## 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

```
C:\Users\HP>git --version
git version 2.49.0.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 "Sparshak Ghosh"

>git config --global user.email "sparky010ghosh@gmail.com"
```

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

```
C:\Users\HP>git config --global --list
user.name=Sparshak Ghosh
user.email=sparky010ghosh@gmail.com
```

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

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

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 "/usr/bin/notepad"
```

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

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

Here '-e' option implies editor

It will show the entire global configuration as shown below,

```
.gitconfig
[user]
    name = Sparshak Ghosh
    email = sparky010ghosh@gmail.com
[core]
    editor = '/c/Program Files/Notepad++/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

```
HP@Sparkyyyy MINGW64 ~
$ git init GitDemo
Initialized empty Git repository in C:/Users/HP/GitDemo/.git/
```

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

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (master)
$ echo "Welcome to GIT demo." > welcome.txt
```

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (master)
$ ls -al
total 21
drwxr-xr-x 1 HP 197121  0 Aug 12 20:49 ./
drwxr-xr-x 1 HP 197121  0 Aug 12 20:46 ../
drwxr-xr-x 1 HP 197121  0 Aug 12 20:46 .git/
-rw-r--r-- 1 HP 197121 21 Aug 12 20:49 welcome.txt
```

5. To verify the content, execute the command

```
HP@Sparkyyyy MINGW64 ~/GitDemo (master)
$ cat welcome.txt
Welcome to GIT demo.
```

6. Check the status by executing

```
HP@Sparkyyyy 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)
```

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
```

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

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

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

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

**10.** Signup with GitHub and create a remote repository **"GitDemo"** and add as origin

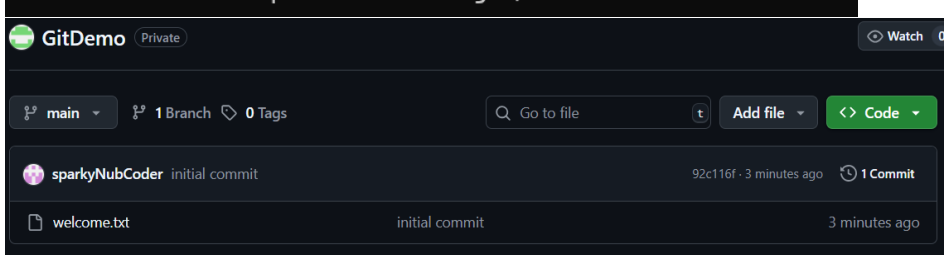
```
HP@Sparkyyyy MINGW64 ~/GitDemo (master)
$ git remote add origin https://github.com/sparshakmr/GitDemo.git
```

**11.** To pull the remote repository, execute

git pull origin master

12. To push the local to remote repository, execute
git push origin master

```
HP@Sparkyyyy MINGW64 ~/GitDemo (master)
$ git branch -M main
git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 245 bytes | 81.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/sparshakmr/GitDemo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```




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**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ touch debug.log

HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ mkdir log
```

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ touch log/example_SOU.log
```

Example_SOU log file created inside that folder

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

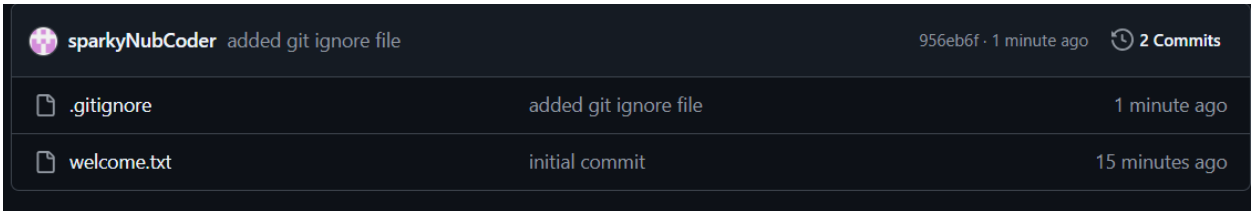
```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

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

Next added the .gitignore and commit is done

**RESULT:**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 592 bytes | 197.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/sparshakmr/GitDemo.git
   92c116f..956eb6f  main -> main
```

**GITHUB:**




## Git HandsOn 3

**Branching**

**Create a new branch "GitNewBranch"**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git branch GitNewBranch
```

**List all the local and remote branches**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git branch -a
  GitNewBranch
* main
  remotes/origin/main
```

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git checkout GitNewBranch
Switched to branch 'GitNewBranch'

HP@Sparkyyyy MINGW64 ~/GitDemo (GitNewBranch)
$ echo "This is a new file in GitNewBranch, Testing" > newfile.txt
```

**Commit the changes to the branch**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (GitNewBranch)
$ git add newfile.txt

HP@Sparkyyyy MINGW64 ~/GitDemo (GitNewBranch)
$ git commit -m "added newfile.tct in GitNewBranch"
[GitNewBranch 0978889] added newfile.tct in GitNewBranch
 1 file changed, 1 insertion(+)
 create mode 100644 newfile.txt
```

**Check the status**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (GitNewBranch)
$ git status
On branch GitNewBranch
nothing to commit, working tree clean
```

**Merging**

**Switch to main**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (GitNewBranch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git diff main GitNewBranch
diff --git a/newfile.txt b/newfile.txt
new file mode 100644
index 0000000..04270c3
--- /dev/null
+++ b/newfile.txt
@@ -0,0 +1 @@
+This is a new file in GitNewBranch, Testing
```

**Visual differences with P4Merge**

git difftool main GitNewBranch
This will open P4Merge for visual comparison.
(its giving me error in my pc)

**Merge the branch into master**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git merge GitNewBranch
Updating 956eb6f..0978889
Fast-forward
 newfile.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 newfile.txt
```

**Observe merge history**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git log --oneline --graph --decorate
* 0978889 (HEAD -> main, GitNewBranch) added newfile.tct in GitNewBranch
* 956eb6f (origin/main) added git ignore file
* 92c116f initial commit
```

**Delete the branch after merging**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git branch -d GitNewBranch
Deleted branch GitNewBranch (was 0978889).
```

## Git HandsOn 4

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git checkout main
Already on 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

HP@Sparkyyyy MINGW64 ~/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
```

2. **Create a branch "GitWork". Add a file "hello.xml".**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git branch GitWork

HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git checkout GitWork
Switched to branch 'GitWork'

HP@Sparkyyyy MINGW64 ~/GitDemo (GitWork)
$ echo "<message>Hello from GitWork</message>" > hello.xml
```

3. **Update the content of "hello.xml" and observe the status**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (GitWork)
$ echo "<message>Hello from GitWork - UPDATED</message>" > hello.xml

HP@Sparkyyyy MINGW64 ~/GitDemo (GitWork)
$ git status
On branch GitWork
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello.xml
```

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (GitWork)
$ git commit -m "added/updated hello.xml on GitWork"
[GitWork 6674b3d] added/updated hello.xml on GitWork
 1 file changed, 1 insertion(+)
 create mode 100644 hello.xml
```

5. **Switch to main.**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (GitWork)
$ git checkout main
Switched to branch 'main'
```

6. **Add a file "hello.xml" to the master and add some different content than previous.**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ echo "<message> Heelo from main branch DIFFERENT </message>" > hello.xml
```

7. **Commit the changes to the master**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git commit -m "added hello.xml with different context"
[main 4dccb53] added hello.xml with different context
 1 file changed, 1 insertion(+)
 create mode 100644 hello.xml
```

8. **Observe the log by executing "git log –oneline –graph –decorate –all"**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git log --oneline --graph --decorate --all
* 4dccb53 (HEAD -> main) added hello.xml with different context
| * 6674b3d (GitWork) added/updated hello.xml on GitWork
|/
* 0978889 added newfile.tct in GitNewBranch
* 956eb6f (origin/main) added git ignore file
* 92c116f initial commit
```

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git diff main GitWork
diff --git a/hello.xml b/hello.xml
index 1d85526..24ecfc9 100644
--- a/hello.xml
+++ b/hello.xml
@@ -1 +1 @@
-<message> Heelo from main branch DIFFERENT </message>
+<message>Hello from GitWork - UPDATED</message>
```

10. **Merge the branch to the master main**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git merge GitWork
Auto-merging hello.xml
```

11. **Observe the git mark up.**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main|MERGING)
$ cat hello.xml
<<<<<<< HEAD
<message> Heelo from main branch DIFFERENT </message>
=======
<message>Hello from GitWork - UPDATED</message>
>>>>>>> GitWork
```

12. **Commit the changes to the main**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main|MERGING)
$ git add hello.xml

HP@Sparkyyyy MINGW64 ~/GitDemo (main|MERGING)
$ git commit -m "merged gitwork into main and added hello.xml"
[main 9b5758b] merged gitwork into main and added hello.xml
```

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
  (use "git push" to publish your local commits)
```

14. **Commit the changes to the .gitignore**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git commit -m "added hello.xml in .gitignore"
[main 242f491] added hello.xml in .gitignore
 1 file changed, 3 insertions(+)
```

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git branch -a
  GitWork
* main
  remotes/origin/main
```

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git branch -d GitWork
Deleted branch GitWork (was 6674b3d).
```

**17. Observe the log by executing "git log –oneline –graph –decorate"**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git log --oneline --graph --decorate
* 242f491 (HEAD -> main) added hello.xml in .gitignore
*   9b5758b merged gitwork into main and added hello.xml
|\
| * 6674b3d added/updated hello.xml on GitWork
* | 4dccb53 added hello.xml with different context
|/
* 0978889 added newfile.tct in GitNewBranch
* 956eb6f (origin/main) added git ignore file
* 92c116f initial commit
```

## Git HandsOn 5

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git branch -a
* main
  remotes/origin/main
```

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

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git pull origin main
From https://github.com/sparshakmr/GitDemo
 * branch            main       -> FETCH_HEAD
Already up to date.
```

**4. Push the changes, which are pending from “Git-T03-HOL_002” to the remote repository.**

```
HP@Sparkyyyy MINGW64 ~/GitDemo (main)
$ git push
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 12 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (15/15), 1.59 KiB | 816.00 KiB/s, done.
Total 15 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To https://github.com/sparshakmr/GitDemo.git
   956eb6f..242f491  main -> main
```

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


YES CHANGES ARE OBSERVED