Once git is installed, we need to configure our identity

1. Open Bash and type git.. it should show the help files. If it shows the help files, git is installed properly. If not, there is some issue with the installation and needs to be fixed
2. Configuring Git
   1. git config –global user.name “Siva Chandrasekar” – Updating metadata regarding the owner name
   2. git config –global user.email “sivacharansrc@gmail.com” – Updating metadata regarding the owner email
   3. git config --global core.autocrlf true – This is to tell the git on line ending behavior for windows machine
   4. git config --global core.autocrlf input – This is to tell the git on line ending behavior for mac machine
   5. git config --global core.editor "vim" – Explicitly configure git to use vim editor for editing
   6. git config –list – list all the config properties for git config
   7. git add “batman.txt” – this is to add tracking for the batman.txt file
   8. git add -A – this is to add all the files that were changed in the current directory for tracking
   9. git add . – add everything under tracking
   10. git status – view the status of any tracked / untracked files in the present working directory
   11. git log – To see the list of entries done in the past. This gives information what were the changes, when the changes were done, and who made those changes..
   12. git commit -am “added new file and made changes” – the “a” flag is to automatically add all the files that has undergone some changes, and commit those. Once the file is tracked initially, it is not required to use git add each time the changes are made, if we use -am with commit
   13. git diff “filename” – This is to view the differences between the current version and previous version
   14. git diff HEAD “filename” – very similar to above command
   15. git diff HEAD~1 “filename” – compare current version to 2 versions previous command
   16. git diff HEAD~1 HEAD~2 “filename” – compare previous version with 2 versions previous
   17. git diff 11f0ad82 3114741b “filename.txt” – compare different versions used on the unique hash key (first 8 characters are sufficient)
   18. Ignoring Specific files from being tracked:
       1. touch .gitignore – creates a file that will eventually contain all the files that needs to be ignored
       2. vim .gitignore – once the file opens, add the name of the file that needs to be ignored (like “data2.csv” – ignores the data2.csv file, “\*.csv” – ignores all csv files) and save the file
       3. From now on, this data2 file in the working directory will never ever be tracked

**Connecting Git with Github:**

To establish a connection with Git and GitHub, SSH key has to be created from Git, and updated in the GitHub account.

**# CREATING A SSH KEY**

$ssh-keygen

$Generating public/private rsa key pair

$Enter a file in which to save the key ***(/c/Users/SSOMA/.ssh/id\_rsa***): (Press Enter to save the key without changing the file name. Note the location where it is saved)

$Enter passphrase (empty for no passphrase): (PASSPHRASE are similar to passwords. They add additional layer of security. When passphrase is entered, there will be no indication of typing. Proceed with the passphrase still)

$Enter same passphrase again: (Type the passphrase again and click enter)

**# COPYING THE SSH KEY AND ADDING TO GITHUB ACCOUNT**

$clip < /c/Users/SSOMA/.ssh/id\_rsa.pub

# Navigate to the https://github.platforms.engineering, Click on Profile Icon, Settings, SSH and GPG Keys > Add SSH Key

# Give a name for the SSH Key, and paste the SSH key that was copied to clipboard from git, and click Add Key

**# TESTING THE GITHUB CONNECTION**

#ssh -T git@github.platforms.engineering

#The authenticity of host 'github.platforms.engineering (52.201.2.23)' can't be established.

#ECDSA key fingerprint is SHA256:XXXXXXXXXXXXXXXXXXXXXXXXXX.

#Are you sure you want to continue connecting (yes/no)? (Type yes and press Enter)

#Warning: Permanently added 'github.platforms.engineering,52.201.2.23' (ECDSA) to the list of known hosts.

#Enter passphrase for key '/c/Users/SSOMA/.ssh/id\_rsa': (Enter the passphrase created at the time of SSH Key creation)

#Hi SSOMA! You've successfully authenticated, but GitHub does not provide shell access.

**# ADD PROJECT TO GITHUB**

# Go to GitHub and create a new repository. Make sure the repository settings as below:

# Provide a name for the repository

# The Project Type is “Public”

# “Initialize this repository with README” is not checked

# Add .gitignore: None

# Create repository

# Copy the SSH address of the repository

# Open Git Editor, and navigate to the project in the local directory

# git add remote add origin git@github.platforms.engineering:SSOMA/GitHub\_Testing.git (Copy SSH Address from GitHub)

# git remote -v (This should give options for pull and push which means that the origin is added successfully)

# git push origin master (origin is the location in the remote (github), and master is the branch in the local repository)

# Enter Passphrase: (Type passphrase and click enter)

**# TO PERMANENTLY SET THE UPSTREAM WORKING BRANCH**

# git push --set-upstream origin master

# After setting the upstream branch, only **git push** is enough for pushing the changes from local to remote

**# SAVE THE PASSPHRASE IDENTITY FOR THE CURRENT**

# eval $(ssh-agent)

# ssh-add (Enter the passphrase and click enter to add the passphrase permanently)

**# PERMANENTLY DISABLE THE PASSPHRASE REQUEST**

# Create a new SSH Key without passphrase and update to GitHub account