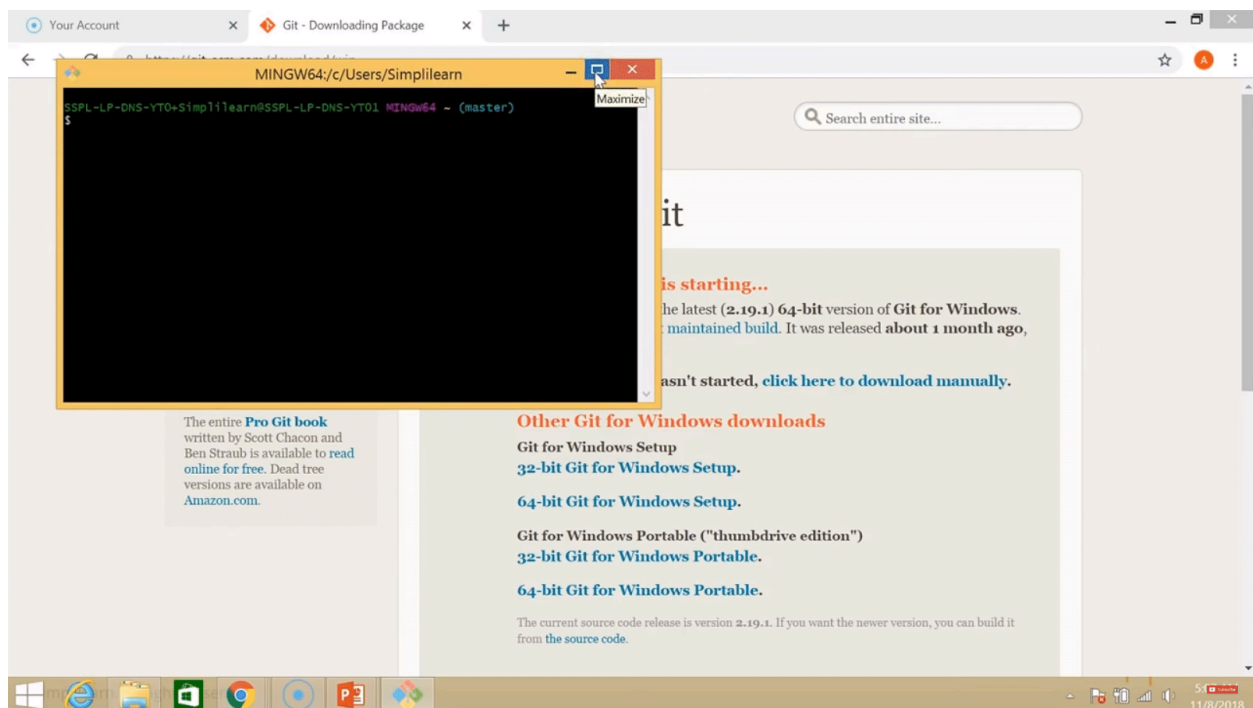


Git Installation on Windows

Let us now look at the various steps in the install git:

Step 1:

Download the [latest version of Git](#) 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:

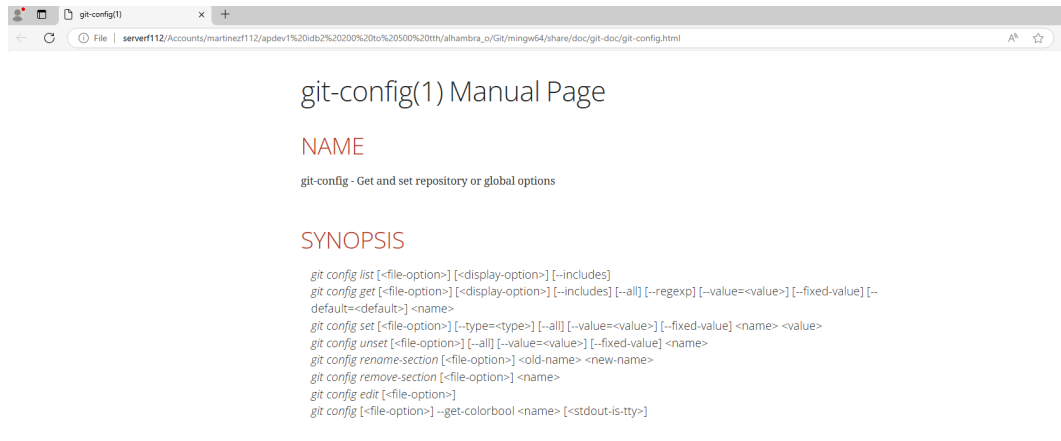
```
$ git --version
```

```
alhambra_o@f112ws08 MINGW64 ~  
$ git --version  
git version 2.47.1.windows.2
```

Step 3:

For any help, use the following command:

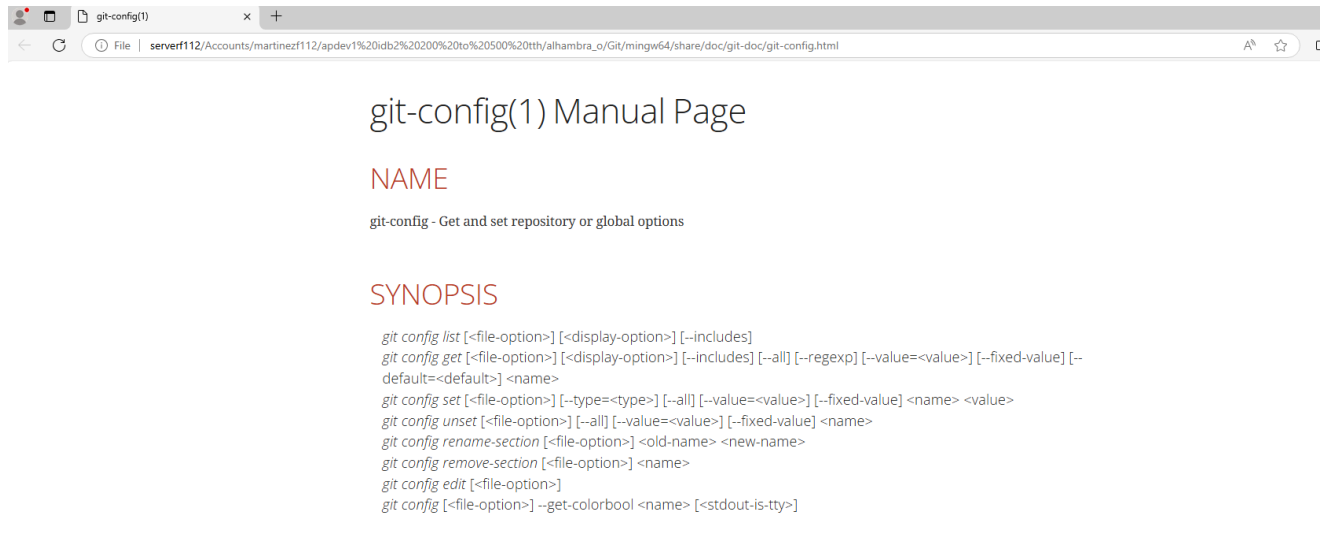
```
$ git help config
```



This command will lead you to a browser of [config 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
```

```
alhambra_o@f112ws08 MINGW64 /z/APPDEV1Alhambra (master)
$ mkdir test

alhambra_o@f112ws08 MINGW64 /z/APPDEV1Alhambra (master)
$ cd test
```

Step 5:

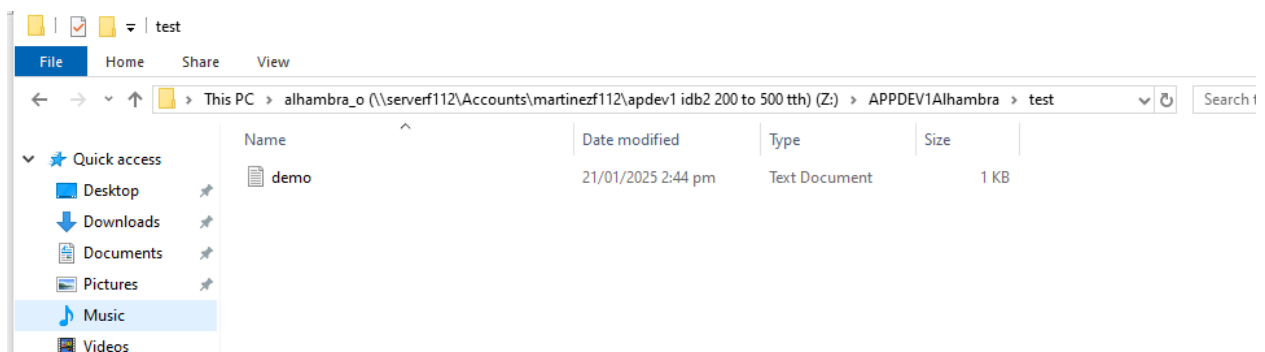
The next step is to initialize the directory:

```
$ git init
```

```
alhambra_o@f112ws08 MINGW64 /z/APPDEV1Alhambra/test (master)
$ git init
Initialized empty Git repository in //SERVERF112/Accounts/martinezf112/apdev1 id
b2 200 to 500 tth/alhambra_o/APPDEV1Alhambra/test/.git/
```

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 Simplilearn." Save and close the file.



Step 7:

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

\$ git status

```
alhabra_o@f112ws08 MINGW64 /z/APPDEV1Alhabra/test (master)
$ git status
On branch master

No commits yet

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

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

Step 8:

Add the "demo" to the current directory using the following command:

\$ git add demo.txt

```
alhabra_o@f112ws08 MINGW64 /z/APPDEV1Alhabra/test (master)
$ git add demo.txt
```

Step 9:

Next, make a commit using the following command:

\$ git commit -m "committing a text file"

```
alhabra_o@f112ws08 MINGW64 /z/APPDEV1Alhabra/test (master)
$ git commit -m "committing a text file"
[master (root-commit) 1falcf8] committing a text file
1 file changed, 1 insertion(+)
create mode 100644 demo.txt
```

Step 10:

Link the Git to a [Github](#) Account:

\$ git config --global user.username

```
alhabra_o@f112ws08 MINGW64 /z/APPDEV1Alhabra/test (master)
$ git config --global user.UBSITAAlhabra
```

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."

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * UBSITAAlhambra / Repository name * test_demo

test_demo is available.

Great repository names are short and memorable. Need inspiration? How about [vigilant-fishstick](#) ?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

[Create repository](#)

Step 12:

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

```
$ git remote add origin <link>
```

```
alhambra_o@f112ws08 MINGW64 /z/APPDEV1Alhambra/test (master)
$ git remote add origin https://github.com/UBSITAAlhambra/demo.git
```

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

Step 13:

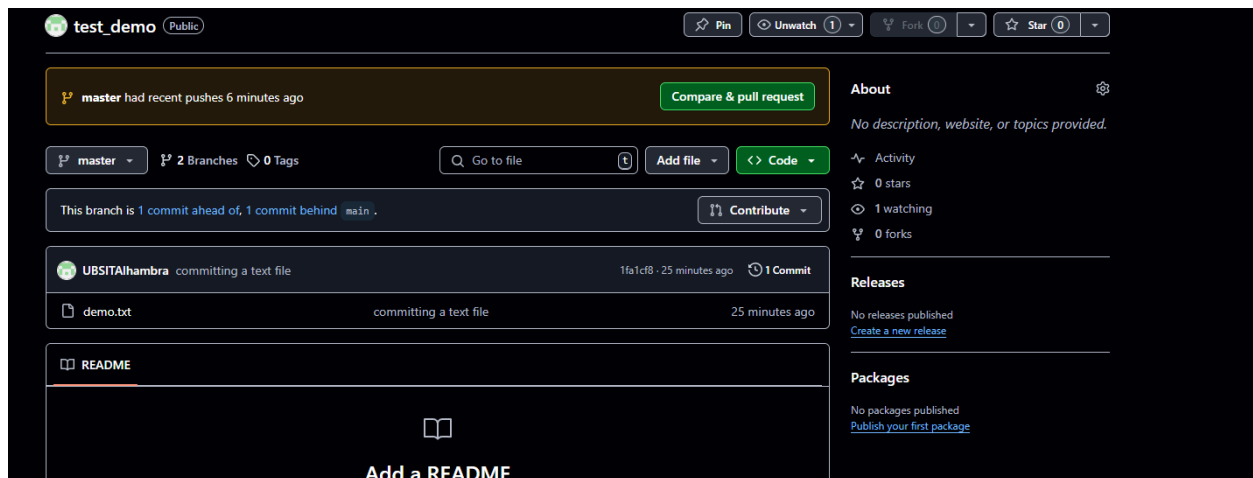
Push the local file onto the remote repository using the following command:

```
$ git push origin master
```

```
alhambra_o@f112ws08 MINGW64 /z/APPDEV1Alhambra/test (master)
$ git push origin master
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 234 bytes | 117.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/UBSITAlhambra/test_demo/pull/new/master
remote:
To https://github.com/UBSITAlhambra/test_demo.git
 * [new branch]      master -> 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.



Additional Customization Options

1. This option enables users to add extra elements such as symbolic links for command lines. Nevertheless, one should always prefer default options for shortcuts or more.
2. 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.name "github_username"`
2. `git config --global user.email "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"`.

Summary: Steps For Git Installation on Windows 10

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

What is Git?

Git is a widely used modern version control system for tracking changes in computer files. The term version control system suggests a system that records all the changes made to a file or set of data, so a specific version can be considered whenever needed. This feature makes the process of collaboration so feasible with all team members, making it considerably more comfortable to work over a big project.

Git makes it possible for several people involved in the project to work together and track each other's progress over time. In software development, the tool helps in Source Code Management. Git favors not only programmers but also non-technical users by keeping track of their project files.

While [working on Git](#), we actively use two repositories.

- Local repository: The local repository is present on our computer and consists of all the files and folders. This Repository is used to make changes locally, review history, and commit when offline.
- Remote repository: The remote repository refers to the server repository that may be present anywhere. This repository is used by all the team members to exchange the changes made.

Both repositories have their own set of commands. There are separate Git Commands that work on different types of repositories.

Git Commands: Working With Local Repositories

- git init
- The command git init is used to create an empty Git repository.
- After the git init command is used, a .git folder is created in the directory with some subdirectories. Once the repository is initialized, the process of creating other files begins.

git init

```
alhambra_o@f112ws08 MINGW64 /z/APPDEV1Alhambra/test (master)
$ git init
Reinitialized existing Git repository in //SERVERF112/Accounts/martinezf112/apdev1 idb2 200 to 500 tth/alhambra_o/APPDEV1Alhambra/test/.git/
```



```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo
$ git init
Initialized empty Git repository in C:/Users/Taha/Git_demo/FirstRepo/.git/
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ |
```

- git add
- Add command is used after checking the status of the files, to add those files to the staging area.
- Before running the commit command, "git add" is used to add any new or modified files.

git add .

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

nothing added to commit but untracked files present (use "git add" to track)
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git add .
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$
```

- git commit
- The commit command makes sure that the changes are saved to the local repository.
- The command "git commit -m <message>" allows you to describe everyone and help them understand what has happened.

git commit -m "commit message"

```

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git status
On branch master

No commits yet

changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   alpha.txt

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git commit -m "alpha"
[master (root-commit) b89b00a] alpha
 1 file changed, 1 insertion(+)
 create mode 100644 alpha.txt

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$

```

- git status
- The git status command tells the current state of the repository.
- The command provides the current working branch. If the files are in the staging area, but not committed, it will be shown by the git status. Also, if there are no changes, it will show the message no changes to commit, working directory clean.

git status

```

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git status
On branch master

No commits yet

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

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

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$

```

```

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git status
On branch master
nothing to commit, working tree clean

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$

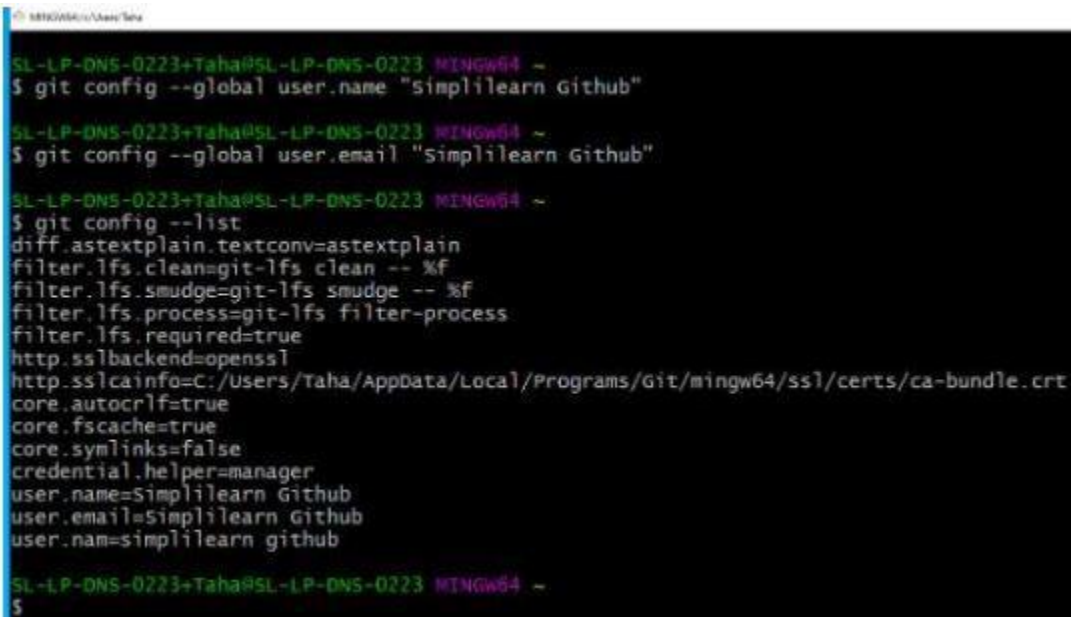
```

- git config

- The git config command is used initially to configure the user.name and user.email. This specifies what email id and username will be used from a local repository.
- When git config is used with --global flag, it writes the settings to all repositories on the computer.

git config --global user.name "any user name"

git config --global user.email <email id>



```

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~
$ git config --global user.name "simplilearn github"

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~
$ git config --global user.email "simplilearn Github"

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Users/Taha/AppData/Local/Programs/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.sylinks=false
credential.helper=manager
user.name=Simplilearn Github
user.email=Simplilearn Github
user.name=simplilearn github
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~
$

```

- git branch
- The git branch command is used to determine what branch the local repository is on.
- The command enables adding and deleting a branch.

Create a new branch
git branch <branch_name>

List all remote or local branches
git branch -a

Delete a branch

```
git branch -d <branch_name>
```

- git checkout
- The git checkout command is used to switch branches, whenever the work is to be started on a different branch.
- The command works on three separate entities: files, commits, and branches.

Checkout an existing branch

```
git checkout <branch_name>
```

Checkout and create a new branch with that name

```
git checkout -b <new_branch>
```

- git merge
- The [git merge](#) command is used to integrate the branches together. The command combines the changes from one branch to another branch.
- It is used to merge the changes in the staging branch to the stable branch.

```
git merge <branch_name>
```

However, these are popular and basic git commands used by developers.

Git Commands: Working With Remote Repositories

- git remote
- The git remote command is used to create, view, and delete connections to other repositories.
- The connections here are not like direct links into other repositories, but as bookmarks that serve as convenient names to be used as a reference.

```
git remote add origin <address>
```

```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo (master)
$ git remote add origin https://github.com/simplilearn-github/FirstRepo.git

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo (master)
$ git remote -v
origin https://github.com/simplilearn-github/FirstRepo.git (fetch)
origin https://github.com/simplilearn-github/FirstRepo.git (push)
```

- git clone
- The git clone command is used to create a local working copy of an existing remote repository.
- The command downloads the remote repository to the computer. It is equivalent to the Git init command when working with a remote repository.

```
git clone <remote_URL>
```

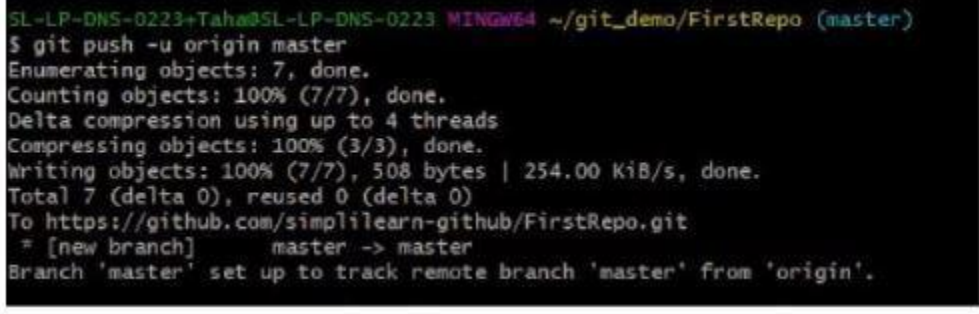
- git pull
- The [git pull command](#) is used to fetch and merge changes from the remote repository to the local repository.
- The command "git pull origin master" copies all the files from the master branch of the remote repository to the local repository.

```
git pull <branch_name> <remote URL>
```

```
chinmayee.deshpande@SL-LP-DNS-0158 MINGW64 ~/git_demo/Changes (master)
$ git pull https://github.com/simplilearn-github/FirstRepo.git
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 16 (delta 1), reused 15 (delta 0), pack-reused 0
Unpacking objects: 100% (16/16), 4.45 MiB | 819.00 KiB/s, done.
From https://github.com/simplilearn-github/FirstRepo
* branch                HEAD              -> FETCH_HEAD
```

- git push
- The command [git push](#) is used to transfer the commits or pushing the content from the local repository to the remote repository.
- The command is used after a local repository has been modified, and the modifications are to be shared with the remote team members.

```
git push -u origin master
```

A terminal window screenshot showing the execution of the 'git push' command. The prompt is 'SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo (master)'. The command entered is '\$ git push -u origin master'. The output shows the process of enumerating, counting, and compressing objects, followed by writing them to the remote repository. It indicates that the 'master' branch is now tracking the remote 'master' branch.

```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo (master)
$ git push -u origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (7/7), 508 bytes | 254.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/simplilearn-github/FirstRepo.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Some Advanced Git Commands

- git stash
- The git stash command takes your modified tracked files and saves it on a pile of incomplete changes that you can reapply at any time. To go back to work, you can use the stash pop.
- The git stash command will help a developer switch branches to work on something else without committing to incomplete work.

```
# Store current work with untracked files
git stash -u
```

```
# Bring stashed work back to the working directory
git stash pop
```

- git log
- The git log command shows the order of the commit history for a repository.
- The command helps in understanding the state of the current branch by showing the commits that lead to this state.

```
git log
```

```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git log
commit b89b00ab7b7b1cc7425583769602c7ac1432ce5d (HEAD -> master)
Author: Simplilearn GitHub <siddam.bharat@simplilearn.net>
Date: Thu Mar 12 07:14:56 2020 +0530
```

alpha

```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ |
```