Git Part 1

Git

- Git is the most popular `version conrol` system in the world. A version control system records the changes that made to our code over time in a special database called `repository`. We can look at our project history and see who has made what changes when and why. If we made a mistake on something we can easily revert our project back to an earlier state.
  In summary, with version control system we can track history and work together. _ Git is free, open source, super fast, and scalable. Operation such as `branching` and `merging` are slow in other version control system like `subversion` or `tfs`But they are very fast in Git.
- Git is everywhere, almost every job description (programming related) mentions Git.

How to use Git?

- Command line: We can use Git on the command line. Open a termianl or command window to execuate git commands. This the fastest sometimes easiest way to get the job done.

- Code editors & IDEs (or Integrated Development Environment): Most code editors and IDEs have built-in support for basic Git features.

- Graphical User Interfaces: There are GUIs specifically made for using Git. You can find the complete list of these tools for different platforms on the Git website. Click here. GUI tools are not always available. GuI tools have somelimitations. You might connect to a server remotly and you may not have permission to install GUI tool. So to my opinion it is better to use command line. There are many people use both GUI tool and command line.

  - A code editor is atext editor that has some features tht 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 creat, 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.

Install Git

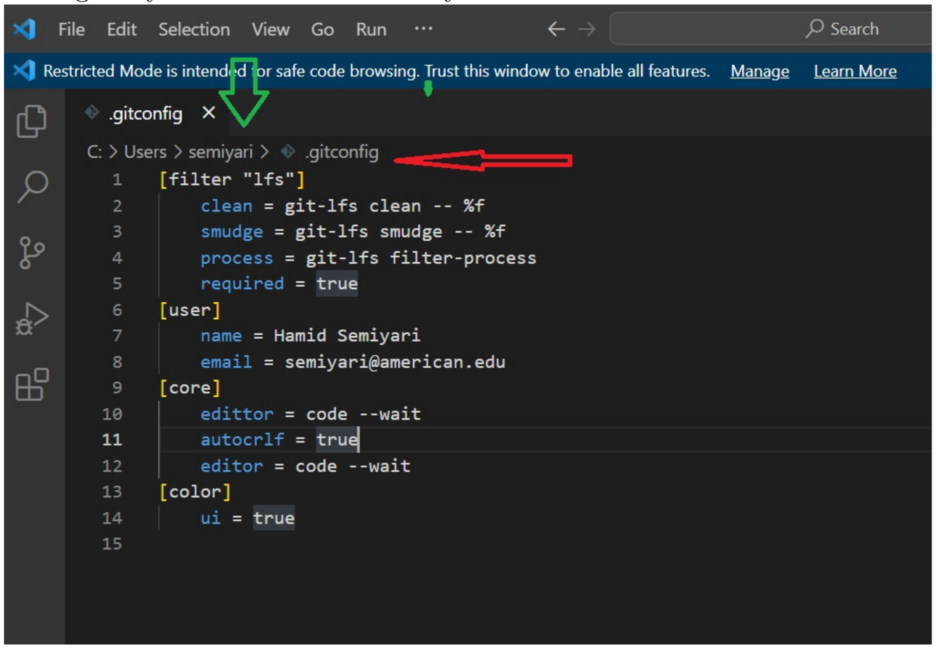
If you have installed Git what version do you have? To answer this you must open a terminal. How?

- Mac: Press `command+space` and type `terminal`
- Windows: click the `search` icon and type `cmd` If you don't have the latest version of Git. Just go to git-scm.com/downloads - Window user need to install `Git Bash` from gitforwindows.org. BASH stands for Bourne Again SHell. It is better to use `BASH` instead of built-in command prompt window. - Type in BASH `git --version` it will tell you the version of your Git.

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: The setting here, applies to all users of the current computer
  - Global Level: The settings here apllies 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". The reason we have double quote is because we have a space in value.
  - `git config --global user.email  semiyari@american.edu` . We don't need double quotes for email.
  - 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 download visual studio code. I am going to use `VSCode` or Visual Studio Code. To add the default editor `git config --global cor.editor "code --wait"`.
    Why wait? With the wait flag we tell the terminal window to wait until we close

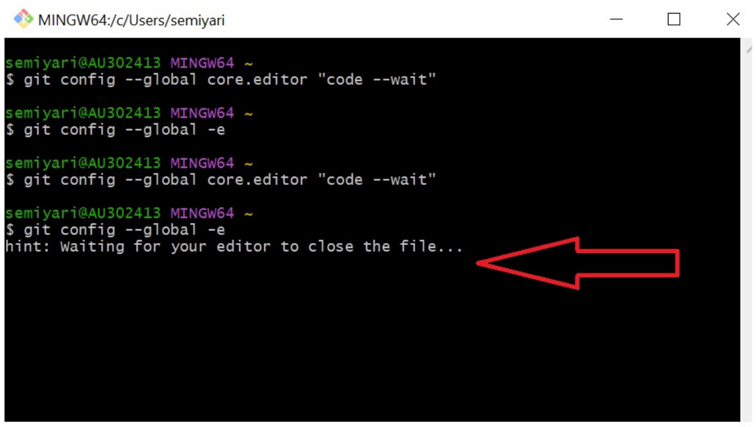
the new `VSCode`.
All these configuration setting are store in a text file.
We can edit that file using our default editor. Which in this case is `VSCode`.
So we can type `git config --global -e`
This will open our default editor to edit all the global setting. If you type the code, you will see the 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 sec-


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
C: > Users > semiyari > .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]
    edittor = code --wait
    autocrlf = true
    editor = code --wait
[color]
    ui = true
```

tions

- ∗ [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`

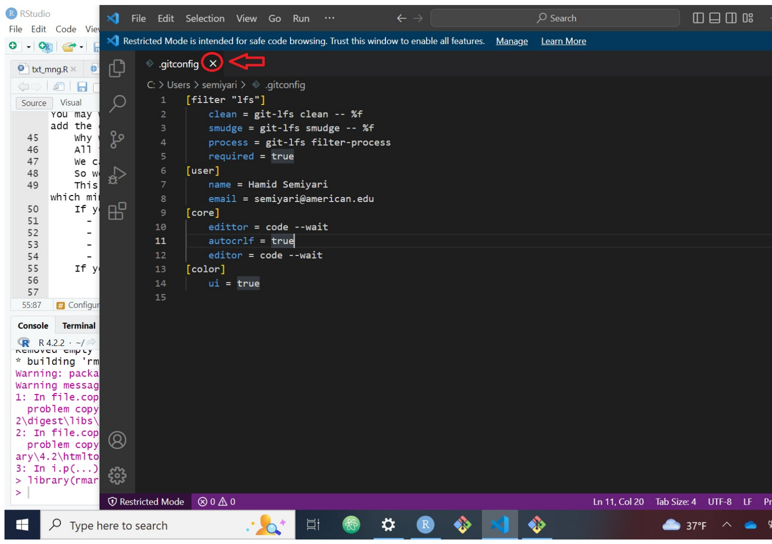


```
semiyari@AU302413 MINGW64 ~
$ git config --global core.editor "code --wait"

semiyari@AU302413 MINGW64 ~
$ git config --global -e

semiyari@AU302413 MINGW64 ~
$ git config --global core.editor "code --wait"

semiyari@AU302413 MINGW64 ~
$ git config --global -e
hint: Waiting for your editor to close the file...
```


Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
.gitconfig
C: > Users > semiyari > .gitconfig
Restricted Mode
Ln 11, Col 20 Tab Size: 4 UTF-8 LF
Type here to search

```
[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]
    edittor = code --wait
    autocrlf = true
    editor = code --wait
[color]
    ui = true
```

· 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`.

  ∗ Carrige Return `\r`: It means that the curser should **go back to the beginning of the line**.

  ∗ Line feed `\n`: It means that the curser 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.

– **Window:** `git config --global core.autocrlf true` (`crlf` is short for Carriage Return Line Fit)

  - **Mac or Linux:** `git config --global core.autocrlf input`

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 - 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