Setting Up Git

When we use Git on a new computer for the first time, we need to configure a few things. Below are a few examples of configurations we will set as we get started with Git:

- our name and email address,
- what our preferred text editor is,
- and that we want to use these settings globally (i.e. for every project).

On a command line, Git commands are written as `git verb options`, where `verb` is what we actually want to do and `options` is additional optional information which may be needed for the `verb`. So here is how Dracula sets up his new laptop:

```
$ git config --global user.name "Vlad Dracula"
$ git config --global user.email "vlad@tran.sylvan.ia"
```

{: .language-bash}

Please use your own name and email address instead of Dracula's. This user name and email will be associated with your subsequent Git activity, which means that any changes pushed to GitHub, BitBucket, GitLab or another Git host server after this lesson will include this information.

For this lesson, we will be interacting with GitHub and so the email address used should be the same as the one used when setting up your GitHub account. If you are concerned about privacy, please review GitHub's instructions for keeping your email address private.

Keeping your email private

If you elect to use a private email address with GitHub, then use that same email address for the `user.email` value, e.g. `username@users.noreply.github.com` replacing `username` with your GitHub one. {: .callout}

Line Endings

As with other keys, when you hit Enter or <- or on Macs, Return on your keyboard, your computer encodes this input as a character.

Different operating systems use different character(s) to represent the end of a line. (You may also hear these referred to as newlines or line breaks.) Because Git uses these characters to compare files, it may cause unexpected issues when editing a file on different machines. Though it is beyond the scope of this lesson, you can read more about this issue in the Pro Git book.

You can change the way Git recognizes and encodes line endings using the `core.autocrlf` command to `git config`. The following settings are recommended:

On macOS and Linux:

```
$ git config --global core.autocrlf input
```

{: .language-bash}

And on Windows:

```
$ git config --global core.autocrlf true
```

{: .language-bash}

{: .callout}

Dracula also has to set his favorite text editor, following this table:

Editor	Configuration command
Atom	`$ git config --global core.editor "atom --wait"`
nano	`$ git config --global core.editor "nano -w"`
BBEdit (Mac, with command line tools)	`$ git config --global core.editor "bbedit -w"`
Sublime Text (Mac)	`$ git config --global core.editor "/Applications/Sublime\ Text.app/Contents/SharedSupport/bin/subl -n -w"`
Sublime Text (Win, 32-bit install)	`$ git config --global core.editor "'c:/program files (x86)/sublime text 3/sublime_text.exe' -w"`
Sublime Text (Win, 64-bit install)	`$ git config --global core.editor "'c:/program files/sublime text 3/sublime_text.exe' -w"`
Notepad (Win)	`$ git config --global core.editor "c:/Windows/System32/notepad.exe"`
Notepad++ (Win, 32-bit install)	`$ git config --global core.editor "'c:/program files (x86)/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"`

Editor	Configuration command
Notepad++ (Win, 64-bit install)	`$ git config --global core.editor "'c:/program files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"`
Kate (Linux)	`$ git config --global core.editor "kate"`
Gedit (Linux)	`$ git config --global core.editor "gedit --wait --new-window"`
Scratch (Linux)	`$ git config --global core.editor "scratch-text-editor"`
Emacs	`$ git config --global core.editor "emacs"`
Vim	`$ git config --global core.editor "vim"`
VS Code	`$ git config --global core.editor "code --wait"`

It is possible to reconfigure the text editor for Git whenever you want to change it.

Exiting Vim

Note that Vim is the default editor for many programs. If you haven't used Vim before and wish to exit a session without saving your changes, press Esc then type `:q!` and hit Enter or <- or on Macs, Return. If you want to save your changes and quit, press Esc then type `:wq` and hit Enter or <- or on Macs, Return. {: .callout}

Git (2.28+) allows configuration of the name of the branch created when you initialize any new repository. Dracula decides to use that feature to set it to `main` so it matches the cloud service he will eventually use.

```
$ git config --global init.defaultBranch main
```

{: .language-bash}

Default Git branch naming

Source file changes are associated with a "branch." For new learners in this lesson, it's enough to know that branches exist, and this lesson uses one branch.
By default, Git will create a branch called `master` when you create a new repository with `git init` (as explained in the next Episode). This term evokes the racist practice of human slavery and the software development community has moved to adopt more inclusive language.

In 2020, most Git code hosting services transitioned to using `main` as the default branch. As an example, any new repository that is opened in GitHub and GitLab default to `main`. However, Git has

not yet made the same change. As a result, local repositories must be manually configured have the same main branch name as most cloud services.

For versions of Git prior to 2.28, the change can be made on an individual repository level. The command for this is in the next episode. Note that if this value is unset in your local Git configuration, the `init.defaultBranch` value defaults to `master`.

{: .callout}

The five commands we just ran above only need to be run once: the flag `--global` tells Git to use the settings for every project, in your user account, on this computer.

You can check your settings at any time:

```
$ git config --list
```

{: .language-bash}

You can change your configuration as many times as you want: use the same commands to choose another editor or update your email address.

Proxy

In some networks you need to use a proxy. If this is the case, you may also need to tell Git about the proxy:

```
$ git config --global http.proxy proxy-url
$ git config --global https.proxy proxy-url
```

{: .language-bash}

To disable the proxy, use

```
$ git config --global --unset http.proxy
$ git config --global --unset https.proxy
```

{: .language-bash} {: .callout}

Git Help and Manual

Always remember that if you forget the subcommands or options of a `git` command, you can access the relevant list of options typing `git <command> -h` or access the corresponding Git manual by typing `git <command> --help`, e.g.:

```
$ git config -h
$ git config --help
```

{: .language-bash}

While viewing the manual, remember the `:` is a prompt waiting for commands and you can press Q to exit the manual.

More generally, you can get the list of available `git` commands and further resources of the Git manual typing:

```
$ git help
```

{: .language-bash}

{: .callout}