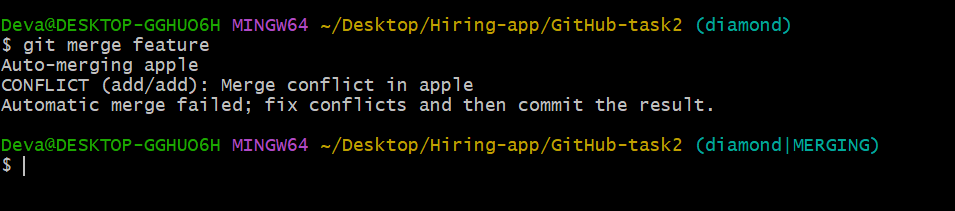
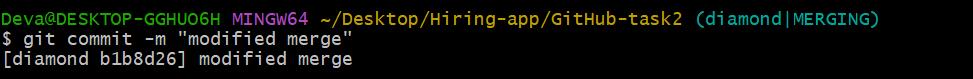
**Git & GitHub Challenges**

1. Resolve Merge Conflicts

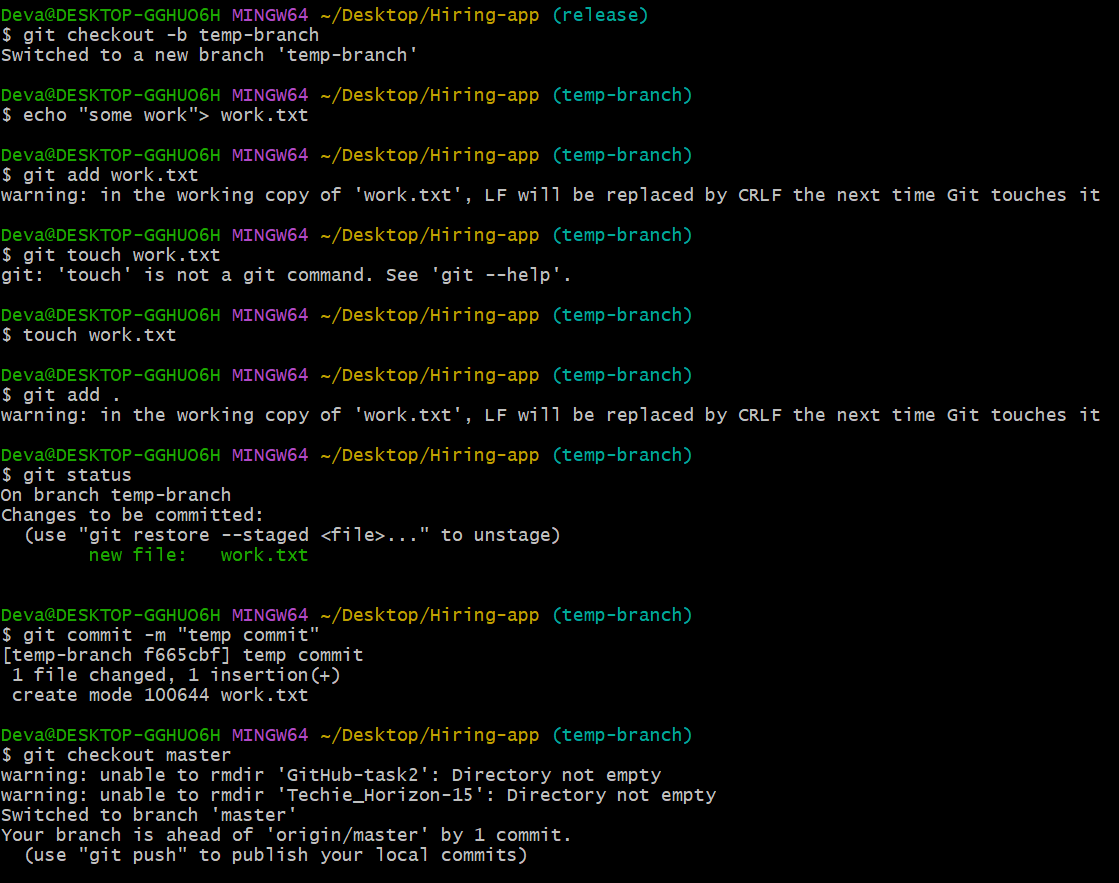


* Create a merge conflict intentionally (two users editing the same line).
* Resolve the conflict and push the changes.

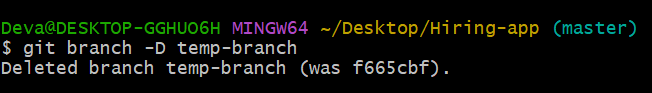


2. Recover Deleted Branch

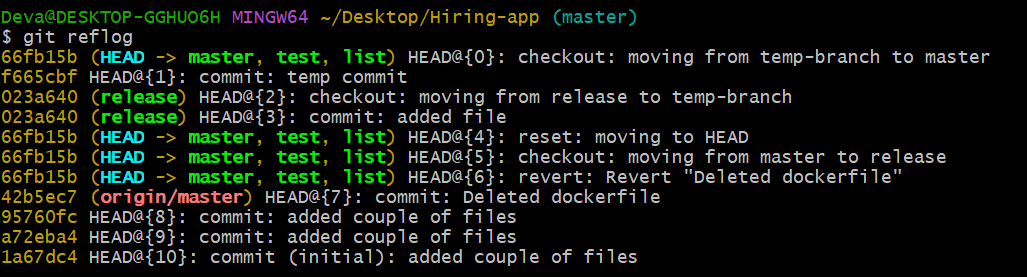
* Delete a local branch and then recover it using the reflog
* Even if you delete a branch ,git keeps its history in reflog.

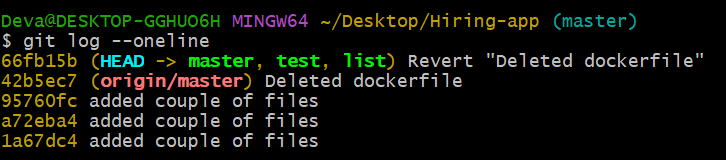


* git checkout master
* git branch –D temp-branch

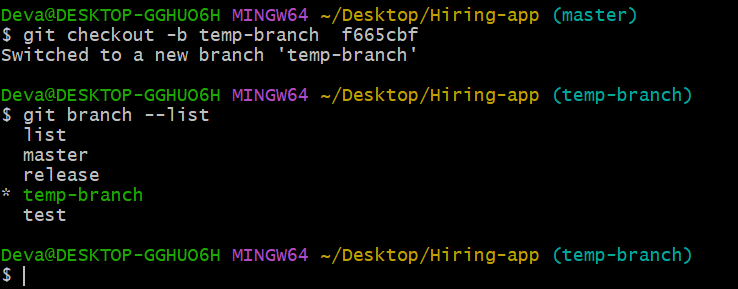


* git reflog – it shows you the history of where your HEAD and branches have pointed, including checkouts, commits, rebases, merges, resets,etc..





* git checkout –b branch <commit\_hash>

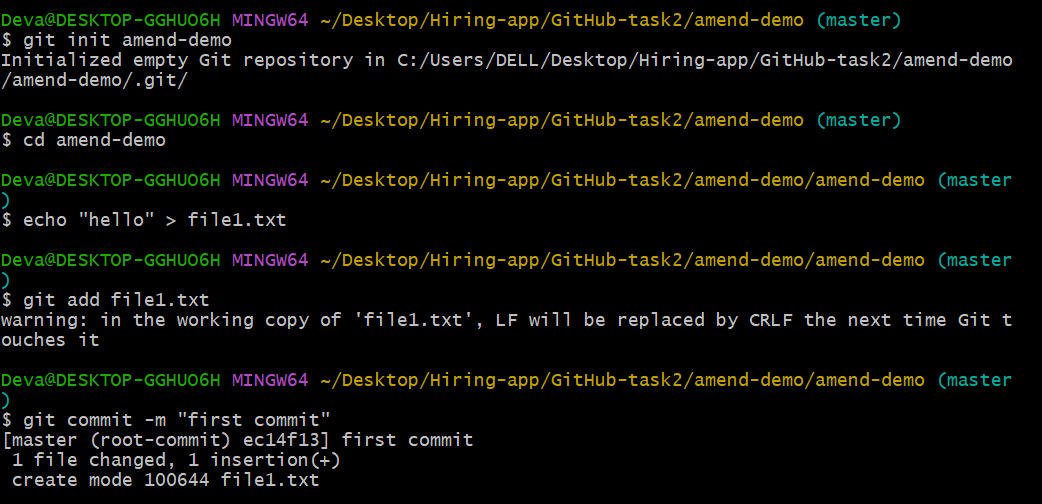


3. Undo Wrong Push

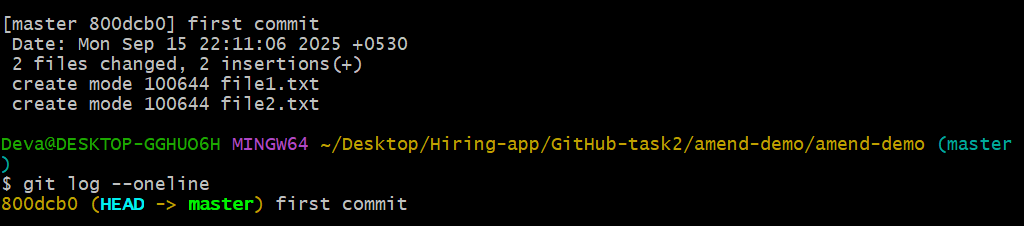
* Push a wrong commit to GitHub, then undo it without losing history.
* Git commit –m “wrong commit”
* Git push origin master
* Git revert <commit id>
* Git push origin master

4. Amend a Commit

* Make a commit, then add a missing file to it using git commit –amend
* git init amend-demo
* cd amend-demo
* echo "Hello" > file1.txt
* git add file1.txt
* git commit -m "First commit"
* echo "This is missing file" > file2.txt
* git add file2.txt
* git commit --amend



* git add file2.txt
* git commit --amend

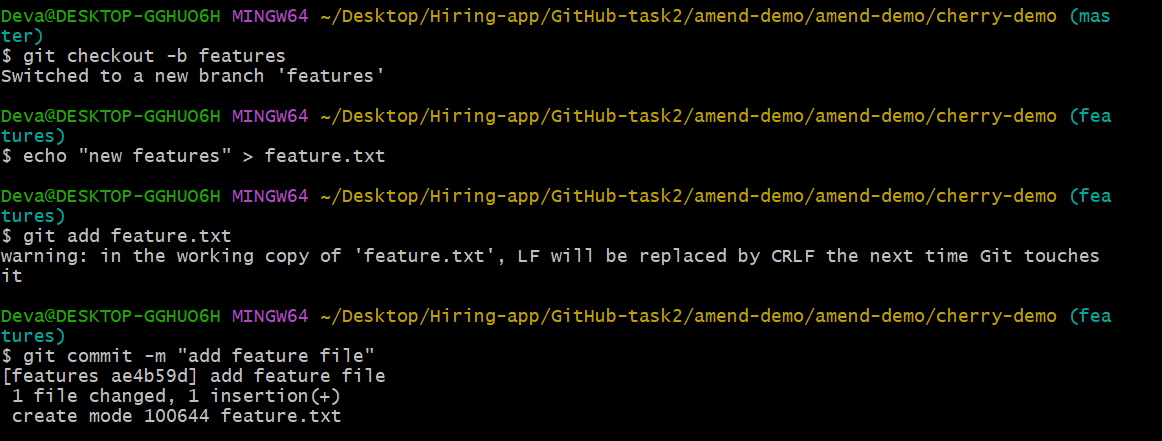


5. Cherry-pick a Commit

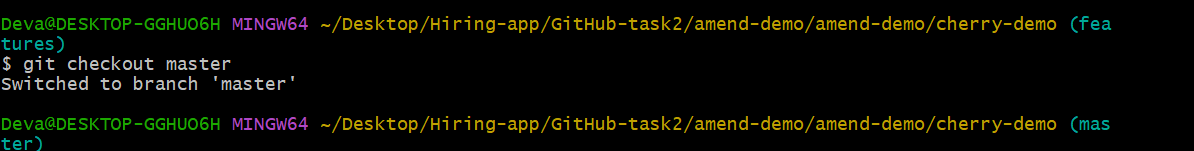
* Take a specific commit from one branch and apply it to another branch
* git init cherry-demo
* cd cherry-demo
* echo "base file" > base.txt
* git add base.txt
* git commit -m "Base commit"



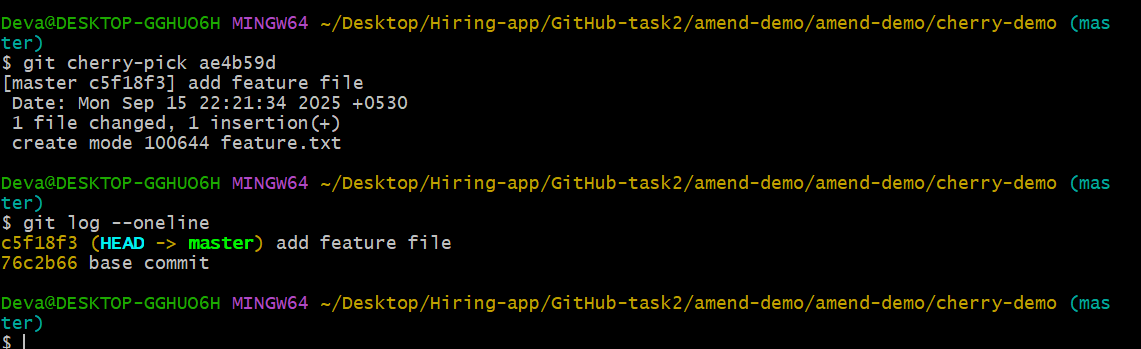
* git checkout -b feature
* echo "new feature" > feature.txt
* git add feature.txt
* git commit -m "Add feature file"



* git checkout main

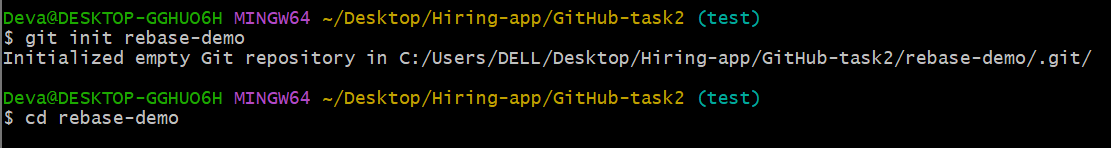


* git cherry-pick
* git log –oneline

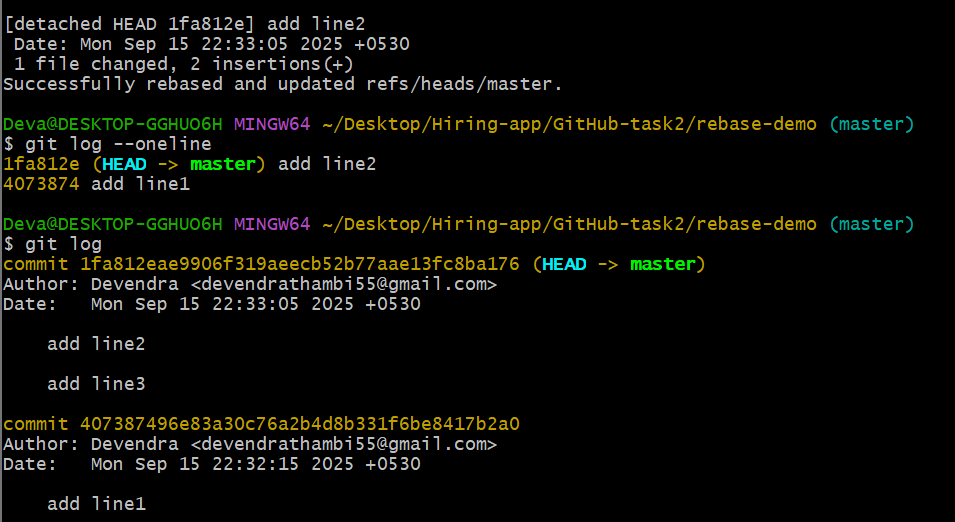


6. Interactive Rebase

* Reorder and squash multiple commits into a single clean commit
* git init rebase-demo
* cd rebase-demo



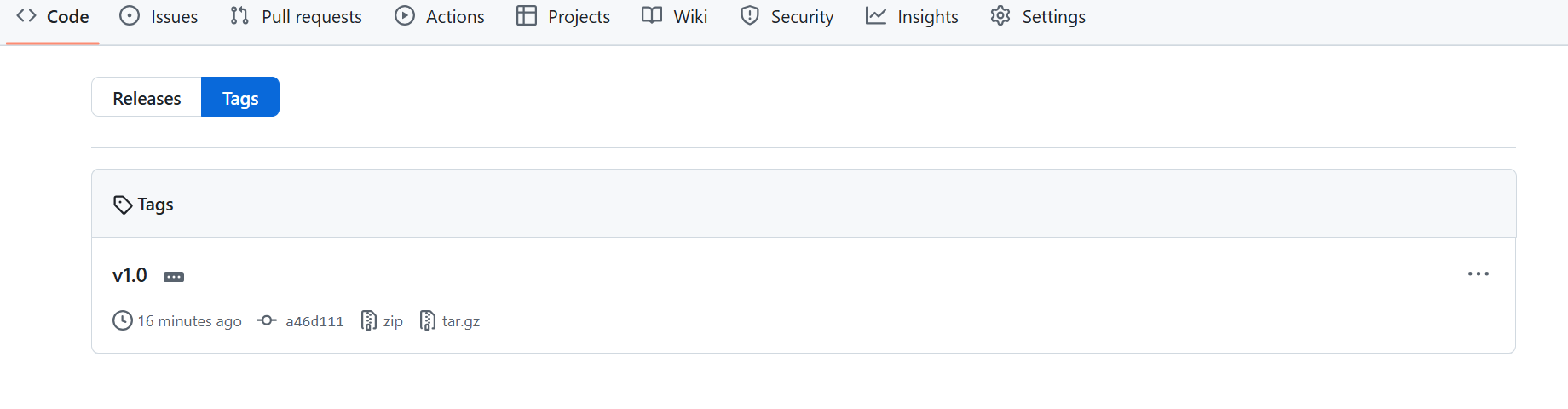
* git rebase -i HEAD~2
* git log



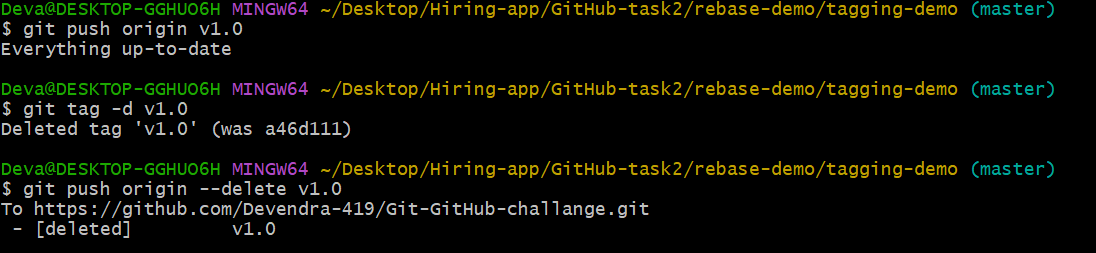
7. Tagging & Release

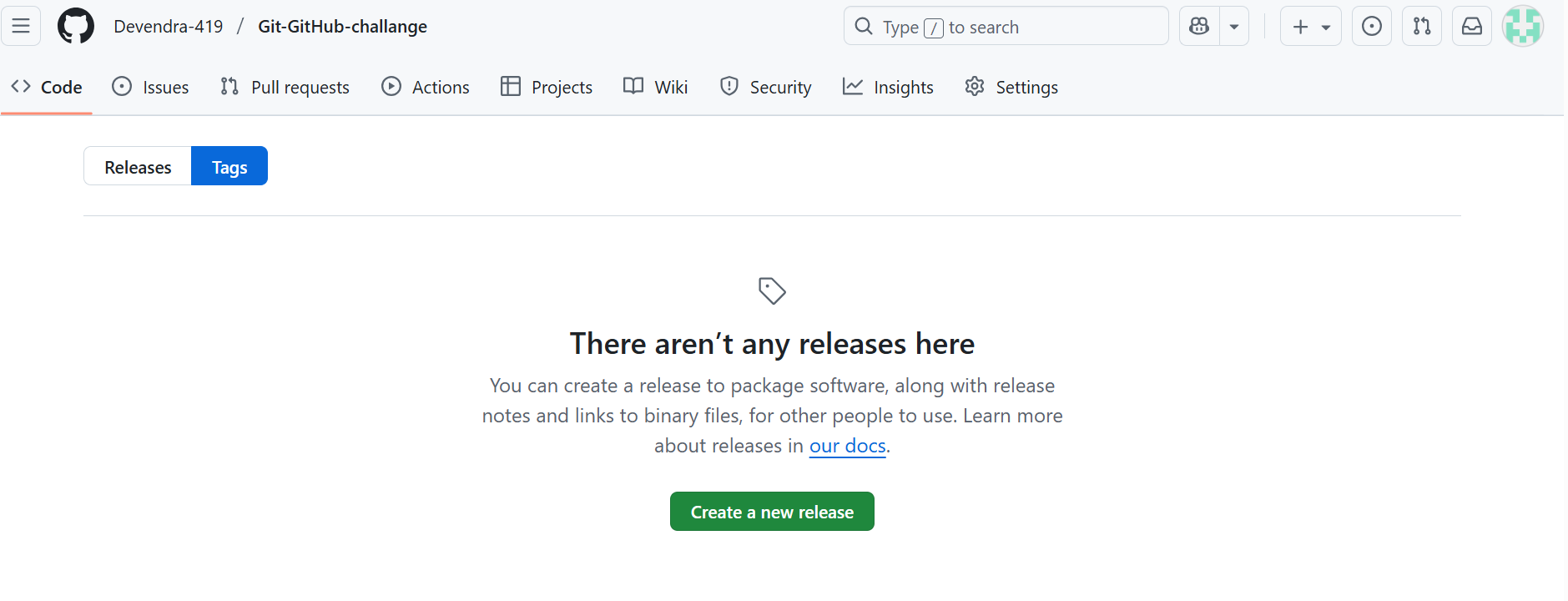
* Create a version tag (v1.0), push it to GitHub, then delete and restore it
* git tag -a v1.0 -m "Initial release of version 1.0"
* git tag v1.0
* git push origin v1.0





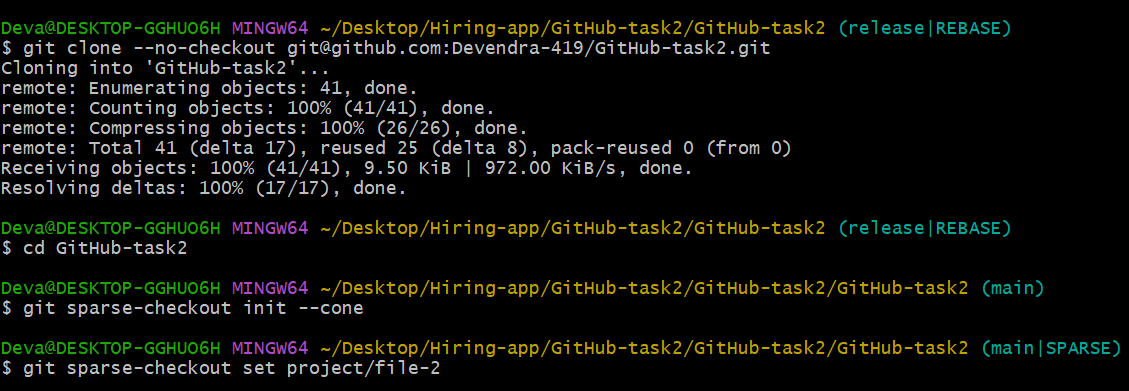
* git tag -d v1.0
* git push origin --delete v1.0



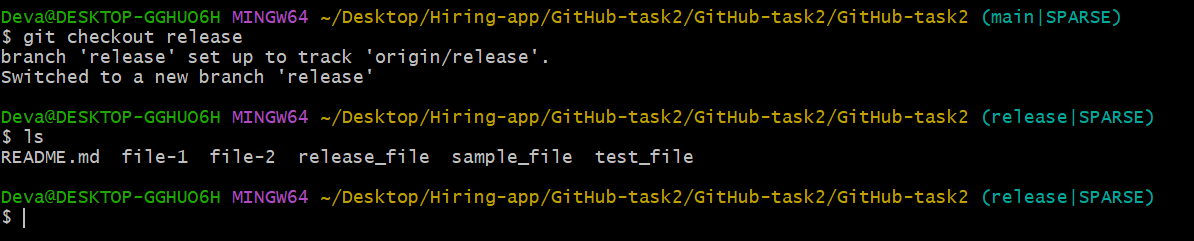


8. Clone with Sparse Checkout

* Clone only a subdirectory of a repo using sparse checkout
* Use git clone --no-checkout and repolink
* Use git sparse-checkout init –cone



* If you want the folder project/doc use
* Git checkout release

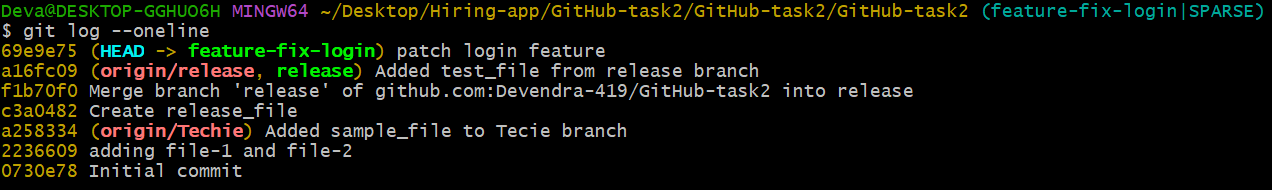


9. Reset vs Revert Challenge

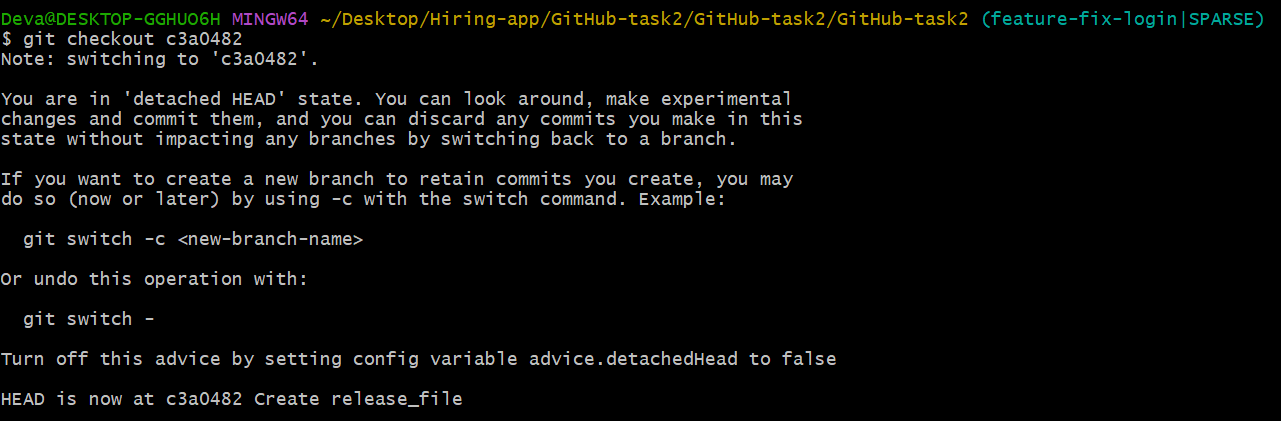
* Demonstrate the difference between git reset --hard and git revert in a repo

10. Detached HEAD Challenge

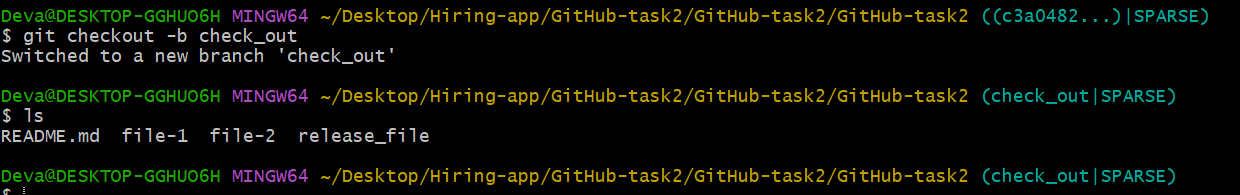
* Checkout a specific commit (detached HEAD state) and create a new branch from it
* Git log --oneline



* Git checkout commit id

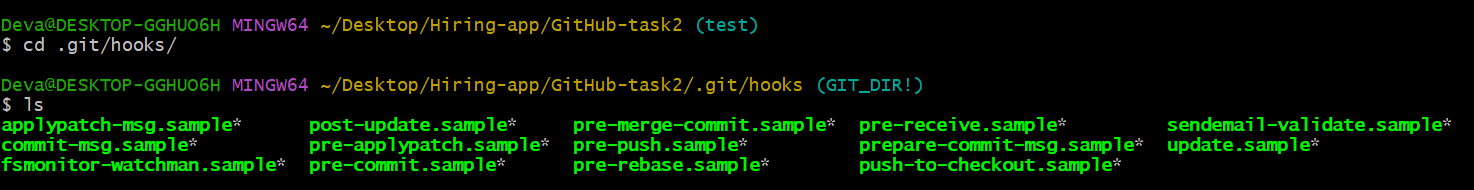


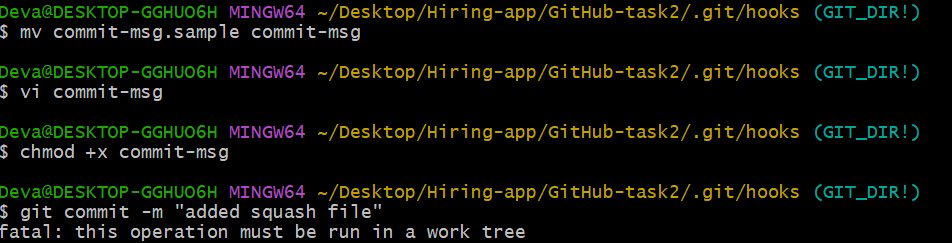
* Git checkout –b check-out



11. Git Hooks Challenge

* Configure a pre-commit hook to reject commits without a message format (e.g., must start with JIRA-XXX).



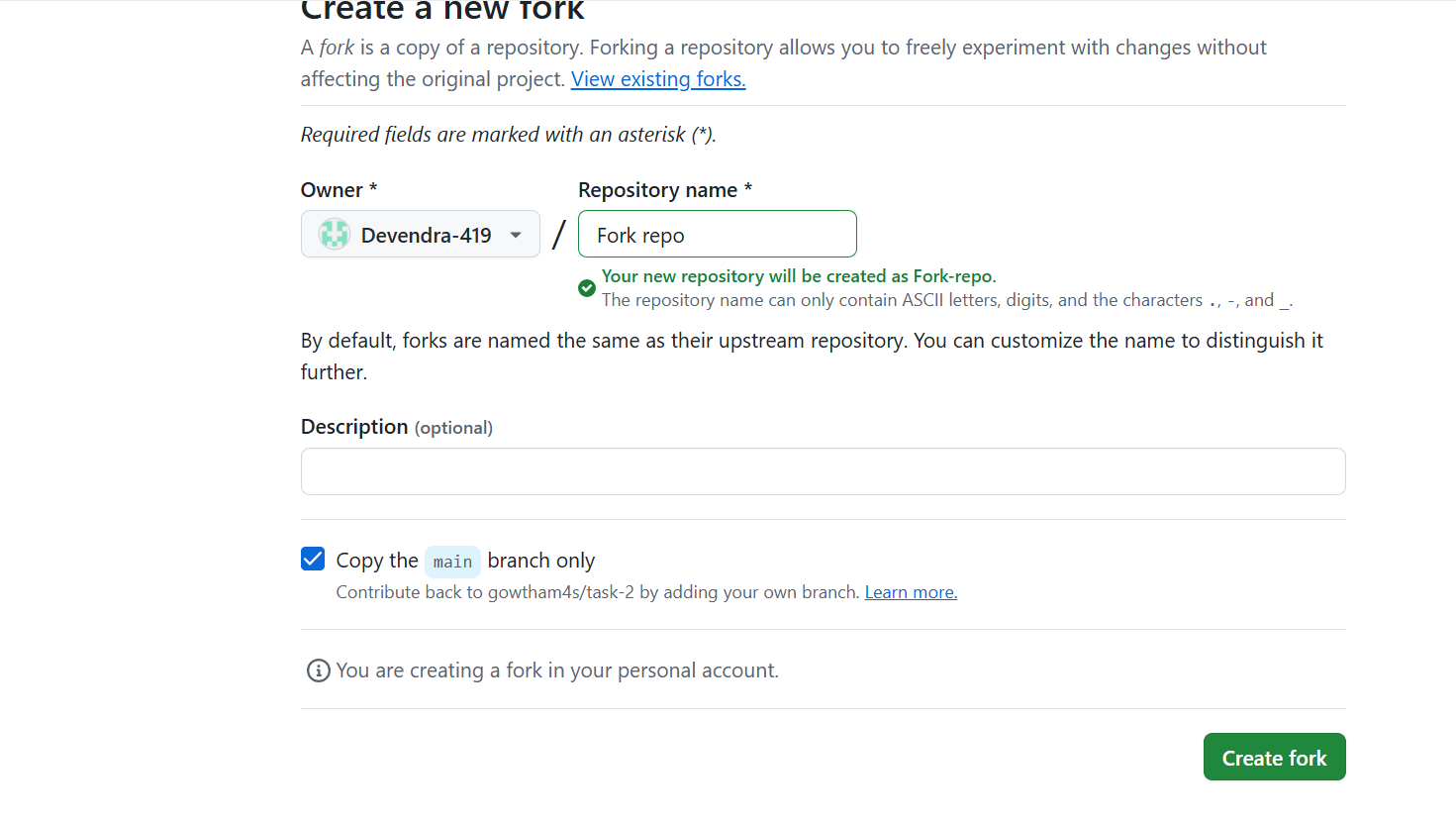


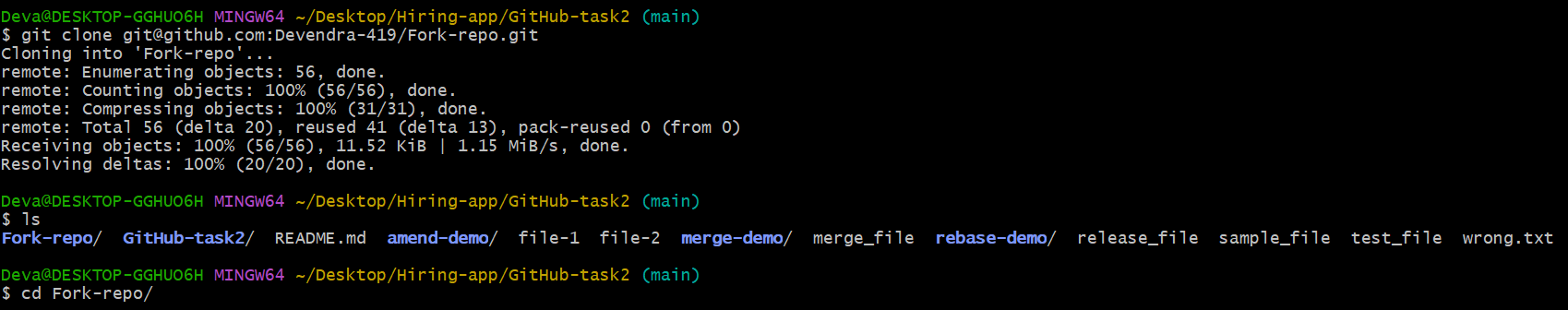
12. Squash Merge vs Rebase Merge

* Show the difference between squash merge and rebase merge with evidence.

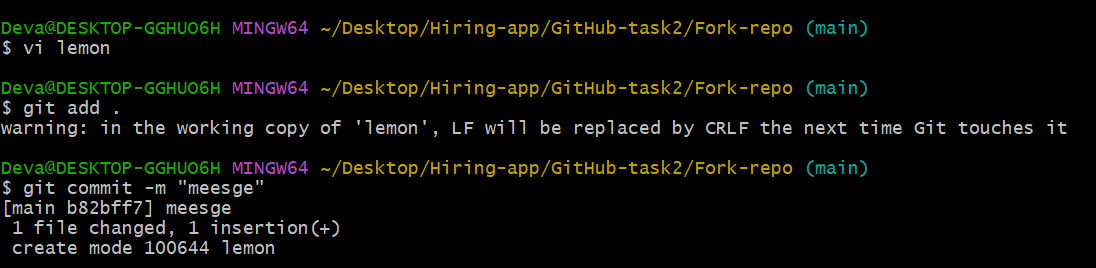
13. Fork & Pull Request Workflow

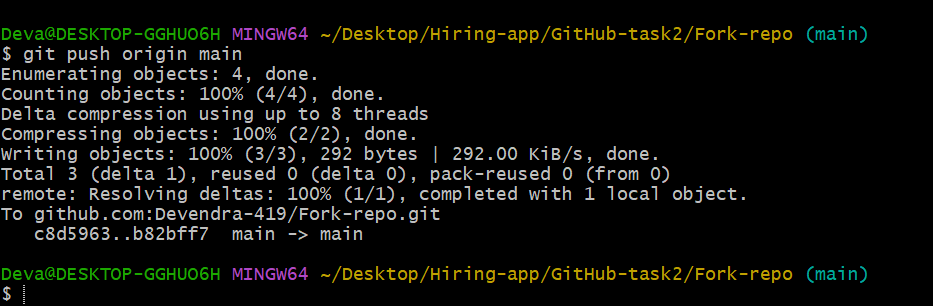
* Fork a repo, make a change, and submit a pull request to the original repo

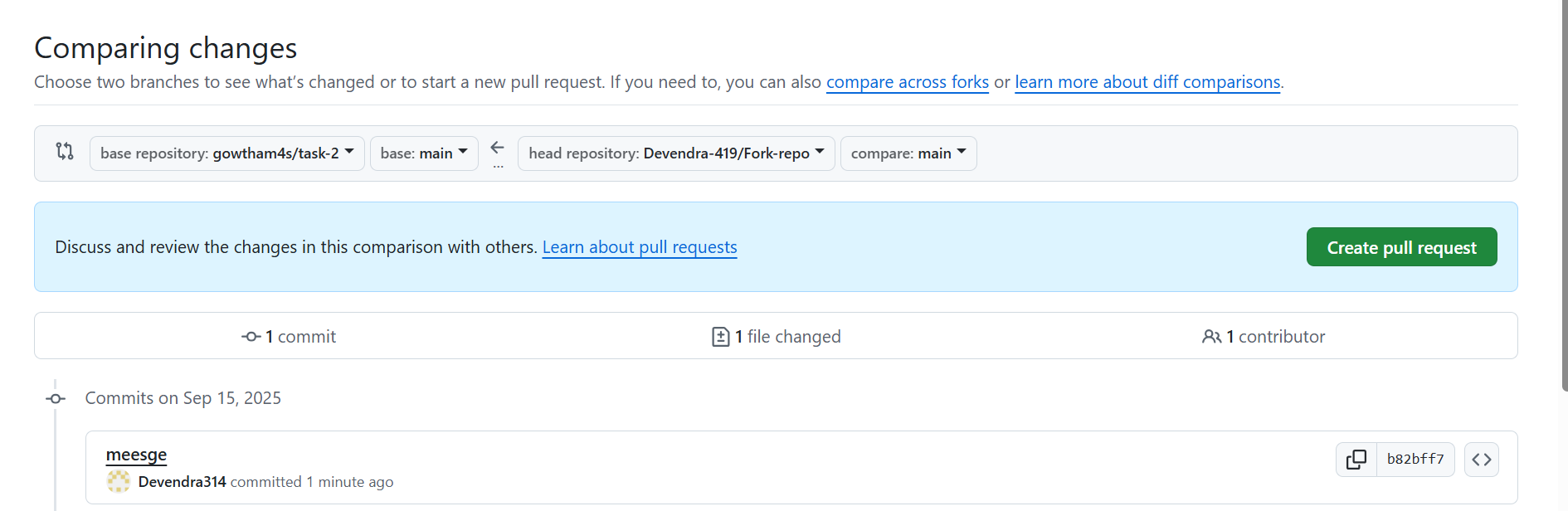




* Edit any file and push it

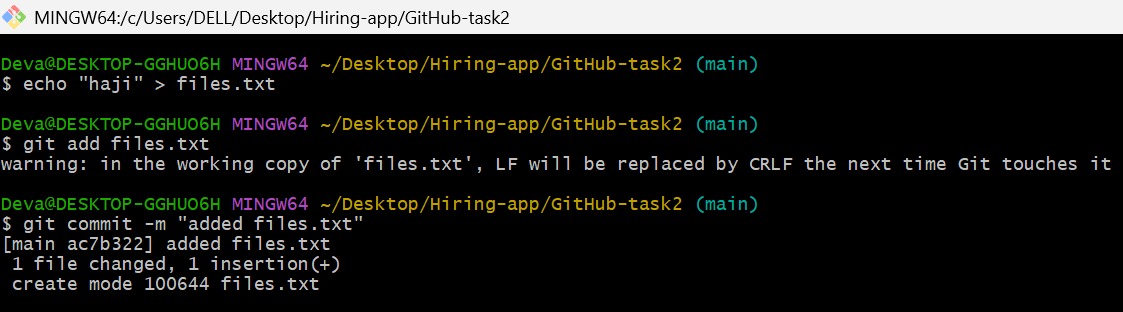




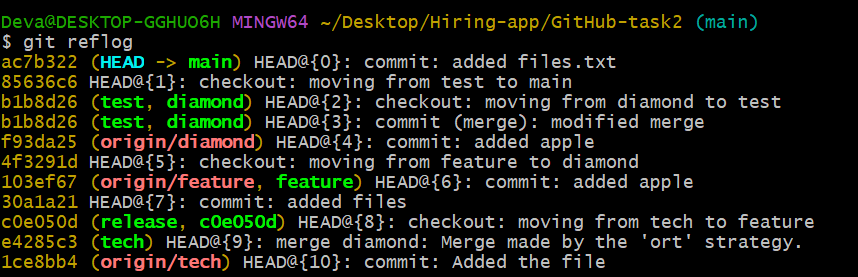


14. Recover Lost Commit

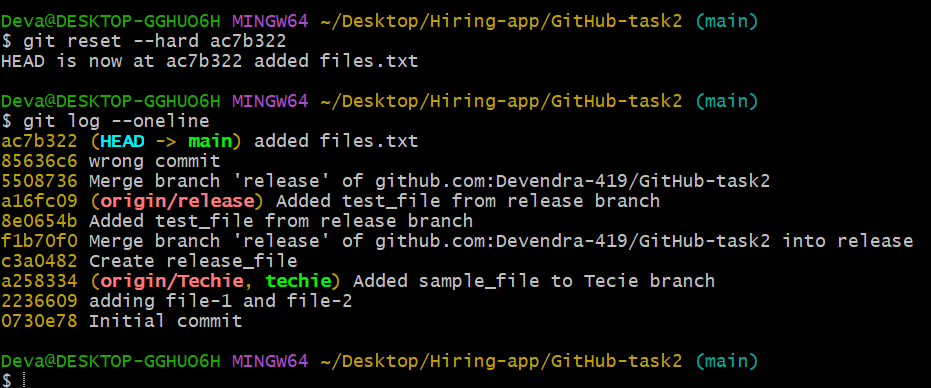
* Commit something, reset hard, and then recover it using git reflog
* **echo "hello" > file.txt**
* **git add file.txt**
* **git commit -m "Added file.txt"**

****

* **git reflog**

****

* **git reset --hard def456**

****