Top 20 Git Commands with Example

[Go to the profile of Saurabh Kulshrestha](https://medium.com/@saurabh_86037?source=post_header_lockup)

[Saurabh Kulshrestha](https://medium.com/@saurabh_86037)Follow

Jul 23, 2018



Git Commands — Edureka

Git & GitHub has steadily risen from being just a preferred skill to a must-have skill for multiple job roles today. In this article, I will talk about the Top 20 Git Commands that you will be using frequently while you are working with Git.

Following are the Git commands which are being covered:

* git config
* git init
* git clone
* git add
* git commit
* git diff
* git reset
* git status
* git rm
* git log
* git show
* git tag
* git branch
* git checkout
* git merge
* git remote
* git push
* git pull
* git stash

So, let’s get started now!!

**git config**

**Usage: git config -global user.name “[name]”**

**Usage: git config -global user.email “[email address]”**

This command sets the author name and email address respectively to be used with your commits.

https://cdn-images-1.medium.com/max/1000/1*8HUmCSPeXDzPg7SXt07QQA.png

**git init**

**Usage: git init [repository name]**

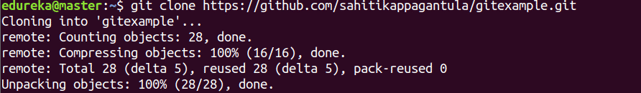
This command is used to start a new repository.

https://cdn-images-1.medium.com/max/1000/1*dWnCmhEhNO9fKt1ThV0v8A.png

**git clone**

**Usage: git clone [url]**

This command is used to obtain a repository from an existing URL.



**git add**

**Usage: git add [file]**

This command adds a file to the staging area.

https://cdn-images-1.medium.com/max/1000/1*P2An1Y9NCCjFKYqOotfKxQ.png

**Usage: git add \***

This command adds one or more to the staging area.

https://cdn-images-1.medium.com/max/1000/1*j4vMw9vJlJmk_pzK-BgRAQ.png

**git commit**

**Usage: git commit -m “[ Type in the commit message]”**

This command records or snapshots the file permanently in the version history.



**Usage: git commit -a**

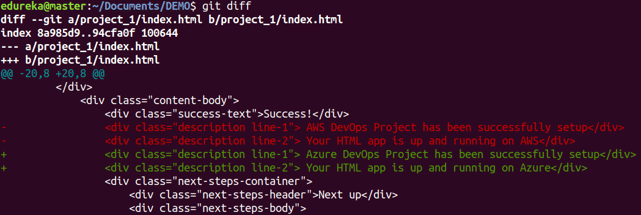
This command commits any files you’ve added with the git add command and also commits any files you’ve changed since then.

https://cdn-images-1.medium.com/max/1000/1*icxzIA5Ues4vDIB5GxsxiA.png

**git diff**

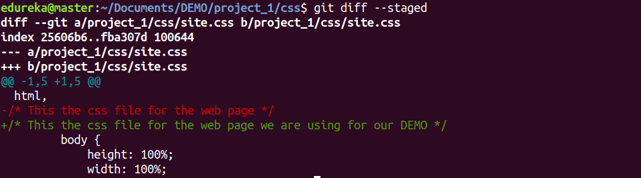
**Usage: git diff**

This command shows the file differences which are not yet staged.



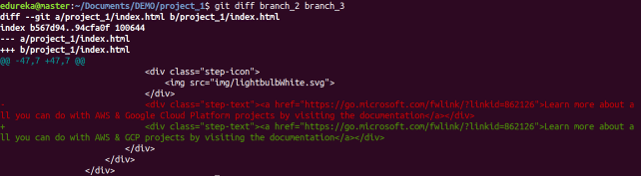
**Usage: git diff –staged**

This command shows the differences between the files in the staging area and the latest version present.



**Usage: git diff [first branch] [second branch]**

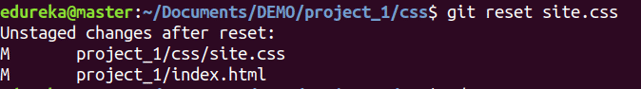
This command shows the differences between the two branches mentioned.



**git reset**

**Usage: git reset [file]**

This command unstages the file, but it preserves the file contents.



**Usage: git reset [commit]**

This command undoes all the commits after the specified commit and preserves the changes locally.

https://cdn-images-1.medium.com/max/1000/1*UxIbcPVz39tcze8g5zo74Q.png

**Usage: git reset -hard [commit]**

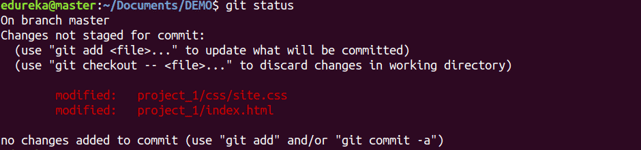
This command discards all history and goes back to the specified commit.

https://cdn-images-1.medium.com/max/1000/1*gTFoK5XixfM0DwHq4MxK6g.png

**git status**

**Usage: git status**

This command lists all the files that have to be committed.



**git rm**

**Usage: git rm [file]**

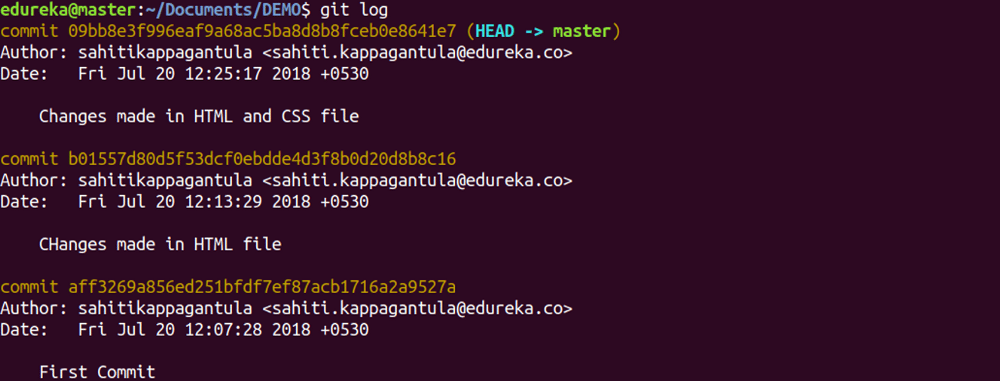
This command deletes the file from your working directory and stages the deletion.

https://cdn-images-1.medium.com/max/1000/1*fYV7NUwxT0FH4_-2ES7Rdw.png

**git log**

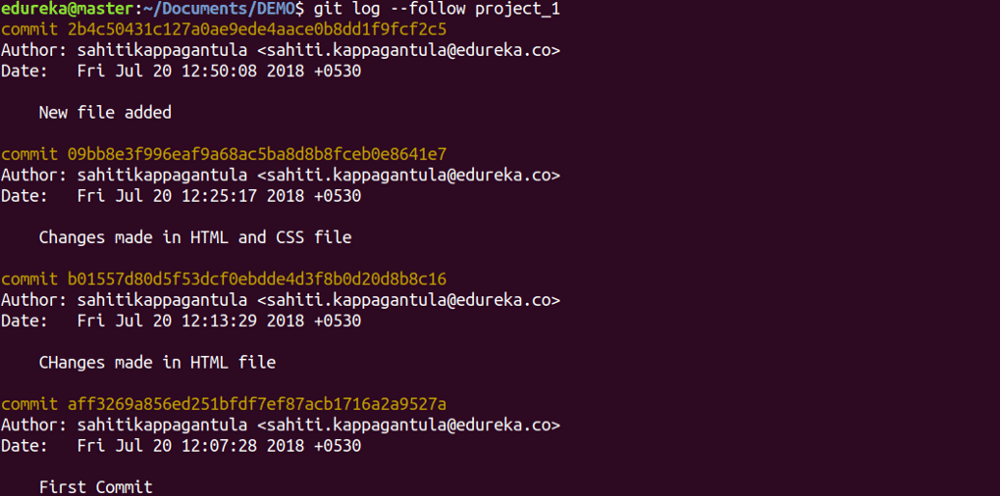
**Usage: git log**

This command is used to list the version history for the current branch.



**Usage: git log -follow[file]**

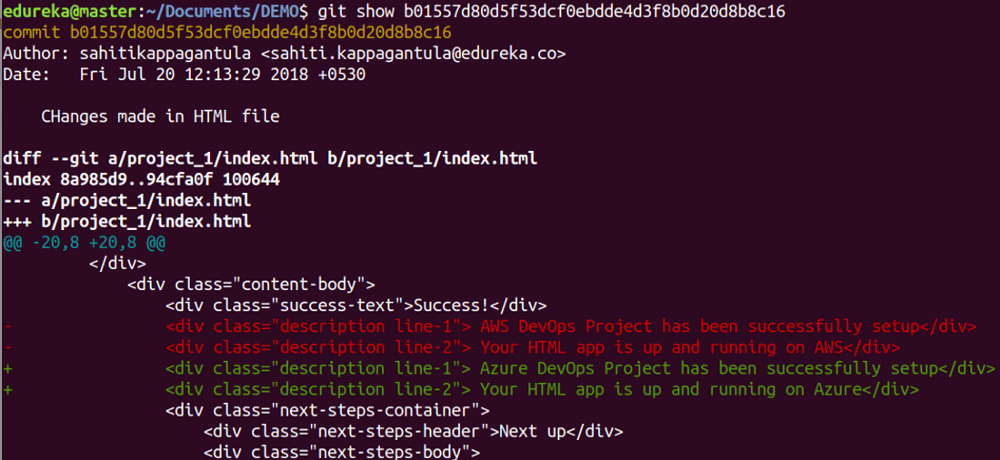
This command lists version history for a file, including the renaming of files also.



**git show**

**Usage: git show [commit]**

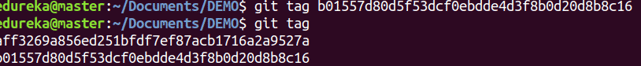
This command shows the metadata and content changes of the specified commit.



**git tag**

**Usage: git tag [commitID]**

This command is used to give tags to the specified commit.



**git branch**

**Usage: git branch**

This command lists all the local branches in the current repository.

https://cdn-images-1.medium.com/max/1000/1*4xRndU8pT95BzNr4qBizYA.png

**Usage: git branch [branch name]**

This command creates a new branch.

https://cdn-images-1.medium.com/max/1000/1*9m0DaShOc5RcvPmwPg8h8g.png

**Usage: git branch -d [branch name]**

This command deletes the feature branch.

https://cdn-images-1.medium.com/max/1000/1*lIWOaErZzOk3uvT4aak9_A.png

**git checkout**

**Usage: git checkout [branch name]**

This command is used to switch from one branch to another.

https://cdn-images-1.medium.com/max/1000/1*jRGfXP4IPICYHXtmmJ4RHQ.png

**Usage: git checkout -b [branch name]**

This command creates a new branch and also switches to it.

https://cdn-images-1.medium.com/max/1000/1*4nn_D7_tT1kuPvWi88UdSw.png

**git merge**

**Usage: git merge [branch name]**

This command merges the specified branch’s history into the current branch.

https://cdn-images-1.medium.com/max/1000/1*QGsQYNhDzEzvjUlXI0FuZQ.png

**git remote**

**Usage: git remote add [variable name] [Remote Server Link]**

This command is used to connect your local repository to the remote server.

https://cdn-images-1.medium.com/max/1000/1*kVQ77mdtHApIpFPk1gbkPw.png

**git push**

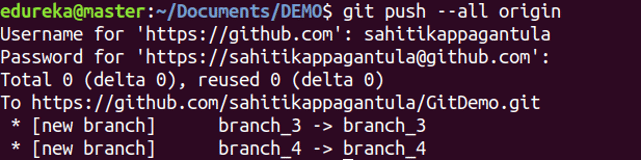
**Usage: git push [variable name] master**

This command sends the committed changes of master branch to your remote repository.



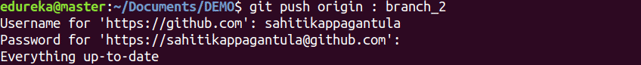
**Usage: git push -all [variable name]**

This command pushes all branches to your remote repository.



**Usage: git push [variable name] :[branch name]**

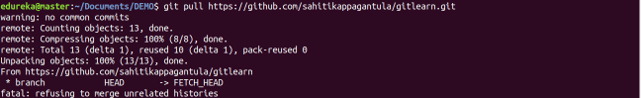
This command sends the branch commits to your remote repository.



**git pull**

**Usage: git pull [Repository Link]**

This command fetches and merges changes on the remote server to your working directory.



**git stash**

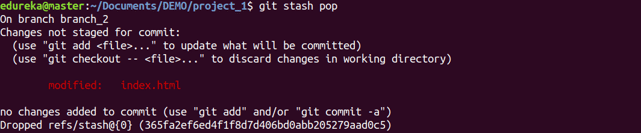
**Usage: git stash save**

This command temporarily stores all the modified tracked files.

https://cdn-images-1.medium.com/max/1000/1*5UAQr_erBnC6Fnda2rS0sw.png

**Usage: git stash pop**

This command restores the most recently stashed files.



**Usage: git stash list**

This command lists all stashed changesets.

https://cdn-images-1.medium.com/max/1000/1*lqr8ZdO78ztoeomIBDFYBA.png

**Usage: git stash drop**

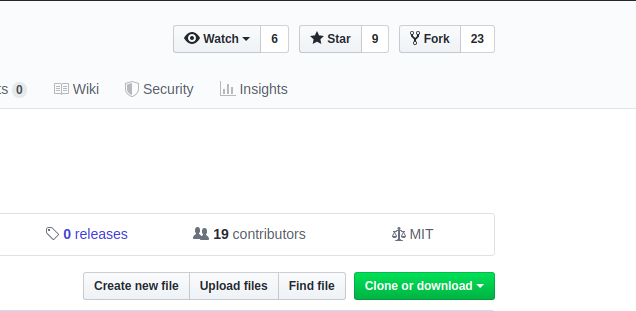
This command discards the most recently stashed changeset.

https://cdn-images-1.medium.com/max/1000/1*ZbCSi45lki09YdPCueuf-w.png

This brings us to the end o

**1. Fork the repository**

Fork the repository by clicking the fork button on the top of the page. This will create an instance of that entire repository in your account.



**2. Clone the repository**

Once the repository is in your account, clone it to your machine to work with it locally.

To clone, click on the clone button and copy the link.

$ git clone [HTTPS ADDRESS]

$ git checkout -b [Branch Name]

**3. Create a branch**

It’s good practice to create a new branch when working with repositories, whether it’s a small project or contributing to a group's work.

Branch name should be short and it should reflect the work we’re doing.

Now create a branch using the git checkout command:

$ git checkout -b [Branch Name]

**4. Make changes and commit them**

Make essential changes to the project and save it.

Then execute git status , and you’ll see the changes.

$ git add .

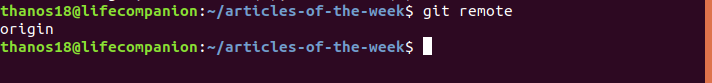
Now commit those changes using the git commit command:

$ git commit -m "Adding an article to week 02 of articles of the week"

**5. Push changes to GitHub**

In order to push the changes to GitHub, we need to identify the remote’s name.

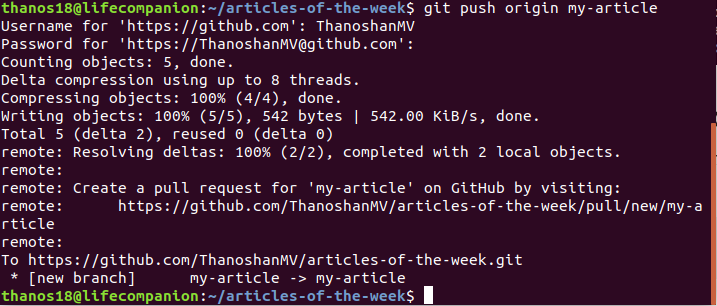
$ git remote



For this repository the remote’s name is “origin”.

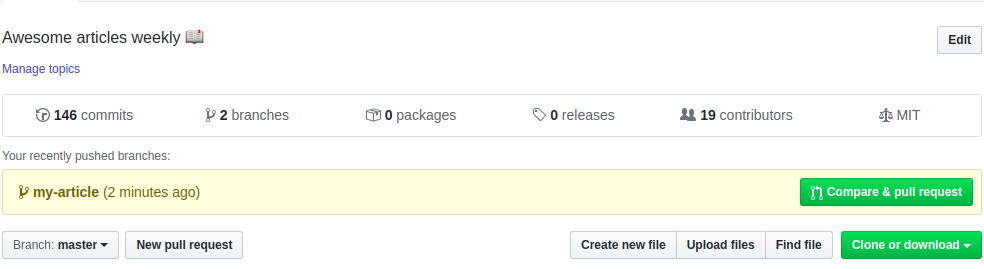
After identifying the remote’s name we can safely push those changes to GitHub.

git push origin [Branch Name]



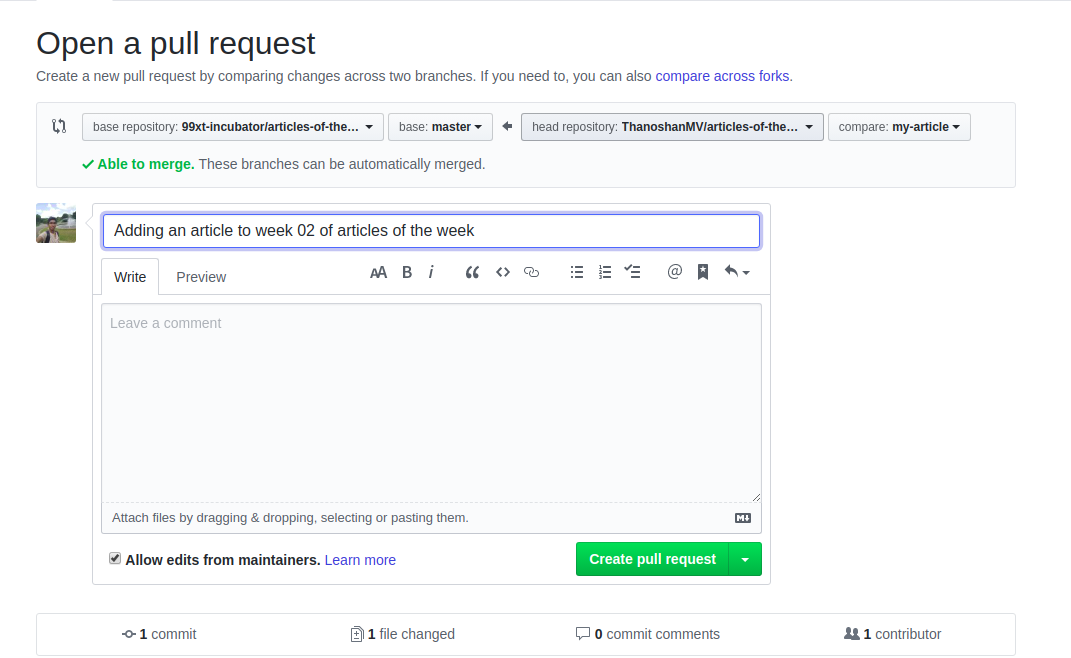
**6. Create pull request**

Go to your repository on GitHub and you’ll see a button “Compare & pull request” and click it.



Please provide necessary details on what you’ve done (You can reference issues using “#”). Now submit the pull request.

Congratulations! You've made your first pull request.



### 7. Sync your forked master branch

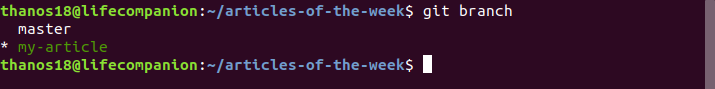
Before submitting any pull requests to the original repository you have to sync your repository to the original one.

Even if you are not going to submit a pull request to the original repository, it’s better to sync with the original repository as some additional features and bug fixes may have been done since you forked the original repository.

Follow these steps to update/sync those changes to your master branch:

1. First, check which branch you are in.

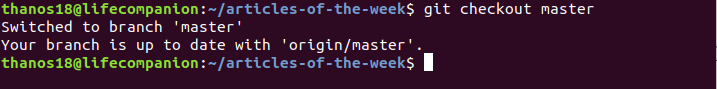
$ git branch



It’ll list all branches and indicates the current or active branch in green.

2. Switch to the master branch.

$ git checkout master

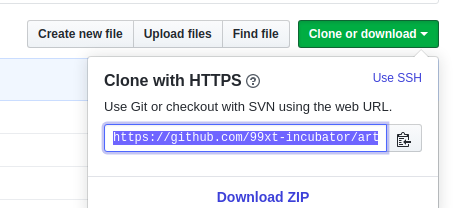


3. Add the original repository as an upstream repository.

In order to pull the changes from the original repository into your forked version, you need to add the original Git repository as an upstream repository.

$ git remote add upstream [HTTPS]

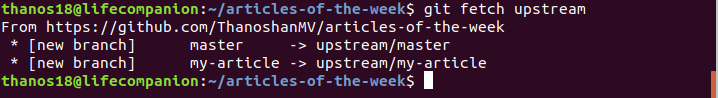
Here, [HTTPS] is the URL that you have to copy from the owner’s repository.

https://www.freecodecamp.org/news/content/images/2020/01/remote-add.png

4. Fetch the repository.

Fetch all of the changes from the original repository. Commits to the original repository will be stored in a local branch called upstream/master.

$ git fetch upstream



5. Merge it.

Merge the changes from the upstream/master into your local master branch. This will bring your fork’s master branch into sync with the upstream repository without losing your local changes.

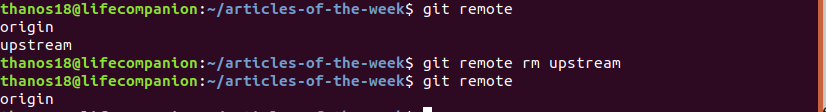
$ git merge upstream/master

6. Push changes to GitHub

At this point your local branch is synced to the original repository’s master branch. If you want to update the GitHub repository, you need to push your changes.

$ git push origin master

**NOTE:** After syncing your forked master branch you can remove that remote if you want to. But you’ll need to update/sync your repository in future too, so it's best practice to keep it.

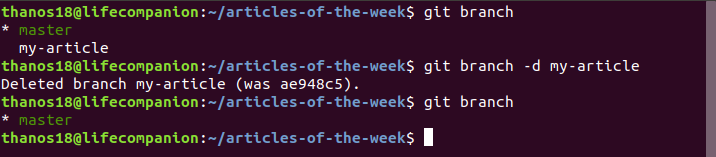


$ git remote rm [Remote Name]

### 8. Delete the unnecessary branch

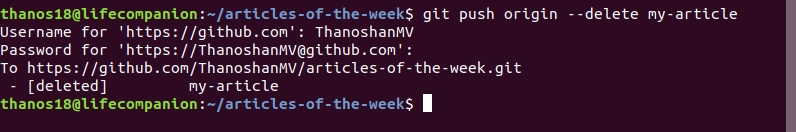
Branches are created for a special purpose. Once that purpose is accomplished, those branches aren’t necessary, so you can delete them.

$ git branch -d [Branch Name]



You can delete the version of it on GitHub, too.

git push origin --delete [Branch Name]



## Conclusion

GitHub is a powerful tool to control version history. Everyone can contribute to open source projects by making pull requests. Contributions aren’t always code – there are other ways to contribute, too.

Finally, I have to tell you that you shouldn't worry if your pull requests are rejected. Maintainers spend a lot time improving their projects, and they know a lot more about their projects than we do. So don't worry if your request isn't merged.

Stay strong, stay positive, and never give up.  
― Roy T. Bennett, [The Light in the Heart](https://www.goodreads.com/work/quotes/49604402)