What is GIT?

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files.

 Its goals include speed, data integrity, and support for distributed, non-linear workflows.

PROCESS OF GIT

**Setup and configuration of GIT**

1.Install Git on the system and using git bash run the following commands for git configuration.

pwd

mkdir projects

cd projects

**2.Configuring the git.**

git version

git config --global user.name "prachi615"

git config --global user.email “prachisinha407@gmail.com”

git config --global –list

**3. Cloning the git and adding a file ,commiting the changes and then pushing the changes to master**

git clone github-https-url # paste in your GitHub HTTPS clone URL

ls

cd github-demo

ls

git status echo "Test Git Quick Start demo" >> start.txt

ls

cat start.txt

git status-It lets us see which changes have been staged, which haven’t, and which files aren’t being tracked by Git.

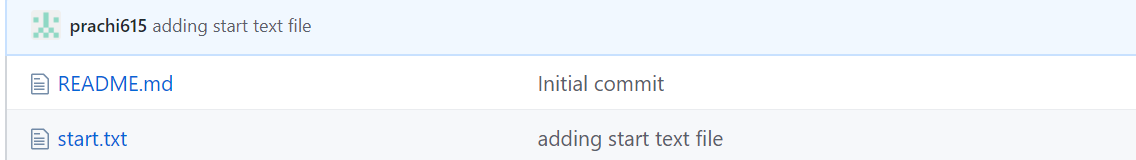
git add start.txt

git status

git commit -m "Adding start text file"

git status

git push origin master



**Feature branches**

**What are feature branches?**

A feature branch is simply a separate branch in your Git repo used to implement a single feature in your project. When “using feature branches,” you are creating a new branch for each new feature you develop, instead of just checking in all your changes into the master branch.

**Why we do branching in git?**

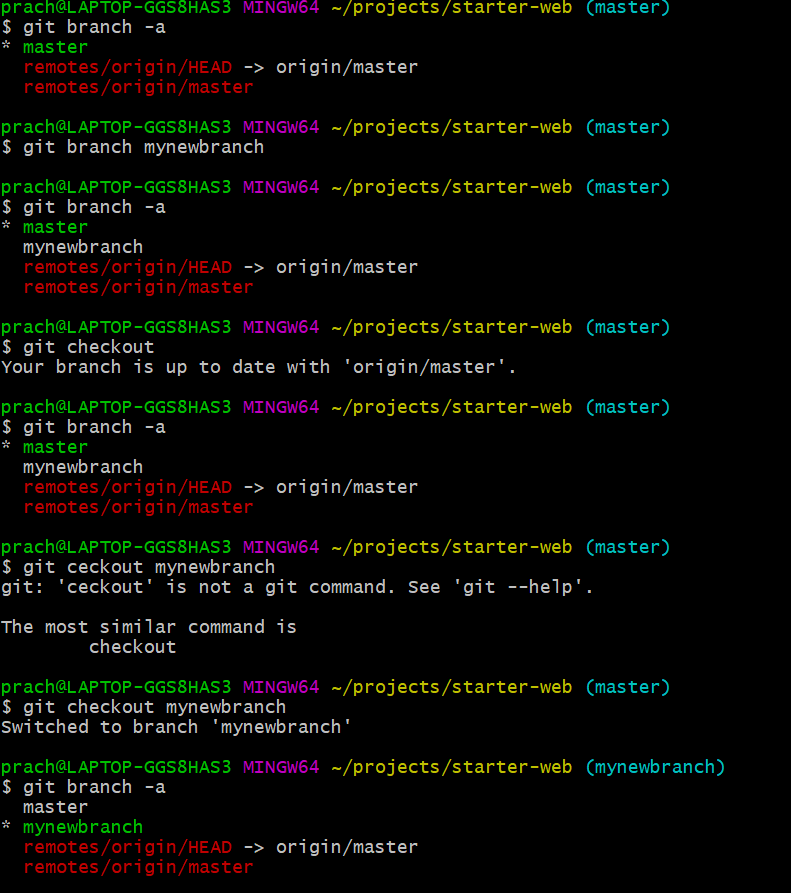
Using branches helps us organize the workflow more efficiently and rather effortlessly.

**What is merging of branches?**

The git merge command lets us take the independent lines of development created by git branch and integrate them into a single branch.

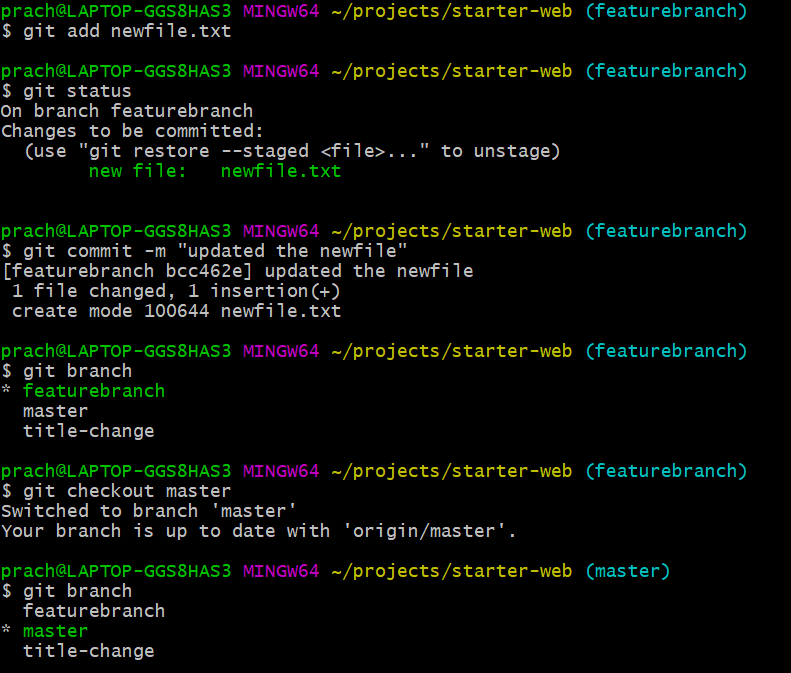
**Creation of feature branches**

STEP 1: Create a feature branch and switch to the new feature branch.

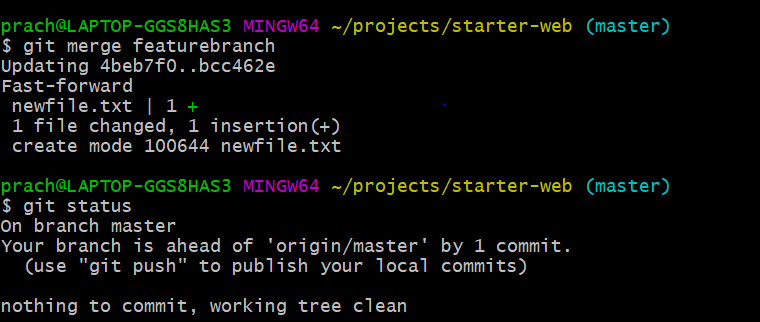


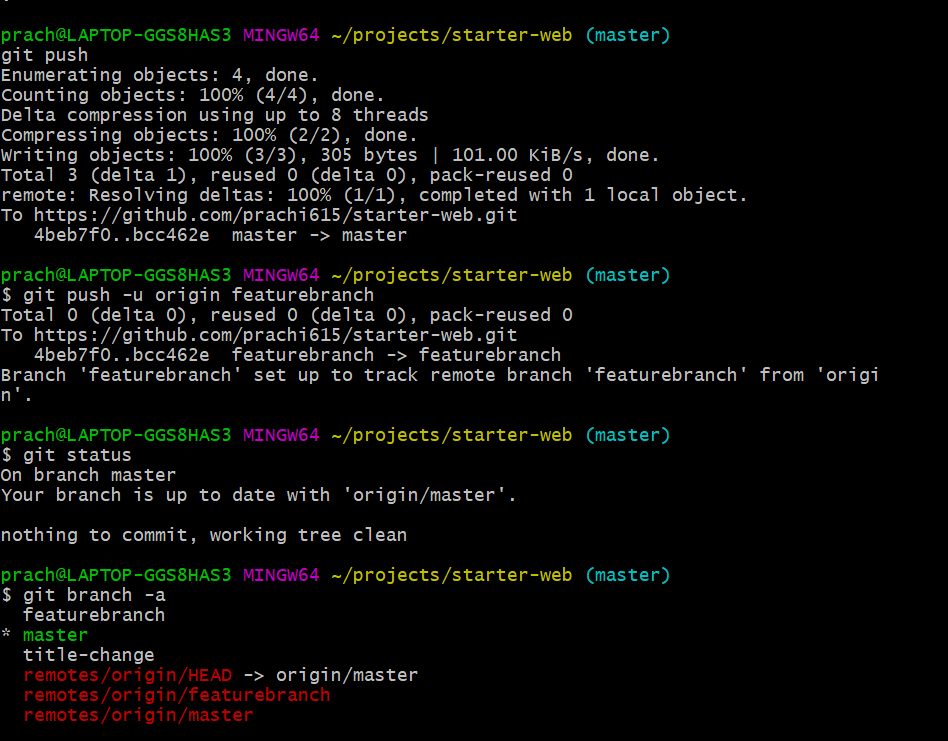
STEP 2: Renaming, deleting of feature branches 

STEP 3: Adding a new file to feature branch and commiting the changes to feature branch.



STEP 4: Merging the feature branches



STEP 5: Pusing the commited changes and making the feature branch as remote branch.

**Conflict resolution**

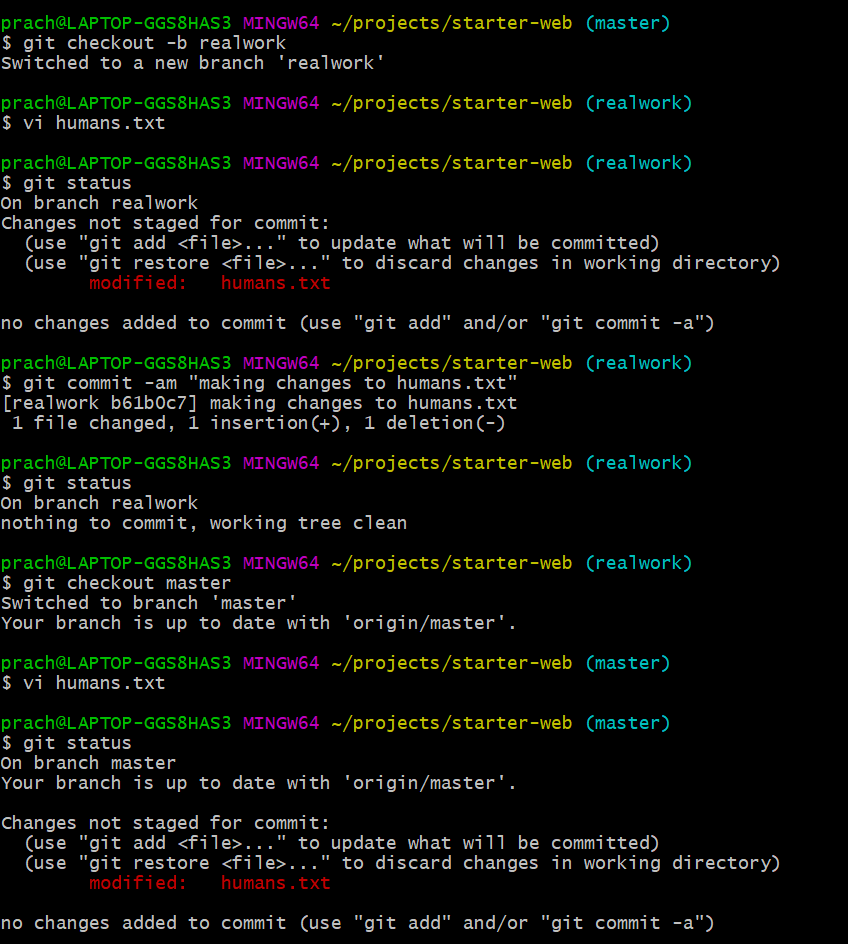
**Why conflicts arise?**

Conflicts generally arise when two people have changed the same lines in a file, or if one developer deleted a file while another developer was modifying it. In these cases, Git cannot automatically determine what is correct.

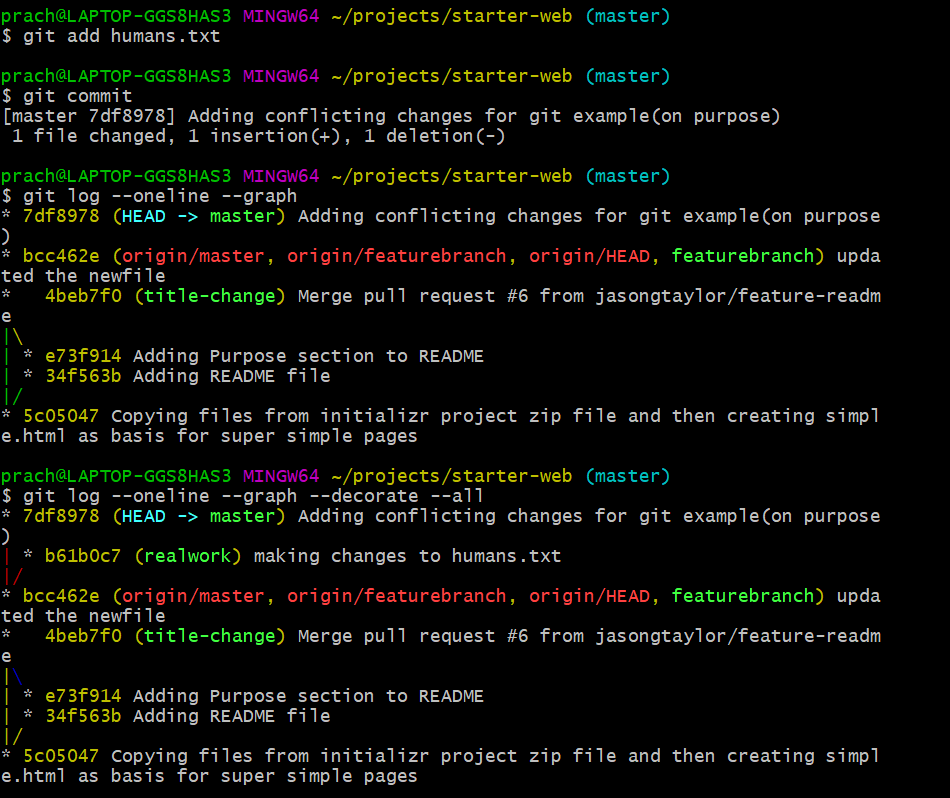
**How to resolve conflicts?**

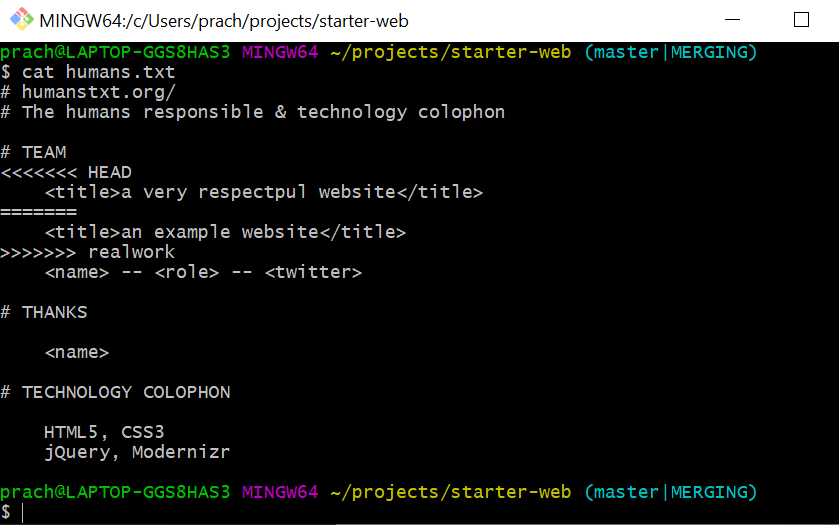
* Identify which files are in conflict (Git should tell you this).
* Open each file and examine the diffs; Git demarcates them.
* Once you've resolved the conflict in a file git add the\_file.
* Once you've resolved all conflicts,try merging the files

STEP 1: Created a conflict for demo purpose in the file humans.txt

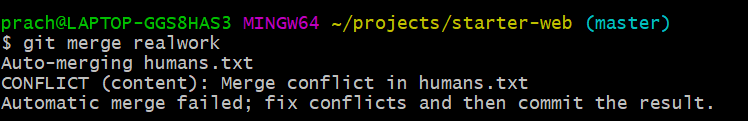


STEP 2:Conflicts are created at two separate branches





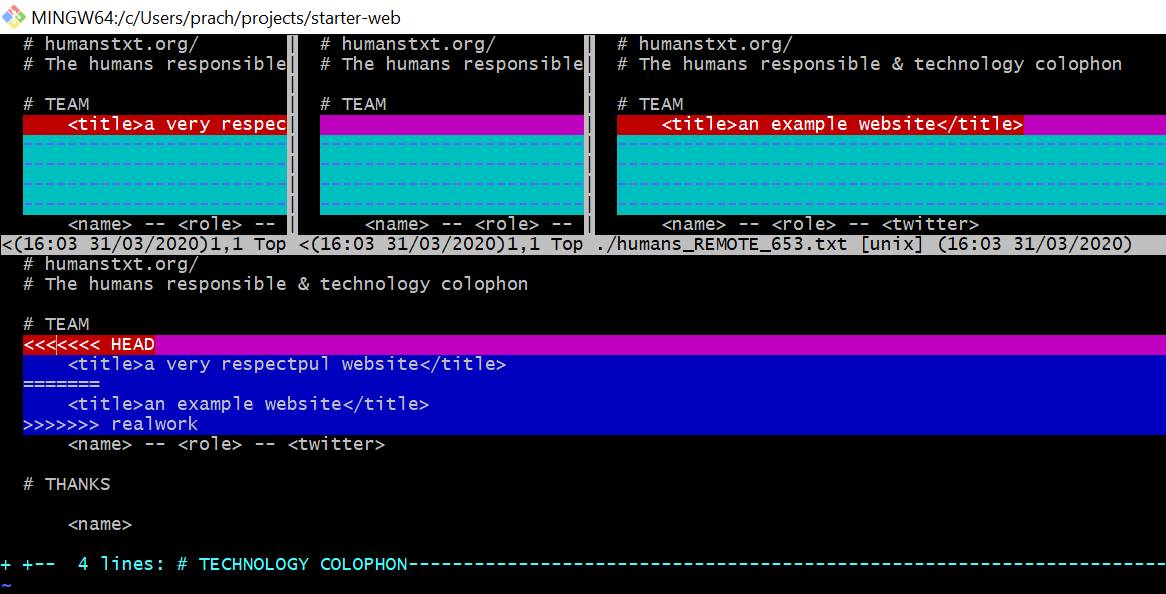
STEP 3:Merge conflict arised.



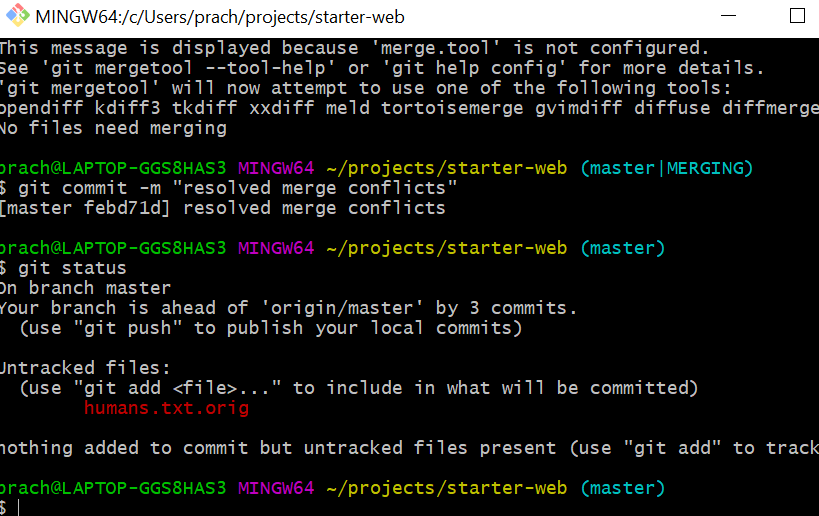
STEP 3:To resolve the conflict use the diff command in git to see the differences in the file .

git diff <<branch1>> <<branch2>>

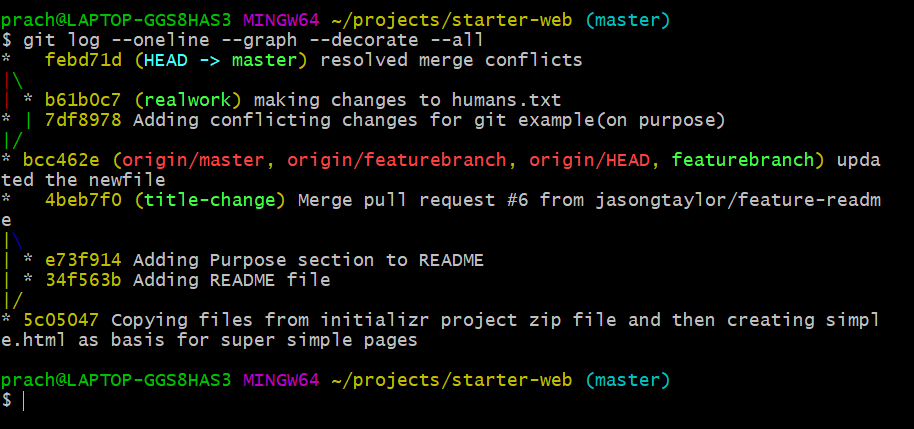
STEP 4:Make changes after opening mergetool to see and make changes whereever a conflict is found.



STEP 5:Once the conflict is resolved check the log to see whether the merge has been resolved or not.



STEP 6:Resolved merge conflicts



**Cloning the branches**

