You can check to see if this installation worked by opening up the terminal window and typing:

git --version

To take advantage of Git superpowers, we have to add a hidden directory called .git/ to our project directory, which contains all of the data Git needs to operate. This is called “initializing.”

Next, navigate to the GA-Blog directory you just created and run this command:

$ git init

To see the .git/ directory, you need to run ls -a from the command line.

git status, which asks Git to give us an update on our project’s status.

$ git status

To create a new text file called you'll use the touch command:

$ touch post.txt

## Staging

Like the terminal, Git doesn’t make any assumptions about what changes you want to save and when you want to save them. Instead, you need to explicitly tell it what to do.

To add this change to your next commit, you'll use the git add command.

$ git add post.txt

The command is add, but we describe the operation by saying that the file has been “staged.” In other words, it has been added to the list of changes that will be officially saved with our next commit.

The files on this list aren’t final, and any of these changes can be removed, or “unstaged.”

Add the change to your next commit using the git add command.

$ git add post.txt

## It’s Not Saved Yet!

Once we’re ready to officially record this version of our project, type:

$ git commit -m "created a new post.txt file"

The -m option allows you to include a message that describes the changes you made for your collaborators or future you.

These should be short but descriptive and clearly indicate what changes each commit makes to the project.

Note: In some versions of Git CLI, you will be asked to provide configuration details before committing. [Here](https://git-scm.com/book/en/v2/Customizing-Git-Git-Configuration) is a link to additional Git configuration directions.

## Let’s Run That Back

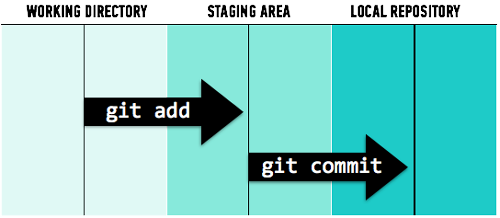
Git allows you to add changes to your project in the local repository with two steps:

$ git add <your-file-name>

$ git commit -m "message"

This might be a strange concept to non-developers who are used to clicking a save icon and moving on. For developers, a two-step save process provides benefits, such as making incremental edits to a challenging build and reviewing each file before committing it.

If you’re curious, you can read more about how developers use the two-step saving process to their advantage [here](https://softwareengineering.stackexchange.com/questions/69178/what-is-the-benefit-of-gits-two-stage-commit-process-staging).



## Commit History

Further into a project, after you’ve made many commits, you might want to look back and see a timeline of those changes.

Git allows developers to view a list of commits, the submission date, the author, the commit message, and a unique number that identifies the commit, called an SHA.

## The Commit History Command

To view the timeline of changes, you can run:

$ git log

This will yield a list of entries that looks like this:

commit 6d33f525a09b9918f75188db164ea2722039830b

Author: Sarah <sarah@gmail.com>

Date: Wed Jan 28 17:44:03 2015 -0500

added a new post

## Putting It All Together: Prep

* Open your terminal and navigate to the directory you’ve been using to store information about this course. If you don’t have one yet, make one.
* Create a directory inside of that called git-practice.
* Navigate into that new directory. You can make sure you’re in the right place using the pwd command.
* Use git init to create a Git repository in the git-practice directory.

Note: Before running git init, make sure you’re not already inside another Git repository. Type git status. If you see fatal: Not a git repository (or any of the parent directories): .git, then you know you’re good to go and you can safely run git init within this folder.

## Putting It All Together: Add and Commit

* Staying in the git-practice directory, run the ls -a command to see the .git directory you’ve just created.
* Create a new file called README.txt and run git status. What output do you get?
* Use the git add README.txt command to add the new file to the staging area.
* Run git status again. How has the output changed?
* Now, commit the changes you made using git commit -m "...". Give the commit an appropriate message. Remember, it should be short but descriptive.

You did it! In due time you’ll be able to complete this process with your eyes closed (although we don’t recommend it).

https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

## So, How Do Developers Do This?

Consider a project like Node.js, a JavaScript framework. Node.js is completely open source, which means that anyone can read (and even copy) the code that makes it work — including you!

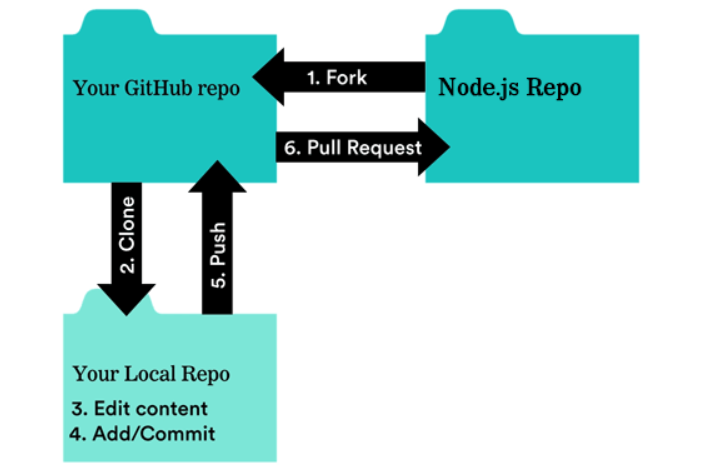
The source code is publicly available [here](https://github.com/nodejs/node) on GitHub. If you visit the main repository, you’ll see that there are more than 2,000 contributors who have committed changes to Node.js.

Although it’s open source and anyone can read or contribute to its code, Node.js is maintained by a company called Joyent. These contributors don’t have the ability to edit the original Joyent repository — that wouldn’t be very efficient or safe. Someone could accidentally make a change that conflicts with someone else’s contributions, causing the code to break. Changes need to be inspected and approved by Joyent before they can officially be added to the project.

## The GitHub Flow

The workflow for contributing to an open-source product or your dev team’s project comprises the following steps:

1. Forking
2. Cloning
3. Editing
4. Adding/committing
5. Pushing
6. Submitting a pull request



## Step 1: Forking

To add a copy of someone else’s GitHub repository to your GitHub account, fork it by clicking the Fork button in the upper right-hand corner.

This forked repository is not perfectly identical, but it includes all of the same source files, issues, and commit history.

By forking Joyent’s repository, for example, you now have a full working copy of the Node.js source code to play with. This way, when you break something (which you will), the main repository won’t be affected.

## Step 2: Cloning

To make a local copy of a fork, you’ll clone the repository. This will save the code on your machine so you can edit it.

To do so, open your terminal, navigate to where you’d like to store the repository, then type:

git clone https://url-to-clone

You can find the URL to clone by clicking the green button that says “Clone or download.”

Hint: If you’re following along in Git Bash on Windows, the commands to copy and paste a repository are a little different than the default Windows copy/paste commands. Use control + insert to copy and shift + insert to paste.

## Step 3: Editing

This is when you make your revolutionary edits to the code and reimagine how Node.js works. Be bold — it's your time to shine!

You’ll do this using a text editor of your choice. [Atom](https://atom.io/) and [Sublime](https://www.sublimetext.com/) are some of the most popular.

## Step 4: Adding and Committing

Remember, you’re editing the code on your local copy of the repository. We know that any time we do this, we need to use some very important Git commands so that our local copy is protected if we goof up.

Recall these commands:

$ git add <your-file-name>

$ git commit -m "message"

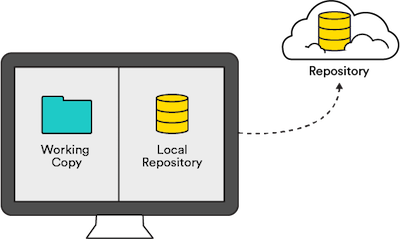
## Step 5: Pushing

Once you’ve committed these changes, your local repository will differ from your remote repository.

To update your remote repository on GitHub, you have to push those changes using the git push origin master command.

You don't need to worry about the origin and master part just yet. However, if you’re curious, here’s a brief overview:

* origin is a shortcut for the URL of your default remote repository (in this case, the repository on GitHub). You can have many remotes if you want, but we’re only going to work with one for now.
* master refers to the branch on your remote repository where you are currently adding your changes. Again, for now, we’re just going to be working on the master branch.



## Step 6: Submitting a Pull Request

At this point, your local and remote repositories contain the changes you’ve made. If you want to share these changes with the original repository owner, Joyent, you can submit a pull request.

A pull request effectively says, “Hello, maintainer of Project X. I made some changes here in my forked copy, and I think they’re good ones. You should add them to your repository.”

Pull requests are a GitHub feature, so you’ll need to head back to the browser to submit them.