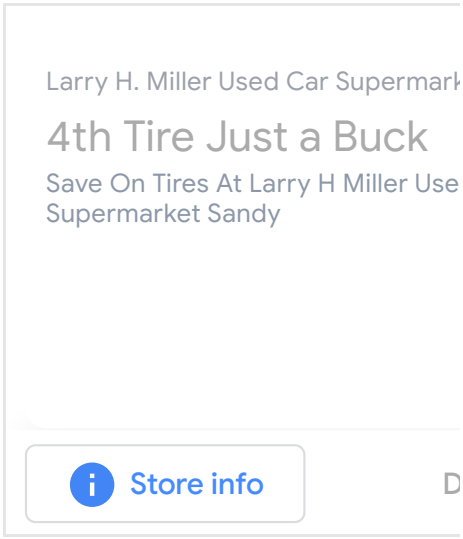
JUNE 30, 2020 / #GIT

Git Clone Branch – How to Clone a Specific Branch

Bolaji Ayodeji

Unlike older centralized version

control systems such as SVN and CVS, Git is distributed. Every developer has the full history and control of their code locally or remotely. They can also access or manipulate several parts of the code as they deem fit from different locations.



Since Linus Torvalds (the famous creator of the Linux operating system kernel) created Git in 2005 for Linux kernel development, it has become the most widely used modern version control system in the world.

In this article, I'll introduce you to the Git clone and Git branch workflows and I'll show you how you can clone a specific branch based on your needs. Let's begin! ?

Prerequisites

- Basic knowledge of the terminal
- Ability to type commands in the terminal
- Git installed (I'll still show you how)
- A GitHub account
- A smile on your face (Put up that smile friend ?)

Quick

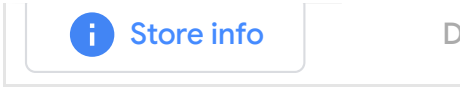
Introduction to Git and GitHub

According to Wikipedia,

> **Git** is a distributed version control system designed to track changes to a project (code) in software development. It is intended to enforce coordination, collaboration, speed, and efficiency among developers.

GitHub, on the other hand, is a web-based hosting service for version control using Git. It offers all of the distributed version control and source code management functionality of Git as well as

adding more features for computer code.



How to Install Git on Windows

Download and install the latest Git for Windows Installer here.

How to Install Git on Linux

Here are the commands based on your Linux distro:

Debian or Ubuntu

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

Fedora

```
sudo dnf install git
```

CentOS

```
sudo yum install git
```

Arch Linux

```
sudo pacman -Sy git
```

Gentoo

```
sudo emerge --ask --verbose dev-
```

How to Install Git on a Mac

Download and install the latest Git for Mac installer here.

Or you can type this

command:

```
brew install git
```

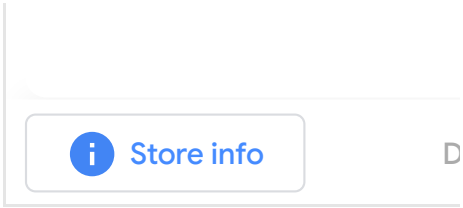
Now that we've got Git installed, let's move on to the tutorial.

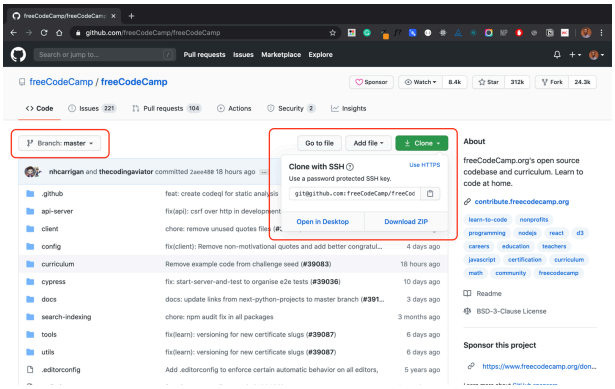
Introduction to Git Clone

Git allows you to manage and version your project(s) in a "repository". This repository is stored on a web-based hosting service for version control, like GitHub.

You can then clone this repository to your local machine and have all the files and branches locally (I'll explain more about branches

soon).





For example, you can clone freeCodeCamp's repository with SSH like so:

```
git clone git@github.com:freeCoc
```

Introduction to Git Branches

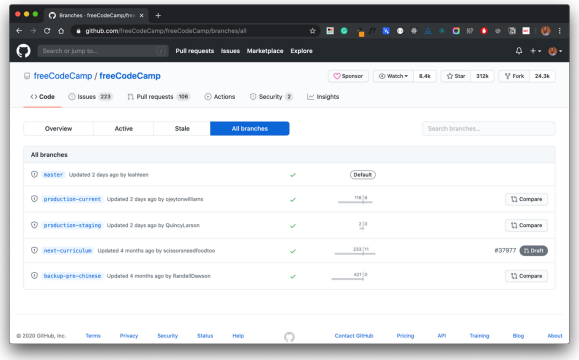
When working on a project, you will likely have different features. And multiple

contributors will be working on this project and its features.

Branches allow you to create a "playground" with the same files in the `master` branch. You can use this branch to build independent features, test new features, make breaking changes, create fixes, write docs or try out ideas without breaking or affecting the production code. When you're done, you merge the branch into the production `master` branch.

Branching is a core concept in Git which is also used in GitHub to manage workflows of different versions of one project. The `master` branch is always the default branch in a

repository that is most often considered "production and deployable code". New branches like `passwordless-auth` or `refactor-signup-ux` can be created from the `master` branch.



All branches in freeCodeCamp's repository

How to Clone Git Branches

While you can clone repositories with the `git clone` command, keep in

mind that this clones the branch and the remote `HEAD`. This is usually `master` by default and includes all other branches in the repository.

So when you clone a repository, you clone the `master` and all other branches. This means you will have to checkout another branch yourself.

Let's say your task on a project is to work on a feature to add passwordless authentication to a user dashboard. And this feature is in the `passwordless-auth` branch.

You really don't need the `master` branch since your "feature branch" will be merged into `master`

afterward. How then do you clone this `passwordless-auth` branch without fetching all other branches with "a bunch of files you don't need"?

I created this sample repository to explain this. This repository holds a simple blog built with Nextjs and has four dummy branches:

- master
- dev
- staging
- passwordless-auth

In Nextjs, any file inside the folder `pages/api` is mapped to the `/api/*` path and will be treated as an API endpoint instead of a `page`. In our

repository, I have created different dummy APIs in this directory to make each branch different.

The `master` branch holds the file **pages/api/hello.js** while `passwordless-auth` holds the file **pages/api/auth.js**. Each file just returns a dummy text response. See `master` 's hello API response here (with a special message for you ?).

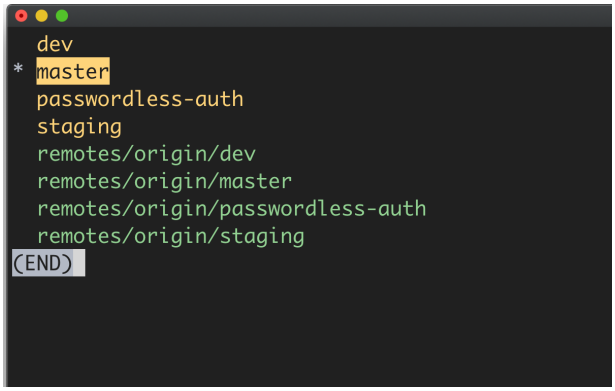
Let's clone the repository:

```
git clone git@github.com:BolajiA
```

This gives us access to all branches in this repository and you can easily toggle between each to see each

version and its files.

```
git branch -a
```

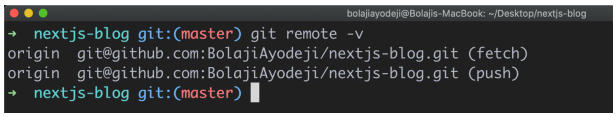


Wondering where the **remotes/origin/..** branches came from?

When you clone a repository, you pull data from a repository on the internet or an internal server known as the **remote**. The word origin is an alias created by your Git to replace the remote URL

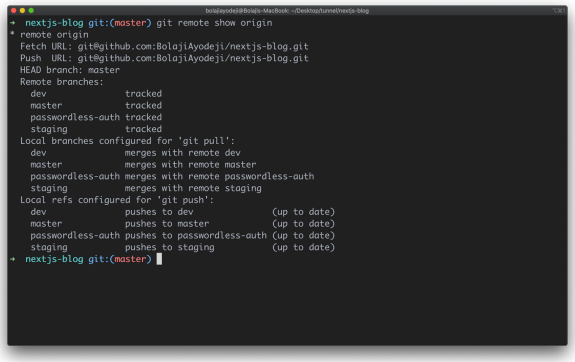
(you can change or specify another alias if you want).

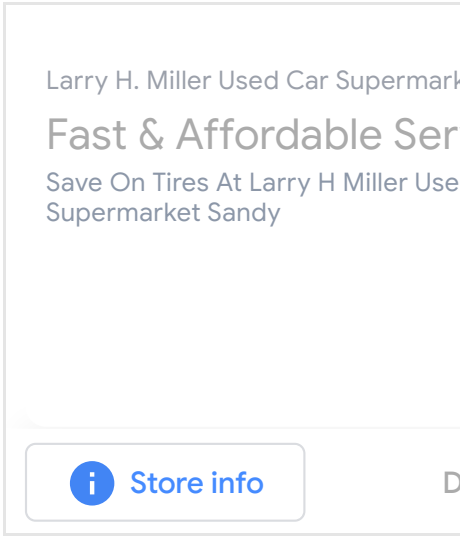
These **remotes/origin/..** branches point you back to the origin repository you cloned from the internet so you can still perform pull/push from the origin.



So when you clone `master` onto your machine, `remotes/origin/master` is the original `master` branch on the internet, and `master` is on your local machine. So you will pull/push from and to the `remotes/origin/master`.

In summary **Remote** is the URL that points you to the repository on the internet while **Origin** is an alias for this remote URL.





How to Clone a Specific Branch

Now let's clone a specific branch from our demo repository. There are two ways to clone a specific branch. You can either:

- Clone the repository,

fetch all branches, and checkout to a specific branch immediately.

- Clone the repository and fetch only a single branch.

Option One

```
git clone --branch <branchname>
```

or

```
git clone -b <branchname> <remot
```

Here **-b** is just an alias for **--branch**

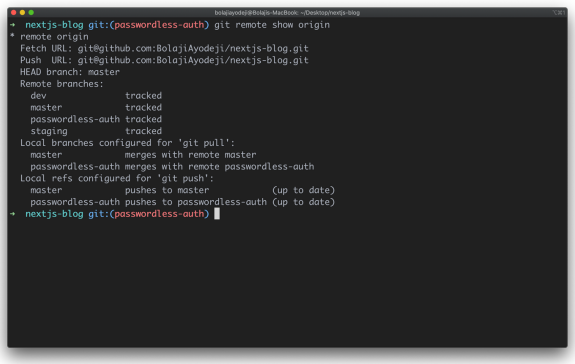
With this, you fetch all the branches in the repository, checkout to the one you

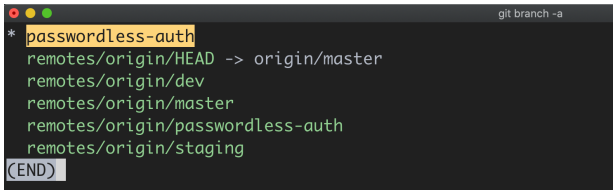
specified, and the specific branch becomes the configured local branch for `git push` and `git pull`. But you still fetched all files from each branch. This might not be what you want right? ?

Let's test it:

```
git clone -b passwordless-auth
```

This automatically configures `passwordless-auth` as the local branch but still tracks other branches.





Option Two

```
git clone --branch <branchname>
```

or

```
git clone -b <branchname> --sing
```

Here **-b** is just an alias for **--branch**

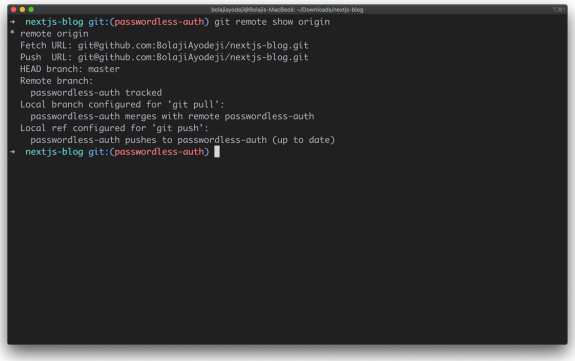
This performs the same action as option one, except that the `--single-branch` option was introduced in Git

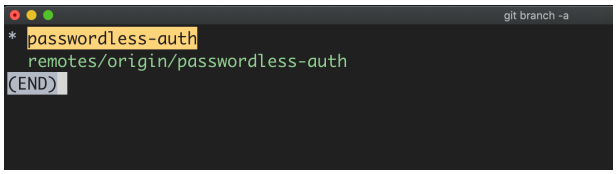
version 1.7.10 and later. It allows you to only fetch files from the specified branch without fetching other branches.

Let's test it:

```
git clone -b passwordless-auth -
```

This automatically configures `passwordless-auth` as the local branch and only tracks this branch.





If you run `cd pages/api` you'll find the `auth.js` file in the `passwordless-auth` branch as expected from the previous setup.

Conclusion

You might be running out of internet or storage space but you need to work on a task in a specific branch. Or you might want to clone a specific branch with limited files for various reasons. Fortunately, Git provides you the flexibility to do this. Flex your muscles and try it out, there's much more "Git" to learn.

One at a time, yeah? ✌?

Bolaji Ayodeji

Software Engineer, Content Creator, Teacher, and Developer Advocate.

If you read this far, thank the author to show them you care.

Say Thanks

Learn to code for free. freeCodeCamp's open source curriculum has helped more than 40,000 people get jobs as developers.

Get started

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) charity organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

You can make a tax-deductible donation here.

Trending Guides

Delete File in Linux	What is :: in Python?	Python PWD Equivalent
JSONObject.toString()	What is SSH in Linux?	Max int Size in Python
Python Bytes to String	Git Pull Remote Branch	Fix Git Merge Conflicts
JavaScript Refresh Page	Git Revert	JSON Comments
Java Use Cases	Python Copy File	Linux cp Command

Java Use Cases
Python list.pop()
HTTP Request Methods
Python range() Function
Check for Empty String JS
Pretty Print JSON in Python
Python Copy File
JS Sum of an Array
Compare Strings in C
Pandas Iterate Over Rows
Initialize ArrayList Java
What Does $ Mean in Python?
Linux cp Command
Python Split String
What Does $ Mean in JS?
Initialize a List in Java
Delete Environment in Conda
Check for Empty List Python

Mobile App

Our Charity

About Alumni Network Open Source Shop Support Sponsors Academic Honesty
Code of Conduct Privacy Policy Terms of Service Copyright Policy