GIT

1. GIT-HOL HANDSON

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

```
$ git version
git version 2.21.0.windows.1
```

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

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

```
$ git config --global user.name "username"
```

Step 3: Add a file to source code repository

1. Open Git bash shell and create a new project “**GitDemo**” by executing the command

```
$ git init GitDemo
Initialized empty Git repository in D:/Development_Avecto/GitDemo/.git/
```

2. Git bash initializes the “**GitDemo**” repository. To verify, execute the command

```
$ ls -al
total 8
drwxr-xr-x 1        1049089 0 Jan 13 11:54 ./
drwxr xr x 1        1049089 0 Jan 13 11:54 ../
drwxr-xr-x 1        1049089 0 Jan 13 11:54 .git/
```

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

3. To create a file **“welcome.txt”** and add content to the file, execute the command

```
$ echo "Welcome to the version control" >> welcome.txt
```

4. To verify if the file “welcome.txt” is created, execute

```
$ ls -al
total 9
drwxr-xr-x 1 494096 1049089  0 Jan 13 12:02 ./
drwxr-xr-x 1 494096 1049089  0 Jan 13 11:54 ../
drwxr-xr-x 1 494096 1049089  0 Jan 13 12:01 .git/
-rw-r--r-- 1 494096 1049089 31 Jan 13 12:02 welcome.txt
```

5. To verify the content, execute the command

```
$ cat welcome.txt
Welcome to the version control
```

6. Check the status by executing

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        welcome.txt
```

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

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

```
$ git add welcome.txt
warning: LF will be replaced by CRLF in welcome.txt.
The file will have its original line endings in your working directory
```

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

```
$ git commit
```

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

9. To check if local and "Working Directory" git repository are same, execute git status

```
$ git config --global user.email "username@cognizant.com"
```

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

```
$ git config --global --list
user.name=username
user.email=username@cognizant.com
```

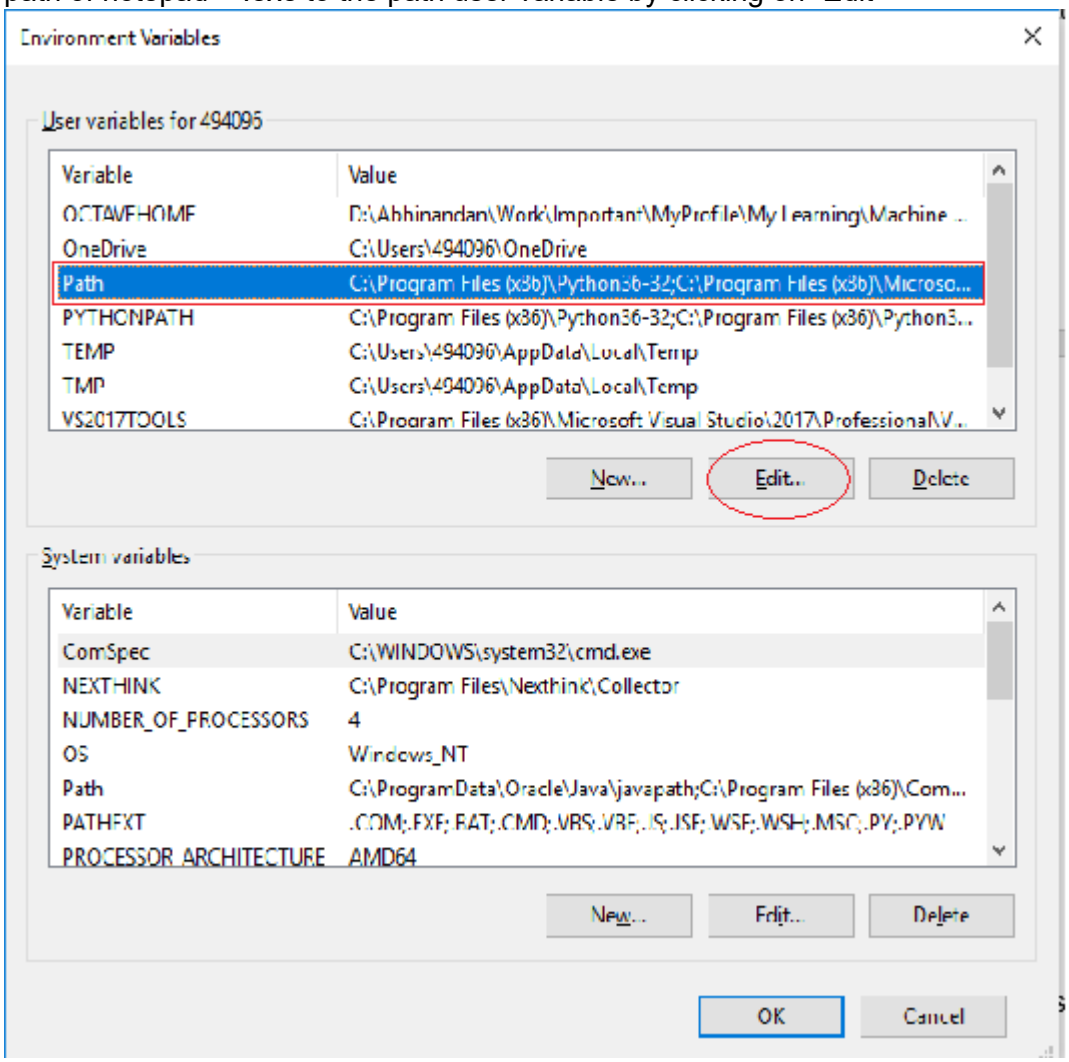
Step 2: Integrate notepad++.exe to Git and make it a default editor

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

```
$ notepad++
bash: notepad++: command not found
```

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”



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

```
$ notepad++
```

Now, notepad++ will open from Git bash shell

3. To create an alias command for notepad++.exe, execute

```
$ notepad++.exe bash -profile
```

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

```
alias npp='notepad++.exe -multiInst -nosession'
```

4. To configure the editor, execute the command

```
$ git config --global core.editor "notepad++.exe -multiInst -nosession"
```

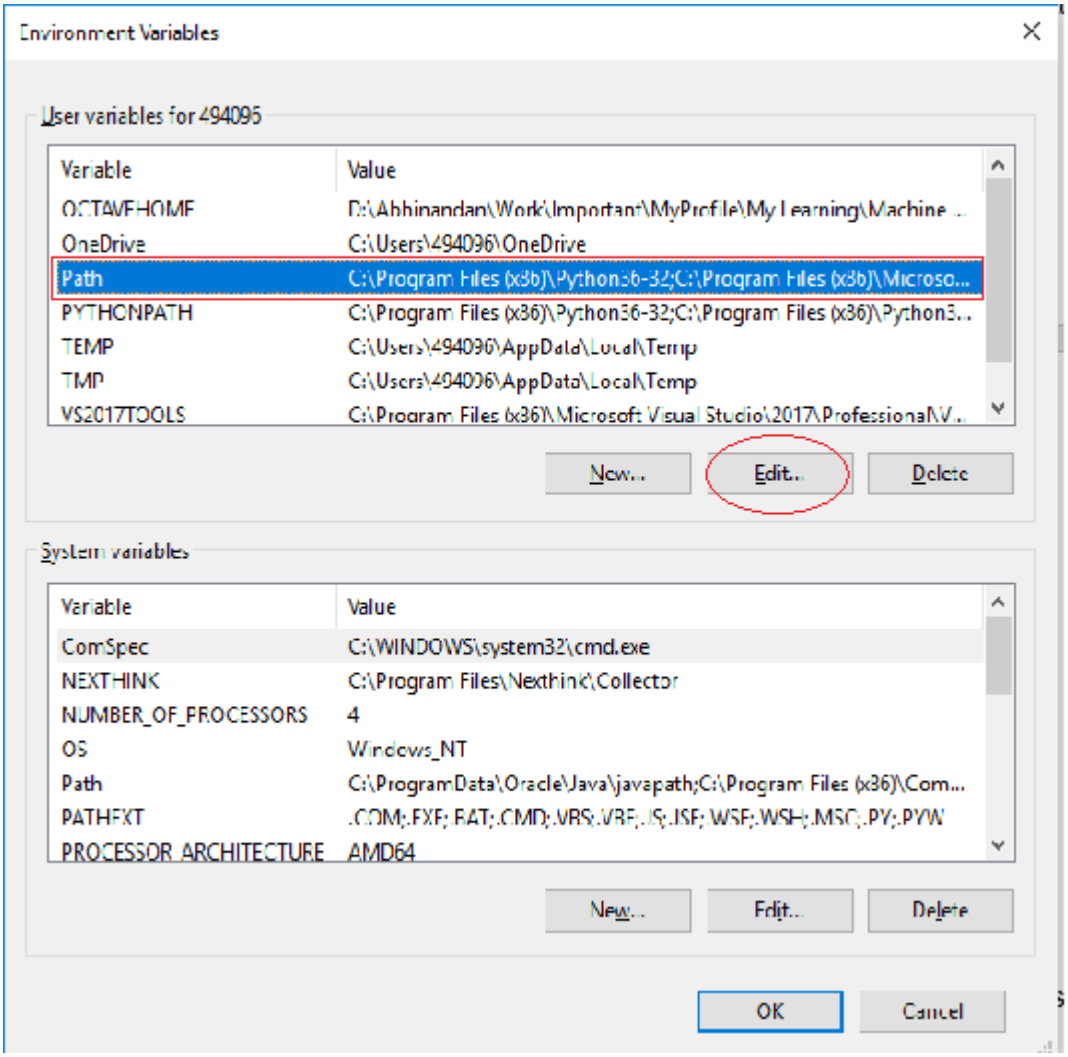
5. To verify if notepad++ is the default editor, execute the command

```
$ git config --global -e
hint: Waiting for your editor to close the file...
```

Here '-e' option implies editor

It will show the entire global configuration as shown below,

```
[user]
    name = username
    email = username@cognizant.com
[core]
    editor = notepad++.exe -multiInst -nosession
```



6. Exit Git bash shell, open bash shell and execute
7. Check the status by executing

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        welcome.txt
```

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

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

```
$ git add welcome.txt
warning: LF will be replaced by CRLF in welcome.txt.
The file will have its original line endings in your working directory
```

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

```
$ git commit
```

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

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

```
$ git status
On branch master
nothing to commit, working tree clean
```

welcome.txt is added to the local repository.

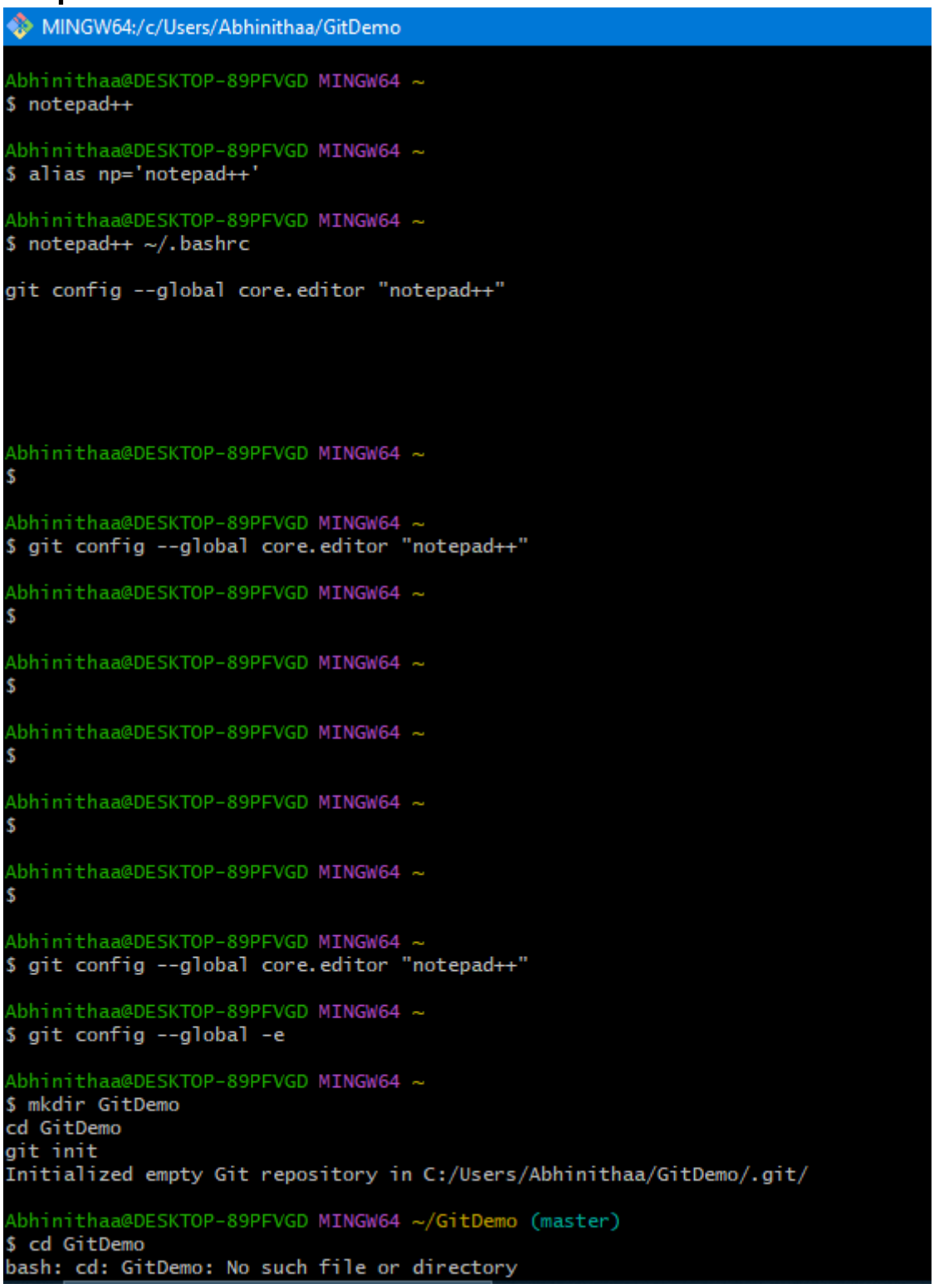
11. Signup with GitLab and create a remote repository **“GitDemo”**
12. To pull the remote repository, execute

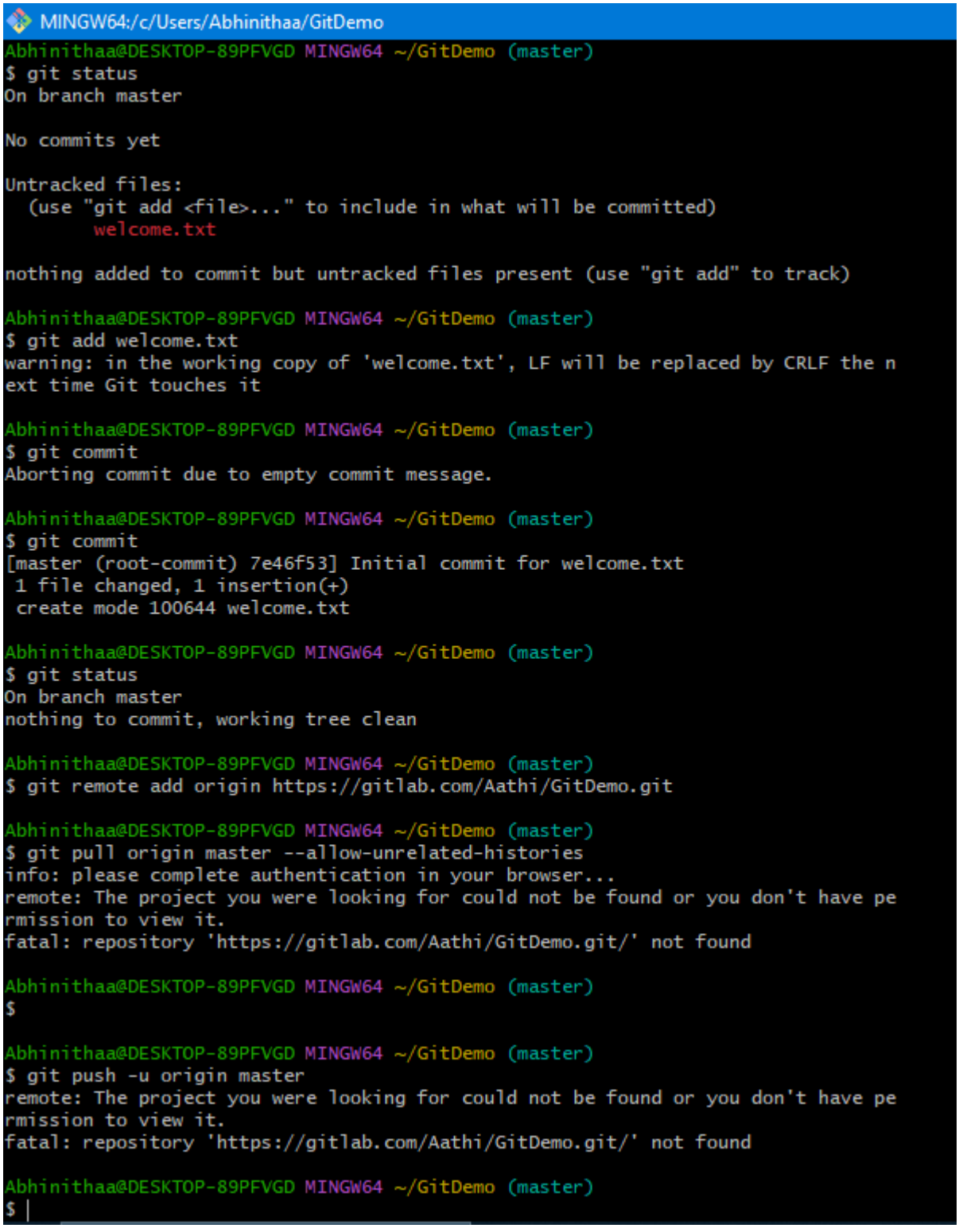
git pull origin master

13. To push the local to remote repository, execute

git push origin master

Output:


MINGW64:/c/Users/Abhinithaa/GitDemo
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$ notepad++
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$ alias np='notepad++'
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$ notepad++ ~/.bashrc
git config --global core.editor "notepad++"
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$ git config --global core.editor "notepad++"
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$ git config --global core.editor "notepad++"
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$ git config --global -e
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$ mkdir GitDemo
cd GitDemo
git init
Initialized empty Git repository in C:/Users/Abhinithaa/GitDemo/.git/
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ cd GitDemo
bash: cd: GitDemo: No such file or directory


MINGW64:/c/Users/Abhinithaa/GitDemo
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git status
On branch master
No commits yet
Untracked files:
(use "git add <file>..." to include in what will be committed)
welcome.txt
nothing added to commit but untracked files present (use "git add" to track)
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git add welcome.txt
warning: in the working copy of 'welcome.txt', LF will be replaced by CRLF the n
ext time Git touches it
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git commit
Aborting commit due to empty commit message.
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git commit
[master (root-commit) 7e46f53] Initial commit for welcome.txt
1 file changed, 1 insertion(+)
create mode 100644 welcome.txt
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git status
On branch master
nothing to commit, working tree clean
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git remote add origin https://gitlab.com/Aathi/GitDemo.git
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git pull origin master --allow-unrelated-histories
info: please complete authentication in your browser...
remote: The project you were looking for could not be found or you don't have pe
rmission to view it.
fatal: repository 'https://gitlab.com/Aathi/GitDemo.git/' not found
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git push -u origin master
remote: The project you were looking for could not be found or you don't have pe
rmission to view it.
fatal: repository 'https://gitlab.com/Aathi/GitDemo.git/' not found
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$

2. GIT_ HOL HANDSON

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.

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

1. **Go to Project Directory**

```
cd ~/GitDemo  # or wherever your Git repo is
```

2. Create a .log File and a log Folder

```
echo "this is a log file" > debug.log
mkdir log
echo "logs should be ignored" > log/error.txt
```

3. Create and Edit `.gitignore`

```
notepad++ .gitignore
```

4. Add the Following Lines

```
*.log
log/
```

5. **Git status**

```
git status
```

6. Commit Changes

```
git add .gitignore
git commit -m "Add .gitignore to exclude log files and log folder"
```

7. **Push changes**

```
git push origin master
```

Output:

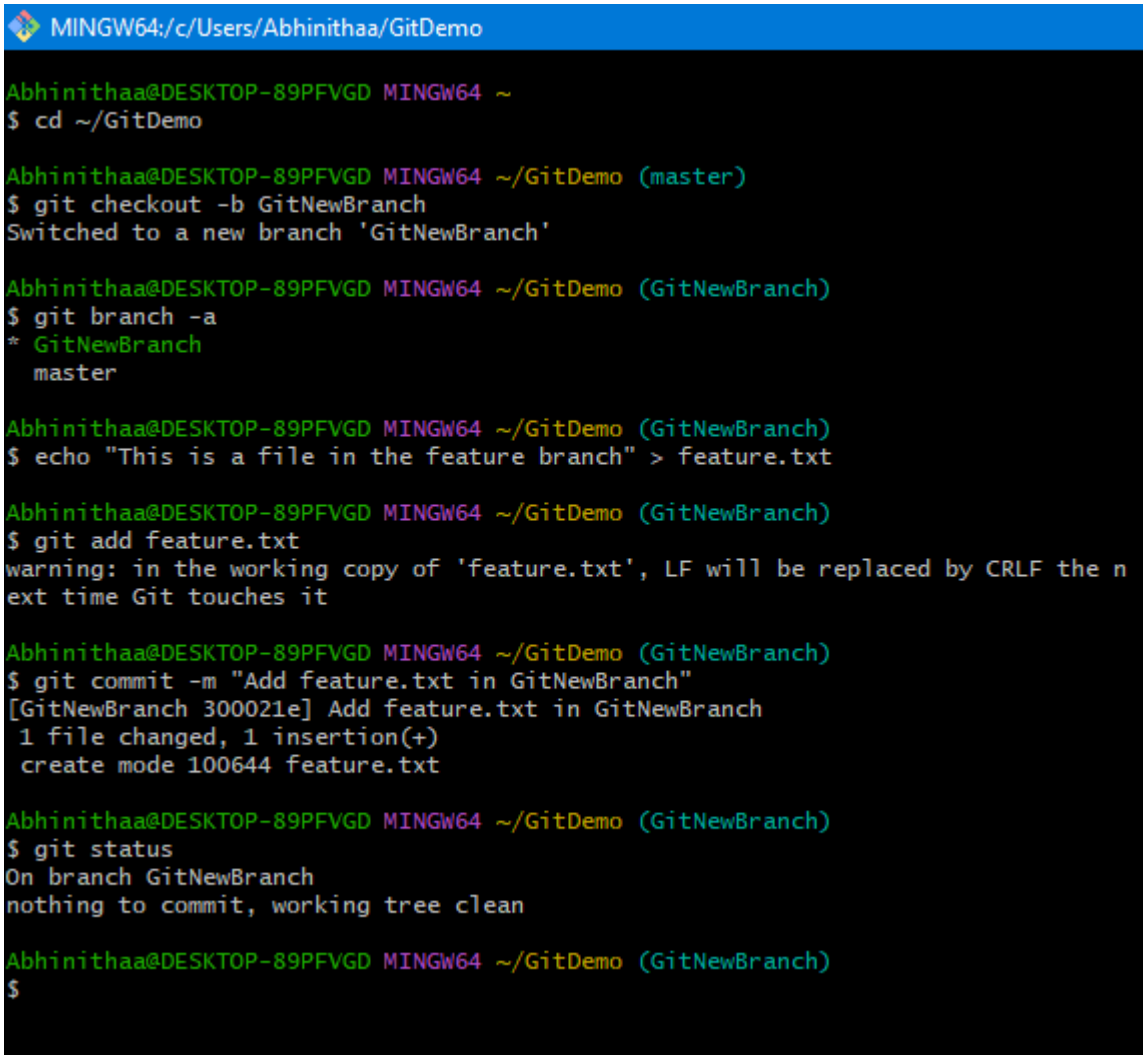
```
This looks like an incorrect setup.
A ~/.bash_profile that loads ~/.bashrc will be created for you.

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~
$ cd ~/GitDemo  # or wherever your Git repo is

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ echo "this is a log file" > debug.log

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ mkdir log

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ echo "logs should be ignored" > log/error.txt

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ notepad++ .gitignore

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git add .gitignore
git commit -m "Add .gitignore to exclude log files and log folder"
[master e14e566] Add .gitignore to exclude log files and log folder
 1 file changed, 2 insertions(+)
 create mode 100644 .gitignore

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git push origin master
remote: The project you were looking for could not be found or you don't have pe
rmission to view it.
```

3. GIT-HOL-HANDSON

Branching:

1. Create a new branch **“GitNewBranch”.**
2. List all the local and remote branches available in the current trunk. Observe the “*” mark which denote the current pointing branch.
3. Switch to the newly created branch. Add some files to it with some contents.
4. Commit the changes to the branch.
5. Check the status with **“git status”** command.

Output:



Merging:

1. Switch to the master
2. List out all the differences between trunk and branch. These provide the differences in command line interface.

3. List out all the visual differences between master and branch using **P4Merge tool**.
4. Merge the source branch to the trunk.
5. Observe the logging after merging using **"git log –oneline –graph –decorate"**
6. Delete the branch after merging with the trunk and observe the git status.

Output:

```
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git checkout master
Already on 'master'

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git diff master GitNewBranch

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git difftool master GitNewBranch

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git merge GitNewBranch
Merge made by the 'ort' strategy.

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git log --oneline --graph --decorate
*   ebc4323 (HEAD -> master) Merge branch 'GitNewBranch'
|\
| * 300021e (GitNewBranch) Add feature.txt in GitNewBranch
* | ac9b285 Add feature.txt in GitNewBranch
|/
* e14e566 Add .gitignore to exclude log files and log folder
* 7e46f53 Initial commit for welcome.txt

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git branch -d GitNewBranch
git status
Deleted branch GitNewBranch (was 300021e).
On branch master
nothing to commit, working tree clean

Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/GitDemo (master)
$ git push origin master
git push origin GitNewBranch
remote: The project you were looking for could not be found or you don't have pe
rmission to view it.
```

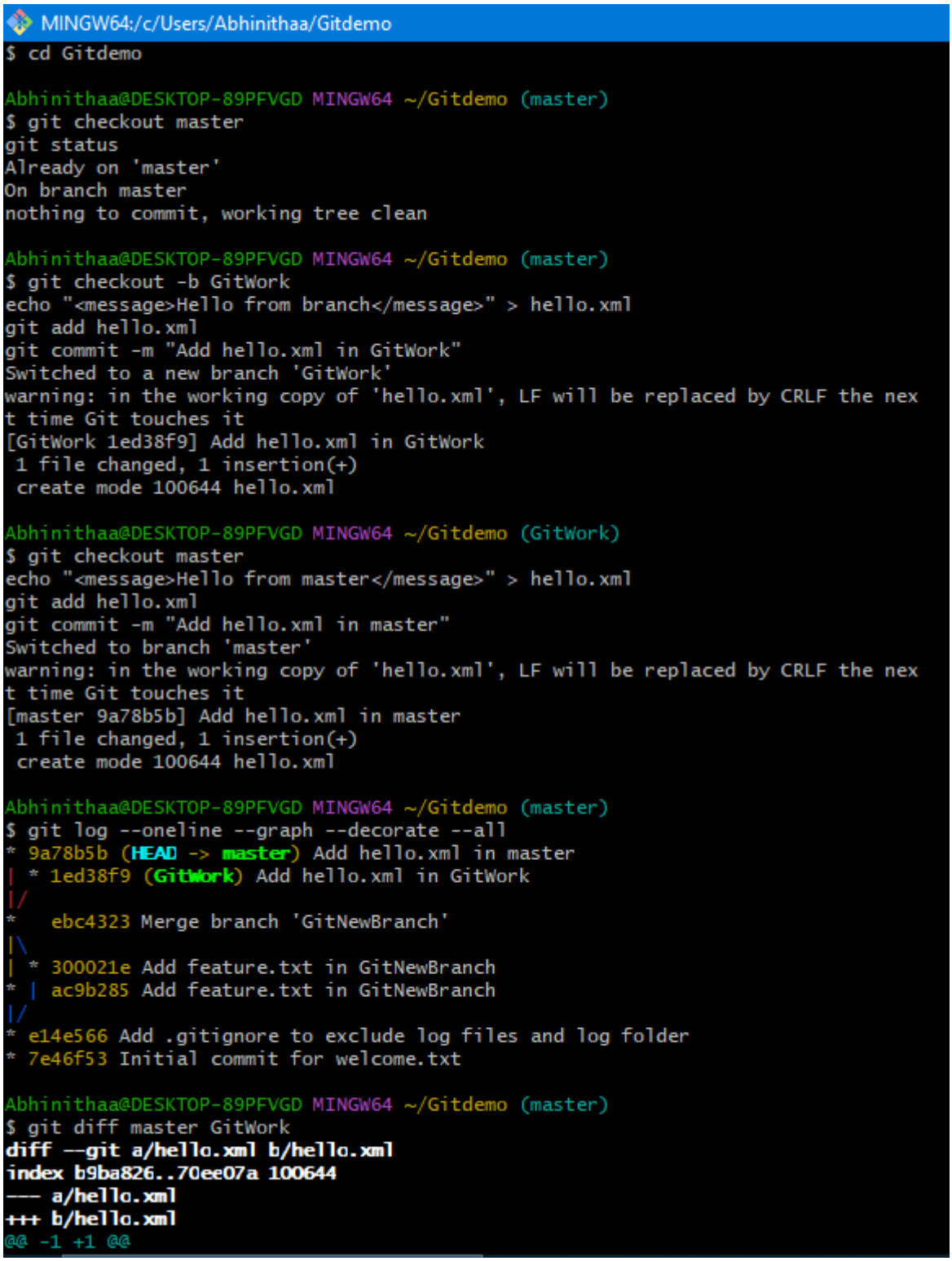
4. GIT-HOL-HANDSON

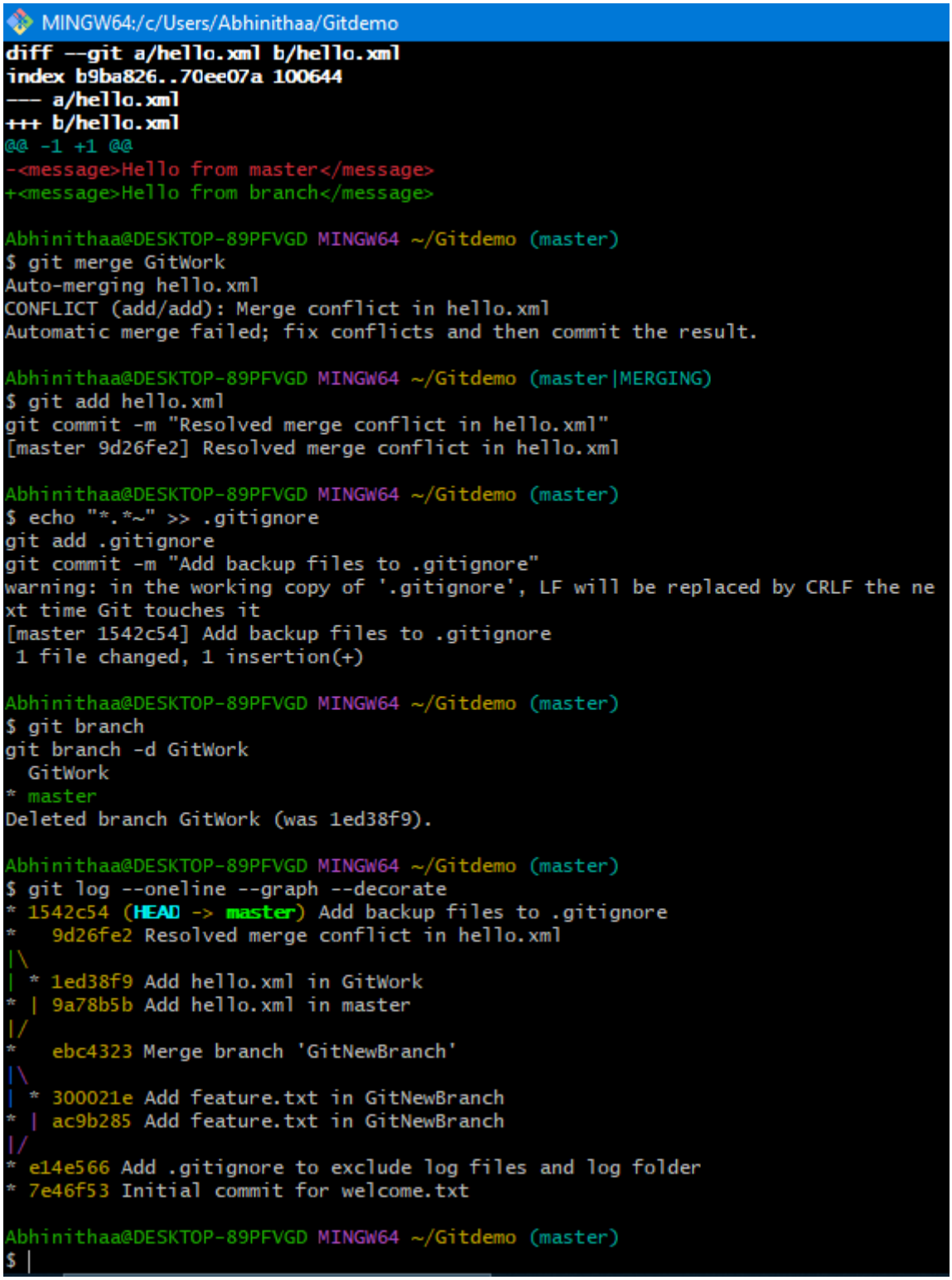
Implement conflict resolution when multiple users are updating the trunk (or master) in such a way that it results into a conflict with the branch's modification.

Steps:

1. Verify if master is in clean state.
2. Create a branch **"GitWork".** Add a file "hello.xml".
3. Update the content of "hello.xml" and observe the status
4. Commit the changes to reflect in the branch
5. Switch to master.
6. Add a file **"hello.xml"** to the master and add some different content than previous.
7. Commit the changes to the master
8. Observe the log by executing **"git log –oneline –graph –decorate –all"**
9. Check the differences with Git diff tool
10. For better visualization, use P4Merge tool to list out all the differences between master and branch
11. Merge the bran to the master
12. Observe the git mark up.
13. Use 3-way merge tool to resolve the conflict
14. Commit the changes to the master, once done with conflict
15. Observe the git status and add backup file to the .gitignore file.
16. Commit the changes to the .gitignore
17. List out all the available branches
18. Delete the branch, which merge to master.
19. Observe the log by executing **"git log –oneline –graph –decorate"**

Output:


MINGW64:/c/Users/Abhinithaa/Gitdemo
$ cd Gitdemo
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/Gitdemo (master)
$ git checkout master
git status
Already on 'master'
On branch master
nothing to commit, working tree clean
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/Gitdemo (master)
$ git checkout -b GitWork
echo "<message>Hello from branch</message>" > hello.xml
git add hello.xml
git commit -m "Add hello.xml in GitWork"
Switched to a new branch 'GitWork'
warning: in the working copy of 'hello.xml', LF will be replaced by CRLF the nex
t time Git touches it
[GitWork 1ed38f9] Add hello.xml in GitWork
1 file changed, 1 insertion(+)
create mode 100644 hello.xml
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/Gitdemo (GitWork)
$ git checkout master
echo "<message>Hello from master</message>" > hello.xml
git add hello.xml
git commit -m "Add hello.xml in master"
Switched to branch 'master'
warning: in the working copy of 'hello.xml', LF will be replaced by CRLF the nex
t time Git touches it
[master 9a78b5b] Add hello.xml in master
1 file changed, 1 insertion(+)
create mode 100644 hello.xml
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/Gitdemo (master)
$ git log --oneline --graph --decorate --all
* 9a78b5b (HEAD -> master) Add hello.xml in master
| * 1ed38f9 (GitWork) Add hello.xml in GitWork
|/
* ebc4323 Merge branch 'GitNewBranch'
|\
| * 300021e Add feature.txt in GitNewBranch
* | ac9b285 Add feature.txt in GitNewBranch
|/
* e14e566 Add .gitignore to exclude log files and log folder
* 7e46f53 Initial commit for welcome.txt
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/Gitdemo (master)
$ git diff master GitWork
diff --git a/hello.xml b/hello.xml
index b9ba826..70ee07a 100644
--- a/hello.xml
+++ b/hello.xml
@@ -1 +1 @@

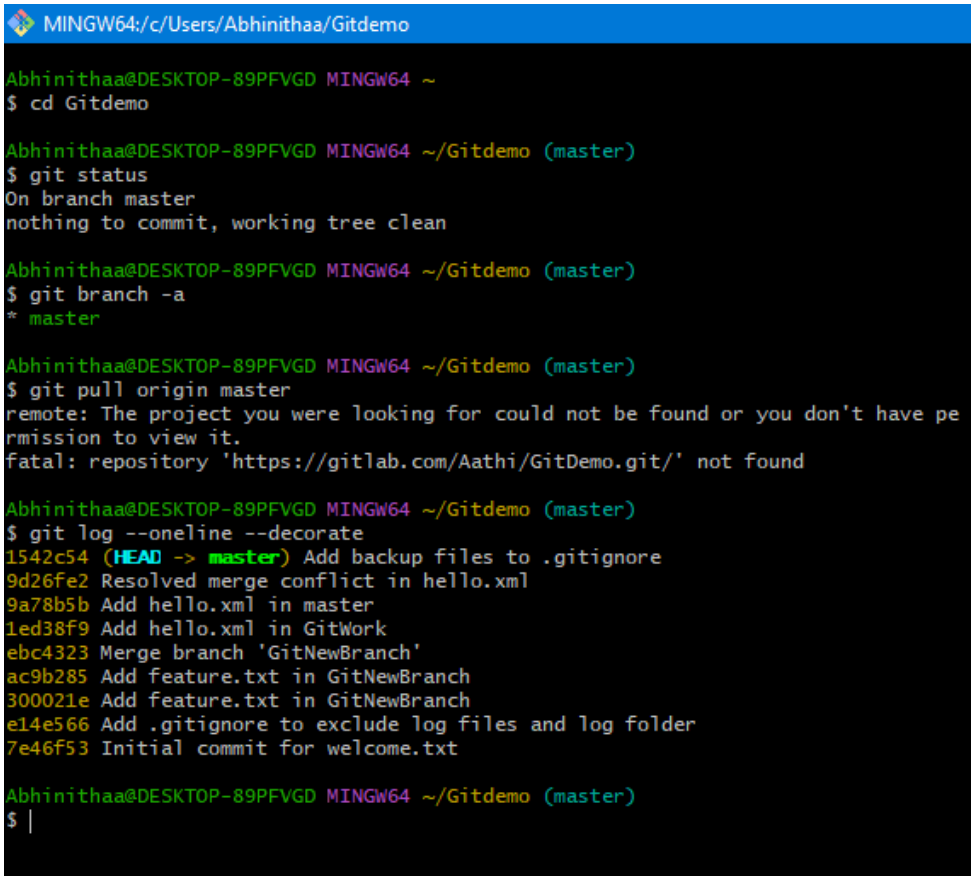

MINGW64:/c/Users/Abhinithaa/Gitdemo
diff --git a/hello.xml b/hello.xml
index b9ba826..70ee07a 100644
--- a/hello.xml
+++ b/hello.xml
@@ -1 +1 @@
-<message>Hello from master</message>
+<message>Hello from branch</message>
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/Gitdemo (master)
$ git merge GitWork
Auto-merging hello.xml
CONFLICT (add/add): Merge conflict in hello.xml
Automatic merge failed; fix conflicts and then commit the result.
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/Gitdemo (master|MERGING)
$ git add hello.xml
git commit -m "Resolved merge conflict in hello.xml"
[master 9d26fe2] Resolved merge conflict in hello.xml
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/Gitdemo (master)
$ echo "*.*~" >> .gitignore
git add .gitignore
git commit -m "Add backup files to .gitignore"
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the ne
xt time Git touches it
[master 1542c54] Add backup files to .gitignore
1 file changed, 1 insertion(+)
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/Gitdemo (master)
$ git branch
git branch -d GitWork
GitWork
* master
Deleted branch GitWork (was 1ed38f9).
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/Gitdemo (master)
$ git log --oneline --graph --decorate
* 1542c54 (HEAD -> master) Add backup files to .gitignore
* 9d26fe2 Resolved merge conflict in hello.xml
|\
| * 1ed38f9 Add hello.xml in GitWork
* | 9a78b5b Add hello.xml in master
|/
* ebc4323 Merge branch 'GitNewBranch'
|\
| * 300021e Add feature.txt in GitNewBranch
* | ac9b285 Add feature.txt in GitNewBranch
|/
* e14e566 Add .gitignore to exclude log files and log folder
* 7e46f53 Initial commit for welcome.txt
Abhinithaa@DESKTOP-89PFVGD MINGW64 ~/Gitdemo (master)
$

5. GIT-HOL-HANDSON

Explain how to clean up and push back to remote Git

Execute steps involving clean up and push back to remote Git.

Steps:

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.

Output: