**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



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

A black background with white text

AI-generated content may be incorrect.

**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”

A screenshot of a computer

AI-generated content may be incorrect.

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,

A computer screen shot of a computer code

AI-generated content may be incorrect.

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

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



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

A black background with white numbers

AI-generated content may be incorrect.

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



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

A screenshot of a computer

AI-generated content may be incorrect.

1. To verify the content, execute the command



1. Check the status by executing

A black screen with white text

AI-generated content may be incorrect.

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

A black background with white text

AI-generated content may be incorrect.

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

A black background with white text

AI-generated content may be incorrect.

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

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

git pull origin master

1. To push the local to remote repository, execute

git push origin master

**Explanation:**

Installing Git Bash and Notepad++ in the pc then create a folder named as gitdemo for the project then set the environment variables for the notepad++ and add the path and click ok, ok, ok.

The commands that I used in the execution:

1. $ git init

Initialized empty Git repository in C:/Users/hemas/GitDemo/.git/

1. $ echo "Welcome to Git Hands-On Lab!" >welcome.txt
2. $ 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)

1. $ git add welcome.txt

warning: in the working copy of 'welcome.txt', LF will be replaced by CRLF the next time Git touches it

1. $ git commit

[master (root-commit) 7ccd431] Initial commit: Added welcome.txt

1 file changed, 1 insertion(+)

create mode 100644 welcome.txt

1. $ git remote add origin https://github.com/Sree-1405/GitDemo.git
2. $ git remote -v

origin https://github.com/Sree-1405/GitDemo.git (fetch)

origin https://github.com/Sree-1405/GitDemo.git (push)

1. $ git push -u origin master

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Writing objects: 100% (3/3), 267 bytes | 66.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/Sree-1405/GitDemo.git

\* [new branch] master -> master

branch 'master' set up to track 'origin/master'.

In the I have created a local repository in github that is add in the git remote add origin:,

For push and pull the project, it is successfully run in notepad++ also then in the github.

From the notepad++ before creating the local repository in the github:

A screenshot of a computer

AI-generated content may be incorrect.

The file that push into the github to local repository the file name is welcome.txt:

A screenshot of a computer

AI-generated content may be incorrect.

**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.

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

**Explanation:**

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

The commands that I used in the execution:

1. $ echo "This is a test log" >test.log
2. $ mkdir log
3. $ echo "Log folder file" >log/output.txt
4. $ touch .gitignore
5. $ notepad++ .gitignore
6. $ git status

On branch master

Your branch is up to date with 'origin/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)

1. $ git add .gitignore
2. $ git commit -m "Added .gitignore to ignore .log files and log folder"

[master 8dd8805] Added .gitignore to ignore .log files and log folder

1 file changed, 2 insertions(+)

create mode 100644 .gitignore

1. $ git push

Enumerating objects: 4, done.

Counting objects: 100% (4/4), done.

Delta compression using up to 12 threads

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 321 bytes | 160.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/Sree-1405/GitDemo.git

7ccd431..8dd8805 master -> master

The file that push into the github to local repository:

A screenshot of a computer

AI-generated content may be incorrect.

**3)**

**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.

**Explanation:**

Branching:

* + - 1. $ git branch GitNewBranch
      2. $ git branch

GitNewBranch

master

* + - 1. $ git switch GitNewBranch

Switched to branch 'GitNewBranch'

* + - 1. $ echo "This is from GitNewBranch" >branchfile.txt
      2. $ git add branchfile.txt

warning: in the working copy of 'branchfile.txt', LF will be replaced by CRLF the next time Git touches it

1. $ git commit -m "Added branchfile.txt in GitNewBranch"

[GitNewBranch 75fa433] Added branchfile.txt in GitNewBranch

1 file changed, 1 insertion(+)

create mode 100644 branchfile.txt

1. $ git status

On branch GitNewBranch

nothing to commit, working tree clean

Merging:

1. $ git switch master

Switched to branch 'master'

Your branch is up to date with 'origin/master'.

1. $ git diff master GitNewBranch

diff --git a/branchfile.txt b/branchfile.txt

new file mode 100644

index 0000000..bd667bc

--- /dev/null

+++ b/branchfile.txt

@@ -0,0 +1 @@

+This is from GitNewBranch

1. $ git merge GitNewBranch

Updating 8dd8805..75fa433

Fast-forward

branchfile.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 branchfile.txt

4. $ git log --oneline --graph --decorate

\* 75fa433 (HEAD -> master, GitNewBranch) Added branchfile.txt in GitNewBranch

\* 8dd8805 (origin/master) Added .gitignore to ignore .log files and log folder

\* 7ccd431 Initial commit: Added welcome.txt

5. $ git branch -d GitNewBranch

Deleted branch GitNewBranch (was 75fa433).

1. $ git status

On branch master

Your branch is ahead of 'origin/master' by 1 commit.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

git push to push all file to local repository to github.

A screenshot of a computer

AI-generated content may be incorrect.

**4)**

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”**

**Explanation:**

* + - 1. $ git switch master

Already on 'master'

Your branch is up to date with 'origin/master'.

* + - 1. $ git status

On branch master

Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

* + - 1. $ git checkout -b GitWork

Switched to a new branch 'GitWork'

* + - 1. $ echo "<message>Hello from GitWork branch</message>" >hello.xml
      2. $ git add hello.xml

warning: in the working copy of 'hello.xml', LF will be replaced by CRLF the next time Git touches it

* + - 1. $ git commit -m "Added hello.xml in GitWork with GitWork content"

[GitWork 0fbdcbe] Added hello.xml in GitWork with GitWork content

1 file changed, 1 insertion(+)

create mode 100644 hello.xml

* + - 1. $ git switch master

Switched to branch 'master'

Your branch is up to date with 'origin/master'.

* + - 1. $ echo "<message>Hello from GitWork branch</message>" >hello.xml
      2. $ git add hello.xml

warning: in the working copy of 'hello.xml', LF will be replaced by CRLF the next time Git touches it

* + - 1. $ git commit -m "Added hello.xml in GitWork with GitWork content"

[master 9a57148] Added hello.xml in GitWork with GitWork content

1 file changed, 1 insertion(+)

create mode 100644 hello.xml

* + - 1. $ git log --oneline --graph --decorate --all

\* 9a57148 (**HEAD** -> **master**) Added hello.xml in GitWork with GitWork content

| \* 0fbdcbe (**GitWork**) Added hello.xml in GitWork with GitWork content

|/

\* 75fa433 (**origin/master**) Added branchfile.txt in GitNewBranch

\* 8dd8805 Added .gitignore to ignore .log files and log folder

\* 7ccd431 Initial commit: Added welcome.txt

* + - 1. $ git merge GitWork

Merge made by the 'ort' strategy.

* + - 1. $ notepad++ hello.xml
      2. $ git add hello.xml
      3. $ git commit -m "Resolved merge conflict in hello.xml between master and GitWork"

On branch master

Your branch is ahead of 'origin/master' by 3 commits.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

* + - 1. $ echo "\*.orig" >> .gitignore
      2. $ git add .gitignore

warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it

* + - 1. $ git commit -m "Added .orig to .gitignore"

[master ae2e23a] Added .orig to .gitignore

1. file changed, 1 insertion(+), 1 deletion(-)
   * + 1. $ git branch -d GitWork

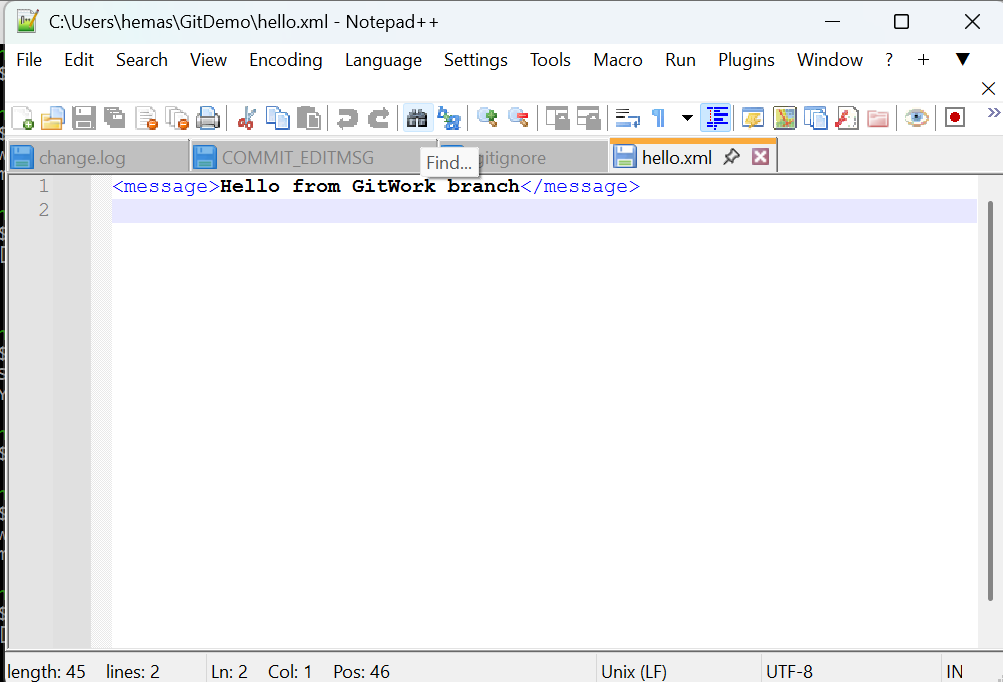
Deleted branch GitWork (was 0fbdcbe).

1. $ git log --online --graph --decorate

fatal: unrecognized argument: --online

1. $ git push

From notepad++:



From github:

A screenshot of a computer

AI-generated content may be incorrect.

**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.

Explanation:

* + - 1. $ rm -r log
      2. $ git status

On branch master

Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

3. $ git branch

\*master

1. $ git branch -d GitWork

error: branch 'GitWork' not found

1. $ git pull origin master

From https://github.com/Sree-1405/GitDemo

\* branch master -> FETCH\_HEAD

Already up to date.

1. $ git push

Everything up-to-date

Nothing is changed in the github local repository this is the final clean and push and commit changes from the git bash.