

# Setting Up Git

Hamid Semiyari

## Git

Git is a version Control System (VCS) for managing changes across multiple version of documents/code in a **Repository** you have created. Git Commonly installed in many computers as a part of Operating System (OS).

- To check if you have installed it:
  - Go to **Terminal** pane in Rstudio (or open in new terminal with **Tools > Terminal > New Terminal**) then type

```
git version
```

or

```
git --version
```

If the version number is lower than 2.43.0 or if the response is blank or something like

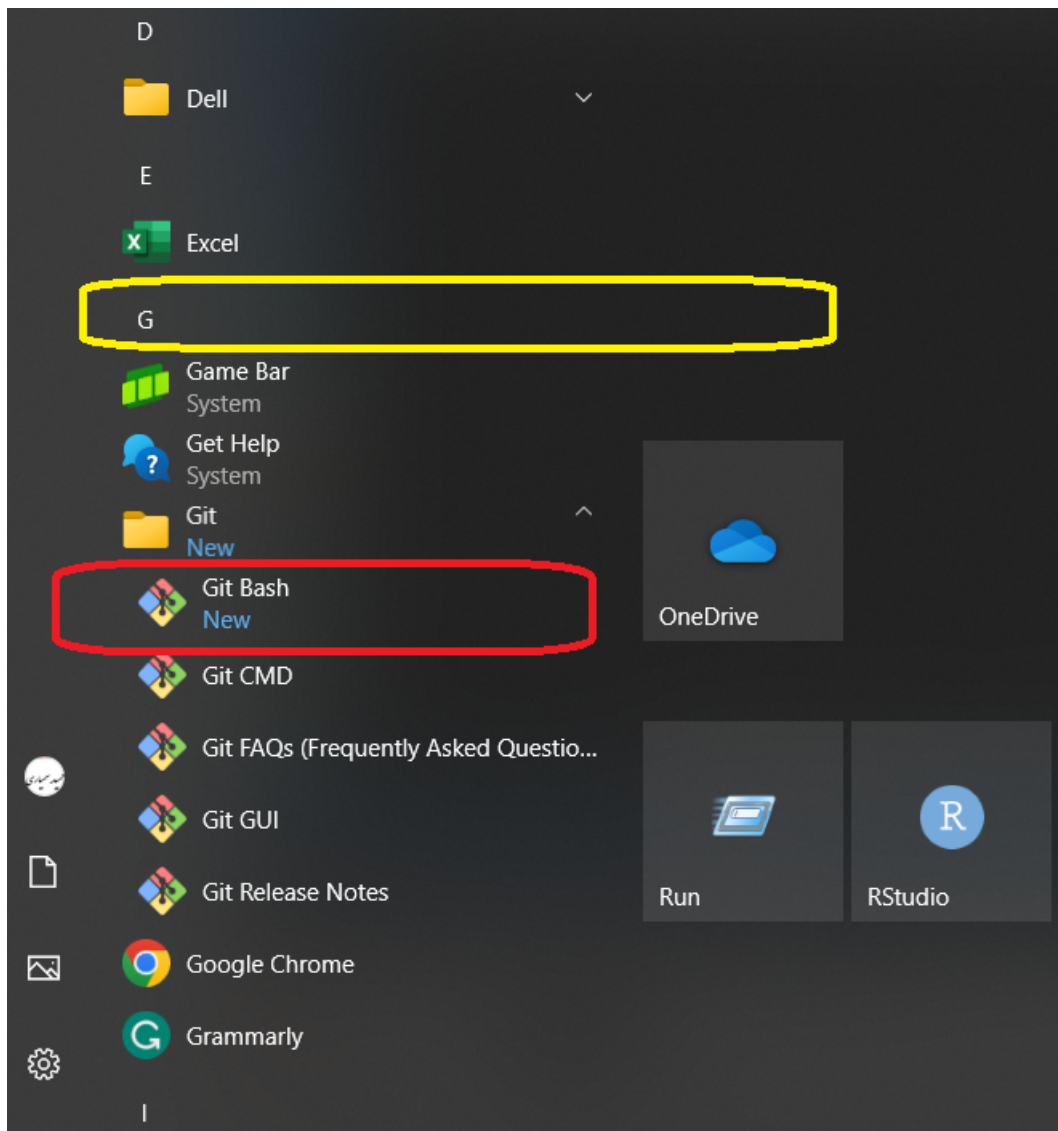
```
git: command not found
```

then you need to install it.

## Install Git

- Windows:
  - Click on <https://git-scm.com/download/win>
  - Click **Download** (Standalone Installer - 64 bit) .When it is finished open the “Explore” window ( press the Windows logo key + E on your keyboard) and go to download folder and open it there.
  - Once its open we click “Next” and we accept the default options until we get the install page and then click “install”.
  - I t takes a while to install and once it is done we can click “Finish”.

Let us check if “Bash” is installed. + Click on Windows menu button and choose all apps, then scroll down to the “G,s” and expand “Git” Options and click “Git Bash” option.



- Type in BASH `git --version` it will tell you the version of your Git.
- Mac:
  - Option 1: Open the terminal and type

```
git --version
```

If git is not installed you will get a prompt asking you if you wish to install it along with

- Option 2: Download and run the installer from: [\\[https://git-scm.com/download/mac\\]](https://git-scm.com/download/mac) (<https://git-scm.com/download/mac>)

- Install `Homebrew`. See [Homebrew Installation](https://brew.sh/) and install in the def

- Linux:

- Open the terminal and type

```
git --version
```

If git is missing install is using your package manager. E.g. by running

```
sudo apt-get install git
```

In the terminal.

### **Question :If you have installed Bash what version do you have?**

- Type in BASH/Terminal `git --version` it will tell you the version of your Git.

```
bash --version
```

### **Question :If you have installed Git what version do you have?**

- Type in BASH/Terminal `git --version` it will tell you the version of your Git.

```
git --version
```

## **Configuring Git**

- The first time we use git we have to specify a few configuration settings. -Name -Email -Default Editor -Line Ending; How it should handle line endings. We can specify this configuration settings in three different levels.
- System Level: Applies to all users of the current computer
- Global Level: The settings here apply to all repositories of the current user
- Local Level: Setting here apply to the current repository or repository in the current folder So we have different setting for different repositories or different projects.
- In Git BASH type:
  - `git config --global user.name "Hamid Semiyari"`
  - `git config --global user.email semiyari@american.edu/`
- We don't need double quotes for email.

- If you are worried about email privacy, then use “@users.noreply.github.com” as your email address (where you have replaced with your GitHub username), then follow the instructions [here](#) to make your email address private on GitHub.

## Default editor

If you do not set this on mac. Git is going to use (by default) vim which many people doesn't like it. You may want to use [visual studio code](#). **You don't have to use Visual Code, You may use another Editors.** I am going to use VSCode or Visual Studio Code. To add the default editor `git config --global core.editor "code --wait"`.

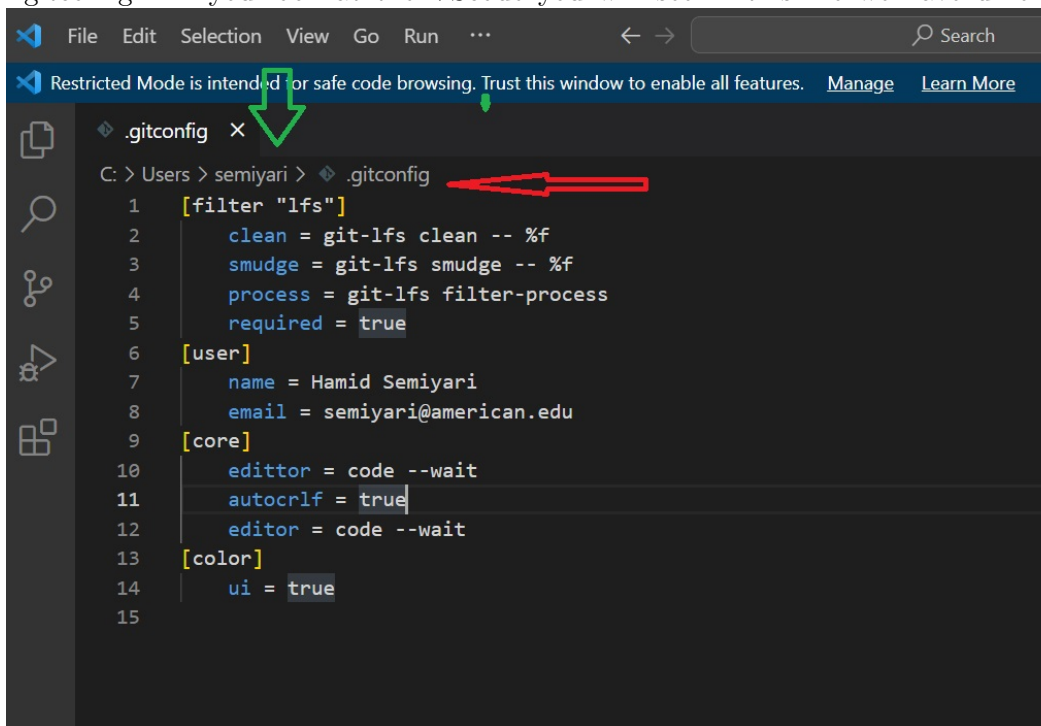
~~Why wait? With the wait flag we tell the terminal window to wait until we close the new VSCode.~~

All these configuration setting are store in a text file.

We can edit that file using our default editor. Which in this case is VSCode.

So we can type `git config --global -e`

This will open our default editor to edit all the global setting. If you type the code, you will see the full path on top, which mine is `users/semiyari` the name of this file is “.gitconfig”. If you look at the VScode you will see in this file we have different sections



```

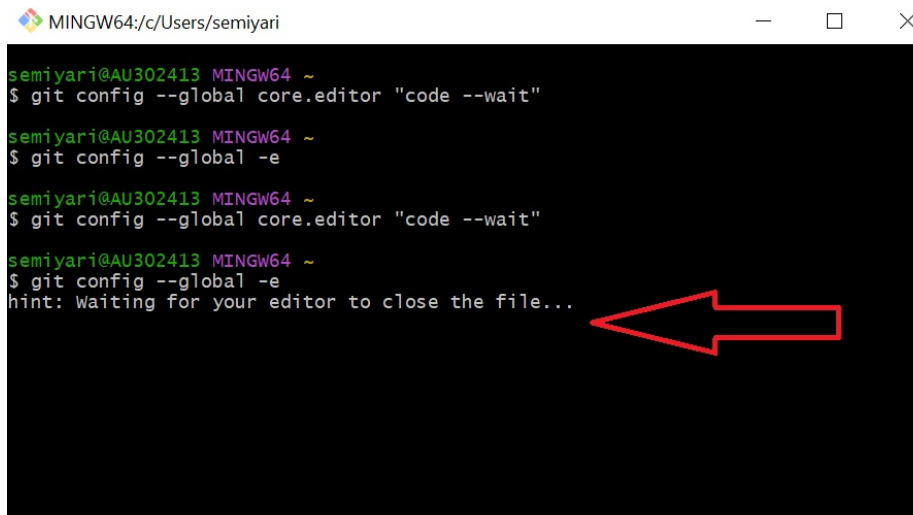
File Edit Selection View Go Run ... Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

.gitconfig X
C: > Users > semiyari > .gitconfig
1  [filter "lfs"]
2      clean = git-lfs clean -- %f
3      smudge = git-lfs smudge -- %f
4      process = git-lfs filter-process
5      required = true
6  [user]
7      name = Hamid Semiyari
8      email = semiyari@american.edu
9  [core]
10     editor = code --wait
11     autocrlf = true
12     editor = code --wait
13  [color]
14     ui = true
15

```

- \[user\] section: Name and Email
- \[core\] section: With two settings

- `\[filter 'lfs"\]` section
- `\[color\]` section
- If you back into terminal you will see it is waiting for us to close the VSCode.

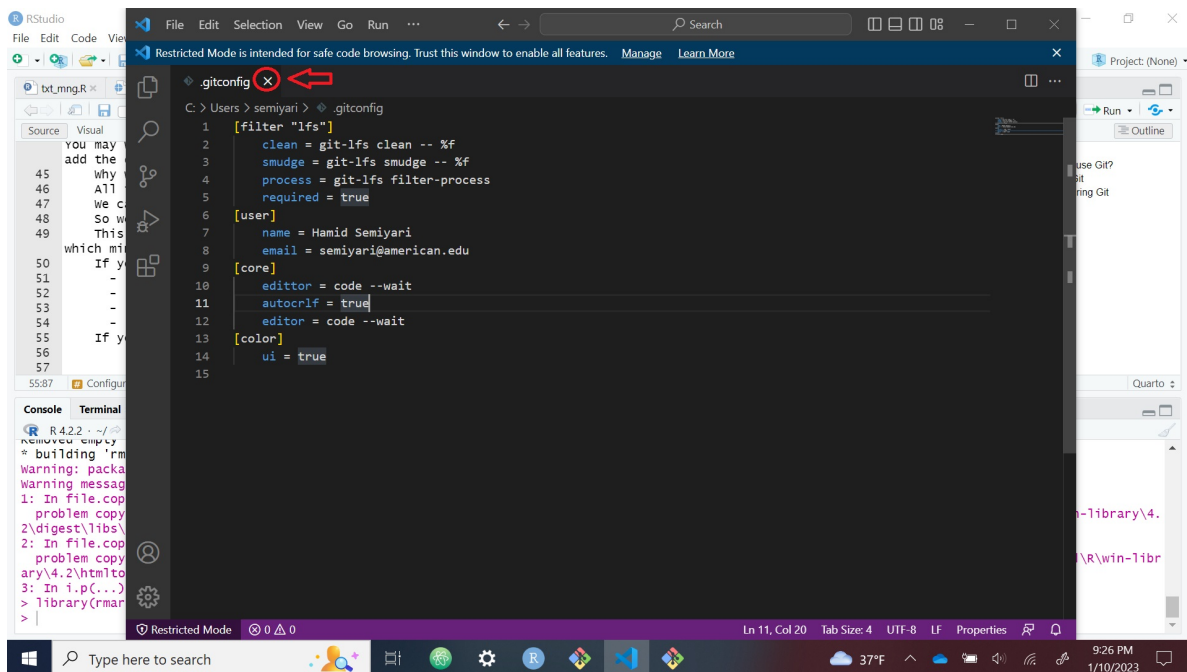


A terminal window titled "MINGW64:/c/Users/semyari" with standard window controls. The terminal shows the following commands and output:

```
semyari@AU302413 MINGW64 ~  
$ git config --global core.editor "code --wait"  
  
semyari@AU302413 MINGW64 ~  
$ git config --global -e  
  
semyari@AU302413 MINGW64 ~  
$ git config --global core.editor "code --wait"  
  
semyari@AU302413 MINGW64 ~  
$ git config --global -e  
hint: Waiting for your editor to close the file...
```

A red arrow points to the "hint: Waiting for your editor to close the file..." message.

- So we have to close the VSCode



- Handle end of lines: This is a very important setting that lots of people miss.

- On Windows end of lines are marked with two special characters `\r` and `\n`.
- Carriage Return `\r`: It means that the cursor should **go back to the beginning of the line**.
- Line feed `\n`: It means that the cursor must **go to the next line**.
- On Mac and Linux systems, the end of lines are indicated with - Line feed `\n`.

That means if we don't handle end of lines properly we are going to run into some issues down the road.

- **Windows:** `git config --global core.autocrlf true` (crlf is short for Carriage Return Line Fit)
- **Mac or Linux:** `git config --global core.autocrlf input`

## You may use different editor

- Choose your favorite editor to be your default
  - Nano (a simple text editor):

```
`git config --global core.editor "nano -w"`
```

- Notepad (Windows only):

```
`git config --global core.editor "c:/Windows/System32/notepad.exe"`
```

- Emacs (for the brave) if it is already installed:

```
`git config --global core.editor "emacs"`
```

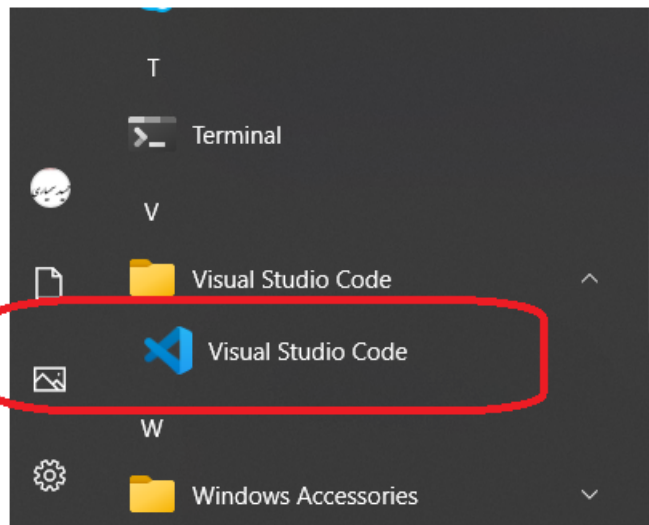
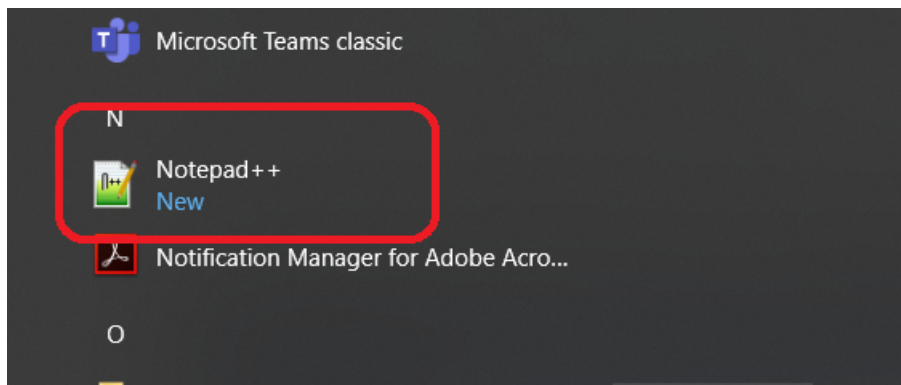
- [Atom](#)

1. Open <https://atom.io/> in your browser.
2. Click the ‘Download’ button to download the Atom installer.
3. Once Downloaded, run the installer to install Atom.

### **How should I check what Text Editor I have in my computer?**

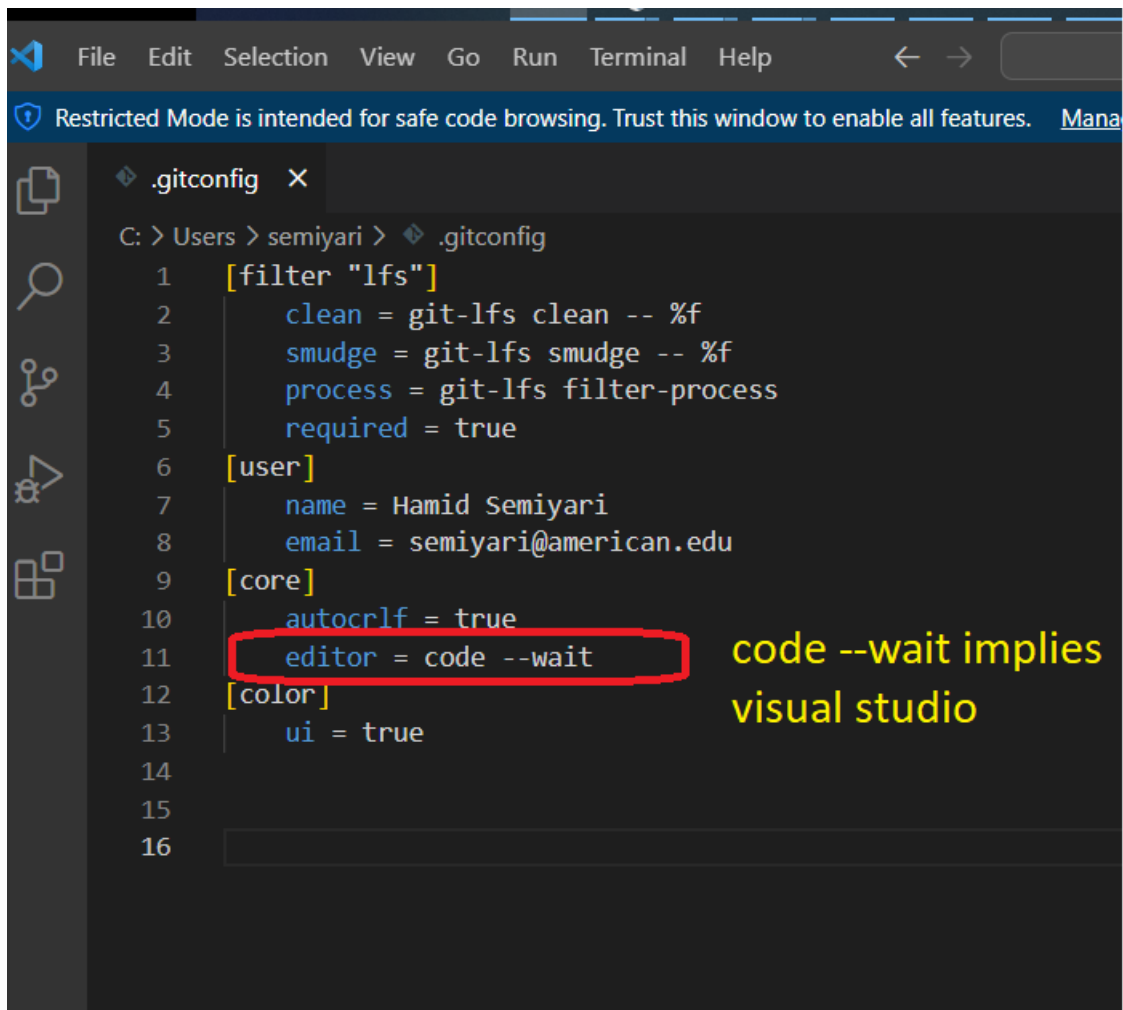
- 1. Simply click your Windows Start button and then go to All Apps and scroll until you locate your Editor.





- 2. Type the following command in your terminal

```
git config --global -e
```



```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Mana

.gitconfig X
C: > Users > semiyari > .gitconfig
1 [filter "lfs"]
2     clean = git-lfs clean -- %f
3     smudge = git-lfs smudge -- %f
4     process = git-lfs filter-process
5     required = true
6 [user]
7     name = Hamid Semiyari
8     email = semiyari@american.edu
9 [core]
10     autocrlf = true
11     editor = code --wait
12 [color]
13     ui = true
14
15
16
```

code --wait implies  
visual studio

If you want to change the editor, you can use:

```
git config --global core.editor <editor>
```

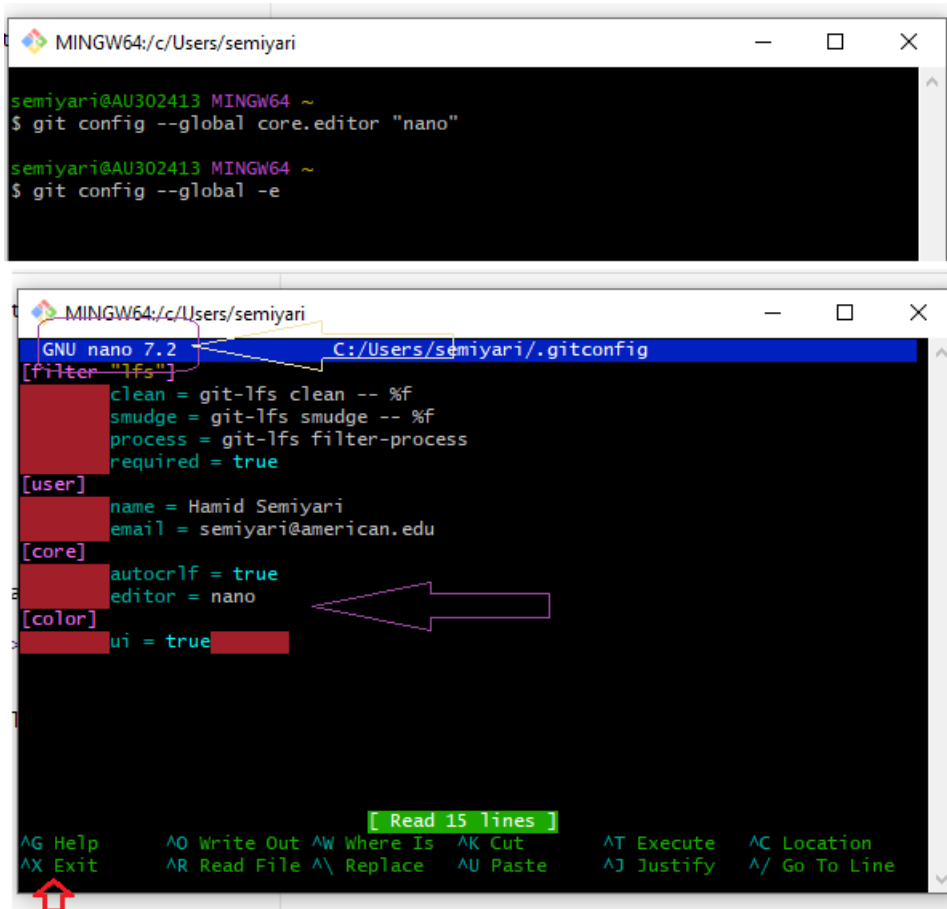
The editor “Nano” is installed by default. If you want to use this one you need to write

```
git config --global core.editor "nano"
```

Then type

```
git config --global -e
```

To exit you just need to hold `ctrl` and click `x`, `ctrl + x`



```
MINGW64/c/Users/semyari
semyari@AU302413 MINGW64 ~
$ git config --global core.editor "nano"
semyari@AU302413 MINGW64 ~
$ git config --global -e

MINGW64/c/Users/semyari
GNU nano 7.2 C:/Users/semyari/.gitconfig
[filter "lfs"]
clean = git-lfs clean -- %f
smudge = git-lfs smudge -- %f
process = git-lfs filter-process
required = true
[user]
name = Hamid Semiyari
email = semiyari@american.edu
[core]
autocrlf = true
editor = nano
[color]
ui = true

[ Read 15 lines ]
^G Help  ^O Write Out ^W Where Is ^K Cut      ^T Execute  ^C Location
^X Exit  ^R Read File ^\ Replace  ^U Paste    ^J Justify  ^_ Go To Line
```

- Question:

What is difference between “Text Editor” and “Integrated Development Environment” or “IDE”?

A code editor is a text editor that has some features that make writing code easier. For instance, code editors will automatically highlight words based on syntax and will automatically indent lines of code correctly. An IDE is usually more complex than a simple text or code editor. It has more features, it takes time to learn, but in the end is more powerful. It is a piece of software that allows developers to create, modify, and debug code easily. A code editor doesn't have the debugging and autocomplete features that an IDE has. Atom is a code editor and Rstudio is an IDE.

- Question: How to change your default text editor for git (and avoid vim)

– To find what your text editor is type the following in your terminal

```
git config --global core.editor
```

- If you see `code --wait`, that means “VSCode”
- If you are an apple user and have not selected your code editor, then by default you have `vim`.
- If you want to learn how to change your code editor click [here](#)

## Getting Help

- How to get help about git commands - If you want to learn more about config commands. Just google “git config”, then you can be directed to the page [git-scm.com/docs/git-config](https://git-scm.com/docs/git-config)
  - We can also get the same information from the terminal window, just type `git config --help` - If you type `git config -h` It gives us a short summary of its commands and options