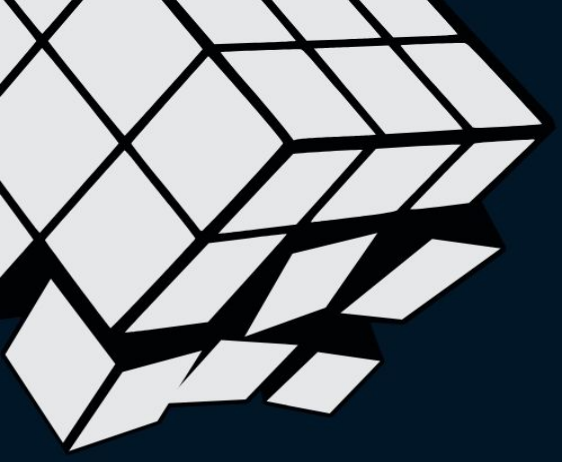


INTRODUCTION TO GIT & GITHUB

Table of Contents

```
git log
```

Here's a screenshot of the process:

```
root@do-dev:~/demo# git init .
Initialized empty Git repository in /root/demo/.git/
root@do-dev:~/demo# touch demo.txt
root@do-dev:~/demo# git add .
root@do-dev:~/demo# git commit -m "First commit"
[master (root-commit) 45f651d] First commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 demo.txt
root@do-dev:~/demo# echo "Wrong changes..." > demo.txt
root@do-dev:~/demo# git add .
root@do-dev:~/demo# git commit -m "Wrong commit..."
[master 9688e23] Wrong commit...
 1 file changed, 1 insertion(+)
root@do-dev:~/demo# git log
commit 9688e23761a6ccbbfaa4362a391b50a63d4ba39a (HEAD -> master)
Author: Bobby Iliev <bobby@bobbyiliev.com>
Date:   Wed Jan 8 08:49:49 2020 +0000

    Wrong commit...

commit 45f651dd4e83205fe1a72ac16bf0d7a3ecfba904
Author: Bobby Iliev <bobby@bobbyiliev.com>
Date:   Wed Jan 8 08:49:27 2020 +0000

    First commit
root@do-dev:~/demo# git reset --soft HEAD~1
root@do-dev:~/demo# git log
commit 45f651dd4e83205fe1a72ac16bf0d7a3ecfba904 (HEAD -> master)
Author: Bobby Iliev <bobby@bobbyiliev.com>
Date:   Wed Jan 8 08:49:27 2020 +0000

    First commit
root@do-dev:~/demo# echo "Fixed changes..." > demo.txt
root@do-dev:~/demo# git add .
root@do-dev:~/demo# git commit -m "Fixed commit..."
[master 081f0fb] Fixed commit...
 1 file changed, 1 insertion(+)
root@do-dev:~/demo# git log
commit 081f0fbf3d32ad7e7946e1ec4cc5b432fc699fef (HEAD -> master)
Author: Bobby Iliev <bobby@bobbyiliev.com>
Date:   Wed Jan 8 08:50:28 2020 +0000

    Fixed commit...

commit 45f651dd4e83205fe1a72ac16bf0d7a3ecfba904
Author: Bobby Iliev <bobby@bobbyiliev.com>
Date:   Wed Jan 8 08:49:27 2020 +0000

    First commit
root@do-dev:~/demo#
```

Note: You can reset your changes by more than one commit by using the following syntax:

```
git reset --soft HEAD~n
```

where `n` is the number of commits you want to reset back.

Another approach would be to use `git revert COMMIT_ID` instead.

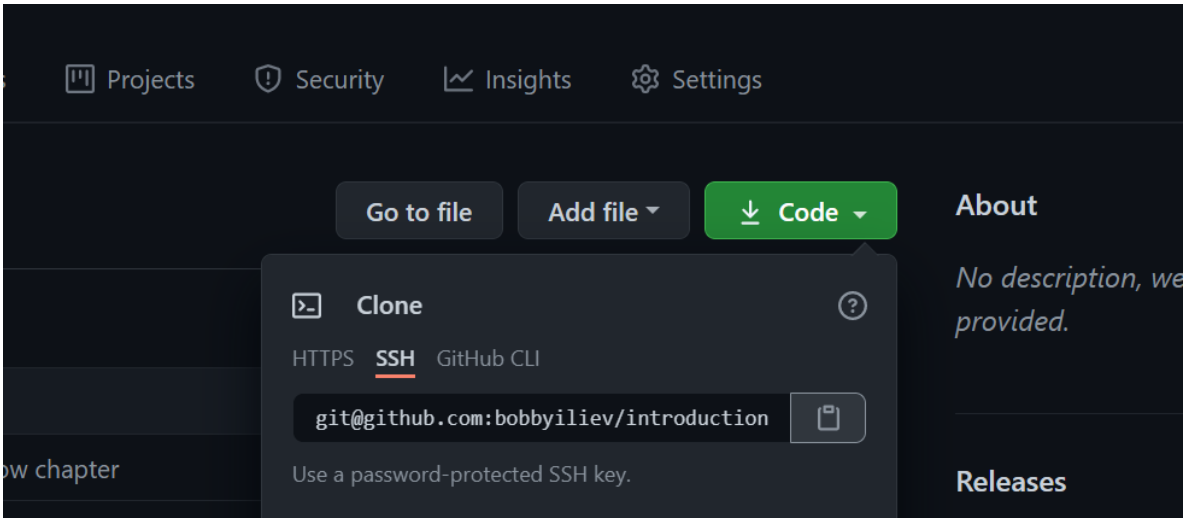
Here is a quick video demo on how to do the above:

Reverting changes

Git Clone

More often than not, rather than starting a new project from scratch, you would either join a company and start working on an existing project, or you would contribute to an already established open source project. So in this case, in order to get the repository from GitHub to your local machine, you would need to use the `git clone` command.

The most straightforward way to clone your GitHub repository is to first visit the repository in your browser, and then click on the green `Code` button and choose the method that you want to use to clone the repository:



In my case, I would go for the SSH method as I already have my SSH keys configured as per chapter 14.

As I am cloning this repository here, the URL would look like this:

```
git@github.com:bobbyiliev/introduction-to-bash-scripting.git
```

Once you have this in my clipboard, head back to your terminal, go to a directory where you would like to clone the repository to and then run the following command:

```
git clone git@github.com:bobbyiliev/introduction-to-bash-
scripting.git
```

The output that you would get will look like this:

```
Cloning into 'introduction-to-bash-scripting'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 215 (delta 7), reused 14 (delta 4), pack-reused
194
Receiving objects: 100% (215/215), 3.08 MiB | 5.38 MiB/s,
done.
Resolving deltas: 100% (114/114), done.
```

Essentially what the `git clone` command does is to more or less download the repository from GitHub to your local folder.

Now you can start making the changes to the project by creating a new branch, writing some code, and finally committing and pushing your changes!

One important thing to keep in mind is that in case that you are not the maintainer of the repository and do not have the right to push to the repository, you would need to first fork the original repository and then clone the forked repository from your account. In the next chapter, we will go through the full process of forking a repository!

Forking in Git

When contributing to an open-source project, you will not be able to make the changes directly to the project. Only the repository maintainers have that privilege.

What you need to do instead is to fork the specific repository, make the changes to the forked project and then submit a pull request to the original project. You will learn more about pull requests in the next chapters.

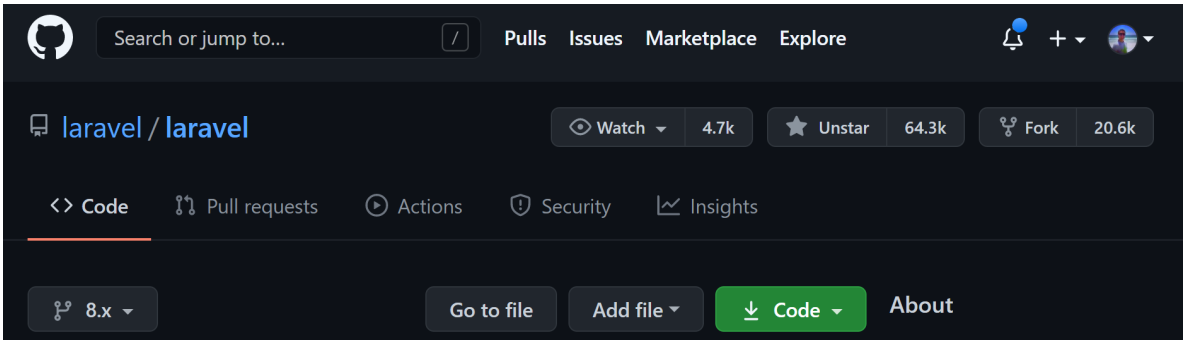
If you clone a repository that you don't have the access to and then try to push the changes directly to that repository, you would get the following error:

```
ERROR: Permission to laravel/laravel.git denied to bobbyiliev.
Fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

This is where Forks come into play!

In order to fork a repository, you need to visit the repository via your browser and click on the Fork button on the top right:



Then choose the account that you want to fork the repository to:

GitHub CLI Installation

As I will be using Ubuntu, to install `gh` you need to run the following commands:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-key
C99B11DEB97541F0
sudo apt-add-repository https://cli.github.com/packages
sudo apt update
sudo apt install gh
```

If you are on a Mac, you can install `gh` using Homebrew:

```
brew install gh
```

For any other operating systems, I recommend following the steps from the official documentation here.

Once you have `gh` installed, you can verify that it works with the following command:

```
gh --version
```

This would output the `gh` version:

```
gh version 1.0.0 (2020-09-16)
https://github.com/cli/cli/releases/tag/v1.0.0
```

In my case, I'm running the latest `gh` v1.0.0, which got released just a couple of days ago.

Authentication

Once you have `gh` installed, you need to login to your GitHub account.

To do so, you need to run the following command:

```
gh auth login
```

You will see the following output:

```
? What account do you want to log into?  [Use arrows to move,
type to filter]
> GitHub.com
  GitHub Enterprise Server
```

You have an option to choose between GitHub.com or GitHub Enterprise. Click enter and then follow the authentication process.

Another useful command is the `gh help` command. This will give you a list with the available `gh` commands that you could use:

```
USAGE
  gh <command> <subcommand> [flags]

CORE COMMANDS
  gist:       Create gists
  issue:      Manage issues
  pr:         Manage pull requests
  release:    Manage GitHub releases
  repo:       Create, clone, fork, and view repositories

ADDITIONAL COMMANDS
  alias:      Create command shortcuts
  api:        Make an authenticated GitHub API request
  auth:       Login, logout, and refresh your authentication
  completion: Generate shell completion scripts
  config:     Manage configuration for gh
  help:       Help about any command

FLAGS
  --help      Show help for command
  --version   Show gh version

EXAMPLES
  $ gh issue create
  $ gh repo clone cli/cli
  $ gh pr checkout 321

ENVIRONMENT VARIABLES
  See 'gh help environment' for the list of supported
environment variables.

LEARN MORE
  Use 'gh <command> <subcommand> --help' for more information
about a command.
  Read the manual at https://cli.github.com/manual

FEEDBACK
  Open an issue using 'gh issue create -R cli/cli'
```

Then let's clone an existing project which we will use to play with. As an example, we can use the LaraSail repository. Rather than cloning the repository using the standard `git clone` command, we will use `gh` to do so:

```
gh repo clone thedevdojo/larasail
```

You will see the following output:

```
Cloning into 'larasail'...
```

After that `cd` into that folder:

```
cd larasail
```

We are now ready to move to some of the more useful `gh` commands!