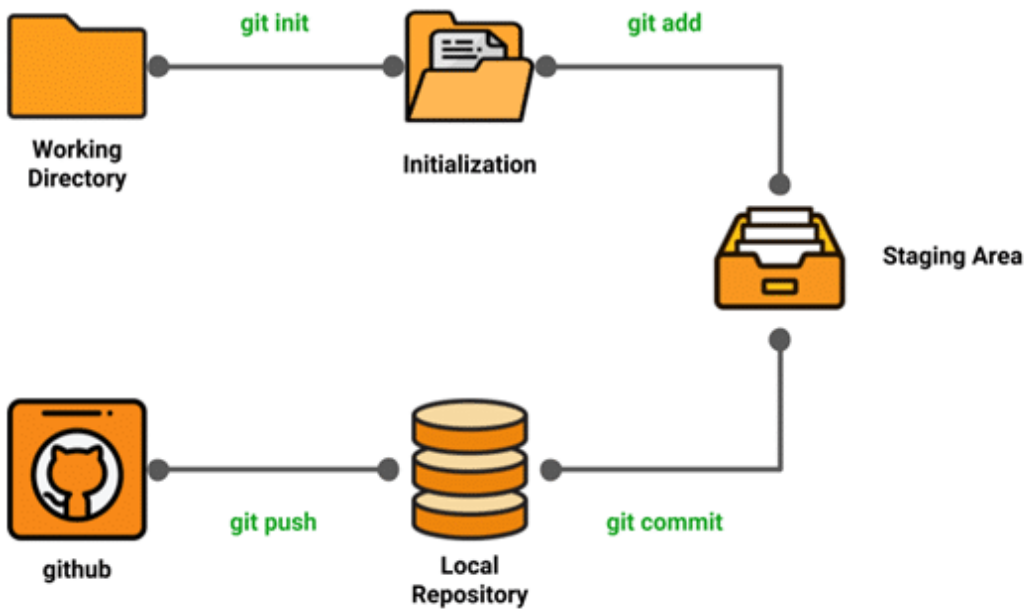
Git is a small yet very efficient version control tool. It helps both programmers and non-programmers keep track of the history of their project files by storing different versions of them.

Git helps developers keep track of the history of their code files by storing them in different versions on its own server repository, i.e., GitHub. Git has all the functionality, performance, security, and flexibility that most of the development teams and individual developers need.

Git Life Cycle

- **Local working directory**: The first stage of a Git project life cycle is the local working directory where our project resides, which may or may not be tracked.



- **Initialization:** To initialize a repository, we give the command git init. With this command, we will make Git aware of the project file in our repository.
- **Staging area:** Now that our source code files, data files, and configuration files are being tracked by Git, we will add the files that we want to commit to the staging area by the git add command. This process can also be called indexing. The index consists of files added to the staging area.

- **Commit:** Now, we will commit our files using the git commit -m ‘our message’ command.

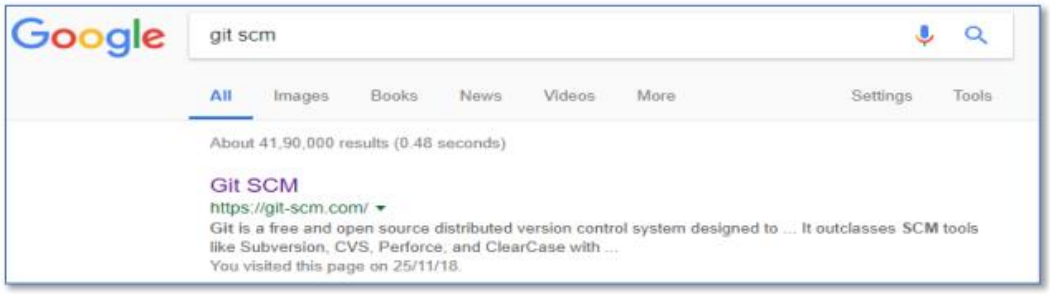
Git life cycle occurs, i.e., in GitHub, we publish our files from the local repository to the remote repository.

Installing Git

GIT INSTALLTION

1. WINDOWS:

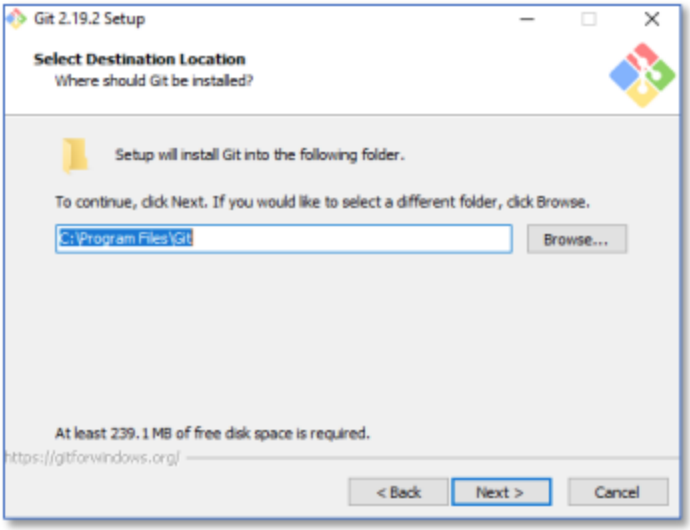
Step 1: Go to Git SCM *https://git-scm.com/*

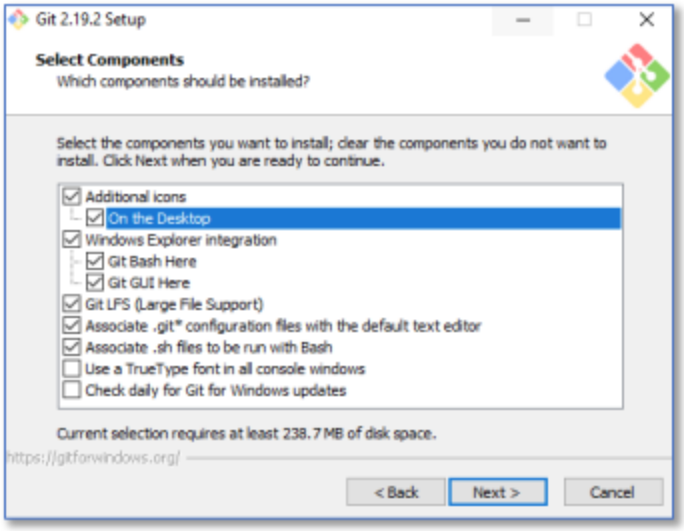


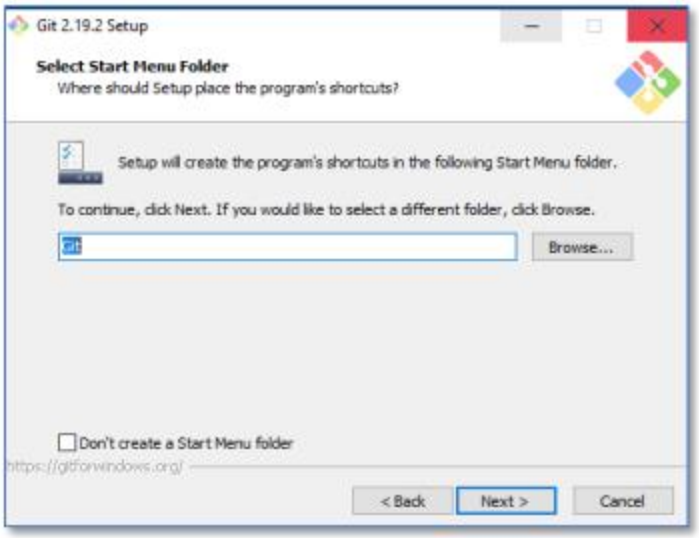
Step 2: Download the 2.19.2 for windows version of Git

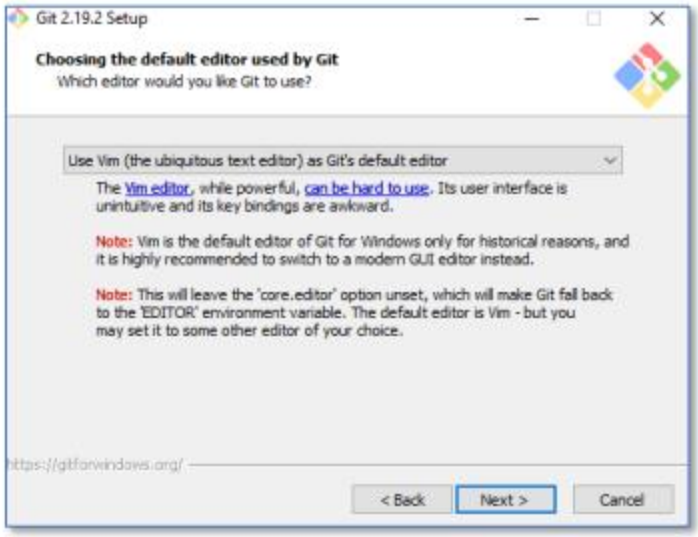
Step 3: Click on the suitable version of Git to start the downloading process.

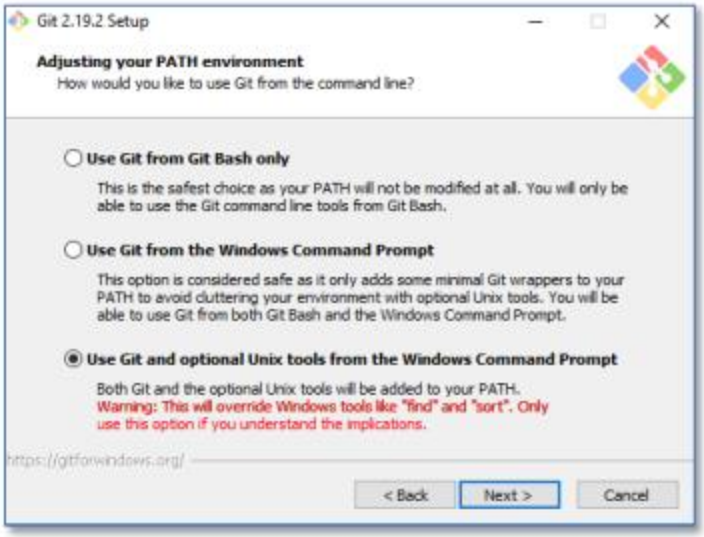
Step 4: Once the downloading is done. Follow the steps:

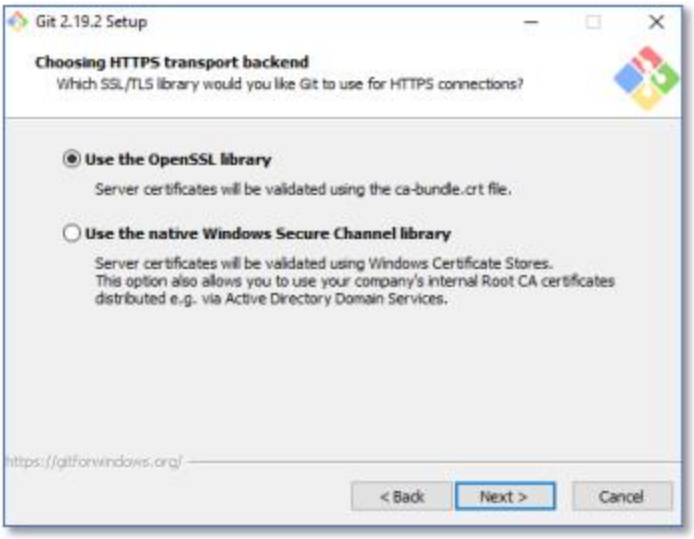

Git 2.19.2 Setup
Select Destination Location
Where should Git be installed?
Setup will install Git into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\Git
Browse...
At least 239.1 MB of free disk space is required.
https://gitforwindows.org/
< Back
Next >
Cancel

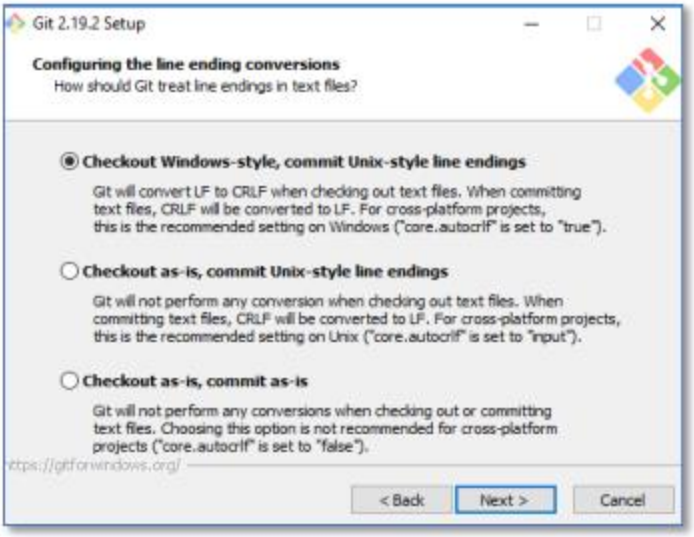

Git 2.19.2 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
On the Desktop
Windows Explorer integration
Git Bash Here
Git GUI Here
Git LFS (Large File Support)
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
Use a TrueType font in all console windows
Check daily for Git for Windows updates
Current selection requires at least 238.7 MB of disk space.
https://gitforwindows.org/
< Back
Next >
Cancel

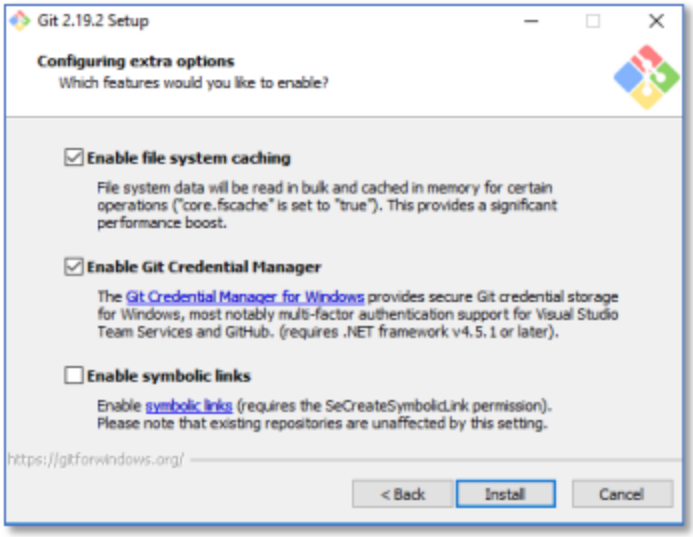

Git 2.19.2 Setup
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Browse...
Don't create a Start Menu folder
https://gitforwindows.org/
< Back
Next >
Cancel

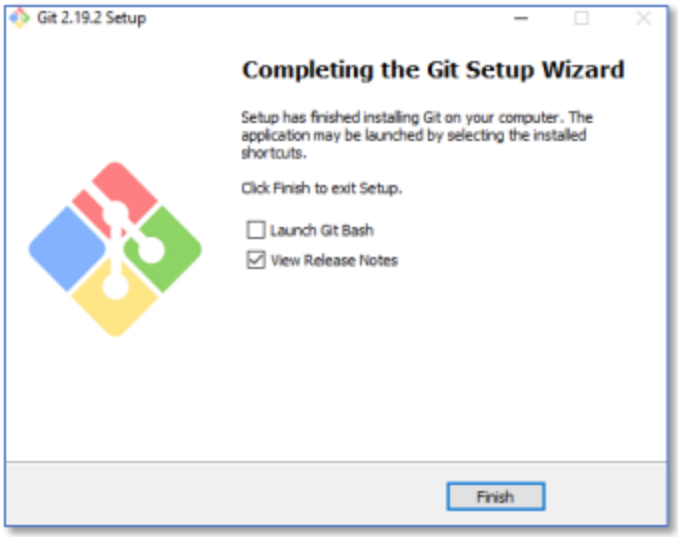

Git 2.19.2 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
Use Vim (the ubiquitous text editor) as Git's default editor
The Vim editor, while powerful, can be hard to use. Its user interface is unintuitive and its key bindings are awkward.
Note: Vim is the default editor of Git for Windows only for historical reasons, and it is highly recommended to switch to a modern GUI editor instead.
Note: This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.
https://gitforwindows.org/
< Back
Next >
Cancel


Git 2.19.2 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the safest choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.
Use Git from the Windows Command Prompt
This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from both Git Bash and the Windows Command Prompt.
Use Git and optional Unix tools from the Windows Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.
https://gitforwindows.org/
< Back
Next >
Cancel


Git 2.19.2 Setup
Choosing HTTPS transport backend
Which SSL/TLS library would you like Git to use for HTTPS connections?
Use the OpenSSL library
Server certificates will be validated using the ca-bundle.crt file.
Use the native Windows Secure Channel library
Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates
distributed e.g. via Active Directory Domain Services.
https://gitforwindows.org/
< Back
Next >
Cancel


Git 2.19.2 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing
text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When
committing text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing
text files. Choosing this option is not recommended for cross-platform
projects ("core.autocrlf" is set to "false").
https://gitforwindows.org/
< Back
Next >
Cancel





Congratulations! You have successfully installed git on your Windows system.

Git Commands

Git is a free, open-source distributed version control system tool designed to handle small to very large projects with speed and efficiency. It has steadily grown from just being a preferred skill to a must-have skill for multiple job roles today.

Here are the top 18 Git commands list:

- git init
- git add
- git commit
- git status
- git remote
- git push
- git clone
- git branch
- git checkout
- git log
- git stash
- git revert
- git diff
- git merge
- git rebase
- git fetch
- git reset
- git pull

GIT INIT:

Usage: git init [repository name]

We have to navigate to our project directory and type the command **git init** to initialize a Git repository for our local project folder. Git will create a hidden **.git** directory and use it for keeping its files organized in other subdirectories.

GIT CONFIG:

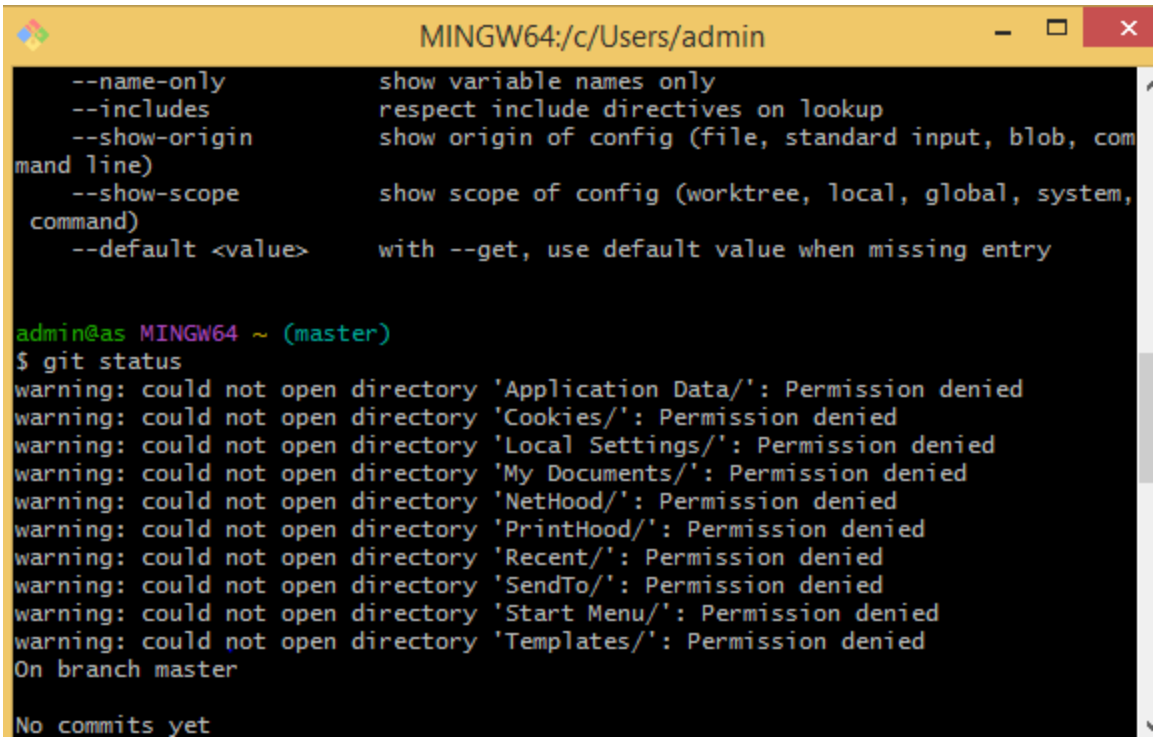
- The git config command is used initially to configure the user.name and user.email. This specifies what email id and username will be used from a local repository.
- When git config is used with --global flag, it writes the settings to all repositories on the compute

```
admin@as MINGW64 ~ (master)
$ git config
usage: git config [<options>]

Config file location
    --global              use global config file
    --system              use system config file
    --local               use repository config file
    --worktree            use per-worktree config file
    -f, --file <file>     use given config file
    --blob <blob-id>      read config from given blob object

Action
    --get                 get value: name [value-pattern]
    --get-all             get all values: key [value-pattern]
    --get-regexp          get values for regexp: name-regex [value-pattern]
    --get-urlmatch        get value specific for the URL: section[.var] URL
    --replace-all         replace all matching variables: name value [value-patt
ern]
    --add                 add a new variable: name value
    --unset               remove a variable: name [value-pattern]
    --unset-all           remove all matches: name [value-pattern]
    --rename-section      rename section: old-name new-name
```

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
```



For any help, use the following command:

GIT HELP CONFIG:

$ 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

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .bash_history
        .eclipse/
        .idlerc/
        .openjfx/
        .p2/
        AppData/
        Contacts/
        Desktop/
        Documents/
        Downloads/
        Favorites/
        IntelGraphicsProfiles/
        Links/
        Music/
        NTUSER.DAT
        NTUSER.DAT{bbed3e3a-0b41-11e3-8249-d6927d06400b}.TxR.0.regtrans-ms
        NTUSER.DAT{bbed3e3a-0b41-11e3-8249-d6927d06400b}.TxR.1.regtrans-ms
        NTUSER.DAT{bbed3e3a-0b41-11e3-8249-d6927d06400b}.TxR.2.regtrans-ms
        NTUSER.DAT{bbed3e3a-0b41-11e3-8249-d6927d06400b}.TxR.blf
        NTUSER.DAT{bbed3e3b-0b41-11e3-8249-d6927d06400b}.TM.blf
```

Create a local directory using the following command:

MK DIR TEST:

$ mkdir test

```
        ntuser.ini
        test/

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

admin@as MINGW64 ~ (master)
$ git help config

admin@as MINGW64 ~ (master)
$  git config --help

admin@as MINGW64 ~ (master)
$ $ git config --help
bash: $: command not found

admin@as MINGW64 ~ (master)
$ $ git config --help
bash: $: command not found

admin@as MINGW64 ~ (master)
$ mkdir demo

admin@as MINGW64 ~ (master)
$ |
```

CD TEST:

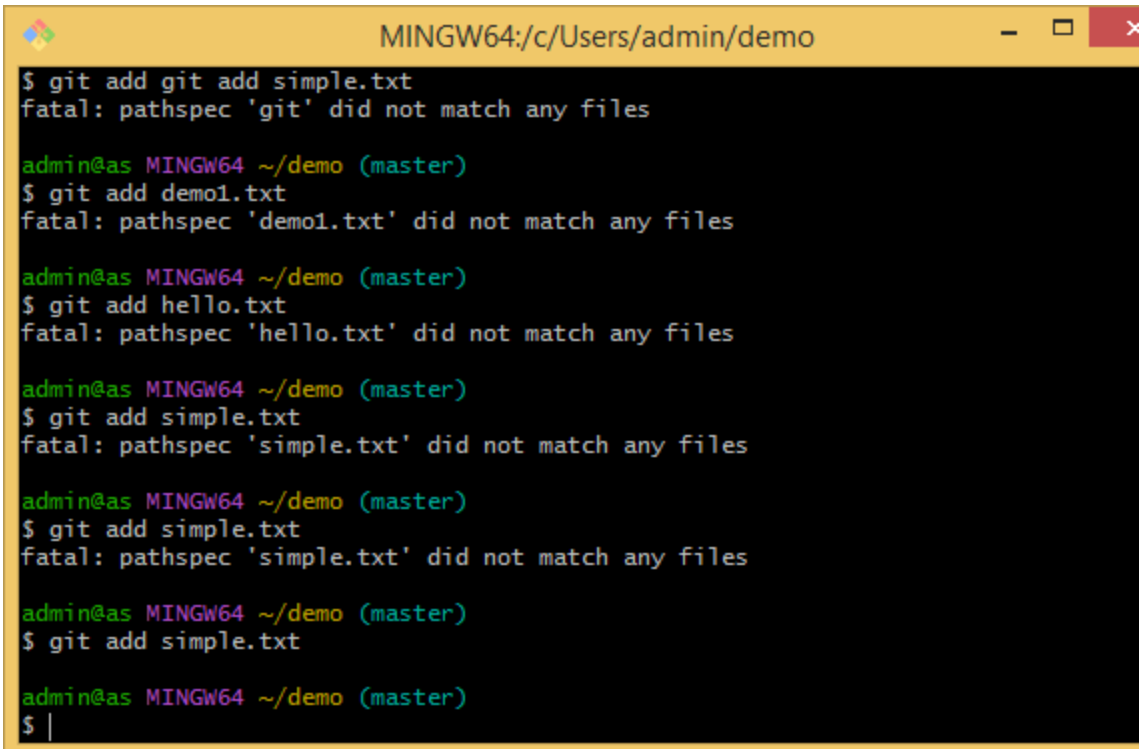
$ cd test

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

GIT ADD TEXT FILE:

$ git add demo.txt

The git add command is used to add file contents to the Index (Staging Area).This command updates the current content of the working tree to the staging area.



GIT COMMITTING A TEXT FILE:

Next, make a commit using the following command:

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

Link the Git to a Github Account: