Git challenges

**1. Resolve Merge Conflicts**

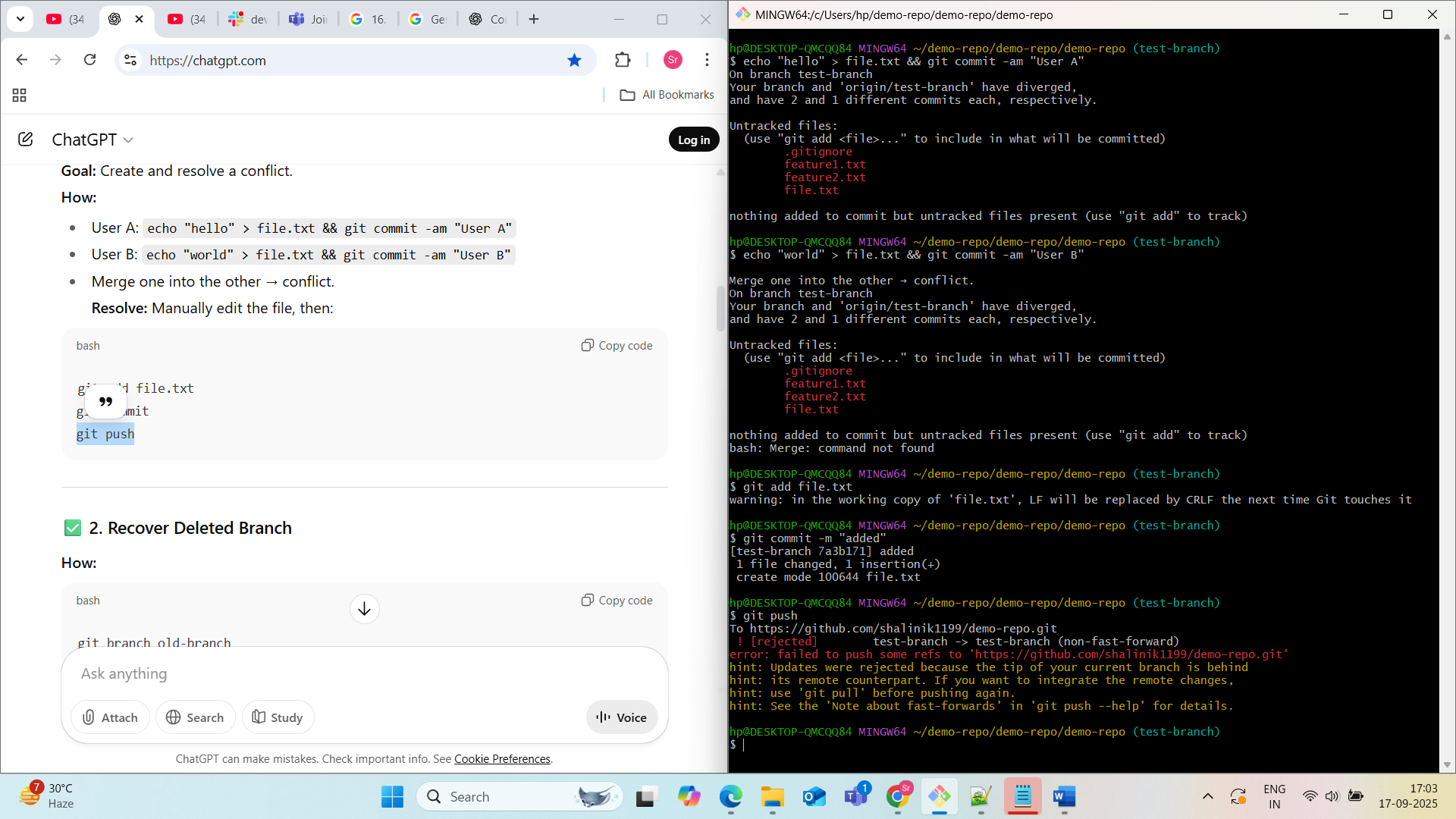
**Goal:** Create and resolve a conflict.  
**How:**

* User A: echo "hello" > file.txt && git commit -am "User A"
* User B: echo "world" > file.txt && git commit -am "User B"
* Merge one into the other → conflict.  
  **Resolve:** Manually edit the file, then:

git add file.txt

git commit

git push



**2. Recover Deleted Branch**

**How:**

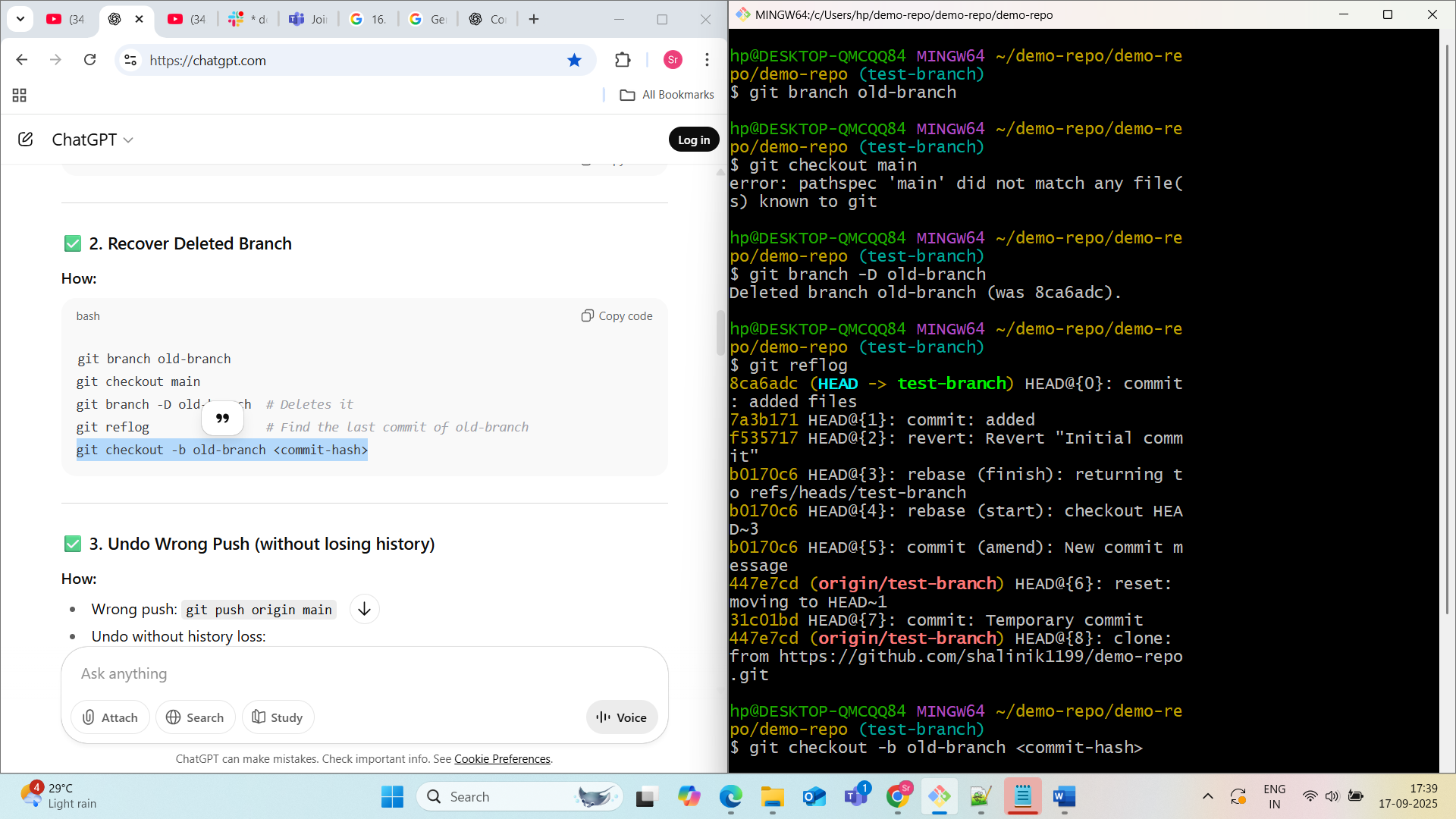
git branch old-branch

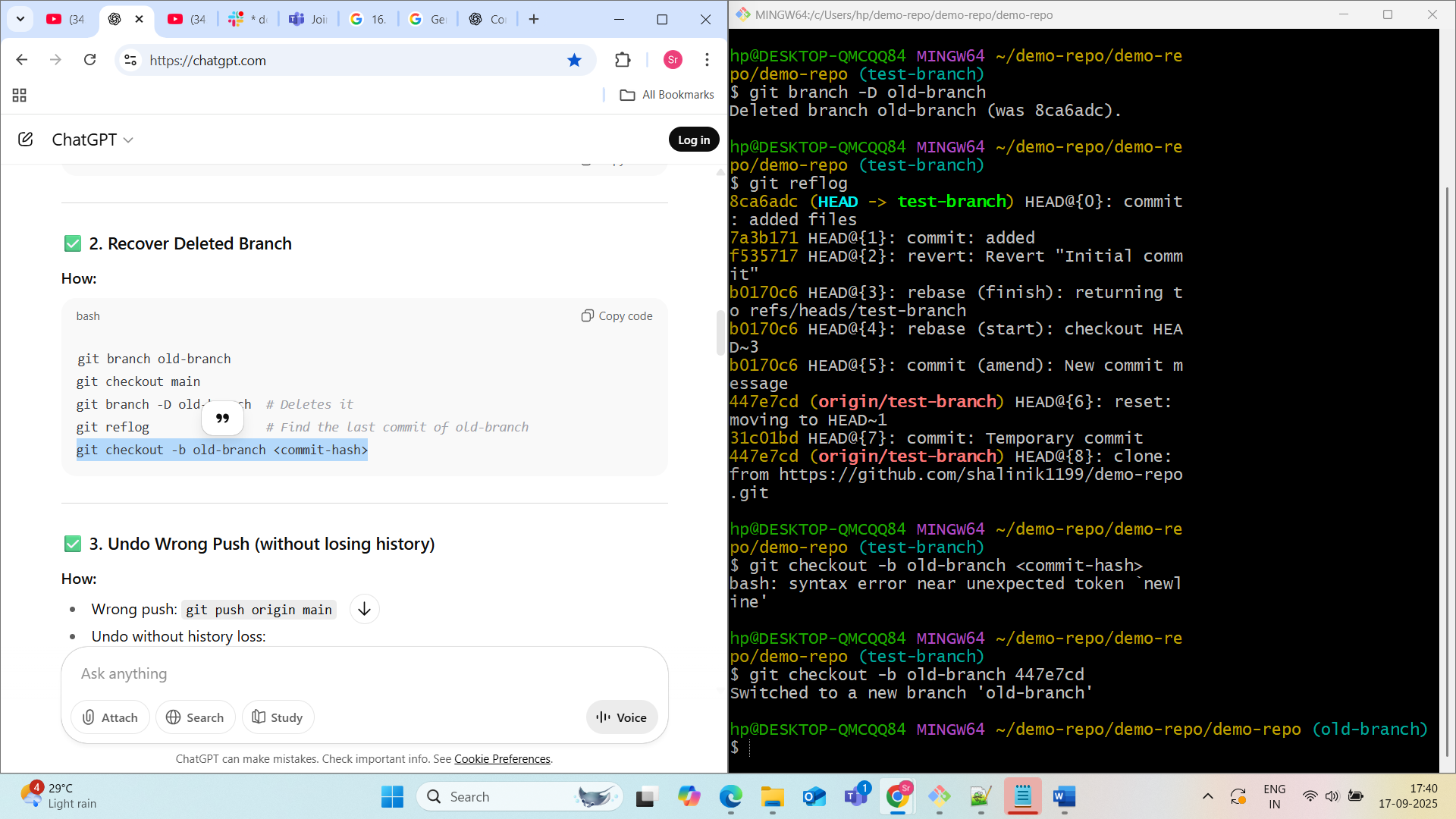
git checkout main

git branch -D old-branch # Deletes it

git reflog # Find the last commit of old-branch

git checkout -b old-branch <commit-hash>





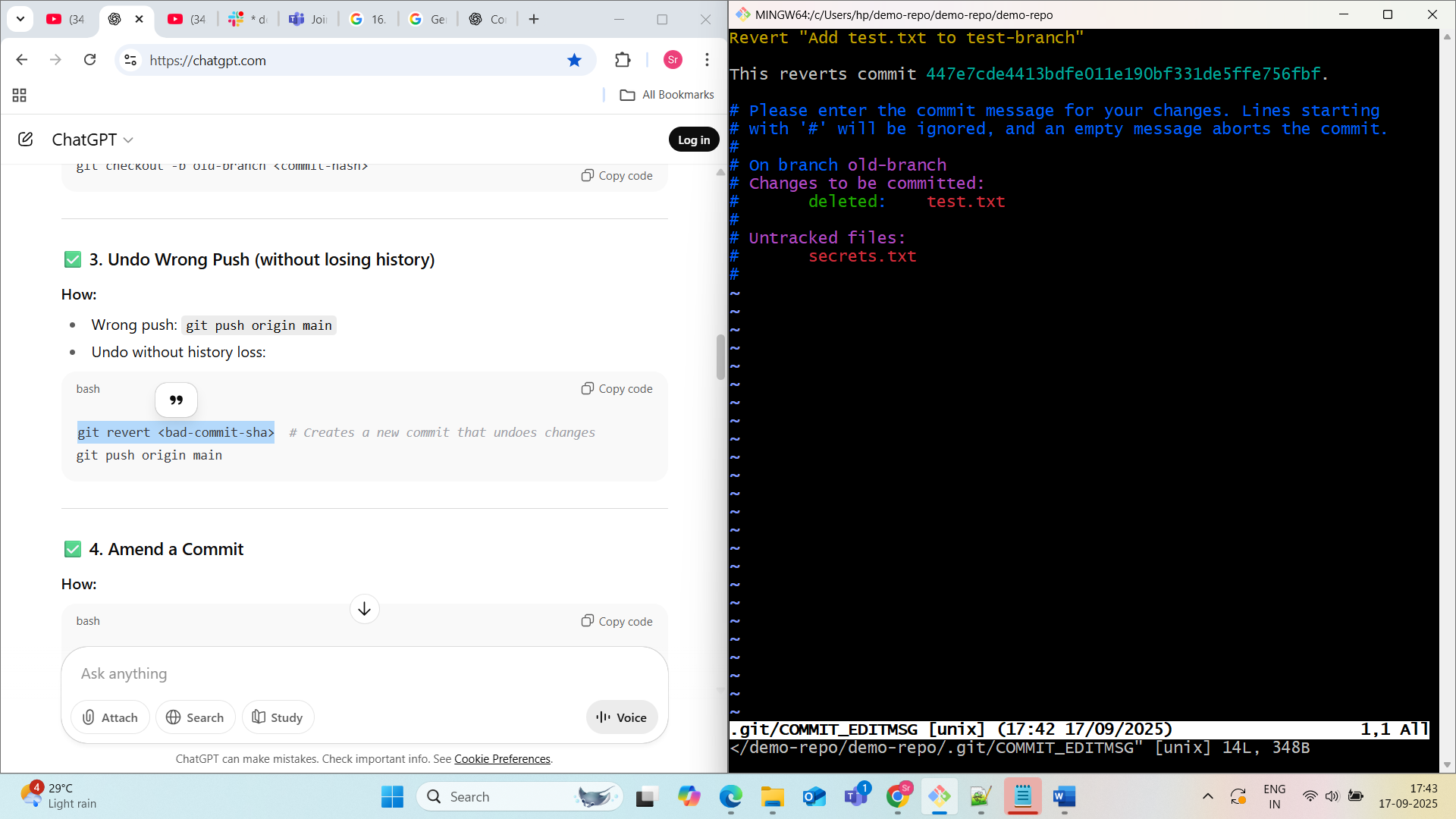
**3. Undo Wrong Push (without losing history)**

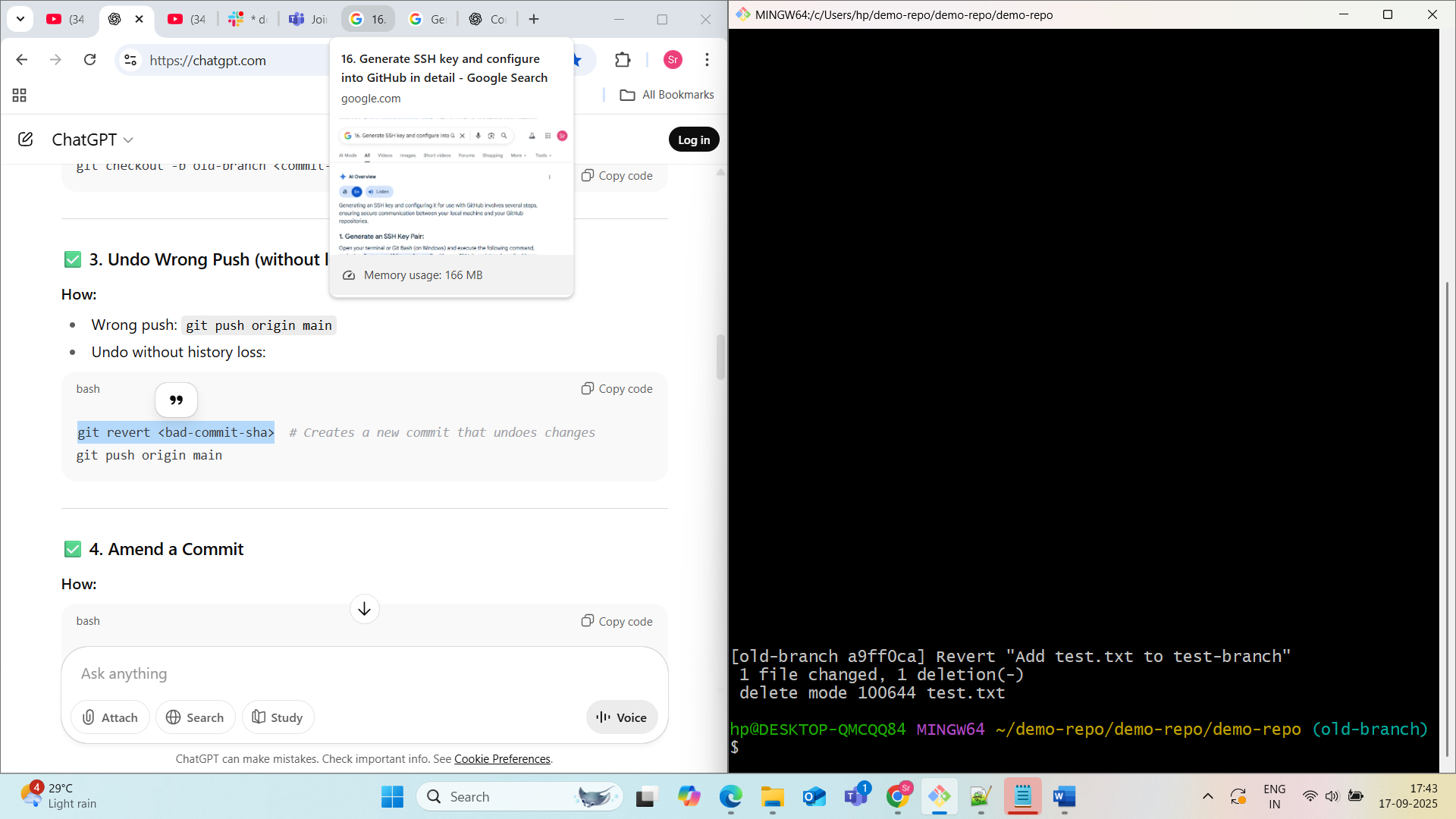
**How:**

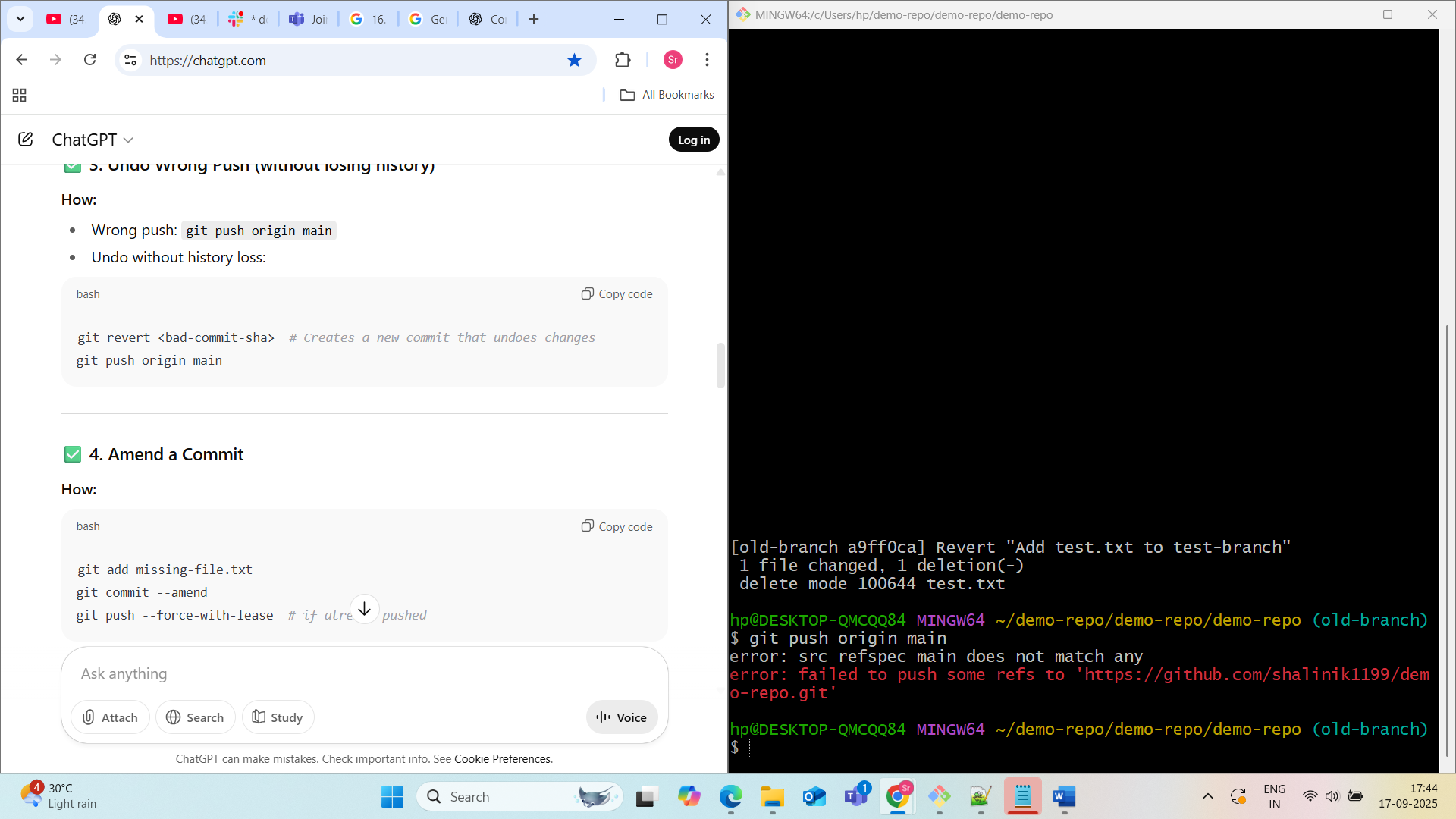
* Wrong push: git push origin main
* Undo without history loss:

git revert <bad-commit-sha> # Creates a new commit that undoes changes

git push origin main







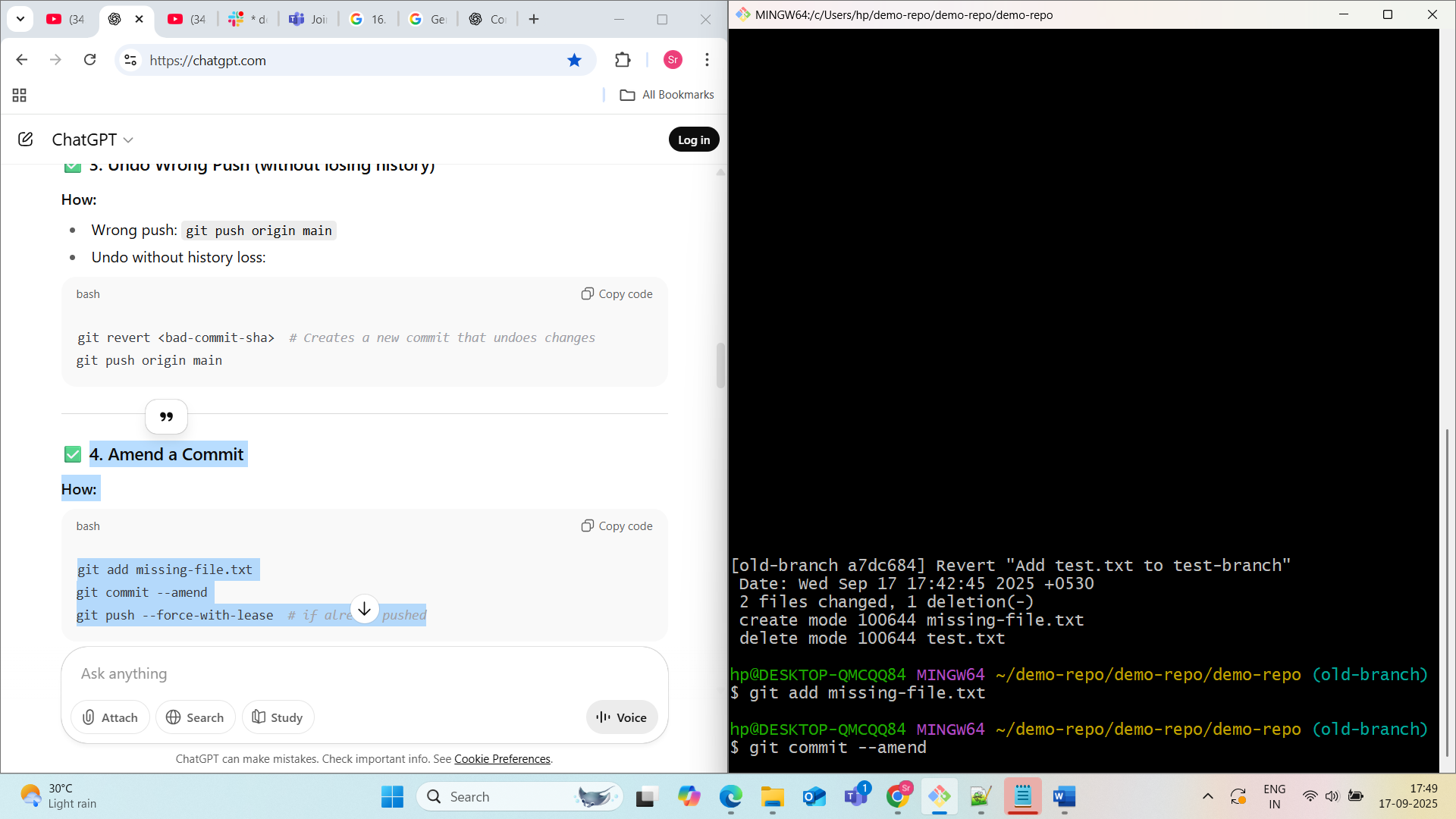
**4. Amend a Commit**

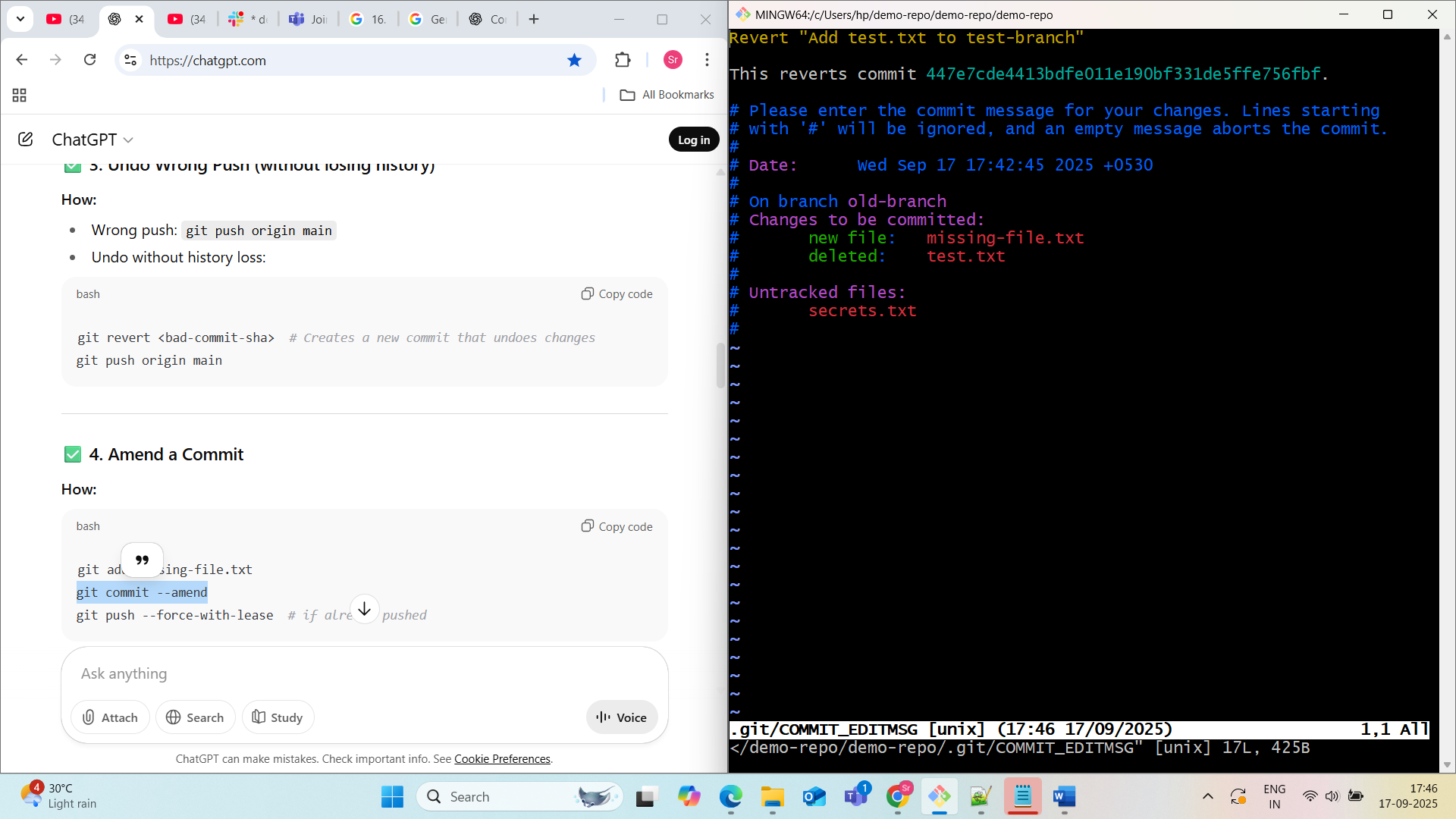
**How:**

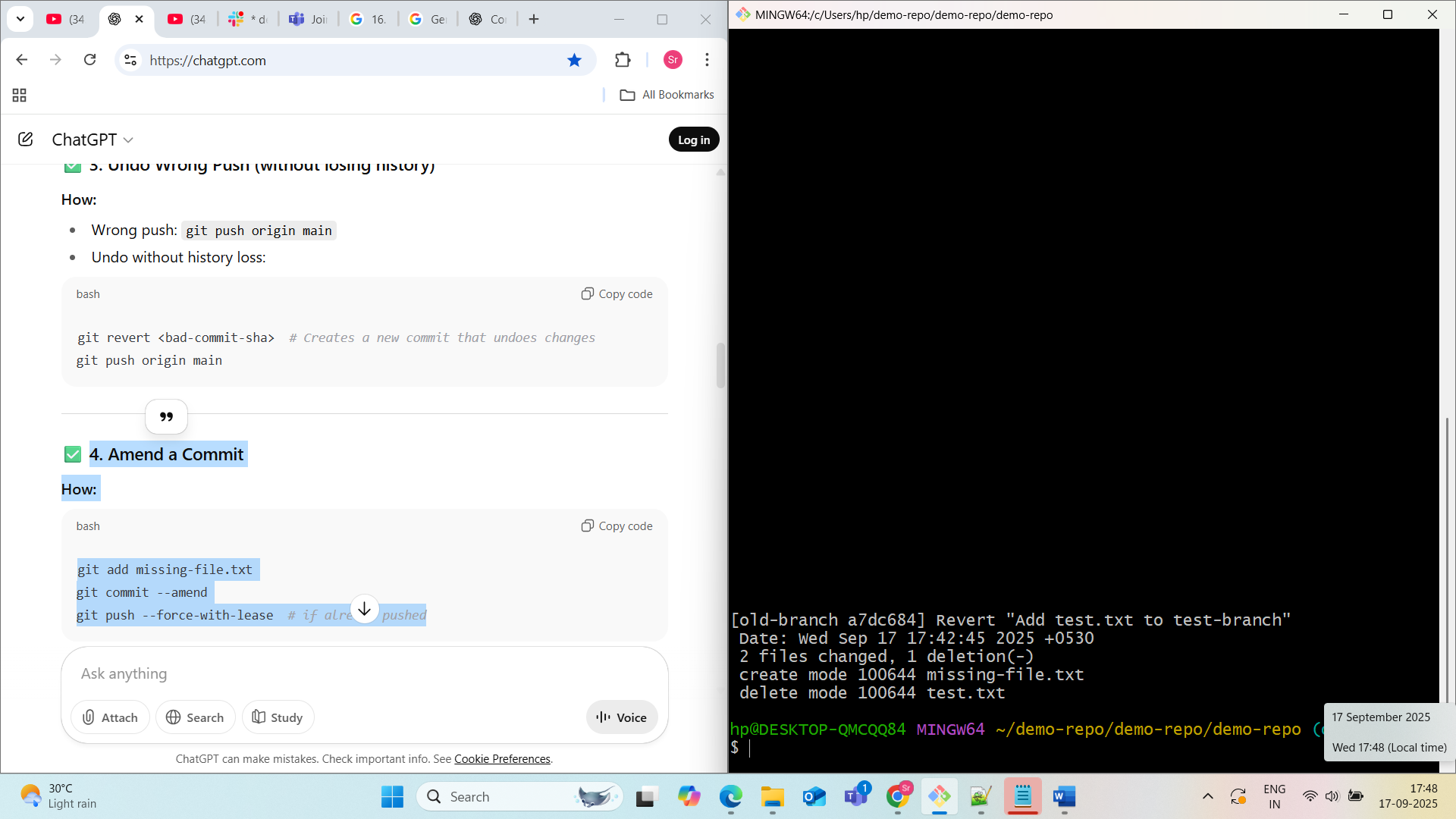
git add missing-file.txt

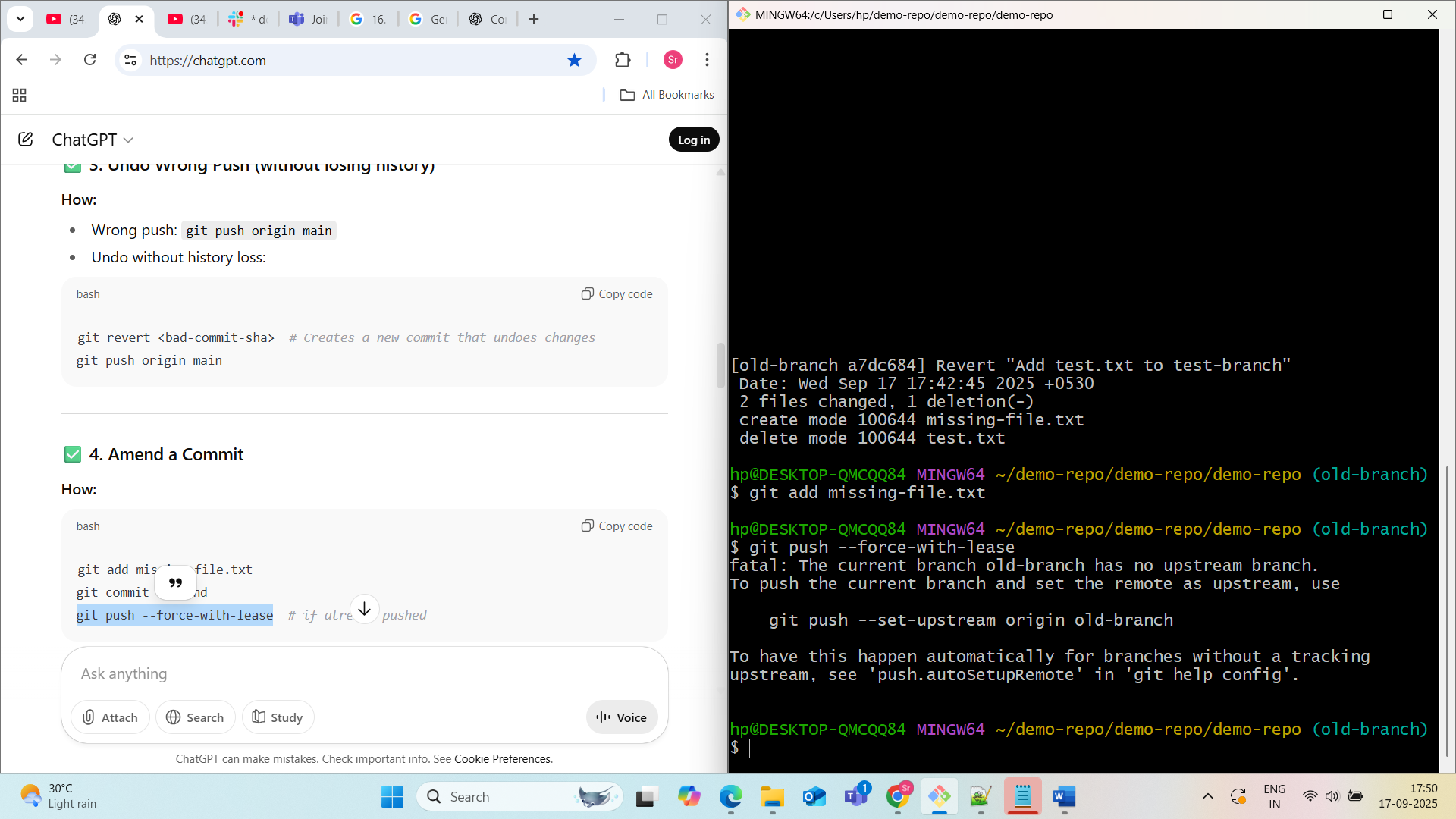
git commit --amend

git push --force-with-lease # if already pushed









**5.Cherry-pick a Commit**

Take a specific commit from one branch and apply it to another branch.

**Cherry-picking** in Git allows you to **select a specific commit from one branch and apply it to another**, without merging the entire branch.

**Why Use git cherry-pick?**

Imagine this situation:

* You’re working on feature-branch, and you made a commit: Fix critical bug.
* But that fix is also needed in main — without bringing in the rest of feature-branch.

**Solution:** Use git cherry-pick to copy that one specific commit to main.

**Syntax:** git cherry-pick <commit-hash>