# Hands-on Lab: Using Git from Your Own Desktop

**Estimated time needed:** : 30 mins

## Objectives

After completing this lab, you will be able to:

1. Clone your GitHub repository locally.
2. Make changes to the cloned files.
3. Add a new file.
4. Check the status.
5. Commit changes.
6. Generate Personal Access Token.
7. Push the changes back to GitHub.

## Pre-requisites

GitHub account, with a project in it, as illustrated in this lab.

GitBash or Git installed on your local desktop, as in this lab.

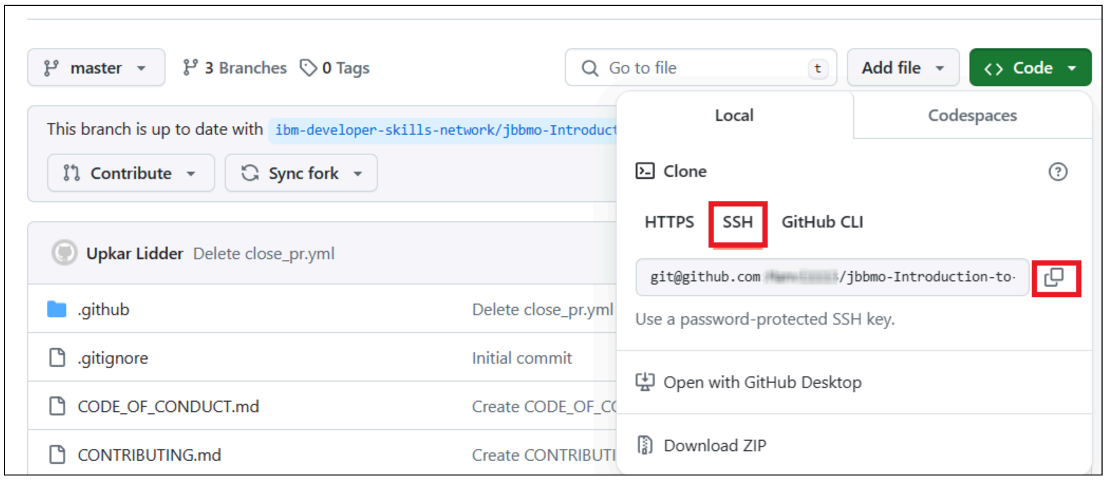
Create SSH keys, as in this lab

Add SSH Key to GitHub, as in this lab

### Exercise 1: Clone a repo

To clone a repo, you need the SSH URL of the repo.

1. To get the SSH URL, login into GitHub.
2. Navigate to the repo you wish to clone.
3. Click the 'Code' button.
4. Click the 'clipboard icon' to copy the SSH URL. Paste this URL where you can access it later.



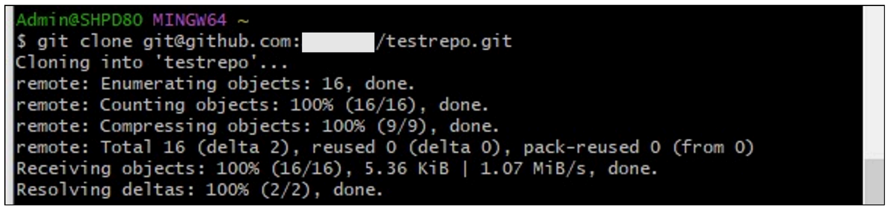
This branch is up to date with ibm-developer-skills-network/jbbmo-Introduct

Upkar Lidder Delete close_pr.yml

git@github.com [redacted]/jbbmo-Introduction-to-

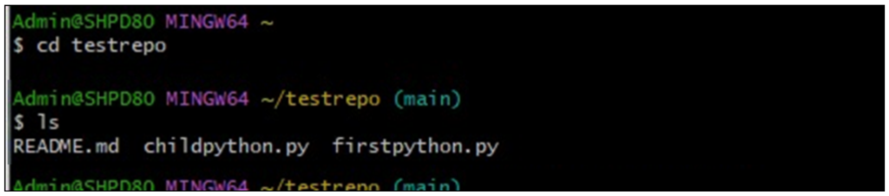
Use a password-protected SSH key.

5. On your desktop, open a terminal or GitBash, if you are using Windows OS.
6. Navigate to a directory where you wish to clone the repo.
7. Run the command `git clone <your repo ssh url>`
8. This will clone the repo on GitHub into your current directory.
9. You can see all the downloaded files under a directory named as your repo name.



```
Admin@SHPD80 MINGW64 ~
$ git clone git@github.com:[redacted]/testrepo.git
Cloning into 'testrepo'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 16 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (16/16), 5.36 KiB | 1.07 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```

10. To ensure that every file was downloaded, navigate to the cloned directory and list the files.

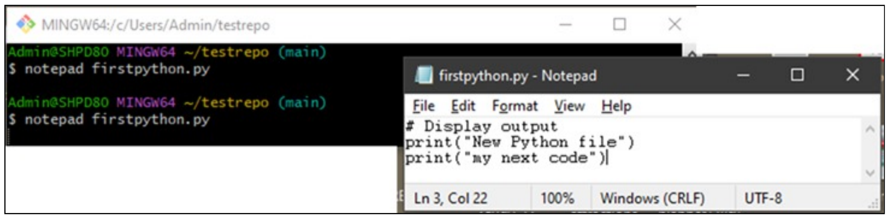


```
Admin@SHPD80 MINGW64 ~
$ cd testrepo

Admin@SHPD80 MINGW64 ~/testrepo (main)
$ ls
README.md  childpython.py  firstpython.py
```

## Exercise 2: Make changes to cloned files

1. Using your favourite editor, open any one of the file inside repo and make changes to the file and save it.



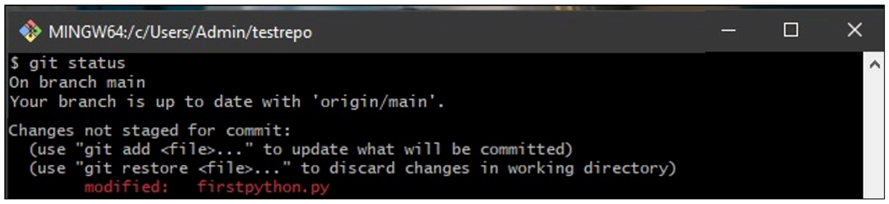
```
Admin@SHPD80 MINGW64 ~/testrepo (main)
$ notepad firstpython.py

Admin@SHPD80 MINGW64 ~/testrepo (main)
$ notepad firstpython.py
```

```
# Display output
print("New Python file")
print("my next code")
```

MINGW64:/c/Users/Admin/testrepo
firstpython.py - Notepad
File Edit Format View Help
Ln 3, Col 22 100% Windows (CRLF) UTF-8

2. Type the command `git status` to show all the modified files.

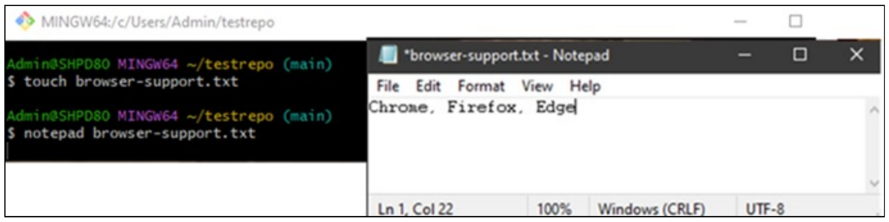


```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   firstpython.py
```

## Exercise 3: Add a new file to the local repo

1. Let us add a new file to the local repo. Using a text editor, create a new file **browser-support.txt**.
2. Add "Chrome, Firefox, Edge" into the file.



```
Admin@SHPD80 MINGW64 ~/testrepo (main)
$ touch browser-support.txt

Admin@SHPD80 MINGW64 ~/testrepo (main)
$ notepad browser-support.txt
```

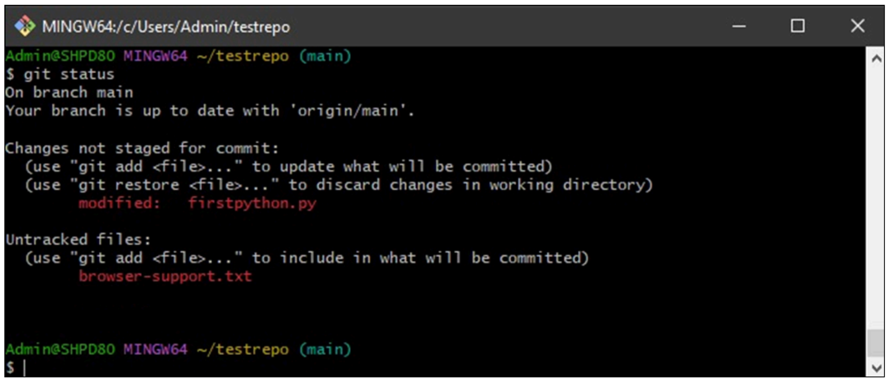
```
Chrome, Firefox, Edge
```

MINGW64:/c/Users/Admin/testrepo
*browser-support.txt - Notepad
File Edit Format View Help
Ln 1, Col 22 100% Windows (CRLF) UTF-8

4. Save the file.

## Exercise 4: Check the status

1. Run `git status` to see info on the modified files.



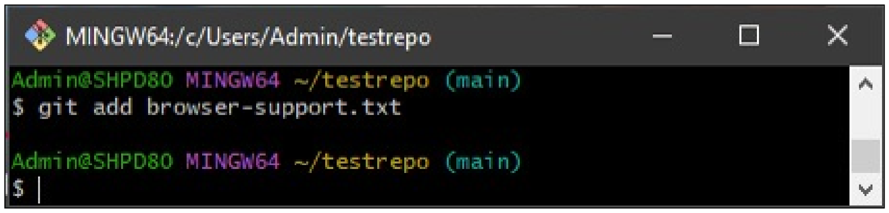
```
Admin@SHPD80 MINGW64 ~/testrepo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   firstpython.py

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


Admin@SHPD80 MINGW64 ~/testrepo (main)
$ |
```

2. Add the file to the repository for committing using `git add browser-support.txt`.

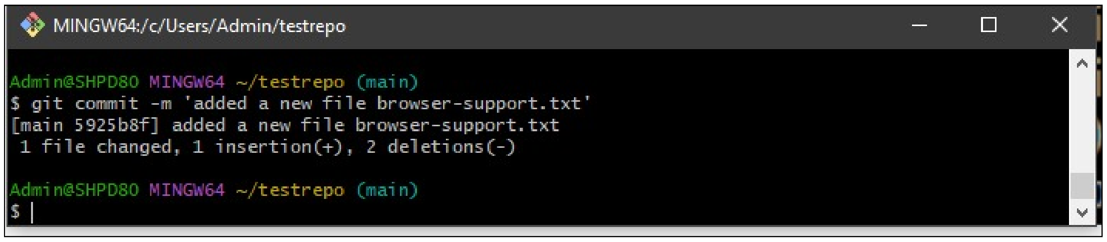


```
MINGW64:/c/Users/Admin/testrepo
Admin@SHPD80 MINGW64 ~/testrepo (main)
$ git add browser-support.txt

Admin@SHPD80 MINGW64 ~/testrepo (main)
$ |
```

### Exercise 5: Commit and push the changes

1. Git commit will record all the changes into the local stating area. To commit the changes you have made, run `git commit -m 'added a new file browser-support.txt'`.



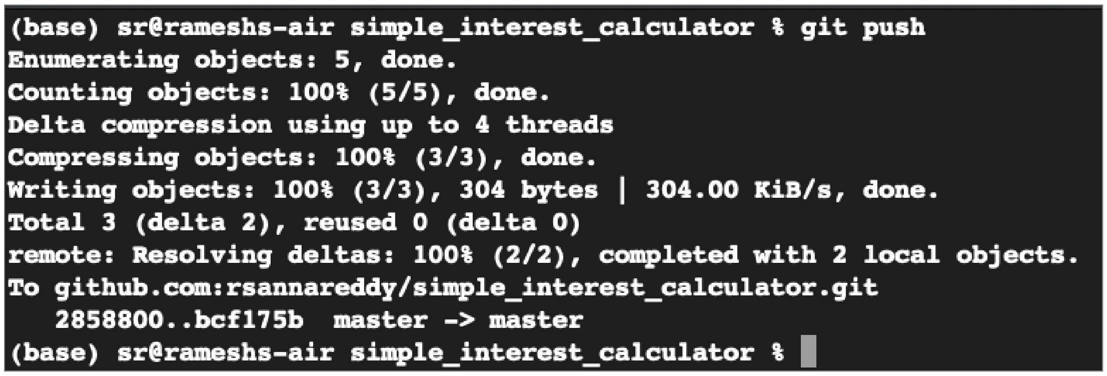
```
MINGW64:/c/Users/Admin/testrepo

Admin@SHPD80 MINGW64 ~/testrepo (main)
$ git commit -m 'added a new file browser-support.txt'
[main 5925b8f] added a new file browser-support.txt
 1 file changed, 1 insertion(+), 2 deletions(-)

Admin@SHPD80 MINGW64 ~/testrepo (main)
$ |
```

Now all the changes you have made this far, get committed locally.

2. The git `push` command will enable you to sync all the changes made locally to the GitHub web repository. Run the `git push` command in GitBash terminal.



```
(base) sr@rameshs-air simple_interest_calculator % git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 304 bytes | 304.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:rsannareddy/simple_interest_calculator.git
   2858800..bcf175b  master -> master
(base) sr@rameshs-air simple_interest_calculator %
```

You can now visit the GitHub repository page and check to ensure that the revised and newly added files are in place.

## Summary

In this lab, you have learned how to clone a GitHub repository, make changes to it, commit the changes locally, and push it back to GitHub.

## Author(s)

**Ramesh Sannareddy**

### Other Contributor(s)

Rav Ahuja

Skills Network