**Git HandsOn 1**

**Step 1: Setup your machine with Git Configuration**

To create a new repository, signup with GitLab and register your credentials

Login to GitLab and create a “GitDemo” project

1. To check if Git client is installed properly: Open Git bash shell and execute



If output shows Git with its version information that indicates, that Git Client installs properly.

1. To configure user level configuration of user ID and email ID execute

git 

1. To check if the configuration is properly set, execute the following command.



**Step 2: Integrate notepad.exe to Git and make it a default editor**

1. To check, if notepad.exe execute from Git bash



If Git bash could not able to recognize notepad++ command that implies notepad++.exe is note added to the environment path variable.

To add path of notepad++.exe to environment variable, go to control panel -> System -> Advanced System settings. Go to Advanced tab -> Environment variables -> Add path of notepad++.exe to the path user variable by clicking on “Edit”



1. Exit Git bash shell, open bash shell and execute



Now, notepad will open from Git bash shell

1. To create an alias command for notepad.exe, execute



It will open notepad++ from bash shell, and create a user profile by adding the line in notepad++



1. To configure the editor, execute the command

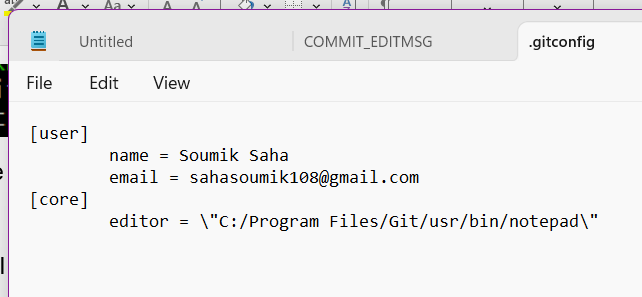


1. To verify if notepad is the default editor, execute the command



Here ‘-e’ option implies editor

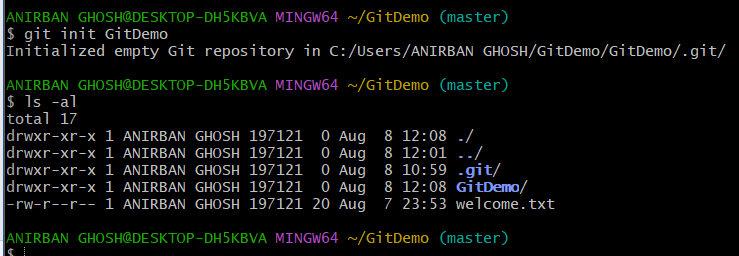
It will show the entire global configuration as shown below,



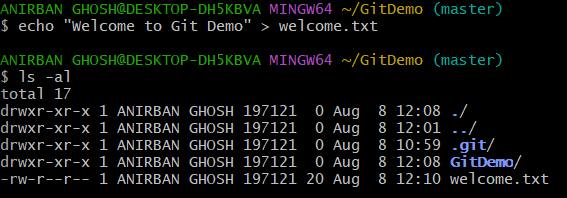
**Step 3: Add a file to source code repository**

1. Open Git bash shell and create a new project “**GitDemo**” by executing the command
2. Git bash initializes the “**GitDemo**” repository. To verify, execute the command

It will display all the hidden files in the Git “working directory”.



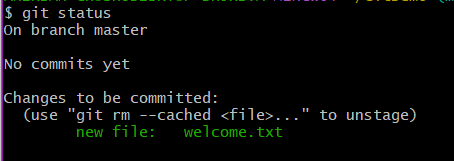
1. To create a file **“welcome.txt”** and add content to the file, execute the command
2. To verify if the file “welcome.txt” is created, execute



1. To verify the content, execute the command



1. Check the status by executing



Now the file **“welcome.txt”** is available in Git “working directory”

1. To make the file to be tracked by Git repository, execute the command

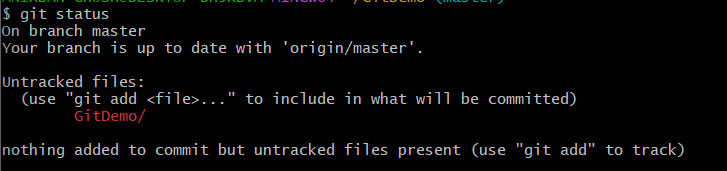


1. To add multi line comments, we are opening default editor to comment. Execute the command



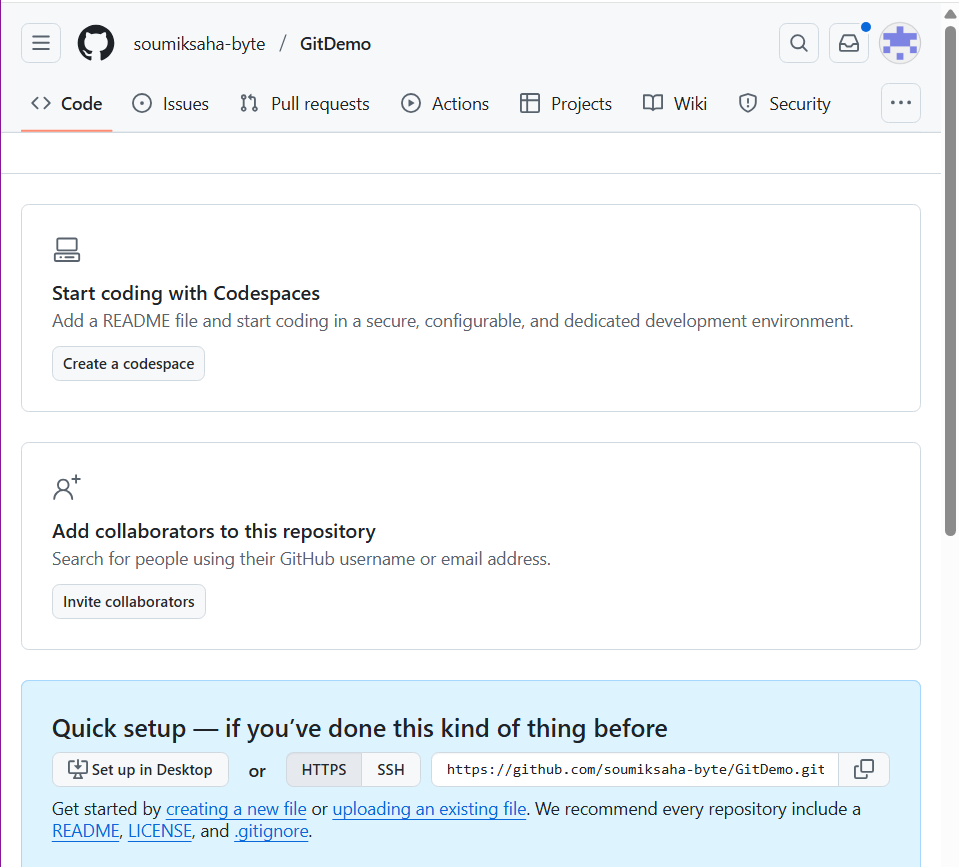
Notepad++ editor will open and to add multi-line comment with default editor

1. To check if local and “Working Directory” git repository are same, execute git status



**welcome.txt** is added to the local repository.

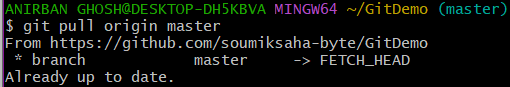
1. Signup with GitLab and create a remote repository **“GitDemo”**

****

****

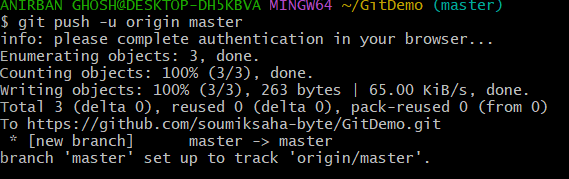
1. To pull the remote repository, execute

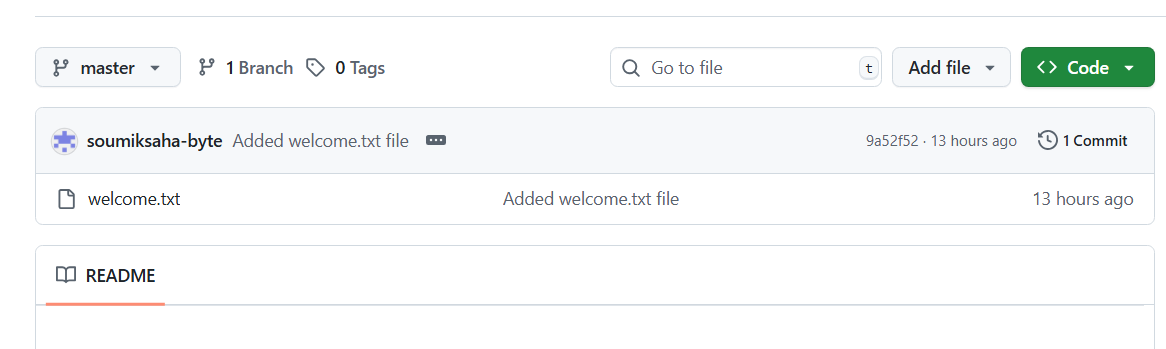
git pull origin master



1. To push the local to remote repository, execute

git push origin master





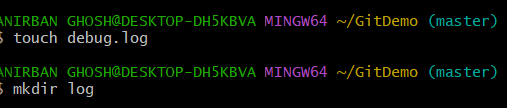
WE CAN SEE THE WELCOME.TXT IS PUSHED

**Git HandsOn 2**

**Create a “.log” file and a log folder in the working directory of Git. Update the .gitignore file in such a way that on committing, these files (.log extensions and log folders) are ignored.**

**= log file created**

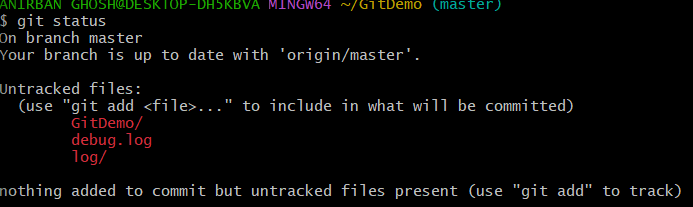
**Log directory made**





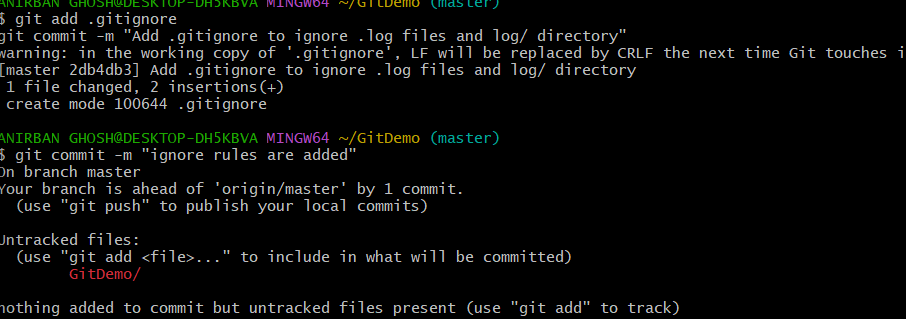
Example\_SOU log file created inside that folder

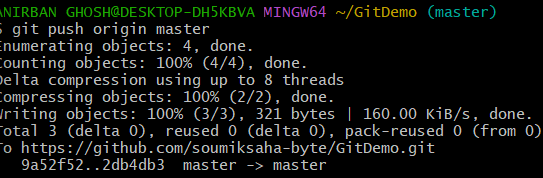
**Verify if the git status reflects the same about working directory, local repository and git repository**



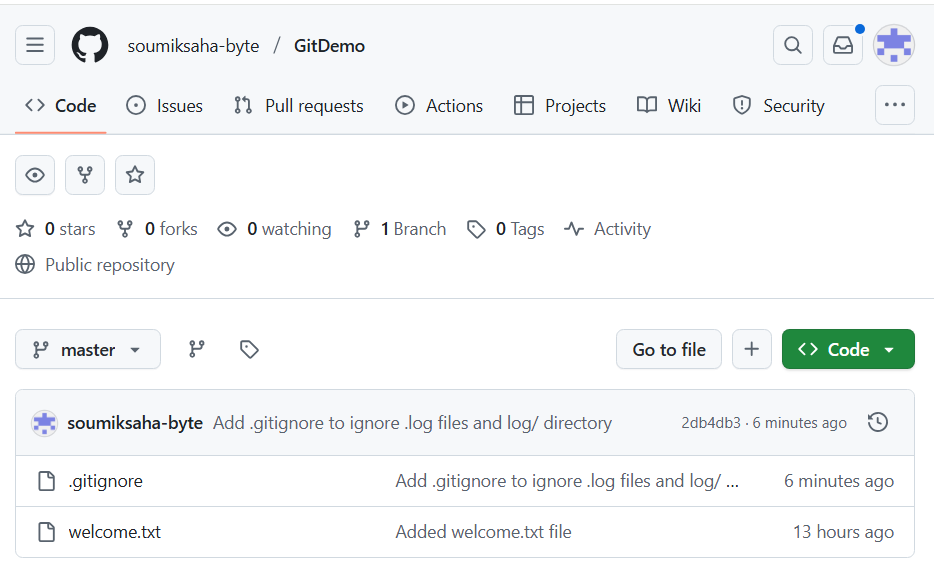
Next added the gitignoore and commit is done

**RESULT:**





**GITHUB:**

****

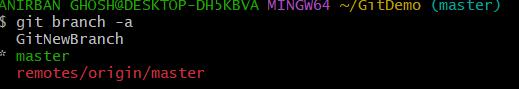
**Git HandsOn 3**

**Branching**

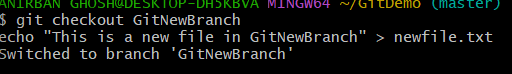
**Create a new branch “GitNewBranch”**



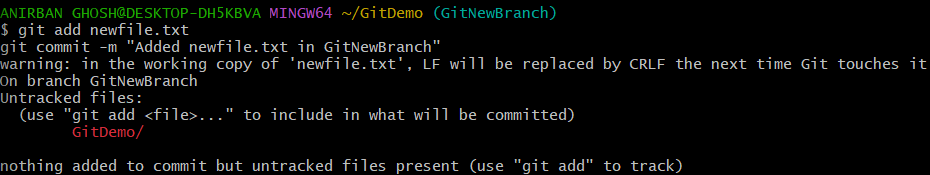
**List all the local and remote branches**

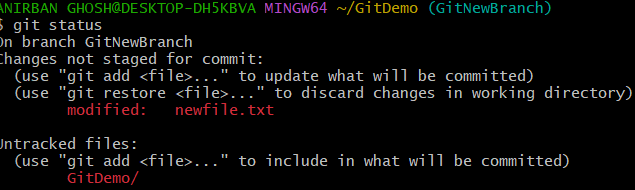


**Switch to the newly created branch & add files**



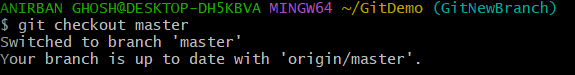
**Commit the changes to the branch**



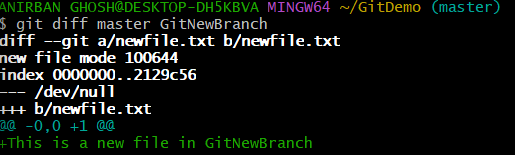
**Check the status** ****

**Merging**

**Switch to master**



**List out all the CLI differences between master and GitNewBranch**



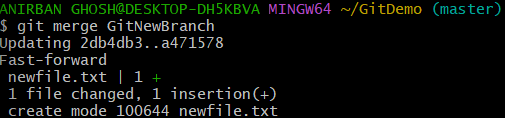
**Visual differences with P4Merge**

git difftool master GitNewBranch

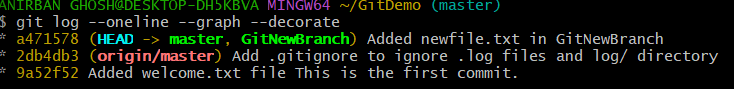
This will open P4Merge for visual comparison.

(its giving me error in my pc)

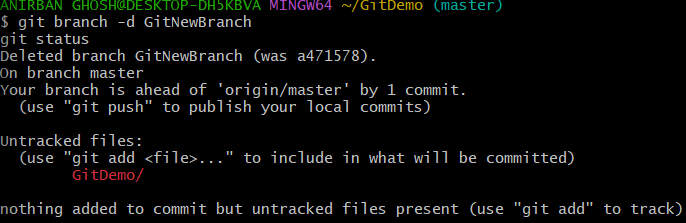
**Merge the branch into master**



**Observe merge history**

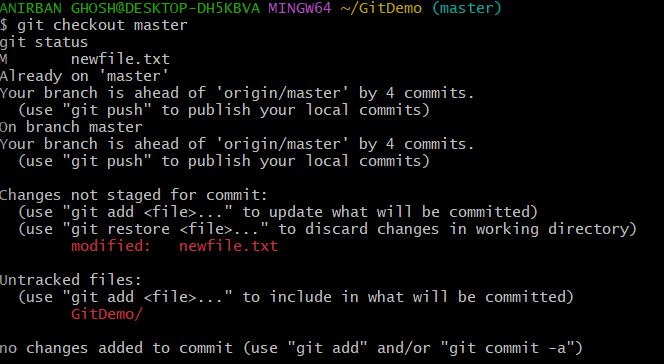
****

**Delete the branch after merging**

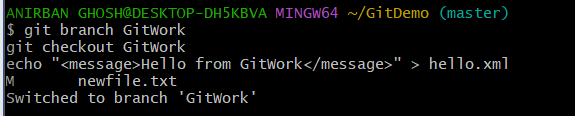


**Git HandsOn 4**

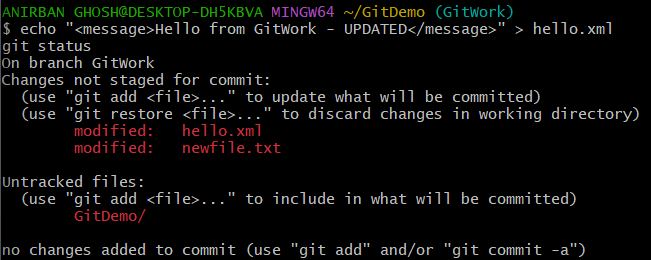
1. **Verify if main is in clean state.**

****

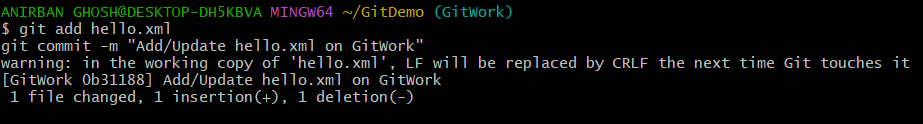
1. **Create a branch “GitWork”. Add a file “hello.xml”.**

****

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

****

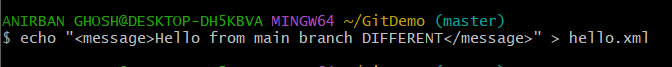
1. **Commit the changes to reflect in the branch**

****

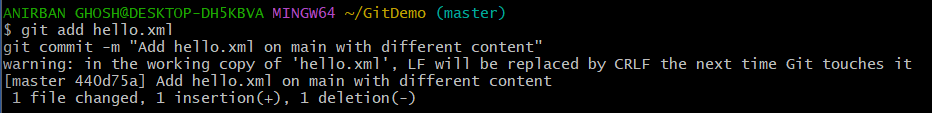
1. **Switch to main.**

****

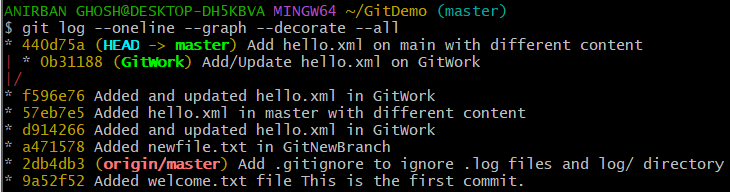
1. **Add a file “hello.xml” to the master and add some different content than previous.**

****

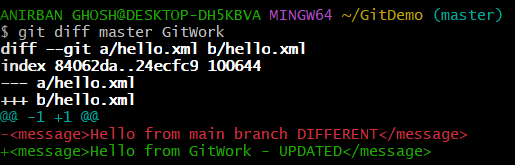
1. **Commit the changes to the master**

****

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

****

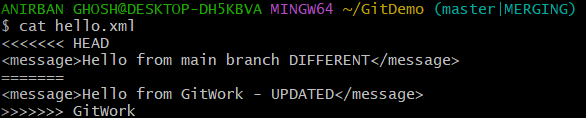
1. **Check the differences with Git diff tool**

****

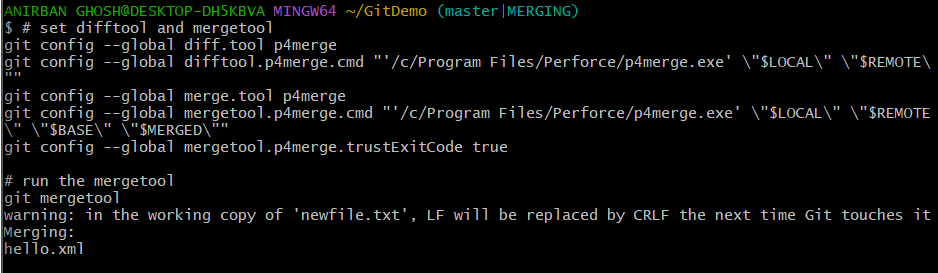
1. **For better visualization, use P4Merge tool to list out all the differences between master and branch**
2. **Merge the bran to the master main**

****

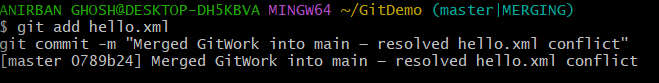
1. **Observe the git mark up.**

****

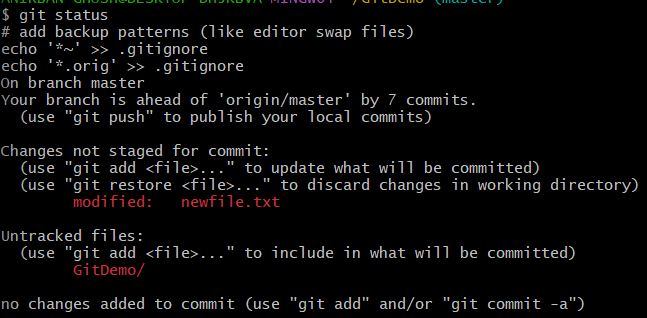
1. **Use 3-way merge tool to resolve the conflict**

****

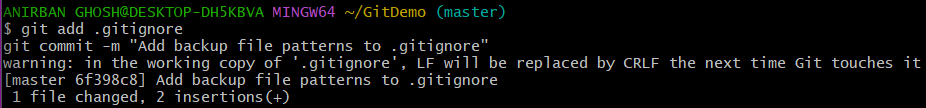
1. **Commit the changes to the master, once done with conflict**

****

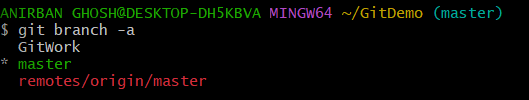
1. **Observe the git status and add backup file to the .gitignore file.**

****

1. **Commit the changes to the .gitignore**

****

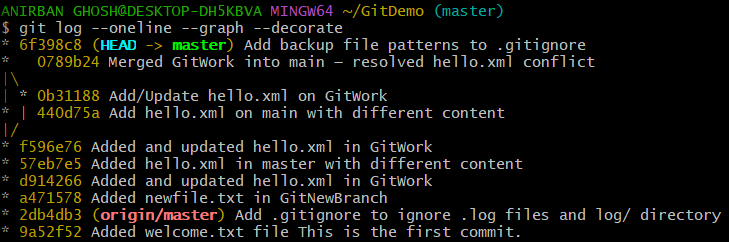
1. **List out all the available branches**



1. **Delete the branch, which merge to master.**

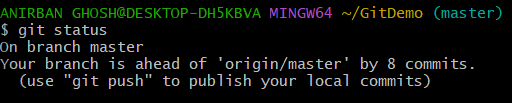
****

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

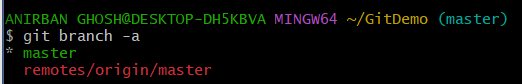
****

**Git HandsOn 5**

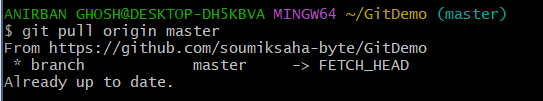
**1. Verify if master is in clean state.**

****

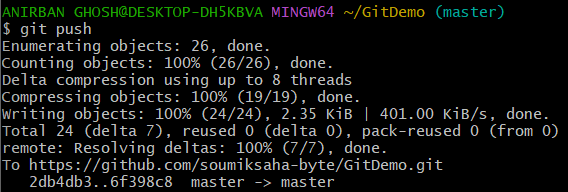
**2. List out all the available branches.**

****

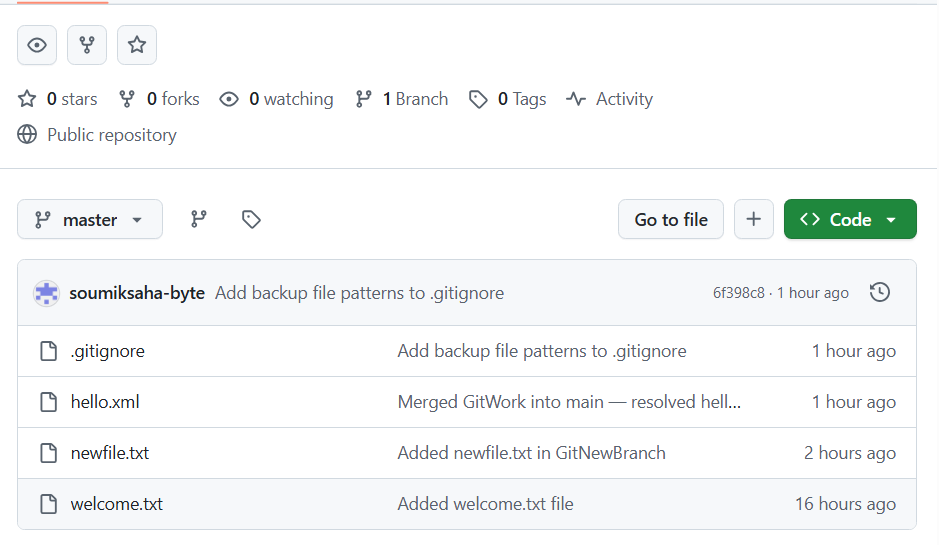
**3. Pull the remote git repository to the master**

****

**4. Push the changes, which are pending from “Git-T03-HOL\_002” to the remote  
repository.**

****

**5. Observe if the changes are reflected in the remote repository.**

****

YES CHANGES ARE OBSERVED