**GitHub Bash Commands**

**Create a new repository on GitHub**

1. To begin, create an account on GITHUB and then sign in to your user account on [GitHub](https://github.com/).
2. In the upper right corner, click the + sign icon, then choose **New repository**. This will take you to a page where you can enter a repository name
3. Once you have entered a repository name and made your selection, select **Create repository**, and you will be taken to your new repository web page.

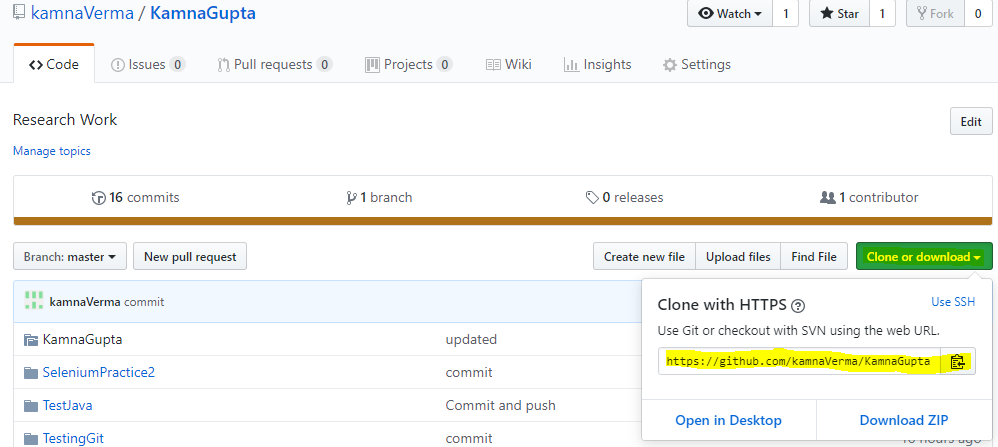
**Git at the command line**

2 types of commands that we use are:

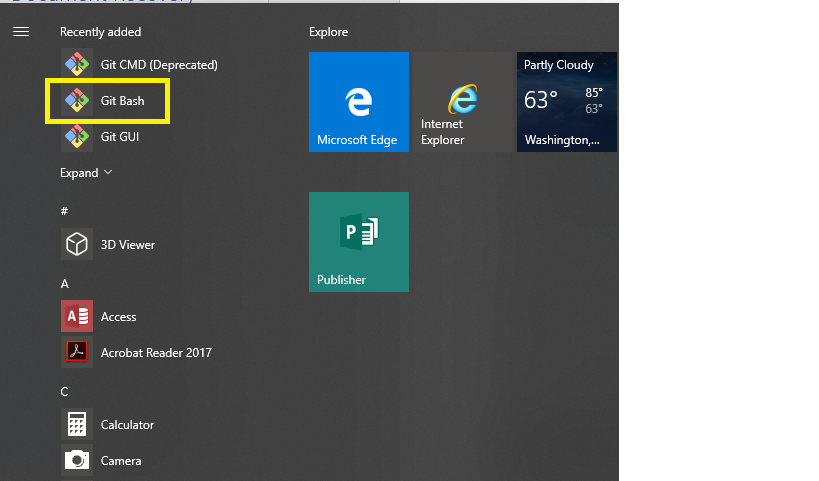
1. **Bash commands:** These bash / shell commands allow you to navigate around your computer, explore directory structures, create and manipulate files and directories, and more. (e.g. ls, cd, mkdir, etc)
2. **Git commands:** These are commands that are specific to git and will only be available if you have git installed on your computer. (e.g. git init, git status, git clone, git push , git commit, git pull etc)

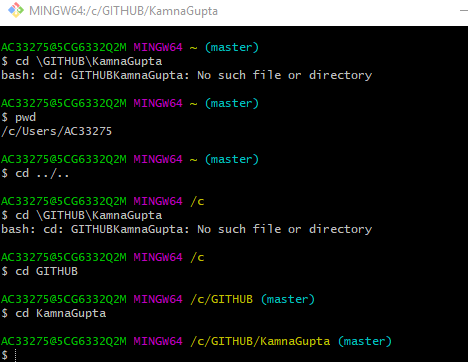
**Clone your repository to your local machine**

Next, clone your newly created repository from GitHub to your local computer. From your repository page on GitHub, click the green button labelled **Clone or download**, and in the “Clone with HTTPs” section, copy the URL for your repository.

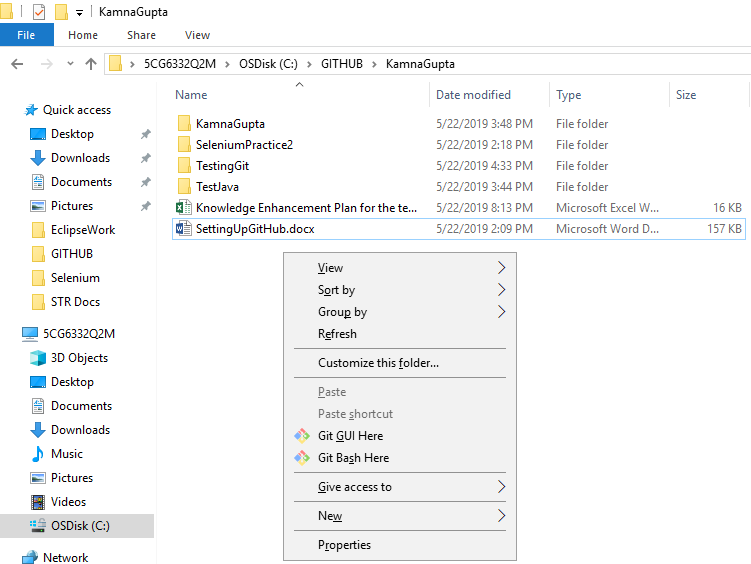


Next, on your local machine, open your bash shell and change your current working directory to the location where you would like to clone your repository. Note that here we are using a bash command **- cd(change directory)** as explained in below screenshot:





Or you can directly go to that location in Windows and right click use GIT Bash here option



**Git command** to clone/copies repository to your local machine:

git clone <https://github.com/YOUR-USERNAME/YOUR-REPOSITORY>



When you run git clone repo-path-here, You might see output like:

Cloning into 'test-repo'...

remote: Counting objects: 5, done.

remote: Compressing objects: 100% (4/4), done.

remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (5/5), done.

Checking connectivity... done.

Note: The repository name and output numbers that you see on your computer, representing the total file size, etc, may differ from the example provided above.

To verify that your repository now exists locally, type ls in your terminal. The ls command lists the files & folders available in your current directory. You should see a directory with the same name as the repository that you created previously on GitHub.

**Tracking changes with git add and git commit**

Next use cd to **c**hange **d**irectories using the syntax:

cd my-repo-name

Replace my-repo-name with the folder name of your repo (this should be your repo name - e.g. 14ers-git)

cd test-repo

If you list all the files in this directory (using ls -a), you should see all of the files that exist in your GitHub repository:

ls -a

.git .gitignore LICENSE README.md

**Important Tip** The .git element is listed when you use the ls -a command shows up is actually a directory which will keep track of your changes (the commits that you make) in git. **Warning:** Do not edit the files in this directory manually!

**Edit a file in your repo**

Create a new file in your local machine and now try to add it to the GiT repository.

Once you are happy with your changes and have saved them, go back to your terminal window and type git status and hit return to execute the command.

1. add the changes, then
2. commit the changes.
3. Push the changes to git repository

Modified files are staged using git add, and following a commit, all files in the staging area are snapshotted and become part of the repository's history,

**Add files**

You can add an individual file or groups of files to git tracking. To add a single file, use

git add file-name-here-with-extension

To add the README.md file that you just modified

git add README.md

To add ALL of the files that you have edited at the same time

git add --all

Use git add --all with caution.

**Commit files**

The git commit command requires a **commit message** that describes the snapshot / changes that you made in that commit.

A commit message should outline what changed and why. These messages

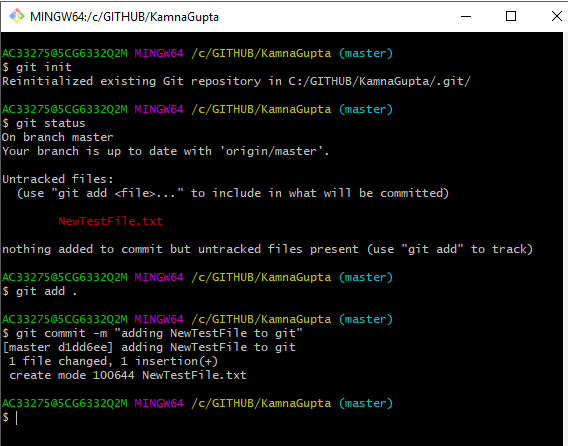
1. help collaborators and your future self understand what was changed and why
2. allow you and your collaborators to find (and undo if necessary) changes that were previously made.

If you are not committing a lot of changes, you can create a short one line commit message using the -mflag:

git commit -m "Editing the README to try out git add/commit"

git commit

Once you save your commit message and exit the text editor, the file that you created will contain your commit message.



**Push changes to GitHub**

So far we have only modified our local copy of the repository. To add the changes to your git repo files on your computer to the version of your repository on GitHub, you need to **push** them GitHub.

You can push your changes to GitHub with:

git push

