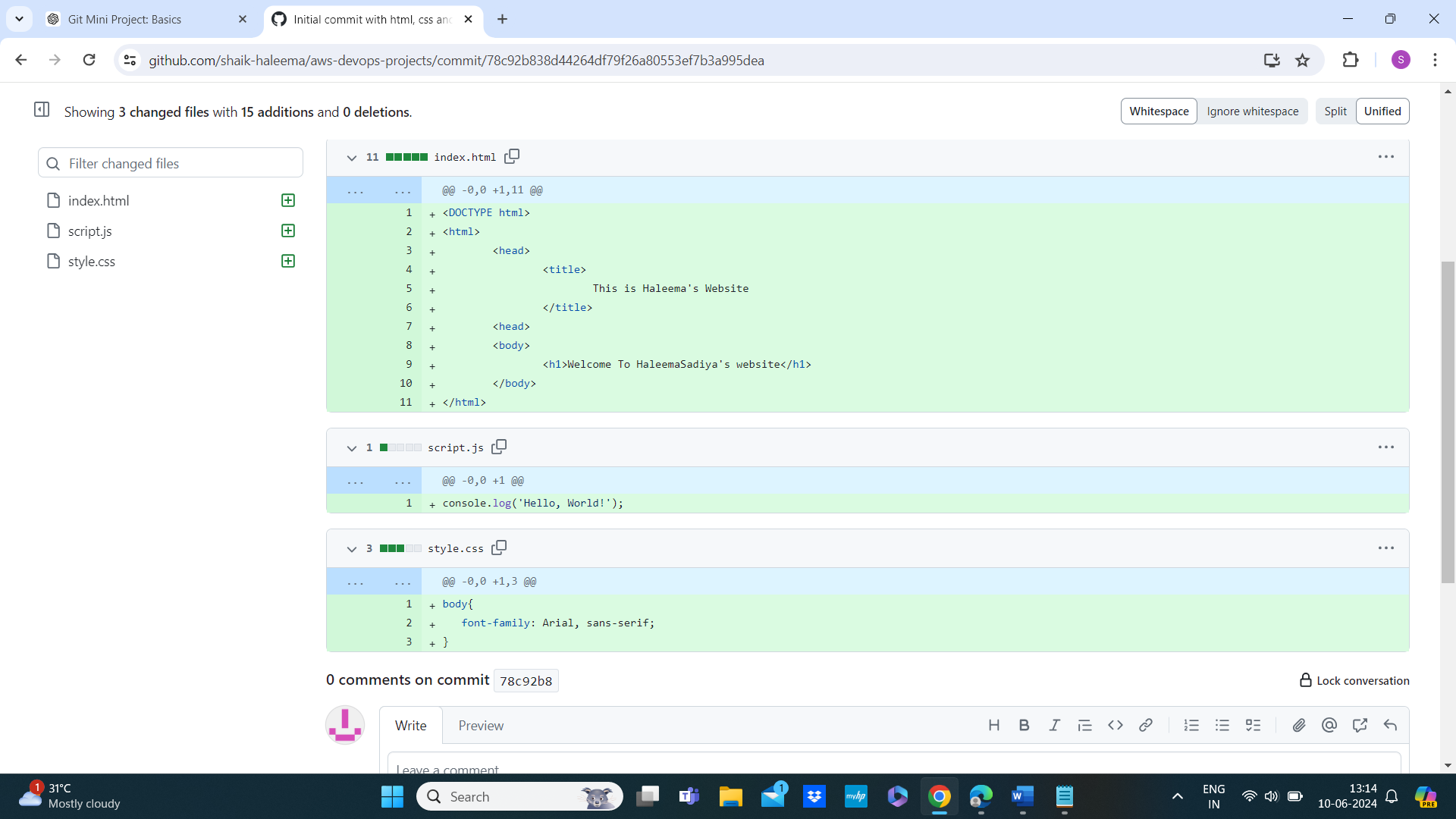




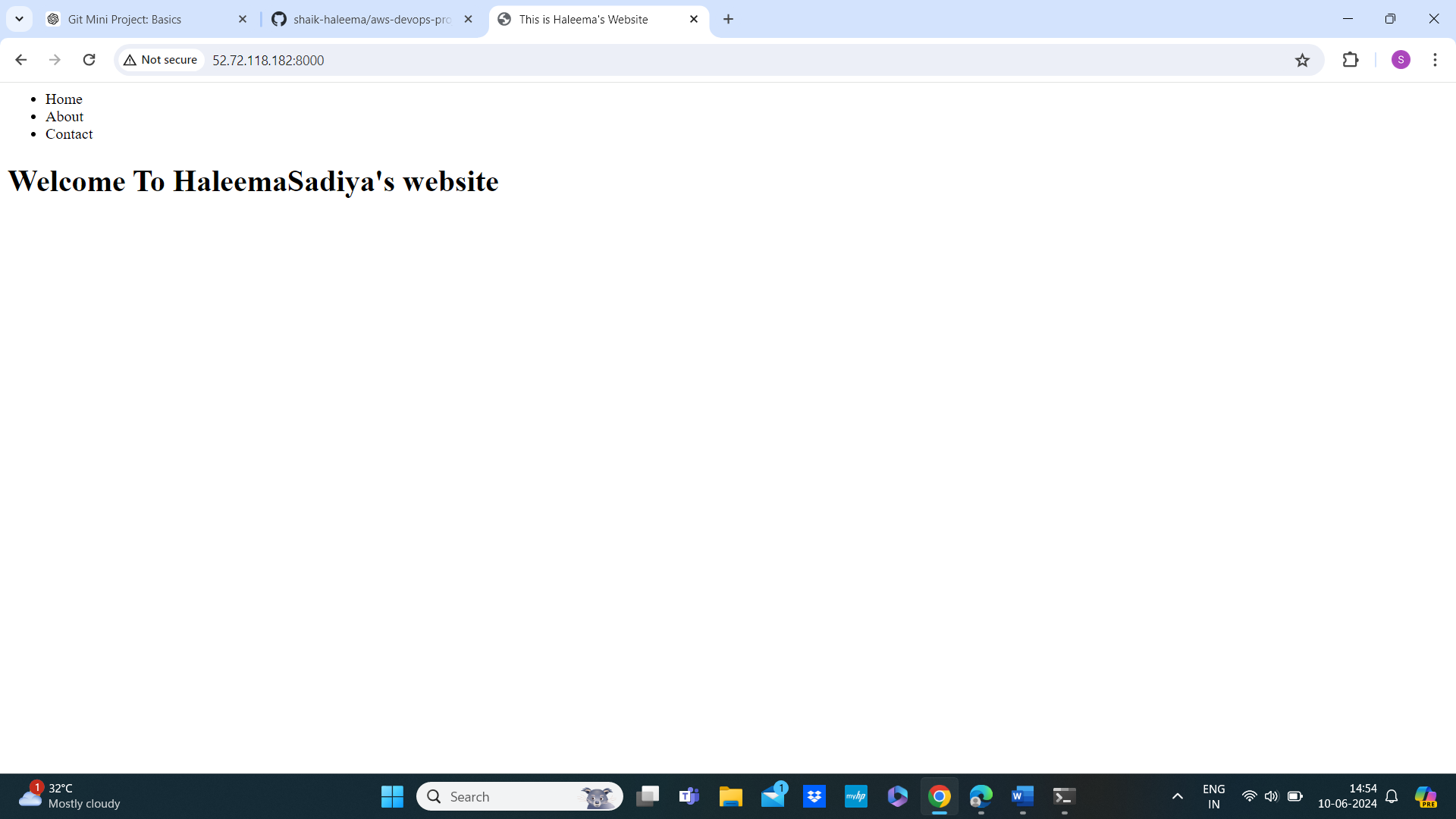








Woow this website is working



Finally terminate the instance to avoid charges

