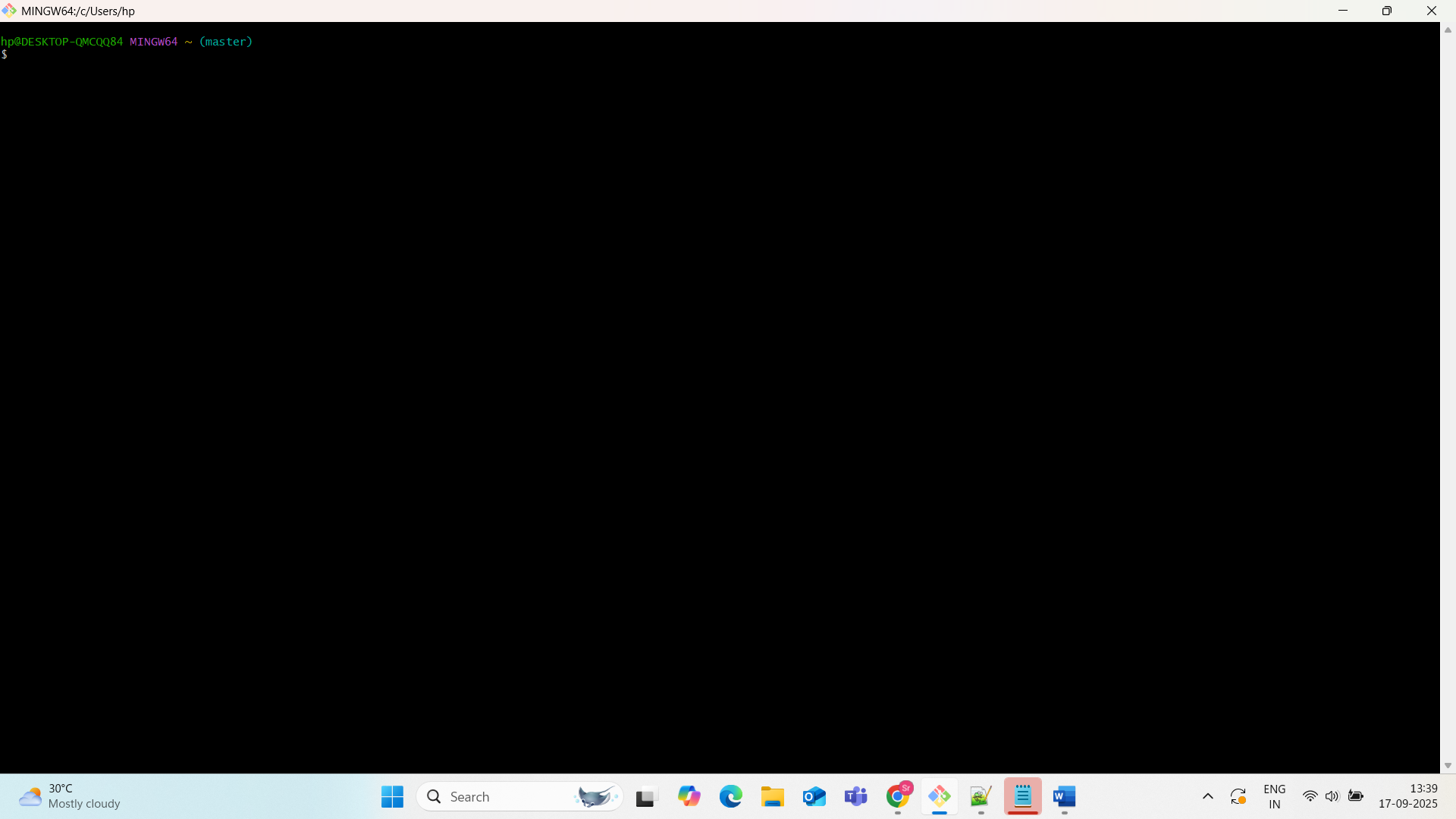
Git and github task2(class9&10)

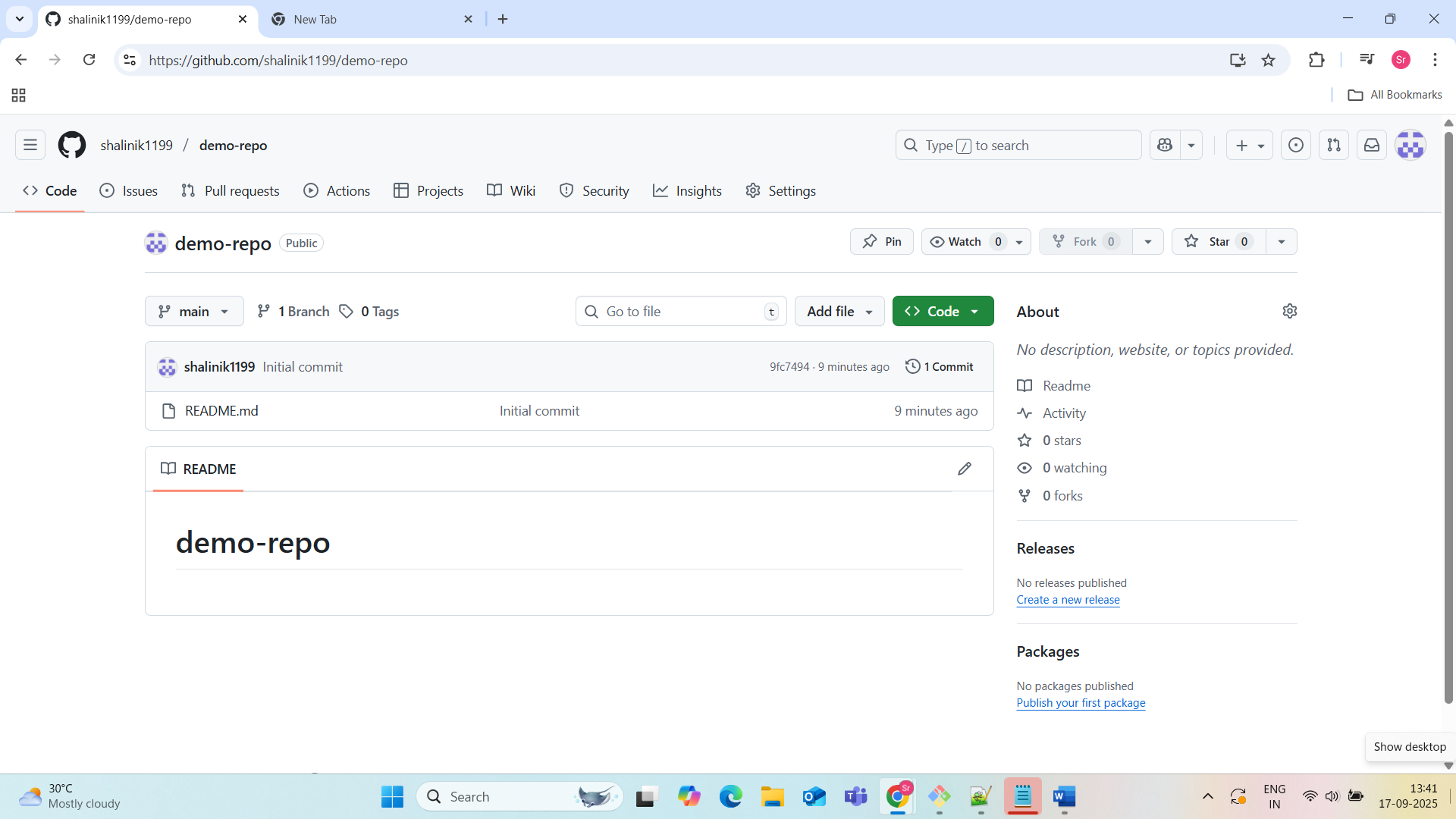
1).install git



**1. Create a repo in GitHub with README.md and .gitignore**

**Steps:**

1. Go to [GitHub](https://github.com/)
2. Click on + > New repository
3. Enter repository name (e.g., demo-repo)
4. Check the options:
   * ✅ Initialize this repository with a README
   * ✅ .gitignore (Choose a template, e.g., Python, Node, etc.)
5. Click Create Repository



**2. Clone the created repo to local**

You want to **copy a repository from GitHub** (the remote) **to your computer** (the local machine), so you can start working with it locally.

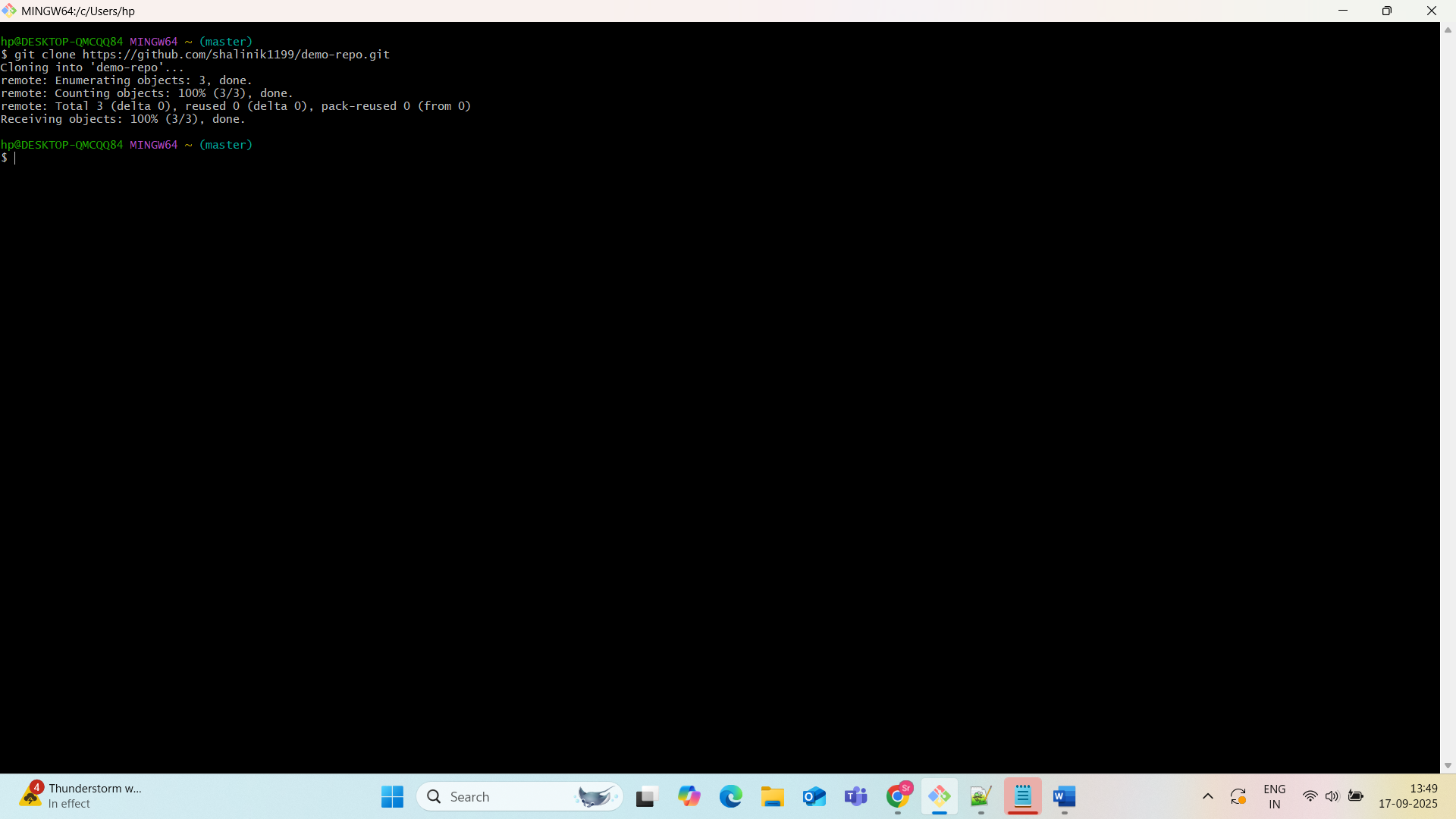
**Command:**

git clone https://github.com/your-username/demo-repo.git

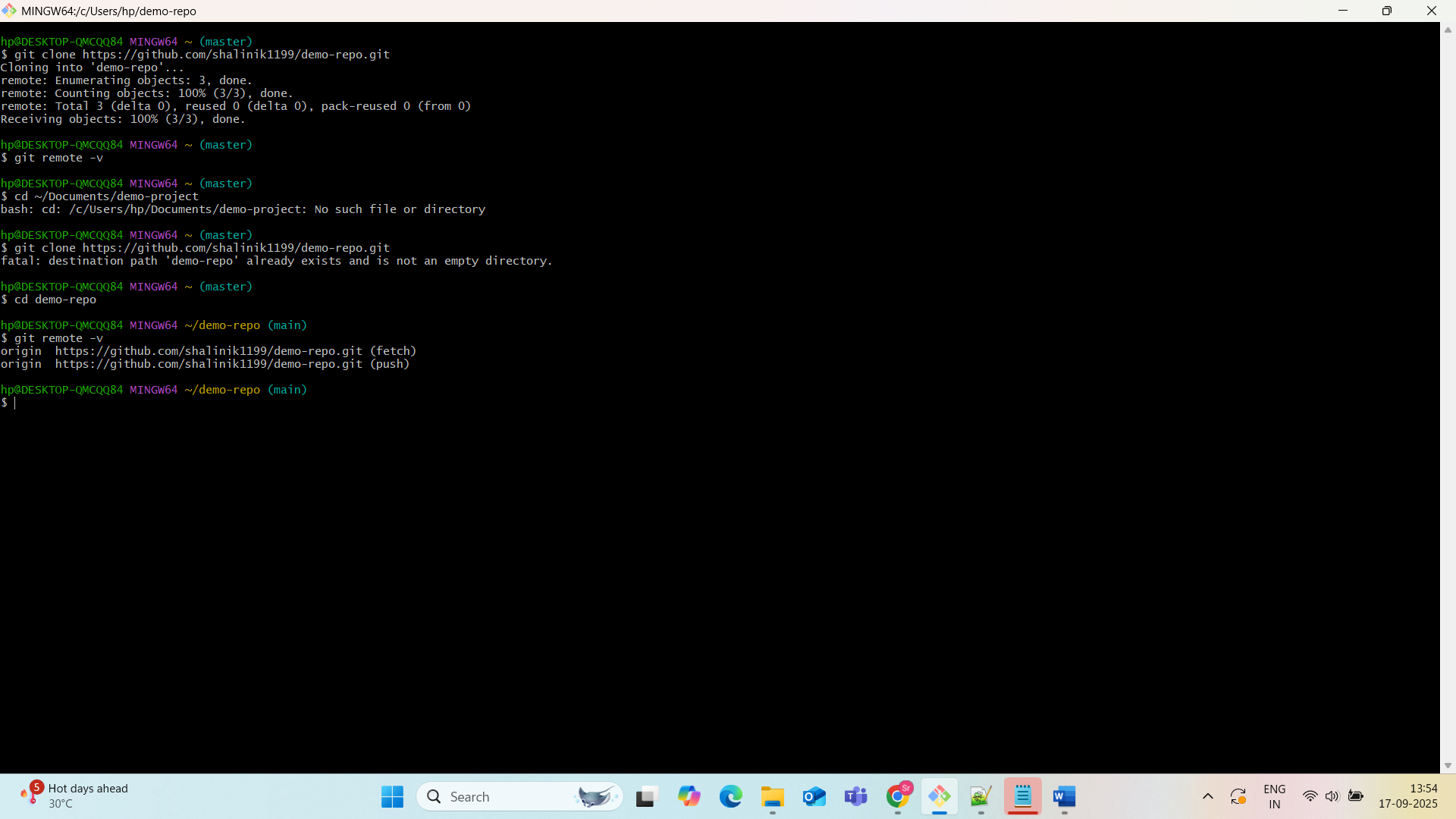
cd demo-repo

**What this does:**

* Connects to GitHub
* Downloads all files, branches, history
* Creates a folder demo-repo/ in your current directory



To check:git remote -v: Output:



**3. Create two files in the local repo**

**Command:**

To create two new files within a local Git repository and prepare them for tracking, follow these steps: Navigate to the Local Repository.

Open your terminal or command prompt and use the cd command to navigate to the root directory of your local Git repository.

Code

cd demo-repo

Create the Files.

Create the two new files using a text editor or the touch command. For example:

Code

touch demofile1.txt  
 touch demofile2.txt

You can then add content to these files using a text editor of your choice. Stage the Files.

Inform Git about the new files by staging them. This adds them to the staging area, preparing them for the next commit. You can stage individual files or all untracked files in the current directory.

To stage individual files:

Code

git add demofile1.txt  
 git add demofile2.txt

To stage all new and modified files in the current directory:

Code

git add .

Verify Staged Files (Optional).

You can check the status of your repository to confirm that the files have been staged correctly.

Code

git status

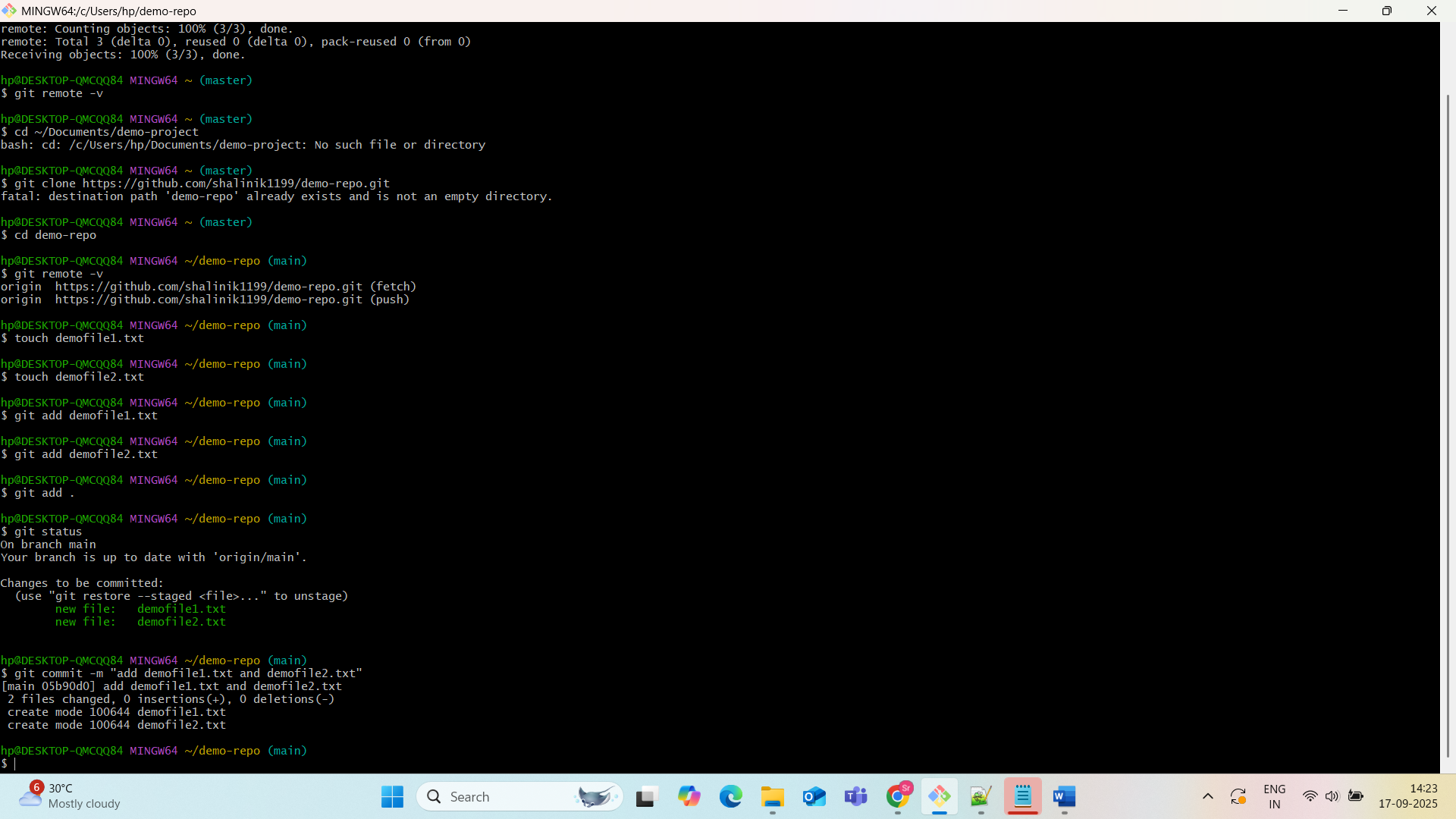
The output should show new file: demofile1.txt and new file: demofile2.txt under "Changes to be committed". Commit the Files.

Create a commit to permanently save the staged files into the repository's history. Provide a descriptive commit message explaining the changes.

Code

git commit -m "Add demofile1.txt and demofile2.txt"

This completes the process of creating and committing the new files to your local Git repository.



4. **Commit two files and push to central repository**

To commit two files and push them to a central repository, follow these detailed steps:

* **Ensure Local Repository is Up-to-Date (Optional but Recommended):**

Before making changes, it is good practice to pull the latest changes from the central repository to avoid merge conflicts.

Code

git pull origin <branch-name>

Replace <branch-name> with the name of your current branch (e.g., main or master). Modify or Create Files.

Make the necessary changes to your two files or create new ones within your local Git repository's working directory. Stage the Changes.

Add the modified or newly created files to the Git staging area. This prepares them for the commit.

Code

git add file1.txt file2.txt

Replace file1.txt and file2.txt with the actual names of your files. Alternatively, to stage all changes in the current directory and its subdirectories, use:

Code

git add .

Commit the Staged Changes.

Commit the staged changes to your local repository with a descriptive commit message.

Code

git commit -m "Descriptive message about the changes made to file1 and file2"

The commit message should clearly explain what changes were made in this commit. Push to the Central Repository.

Push your local commits to the remote central repository.

Code

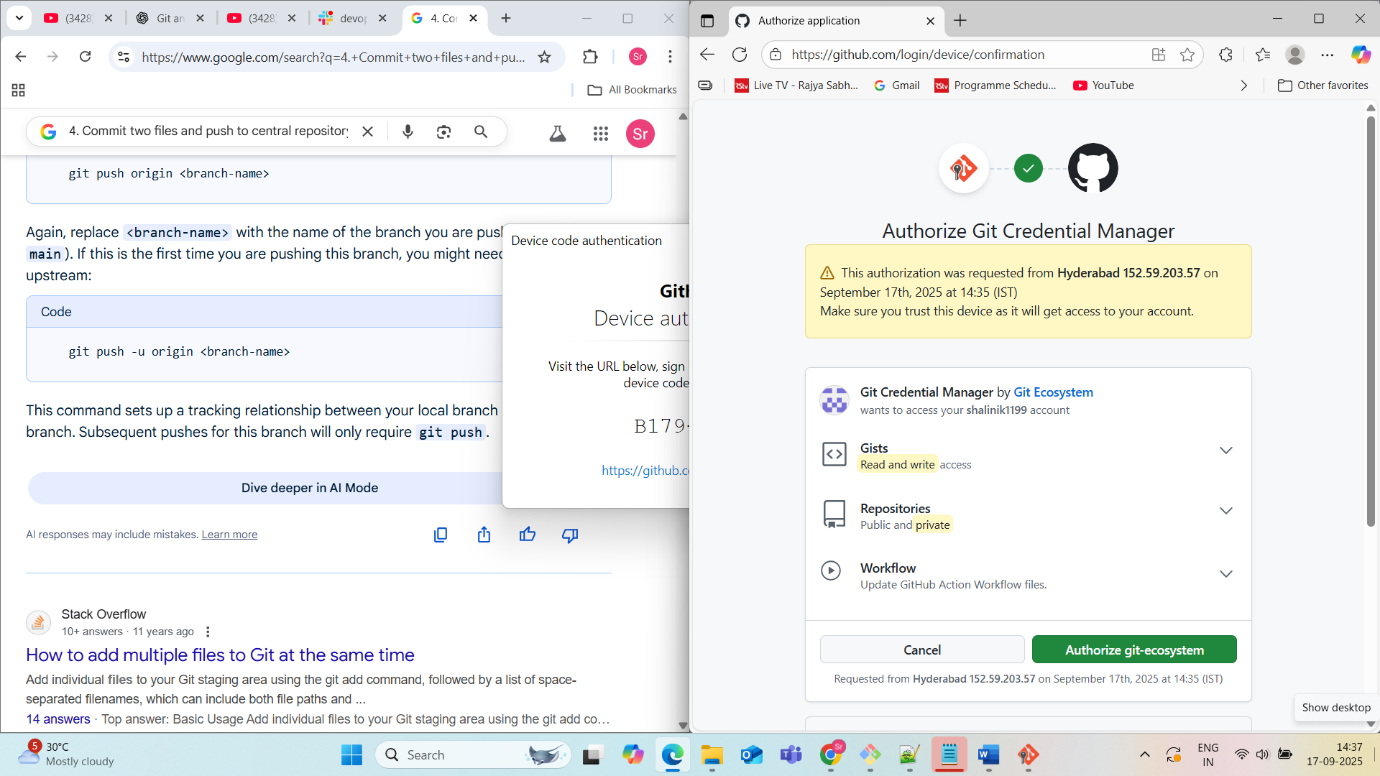
git push origin <branch-name>

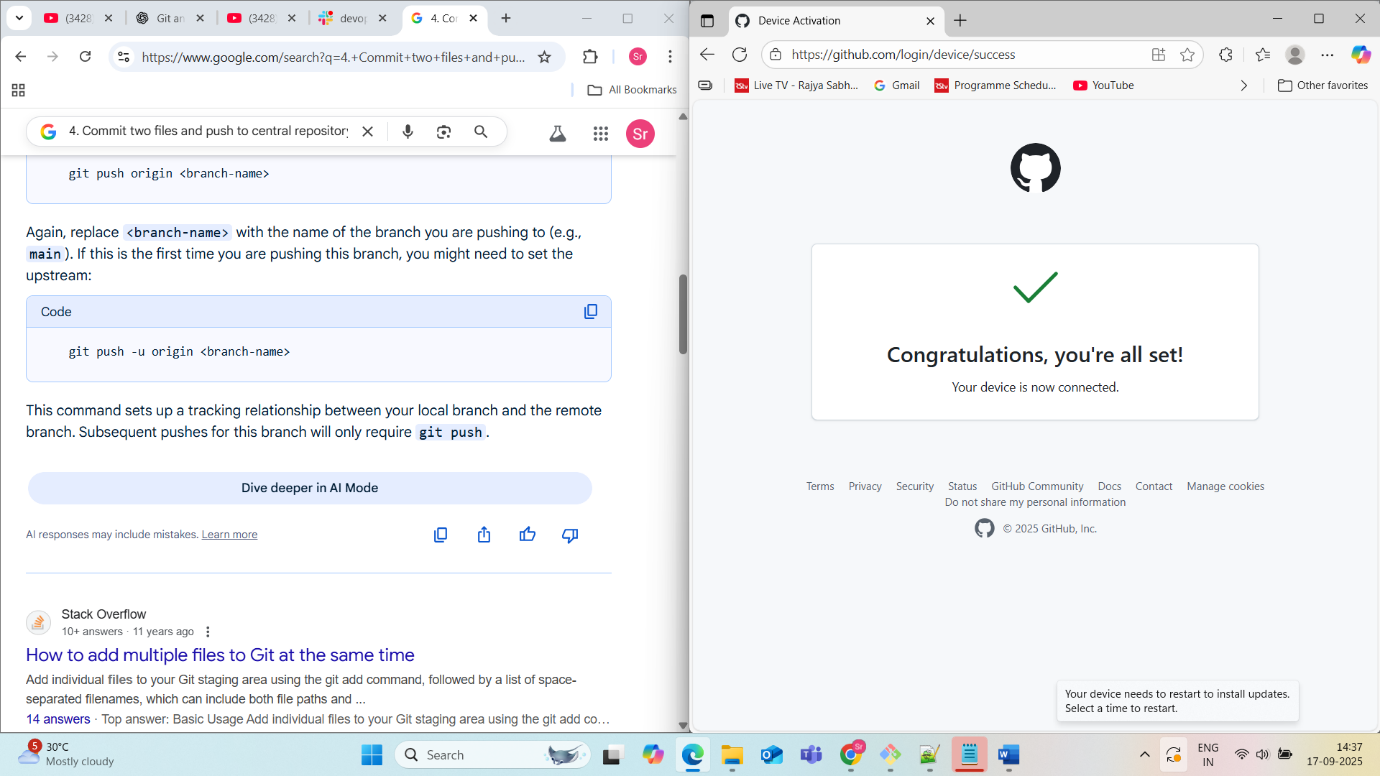
Again, replace <branch-name> with the name of the branch you are pushing to (e.g., main). If this is the first time you are pushing this branch, you might need to set the upstream:

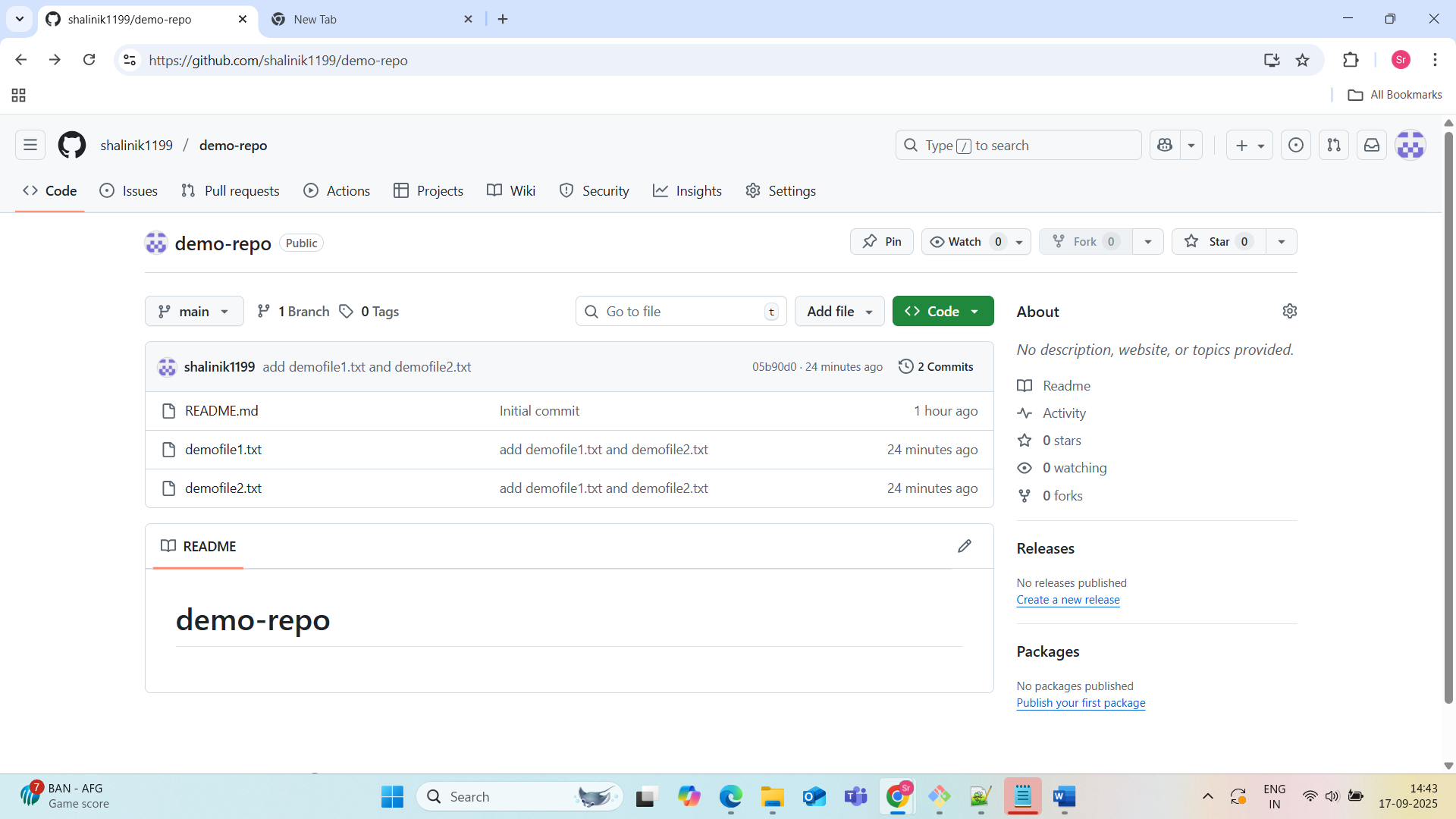
Code

git push -u origin <branch-name>

This command sets up a tracking relationship between your local branch and the remote branch. Subsequent pushes for this branch will only require git push.







**5.Create a branch in local, create a sample file, and push**

✅ Create a branch in your **local Git repo**,  
✅ Add a **sample file**,  
✅ Push both the **branch and the file to GitHub**.

**Prerequisites:**

* You already have a **Git repo cloned locally** (e.g., demo-repo)
* You are inside that repo folder in your terminal:

cd path/to/demo-repo

**Step 1: Check Current Branch**

Before creating a new branch, check your current branch (usually main or master):

git branch

Output example:

\* main

The \* indicates your current branch.

**Step 2: Create and Switch to a New Branch**

git checkout -b my-feature-branch

checkout -b: both **creates** and **switches** to the new branch

* my-feature-branch: name your branch (you can use any meaningful name)

Output:

Switched to a new branch 'my-feature-branch'

You can verify:

git branch

Output:

main

\* my-feature-branch

**Step 3: Create a Sample File**

You can create a sample file using the echo command or any text editor.

echo "This is a sample feature file" > feature.txt

**Step 4: Stage the File for Commit**

git add feature.txt

This adds feature.txt to the staging area, preparing it to be committed.

**Step 5: Commit the File**

git commit -m "Add feature.txt to my-feature-branch"

Output:[my-feature-branch abc1234] Add feature.txt to my-feature-branch

1 file changed, 1 insertion(+)

create mode 100644 feature.txt

**Step 6: Push the New Branch to GitHub**

git push origin my-feature-branch

* Sends the my-feature-branch and its commits to GitHub
* Doesn’t affect the main branch

Output:

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

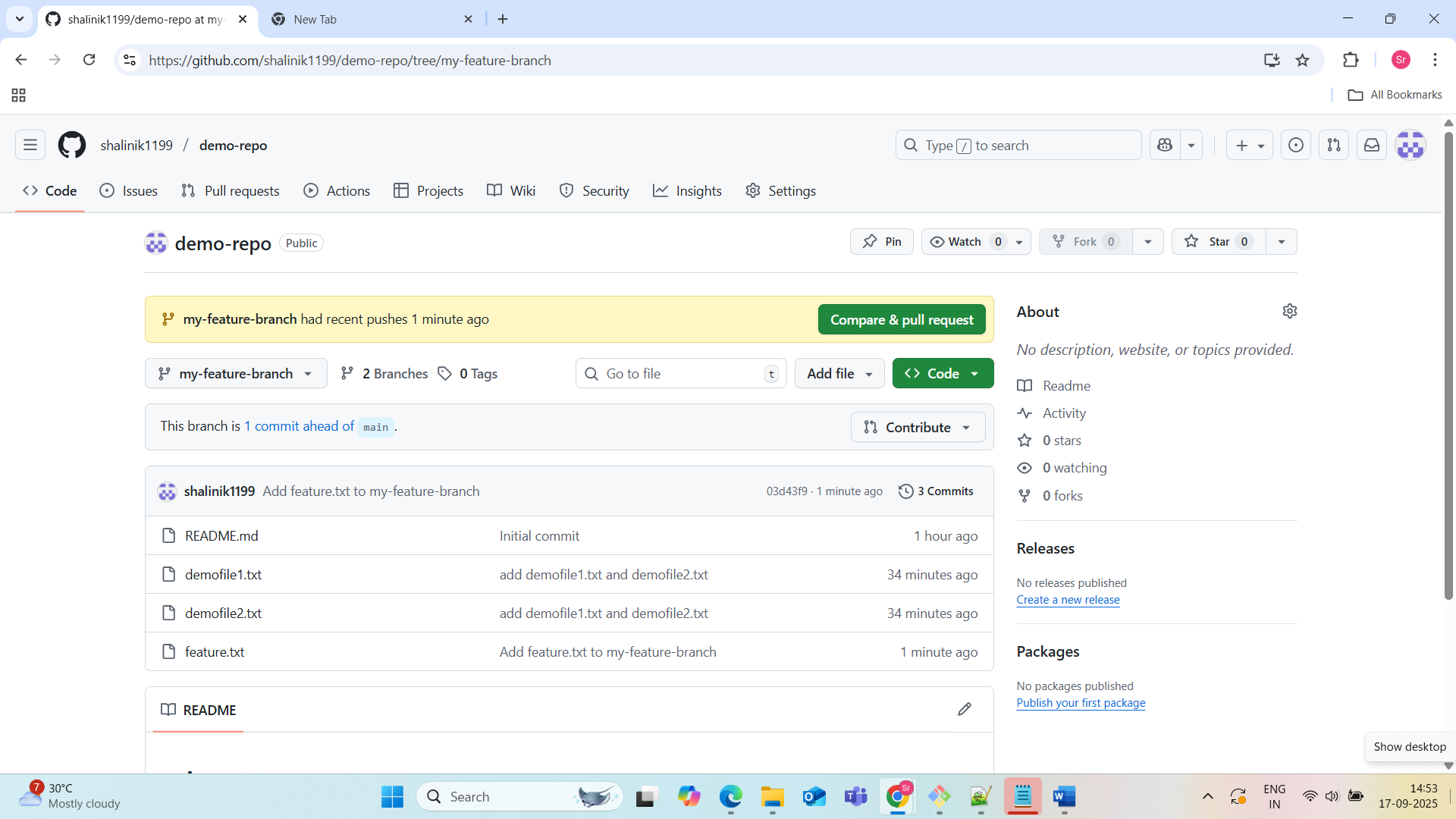
...

To https://github.com/your-username/demo-repo.git

\* [new branch] my-feature-branch -> my-feature-branch

**Step 7: Verify on GitHub**

1. Go to your repo on GitHub
2. Click the <> Code tab
3. Click the **branch dropdown** (top-left) — you’ll see my-feature-branch listed
4. Click it to switch and view the new file feature.txt



**6. Create a branch in GitHub and clone that to local**

Creating a Branch in GitHub

* **Navigate to your Repository:**

Open your web browser and go to the main page of your repository on GitHub.

* **Access Branch Options:**

Above the list of files, locate the branch dropdown menu (usually displaying "main" or "master"). Click on it.

* **Create New Branch:**

In the dropdown menu, type the desired name for your new branch in the text field (e.g., feature/my-new-feature).

* **Select Source (Optional):**

By default, the new branch will be created from the currently selected branch (e.g., main). If you want to create it from a different branch, select it from the "Branch source" dropdown.

* **Confirm Creation:**

Press Enter or click "Create branch" to finalize the creation of your new branch on GitHub.

Cloning the Branch to Local

* **Copy Repository URL:** On the main page of your GitHub repository, click the "Code" button. Choose either HTTPS or SSH and copy the provided URL.
* **Open Terminal/Command Prompt:** Open your preferred terminal or command prompt on your local machine.
* **Navigate to Desired Directory:** Use the cd command to navigate to the directory where you want to clone the repository.

Code

cd /path/to/your/desired/directory

* **Clone the Specific Branch:** Use the git clone command with the -b flag to specify the branch you want to clone.

Code

git clone <https://github.com/username/repository.git>

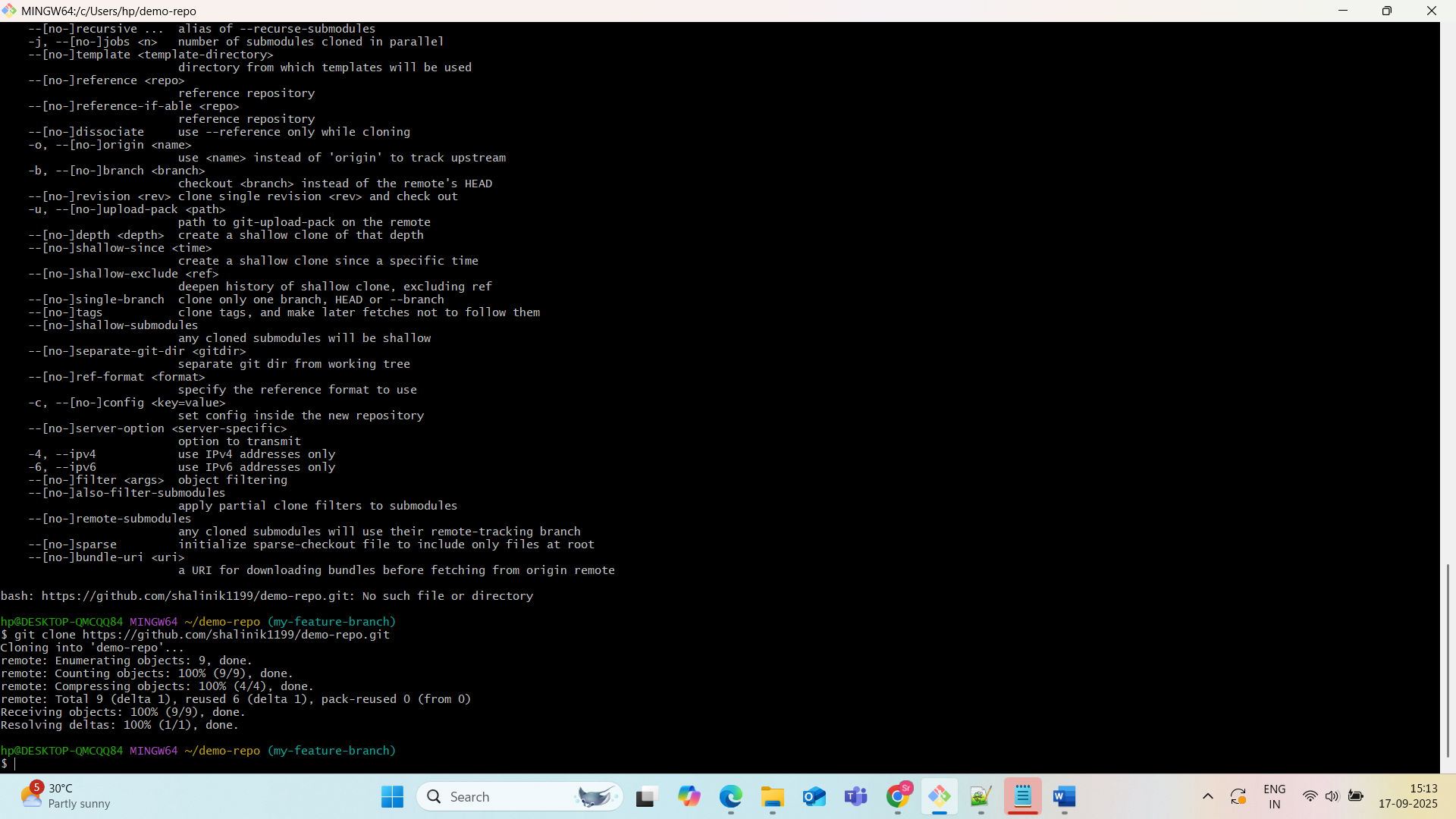
Replace your-branch-name, your-username, and your-repository-name with the actual values.

* **Verify Branch:** Navigate into the newly cloned repository directory and use the git branch command to confirm that you are on the correct branch.

Code

cd your-repository-name  
 git branch

The output will show your currently active branch, which should be the one you specified during cloning.

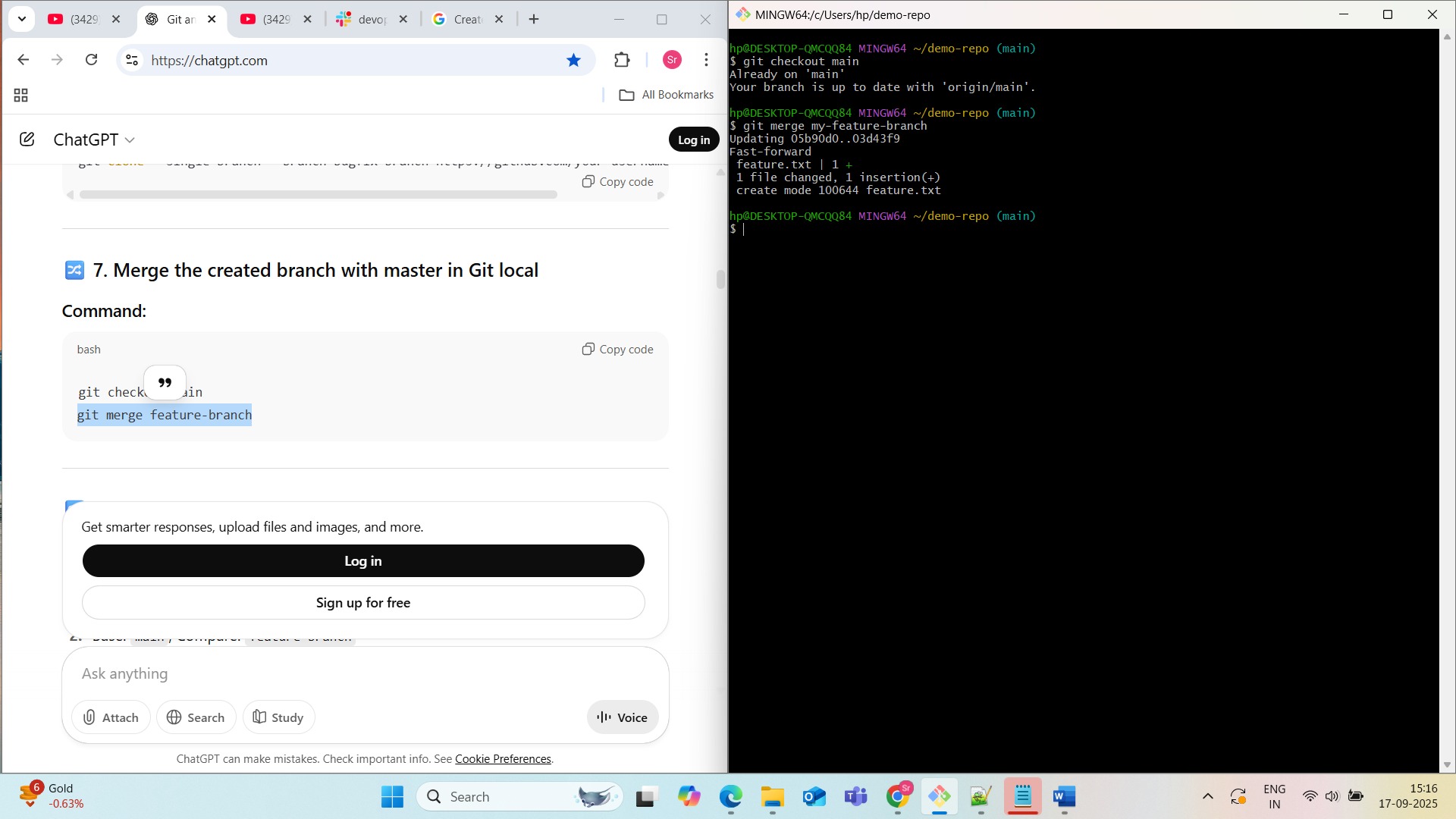


**7. Merge the created branch with master in Git local**

**Command:**

git checkout main

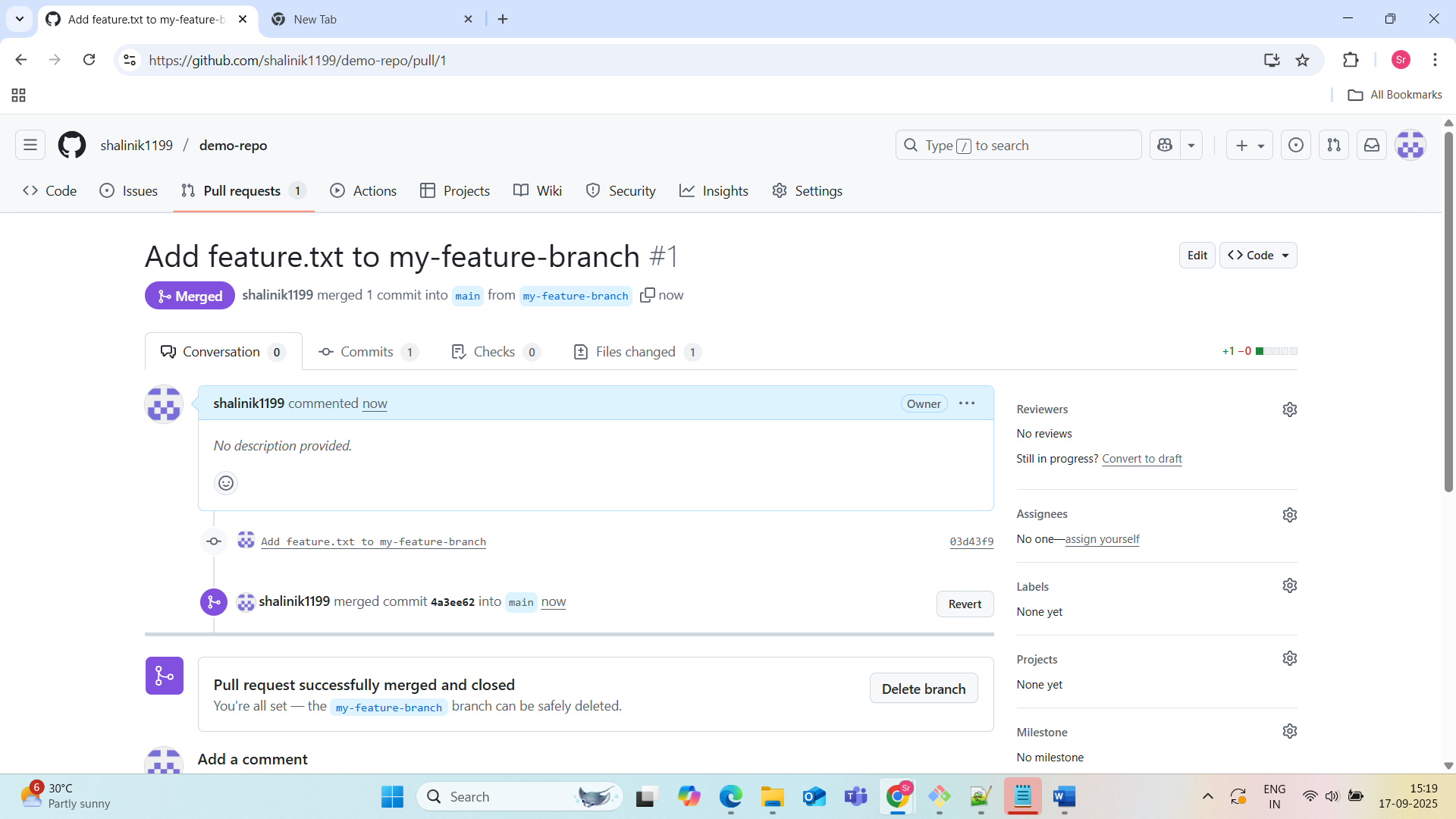
git merge my-feature-branch



**8. Merge the created branch with master in GitHub via Pull Request**

**GitHub Steps:**

1. Go to Pull Requests > New Pull Request
2. Base: main, Compare: feature-branch
3. Click Create Pull Request
4. Add details > Click Merge Pull Request



**9. Create a file in local and send it to branch in GitHub**

**Command:**

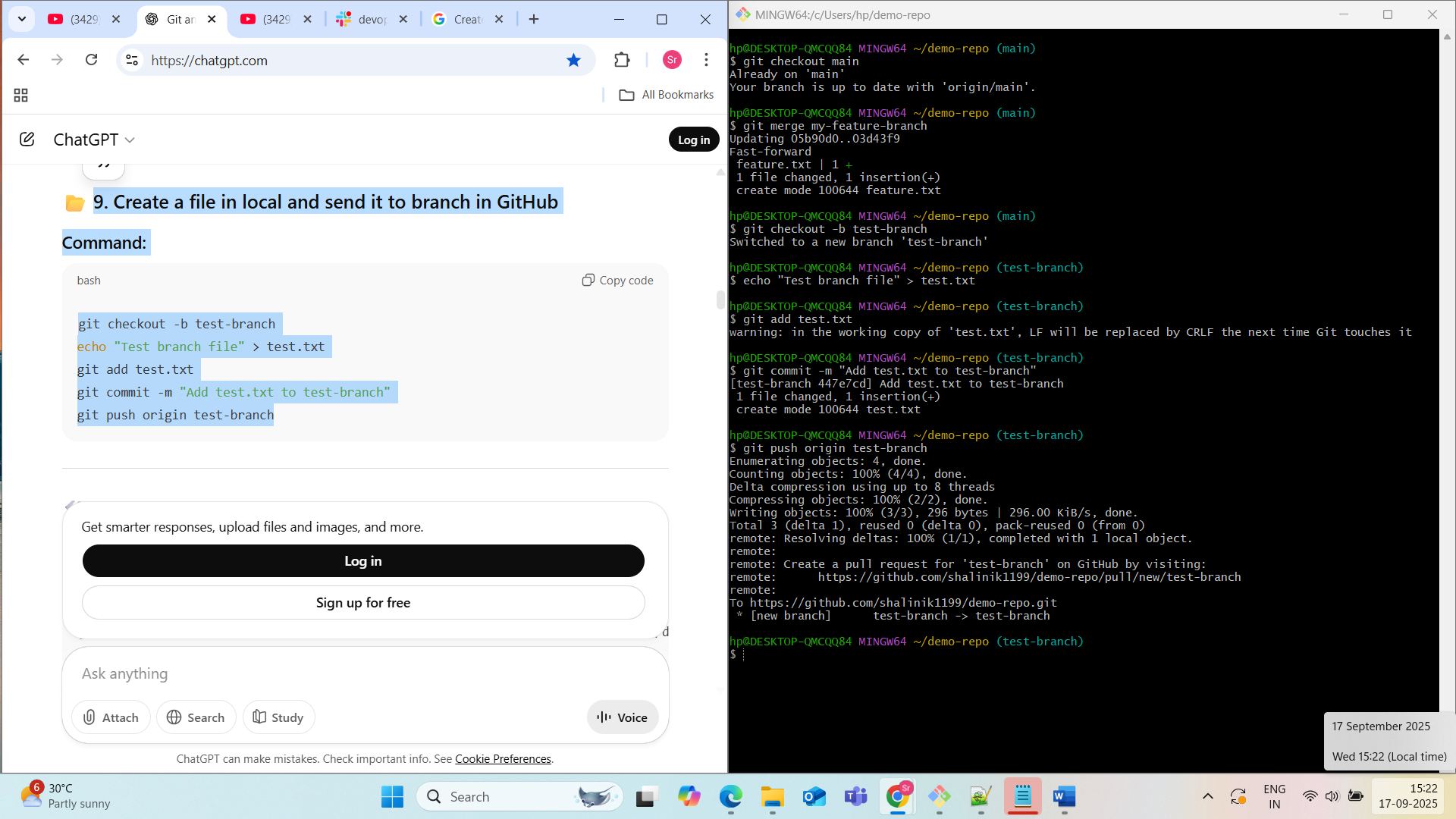
git checkout -b test-branch

echo "Test branch file" > test.txt

git add test.txt

git commit -m "Add test.txt to test-branch"

git push origin test-branch



**10. Clone only a branch from GitHub to local**

Cloning only a specific branch from a GitHub repository to your local machine can be achieved using the git clone command with the -b and --single-branch flags. This method ensures that only the specified branch's history and files are fetched, making the local repository smaller and faster to clone.

Here is the detailed, step-by-step process:

1. Navigate to the desired directory:
2. Get the repository URL:

3. Clone the specific branch:

Execute the git clone command with the following options:

* -b <branch\_name>: Specifies the branch you want to clone.
* --single-branch: Ensures that only the history and files of the specified branch are fetched, not all branches.

Code

git clone -b <branch\_name> --single-branch <repository\_url>

4. Verify the cloned branch (Optional):

Navigate into the newly cloned repository directory and use git branch to confirm that only your desired branch is present locally.

Code

cd your-repository-name  
git branch

This will show you the local branches, which should only include the branch you cloned.

**Step 1: Copy the GitHub Repository URL**

1. Go to your GitHub repository (e.g., demo-repo)
2. Click the green **Code** button
3. Copy the **HTTPS** link:

Step 2: Run the git clone Command

git clone --branch <branch-name> --single-branch <repo-url>

**Step 3: Enter the Cloned Repository**

cd demo-repo

You are now in a local repo that **only contains the test-branch**.

**Step 4: Verify the Branch**

git branch

Output:

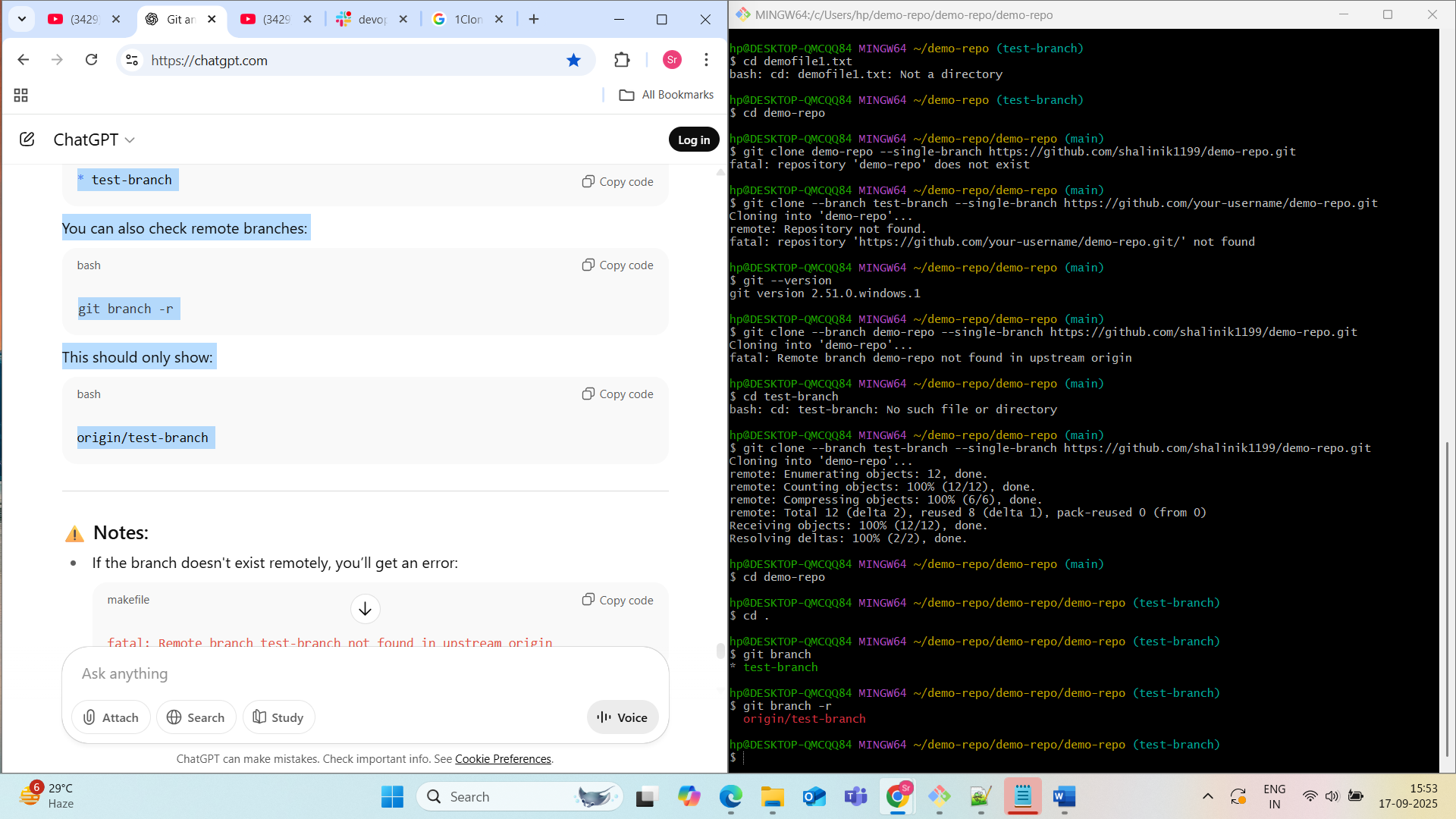
\* test-branch

You can also check remote branches:

git branch -r

This should only show:

origin/test-branch



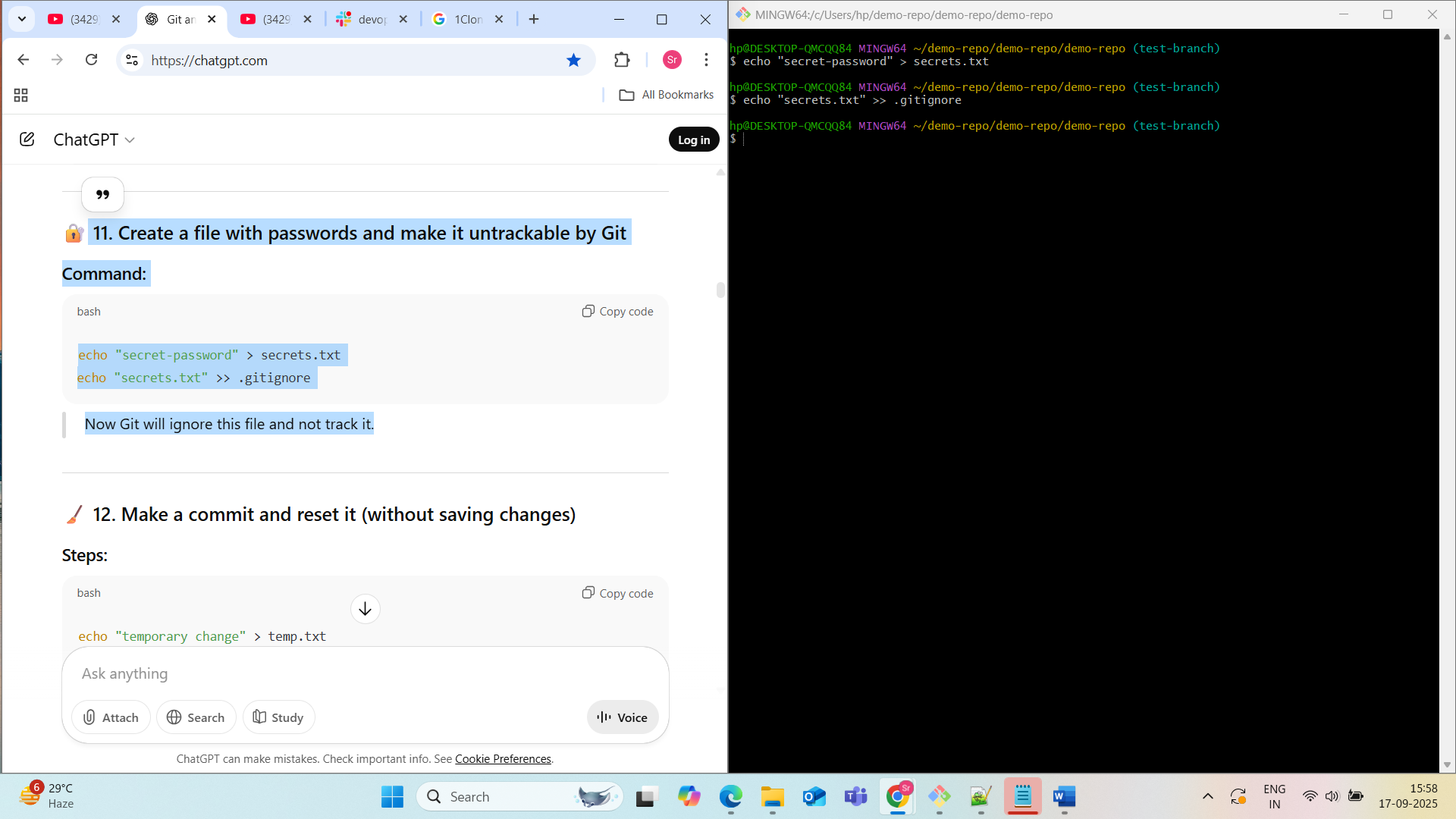
**11. Create a file with passwords and make it untrackable by Git**

**Command:**

echo "secret-password" > secrets.txt

echo "secrets.txt" >> .gitignore

Now Git will ignore this file and not track it.



**12. Make a commit and reset it (without saving changes)**

**Steps:**

echo "temporary change" > temp.txt

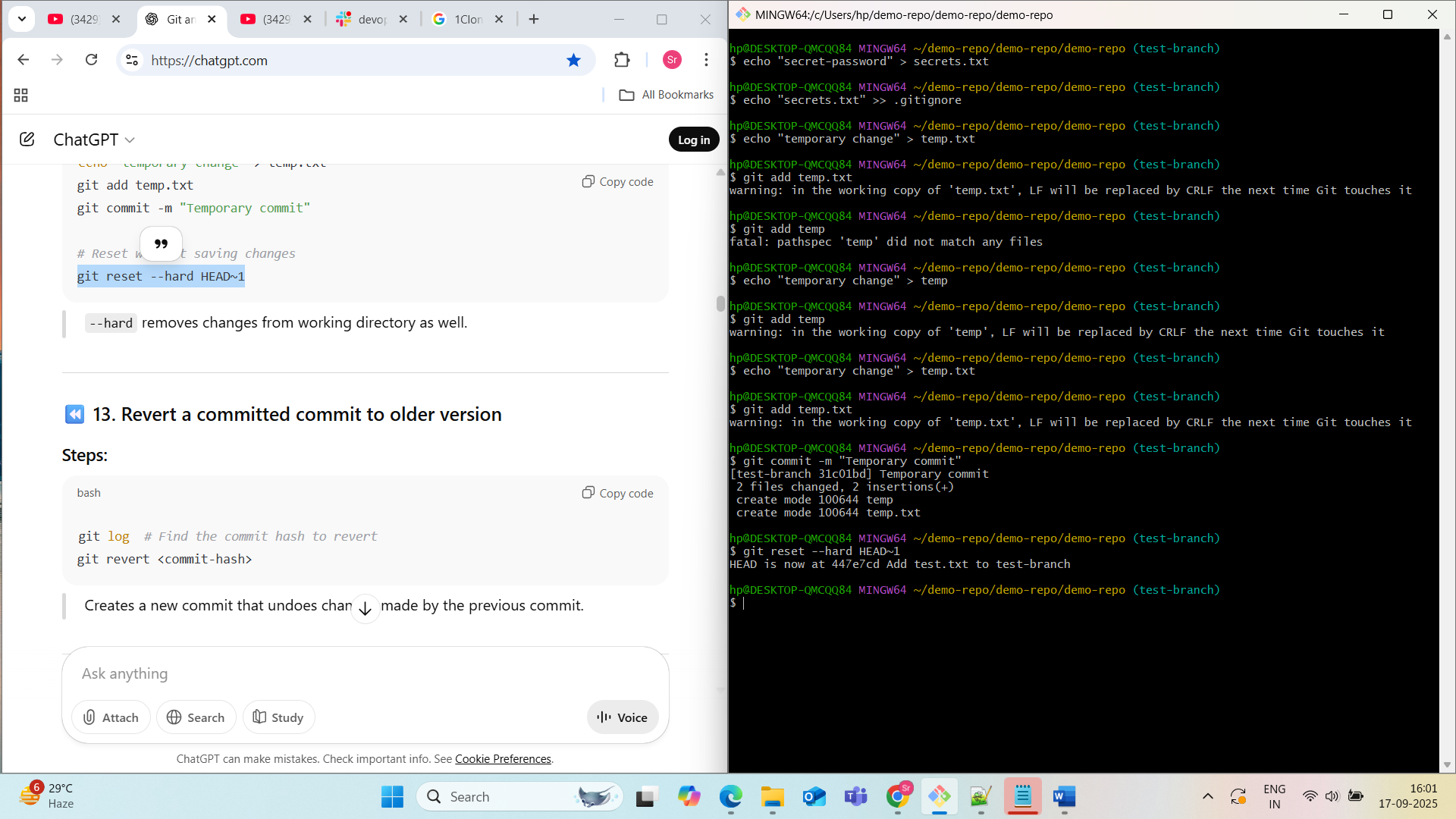
git add temp.txt

git commit -m "Temporary commit"

# Reset without saving changes

git reset --hard HEAD~1

--hard removes changes from working directory as well.



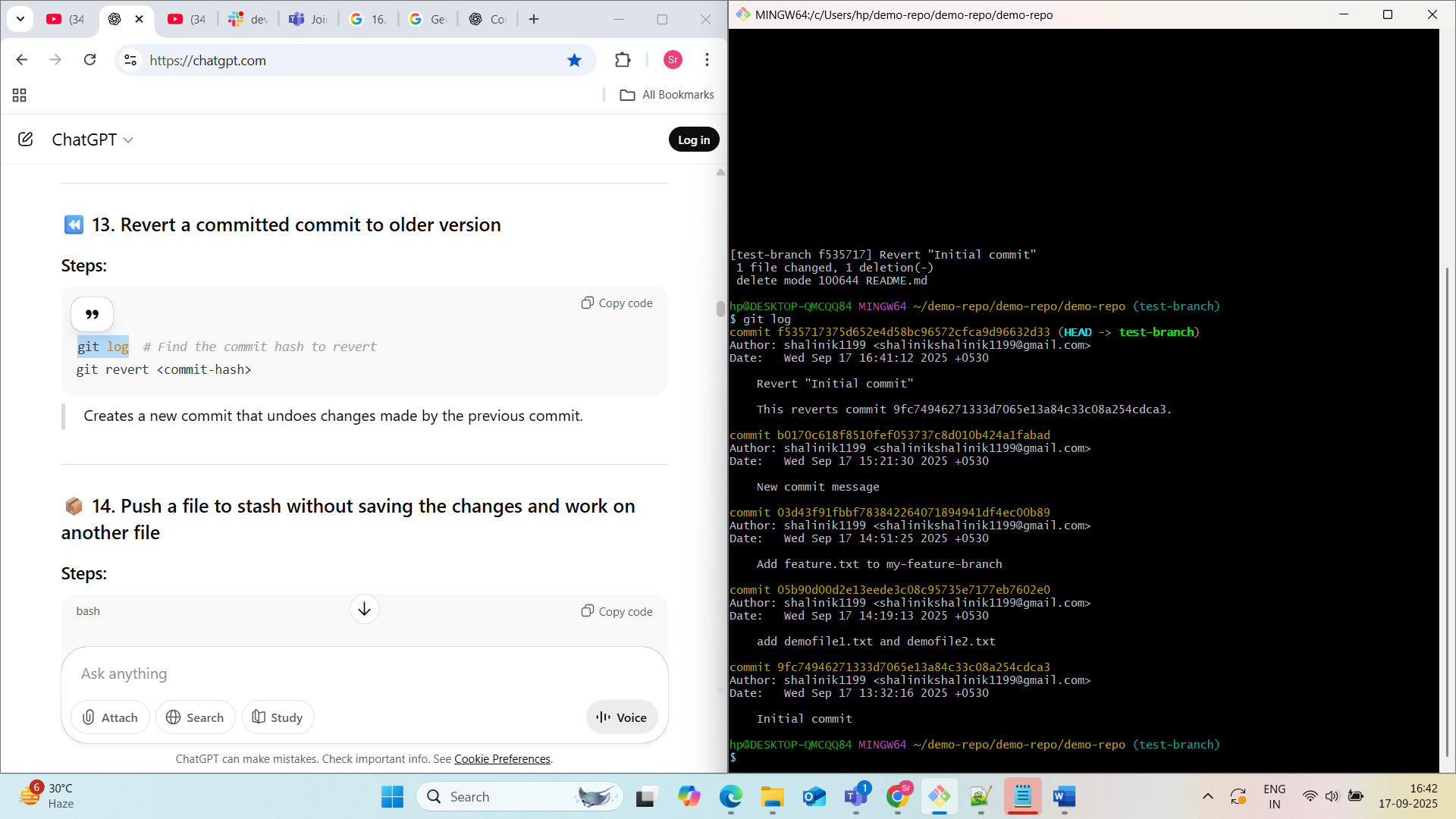
**13. Revert a committed commit to older version**

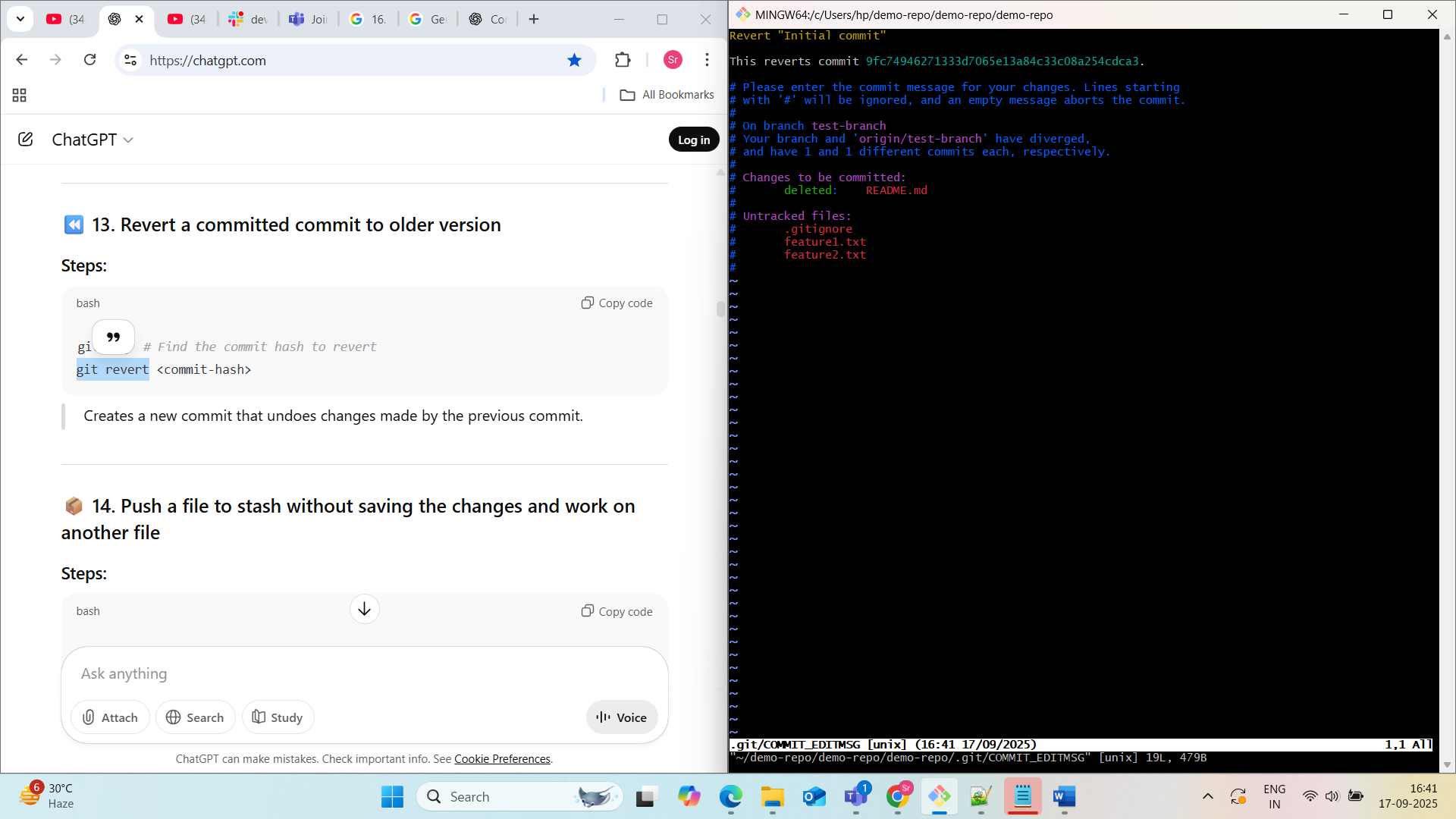
**Steps:**

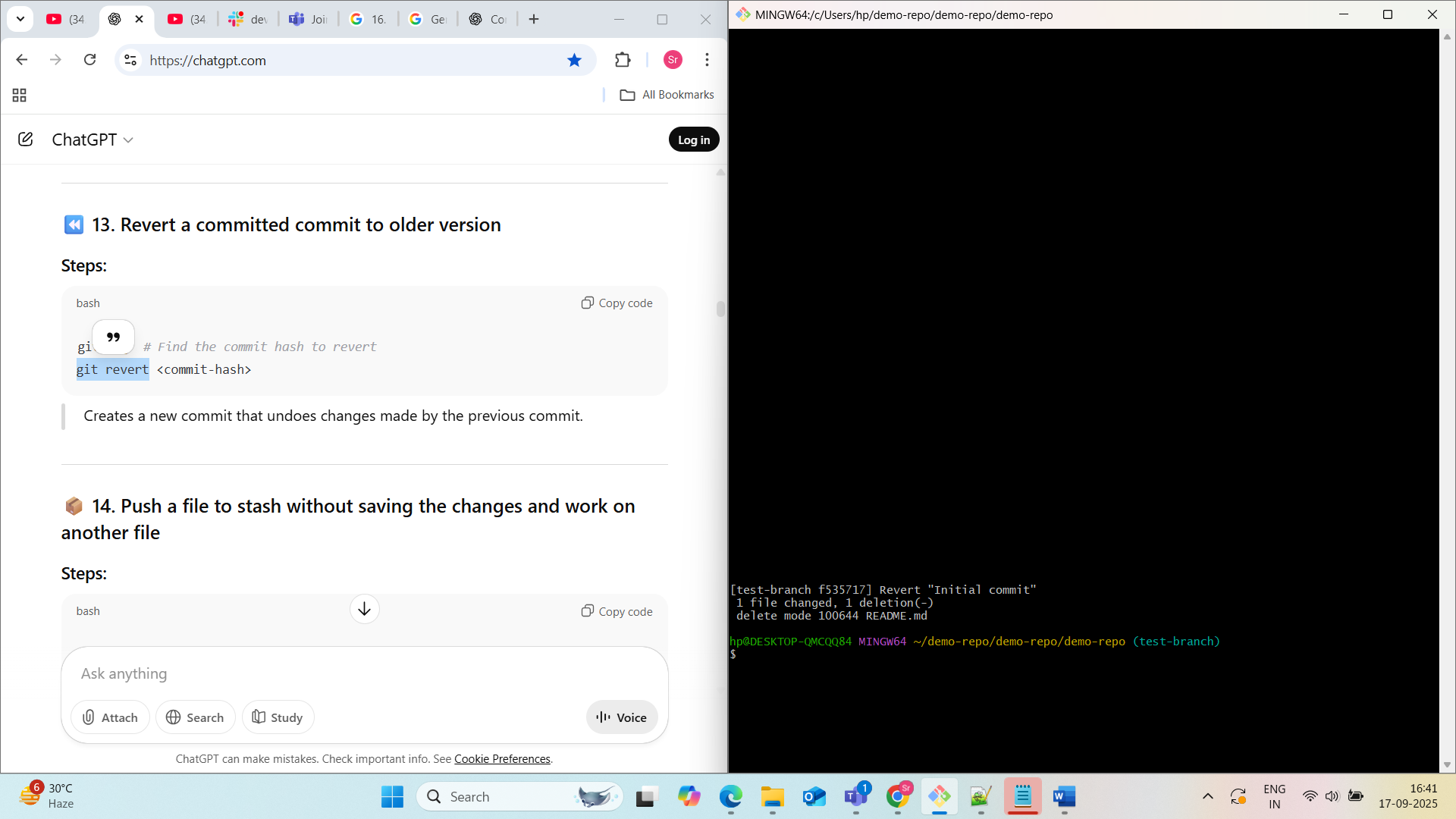
git log # Find the commit hash to revert

git revert <commit-hash>

Creates a new commit that undoes changes made by the previous commit.







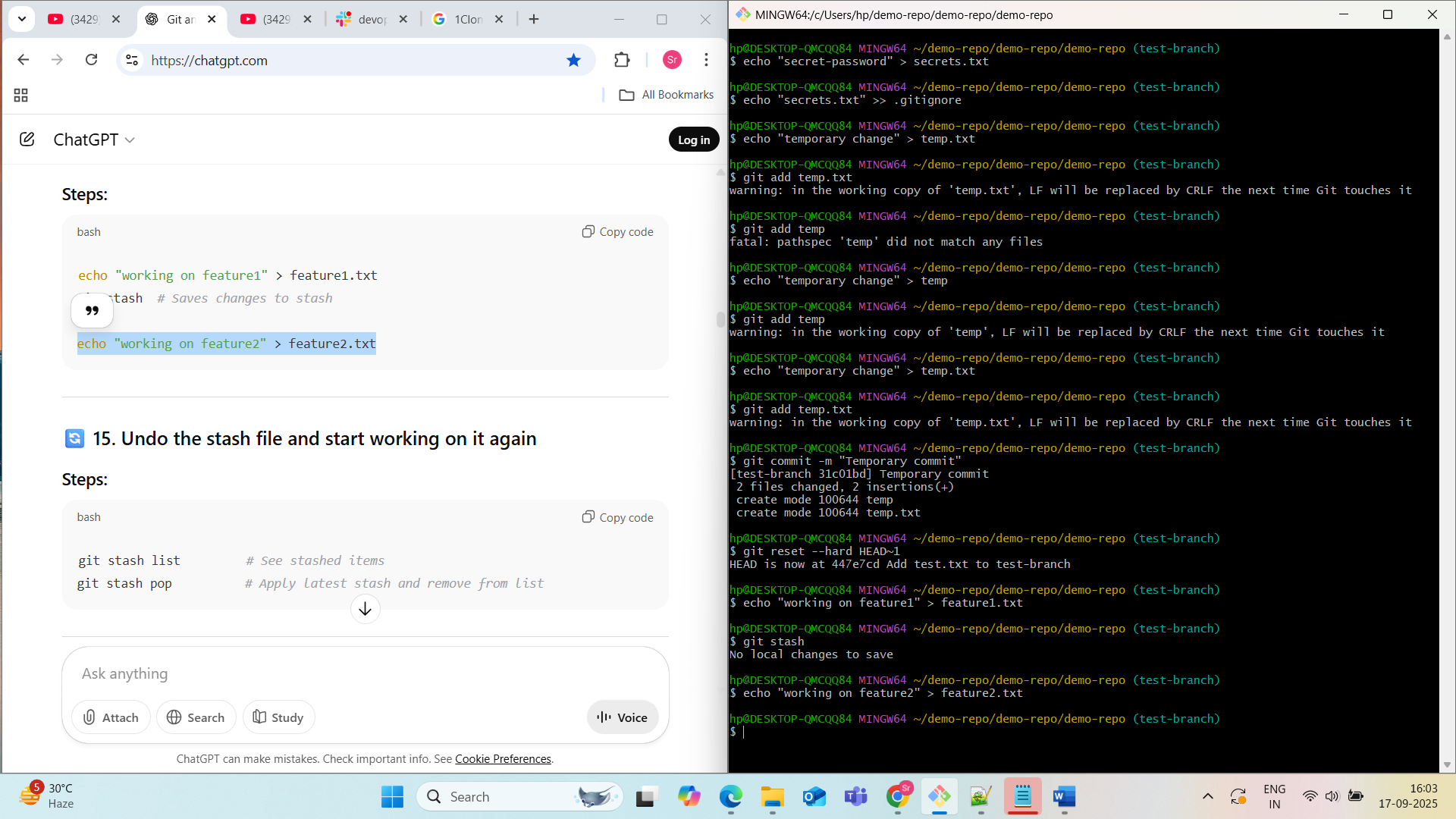
**14. Push a file to stash without saving the changes and work on another file**

**Steps:**

echo "working on feature1" > feature1.txt

git stash # Saves changes to stash

echo "working on feature2" > feature2.txt

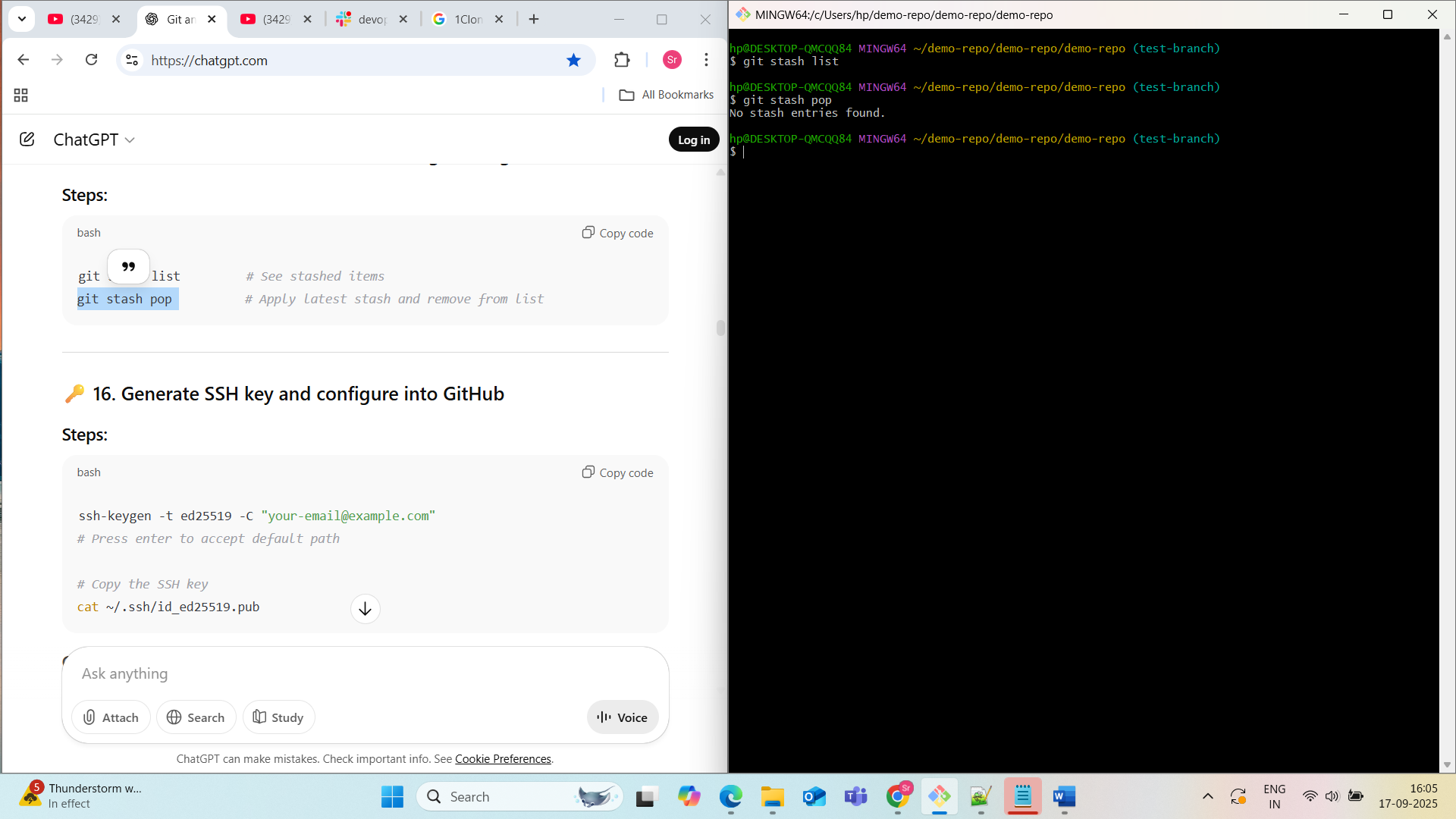


**15. Undo the stash file and start working on it again**

**Steps:**

git stash list # See stashed items

git stash pop # Apply latest stash and remove from list



**16. Generate SSH key and configure into GitHub**

**Steps:**

ssh-keygen -t ed25519 -C "your-email@example.com"

# Press enter to accept default path

# Copy the SSH key

cat ~/.ssh/id\_ed25519.pub

**GitHub:**

1. Go to **Settings** > **SSH and GPG keys**
2. Click **New SSH key**
3. Paste the copied key and save

(Or)

Generating an SSH key and configuring it with GitHub allows for secure and passwordless authentication when interacting with your repositories.

1. Generate an SSH Key Pair:

* Open your terminal Git Bash .
* Execute the following command, replacing your\_email@example.com with the email associated with your GitHub account:

Code

ssh-keygen -t ed25519 -C "your\_email@example.com"

* ed25519 is a secure, modern algorithm. If your system doesn't support it, you can use rsa -b 4096 instead.
* When prompted to "Enter a file in which to save the key," press Enter to accept the default location (~/.ssh/id\_ed25519).
* You will then be prompted to "Enter passphrase (empty for no passphrase):". You can either set a secure passphrase (recommended for added security) or leave it empty by pressing Enter twice.

2. Add the SSH Key to the SSH Agent (Optional but Recommended):

* Start the SSH agent in the background:

Code

eval "$(ssh-agent -s)"

* Add your private SSH key to the agent:

Code

ssh-add ~/.ssh/id\_ed25519

* If you used a different key name or algorithm, adjust the path accordingly.
* If you set a passphrase, you will be prompted to enter it here.

3. Add the Public SSH Key to GitHub:

* Copy your public SSH key to your clipboard.
  + **Windows (Git Bash):** cat ~/.ssh/id\_ed25519.pub | clip
* Navigate to GitHub and log in.
* Click your profile picture in the upper-right corner and select Settings.
* In the left sidebar, click SSH and GPG keys.
* Click the New SSH key or Add SSH key button.
* Provide a descriptive Title for your key (e.g., "My Laptop SSH Key").
* Paste the copied public key into the Key field.
* Click Add SSH key. You may need to confirm your GitHub password.

4. Test Your SSH Connection:

* In your terminal, execute:

17. **Configure webhooks to GitHub**

**18. Basic understanding of .git folder**

.git/ is a hidden folder that contains:

* **HEAD** – current branch reference
* **config** – repo config
* **objects/** – all commits, blobs, trees
* **refs/** – branches and tags
* **logs/** – history of refs
* It makes a repo a **Git repo**.

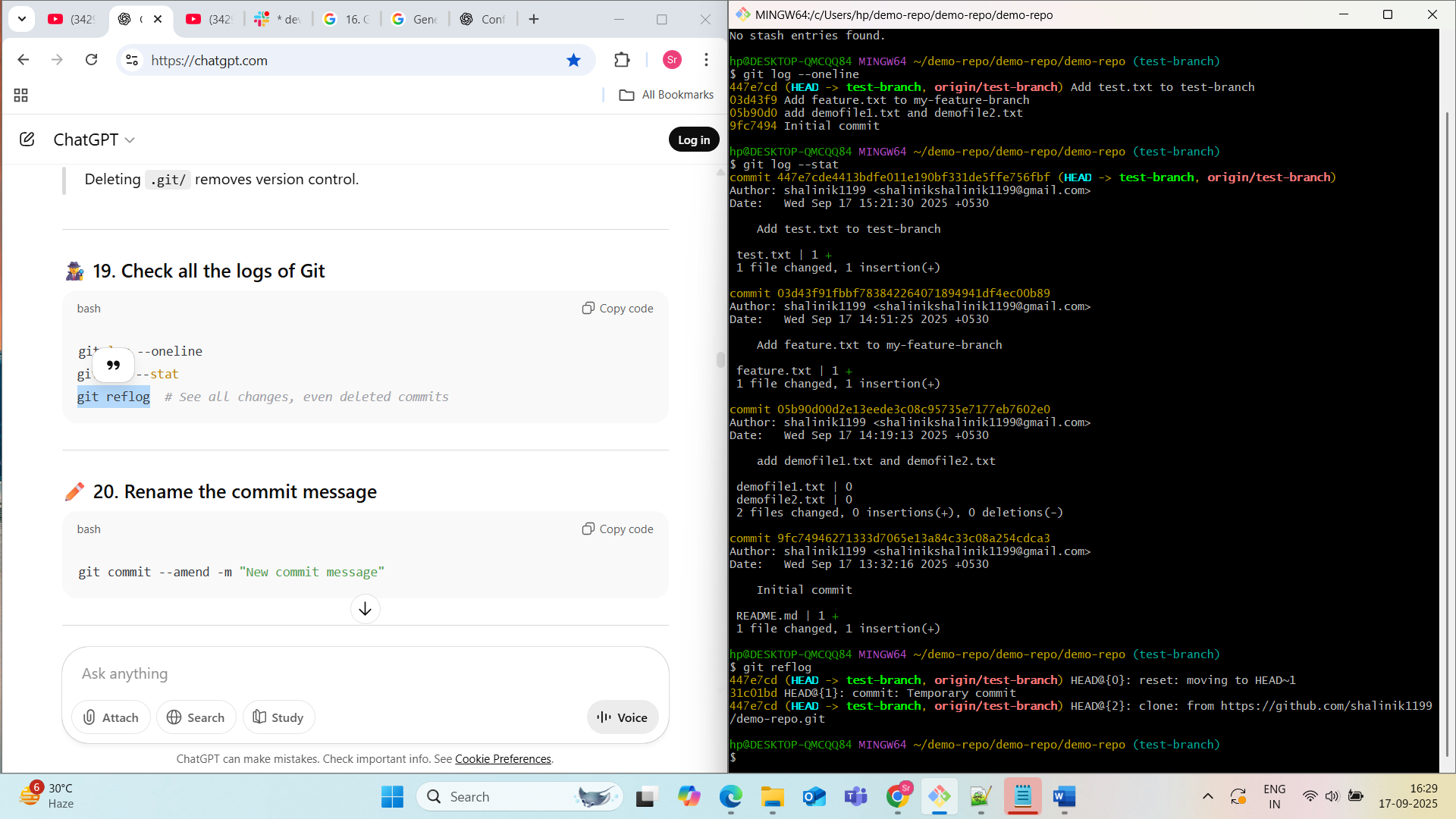
Deleting .git/ removes version control.

**19. Check all the logs of Git**

git log --oneline

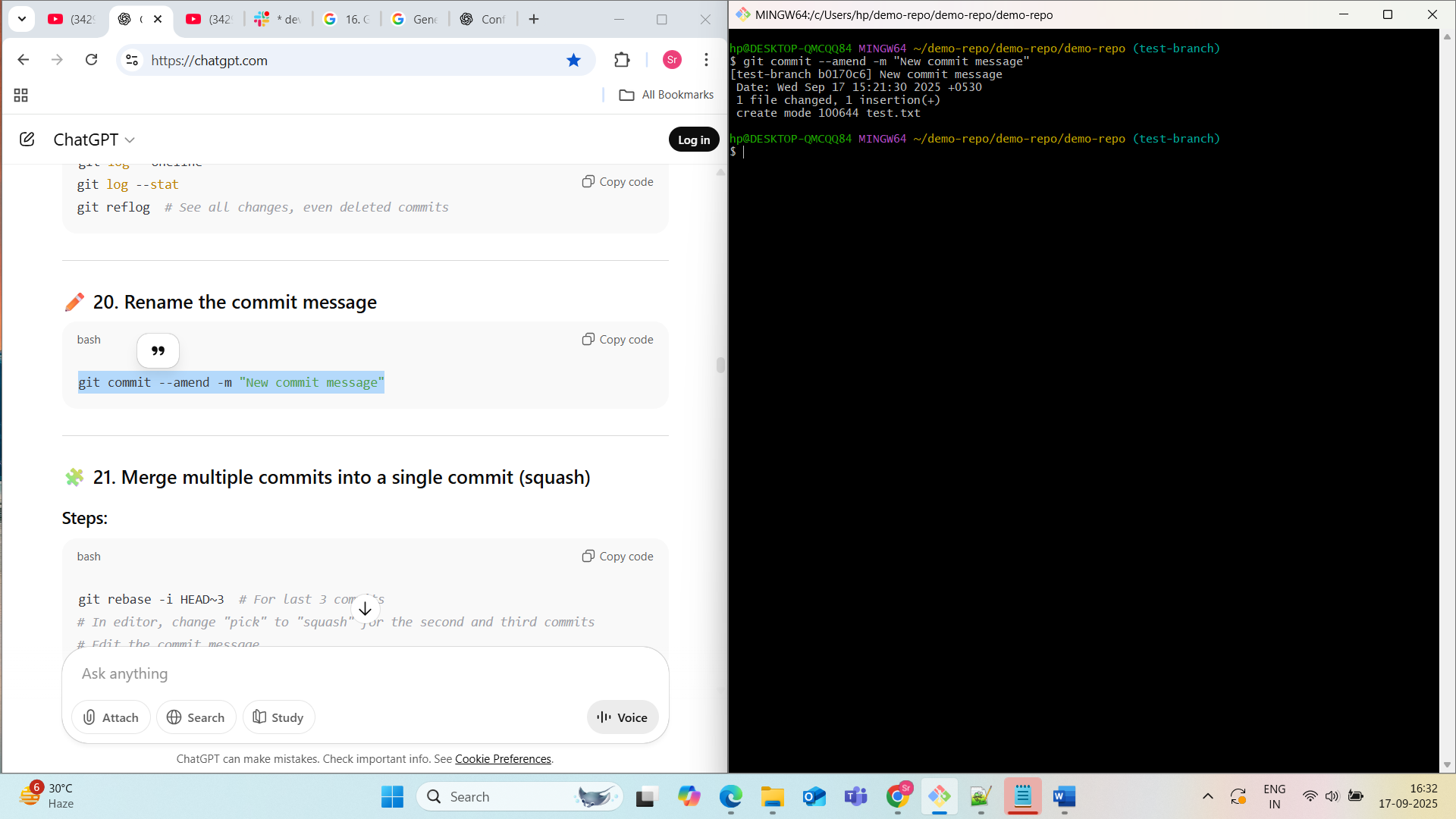
git log --stat

git reflog # See all changes, even deleted commits



**20. Rename the commit message**

git commit --amend -m "New commit message"



**21. Merge multiple commits into a single commit (squash)**

**Steps:**

git rebase -i HEAD~3 # For last 3 commits

# In editor, change "pick" to "squash" for the second and third commits

# Edit the commit message



