Exercise 1 : Write a procedure to Deploy Version Control System / Source Code Management, install git and create a GitHub account.

Aim:

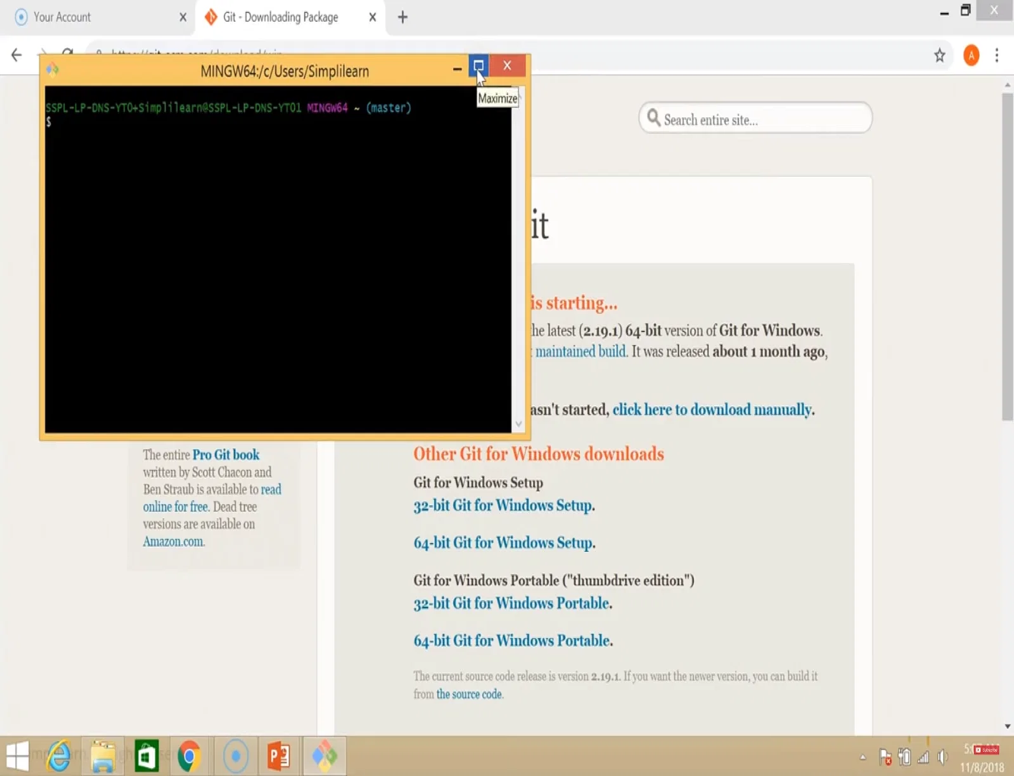
Procedure:

Steps For Git Installation on Windows

1. Download and install Git
2. Git bash interface
3. Basic Git commands
4. Create a local repository
5. Connect to the remote repository
6. Push the file to GitHub

Step 1:

Download the [latest version of Git](https://git-scm.com/downloads) and choose the 64/32 bit version. After the file is downloaded, install it in the system. Once installed, select Launch the Git Bash, then click on finish. The Git Bash is now launched.



Step 2:

Check the Git version:

| $ git --version |
| --- |

Step 3:

For any help, use the following command:

| $ git help config |
| --- |

This command will lead you to a browser of [config commands](https://www.simplilearn.com/tutorials/git-tutorial/git-commands). Basically, the help the command provides a manual from the help page for the command just following it (here, it's config).

Another way to use the same command is as follows:

| $ git config --help |
| --- |

Step 4:

Create a local directory using the following command:

| $ mkdir test  $ cd test |
| --- |

Step 5:

The next step is to initialize the directory:

| $ git init |
| --- |

Step 6:

Go to the folder where “test” is created and create a text document named “demo.” Open “demo” and put any content, like “Hello MCA.” Save and close the file.

Step 7:

Enter the Git bash interface and type in the following command to check the status:

| $ git status |
| --- |

Step 8:

Add the “demo” to the current directory using the following command:

| $ git add demo.txt |
| --- |

Step 9:

Next, make a commit using the following command:

| $ git commit -m “committing a text file” |
| --- |

Step 10:

Link the Git to a [Github](https://www.simplilearn.com/tutorials/git-tutorial/what-is-github) Account:

| $ git config --global user.username |
| --- |

Note: simplilearn-github is the username on the Github account.

Step 11:

Open your Github account and create a new repository with the name "test\_demo" and click on "Create repository." This is the remote repository. Next, copy the link of "test\_demo."

Step 12:

Go back to Git bash and link the remote and local repository using the following command:

| $ git remote add origin <link> |
| --- |

Here, <link> is the link copied in the previous step.

Step 13:

Push the local file onto the remote repository using the following command:

| $ git push origin master |
| --- |

Step 14:

Move back to Github and click on "test\_demo" and check if the local file "demo.txt" is pushed to this repository.

1. There are some experimental options available such as pseudo control Support or Built in file system monitor concerning your installed Git version.

How to Launch Git in Windows?

There are two methods to launch git in windows. One is launching git using a bash scripting shell with the help of the command line and another is launching git using a graphical user interface.

1. To launch git via bash scripting shell,   
   First, open the window and search for git bash and open it.
2. To launch git via graphical user interface(GUI), similarly, first open the window and search for git GUI and click on the application icon and open it.

Configure GitHub Credentials

You can configure your local GitHub installation with credentials by using the following commands. Also, don't forget to add your own GitHub credentials for username and email address.

1. git config –global user.n   
   ame "github\_username"
2. git config –global user.e   
   mail "email\_address"

Clone a GitHub Repository

1. Initially you need to click the options repository on GitHub.
2. Then in the top right corner, click the option clone or download where a small drop-down box will appear having a URL for cloning over HTTPS.
3. Then enter into your Powershell windows and write clone URL as:  
   git clone repository\_url
4. On the other hand, you can clone a github repository with SSH URLs where first you need to generate an SSH key pair on your windows workstation as well as need to assign a public key to your GitHub account.

List Remote Repositories

1. Make a copy of the repository from GitHub for your working directory.
2. Ensure that the working directory should have the project name as   
   "cd git\_project" and replace the project name from the downloaded repository.
3. If the above option doesn't work, you can list the content using "ls command" for the current directory, especially to check your exact number of spellings.
4. Besides, you can list the remote repository in the sub-directory as "git remote -v".

Output:

—---------------------------------------------------------------------------------------------

Exercise 2: Write a procedure to perform various GIT operations on local and Remote repositories using GIT Cheat-Sheet

Aim:

Procedure:

1. Git configuration

* **Git config**Get and set configuration variables that control all facets of how Git looks and operates.  
  **Set the name:**$ git config --global user.name "User name"  
  **Set the email:**$ git config --global user.email "gitcheatsheet@gmail.com"  
  **Set the default editor:**$ git config --global core.editor Vim  
  **Check the setting:**$ git config -list
* **Git alias  
  Set up an alias** for each command:  
  $ git config --global alias.co checkout  
  $ git config --global alias.br branch  
  $ git config --global alias.ci commit  
  $ git config --global alias.st status

2. Starting a project

* **Git init  
  Create a local repository:**$ git init
* **Git clone  
  Make a local copy** of the server repository.  
  $ git clone

3. Local changes

* **Git add  
  Add a file** to staging (Index) area:  
  $ git add Filename  
  **Add all files** of a repo to staging (Index) area:  
  $ git add\*
* **Git commit  
  Record** or snapshots the file permanently in the version history **with a message**.  
  $ git commit -m " Commit Message"

4. Track changes

* **Git diff**Track the changes that have not been staged: $ git diff  
  Track the changes that have staged but not committed:  
  $ git diff --staged  
  Track the changes after committing a file:  
  $ git diff HEAD  
  Track the changes between two commits:  
  $ git diff Git Diff Branches:  
  $ git diff < branch 2>
* **Git status**Display the state of the working directory and the staging area.  
  $ git status
* **Git show Shows objects:**$ git show

5. Commit History

* **Git log**Display the most recent commits and the status of the head:  
  $ git log  
  Display the output as one commit per line:  
  $ git log -oneline  
  Displays the files that have been modified:  
  $ git log -stat  
  Display the modified files with location:  
  $ git log -p
* **Git blame**Display the modification on each line of a file:  
  $ git blame <file name>

6. Ignoring files

* **.gitignore**Specify intentionally untracked files that Git should ignore. Create .gitignore:  
  $ touch .gitignore List the ignored files:  
  $ git ls-files -i --exclude-standard

7. Branching

* **Git branch Create branch:**$ git branch List Branch:  
  $ git branch --list Delete a Branch:  
  $ git branch -d Delete a remote Branch:  
  $ git push origin -delete Rename Branch:  
  $ git branch -m
* **Git checkout**Switch between branches in a repository.  
  Switch to a particular branch:  
  $ git checkout   
  Create a new branch and switch to it:   
  $ git checkout -b Checkout a Remote branch:  
  $ git checkout
* **Git stash**Switch branches without committing the current branch. Stash current work:  
  $ git stash  
  Saving stashes with a message:  
  $ git stash save ""  
  Check the stored stashes:  
  $ git stash list  
  Re-apply the changes that you just stashed:  
  $ git stash apply  
  Track the stashes and their changes:  
  $ git stash show  
  Re-apply the previous commits:  
  $ git stash pop  
  Delete a most recent stash from the queue:  
  $ git stash drop  
  Delete all the available stashes at once:  
  $ git stash clear  
  Stash work on a separate branch:  
  $ git stash branch
* **Git cherry pic**Apply the changes introduced by some existing commit:  
  $ git cherry-pick

8. Merging

* **Git merge**Merge the branches:  
  $ git merge   
  Merge the specified commit to currently active branch:  
  $ git merge
* **Git rebase**Apply a sequence of commits from distinct branches into a final commit.  
  $ git rebase   
  Continue the rebasing process:  
  $ git rebase -continue Abort the rebasing process:  
  $ git rebase --skip
* **Git interactive rebase**Allow various operations like edit, rewrite, reorder, and more on existing commits.  
  $ git rebase -i

9. Remote

* **Git remote**Check the configuration of the remote server:  
  $ git remote -v  
  Add a remote for the repository:  
  $ git remote add Fetch the data from the remote server:  
  $ git fetch   
  Remove a remote connection from the repository:  
  $ git remote rm   
  Rename remote server:  
  $ git remote rename   
  Show additional information about a particular remote:  
  $ git remote show   
  Change remote:  
  $ git remote set-url
* **Git origin master**Push data to the remote server:   
  $ git push origin master Pull data from remote server:  
  $ git pull origin master

10. Pushing Updates

* **Git push**Transfer the commits from your local repository to a remote server. Push data to the remote server:  
  $ git push origin master Force push data:  
  $ git push -f  
  Delete a remote branch by push command:  
  $ git push origin -delete edited

11. Pulling updates

* **Git pull**Pull the data from the server:  
  $ git pull origin master  
  Pull a remote branch:  
  $ git pull
* **Git fetch**Download branches and tags from one or more repositories. Fetch the remote repository:  
  $ git fetch< repository Url> Fetch a specific branch:  
  $ git fetch   
  Fetch all the branches simultaneously:  
  $ git fetch -all  
  Synchronize the local repository:  
  $ git fetch origin

12. Undo changes

* **Git revert**Undo the changes:  
  $ git revert  
  Revert a particular commit:  
  $ git revert
* **Git reset**Reset the changes:  
  $ git reset -hard  
  $ git reset -soft:  
  $ git reset --mixed

13. Removing files

* **Git rm**Remove the files from the working tree and from the index:  
  $ git rm <file Name>  
  Remove files from the Git But keep the files in your local repository:  
  $ git rm –cached

Output: