# Git Hands-On Lab — Merge & Conflict Resolution (Execution Report)

## Objective

Demonstrate branching, making branch-specific changes, creating conflicting changes on master, and preparing for a merge + conflict resolution.

## Prerequisites

• Git installed and configured.

• GitDemo repository initialized and connected to remote.

• (Optional) P4Merge installed for visual diffs/merges.

## Execution Steps

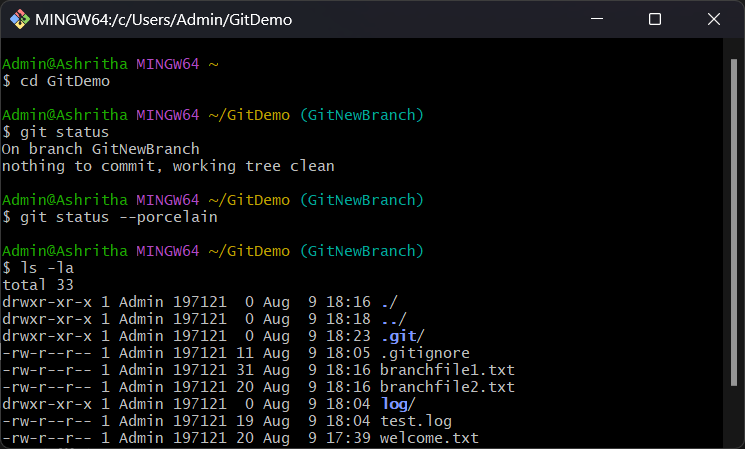
### Step 1 — Navigate to repository & check state

Command:

cd GitDemo  
git status

Description: Ensured we are inside the GitDemo repository and the working tree was clean on the starting branch.

Screenshot:



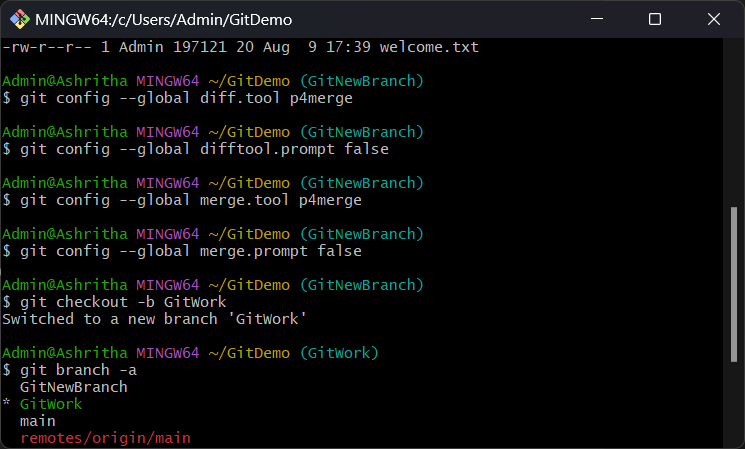
### Step 2 — Configure diff/merge tool and create branch

Commands:

git config --global diff.tool p4merge  
git config --global difftool.prompt false  
git config --global merge.tool p4merge  
git config --global merge.prompt false  
git checkout -b GitWork  
git branch -a

Description: Set up P4Merge (optional) and created a new branch GitWork.

Screenshot:



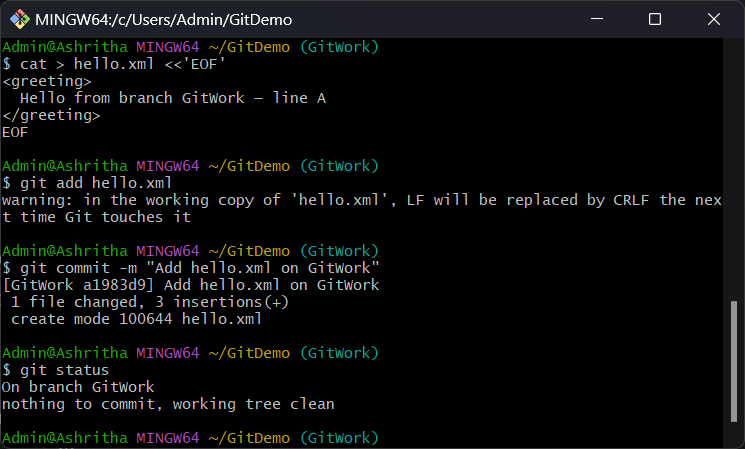
### Step 3 — Create hello.xml on GitWork and commit

Commands:

cat > hello.xml <<'EOF'  
<greeting>  
 Hello from branch GitWork – line A  
</greeting>  
EOF  
git add hello.xml  
git commit -m "Add hello.xml on GitWork"

Description: Created a branch-specific file and committed it on the GitWork branch.

Screenshot:



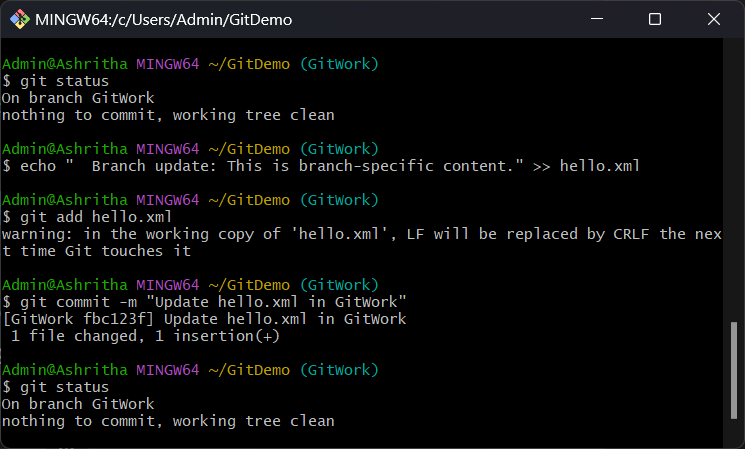
### Step 4 — Update hello.xml on GitWork and commit

Commands:

echo "Branch update: This is branch-specific content." >> hello.xml  
git add hello.xml  
git commit -m "Update hello.xml in GitWork"

Description: Added an extra line to hello.xml and committed the change on GitWork.

Screenshot:



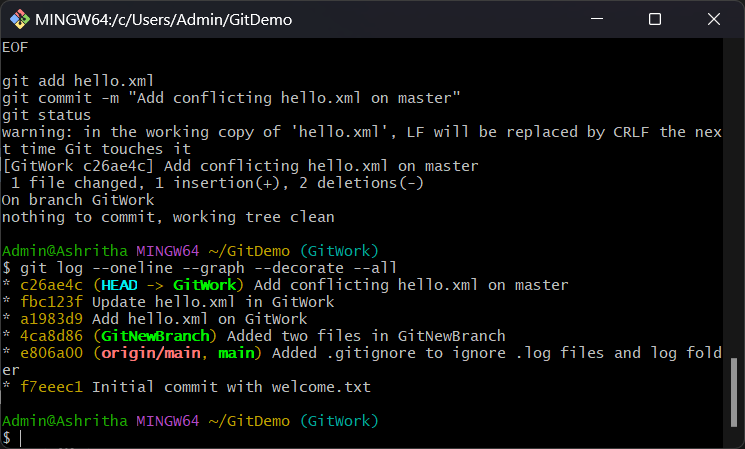
### Step 5 — Add conflicting hello.xml on master and commit

Commands (example sequence):

git checkout main  
cat > hello.xml <<'EOF'  
<greeting>  
 Hello from master — different content  
</greeting>  
EOF  
git add hello.xml  
git commit -m "Add conflicting hello.xml on master"

Description: Created a different version of hello.xml on main to create a deliberate conflict for the future merge.

Screenshot:



### Step 6 — Inspect history and prepare for merge

Command:

git log --oneline --graph --decorate --all

Description: The commit graph shows diverging commits (separate commits on GitWork and main). This confirms that a merge will produce a conflict in hello.xml.

### Step 7 — Merge and resolve conflicts (instructions)

To perform the merge and resolve conflicts, follow these steps:  
  
1. Switch to main:  
 git checkout main  
  
2. Merge GitWork into main:  
 git merge GitWork  
  
 If conflicts occur, Git will list the files with conflicts (e.g., hello.xml).  
  
3. Launch the configured merge tool to resolve conflicts:  
 git mergetool  
  
 Resolve conflicts in the 3-way view (LOCAL, BASE, REMOTE). Save the resolved file(s) and exit the merge tool.  
  
4. Stage and commit the resolution:  
 git add hello.xml  
 git commit -m "Resolved merge conflict for hello.xml"  
  
5. Push the resolved main branch to the remote:  
 git push origin main  
  
6. Optionally, add temporary backup files to .gitignore, commit and push the .gitignore if needed.

Note: I did not find screenshots of the actual merge/mergetool UI in the images you provided. If you run the merge and capture screenshots of the conflict markers and the merge tool, I will insert them into the document.

### Cleanup & Verification

Commands:

git branch -a  
git branch -d GitWork  
 git log --oneline --graph --decorate

Description: After resolving conflicts and committing, delete the branch if it is no longer needed and inspect the history.

## Screenshots included

1. Screenshot 2025-08-09 182404.png

2. Screenshot 2025-08-09 182551.png

3. Screenshot 2025-08-09 182727.png

4. Screenshot 2025-08-09 182803.png

5. Screenshot 2025-08-09 182856.png

## Conclusion

The repository now contains branch-specific changes and a conflicting file on main. Follow the merge steps above to resolve conflicts. Share screenshots of the merge/mergetool run if you want them embedded.