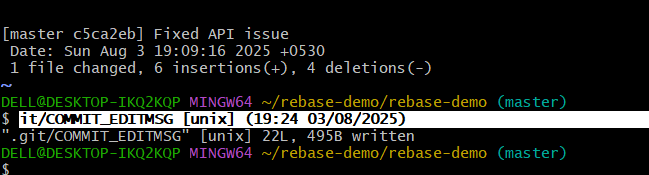
# DAY -4

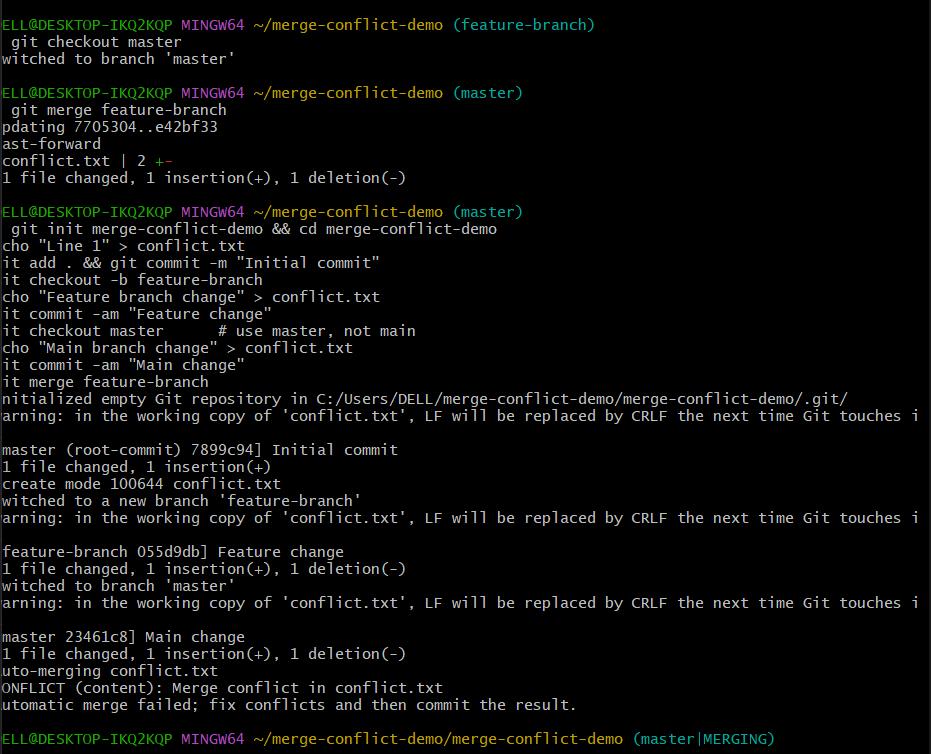
## Challenge 1: Interactive Rebase to Modify Commit History

* Step 1: View recent commits:  
   git log --oneline -n 5
* Step 2: Start an interactive rebase:  
   git rebase -i HEAD~3
* Step 3: Modify commits:  
   - Rename: change 'pick' to 'reword'  
   - Squash: change second/third 'pick' to 'squash'  
   - Reorder: rearrange lines
* Step 4: Save and finish rebase:  
   git rebase --continue
* Step 5: Push changes:  
   git push origin feature-branch –force



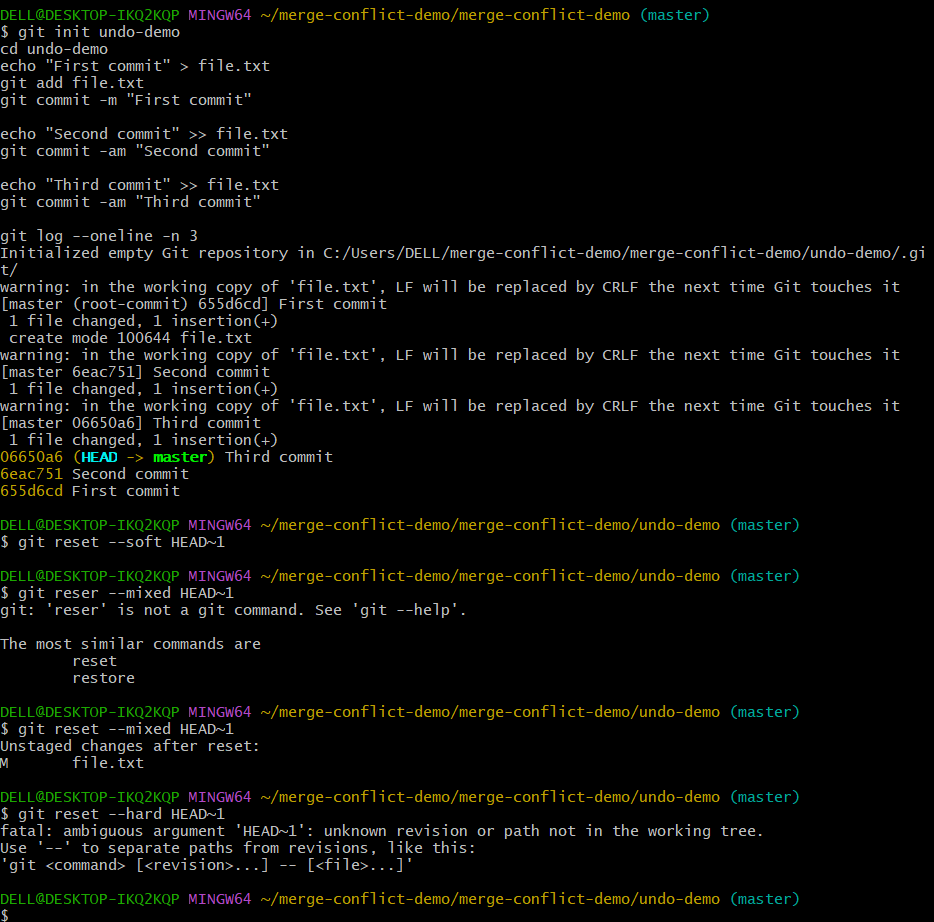
## Challenge 2: Use git cherry-pick

* Step 1: Checkout and view commits:  
   git checkout feature-branch  
   git log --oneline --graph -n 5
* Step 2: Switch to main:  
   git checkout main
* Step 3: Apply commit:  
   git cherry-pick <commit-hash>
* Step 4: If conflict:  
   git add <resolved-file>  
   git cherry-pick --continue
* Step 5: Push:  
   git push origin main



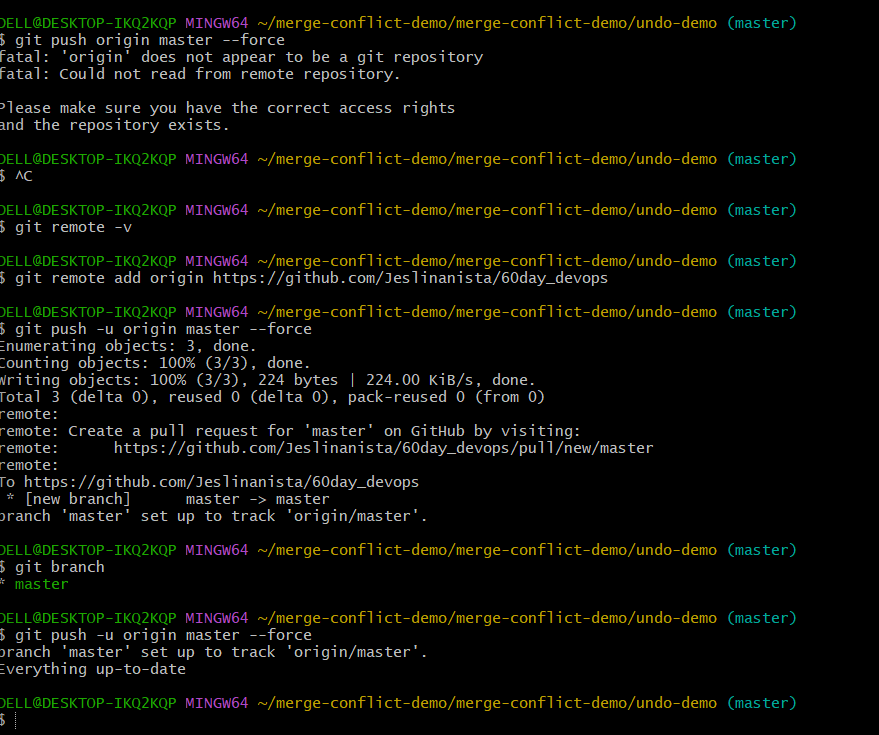
## Challenge 3: Merge Conflict Scenario

* Step 1: Init and commit:  
   git init merge-conflict-demo  
   cd merge-conflict-demo  
   echo "Line 1" > conflict.txt  
   git add conflict.txt  
   git commit -m "Initial commit"
* Step 2: Feature branch:  
   git checkout -b feature-branch  
   echo "Feature branch change" > conflict.txt  
   git commit -am "Modified in feature-branch"
* Step 3: Main branch:  
   git checkout main  
   echo "Main branch change" > conflict.txt  
   git commit -am "Modified in main"
* Step 4: Merge and resolve:  
   git merge feature-branch  
   Resolve conflict and run:  
   git add conflict.txt  
   git commit -m "Resolved conflict"
* Step 5: Rebase method:  
   git checkout main  
   git reset --hard HEAD~1  
   git rebase feature-branch  
   Resolve conflict and run:  
   git add conflict.txt  
   git rebase --continue
* Step 6: Push:  
   git push origin main –force



## Challenge 4: Undo Commit with reset and revert

* Step 1: Create sample repo and commits
* Option 1: reset  
   - Soft: git reset --soft HEAD~1  
   - Mixed: git reset --mixed HEAD~1  
   - Hard: git reset --hard HEAD~1
* Option 2: revert  
   git revert HEAD
* Push changes accordingly:  
   - Reset: git push origin main --force  
   - Revert: git push origin main



## Challenge 5: Amend Last Commit

* Step 1: Check last commit:  
   git log --oneline -n 1
* Step 2: Amend message:  
   git commit --amend -m "Updated message"
* Step 3: Add forgotten file:  
   echo "New content" > forgotten.txt  
   git add forgotten.txt  
   git commit --amend --no-edit
* Step 4: Push:  
   git push origin main –force

## Challenge 6: Git Hooks

* Step 1: Go to hooks:  
   cd .git/hooks
* Step 2: Create pre-commit:  
   nano pre-commit
* Add hook code and make executable:  
   chmod +x pre-commit
* Test it with trailing whitespace
* Bonus: Move hooks to .githooks and use:  
   git config core.hooksPath .githooks

## Challenge 7: Rebase Feature Branch

* Step 1: Checkout feature branch:  
   git checkout feature-branch
* Step 2: Update main:  
   git fetch origin  
   git checkout main  
   git pull origin main
* Step 3: Rebase:  
   git checkout feature-branch  
   git rebase main
* Step 4: Resolve and continue:  
   git add <resolved-file>  
   git rebase --continue
* Step 5: Push:  
   git push origin feature-branch --force

## Challenge 8: Squash Commits

* Step 1: Create branch:  
   git checkout -b feature-branch
* Step 2: Make commits
* Step 3: Rebase interactively:  
   git rebase -i HEAD~3
* Step 4: Squash and edit message
* Step 5: Push:  
   git push origin feature-branch --force
* Step 6: Verify:  
   git log --oneline -n 3

## Bonus: .git Folder Overview

* - .git/hooks/ → Git hooks (pre-commit, post-commit)
* - .git/objects/ → All blobs, trees, commits
* - .git/refs/ → Branches, tags, remotes
* - .git/HEAD → Pointer to current branch
* - .git/index → Staging area
* - .git/logs/ → Reflog history
* - .git/config → Local repo config