

EXERCISE - 1

Objectives

Familiar with Git commands like git init, git status, git add, git commit, git push, and git pull.

In this hands-on lab, you will learn how to

- Setup your machine with Git Configuration
- Integrate notepad++.exe to Git and make it a default editor
- Add a file to source code repository

Prerequisites

- Install Git Bash client in your machine

Notes*:

Please follow the below steps for creating a free account in GitHub.

Don't use cognizant credentials to login to GitHub.

Estimated time to complete this lab: **30 minutes**.

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

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

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

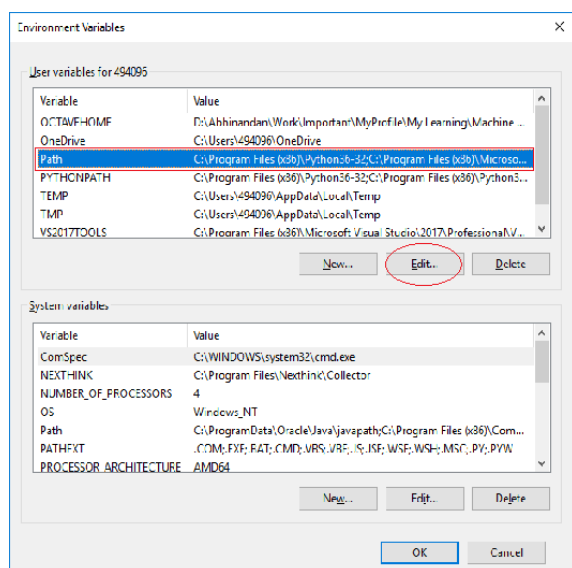
```
$ git config --global --list
user.name=username
user.email=username@cognizant.com
```

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 not 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
```

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

welcome.txt is added to the local repository.

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

11. To pull the remote repository, execute

`git pull origin master`

12. To push the local to remote repository, execute

`git push origin master`

SOLUTION

1. Download and install Git and verify the installation by typing `git version`

```
shelian@DESKTOP-HH52HTM MINGW32 ~ (master)
$ git version
git version 2.44.0.windows.1
```

2. Create an account in GitLab and perform user level configuration by:

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

```
shelian@DESKTOP-HH52HTM MINGW32 ~ (master)
$ git config --global user.name "shelian"

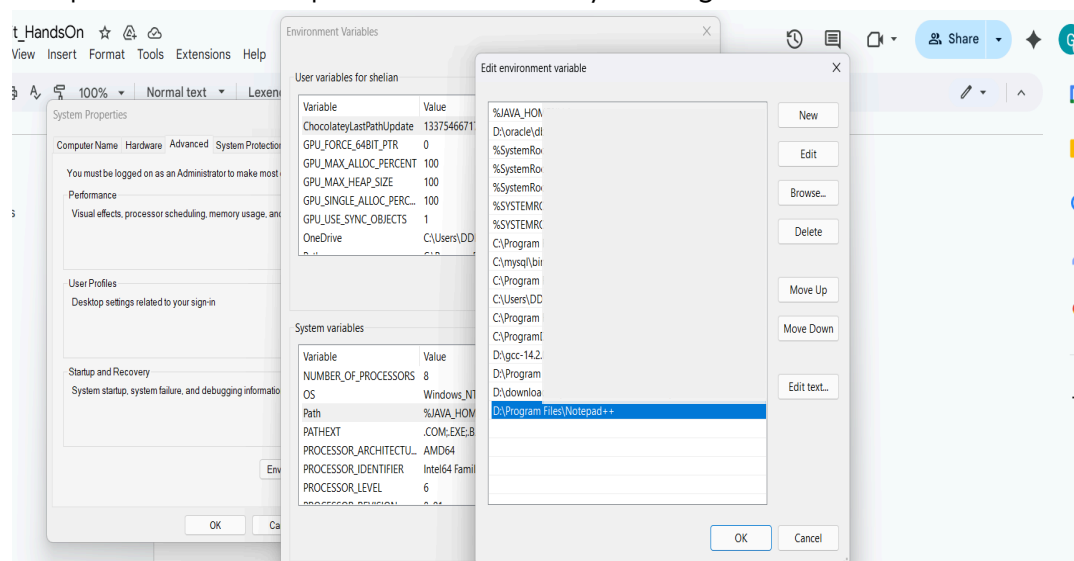
shelian@DESKTOP-HH52HTM MINGW32 ~ (master)
$ git config --global user.email "lianis.ee7@gmail.com"
```

3. Verify the configuration is properly set

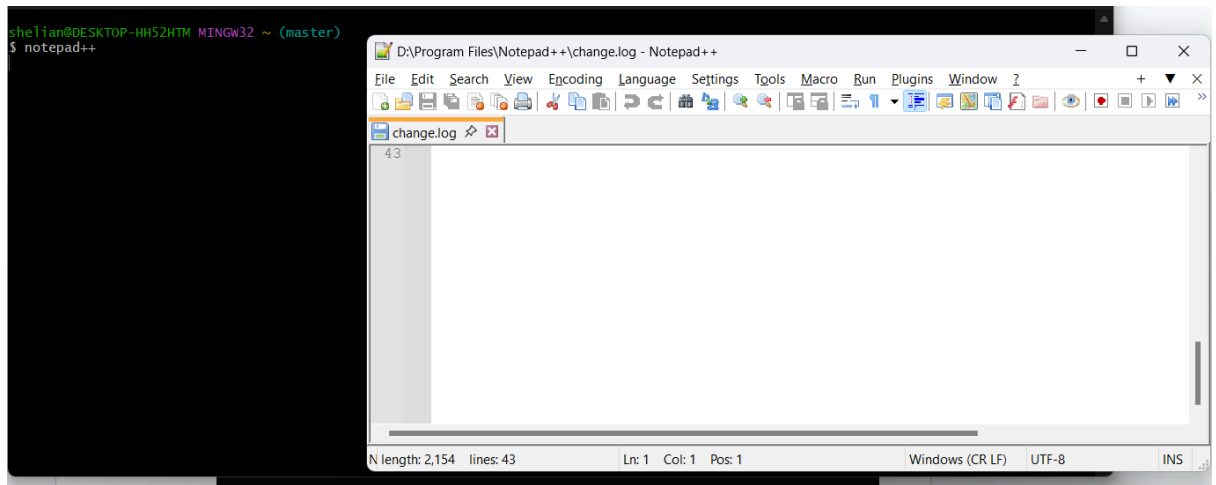
```
git config --global --list
```

```
shelian@DESKTOP-HH52HTM MINGW32 ~ (master)
$ git config --global --list
user.name=shelian
user.email=lianis.ee7@gmail.com
```

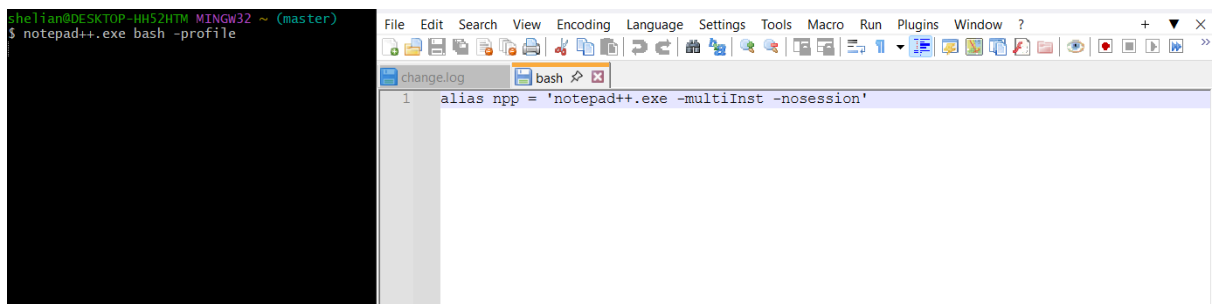
4. Set the path of Notepad++ by: go to control panel -> System -> Advanced System settings. Go to Advanced tab -> Environment variables -> Add path of notepad++.exe to the path user variable by clicking on "Edit"



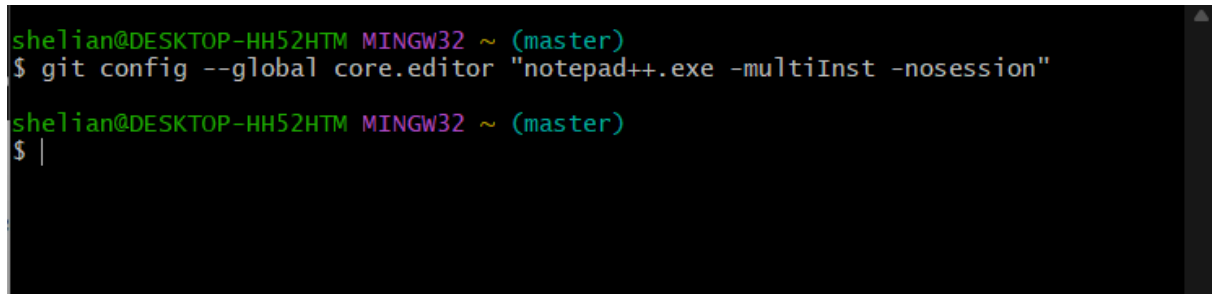
5. Integrate notepad++.exe to Git and make it as default editor. Start by running **notepad++**



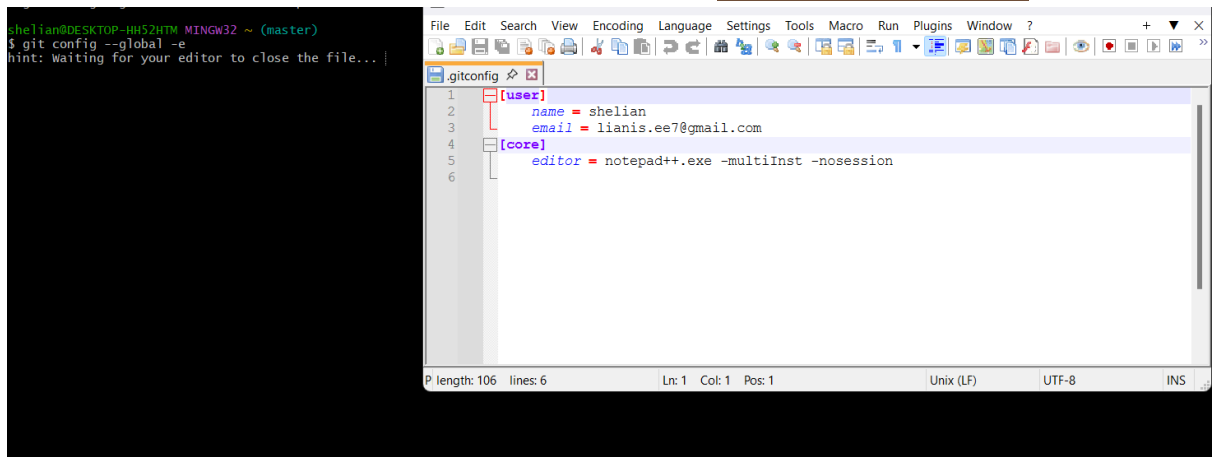
6. Create an alias command for **notepad++.exe** by running **notepad++.exe bash -profile** and adding the line : **alias npp = 'notepad++.exe -multiInst -nosession'**



7. Configure the editor by running **git config --global core.editor "notepad++.exe -multiInst -nosession"**



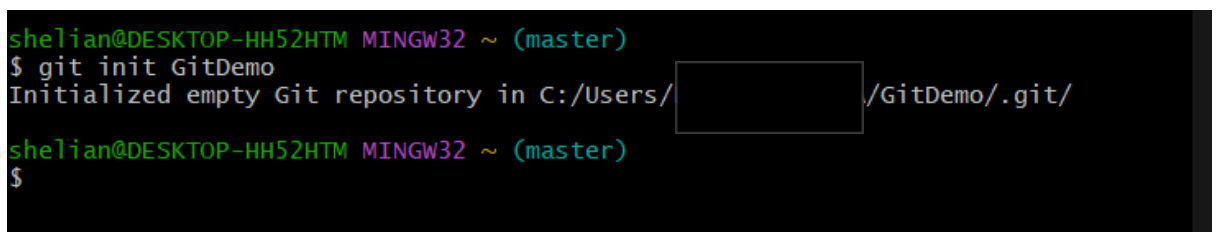
8. Verify if Notepad++ is the default editor by running `git config --global -e`



```
shelian@DESKTOP-HH52HTM MINGW32 ~ (master)
$ git config --global -e
hint: Waiting for your editor to close the file...

.gitconfig
1 [user]
2   name = shelian
3   email = lianis.ee7@gmail.com
4 [core]
5   editor = notepad++.exe -multiInst -nosession
6
```

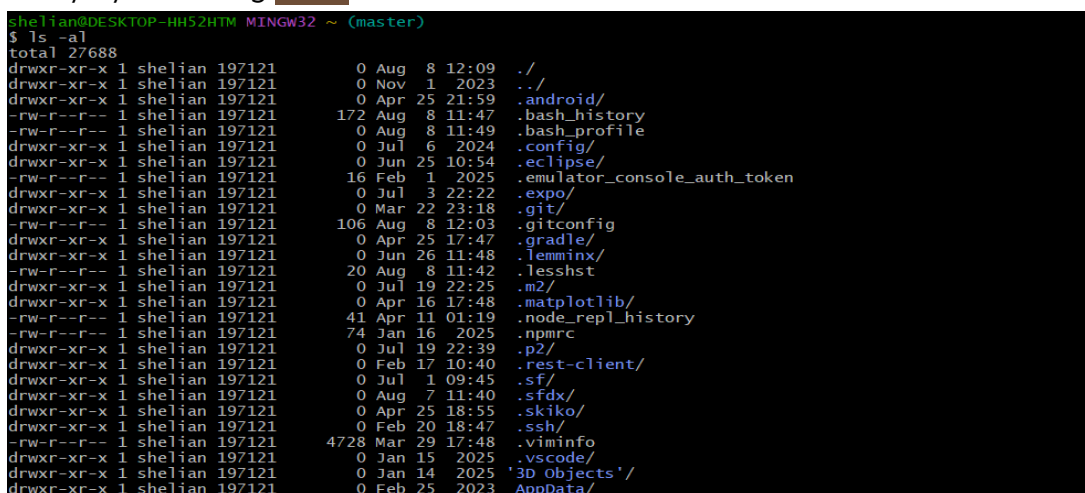
9. Add a file to source code repository. Create a new project called “GitDemo” as follows



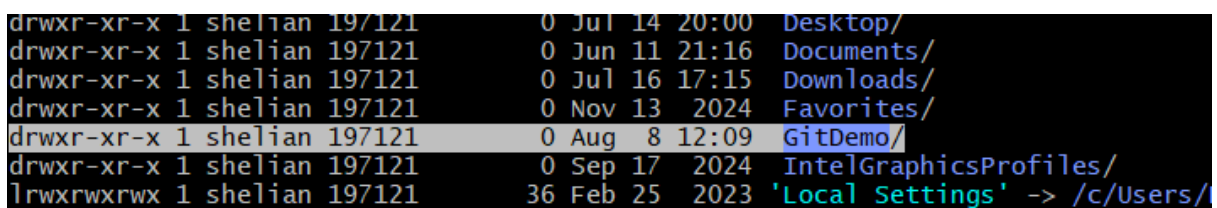
```
shelian@DESKTOP-HH52HTM MINGW32 ~ (master)
$ git init GitDemo
Initialized empty Git repository in C:/Users/shelian/Desktop/GitDemo/.git/

shelian@DESKTOP-HH52HTM MINGW32 ~ (master)
$
```

10. Verify by executing `ls -al`



```
shelian@DESKTOP-HH52HTM MINGW32 ~ (master)
$ ls -al
total 27688
drwxr-xr-x 1 shelian 197121 0 Aug 8 12:09 ./
drwxr-xr-x 1 shelian 197121 0 Nov 1 2023 ../
drwxr-xr-x 1 shelian 197121 0 Apr 25 21:59 .android/
-rw-r--r-- 1 shelian 197121 172 Aug 8 11:47 .bash_history
-rw-r--r-- 1 shelian 197121 0 Aug 8 11:49 .bash_profile
drwxr-xr-x 1 shelian 197121 0 Jul 6 2024 .config/
drwxr-xr-x 1 shelian 197121 0 Jun 25 10:54 .eclipse/
-rw-r--r-- 1 shelian 197121 16 Feb 1 2025 .emulator_console_auth_token
drwxr-xr-x 1 shelian 197121 0 Jul 3 22:22 .expo/
drwxr-xr-x 1 shelian 197121 0 Mar 22 23:18 .git/
-rw-r--r-- 1 shelian 197121 106 Aug 8 12:03 .gitconfig
drwxr-xr-x 1 shelian 197121 0 Apr 25 17:47 .gradle/
drwxr-xr-x 1 shelian 197121 0 Jun 26 11:48 .lemminx/
-rw-r--r-- 1 shelian 197121 20 Aug 8 11:42 .lessht
drwxr-xr-x 1 shelian 197121 0 Jul 19 22:25 .m2/
drwxr-xr-x 1 shelian 197121 0 Apr 16 17:48 .matplotlib/
-rw-r--r-- 1 shelian 197121 41 Apr 11 01:19 .node_repl_history
-rw-r--r-- 1 shelian 197121 74 Jan 16 2025 .npmrc
drwxr-xr-x 1 shelian 197121 0 Jul 19 22:39 .p2/
drwxr-xr-x 1 shelian 197121 0 Feb 17 10:40 .rest-client/
drwxr-xr-x 1 shelian 197121 0 Jul 1 09:45 .sf/
drwxr-xr-x 1 shelian 197121 0 Aug 7 11:40 .sfdx/
drwxr-xr-x 1 shelian 197121 0 Apr 25 18:55 .skiko/
drwxr-xr-x 1 shelian 197121 0 Feb 20 18:47 .ssh/
-rw-r--r-- 1 shelian 197121 4728 Mar 29 17:48 .viminfo
drwxr-xr-x 1 shelian 197121 0 Jan 15 2025 .vscode/
drwxr-xr-x 1 shelian 197121 0 Jan 14 2025 '3D Objects'/
drwxr-xr-x 1 shelian 197121 0 Feb 25 2023 AppData/
```



```
drwxr-xr-x 1 shelian 197121 0 Jul 14 20:00 Desktop/
drwxr-xr-x 1 shelian 197121 0 Jun 11 21:16 Documents/
drwxr-xr-x 1 shelian 197121 0 Jul 16 17:15 Downloads/
drwxr-xr-x 1 shelian 197121 0 Nov 13 2024 Favorites/
drwxr-xr-x 1 shelian 197121 0 Aug 8 12:09 GitDemo/
drwxr-xr-x 1 shelian 197121 0 Sep 17 2024 IntelGraphicsProfiles/
lrwxrwxrwx 1 shelian 197121 36 Feb 25 2023 'Local Settings' -> /c/Users/shelian/
```


11. Create a "welcome.txt" file and add some content

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$ echo "Welcome to version control! It's makes work less stressful!" >> welcome.txt

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$ |
```

12. Verify the creation of welcome.txt by ls -al

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$ ls -al
total 21
drwxr-xr-x 1 sheliam 197121  0 Aug  8 12:18 ./
drwxr-xr-x 1 sheliam 197121  0 Aug  8 12:16 ../
drwxr-xr-x 1 sheliam 197121  0 Aug  8 12:09 .git/
-rw-r--r-- 1 sheliam 197121 60 Aug  8 12:18 welcome.txt

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$
```

13. Verify whether content was added to welcome.txt by cat welcome.txt

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$ cat welcome.txt
Welcome to version control! It's makes work less stressful!

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$
```

14. Make the file to be tracked by Git repository by git add welcome.txt

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$ git add welcome.txt
warning: in the working copy of 'welcome.txt', LF will be replaced by CRLF the next time Git touches it

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$
```

15. Add a multi line comment by **git commit**

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$ git commit
hint: Waiting for your editor to close the file...

COMMIT_EDITMSG
1
2 # Please enter the commit message for your changes. Lines starting
3 # with '#' will be ignored, and an empty message aborts the commit.
4 #
5 # On branch master
6 #
7 # Initial commit
8 #
9 # Changes to be committed:
10 #   new file:   welcome.txt
11 #
12 Created a new file called welcome.txt

N length: 270  lines: 12  Ln: 12  Col: 38  Pos: 271  Unix (LF)  UTF-8  INS

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$ git commit
[master (root-commit) e63162b] Created a new file called welcome.txt
1 file changed, 1 insertion(+)
create mode 100644 welcome.txt

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$
```

16. Execute **git status** to check if local and working directory git repository are same

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$ git status
On branch master
nothing to commit, working tree clean

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (master)
$
```

17. Set up the gitlab account and repo. Add the remote origin

Your work / Projects / New project / Create blank project

Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name

GitDemo

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL

https://gitlab.com/ shelian-group

Project slug

gitdemo

Project deployment target (optional)

Select the deployment target

Visibility Level

☒ Private
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

☐ Internal
The project can be accessed by any logged in user except external users.

☐ Public
The project can be accessed without any authentication.

Project Configuration

☐ Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. [Learn more.](#)

☐ Enable Secret Detection
Scan your code for secrets and credentials to prevent unauthorized access. [Learn more.](#)

[Experimental settings](#)

[Create project](#) [Cancel](#)

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git remote add origin git@gitlab.com:shelian-group/gitdemo.git

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git push --set-upstream origin --all
The authenticity of host 'gitlab.com (2606:4700:90:0:f22e:fbec:5bed:a9b9)' can't
be established.
ED25519 key fingerprint is SHA256:eUXGGm1YGsMAS7vkcx6JOJdOGHPem5gQp4taiCfCLB8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'gitlab.com' (ED25519) to the list of known hosts.
git@gitlab.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

18. Generate ssh key for GitLab pushing and puling purpose

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ ssh-keygen -t rsa -b 4096 -C "lianis.ee7@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/DDKVIJAYAWADA/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/DDKVIJAYAWADA/.ssh/id_rsa
Your public key has been saved in /c/Users/DDKVIJAYAWADA/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:hkUnCfns8a2eHJPWZ3ufy/LOKGGK/IkcgVnGZuYJGS1g lianis.ee7@gmail.com
The key's randomart image is:
+---[RSA 4096]---+
|  oEo+o+         |
| . +.o++         |
| o * .           |
| *o* .           |
| o.oS= .         |
| .. oo.          |
| . =.. o.        |
| .o*+. +oo+      |
| =++oo=*+        |
+---[SHA256]-----+
```

19. Push the local repo to remote

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git push --set-upstream origin --all
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 288 bytes | 96.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To gitlab.com:shelian-group/gitdemo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

20. Pull from the remote repo

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git pull origin main
From gitlab.com:shelian-group/gitdemo
 * branch            main      -> FETCH_HEAD
Already up to date.

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

EXERCISE - 2

Objectives

- Explain git ignore
- Explain how to ignore unwanted files using git ignore

In this hands-on lab, you will learn how to:

- Implement git ignore command to ignore unwanted files and folders

Prerequisites

The following are the pre-requisites to complete this hands-on lab:

- Setting up Git environment
- Integrate notepad++ as a default editor
- A Git repository in the local system and a remote repository in GitLab

Notes*:

Please follow the below steps for creating a free account in GitHub.

Do not use cognizant credentials to login to GitHub.

Estimated time to complete this lab: **20 minutes.**

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.

SOLUTION

1. Create `logging.log` file in the working directory

```
sheliam@DESKTOP-HH52HTM MINGW32 ~ (master)
$ cd GitDemo

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ touch logging.log

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ ls
logging.log  welcome.txt

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

2. Create `logs` folder in the working directory

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ mkdir logs

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ ls
logging.log  logs/  welcome.txt

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ cd logs

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo/logs (main)
$ touch logger.log

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo/logs (main)
$ ls
logger.log

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo/logs (main)
$ |
```

3. Append some content to the `logging.log` file

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo/logs (main)
$ cd ..

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ echo "Hey I am the logging.log file inside the GitDemo folder" > logging.log

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ cat logging.log
Hey I am the logging.log file inside the GitDemo folder

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

4. Append some content to the `logs/logger.log` file

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo/logs (main)
$ echo "I am the log file called logger.log inside the logs folder which is inside the GitDemo folder" > logger.log

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo/logs (main)
$ cat logger.log
I am the log file called logger.log inside the logs folder which is inside the GitDemo folder

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo/logs (main)
$ |
```

5. Check to see if `.gitignore` exists

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ ls -al
total 22
drwxr-xr-x 1 shelian 197121 0 Aug 9 12:15 ./
drwxr-xr-x 1 shelian 197121 0 Aug 8 12:16 ../
drwxr-xr-x 1 shelian 197121 0 Aug 8 22:06 .git/
-rw-r--r-- 1 shelian 197121 56 Aug 9 12:20 logging.log
drwxr-xr-x 1 shelian 197121 0 Aug 9 12:16 logs/
-rw-r--r-- 1 shelian 197121 60 Aug 8 12:18 welcome.txt

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

6. Create a `.gitignore` file and verify

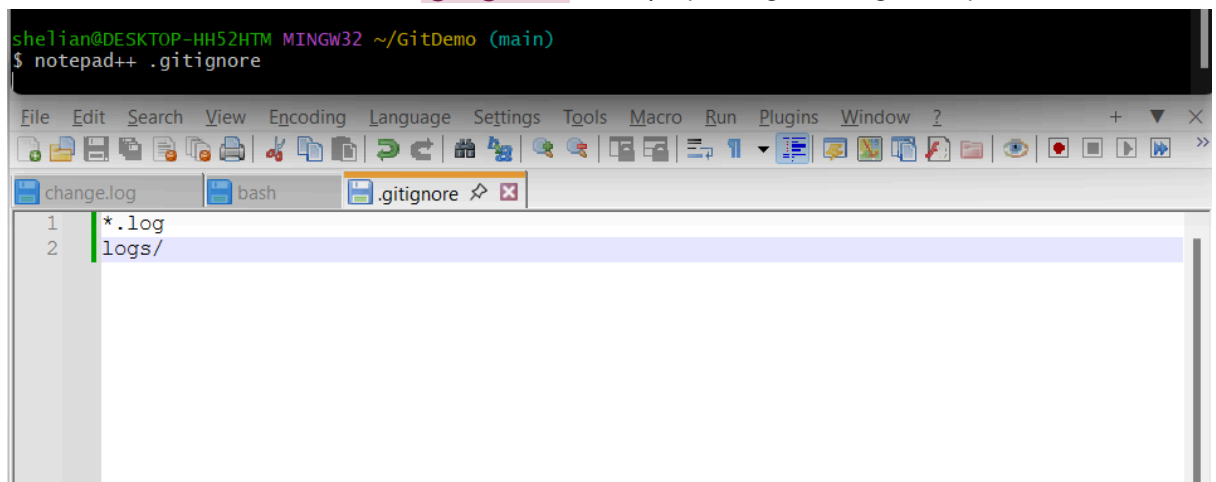
```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ touch .gitignore

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ ls
logging.log  logs/  welcome.txt

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ ls -al
total 26
drwxr-xr-x 1 shelian 197121 0 Aug 9 12:23 ./
drwxr-xr-x 1 shelian 197121 0 Aug 8 12:16 ../
drwxr-xr-x 1 shelian 197121 0 Aug 8 22:06 .git/
-rw-r--r-- 1 shelian 197121 0 Aug 9 12:23 .gitignore
-rw-r--r-- 1 shelian 197121 56 Aug 9 12:20 logging.log
drwxr-xr-x 1 shelian 197121 0 Aug 9 12:16 logs/
-rw-r--r-- 1 shelian 197121 60 Aug 8 12:18 welcome.txt

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

7. Add these above files into the `.gitignore` file by opening it using Notepad++



The screenshot shows the Notepad++ application window with the `.gitignore` file open. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and Help. The toolbar contains various icons for file operations. The file explorer shows `change.log`, `bash`, and `.gitignore`. The text area contains the following content:

```
1 *.log
2 logs/
```

8. Check the current status by `git status`

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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)

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

9. Stage the `.gitignore` file

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git add .

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```


10. Commit the changes with a meaningful commit message and check the status

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git commit -m "Added the log file and logs folder into the .gitignore file"
[main 16e4d46] Added the log file and logs folder into the .gitignore file
1 file changed, 2 insertions(+)
create mode 100644 .gitignore

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

11. Push the changes and verify the status

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 314 bytes | 157.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To gitlab.com:shelian-group/gitdemo.git
   e63162b..16e4d46  main -> main

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

EXERCISE - 3

Objectives

- Explain branching and merging
- Explain about creating a branch request in GitLab
- Explain about creating a merge request in GitLab

In this hands-on lab, you will learn how to:

- Construct a branch, do some changes in the branch, and merge it with master (or trunk)

Prerequisites

The following are the pre-requisites to complete this hands-on lab:

- Setting up Git environment with P4Merge tool for Windows

Notes*:

Please follow the below steps for creating a free account in GitHub.

Do not use cognizant credentials to login to GitHub.

Estimated time to complete this lab: **30 minutes**.

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

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.

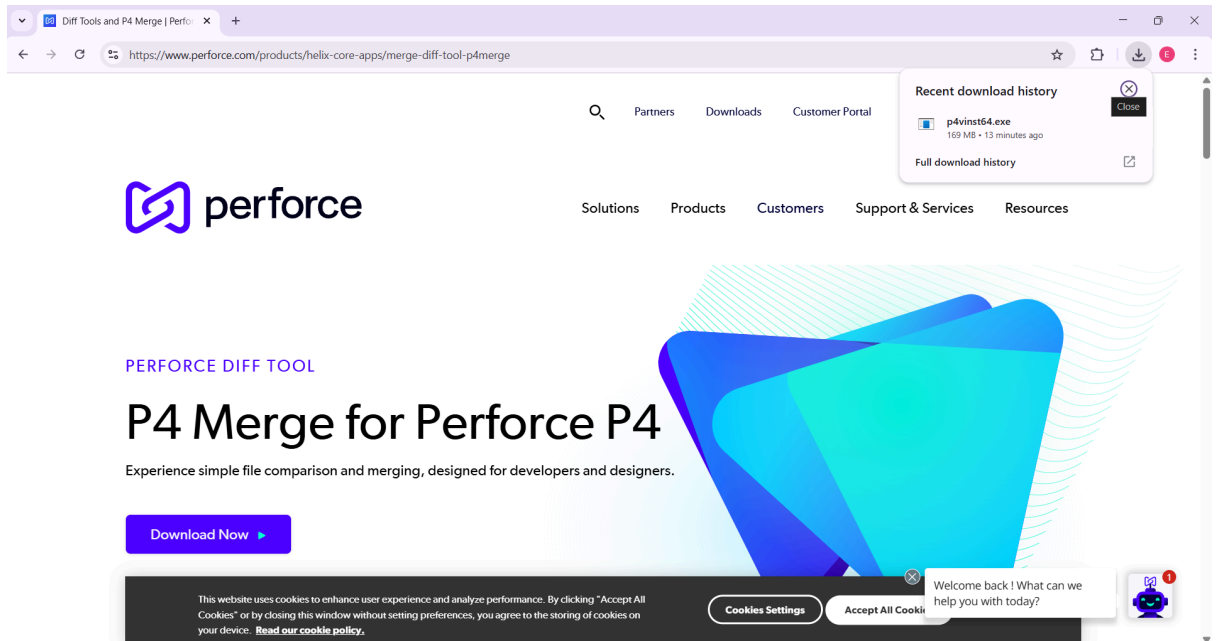
Merging:

1. Switch to the master

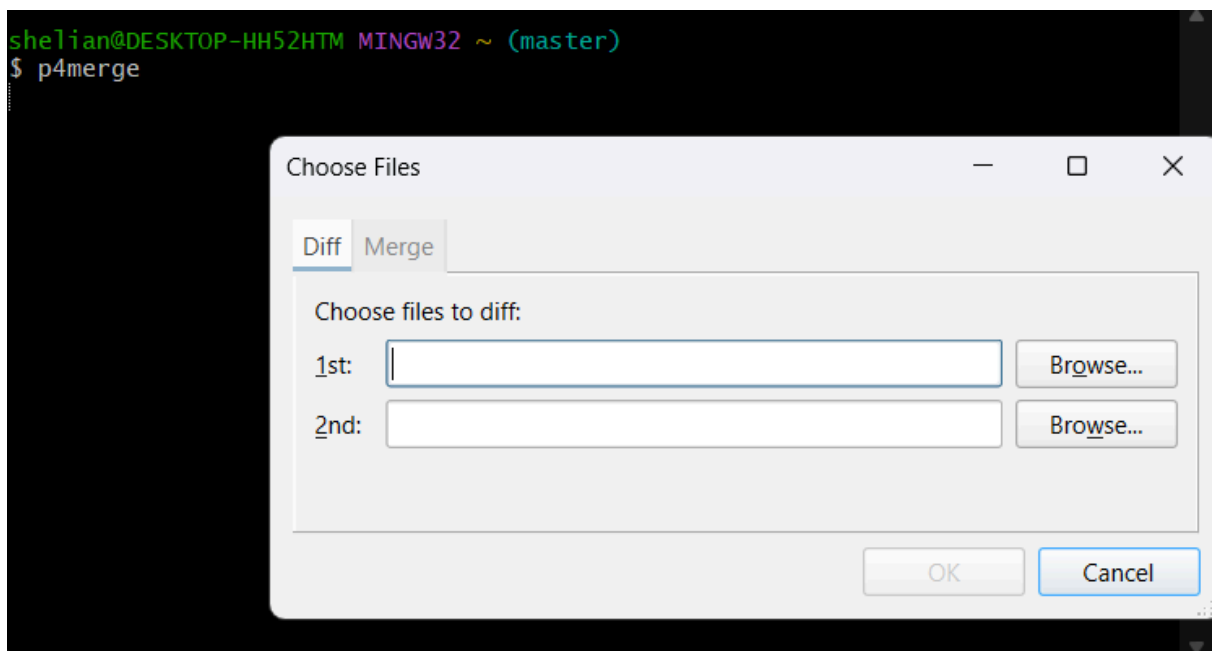
2. List out all the differences between trunk and branch. These provide the differences in command line interface.
3. List out all the visual differences between master and branch using **P4Merge tool**.
4. Merge the source branch to the trunk.
5. Observe the logging after merging using “**git log -oneline -graph -decorate**”
6. Delete the branch after merging with the trunk and observe the git status.

SOLUTION

1. Install the P4Merge tool



2. Verify installation



3. Create a new branch called `GitNewBranch` by using the command `git branch <branchName>`

```
shelian@DESKTOP-HH52HTM MINGW32 ~ (master)
$ cd GitDemo

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git branch GitNewBranch
```

4. List all local and remote branches available in the trunk. The * denotes the current pointing branch which is `main` in our case

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git branch -a
  GitNewBranch
* main
  remotes/old-origin/main
  remotes/origin/main

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

5. Switch to the newly created branch by using `git checkout <branchname>` and list the branches to verify we have switched

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git checkout GitNewBranch
Switched to branch 'GitNewBranch'

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$ git branch -a
* GitNewBranch
  main
  remotes/old-origin/main
  remotes/origin/main

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$ |
```

6. Add a file in this new branch

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$ echo "Hello world of branches!" > welcomeBranches.txt

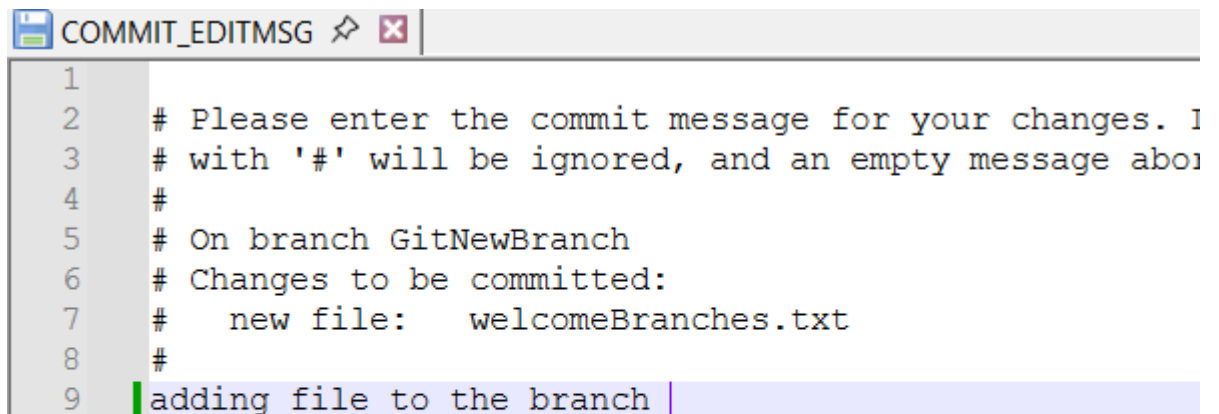
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$ ls
logging.log  logs/  welcome.txt  welcomeBranches.txt

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$
```

7. Stage and Commit the changes to this branch

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$ git add welcomeBranches.txt
warning: in the working copy of 'welcomeBranches.txt', LF will be replaced by CR
LF the next time Git touches it

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$ git commit
hint: Waiting for your editor to close the file... |
```



```
COMMIT_EDITMSG
1
2 # Please enter the commit message for your changes. Lines
3 # starting with '#' will be ignored, and an empty message aborts the commit.
4 #
5 # On branch GitNewBranch
6 # Changes to be committed:
7 #   new file:   welcomeBranches.txt
8 #
9 adding file to the branch
```

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$ git commit
[GitNewBranch f8711cb] adding file to the branch
1 file changed, 1 insertion(+)
create mode 100644 welcomeBranches.txt

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$
```

8. Check the status by `git status`

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$ git status
On branch GitNewBranch
nothing to commit, working tree clean

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$ |
```

9. Now, we will merge the branches. First, switch to the `main` branch

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitNewBranch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git branch -a
  GitNewBranch
* main
  remotes/old-origin/main
  remotes/origin/main

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

10. See the differences in CLI by `git diff <branch1> <branch2>`

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git diff main GitNewBranch
diff --git a/welcomeBranches.txt b/welcomeBranches.txt
new file mode 100644
index 0000000..3e744c3
--- /dev/null
+++ b/welcomeBranches.txt
@@ -0,0 +1 @@
+Hello world of branches!

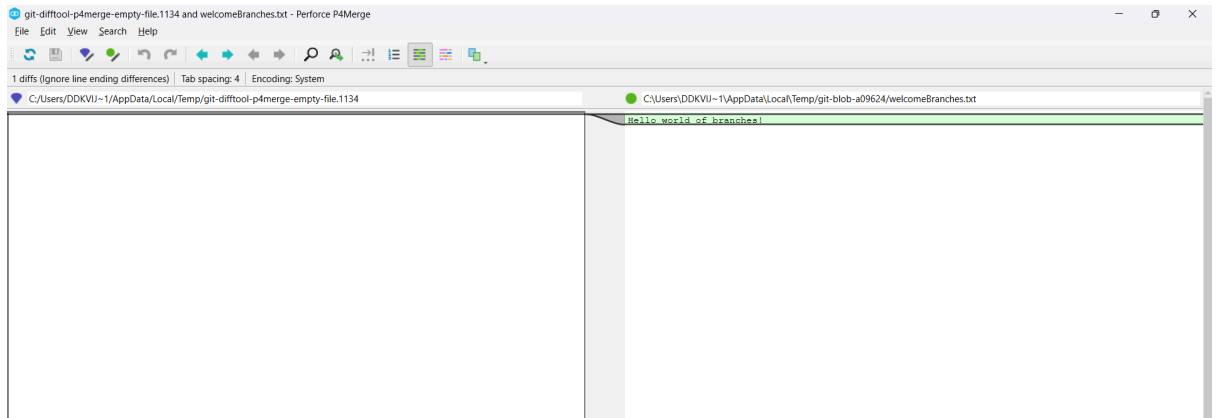
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

11. See the visual differences by using the P4merge tool. First, configure the P4 Merge tool and then open the tool by `git difftool <branch1> <branch2>`

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git config --global merge.tool p4merge

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git difftool main GitNewBranch

Viewing (1/1): 'welcomeBranches.txt'
Launch 'p4merge' [Y/n]? y
```



12. Merge the source branch to the trunk by `git merge <branchToMerge>`

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git merge GitNewBranch
Updating 16e4d46..f8711cb
Fast-forward
 welcomeBranches.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 welcomeBranches.txt

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

13. Observe the logging by `git log --oneline --graph --decorate`

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git log --oneline --graph --decorate
* f8711cb (HEAD -> main, GitNewBranch) adding file to the branch
* 16e4d46 (origin/main) Added the log file and logs folder into the .gitignore f
ile
* e63162b Created a new file called welcome.txt

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```


14. Delete the branch after merging

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git branch -d GitNewBranch
Deleted branch GitNewBranch (was f8711cb).

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

15. Observe status by `git status`

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

16. Push to the remote repo

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 344 bytes | 344.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To gitlab.com:shelian-group/gitdemo.git
   16e4d46..f8711cb  main -> main



shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.


nothing to commit, working tree clean

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

17. A snapshot of the remote repo

shelian-group / **GitDemo**

 **GitDemo**  Free

 main ▾

 gitdemo


+

 ▾


Find file

Code ▾




⋮

 **adding file to the branch**
E Shelian Gladis authored 21 minutes ago

f8711cbd



History

Name	Last commit	Last update
 .gitignore	Added the log file and logs folder into t...	1 hour ago
 welcome.txt	Created a new file called welcome.txt	1 day ago
 welcomeBranches.txt	adding file to the branch	21 minutes ago

EXERCISE - 4

Objectives

- Explain how to resolve the conflict during merge.

In this hands-on lab, you will learn how to:

- 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.

Prerequisites

The following are the pre-requisites to complete this hands-on lab:

- Hands-on ID: **"Git-T03-HOL_001"**

Notes*:

Please follow the below steps for creating a free account in GitHub.

Do not use cognizant credentials to login to GitHub.

Estimated time to complete this lab: **30 minutes**.

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.
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”

SOLUTION

1. Verify main is in clean state by `git status`

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

2. Create a new branch `GitWork` and verify

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git branch GitWork

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git branch -a
GitWork
* main
remotes/old-origin/main
remotes/origin/main
```

3. Move into that branch and verify

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git switch GitWork
Switched to branch 'GitWork'

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ git branch -a
* GitWork
main
remotes/old-origin/main
remotes/origin/main

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$
```

4. Create a new file `hello.xml` in this branch

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ touch hello.xml

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ ls
hello.xml  logging.log  logs/  welcome.txt  welcomeBranches.txt

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ |
```

5. Add some content into this file

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ echo "<message>Hello from GitWork\!</message>" > hello.xml

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ cat hello.xml
<message>Hello from GitWork\!</message>

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ |
```

6. Check the status of this branch

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ git status
On branch GitWork
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    hello.xml

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

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ |
```

7. Commit the changes in `GitWork`

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ git add hello.xml
warning: in the working copy of 'hello.xml', LF will be replaced by CRLF the next time Git touches it

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ git commit -m "added hello.xml to GitWork branch"
[GitWork 7b27aee] added hello.xml to GitWork branch
1 file changed, 1 insertion(+)
create mode 100644 hello.xml

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$
```

8. Switch to **main** branch

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (GitWork)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

9. Add **hello.xml** to **main** branch and some different content to the file

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ echo "<message>Hello from main </message>" > hello.xml

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ cat hello.xml
<message>Hello from main </message>

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ ls
hello.xml  logging.log  logs/  welcome.txt  welcomeBranches.txt

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

10. Commit the changes in **main** branch

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

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

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git add hello.xml
warning: in the working copy of 'hello.xml', LF will be replaced by CRLF the next time Git touches it

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git commit -m "added content in hello.xml from main"
[main 3315529] added content in hello.xml from main
1 file changed, 1 insertion(+)
create mode 100644 hello.xml

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

11. Observe the log by executing `git log --oneline --graph --decorate --all`

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git log --oneline --graph --decorate --all
* 3315529 (HEAD -> main) added content in hello.xml from main
| * 7b27aee (GitWork) added hello.xml to GitWork branch
|/
* f8711cb (origin/main) adding file to the branch
* 16e4d46 Added the log file and logs folder into the .gitignore file
* e63162b Created a new file called welcome.txt
* e8d98ee (old-origin/main) Initial commit

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

12. Check difference between the branches by `git diff <b1> <b2>`

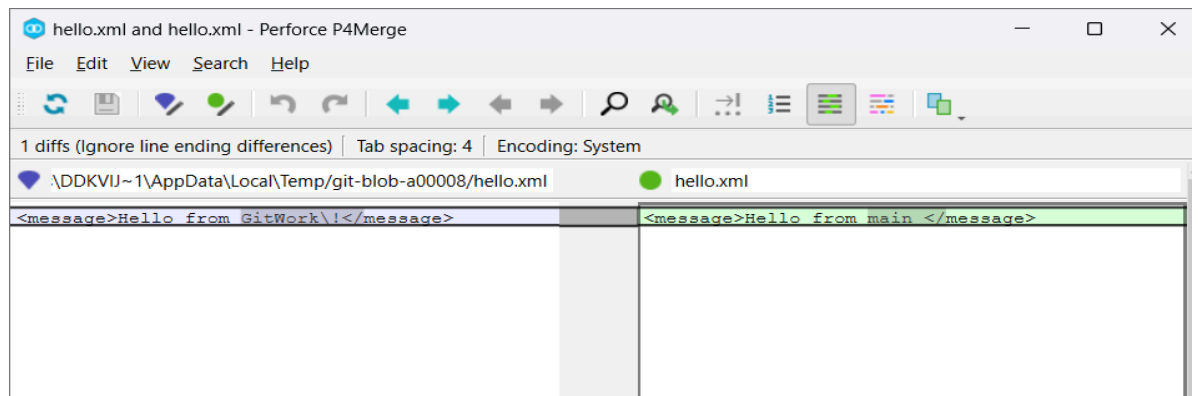
```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git diff main GitWork
diff --git a/hello.xml b/hello.xml
index ec4ed93..713dd7b 100644
--- a/hello.xml
+++ b/hello.xml
@@ -1,1 @@
- <message>Hello from main </message>
+ <message>Hello from GitWork\!</message>

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

13. Use visualisation tool P4 Merge `git difftool <b2>`

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git difftool GitWork

Viewing (1/1): 'hello.xml'
Launch 'p4merge' [Y/n]? y
```



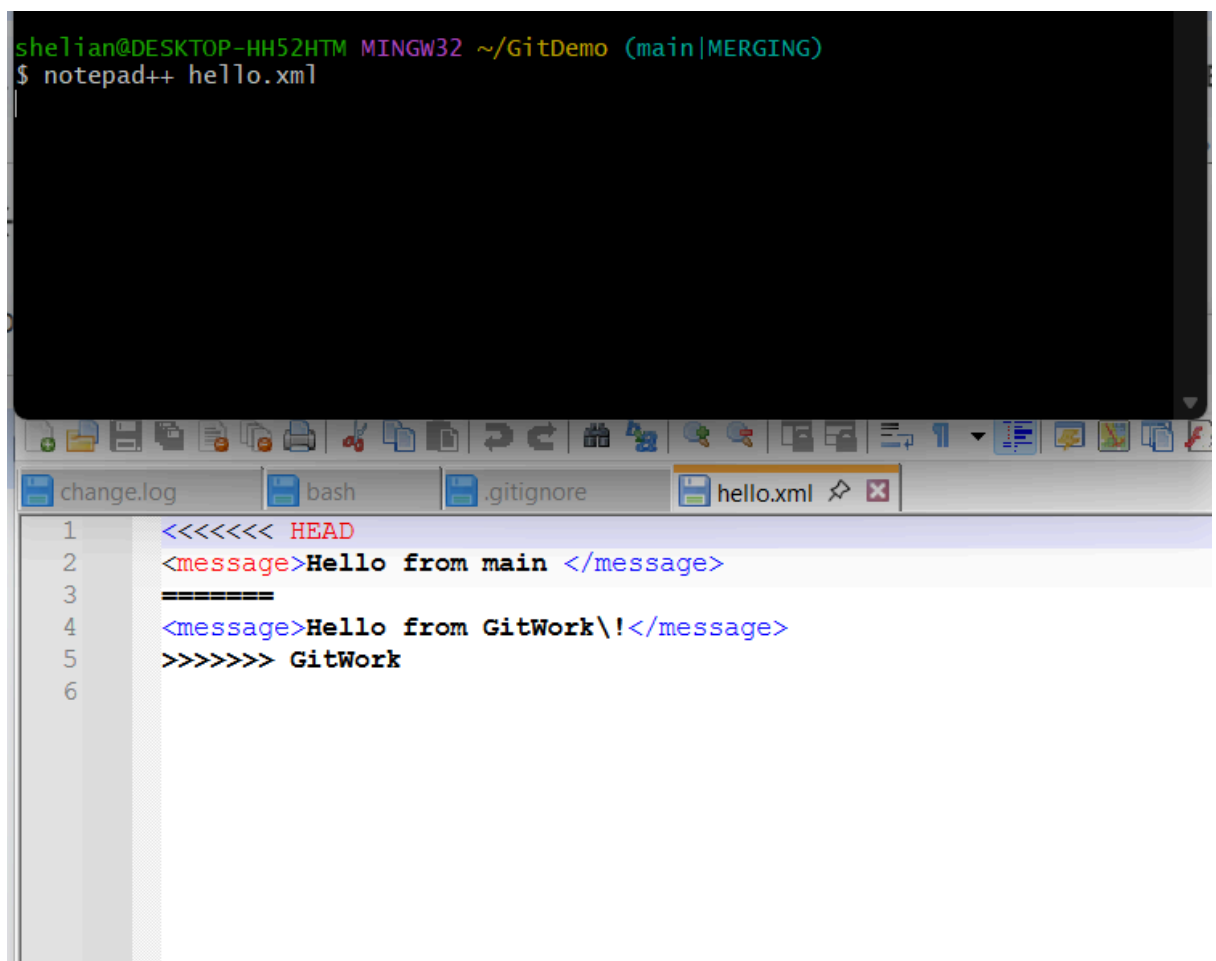
14. Merge **GitWork** with **main**

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ 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.

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main|MERGING)
$
```

15. Observe the conflict markup by opening the **hello.xml** file

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main|MERGING)
$ notepad++ hello.xml
```



The screenshot shows the Notepad++ editor with the file `hello.xml` open. The editor displays conflict markup for the `hello.xml` file. The markup is as follows:

```
1 <<<<<< HEAD
2 <message>Hello from main </message>
3 =====
4 <message>Hello from GitWork\!</message>
5 >>>>>> GitWork
6
```

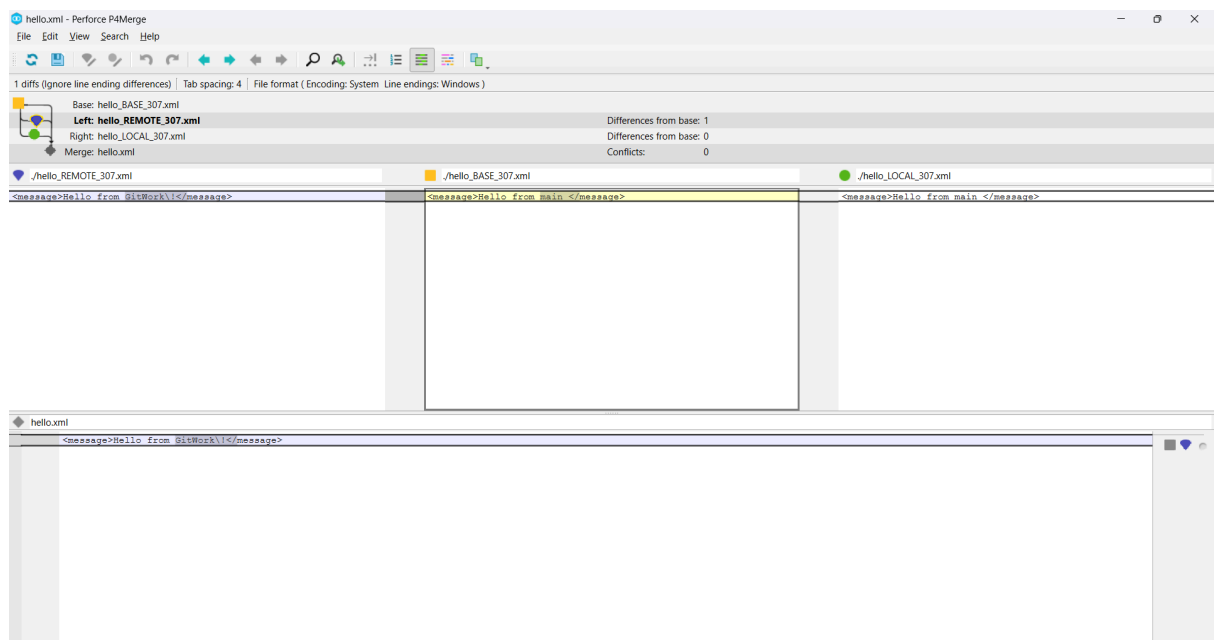
- **<<<<<< HEAD**
Everything below this line (until the =====) is the version from the branch we currently checked out (in this case, main).
- **<message>Hello from main </message>**
This is the content in main.

- =====
This separates the two conflicting changes.
- <message>Hello from GitWork\!</message>
This is the version from the branch tried to merge into main (here: GitWork).
- >>>>>> GitWork
Marks the end of the conflict block, showing the name of the branch that caused the conflict.

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

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main|MERGING)
$ git mergetool
Merging:
hello.xml

Normal merge conflict for 'hello.xml':
  {local}: created file
  {remote}: created file
error: invalid path './hello_BASE_307.xml'
```



- Make the merge/desired changes in the bottom window

17. After resolving, opening the tool again gives

```
shel'ian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main|MERGING)
$ git mergetool
No files need merging

shel'ian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main|MERGING)
$
```

18. Now, commit the file after resolving the conflict

```
shel'ian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main|MERGING)
$ git add hello.xml

shel'ian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main|MERGING)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
  modified:   hello.xml

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

shel'ian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main|MERGING)
$ git commit -m "resolved merge conflict"
[main 4349c9b] resolved merge conflict
```

19. Git mergetool may create a `.orig` backup file which we will add to `.gitignore`

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
(use "git push" to publish your local commits)

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

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

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ echo "*.orig" >> .gitignore

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
(use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore

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

no changes added to commit (use "git add" and/or "git commit -a")

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

20. Commit the `.gitignore` file

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git add .gitignore
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git commit -m "ignore merge backup files"
[main 76c8bfe] ignore merge backup files
1 file changed, 1 insertion(+), 1 deletion(-)

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

21. List out all branches

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git branch -a
GitWork
* main
  remotes/old-origin/main
  remotes/origin/main

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

22. Push to the remote repository

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.23 KiB | 315.00 KiB/s, done.
Total 12 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
To gitlab.com:shelian-group/gitdemo.git
   f8711cb..76c8bfe  main -> main

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

23. A snapshot of remote repository

The screenshot displays the GitLab web interface for a repository named 'GitDemo' under the 'shelian-group'. The interface is split into two sections. The top section shows the root of the 'gitdemo' branch, with a dropdown menu set to 'main' and a button to switch to 'gitdemo'. Below this, a commit summary for 'ignore merge backup files' by 'E Shelian Gladis' is shown, along with a table of files. The bottom section shows the 'hello.xml' file, which has a commit summary for 'resolved merge conflict' by 'E Shelian Gladis'. The file content is displayed as XML code.

shelian-group / GitDemo

main gitdemo

gitdemo

ignore merge backup files
E Shelian Gladis authored 3 minutes ago

Name	Last commit	Last update
.gitignore	ignore merge backup files	3 minutes ago
hello.xml	resolved merge conflict	7 minutes ago
welcome.txt	Created a new file called welcome.txt	1 day ago
welcomeBranches.txt	adding file to the branch	2 hours ago

shelian-group / GitDemo

main gitdemo / hello.xml

hello.xml

resolved merge conflict
E Shelian Gladis authored 7 minutes ago

hello.xml 75 B

```
1 <message>Hello from GitWork\!</message>
2 <message>Hello from main</message>
3
```

24. Delete the `GitWork` branch

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git branch -d GitWork
Deleted branch GitWork (was 7b27aee).

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

25. View the logs by `git log --oneline --graph --decorate`

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git log --oneline --graph --decorate
* 76c8bfe (HEAD -> main, origin/main) ignore merge backup files
*   4349c9b resolved merge conflict
| \
|  * 7b27aee added hello.xml to GitWork branch
* | 3315529 added content in hello.xml from main
| /
* f8711cb adding file to the branch
* 16e4d46 Added the log file and logs folder into the .gitignore file
* e63162b Created a new file called welcome.txt

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

EXERCISE - 5

Objectives

- Explain how to clean up and push back to remote Git

In this hands-on lab, you will learn how to:

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

Prerequisites

The following are the pre-requisites to complete this hands-on lab:

- Hands-on ID: **“Git-T03-HOL_002”**

Notes*:

Please follow the below steps for creating a free account in GitHub.

Do not use cognizant credentials to login to GitHub.

Estimated time to complete this lab: **10 minutes**.

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

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.

SOLUTION

1. Verify main is in clean state by `git status`

```
sheliam@DESKTOP-HH52HTM MINGW32 ~ (master)
$ cd GitDemo

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

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

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

2. Stage and Commit the changes

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git add .

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git commit -m "committing all files"
[main 35b0206] committing all files
 1 file changed, 5 insertions(+)
 create mode 100644 hello.xml.orig

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

3. List out all available branches

```
sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git branch -a
* main
  remotes/old-origin/main
  remotes/origin/main

sheliam@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```


4. Pull remote git repo to main

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git pull origin main
From gitlab.com:shelian-group/gitdemo
* branch      main      -> FETCH_HEAD
Already up to date.

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

5. Push pending changes to remote repo

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 331 bytes | 331.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
To gitlab.com:shelian-group/gitdemo.git
   76c8bfe..35b0206  main -> main

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

6. Check the status

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```


7. Noticed error in .gitignore file and corrected it in remote repo


shelian-group / GitDemo / Repository






main ▾ gitdemo

gitdemo

+ ▾ Find file Code ▾ ⋮

 Edit .gitignore
E Shelian Gladis authored 23 seconds ago

979a4400  History


Name	Last commit	Last update
 .gitignore	Edit .gitignore	23 seconds ago
 hello.xml	resolved merge conflict	1 hour ago
 hello.xml.orig	committing all files	4 minutes ago
 welcome.txt	Created a new file called welcome.txt	1 day ago
 welcomeBranches.txt	adding file to the branch	3 hours ago


shelian-group / GitDemo / Repository





main ▾ gitdemo

gitdemo

+ ▾ Find file Code ▾ ⋮

 Delete hello.xml.orig
E Shelian Gladis authored 14 seconds ago

da425304  History

Name	Last commit	Last update
 .gitignore	Edit .gitignore	4 minutes ago
 hello.xml	resolved merge conflict	1 hour ago
 welcome.txt	Created a new file called welcome.txt	1 day ago
 welcomeBranches.txt	adding file to the branch	4 hours ago

8. Pull the changes to local repo

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 280 bytes | 14.00 KiB/s, done.
From gitlab.com:shelian-group/gitdemo
 * branch          main      -> FETCH_HEAD
   35b0206..979a440  main      -> origin/main
Updating 35b0206..979a440
Fast-forward
 .gitignore | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git push origin main
Everything up-to-date

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ |
```

```
shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git pull origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (2/2), 216 bytes | 21.00 KiB/s, done.
From gitlab.com:shelian-group/gitdemo
 * branch          main      -> FETCH_HEAD
   979a440..da42530  main      -> origin/main
Updating 979a440..da42530
Fast-forward
 hello.xml.orig | 5 -----
 1 file changed, 5 deletions(-)
 delete mode 100644 hello.xml.orig

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$ git push origin main
Everything up-to-date

shelian@DESKTOP-HH52HTM MINGW32 ~/GitDemo (main)
$
```

9. Observe if changes are reflected in remote repo

shelian-group / GitDemo / Repository

main

gitdemo

gitdemo

Delete hello.xml.orig
E Shelian Gladis authored 2 minutes ago

da425304

History

Name	Last commit	Last update
.gitignore	Edit .gitignore	7 minutes ago
hello.xml	resolved merge conflict	1 hour ago
welcome.txt	Created a new file called welcome.txt	1 day ago
welcomeBranches.txt	adding file to the branch	4 hours ago

shelian-group / GitDemo / Commits

main

gitdemo

Author

Browse files

Search by message

Aug 09, 2025

Delete hello.xml.orig
E Shelian Gladis authored 3 minutes ago

da425304

Edit .gitignore
E Shelian Gladis authored 7 minutes ago

979a4400

committing all files
E Shelian Gladis authored 11 minutes ago

35b02068

ignore merge backup files
E Shelian Gladis authored 1 hour ago

76c8bfe6

resolved merge conflict
E Shelian Gladis authored 1 hour ago

4349c9bc

added content in hello.xml from main
E Shelian Gladis authored 2 hours ago

33155293

added hello.xml to GitWork branch
E Shelian Gladis authored 2 hours ago

7b27aee9

adding file to the branch
E Shelian Gladis authored 4 hours ago

f8711cbd

Added the log file and logs folder into the .gitignore file
E Shelian Gladis authored 4 hours ago

16e4d46a

Aug 08, 2025

Created a new file called welcome.txt
E Shelian Gladis authored 1 day ago

e63162bd

No results found

Edit your search and try again.

Project

GitDemo

Learn GitLab 15%

Pinned

Issues 0

Merge requests 0

Manage

Plan

Code

Merge requests 0

Repository

Branches

Commits

Tags

Repository graph

Compare revisions

Snippets

Build

Secure

Deploy

Operate

Monitor

Analyze

Settings

What's new 5

Help