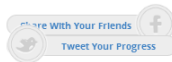Git allows groups of people to work on the same documents (often code) at the same time, and without stepping on each other's toes. It's a distributed version control system.

Our terminal prompt below is currently in directory we decided to name "octobox". To initialize a Git repository here, type the following command:

→ **git init**

```
Press enter to submit commands

> git init
```

My Octobox Repository

Advice

Directory:
A folder used for storing multiple files.

Repository:
A directory where Git has been initialized to start version controlling your files.

Clicky Click:
Click on the instructions preceded by an arrow. They will be copied into the terminal prompt.

CREATED BY
code school

Share With Your Friends
Tweet Your Progress
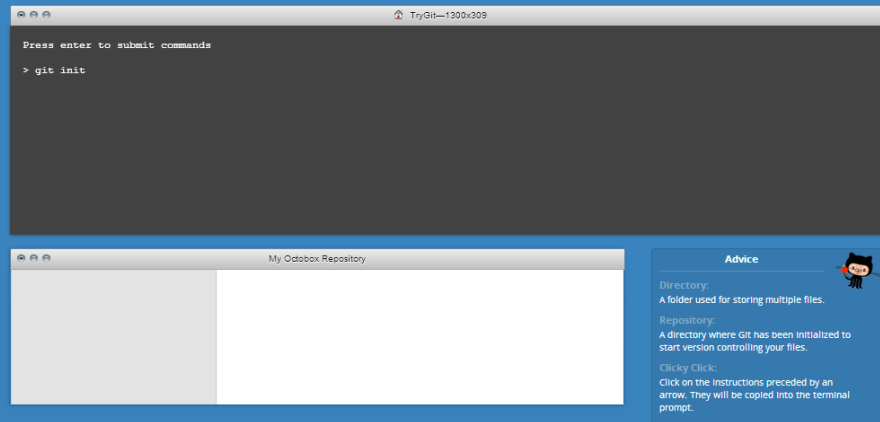
SPONSORED BY
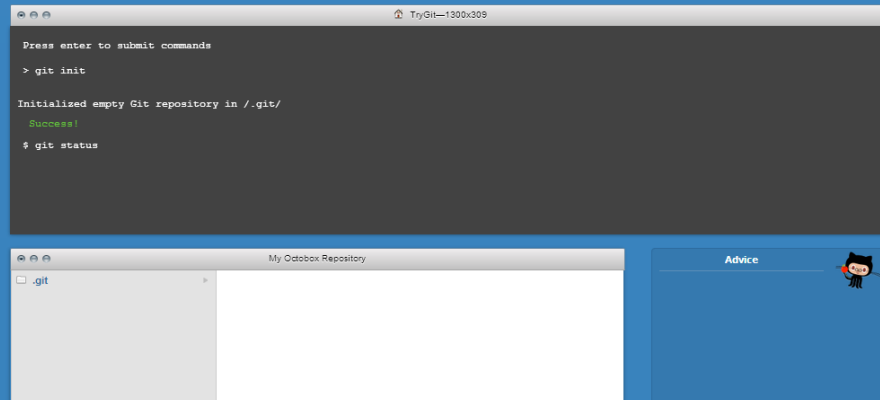GitHub

⊕ Need support?

1.2 · Checking the Status

Good job! As Git just told us, our "octobox" directory now has an empty repository in **/.git/**. The repository is a hidden directory where Git operates.
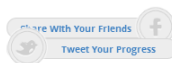
To save your progress as you go through this tutorial -- and earn a badge when you successfully complete it -- head over to **create a free Code School account**. We'll wait for you here.

Next up, let's type the **git status** command to see what the current state of our project is:

→ **git status**

```
Press enter to submit commands

> git init

Initialized empty Git repository in /.git/

   Success!

$ git status
```

My Octobox Repository

☐ .git

Advice

CREATED BY
code school

Share With Your Friends
Tweet Your Progress

SPONSORED BY
GitHub

⊕ Need support?

1.3 · Adding & Committing

I created a file called **octocat.txt** in the octobox repository for you (as you can see in the browser below).

You should run the **git status** command again to see how the repository status has changed:

git status

```
Initialized empty Git repository in /.git/
    Success!
  $ git status

# On branch master
#
# Initial commit
#
nothing to commit (create/copy files and use "git add" to track)
    Success!
  $ git status
```

My Octobox Repository

☐ .git
octocat.txt

Advice

Tip:
It's healthy to run `git status` often. Sometimes things change and you don't notice it.

1.4 · Adding Changes

Good, it looks like our Git repository is working properly. Notice how Git says **octocat.txt** is "untracked"? That means Git sees that **octocat.txt** is a new file.

To tell Git to start tracking changes made to **octocat.txt**, we first need to add it to the staging area by using **git add**.

git add octocat.txt

```
    Success!
  $ git status

# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       octocat.txt
nothing added to commit but untracked files present (use "git add" to track)
    Success!
  $ git add octocat.txt
```

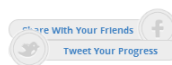### My Octobox Repository

☐ .git
octocat.txt

Advice

staged:
Files are ready to be committed.

unstaged:
Files with changes that have not been prepared to be commited.

untracked:
Files aren't tracked by Git yet. This usually indicates a newly created file.

deleted:
File has been deleted and is waiting to be removed from Git.

1.5 · Checking for Changes

Good job! Git is now tracking our **octocat.txt** file. Let's run **git status** again to see where we stand:
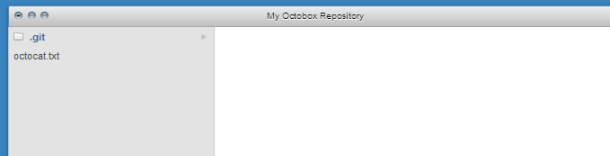
→ **git status**

```
$ Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       octocat.txt
nothing added to commit but untracked files present (use "git add" to track)
   Success!

$ git add octocat.txt



   Nice job, you've added octocat.txt to the Staging Area

$ git status
```

My Octobox Repository

.git
octocat.txt

Advice

add all:
You can also type `git add -A .` where the dot stands for the current directory, so everything in and beneath it is added. The `-A` ensures even file deletions are included.

git reset:
You can use `git reset <filename>` to remove a file or files from the staging area.

1.6 · Committing

Notice how Git says **changes to be committed**? The files listed here are in the **Staging Area**, and they are not in our repository yet. We could add or remove files from the stage before we store them in the repository.

To store our staged changes we run the **commit** command with a message describing what we've changed. Let's do that now by typing:
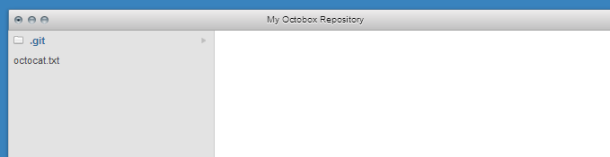
→ **git commit -m "Add cute octocat story"**

```
   Nice job, you've added octocat.txt to the Staging Area

$ git status

# On branch master
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   octocat.txt
#
   Success!

$ git commit -m "Add cute octocat story"
```
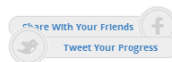
My Octobox Repository

.git
octocat.txt

Advice

Staging Area:
A place where we can group files together before we "commit" them to Git.

Commit
A "commit" is a snapshot of our repository. This way if we ever need to look back at the changes we've made (or if someone else does), we will see a nice timeline of all changes.

1.7 · Adding All Changes

Great! You also can use wildcards if you want to add many files of the same type. Notice that I've added a bunch of .txt files into your directory below.

I put some in a directory named "octofamily" and some others ended up in the root of our "octobox" directory. Luckily, we can add all the new files using a wildcard with **git add**. Don't forget the quotes!

→ **git add '*.txt'**

```
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   octocat.txt
#

  Success!

$ git commit -m "Add cute octocat story"


[master (root-commit) 20b5ccd] Add cute octocat story
 1 file changed, 1 insertion(+)
 create mode 100644 octocat.txt

  Success!

$ git add '*.txt'
```

TryGit—1300x309

My Octobox Repository

☐ .git
☐ octofamily
blue_octocat.txt
octocat.txt
red_octocat.txt

Advice

Wildcards:
We need quotes so that Git will receive the wildcard before our shell can interfere with it. Without quotes our shell will only execute the wildcard search within the current directory. Git will receive the list of files the shell found instead of the wildcard and it will not be able to add the files inside of the octofamily directory.

CREATED BY
c<>de school

Share With Your Friends
Tweet Your Progress

SPONSORED BY
GitHub

⊕ Need support?

1.8 · Committing All Changes

Okay, you've added all the text files to the staging area. Feel free to run git status to see what you're about to commit.

If it looks good, go ahead and run:

→ **git commit -m 'Add all the octocat txt files'**

```
$ git commit -m "Add cute octocat story"


[master (root-commit) 20b5ccd] Add cute octocat story
 1 file changed, 1 insertion(+)
 create mode 100644 octocat.txt

  Success!

$ git add '*.txt'



  Success!

$ git commit -m 'Add all the octocat txt files'
```

TryGit—1300x309

My Octobox Repository

☐ .git
☐ octofamily
blue_octocat.txt
octocat.txt
red_octocat.txt

Advice

Check all the things!

When using wildcards you want to be extra careful when doing commits. Make sure to check what files and folders are staged by using `git status` before you do the actual commit. This way you can be sure you're committing only the things you want.

CREATED BY
c<>de school

Share With Your Friends
Tweet Your Progress
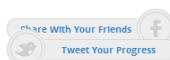
SPONSORED BY
GitHub

⊕ Need support?

1.9 · History

So we've made a few commits. Now let's browse them to see what we changed.

Fortunately for us, there's **git log**. Think of Git's log as a journal that remembers all the changes we've committed so far, in the order we committed them. Try running it now:

→ **git log**

```
TryGit—1300x309

  Success!

$ git commit -m 'Add all the octocat txt files'

[master 3852b4d] Add all the octocat txt files
 4 files changed, 4 insertions(+)
 create mode 100644 blue_octocat.txt
 create mode 100644 octofamily/baby_octocat.txt
 create mode 100644 octofamily/momma_octocat.txt
 create mode 100644 red_octocat.txt

  Success!

$ git log
```

My Octobox Repository

☐ .git
☐ octofamily
blue_octocat.txt
octocat.txt
red_octocat.txt

Advice

More useful logs:

Use `git log --summary` to see more information for each commit. You can see where new files were added for the first time or where files were deleted. It's a good overview of what's going on in the project.

CREATED BY
c<>de school

Share With Your Friends
Tweet Your Progress
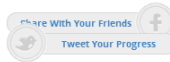
SPONSORED BY
GitHub

⊕ Need support?

1.10 · Remote Repositories

Great job! We've gone ahead and created a new empty GitHub repository for you to use with Try Git at **https://github.com/try-git/try_git.git**. To push our local *repo* to the GitHub server we'll need to add a remote repository.

This command takes a *remote name* and a *repository URL*, which in your case is **https://github.com/try-git/try_git.git**.

Go ahead and run **git remote add** with the options below:

→ **git remote add origin https://github.com/try-git/try_git.git**

```
TryGit—1300x309

$ git log

commit 3852b4db1634463d0bb4d267edb7b3f9cd02ace1
Author: Try Git <try_git@github.com>
Date:   Sat Oct 10 08:30:00 2020 -0500

    Add all the octocat txt files

commit b652edfd888cd3d5e7fcb857d0dabc5a0fcb5e28
Author: Try Git <try_git@github.com>
Date:   Sat Oct 10 08:30:00 2020 -0500

    Added cute octocat story

  Success!

$ git remote add origin https://github.com/try-git/try_git.git
```

My Octobox Repository

☐ .git
☐ octobox

Advice

git remote:

Git doesn't care what you name your remotes, but it's typical to name your main one `origin`.

It's also a good idea for your main repository to be on a remote server like GitHub in case your machine is lost at sea during a transatlantic boat cruise or crushed by three monkey statues during an earthquake.

CREATED BY
c<>de school

Share With Your Friends
Tweet Your Progress
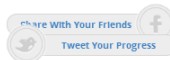
SPONSORED BY
GitHub

⊕ Need support?

The push command tells Git where to put our commits when we're ready, and boy we're ready. So let's push our local changes to our **origin** repo (on GitHub).

The name of our remote is **origin** and the default local branch name is **master**. The **-u** tells Git to remember the parameters, so that next time we can simply run **git push** and Git will know what to do. Go ahead and push it!

→ **git push -u origin master**

```
    Add all the octocat txt files
commit b652edfd888cd3d5e7fcb857d0dabc5a0fcb5e28
Author: Try Git <try_git@github.com>
Date:    Sat Oct 10 08:30:00 2020 -0500

    Added cute octocat story

  Success!

$ git remote add origin https://github.com/try-git/try_git.git



  Success!

$ git push -u origin master
```
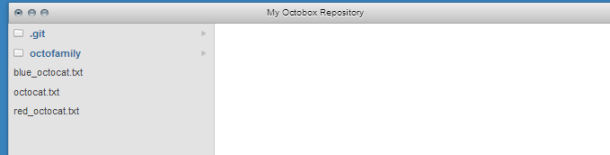
TryGit—1300x309

My Octobox Repository

☐ .git
☐ octofamily
blue_octocat.txt
octocat.txt
red_octocat.txt

Advice

Cool Stuff:
When you start to get the hang of git you can do some really cool things with **hooks** when you push.

For example, you can upload directly to a webserver whenever you push to your master remote instead of having to upload your site with an ftp client. Check out Customizing Git - Git Hooks for more information.

Share With Your Friends
Tweet Your Progress

Need support?

Let's pretend some time has passed. We've invited other people to our github project who have pulled your changes, made their own commits, and pushed them.

We can check for changes on our GitHub repository and pull down any new changes by running:

→ **git pull origin master**

```
  Success!

$ git remote add origin https://github.com/try-git/try_git.git



  Success!

$ git push -u origin master

Branch master set up to track remote branch master from origin.
  Success!

$ git pull origin master
```
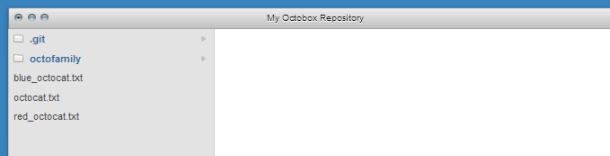
TryGit—1300x309

My Octobox Repository

☐ .git
☐ octofamily
blue_octocat.txt
octocat.txt
red_octocat.txt

Advice

git stash:
Sometimes when you go to pull you may have changes you don't want to commit just yet. One option you have, other than commiting, is to stash the changes.

Use the command ' **git stash** ' to stash your changes, and ' **git stash apply** ' to re-apply your changes after your pull.

Share With Your Friends
Tweet Your Progress

Need support?

1.13 · Differences

Uh oh, looks like there has been some additions and changes to the octocat family. Let's take a look at what is **different** from our last commit by using the **git diff** command.

In this case we want the diff of our most recent commit, which we can refer to using the **HEAD** pointer.

→ **git diff HEAD**

```
                                        TryGit—1300x309

Branch master set up to track remote branch master from origin.

  Success!

$ git pull origin master


Updating 3852b4d..3e70b0f
Fast-forward
 yellow_octocat.txt |   1
 1 file changed, 1 insertion(+)
 create mode 100644 yellow_octocat.txt

  Success!

$ git diff HEAD
```

My Octobox Repository

```
.git
octofamily
blue_octocat.txt
octocat.txt
red_octocat.txt
```

Advice

HEAD
The HEAD is a pointer that holds your position within all your different commits. By default HEAD points to your most recent commit, so it can be used as a quick way to reference that commit without having to look up the SHA.

1.14 · Staged Differences

Another great use for **diff** is looking at changes within files that have already been staged. Remember, staged files are files we have told git that are ready to be committed.

Let's use **git add** to stage **octofamily/octodog.txt**, which I just added to the family for you.

→ **git add octofamily/octodog.txt**

```
                                        TryGit—1300x309

  create mode 100644 yellow_octocat.txt

  Success!

$ git diff HEAD


diff --git a/octocat.txt b/octocat.txt
index 7d8d808..e725ef6 100644
--- a/octocat.txt
+++ b/octocat.txt
@@ -1 +1 @@
-A link to the Octocats
+Also, check out the new add to family

  Success!

$ git add octofamily/octodog.txt
```

My Octobox Repository

```
.git
octofamily
blue_octocat.txt
octocat.txt
red_octocat.txt
```

Advice

Commit Etiquette:
You want to try to keep related changes together in separate commits. Using 'git diff' gives you a good overview of changes you have made and lets you add files or directories one at a time and commit them separately.

1.15 · Staged Differences (cont'd)

Good, now go ahead and run **git diff** with the **--staged** option to see the changes you just staged. You should see that **octodog.txt** was created.

→ **git diff --staged**

```
diff --git a/octocat.txt b/octocat.txt
index 7d8d808..e725ef6 100644
--- a/octocat.txt
+++ b/octocat.txt
@@ -1 +1 @@

    Success!

$ git add octofamily/octodog.txt


    Success!

$ git diff --staged
```

My Octobox Repository

▢ .git
▢ octofamily
blue_octocat.txt
octocat.txt
red_octocat.txt

Advice

Commit Etiquette:
You want to try to keep related changes together in separate commits. Using 'git diff' gives you a good overview of changes you have made and lets you add files or directories one at a time and commit them separately.

CREATED BY
c<>de school

Share With Your Friends
Tweet Your Progress

Need support?

SPONSORED BY
GitHub

1.16 · Resetting the Stage

So now that octodog is part of the family, octocat is all depressed. Since we love octocat more than octodog, we'll turn his frown around by removing **octodog.txt**.

You can unstage files by using the **git reset** command. Go ahead and remove **octofamily/octodog.txt**.

→ **git reset octofamily/octodog.txt**

```
    Success!

$ git diff --staged

diff --git a/octofamily/octodog.txt b/octofamily/octodog.txt
new file mode 100644
index 0000000..cfbc74a
--- /dev/null
+++ b/octofamily/octodog.txt
@@ -0,0 +1 @@

    Success!

$ git reset octofamily/octodog.txt
```

My Octobox Repository

▢ .git
▢ octofamily
blue_octocat.txt
octocat.txt
red_octocat.txt

Advice

Commit Etiquette:
You want to try to keep related changes together in separate commits. Using 'git diff' gives you a good overview of changes you have made and lets you add files or directories one at a time and commit them separately.

CREATED BY
c<>de school

Share With Your Friends
Tweet Your Progress
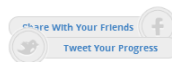
Need support?

SPONSORED BY
GitHub

1.17 · Undo

git reset did a great job of unstaging octodog.txt, but you'll notice that he's still there. He's just not staged anymore. It would be great if we could go back to how things were before octodog came around and ruined the party.

Files can be changed back to how they were at the last commit by using the command: **git checkout -- <target>**. Go ahead and get rid of all the changes since the last commit for **octocat.txt**
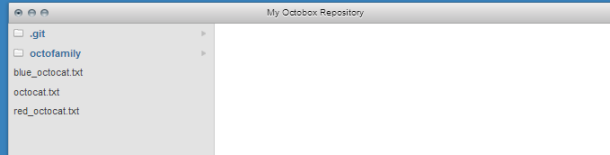
→ **git checkout -- octocat.txt**

```
diff --git a/octofamily/octodog.txt b/octofamily/octodog.txt
new file mode 100644
index 0000000..cfbc74a
--- /dev/null
+++ b/octofamily/octodog.txt
@@ -0,0 +1 @@

  Success!

  $ git reset octofamily/octodog.txt


  Success!

  $ git checkout -- octocat.txt
```

My Octobox Repository

- ☐ .git ▶
- ☐ octofamily ▶
- blue_octocat.txt
- octocat.txt
- red_octocat.txt

Advice

The '--'
So you may be wondering, why do I have to use this '--' thing? `git checkout` seems to work fine without it. It's simply promising the command line that there are no more options after the '--'. This way if you happen to have a branch named `octocat.txt`, it will still revert the file, instead of switching to the branch of the same name.

CREATED BY
c<>de school

Share With Your Friends
Tweet Your Progress

SPONSORED BY
GitHub

Need support?

1.18 · Branching Out

When developers are working on a feature or bug they'll often create a copy (aka. **branch**) of their code they can make separate commits to. Then when they're done they can merge this branch back into their main **master** branch.

We want to remove all these pesky octocats, so let's create a branch called **clean_up**, where we'll do all the work:
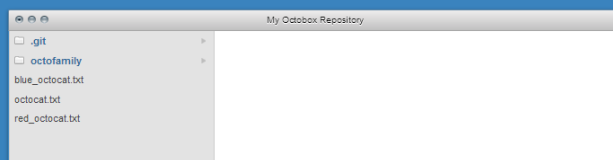
→ **git branch clean_up**

```
  Success!

  $ git reset octofamily/octodog.txt


  Success!

  $ git checkout -- octocat.txt


  Success!

  $ git branch clean_up
```

My Octobox Repository

- ☐ .git ▶
- ☐ octofamily ▶
- blue_octocat.txt
- octocat.txt
- red_octocat.txt

Advice

Branching

Branches are what naturally happens when you want to work on multiple features at the same time. You wouldn't want to end up with a master branch which has Feature A half done and Feature B half done.

Rather you'd separate the code base into two "snapshots" (branches) and work on and commit to them separately. As soon as one was ready, you might merge this branch back into the master branch and push it to the remote server.

CREATED BY
c<>de school

Share With Your Friends
Tweet Your Progress
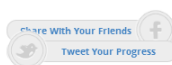
SPONSORED BY
GitHub

Need support?

1.19 · Switching Branches

Great! Now if you type **git branch** you'll see two local branches: a main branch named **master** and your new branch named **clean_up**.

You can switch branches using the **git checkout <branch>** command. Try it now to switch to the **clean_up** branch:

git checkout clean_up

```
Success!

$ git checkout -- octocat.txt


  Success!

$ git branch clean_up


  Success!

$ git checkout clean_up
```

My Octobox Repository

- .git
- octofamily
- blue_octocat.txt
- octocat.txt
- red_octocat.txt

Advice

All at Once

You can use:

```
git checkout -b new_branch
```

to checkout and create a branch at the same time. This is the same thing as doing:

```
git branch new_branch
git checkout new_branch
```

1.20 · Removing All The Things

Ok, so you're in the **clean_up** branch. You can finally remove all those pesky octocats by using the **git rm** command which will not only remove the actual files from disk, but will also stage the removal of the files for us.

You're going to want to use a wildcard again to get all the octocats in one sweep, go ahead and run:

git rm '*.txt'

```
Success!

$ git branch clean_up


  Success!

$ git checkout clean_up

Switched to branch 'clean_up'
  Success!
$ git rm '*.txt'
```

My Octobox Repository

- .git
- octofamily
- blue_octocat.txt
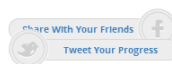- octocat.txt
- red_octocat.txt

Advice

Remove all the things!

Removing one file is great and all, but what if you want to remove an entire folder? You can use the recursive option on git rm:

```
git rm -r folder_of_cats
```

This will recursively remove all folders and files from the given directory.

1.21 · Commiting Branch Changes

Now that you've removed all the cats you'll need to commit your changes.

Feel free to run **git status** to check the changes you're about to commit.

→ **git commit -m "Remove all the cats"**

tryGit

```
                                          ⌂ TryGit—1300x309
Switched to branch 'clean_up'

   Success!

 $ git rm '*.txt'


rm 'blue_octocat.txt'
rm 'octocat.txt'
rm 'octofamily/baby_octocat.txt'
rm 'octofamily/momma_octocat.txt'
rm 'red_octocat.txt'

   Success!

 $ git commit -m "Remove all the cats"
```

My Octobox Repository

☐ .git
☐ octofamily

Advice

The '-a' option

If you happen to delete a file without using `'git rm'` you'll find that you still have to `'git rm'` the deleted files from the working tree. You can save this step by using the `'-a'` option on `'git commit'`, which auto removes deleted files with the commit.

`git commit -am "Delete stuff"`

1.22 · Switching Back to master

Great, you're almost finished with the cat... er the bug fix, you just need to switch back to the **master** branch so you can copy (or **merge**) your changes from the **clean_up** branch back into the **master** branch.

Go ahead and checkout the **master** branch:

→ **git checkout master**

tryGit

```
                                          ⌂ TryGit—1300x309
rm 'red_octocat.txt'

   Success!

 $ git commit -m "Remove all the cats"


[clean_up 63540fe] Remove all the cats
 5 files changed, 5 deletions(-)
 delete mode 100644 blue_octocat.txt
 delete mode 100644 octocat.txt
 delete mode 100644 octofamily/baby_octocat.txt
 delete mode 100644 octofamily/momma_octocat.txt
 delete mode 100644 red_octocat.txt

   Success!

 $ git checkout master
```

My Octobox Repository

☐ .git
☐ octofamily

Advice

Pull Requests

If you're hosting your repo on GitHub, you can do something called a pull request.

A pull request allows the boss of the project to look through your changes and make comments before deciding to merge in the change. It's a really great feature that is used all the time for remote workers and open-source projects.

Check out the pull request help page for more information.

1.23 · Preparing to Merge

Alrighty, the moment has come when you have to merge your changes from the **clean_up** branch into the **master** branch. Take a deep breath, it's not that scary.

We're already on the **master** branch, so we just need to tell Git to merge the **clean_up** branch into it:

→ **git merge clean_up**

```
TryGit—1300x309
[clean_up 63340ie] Remove all the cats
 5 files changed, 5 deletions(-)
 delete mode 100644 blue_octocat.txt
 delete mode 100644 octocat.txt
 delete mode 100644 octofamily/baby_octocat.txt
 delete mode 100644 octofamily/momma_octocat.txt
 delete mode 100644 red_octocat.txt

    Success!

$ git checkout master


Switched to branch 'master'

    Success!

$ git merge clean_up
```
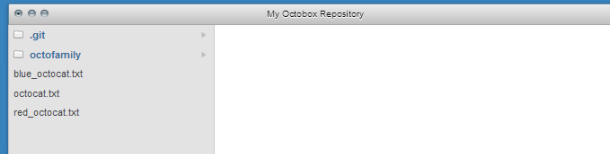
My Octobox Repository

- .git
- octofamily
- blue_octocat.txt
- octocat.txt
- red_octocat.txt

Advice

Merge Conflicts

Merge Conflicts can occur when changes are made to a file at the same time. A lot of people get really scared when a conflict happens, but fear not! They aren't that scary, you just need to decide which code to keep.

Merge conflicts are beyond the scope of this course, but if you're interested in reading more, take a look the section of the Pro Git book on how conflicts are presented.

Share With Your Friends
Tweet Your Progress
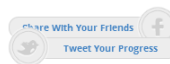
Need support?

1.24 · Keeping Things Clean

Congratulations! You just accomplished your first successful bugfix and merge. All that's left to do is clean up after yourself. Since you're done with the **clean_up** branch you don't need it anymore.

You can use **git branch -d <branch name>** to delete a branch. Go ahead and delete the **clean_up** branch now:

→ **git branch -d clean_up**

```
TryGit—1300x309
Updating 3852b4d..ec6888b
Fast-forward
 blue_octocat.txt                | 1 -
 octocat.txt                     | 1 -
 octofamily/baby_octocat.txt     | 1 -
 octofamily/momma_octocat.txt    | 1 -
 red_octocat.txt                 | 1 -
 5 files changed, 5 deletions(-)
 delete mode 100644 blue_octocat.txt
 delete mode 100644 octocat.txt
 delete mode 100644 octofamily/baby_octocat.txt
 delete mode 100644 octofamily/momma_octocat.txt
 delete mode 100644 red_octocat.txt

    Success!

$ git branch -d clean_up
```

My Octobox Repository

- .git
- octofamily
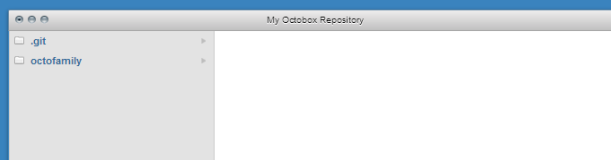
Advice

Force delete

What if you have been working on a feature branch and you decide you really don't want this feature anymore? You might decide to delete the branch since you're scrapping the idea. You'll notice that `git branch -d bad_feature` doesn't work. This is because `-d` won't let you delete something that hasn't been merged.

You can either add the `--force (-f)` option or use `-D` which combines `-d -f` together into one command.

Share With Your Friends
Tweet Your Progress
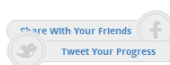
Need support?

1.25 · The Final Push

Here we are, at the last step. I'm proud that you've made it this far, and it's been great learning Git with you. All that's left for you to do now is to push everything you've been working on to your remote repository, and you're done!

→ **git push**

```
TryGit—1300x309
req_octocat.txt
5 files changed, 5 deletions(-)
 delete mode 100644 blue_octocat.txt
 delete mode 100644 octocat.txt
 delete mode 100644 octofamily/baby_octocat.txt
 delete mode 100644 octofamily/momma_octocat.txt
 delete mode 100644 red_octocat.txt

    Success!

$ git branch -d clean_up

Deleted branch clean_up (was ec6888b).

    Success!

$ git push
```

My Octobox Repository
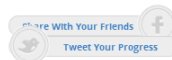
☐ .git
☐ octofamily

Advice

Learning more about Git

We only scratched the surface of Git in this course. There is so much more you can do with it. Check out the Git documentation for a full list of functions.

The Pro Git book, by Scott Chacon, is an excellent resource to teach you the inner workings of Git.

help.github and GitHub Training are also great for anything related to Git in general and using Git with GitHub.

CREATED BY
c<>de school

Share With Your Friends   f
Tweet Your Progress   🐦

SPONSORED BY
GitHub

⊕ Need support?

1.25 · The Final Push

Great! You now have a little taste of the greatness of Git. You can take a look at the wrap up page for a little more information on Git and GitHub, oh, and of course your badge!

Wrap It all Up

```
TryGit—1300x309
$ git branch -d clean_up

Deleted branch clean_up (was ec6888b).

    Success!

$ git push

To https://github.com/try-git/try_git.git
   3e70b0f..d683702  master -> master

    Success!

>
```
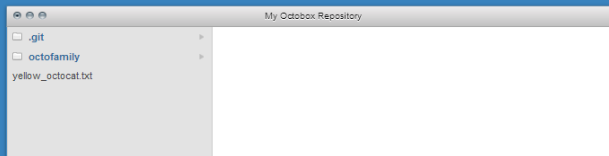
My Octobox Repository

☐ .git
☐ octofamily
yellow_octocat.txt

Advice

Learning more about Git

We only scratched the surface of Git in this course. There is so much more you can do with it. Check out the Git documentation for a full list of functions.

The Pro Git book, by Scott Chacon, is an excellent resource to teach you the inner workings of Git.

help.github and GitHub Training are also great for anything related to Git in general and using Git with GitHub.

CREATED BY
c<>de school

Share With Your Friends   f
Tweet Your Progress   🐦
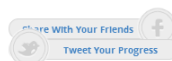
SPONSORED BY
GitHub

⊕ Need support?

Congratulations!

Nice job completing the TryGit course from Code School and GitHub. You've earned that cute little badge over there. To gaze at its perfectly symmetrical whiskers for all eternity, sign in or create a free account. And let your friends know how easy it is to try Git!

Share With Your Friends f

Tweet Your Progress t

Git Real at Code School!

If you would like to learn more, Code School has released another **Git course** for helping you work alone or with a team.

Play Level 1 for Free »

🐙 Extra Resources for Learning Git

GitHub Training »

Free online version of Pro Git »

GitHub Learning Resources »

Git Immersion »

git - the simple guide »

Go to GitHub »

◉ Related Code School Courses

Real-time Web with Node.js

Anatomy of Backbone.js

CoffeeScript

Rails for Zombies

Go to Code School »