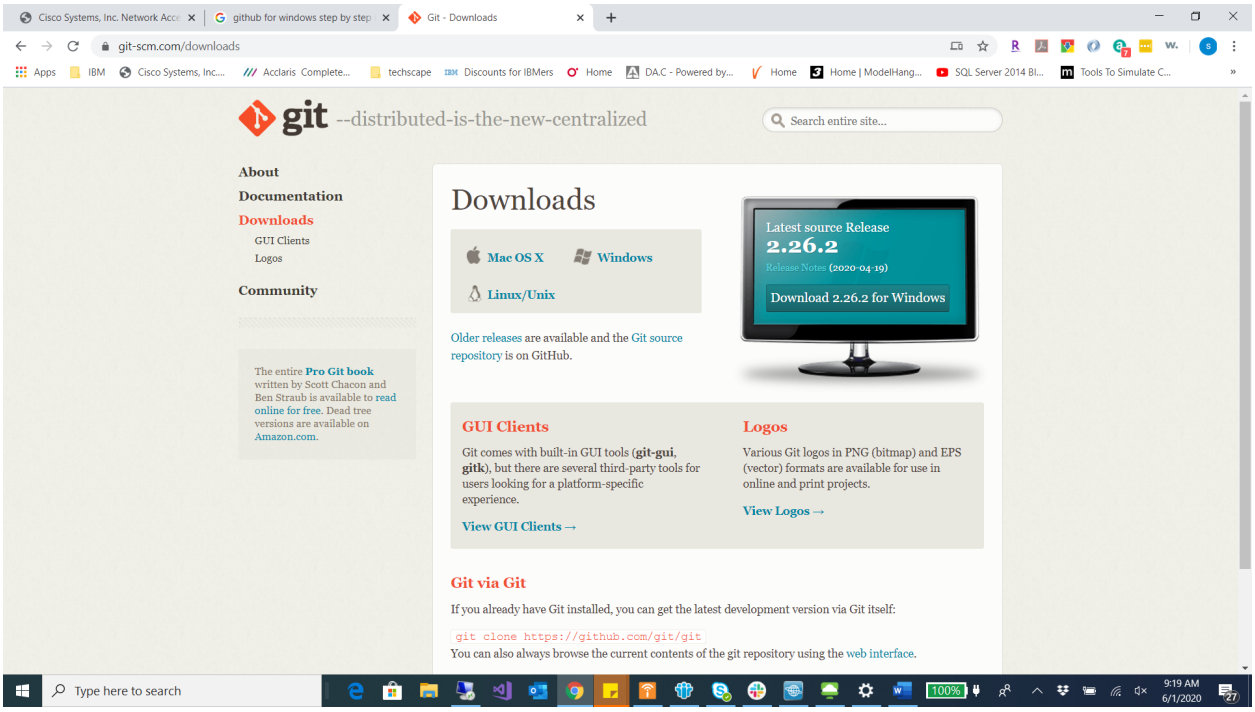
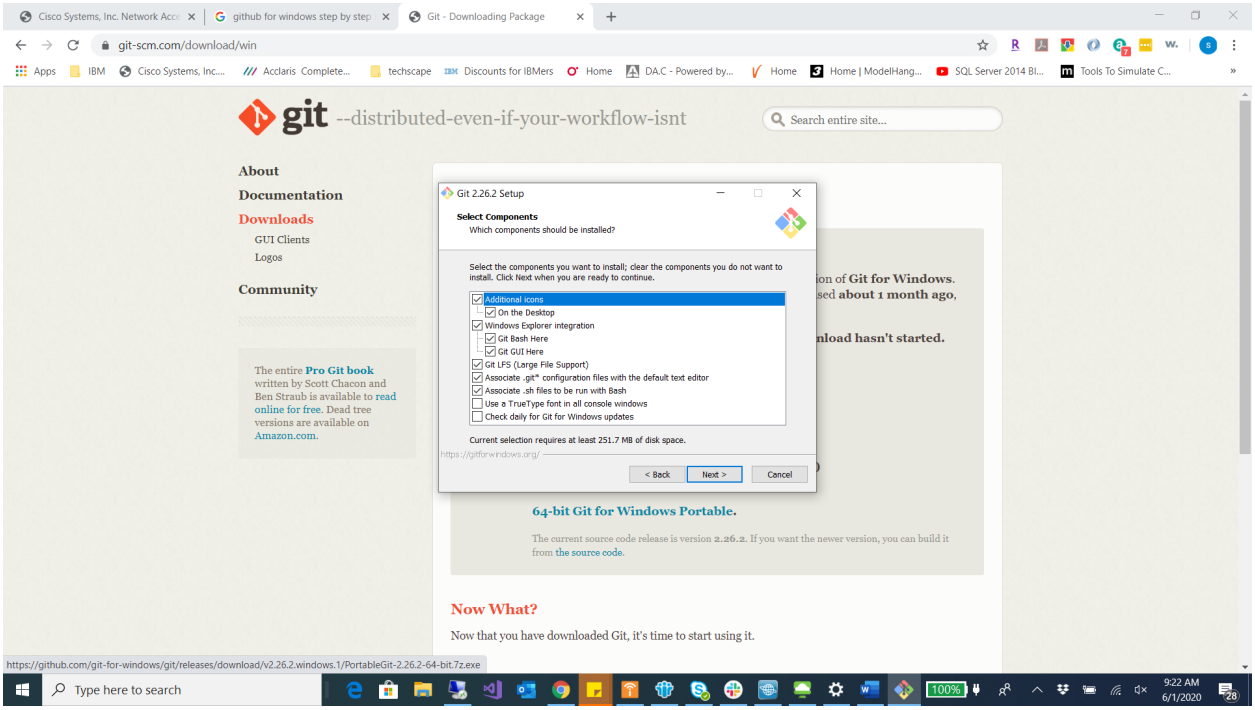
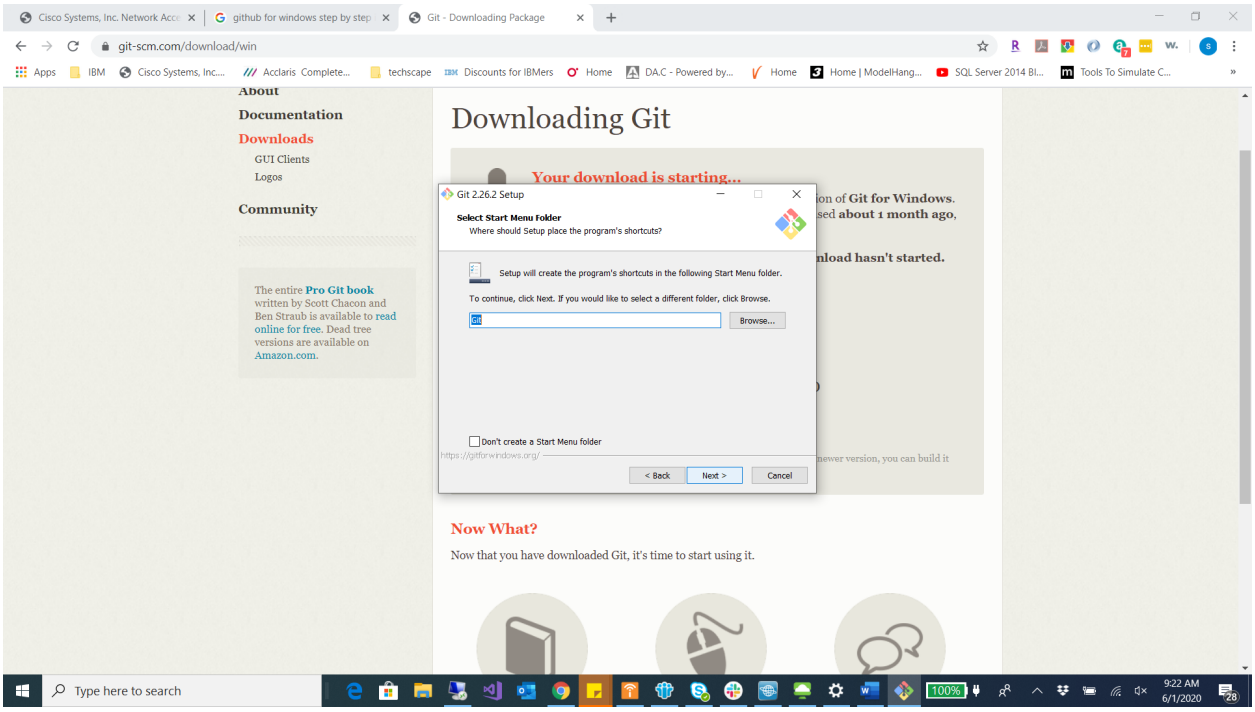
Installing GIT

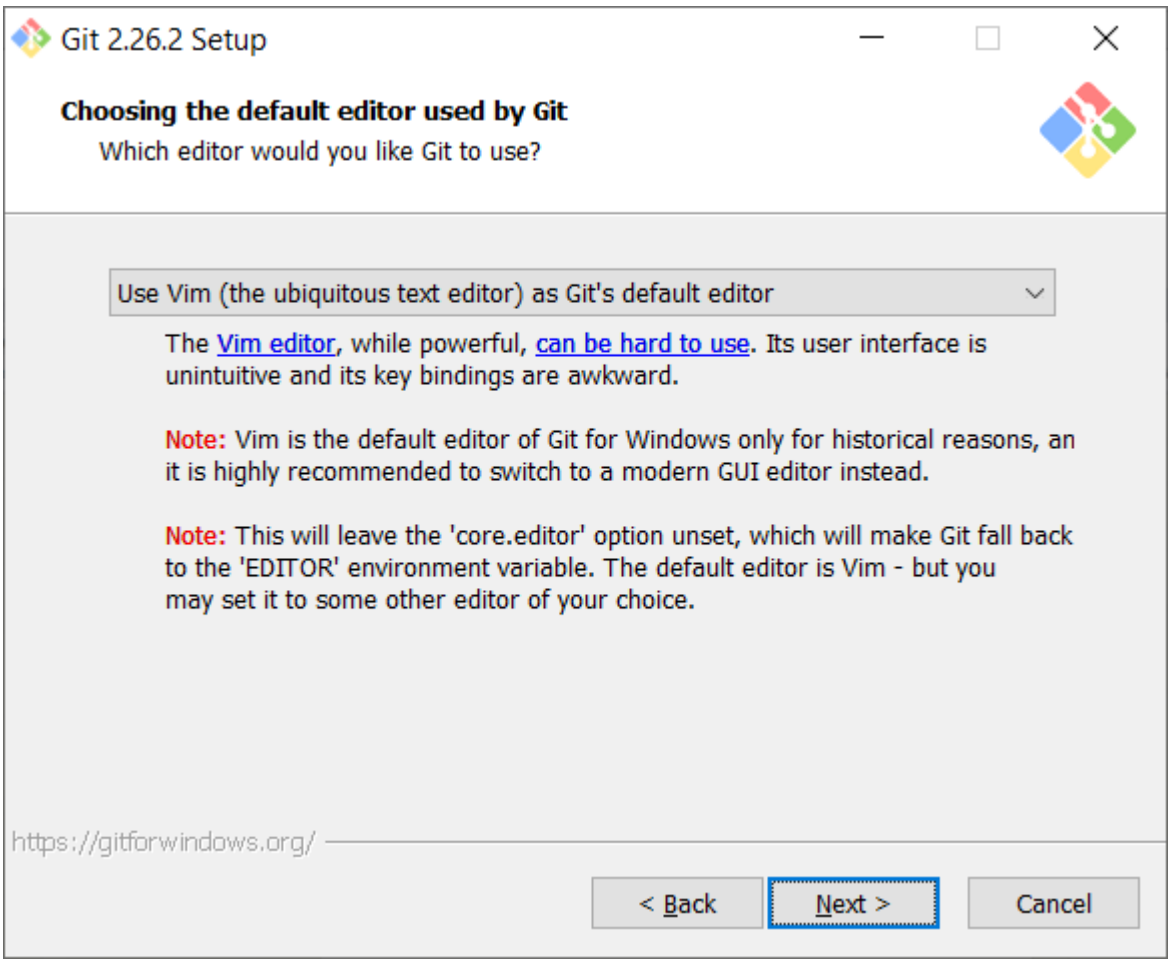
1. https://git-scm.com/downloads
2. 

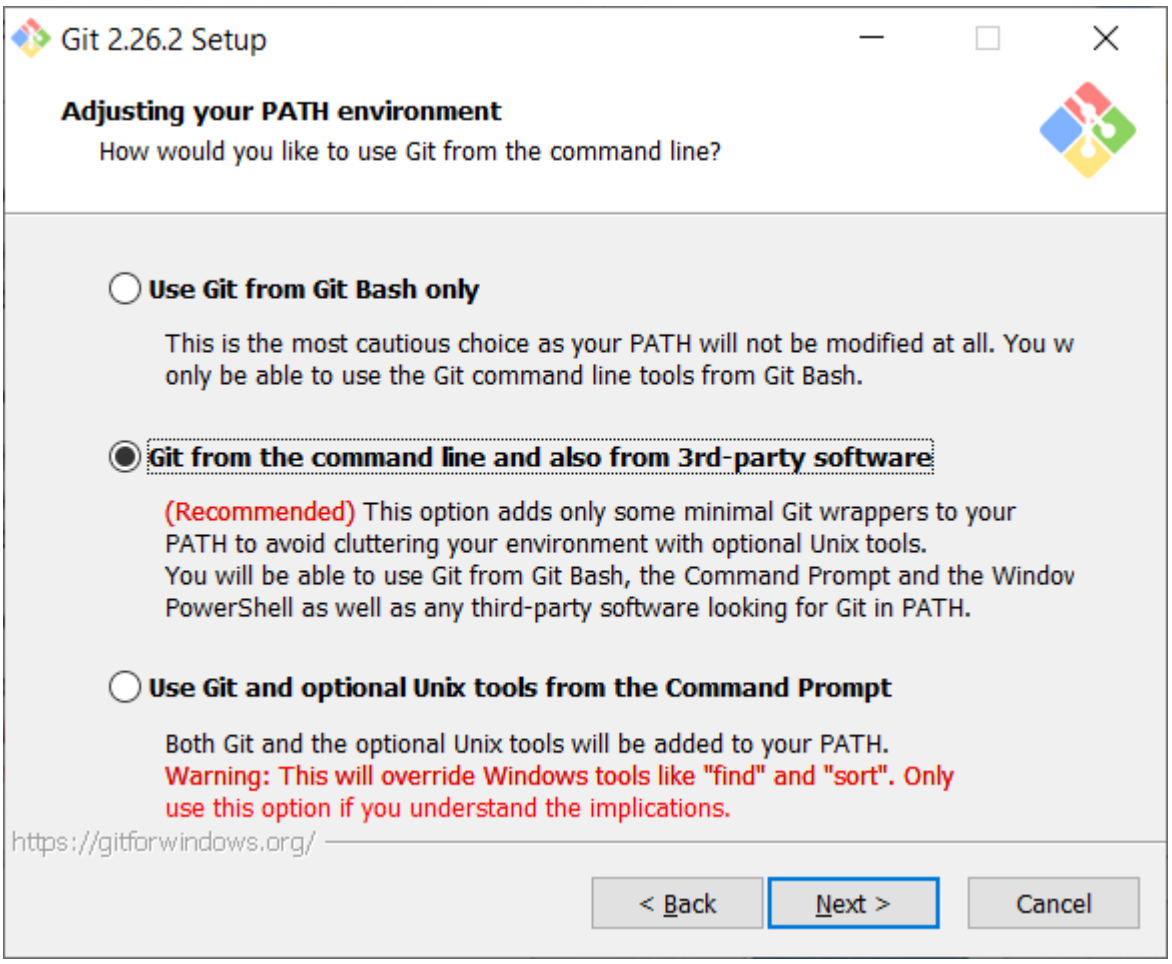
3. Open the downloaded file (choose Windows or mac os) based on the machine you have .
4. Then chose the 64 bit version or 32 bit version based on the Operating system
5. The file downloads starts. Open the file after the download has completed. It could be in your downloads folder

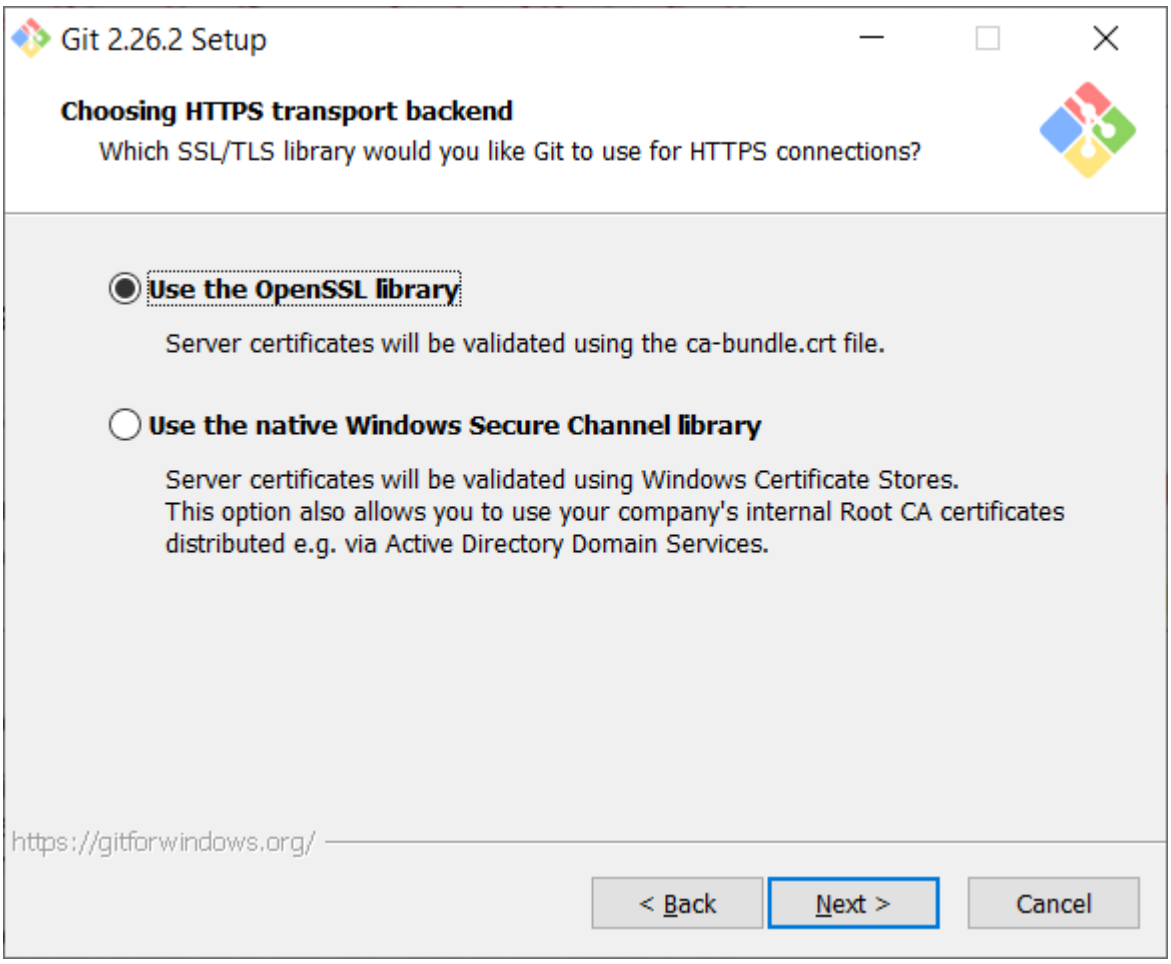

Git - Downloading Package
git-scm.com/download/win
git --distributed-even-if-your-workflow-isnt
Search entire site...
About
Documentation
Downloads
GUI Clients
Logos
Community
Git 2.26.2 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the 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 251.7 MB of disk space.
< Back
Next >
Cancel
64-bit Git for Windows Portable.
Now What?
Now that you have downloaded Git, it's time to start using it.
Type here to search

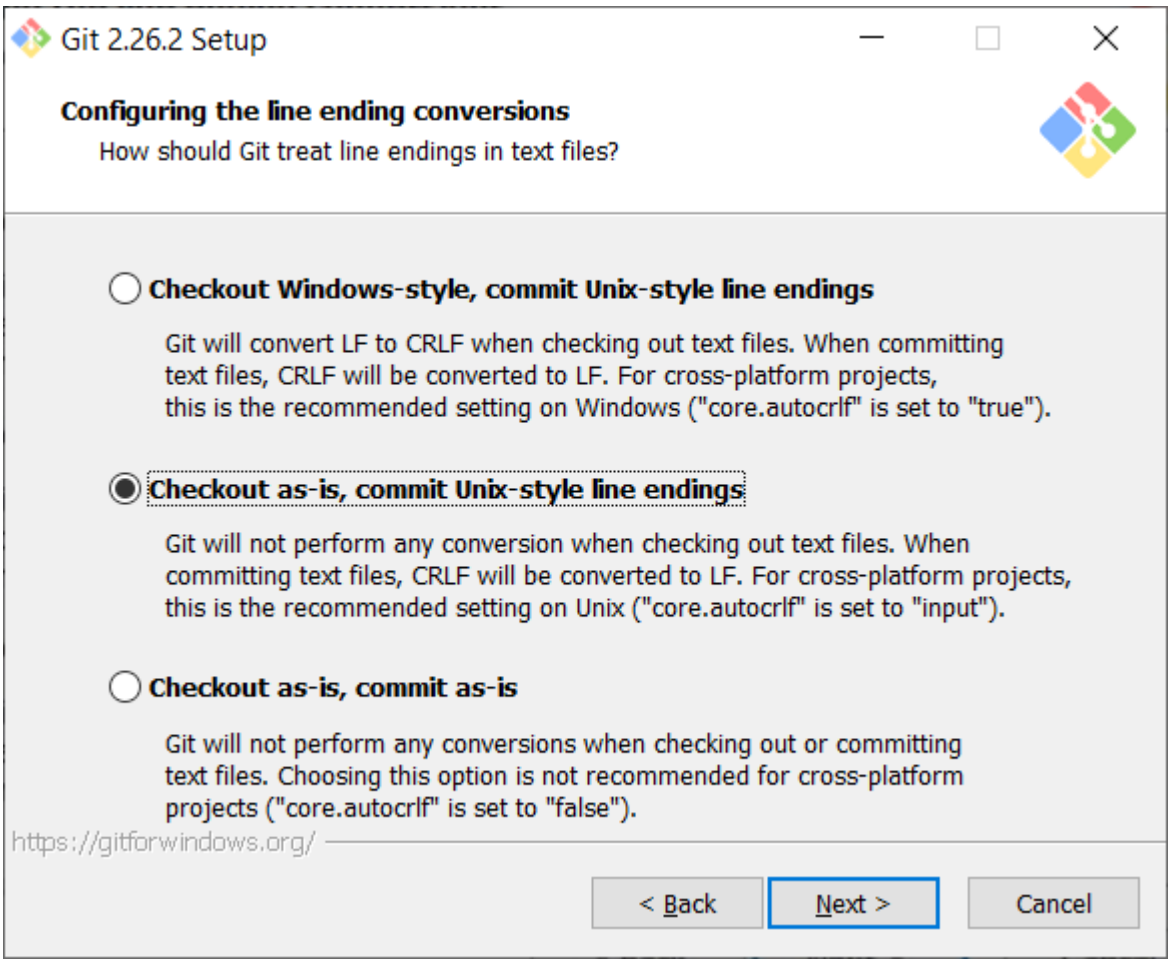
6. Hit Next

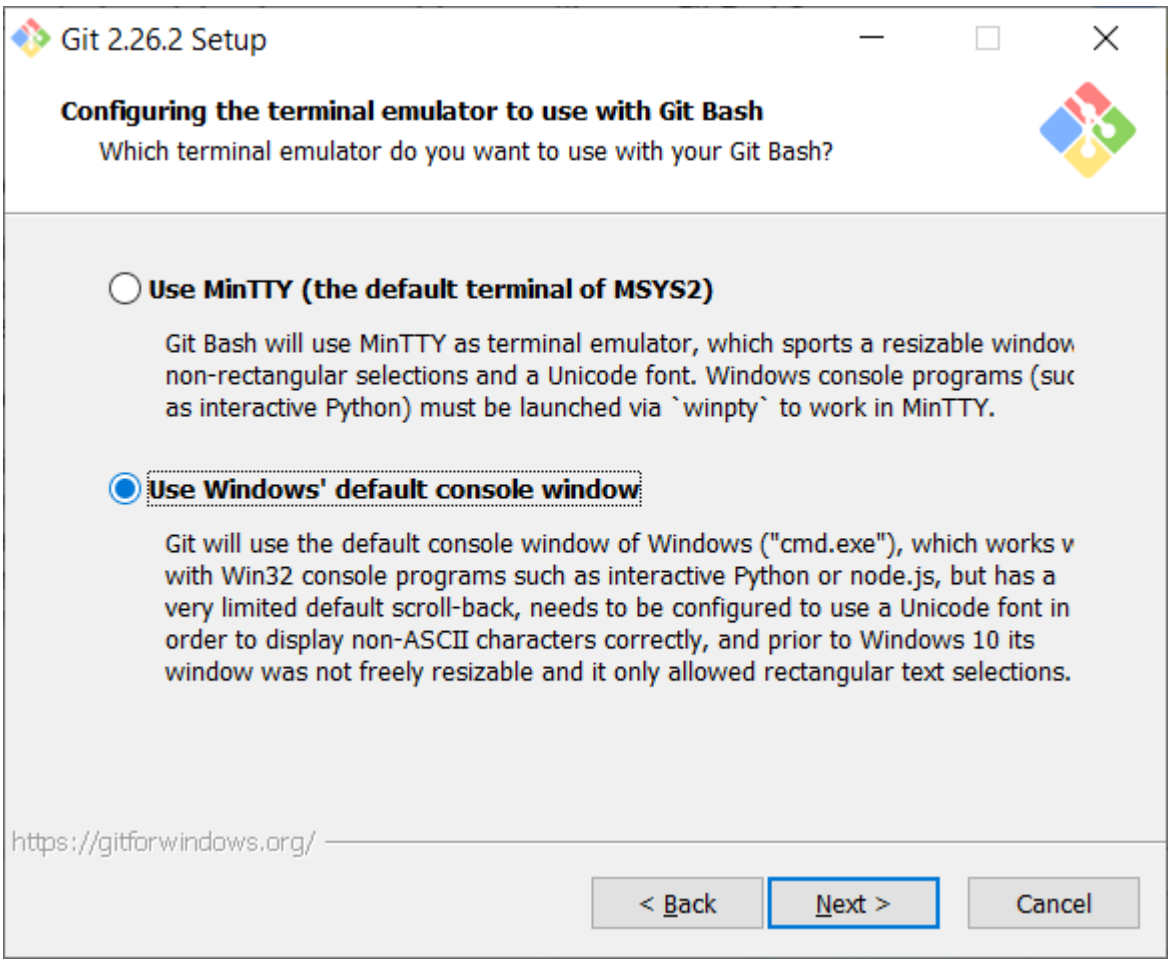

Git - Downloading Package
git-scm.com/download/win
About
Documentation
Downloads
GUI Clients
Logos
Community
Downloading Git
Your download is starting...
Git 2.26.2 Setup
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Git
Browse...
Don't create a Start Menu folder
< Back
Next >
Cancel
Now What?
Now that you have downloaded Git, it's time to start using it.
Type here to search


Git 2.26.2 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, an 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

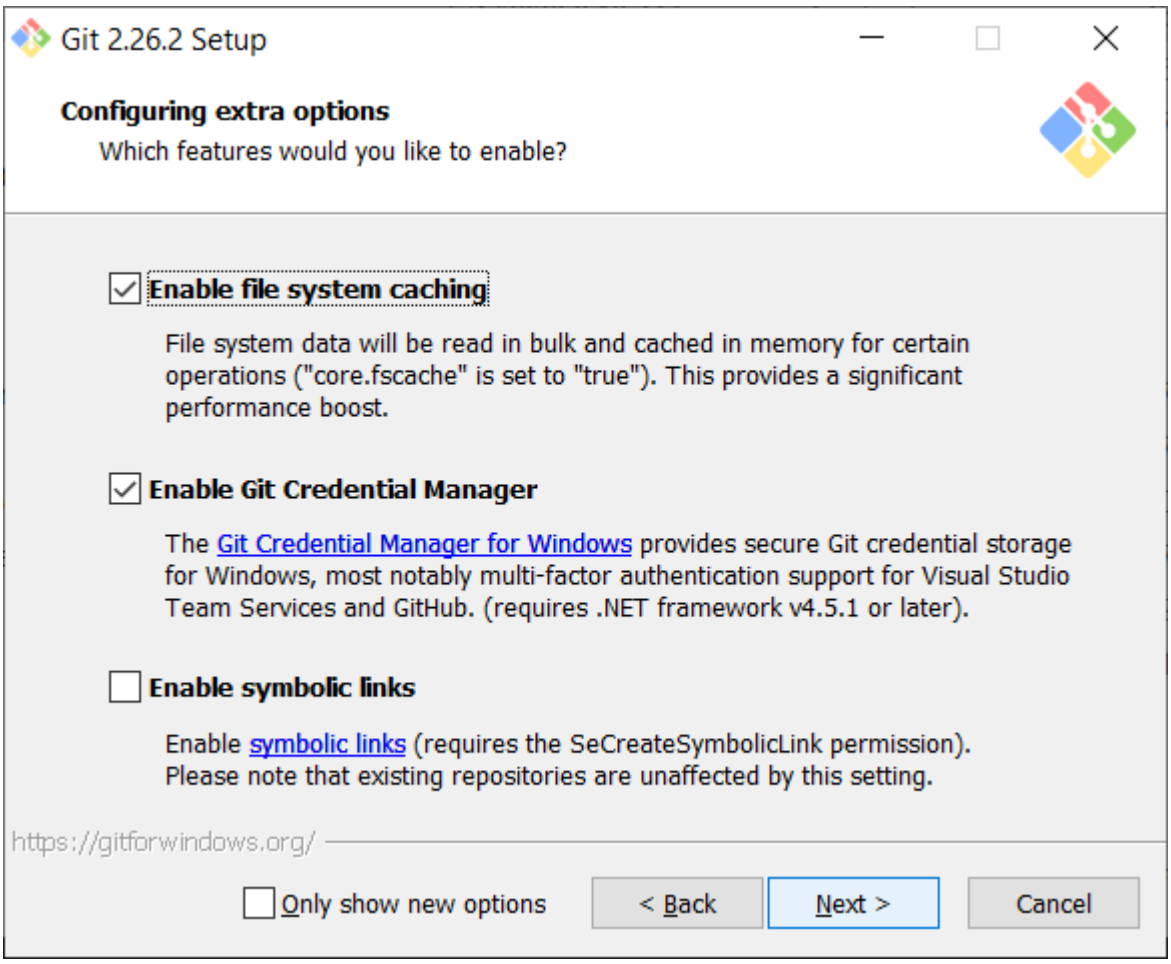

Git 2.26.2 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You w
only be able to use the Git command line tools from Git Bash.
Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your
PATH to avoid cluttering your environment with optional Unix tools.
You will be able to use Git from Git Bash, the Command Prompt and the Windov
PowerShell as well as any third-party software looking for Git in PATH.
Use Git and optional Unix tools from the 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 2.26.2 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 2.26.2 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


Git 2.26.2 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable windo
non-rectangular selections and a Unicode font. Windows console programs (suc
as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works v
with Win32 console programs such as interactive Python or node.js, but has a
very limited default scroll-back, needs to be configured to use a Unicode font in
order to display non-ASCII characters correctly, and prior to Windows 10 its
window was not freely resizable and it only allowed rectangular text selections.
https://gitforwindows.org/
< Back
Next >
Cancel

Hit next and



Hit next to Install

SSH KEYS

Generating SSH keys allows developers to interface with certain remote services without having to constantly type out login information. You're going to set up an SSH key for GitHub.

Without a key, you won't be able to push your code to GitHub without entering a password each time; trust us, that would be as irritating as needing a key to open every door in your home.

1. If you haven't signed up for a GitHub account yet, you'll need to do so before moving on with these steps. Visit https://github.com.

2. Open up Bash.

3. We need to set up SSH keys. First, let's make sure you don't already have a set of keys on your computer. Type this into your Bash window(copying and pasting will not work):

=> ls –al ~/.ssh

If no keys pop up, move onto step 4.

If keys do pop up, check that none of them are listed under id_rsa , like in this image:

If you do find a key with a matching name, then you can either overwrite it by following steps 4 to 6, or you can use the same key in steps 10 and beyond. Be advised that you'll have to remember the password tied to your key if you decide not to overwrite it.

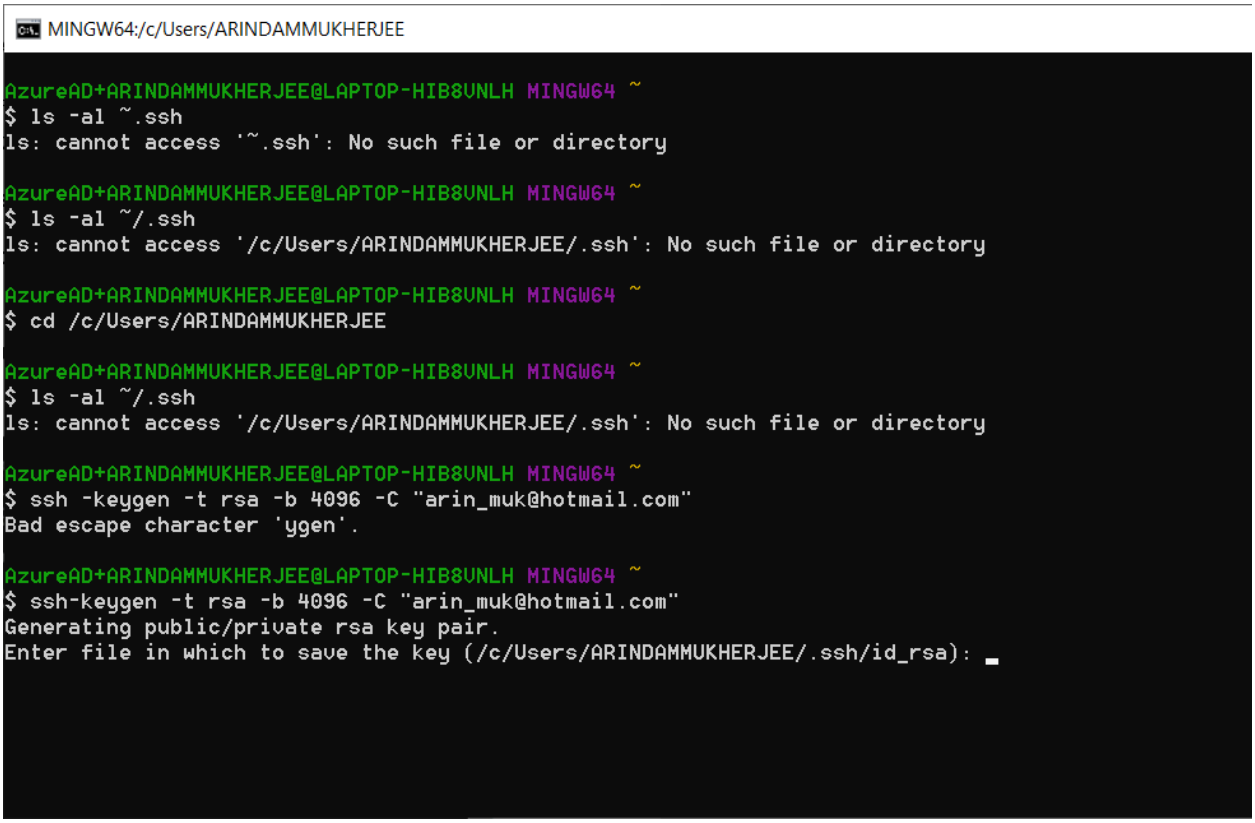
4. Type in this command along with your email to generate your keys

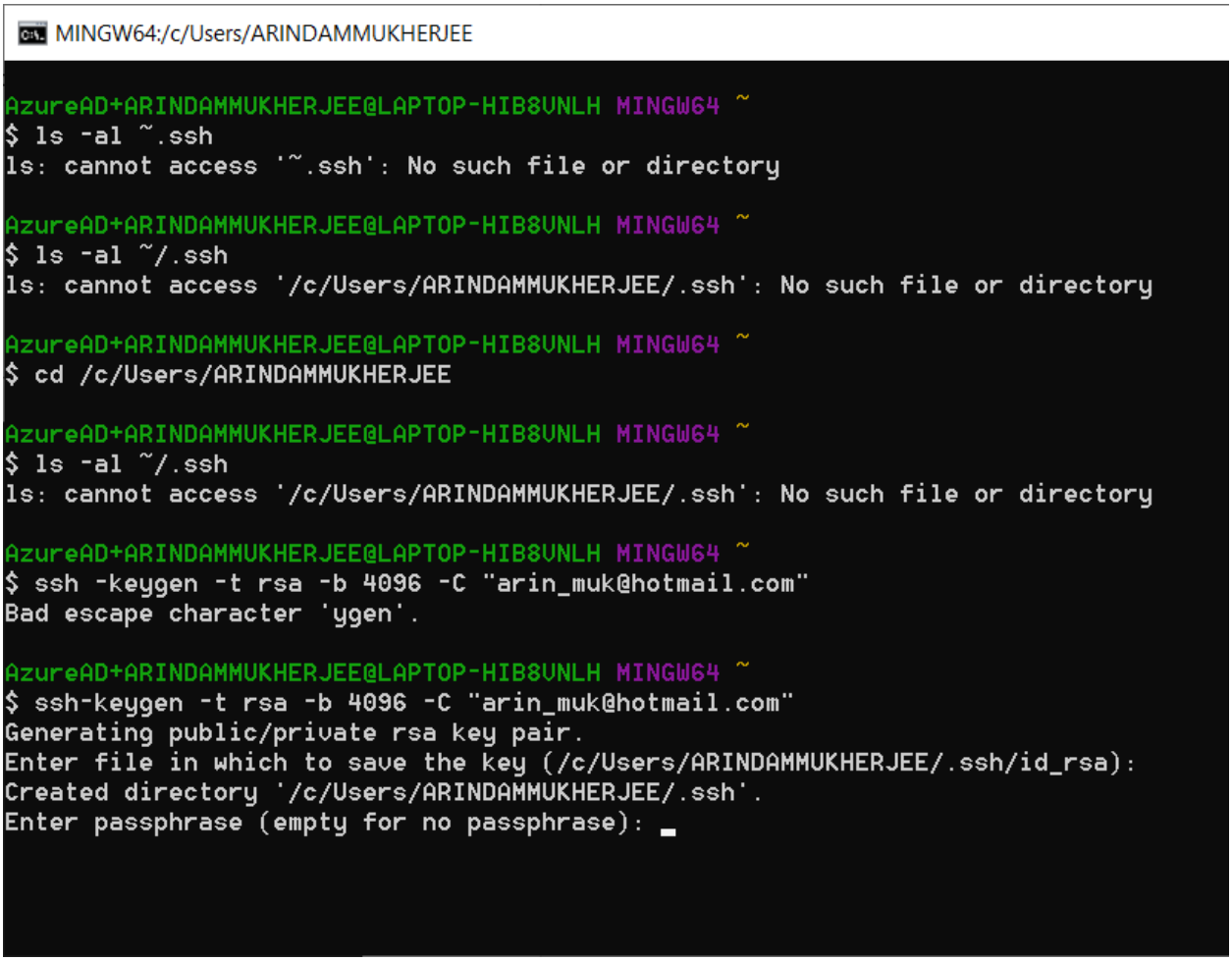
ssh-keygen –t rsa –b 4096 –C "YOURGITHUBEMAIL@PLACEHOLDER.NET"

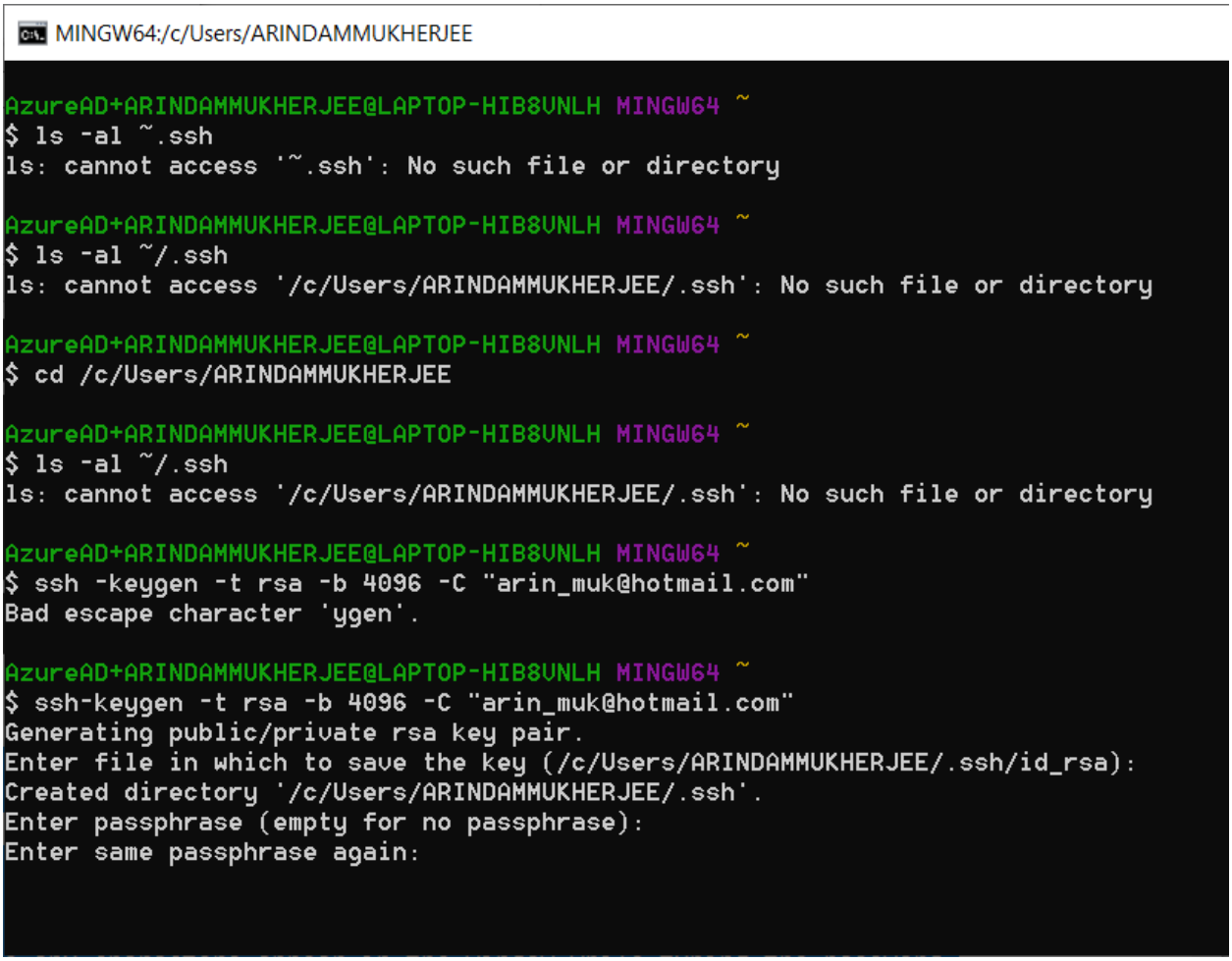
5. When asked to enter a file to save the key, just hit enter.

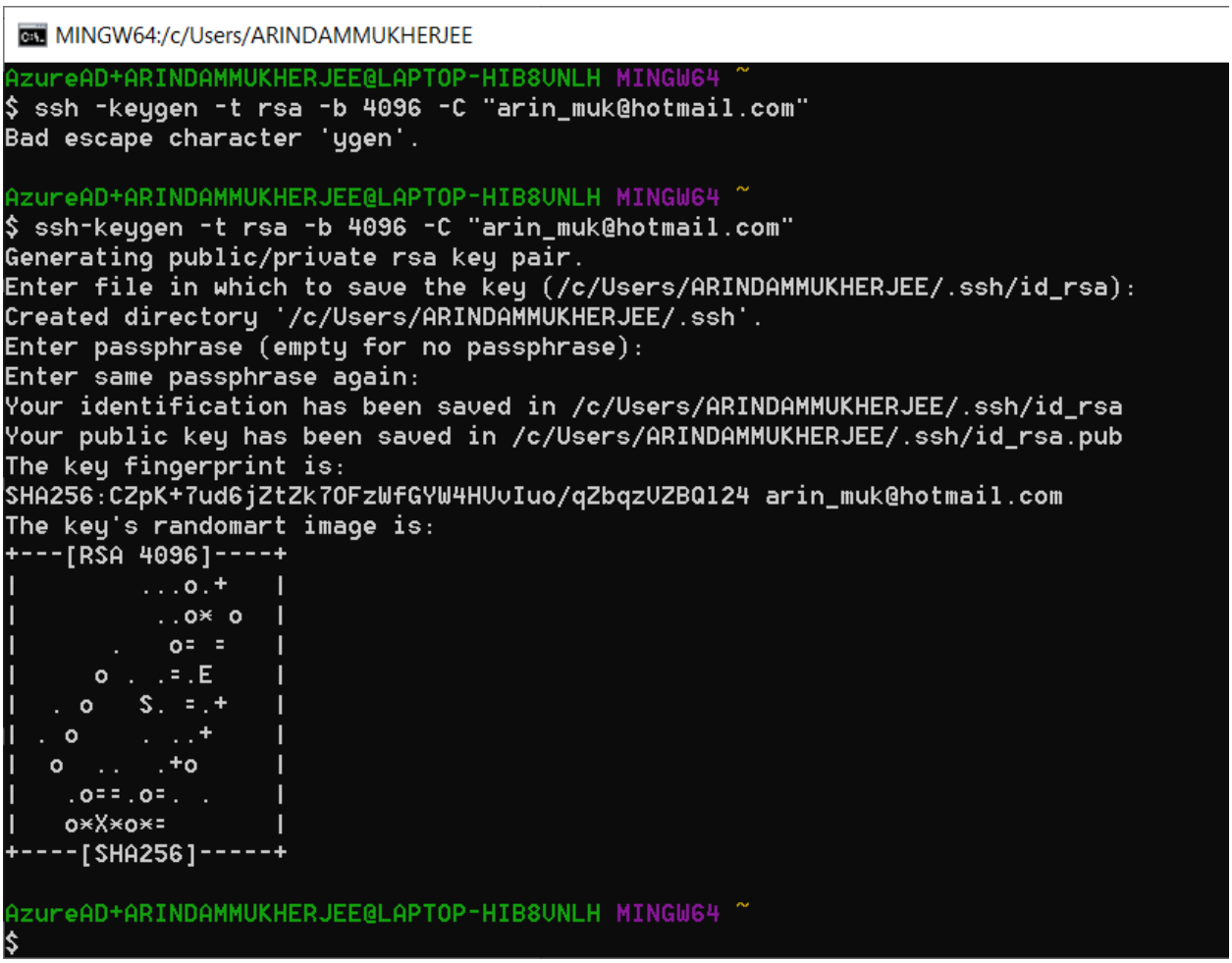
Also enter a passphrase for your key. Note: You shouldn't see any characters appear in the window while typing the password.

6. When you're finished, your window should look like this:


MINGW64:/c/Users/ARINDAMMUKHERJEE
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ls -al ~.ssh
ls: cannot access '~.ssh': No such file or directory
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ls -al ~/.ssh
ls: cannot access '/c/Users/ARINDAMMUKHERJEE/.ssh': No such file or directory
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ cd /c/Users/ARINDAMMUKHERJEE
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ls -al ~/.ssh
ls: cannot access '/c/Users/ARINDAMMUKHERJEE/.ssh': No such file or directory
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ssh -keygen -t rsa -b 4096 -C "arin_muk@hotmail.com"
Bad escape character 'ygen'.
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ssh-keygen -t rsa -b 4096 -C "arin_muk@hotmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/ARINDAMMUKHERJEE/.ssh/id_rsa):


MINGW64:/c/Users/ARINDAMMUKHERJEE
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ls -al ~.ssh
ls: cannot access '~.ssh': No such file or directory
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ls -al ~/.ssh
ls: cannot access '/c/Users/ARINDAMMUKHERJEE/.ssh': No such file or directory
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ cd /c/Users/ARINDAMMUKHERJEE
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ls -al ~/.ssh
ls: cannot access '/c/Users/ARINDAMMUKHERJEE/.ssh': No such file or directory
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ssh -keygen -t rsa -b 4096 -C "arin_muk@hotmail.com"
Bad escape character 'ygen'.
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ssh-keygen -t rsa -b 4096 -C "arin_muk@hotmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/ARINDAMMUKHERJEE/.ssh/id_rsa):
Created directory '/c/Users/ARINDAMMUKHERJEE/.ssh'.
Enter passphrase (empty for no passphrase):

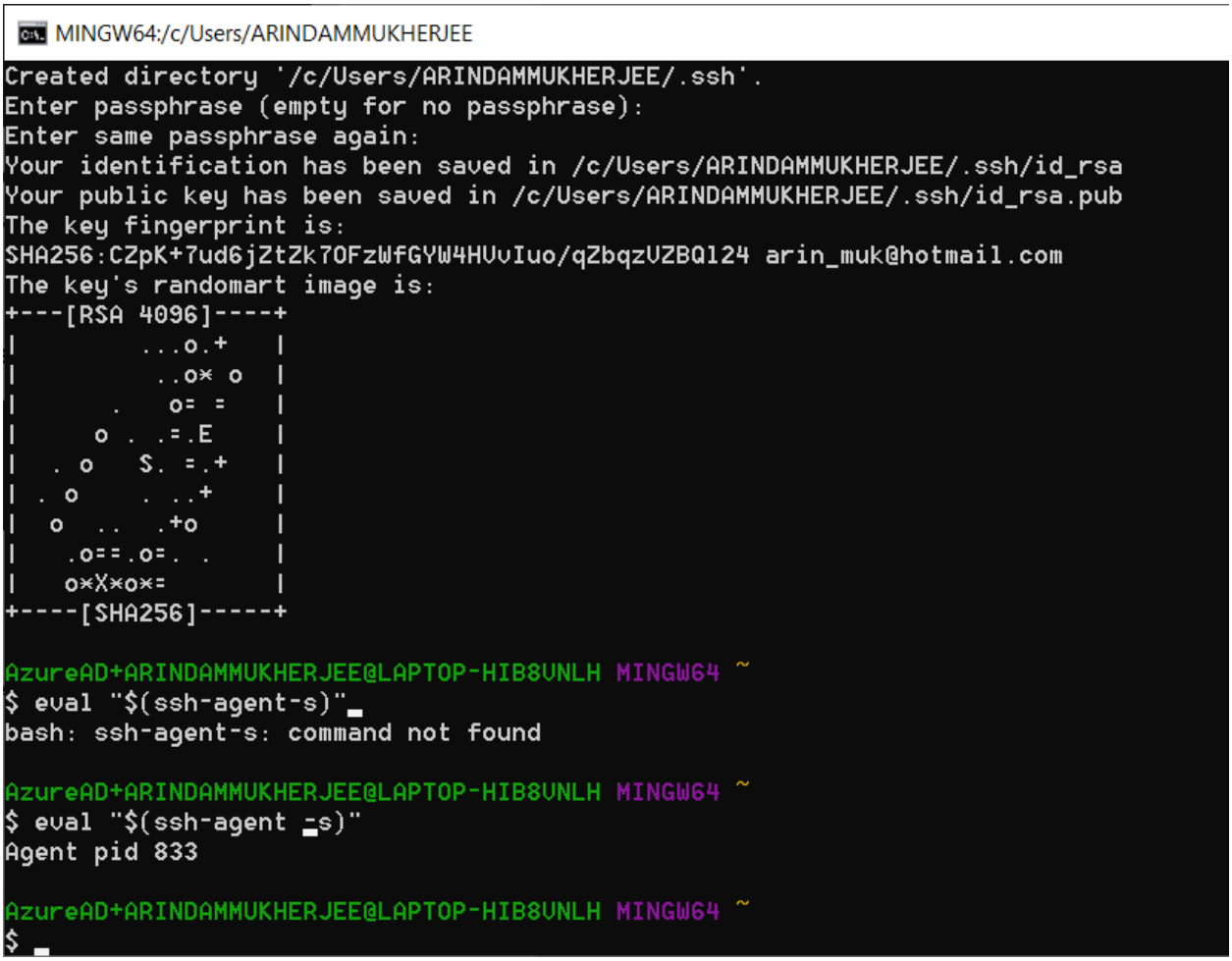

MINGW64:/c/Users/ARINDAMMUKHERJEE
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ls -al ~.ssh
ls: cannot access '~.ssh': No such file or directory
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ls -al ~/.ssh
ls: cannot access '/c/Users/ARINDAMMUKHERJEE/.ssh': No such file or directory
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ cd /c/Users/ARINDAMMUKHERJEE
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ls -al ~/.ssh
ls: cannot access '/c/Users/ARINDAMMUKHERJEE/.ssh': No such file or directory
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ssh -keygen -t rsa -b 4096 -C "arin_muk@hotmail.com"
Bad escape character 'ygen'.
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ ssh-keygen -t rsa -b 4096 -C "arin_muk@hotmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/ARINDAMMUKHERJEE/.ssh/id_rsa):
Created directory '/c/Users/ARINDAMMUKHERJEE/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:

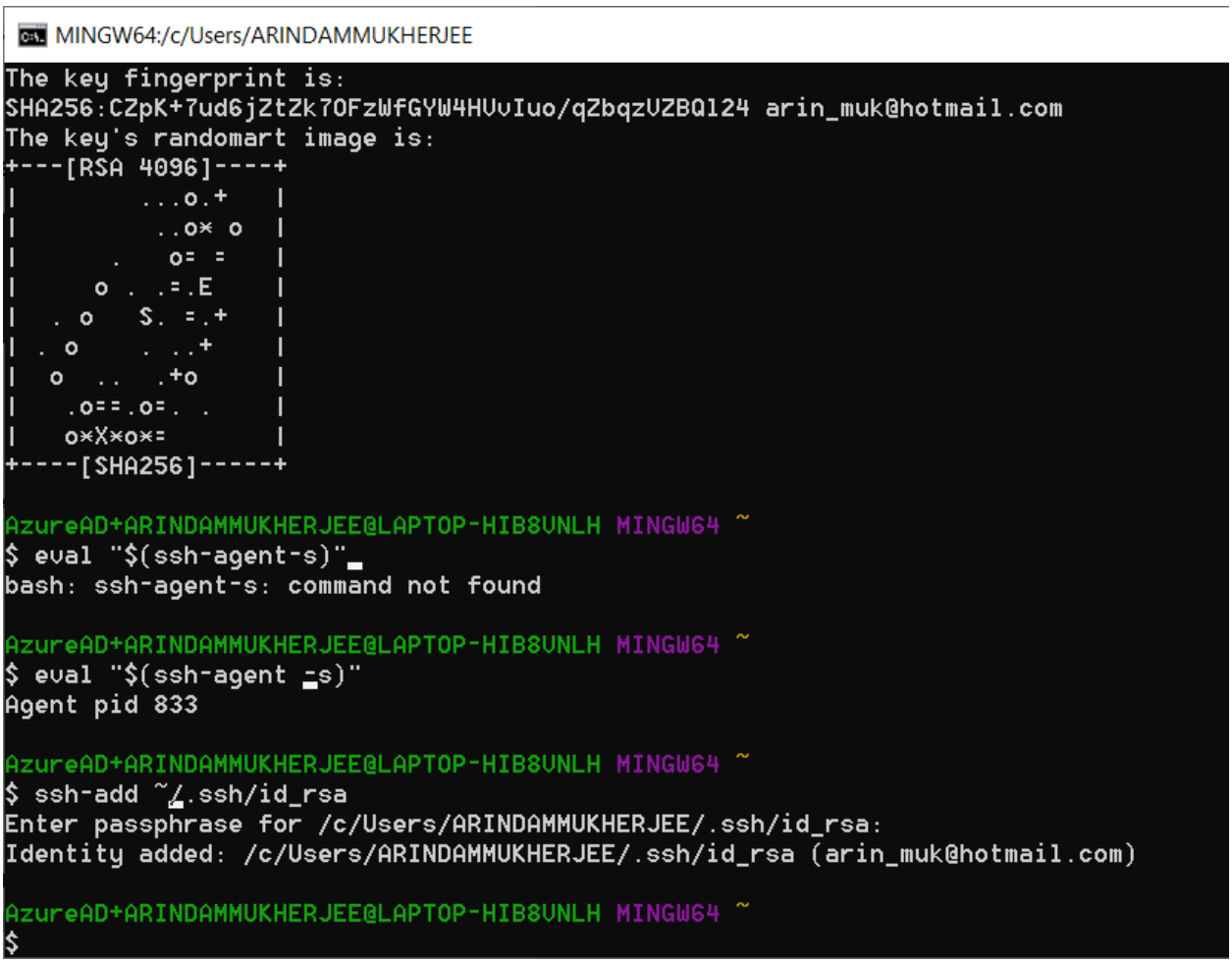


For the next step, we need to use a tool called ssh agent to link our key with our machine. Let's test whether ssh-agent is working. Run this command in Bash:

eval "$(ssh-agent –s)"

If your Bash window looks like the below image, move onto the next step.


MINGW64:/c/Users/ARINDAMMUKHERJEE
Created directory '/c/Users/ARINDAMMUKHERJEE/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/ARINDAMMUKHERJEE/.ssh/id_rsa
Your public key has been saved in /c/Users/ARINDAMMUKHERJEE/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:CZpK+7ud6jZtZk7OFzWfGYW4HUvIuo/qZbqzUZBQ124 arin_muk@hotmail.com
The key's randomart image is:
+---[RSA 4096]----+
| ...o.+ |
| ..o* o |
| .   o= = |
| o . .=.E |
| . o $. =.+ |
| . o   . ..+ |
| o ..   .+o |
| .o=.o=. . |
| o*X*o*= |
+----[SHA256]-----+
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ eval "$(ssh-agent-s)"
bash: ssh-agent-s: command not found
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$ eval "$(ssh-agent -s)"
Agent pid 833
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 ~
$



Now run this command:

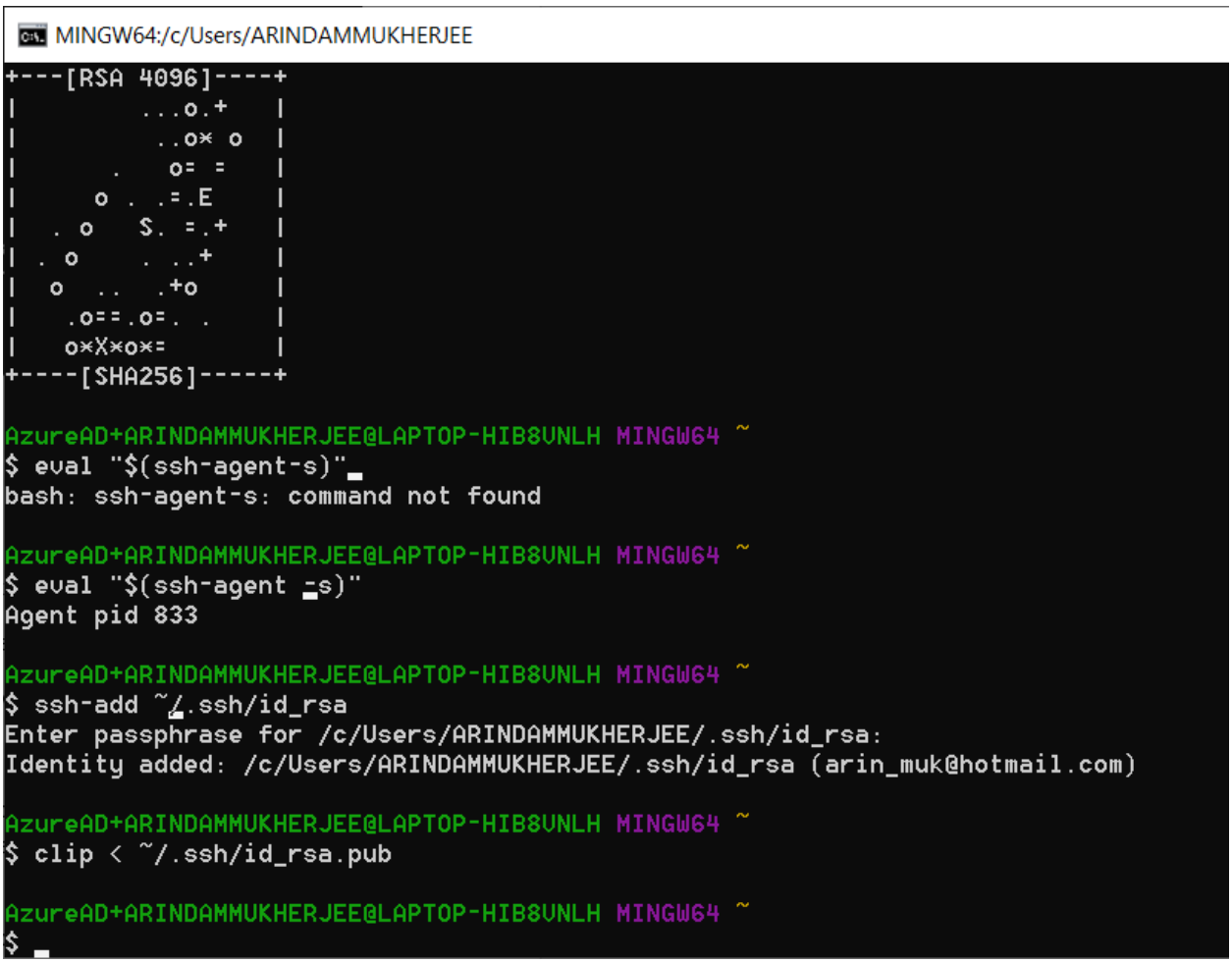
ssh-add ~/.ssh/id_rsa

9. When prompted for a passphrase, enter the one associated with the key.

If you've forgotten this password, just create a new one, starting with step 4.

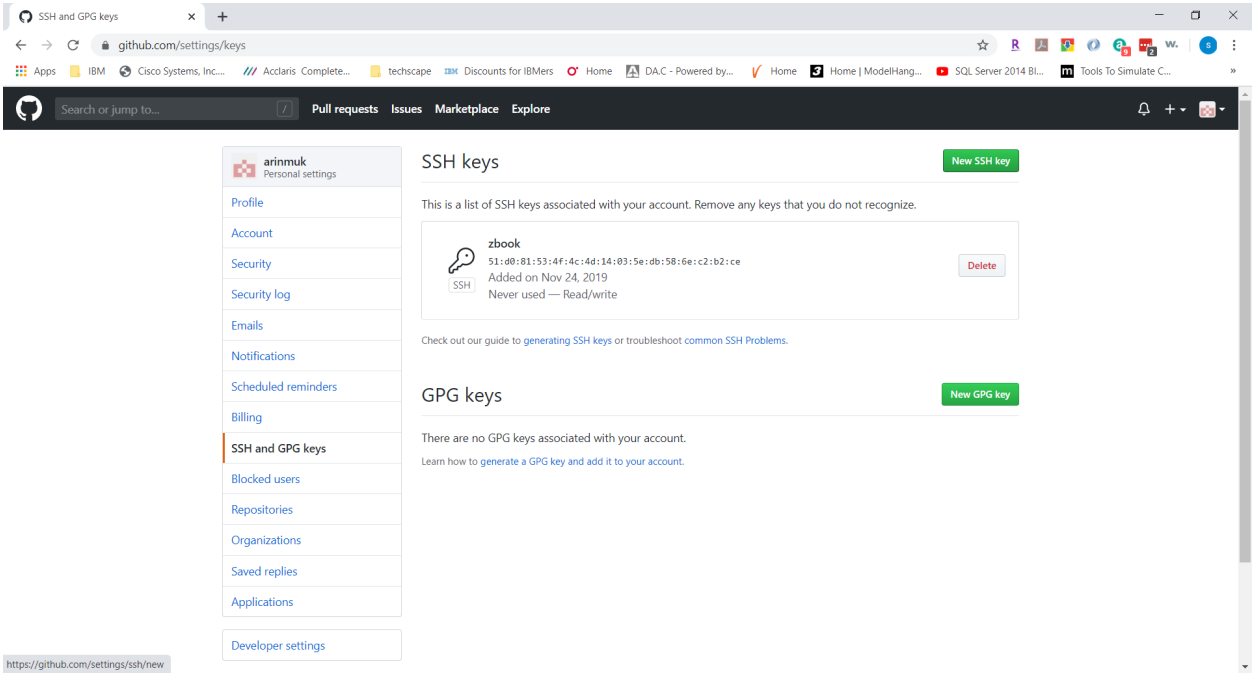
10. We need to add the key to GitHub. Copy the key to your clipboard by entering this command:

clip < ~/.ssh/id_rsa.pub



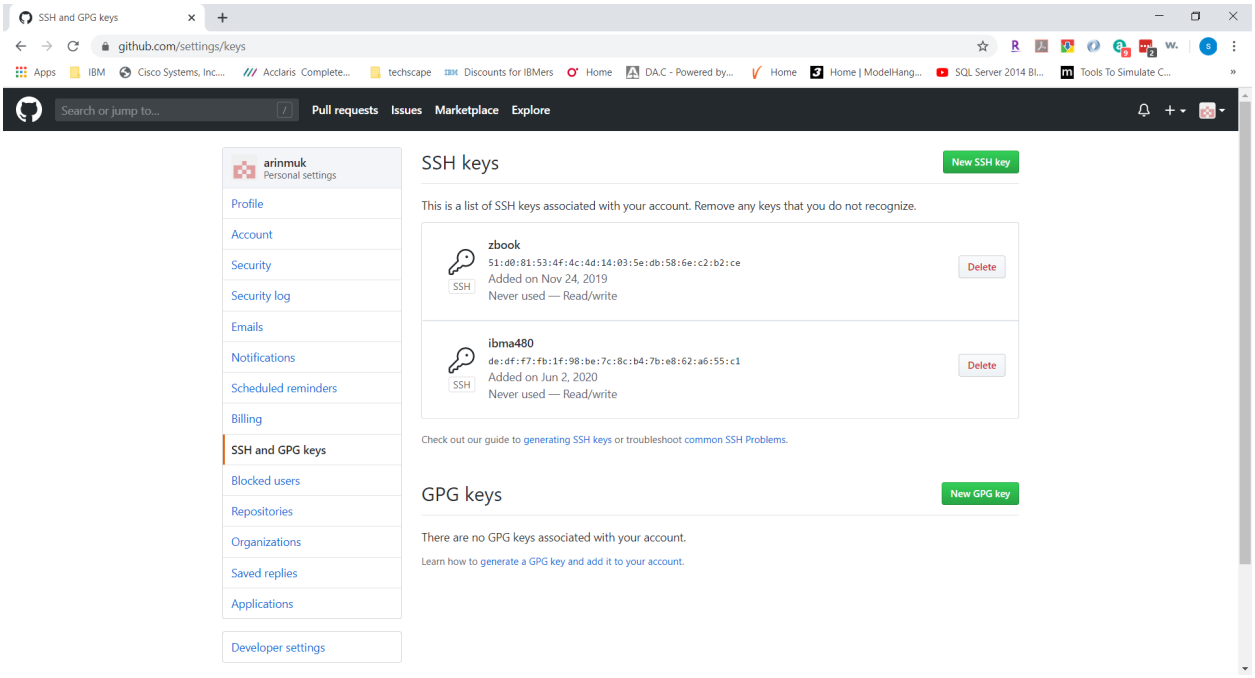
You shouldn't see any kind of message when you run this command. If you do, make sure you entered it correctly. Do not copy anything else to your clipboard until you finish the next instructions. Otherwise, you'll have to repeat this step again.

11. Go to https://github.com/settings/ssh. Click the "New SSH Key" button.



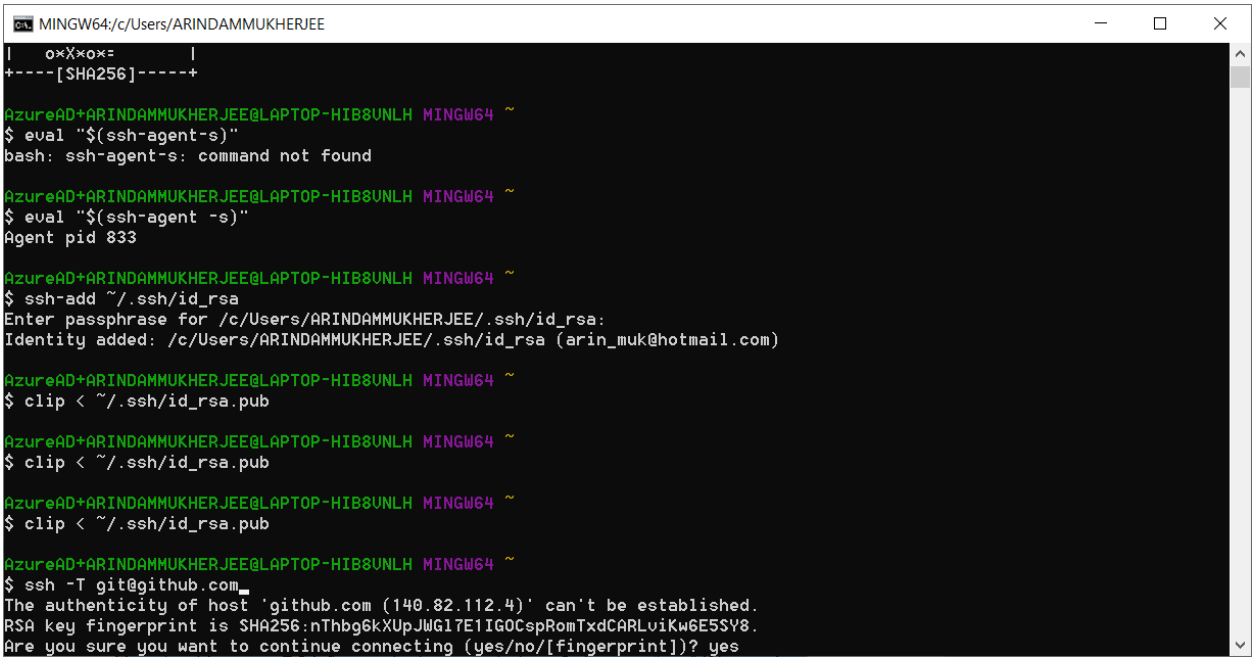
When the form pops up, enter a name for your computer in the Title input. In the Key input, paste the SSH key you copied in step 10.

Paste the key and save/add

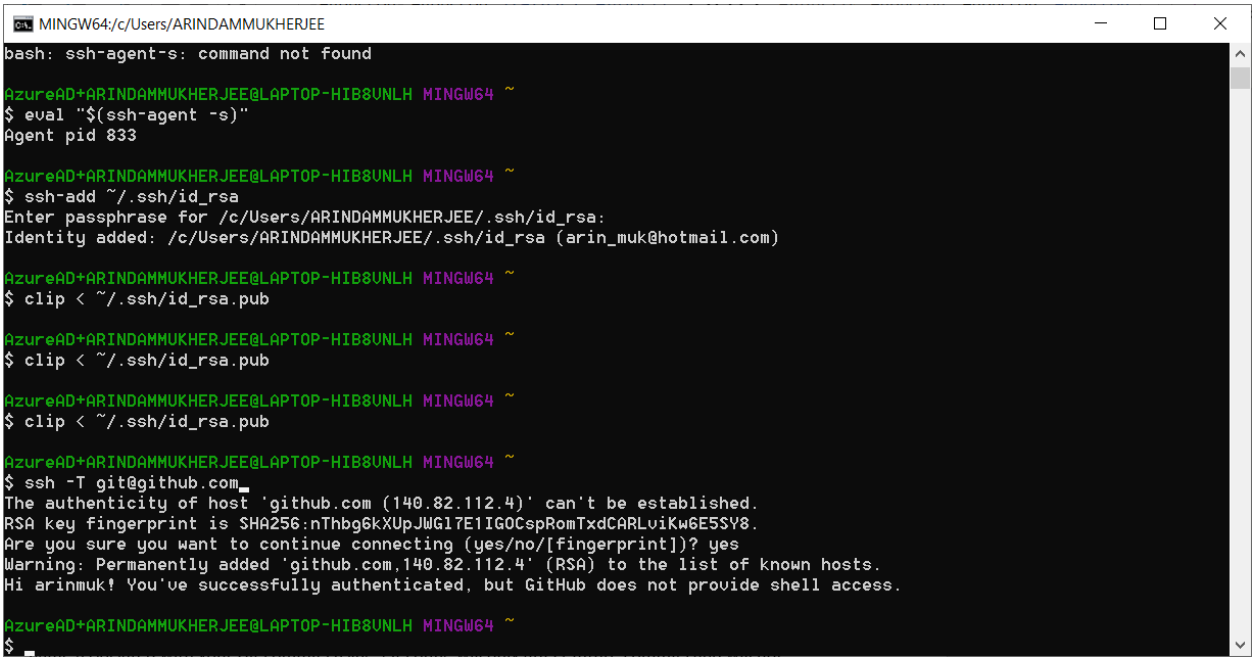


13. Now we just need to add GitHub to your computer's list of acceptable SSH hosts. Go back to your Bash window. Type in this command: ssh –T git@github.com

You should see an RSA fingerprint in your window. Only enter "yes" if it matches the one highlighted in the image below.



Type “yes” and hit Enter



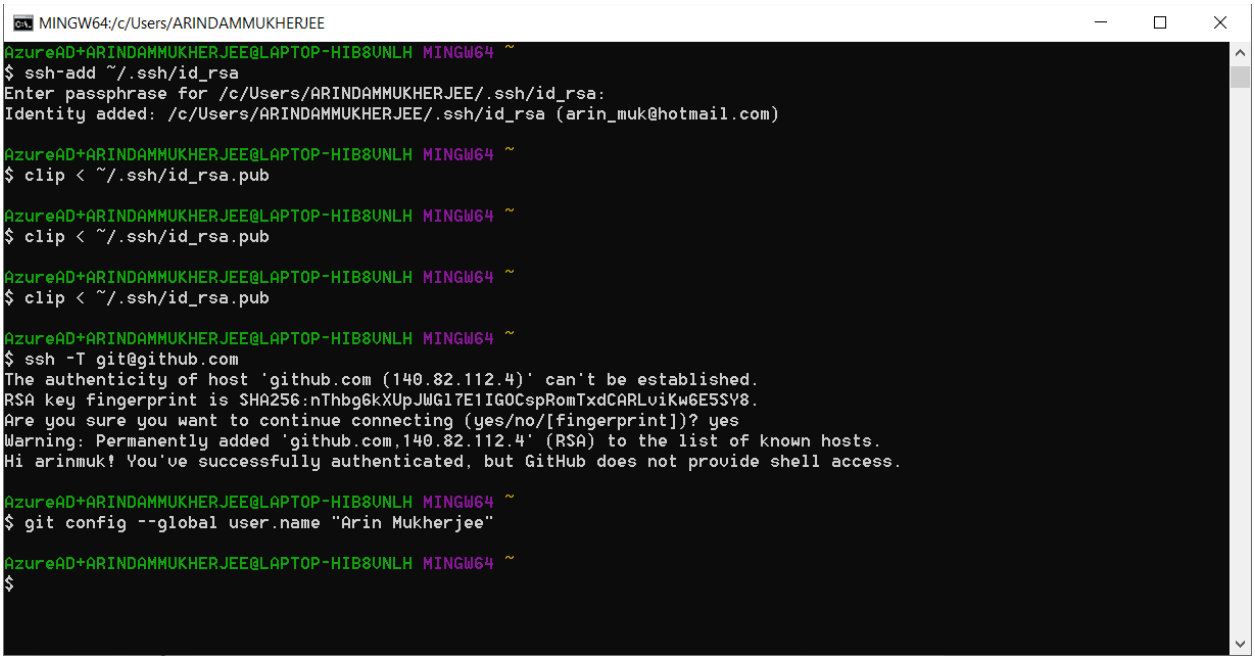
Setting your Git username for every repository on your computer

Git uses a username to associate commits with an identity. The Git username is not the same as your GitHub username.

You can change the name that is associated with your Git commits using the git config command. The new name you set will be visible in any future commits you push to GitHub from the command line. If you'd like to keep your real name private, you can use any text as your Git username. Changing the

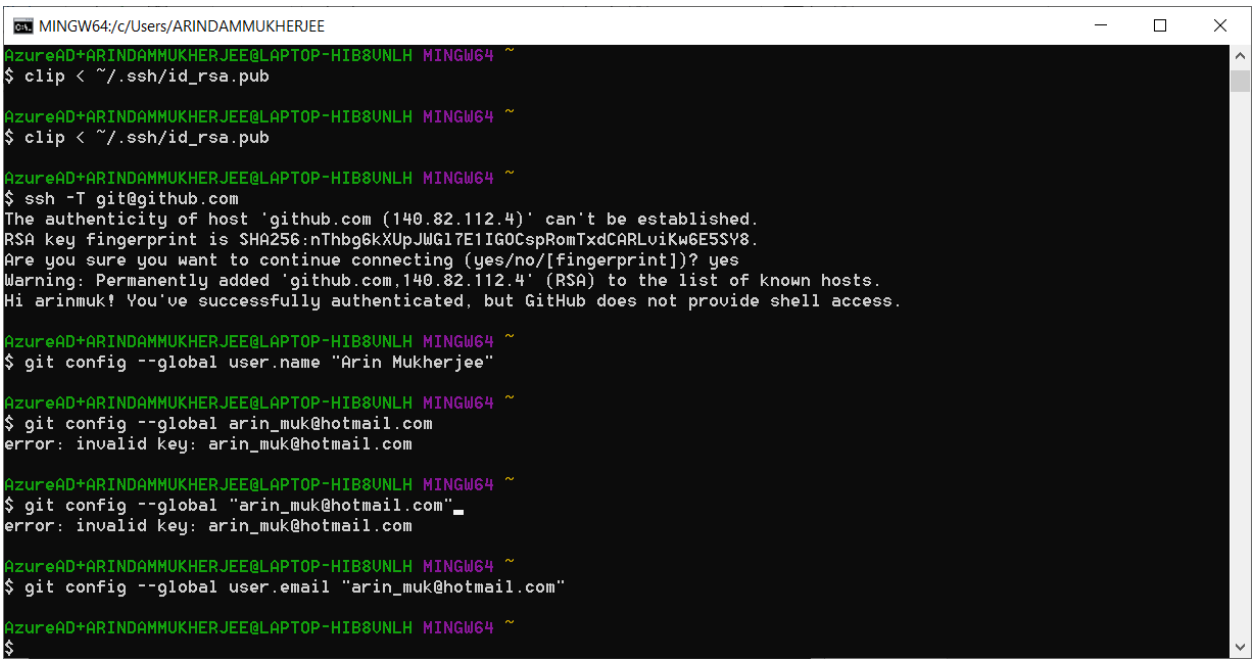
name associated with your Git commits using git config will only affect future commits and will not change the name used for past commits

Setting your email address for every repository on your computer



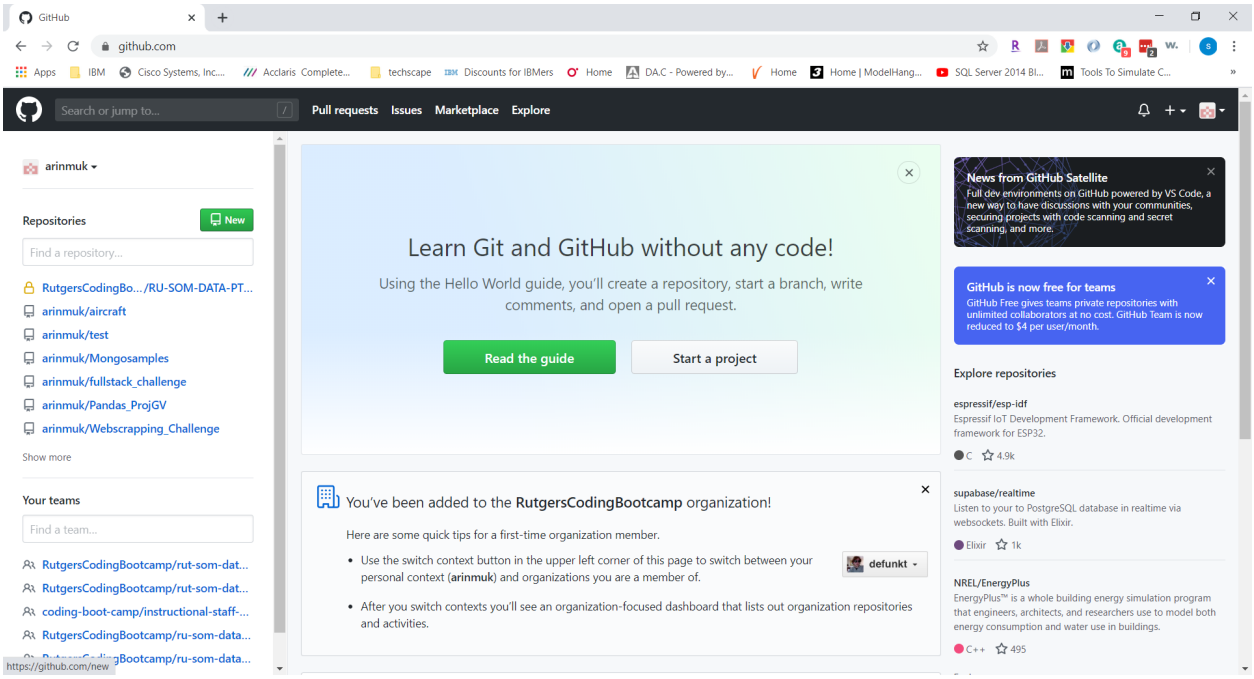
Above picture: Git config to ensure your username eis registered for commits. Next picture shows the command to add your email for the commits.

git config --global user.email “xxxx@xxxxmail.com"

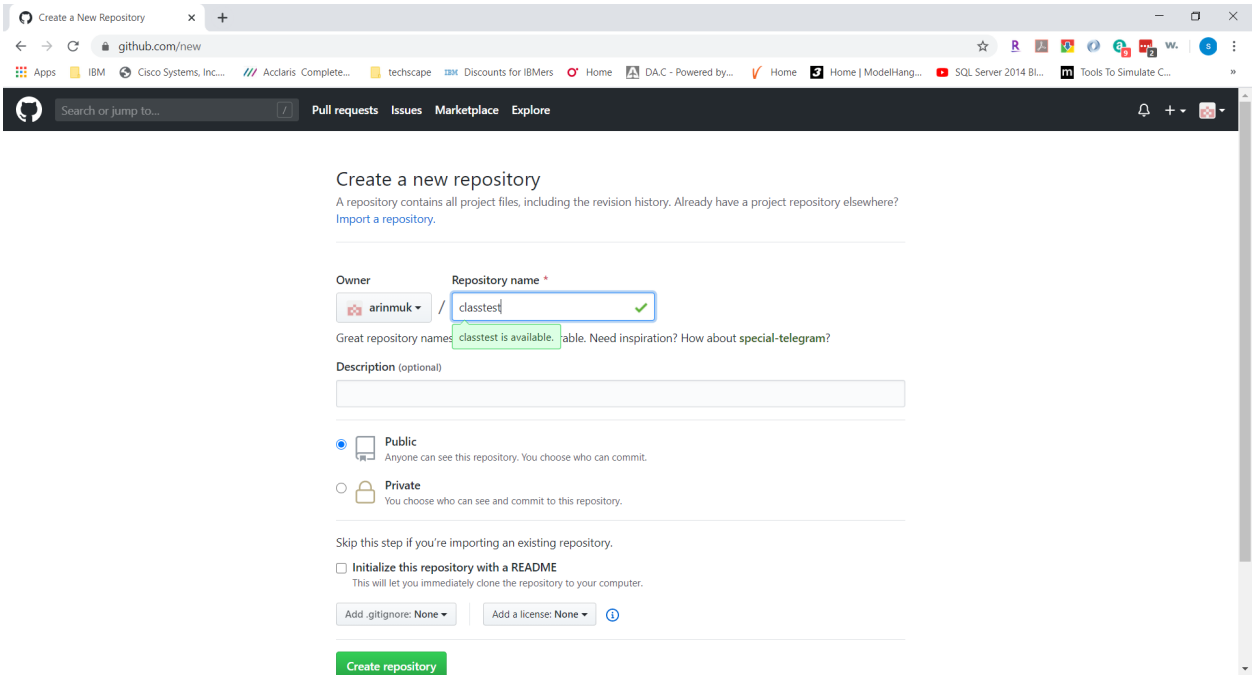


How to use git to work and upload home work

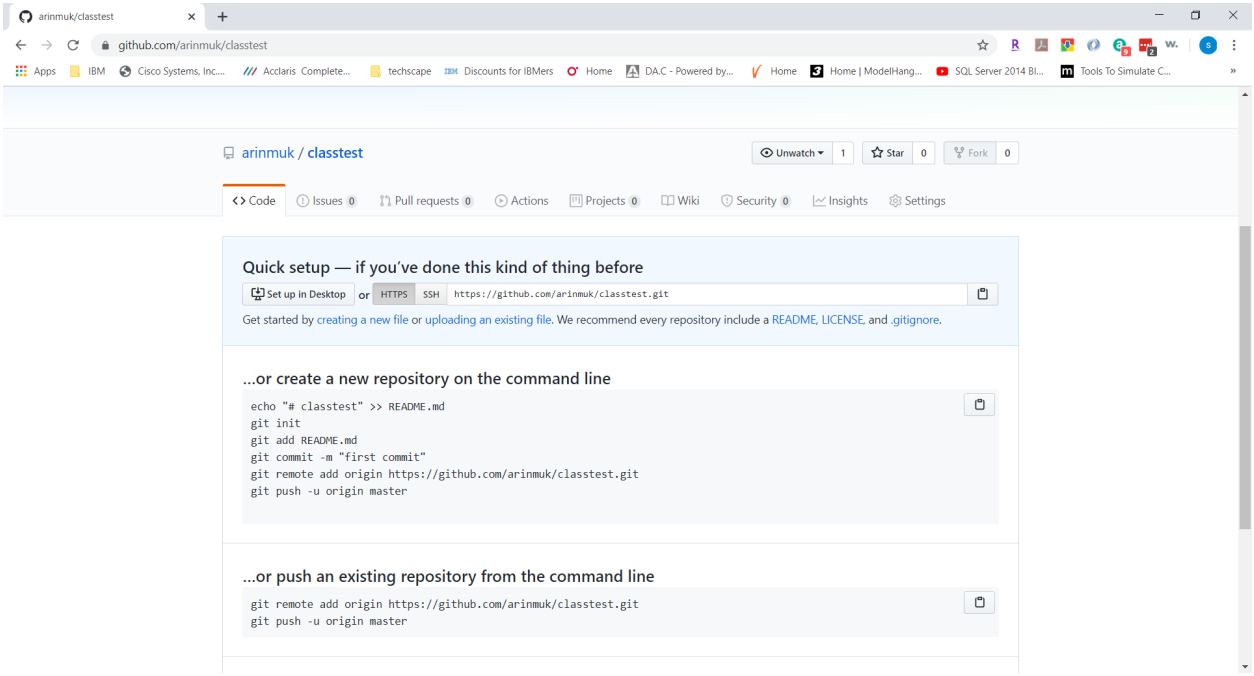
1. Create the “new” repository at github.com



2. Enter the name of the repository (normally the homework name).create repository.



3. Once the repo is created you should see something like this



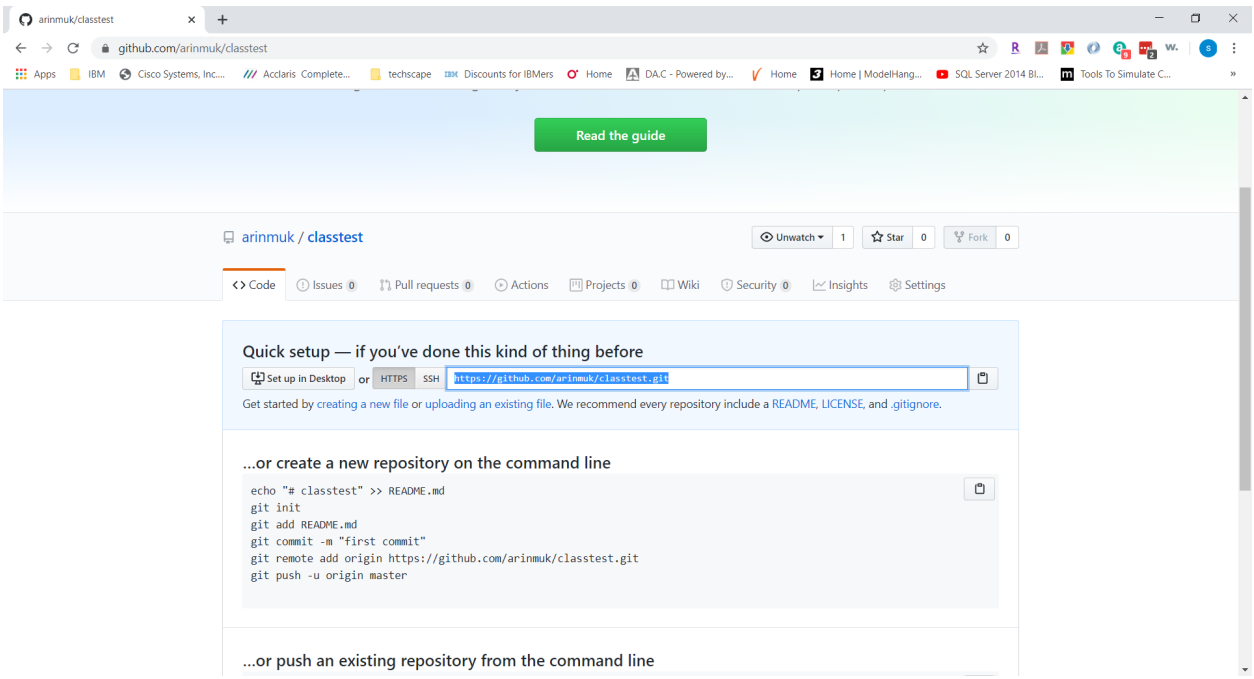
4. As best practice , we should create a readme file and have some details about the project or work .Generally description , libraries and how to run the program is included.
5. Now lets go back the bash terminal
6. Create folder say “test” (be careful as to where you create the folder . Exact location is required). Folder in gitbash can be created via command

$ mkdir test

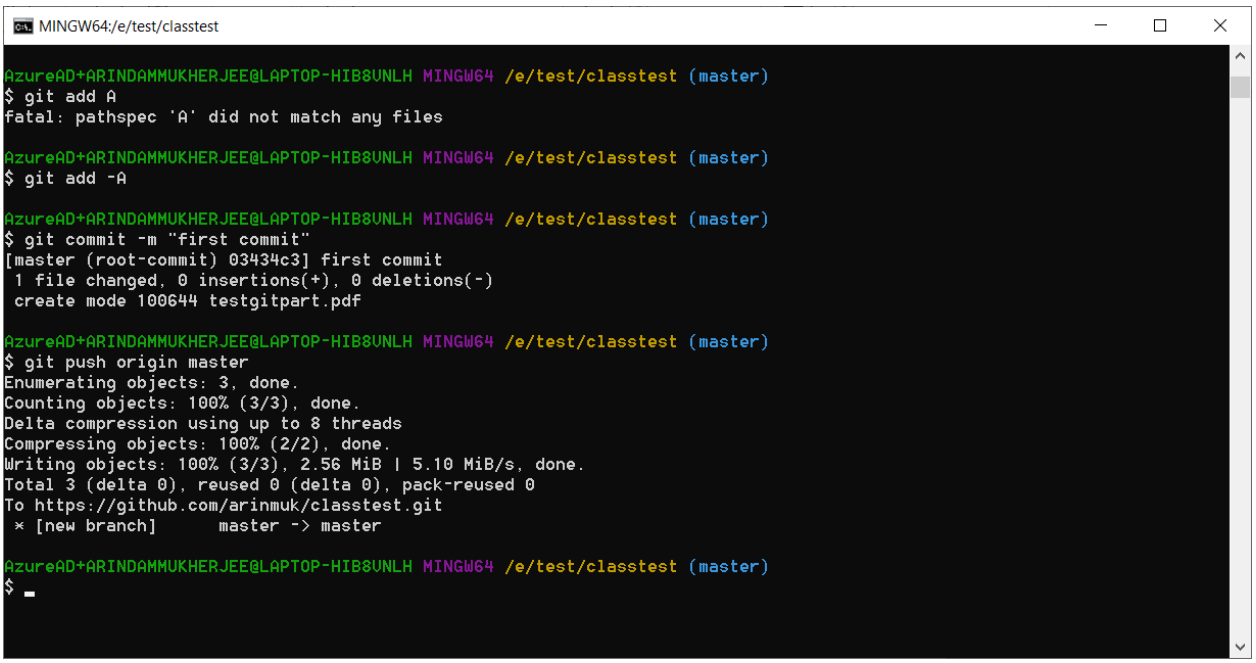
Then change to folder or go into that folder

$ cd /test

7. Once you go to the folder lets return to the webpage in chrome where we had created the repository and copy the highlighted line in memory using ctrl+c

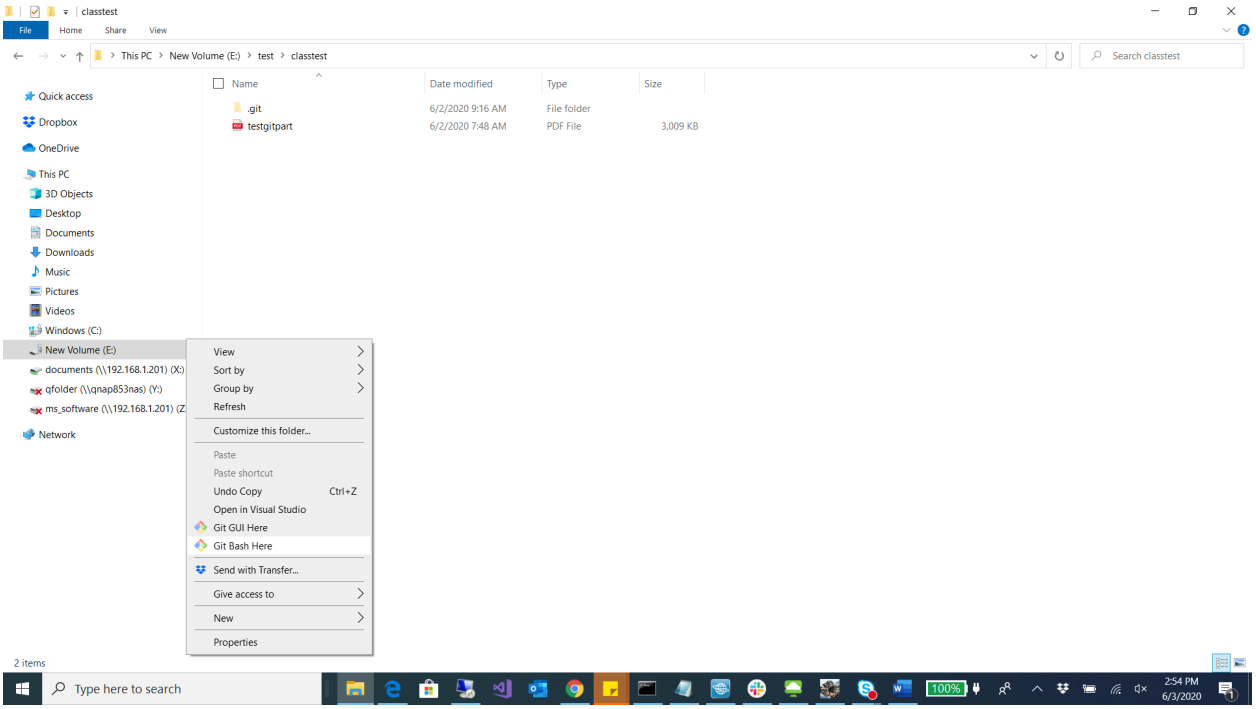


8. Back to git bash and type in
   git clone and the link you copied to memory from the chrome window
   $git clone https://github.com/arinmuk/classtest.git
   This will link up your local folder to the git repo at github
9. You can now start working in your local folder . After you are done working or want to commit your changes , following commands in order has to be given
   $ git add -A
   (this adds the files or changes made to files in a holding area)
   $ git commit -m “first commit”
   (this commits the file to the local repository)
   $ git push origin master
   (this pushes the file to the github repository)

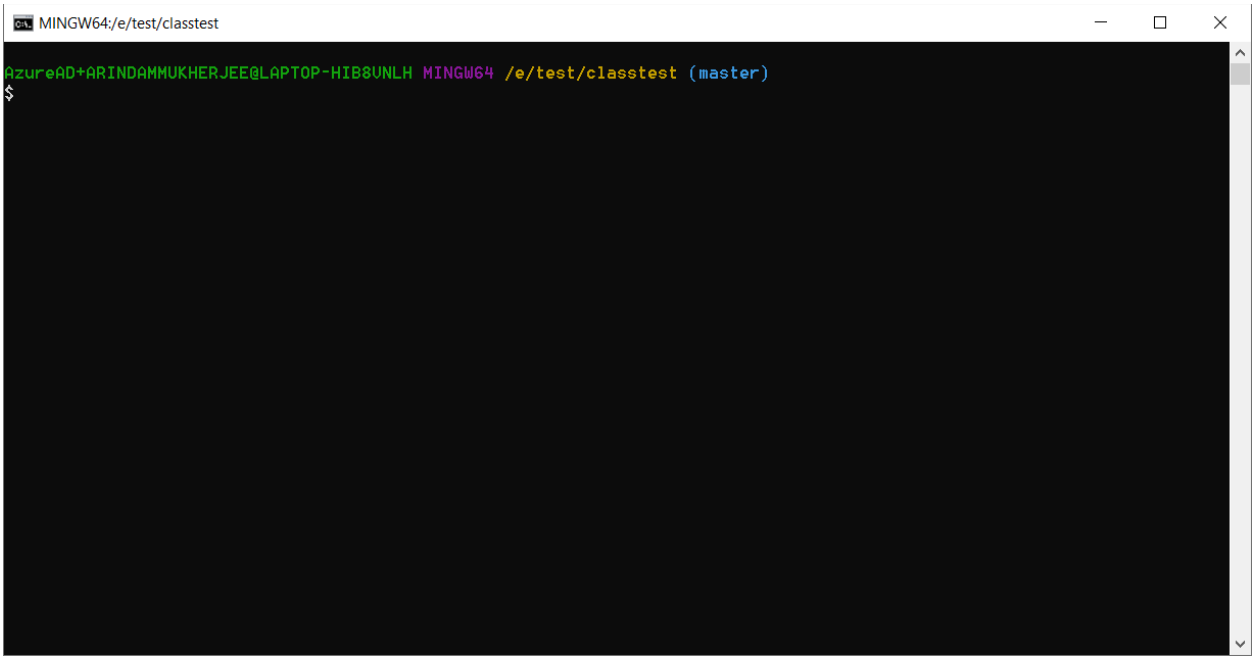

MINGW64:/e/test/classtest
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 /e/test/classtest (master)
$ git add A
fatal: pathspec 'A' did not match any files
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 /e/test/classtest (master)
$ git add -A
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 /e/test/classtest (master)
$ git commit -m "first commit"
[master (root-commit) 03434c3] first commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 testgitpart.pdf
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 /e/test/classtest (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 2.56 MiB | 5.10 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/arinmuk/classtest.git
* [new branch] master -> master
AzureAD+ARINDAMMUKHERJEE@LAPTOP-HIB8UNLH MINGW64 /e/test/classtest (master)
$

How To Download Git Repository posting from the class repository

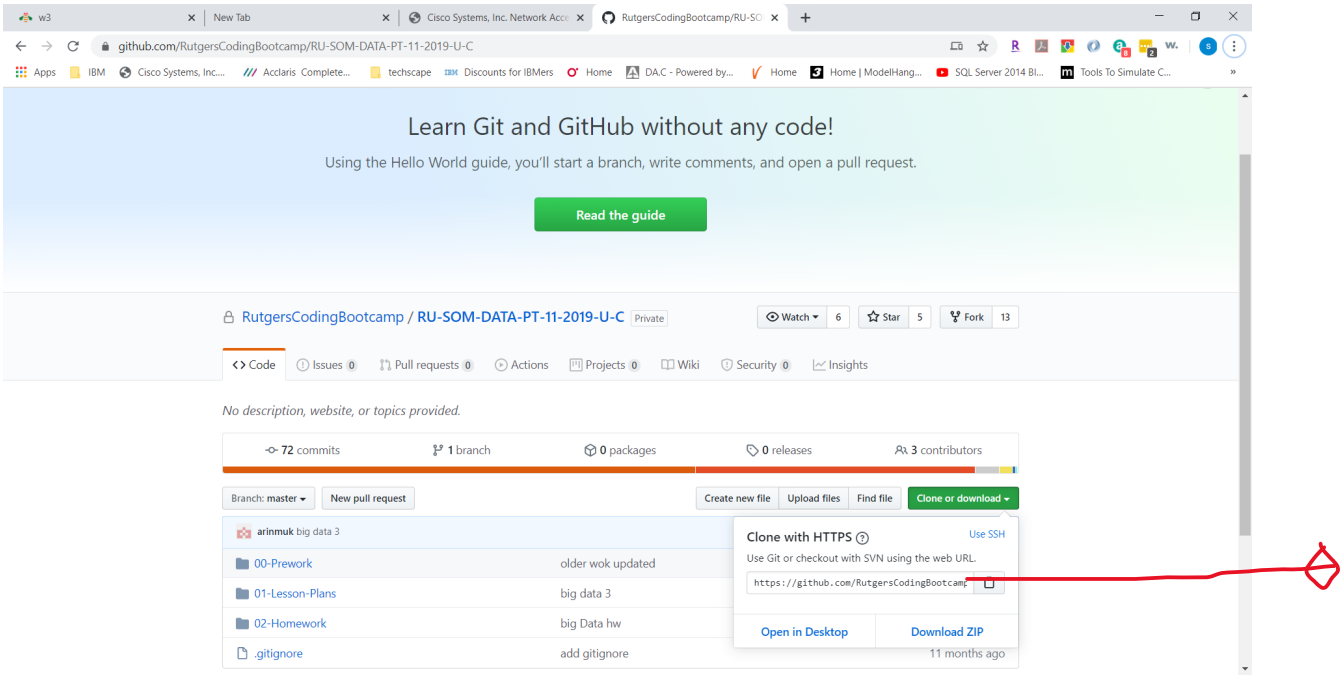
1. Create Folder in your local machine. Lets call it MyClassRepo
2. Git bash Into this folder



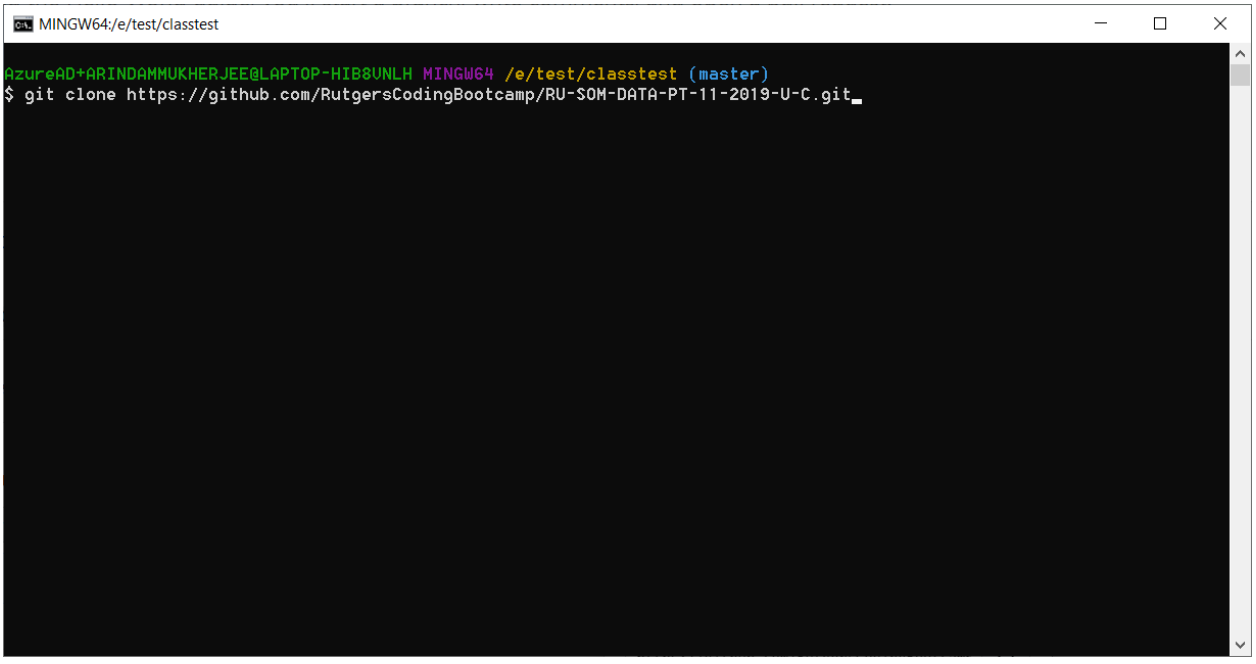
3. When you choose Gitbash here, it will open up like below



4. Now open to class repository from google chrome
   Link for the class repo
   https://github.com/RutgersCodingBootcamp/RU-SOM-DATA-PT-11-2019-U-C
5. choose clone or download . The screenshot below shows it. You are looking for the link to copy . select the link and copy it using CTRL+C



6. Go back to the git bash window that you have opened earlier and type in
7. git clone <<<<<<<<<<paste the link you copied from the github>>
   So it will look like below



You are all set for the first initial clone .

On a daily basis after the class you should

a) git bash into this folder
b) and run “git pull”
   This will syn your local repo with all the class materials.

   Note : Very Important
   You should not make any changes to the files in this folder or repo locally. If you need work on this , copy the folder to a working folder and make changes in the working folder .
   This will keep the repo intact and not change the files when you do a git pull.