Getting Started with Git and GitHub

Never fear losing work with this professional versioning system

Getting Started with Git and GitHub

In this tutorial, we walk through the process for using Git locally on your personal computer, and using GitHub to back it up. We walk through creating your personal GitHub account, setting up Git on your computer, starting your first Git repository, and connecting that repository to a GitHub repository.

This tutorial assumes that you've completed the lessons on Learn Command Lineand Learn Git. Now, prepare to use those skills on your personal computer! If some steps in this tutorial are confusing, have no fear; it will all come together by the end.

What are Git and GitHub?

This tutorial refers to Git and GitHub repeatedly. *Git* is a widely-used version control system used to manage code. Git allows you to save drafts of your code so that you can look back at previous versions and potentially undo complicated errors. A project managed with Git is called a *Git repository*.

GitHub is popular hosting service for Git repositories. GitHub allows you to store your local Git repositories in the cloud. With GitHub, you can backup your personal

files, share your code, and collaborate with others.

In short, GitHub is a tool for working with Git. There are other services to host Git repositories, but GitHub is a trusted, free service used by organizations across the world, big and small.

Create a GitHub Account

To use GitHub, you will need a GitHub account.

In your own browser:

> Open a new browser tab
>
> Navigate to https://github.com/
>
> Create an account

If you already have GitHub account, continue to the next exercise.

After you sign up, you will receive a verification e-mail. Be sure to verify your e-mail address to GitHub by following the instructions in that e-mail.

Git Setup for Mac and Windows

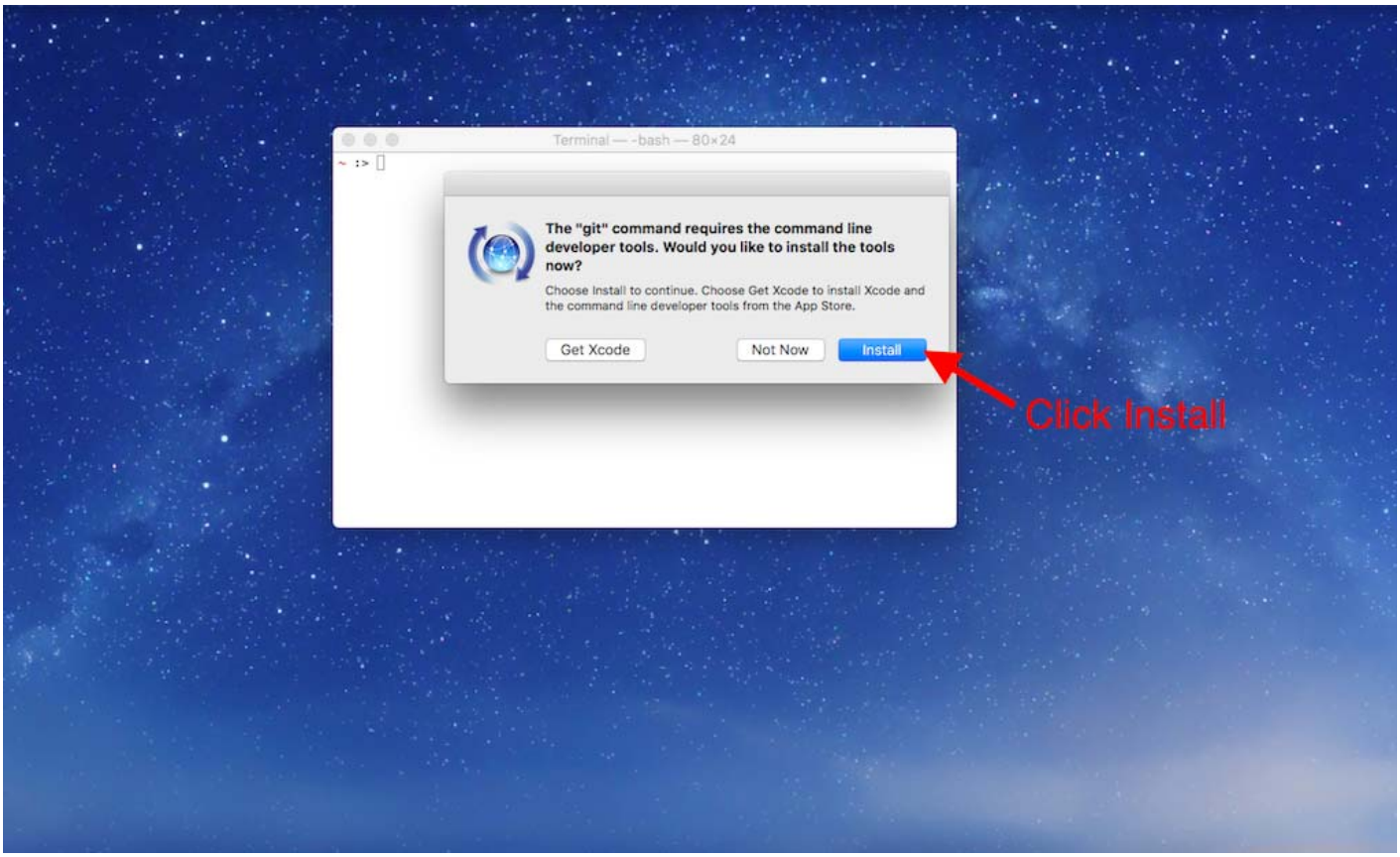
Next, we will set up Git on your personal computer. Follow the instructions for your operating system.

Mac users:

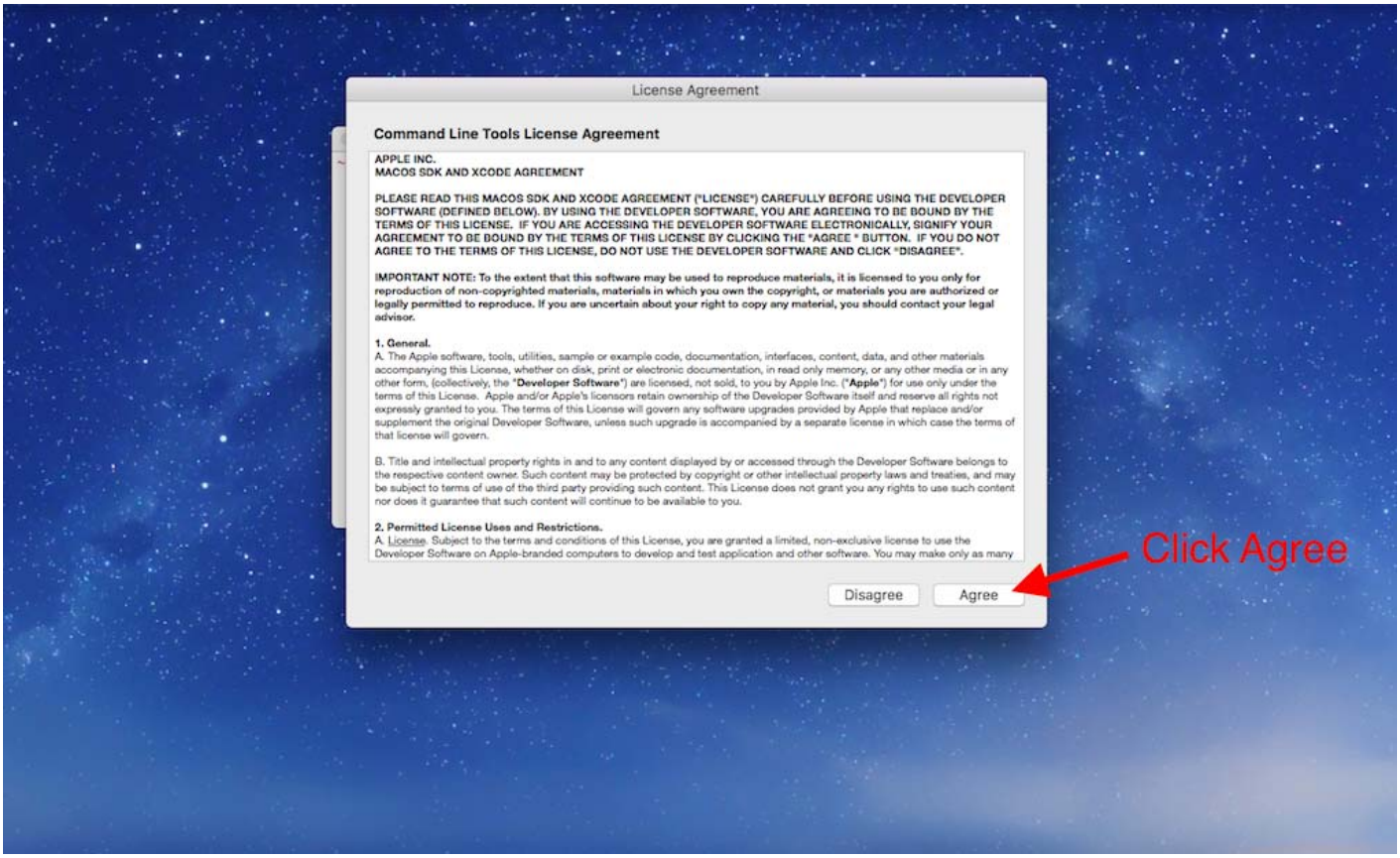
1. Launch the ***Terminal*** application. You can find it in **/Applications/Utilities/**. You can also use the ***Spotlight*** search tool (the little magnifying glass in the top right of your screen) to search for ***Terminal***. Once ***Spotlight*** locates it, click on the result that says ***Terminal***.

2. When ***Terminal*** opens, type in `git` and press enter.

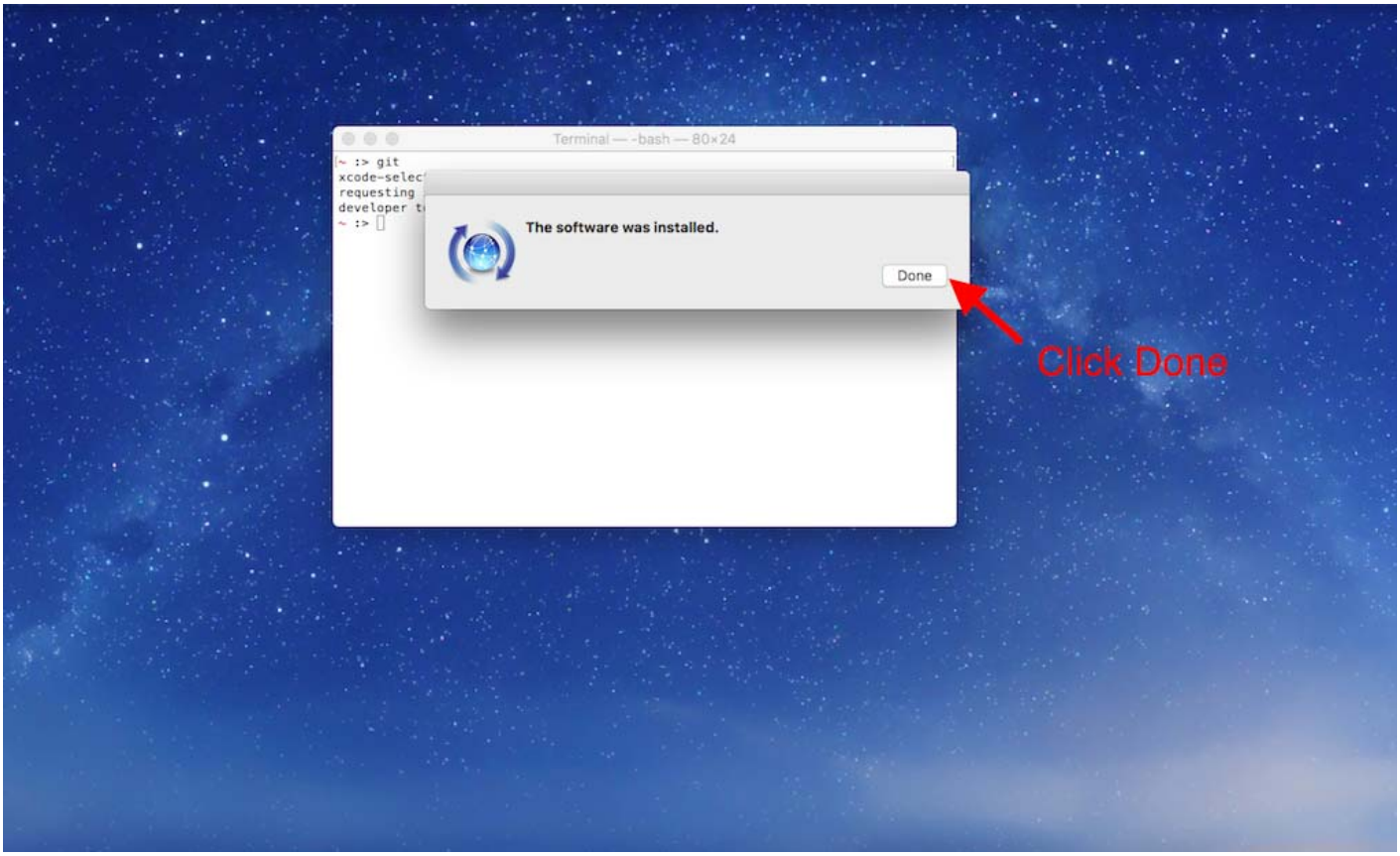
3. If you don't already have Git installed, a dialog will appear saying that "The 'git' command requires the command line developer tools. Would you like to install the tools now?" Click "Install".



Then click "Agree to the Terms of Service" when requested.



4. When the download finishes, the installer will go away on its own signifying that Git is now installed! Click “Done” to finish the installation process.



5. Navigate to GitHub’s articles on setting up your Git username and email and follow the instructions for each using Terminal.

6. GitHub offers two authentication options, HTTPS and SSH, to keep your work secure. This is a security measure which prevents anyone who isn’t authorized from making changes to your GitHub repository. In this article, we will use HTTPS. Navigate to GitHub’s article on caching your password and follow the instructions to configure your computer to be able to use HTTPS.

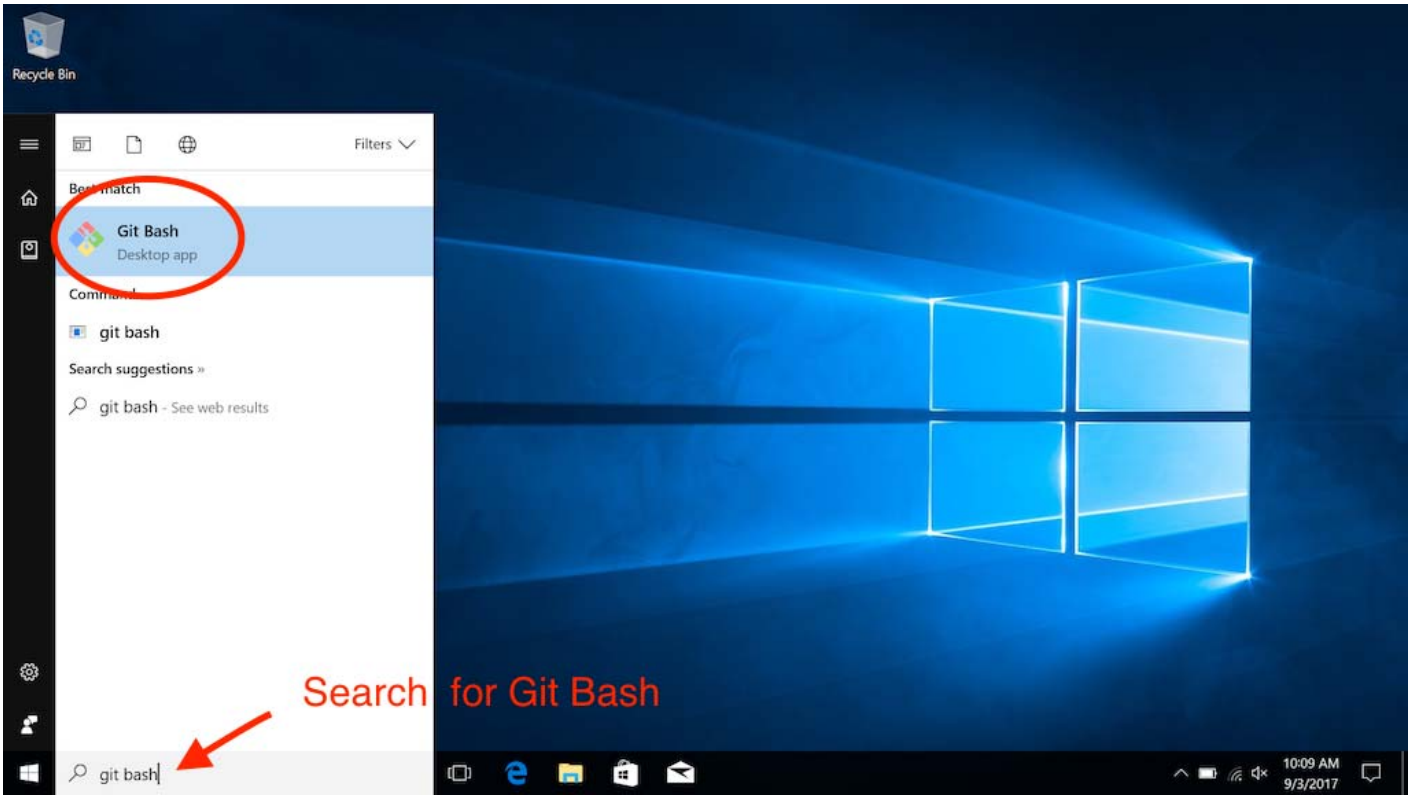
Now skip down to the “Try it Out!” section below.

Windows users:

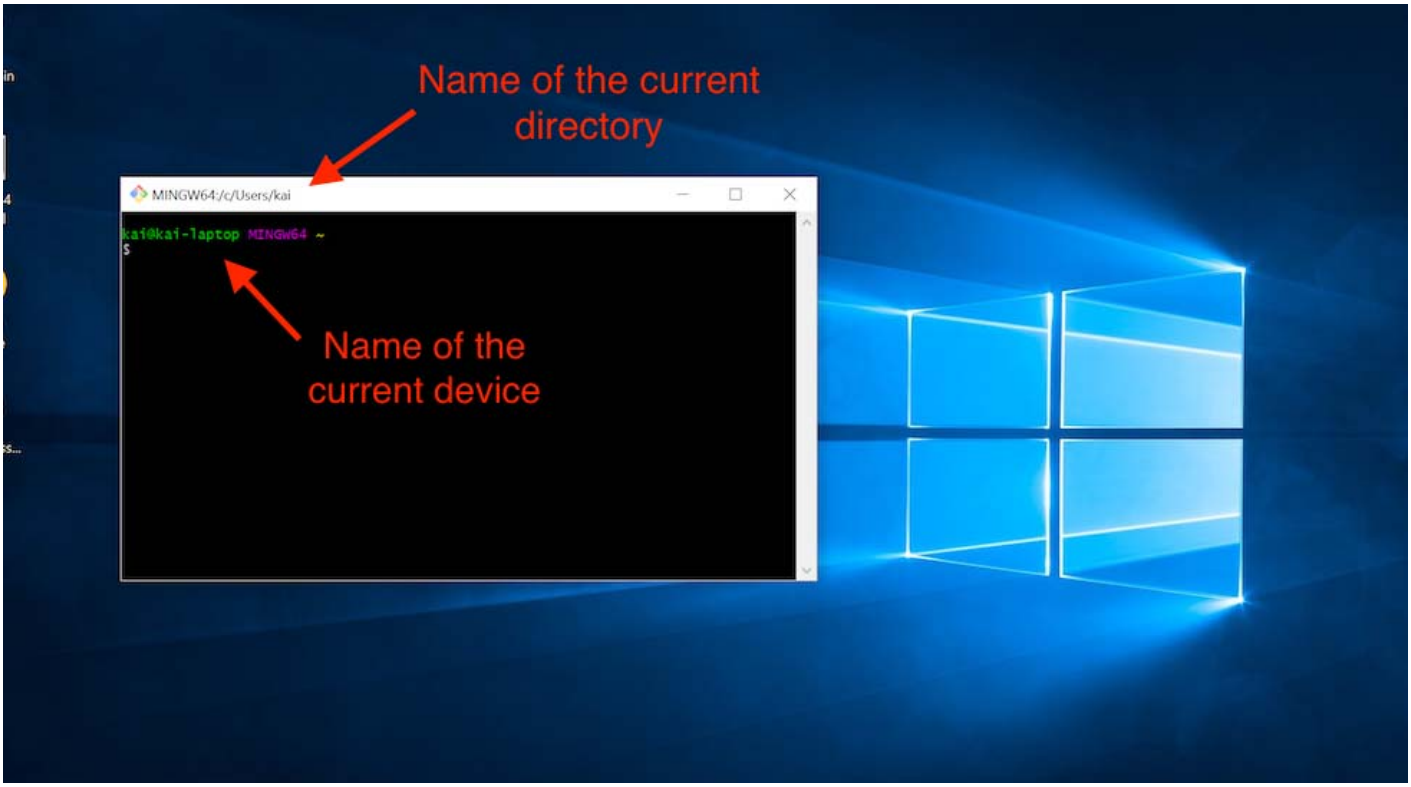
This portion of the guide assumes you have already installed a program called Git Bash which allows us access to Git on Windows. If you have not installed Git Bash, please refer to the previous tutorial on Command Line Interface (CLI) Setup and

follow the instructions for installing Git Bash on Windows. Once you complete that you can continue with this guide.

1. Open the Start menu and search for the app, git bash. You should see 'Git Bash Desktop app' appear. Press Enter or click on the Git Bash icon to open the app.

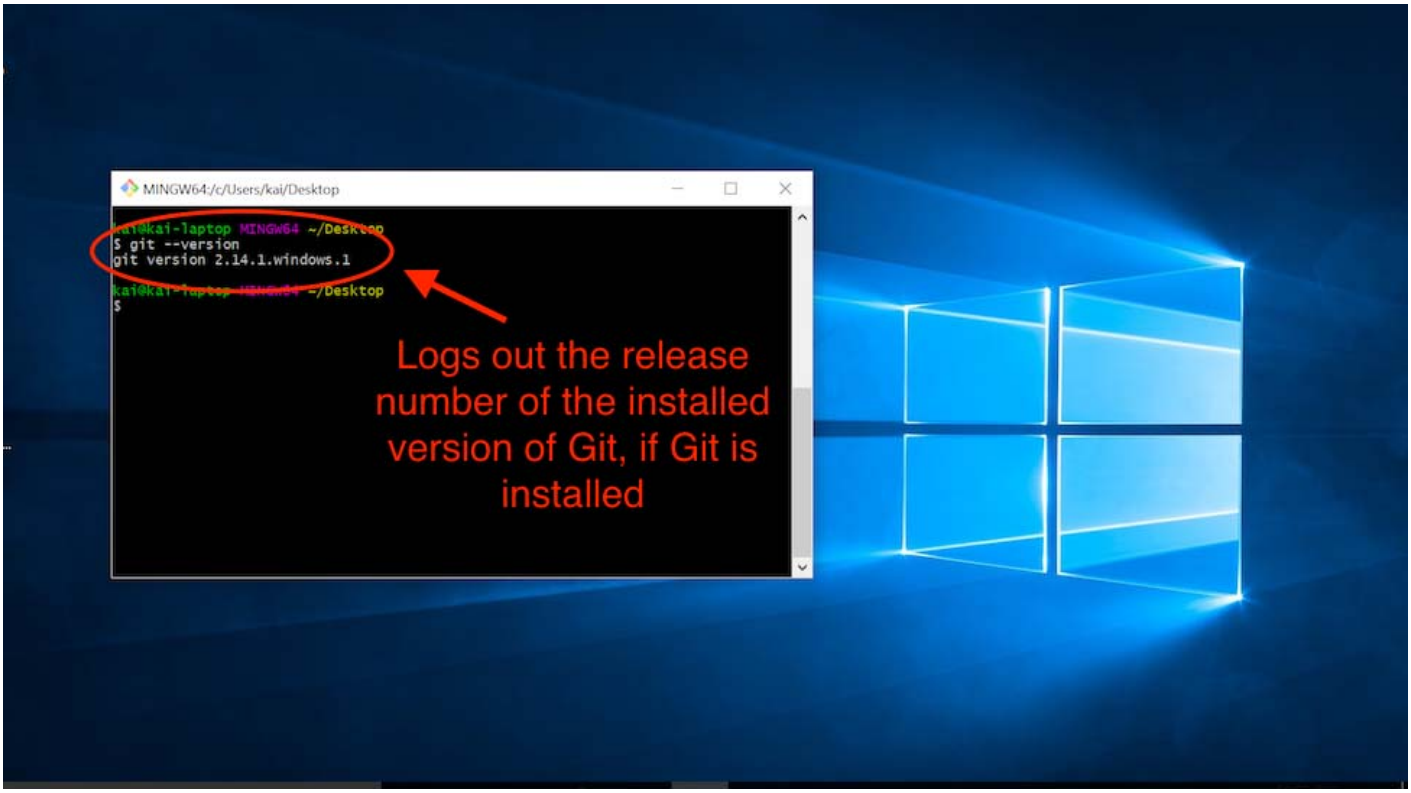


A new window will open that looks like this:



This window is our CLI, where we will use our Git commands.

2. If you want to make sure that Git is installed, run `git --version` in the CLI. You should see a response that gives you the version of Git installed. It will look like this:



Git can now be used in the Git Bash app!

3. Navigate to GitHub's articles on setting up your Git username and email and follow the instructions for each using Git Bash.

4. GitHub offers two authentication options, HTTPS and SSH, to keep your work secure. This is a security measure which prevents anyone who isn't authorized from making changes to your GitHub repository. In this article, we will use HTTPS. Navigate to GitHub's article on caching your password and follow the instructions to configure your computer to be able to use HTTPS.

Try It Out!

Now you have everything you need to practice your Git skills on your local computer. Take a moment to run the commands below to initialize a Git

repository. We will use this Git repository again later in this tutorial so make sure you complete these steps exactly as described.

- `mkdir git_practice` to make a new directory to practice.
- `cd git_practice` to make the new directory your working directory.
- `git init` to turn the current, empty directory into a fresh Git repository.
- `echo "Hello Git and GitHub" >> README.txt` to create a new README file (more on this later) with some sample text.
- `git add README.txt` to add the new file to the Git staging area.
- `git commit -m "First commit"` to make your first commit with the new README file.

Your First Remote Repository on GitHub

Finally, we'll create a repository on GitHub and then link it to a local repository on your computer. This allows you to backup your work constantly and safely, so you never need to worry about losing your work again!

Now, let's connect our local Git repository to GitHub.

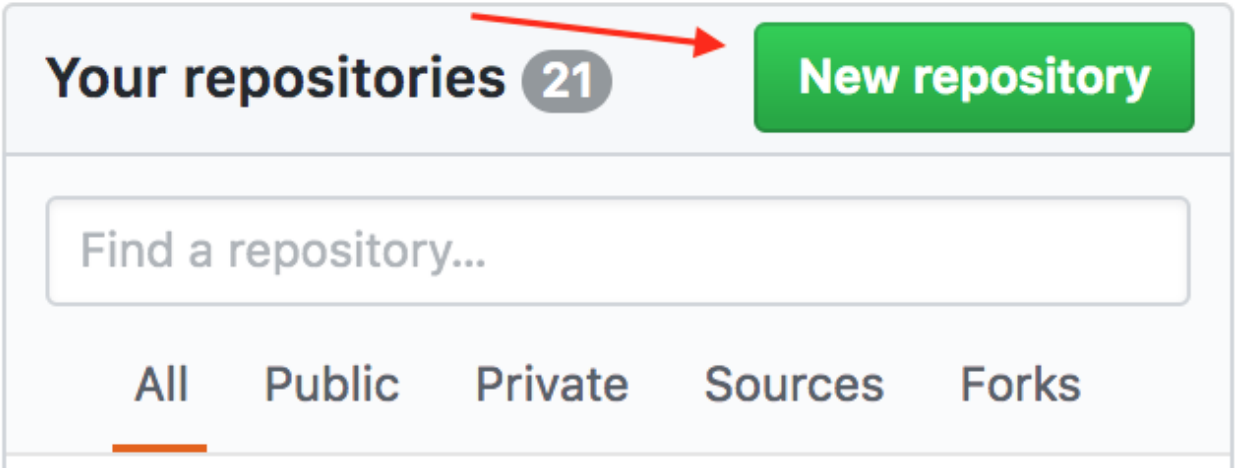
Instructions

1. In your Command Line Interface, make sure your current working directory is your new Git repository. Navigate there if not.

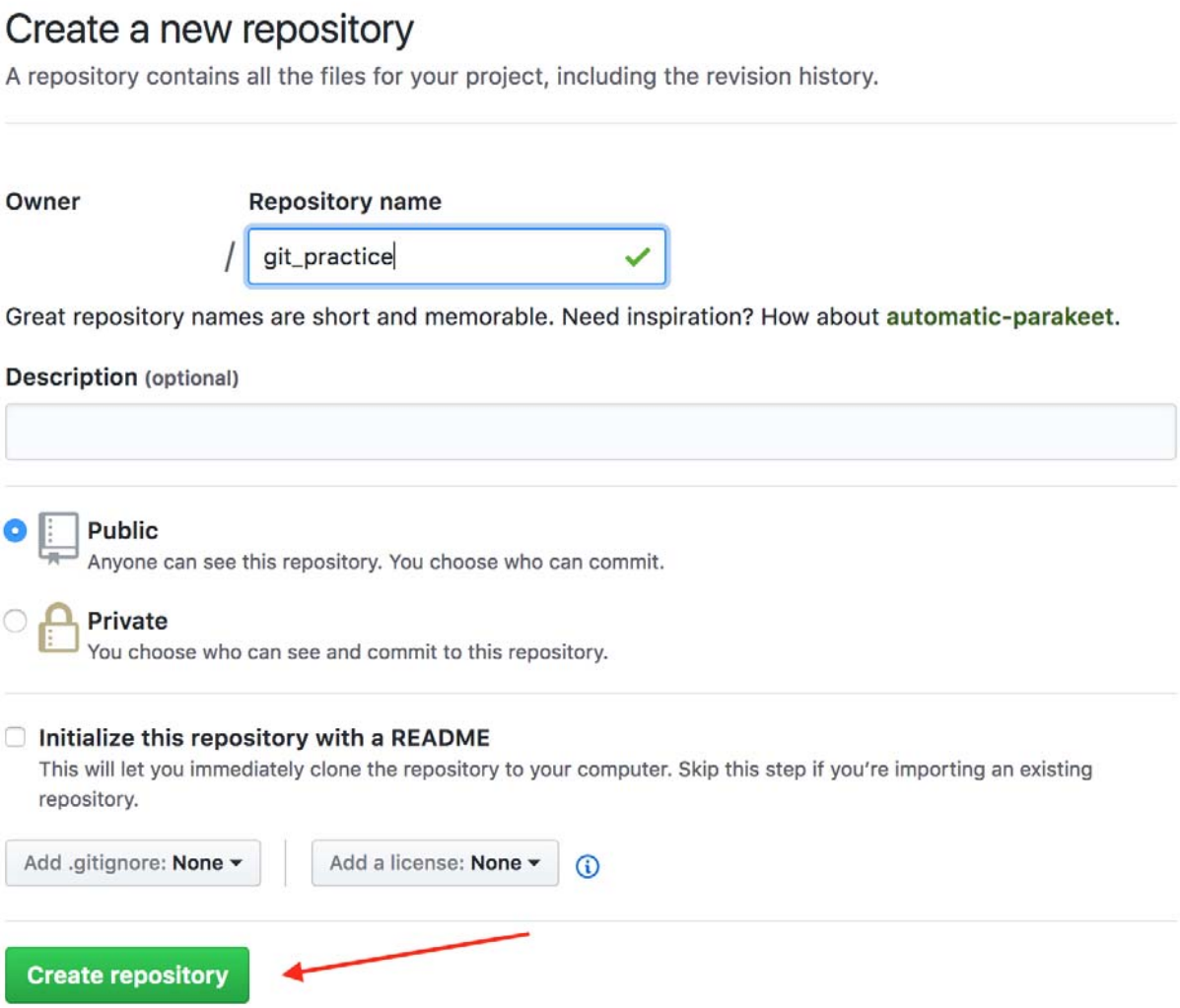
2. Check the status of which files and folders are new or have been edited. There should be no files modified.

```
$ git status
```

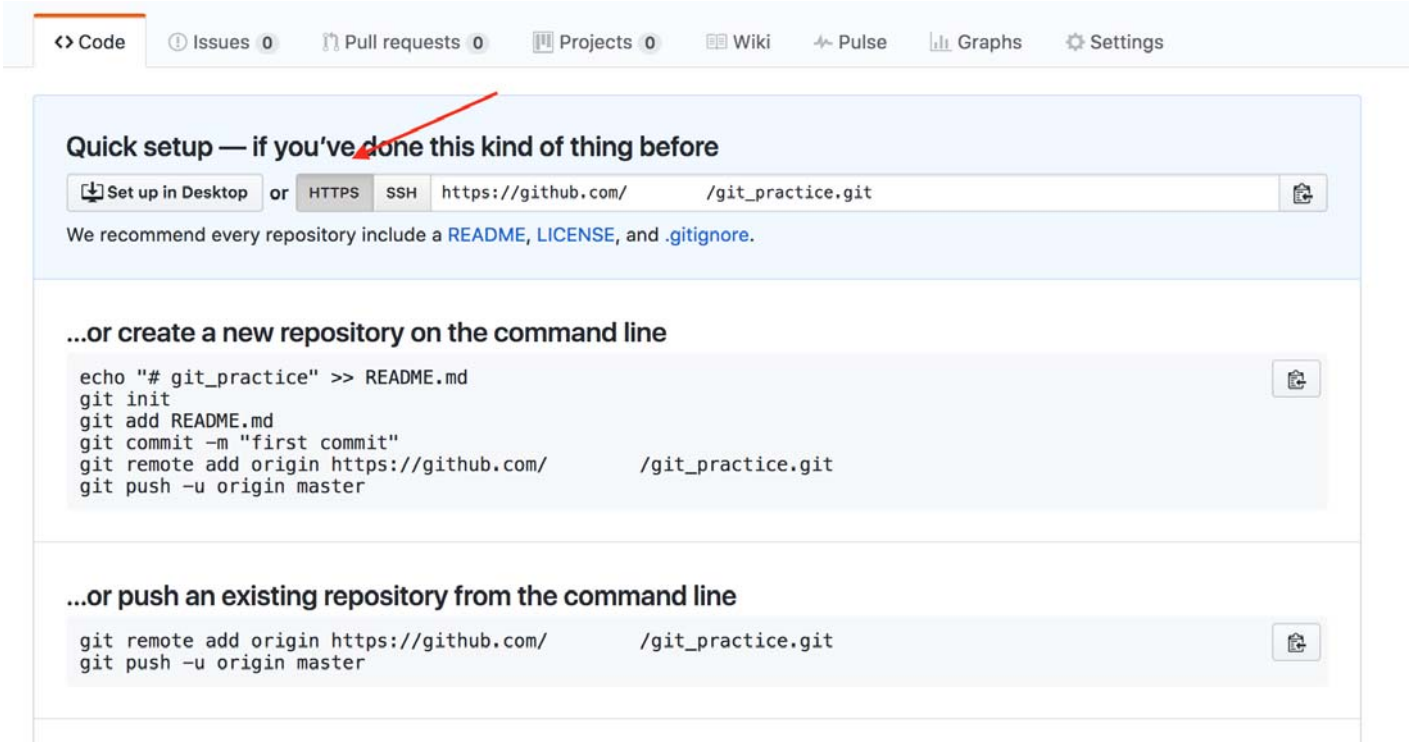
3. On GitHub, create a new repository by clicking the **New repository** button on the home page.



4. On the new repository page, give your repository a name. It's not necessary, but it would be convenient to name it the same as the directory, **git_practice**. After naming the repository, click **Create repository**.

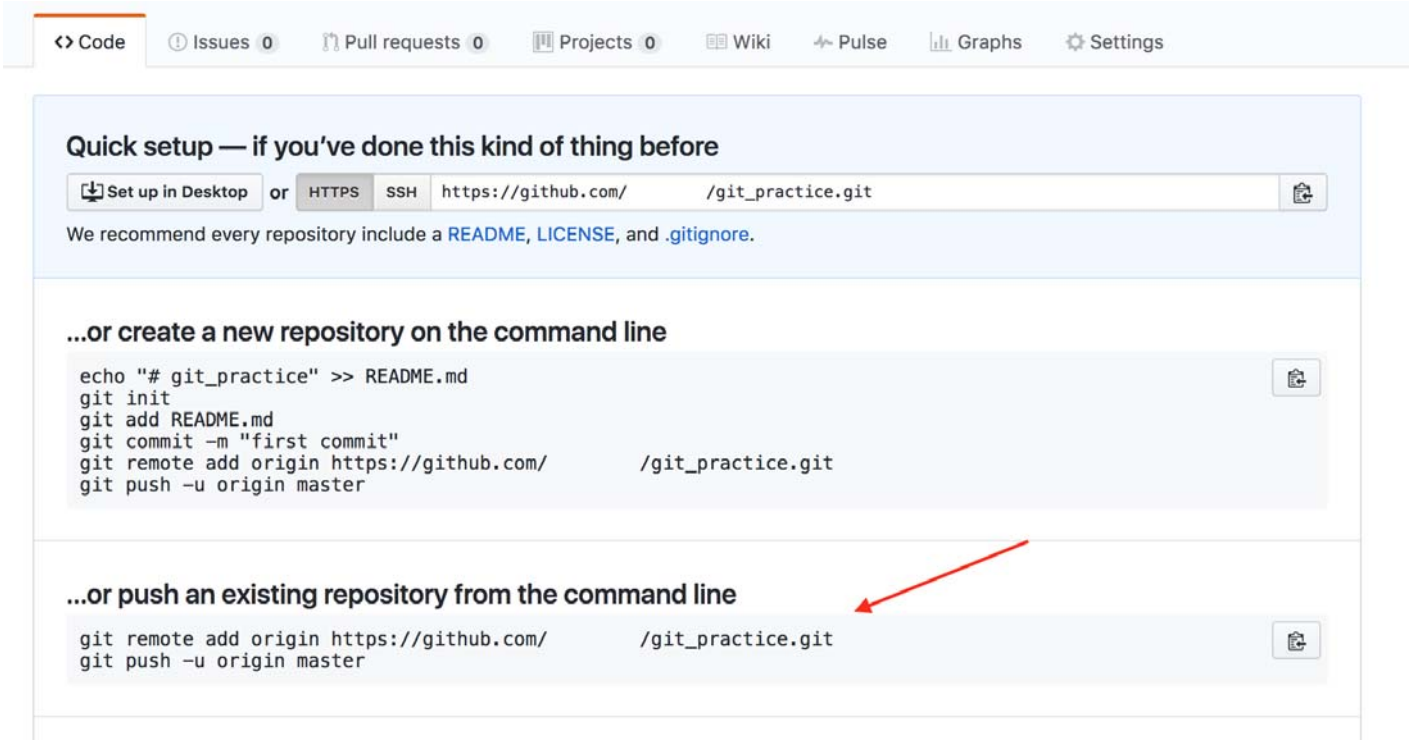


5. After creating a repository, GitHub displays the repository page. At the top of the page, make sure "HTTPS" is selected.



6.The repository is empty, so it's time to connect it to your existing work. Copy the Git commands on the GitHub page, under the title "...or push an existing repository from the command line", and paste them into your Command Line Interface. Running these commands will add a remote repository, and then push your local repository to the remote repository.

When asked for a username and password, type in your GitHub username and password and press `enter` after each. Don't be alarmed if you can't see the characters you are typing, they are intentionally hidden as a security measure.



Note: If you set up two-factor authentication with GitHub (don't worry if you didn't), follow these instructions to generate an OAuth token to be used instead of your password in bash. By default, GitHub does not set up two-factor authentication. If you are not familiar with two-factor authentication, you don't have to generate an OAuth token.

7. Once your Command Line Interface reports that the push is complete, refresh the page on GitHub. You should now see the text you wrote earlier in the README file, "Hello Git and GitHub."

GitHub automatically displays the contents of a file named **README.txt** if it exists in the repository. The README file is the perfect place to write a description of your project.

There you have it! Your first GitHub repository, linked to your local Git repository. You've taken some huge leaps, so be proud! Now you can use your knowledge of Git to track progress on your local computer, and push that progress to GitHub whenever you want. You can rest easy knowing that each step of your progress is safely stored in GitHub.