**4.Git – HOL**

**1.Explain how to resolve the conflict during merge**

1. Try to Merge

Run the merge command:

git merge other-branch-name

If there’s a conflict, Git will stop and show a warning.

2. Check Conflict Files

Git will list the files with conflicts. Open those files — you’ll see something like:

<<<<<<< HEAD

your changes

=======

other branch's changes

>>>>>>> other-branch-name

3. Manually Fix the Conflict

Edit the file and choose what content to keep (yours, theirs, or combine). Then remove the conflict markers (<<<<<<<, =======, >>>>>>>).

4. Mark Conflict as Resolved

After fixing, stage the resolved files:

git add filename

5. Commit the Merge

Now finish the merge by committing:

git commit -m "Resolved merge conflict"

**Please follow the instructions to complete the hands-on. Each instruction expect a command for the Git Bash.**

1. **Verify if master is in clean state.**

git checkout master

git status

1. Create a branch **“GitWork”.** Add a file “hello.xml”.

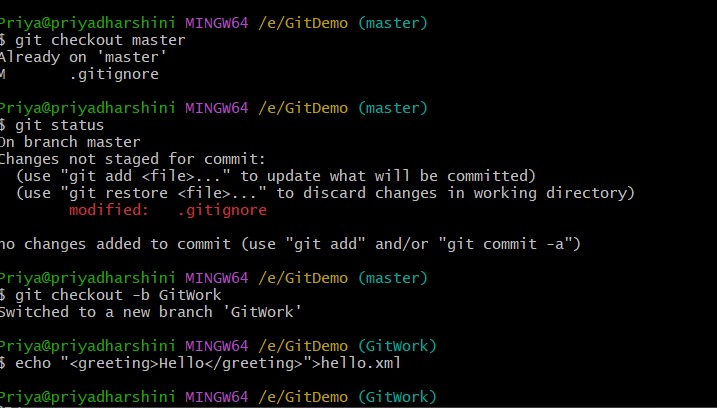
git checkout -b GitWork

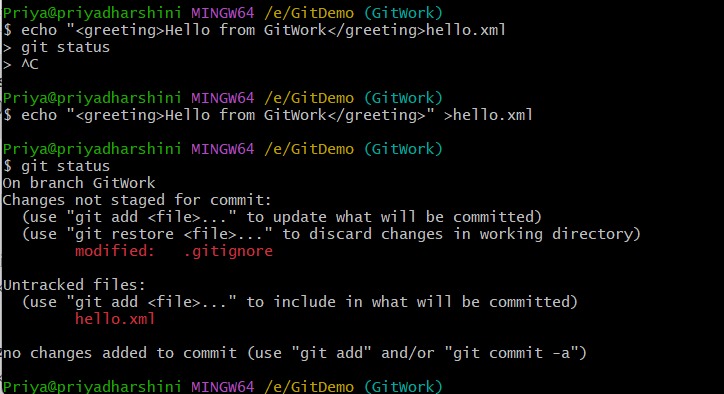
echo "<greeting>Hello</greeting>" > hello.xml

1. Update the content of “hello.xml” and observe the status

echo "<greeting>Hello from GitWork</greeting>" > hello.xml

git status





1. Commit the changes to reflect in the branch

git add hello.xml

git commit -m "GitWork: Added hello.xml with branch-specific content"

1. Switch to master.

git checkout master

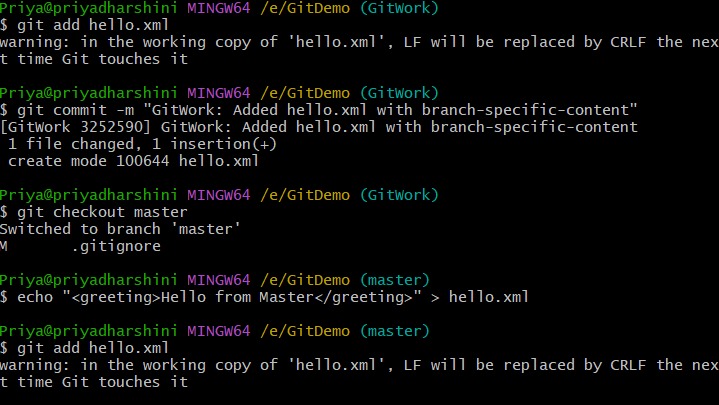
1. Add a file **“hello.xml”** to the master and add some different content than previous.

echo "<greeting>Hello from Master</greeting>" > hello.xml

1. Commit the changes to the master

git add hello.xml

git commit -m "Master: Added hello.xml with master-specific content"

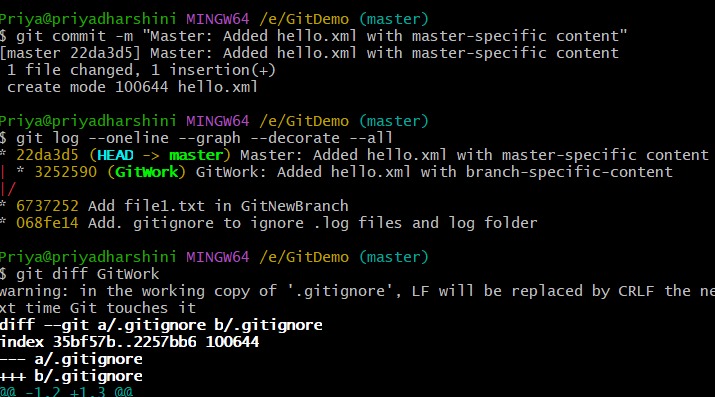


1. Observe the log by executing **“git log –oneline –graph –decorate –all”**

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

1. Check the differences with Git diff tool

git diff GitWork



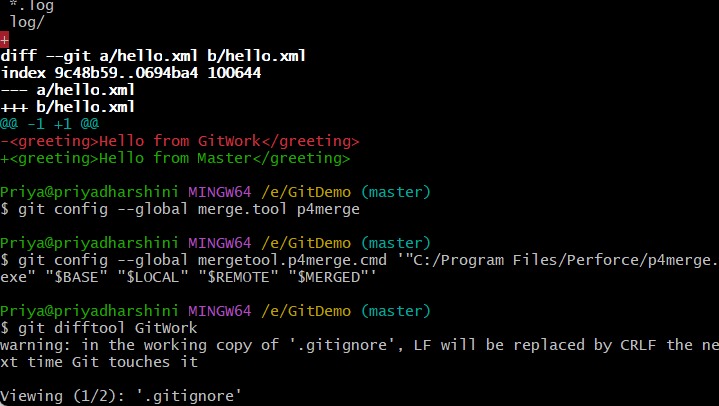
1. For better visualization, use P4Merge tool to list out all the differences between master and branch

git config --global merge.tool p4merge

git config --global mergetool.p4merge.cmd '"C:/Program Files/Perforce/p4merge.exe" "$BASE" "$LOCAL" "$REMOTE" "$MERGED"'

👀 Then view differences:

git difftool GitWork

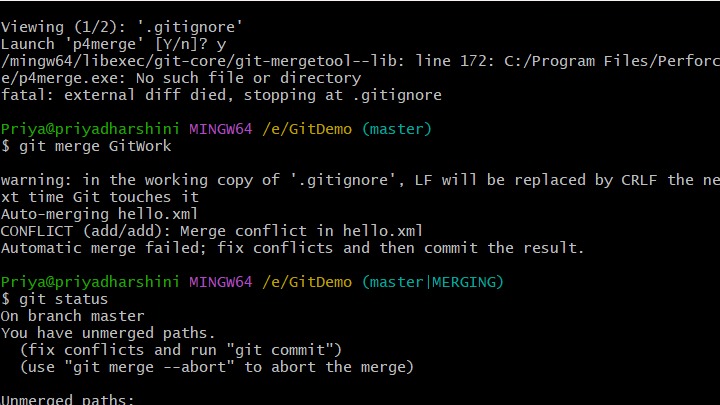


1. Merge the branch to the master

git merge GitWork

1. Observe the git mark up.

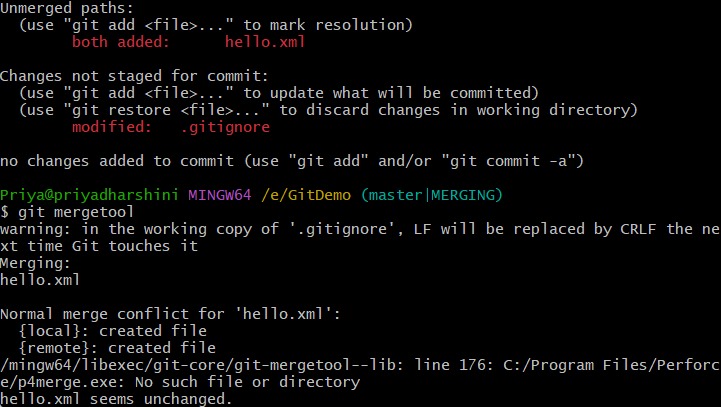
git status



1. Use 3-way merge tool to resolve the conflict

git mergetool

Then resolve the conflict in P4Merge and save the final result.



1. Commit the changes to the master, once done with conflict

git commit -m "Resolved conflict between master and GitWork on hello.xml"

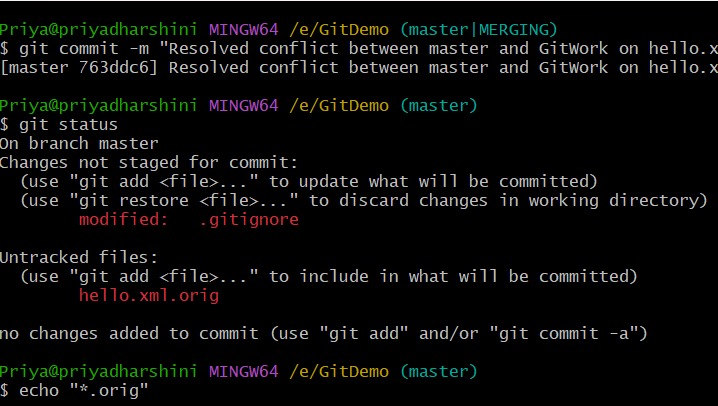
1. Observe the git status and add backup file to the .gitignore file.

Observe:

git status

If you see a file like hello.xml.orig, then:

echo "\*.orig" >> .gitignore



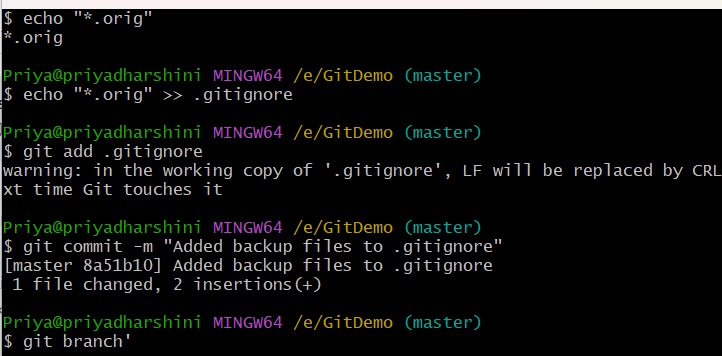
1. Commit the changes to the .gitignore

git add .gitignore

git commit -m "Added backup files to .gitignore"

1. List out all the available branches

git branch



1. Delete the branch, which merge to master.

git branch -d GitWork

1. Observe the log by executing **“git log –oneline –graph –decorate”**

git log --oneline --graph –decorate

