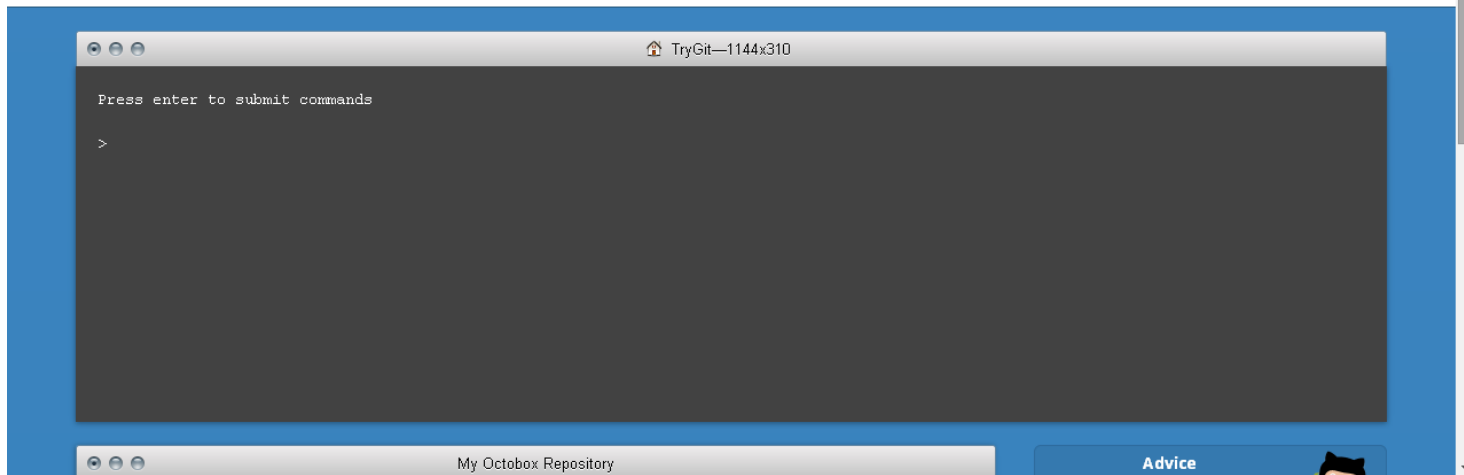


1.1 · Got 15 minutes and want to learn Git?

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**



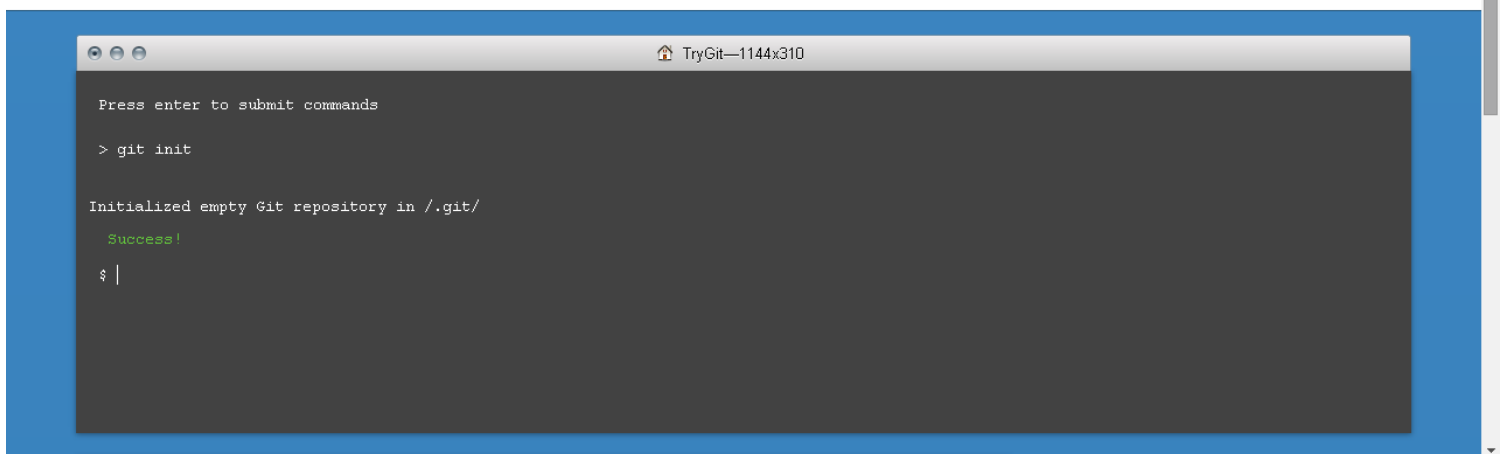
1.2 · Checking the Status

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**



rudin23 x Code School - Try Git x

https://try.github.io/levels/1/challenges/3?__utma=1.1640068749.1407449811.1407449811.1407449811.18__utmb=1.9.10.1407449811&__ut

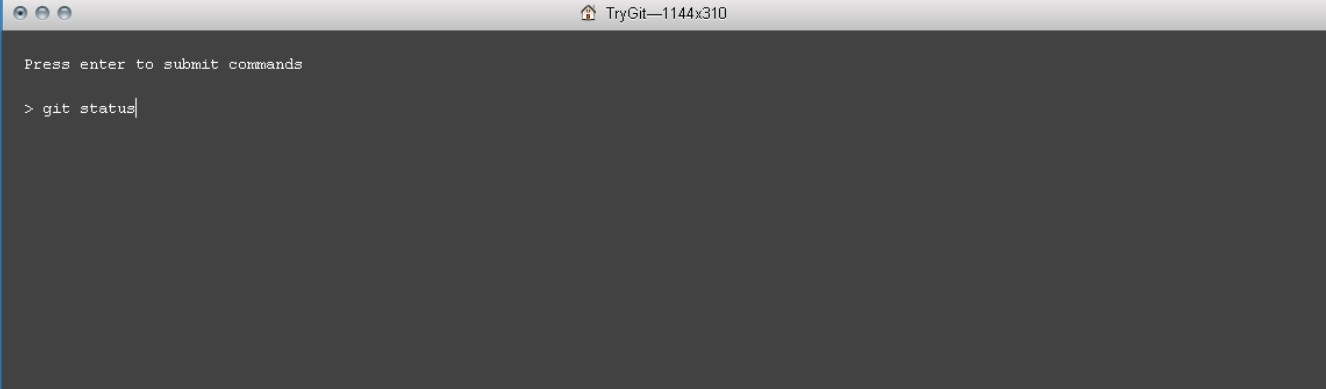

Aplicaciones Más visitado Combinación de teclas... 65 Keyboard Shortcut... Autodesk Maya ATS código-ascii AREA | Maya Learning... Maya Getting Started:... Guías GitHub

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**



```
Press enter to submit commands

> git status
```

My Octobox Repository

Advice

Tip:

rudin23 x Code School - Try Git x

https://try.github.io/levels/1/challenges/4

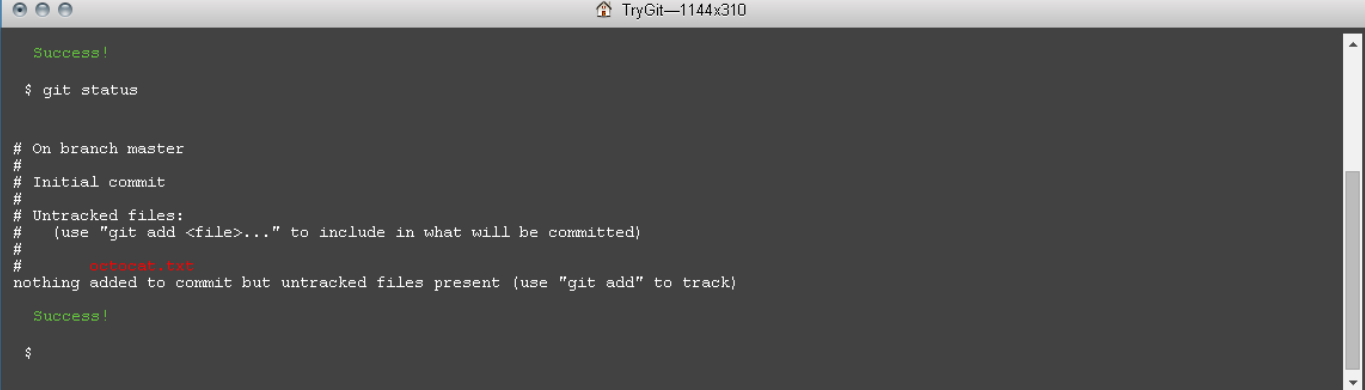

Aplicaciones Más visitado Combinación de teclas... 65 Keyboard Shortcut... Autodesk Maya ATS código-ascii AREA | Maya Learning... Maya Getting Started:... Guías GitHub

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!

$
```

My Octobox Repository

Advice

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**



```
TryGit—1144x310
# 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
$
```

My Octobox Repository

Advice

add all:

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"**



```
TryGit—1144x310
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!
$ |
```

My Octobox Repository

Advice



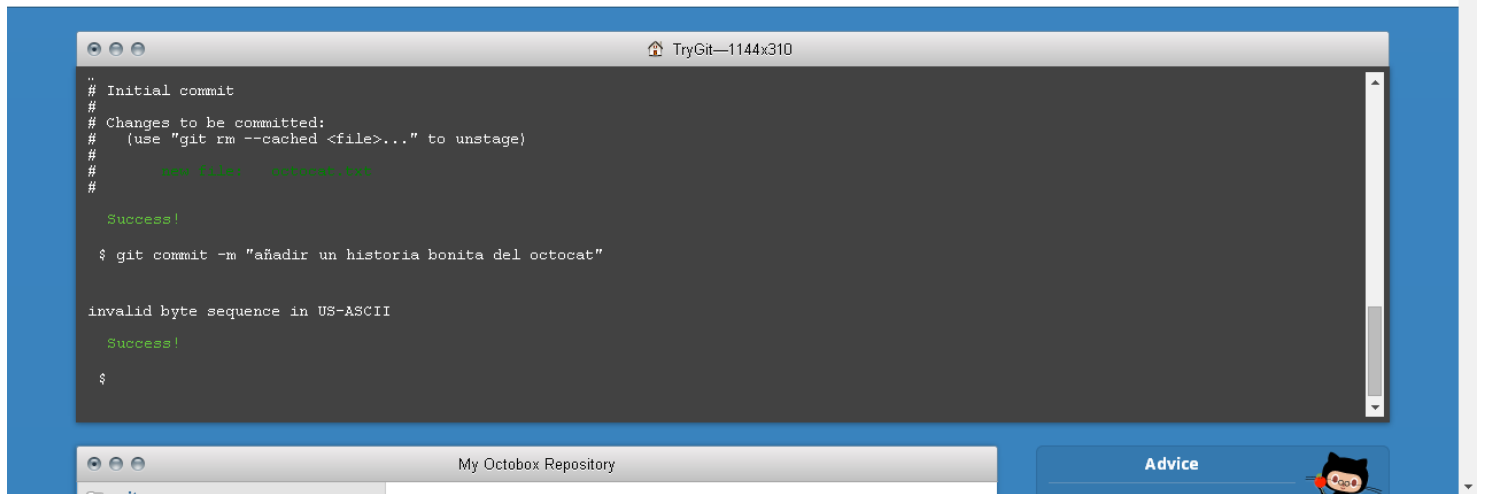
1.7 · Adding All Changes

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'**

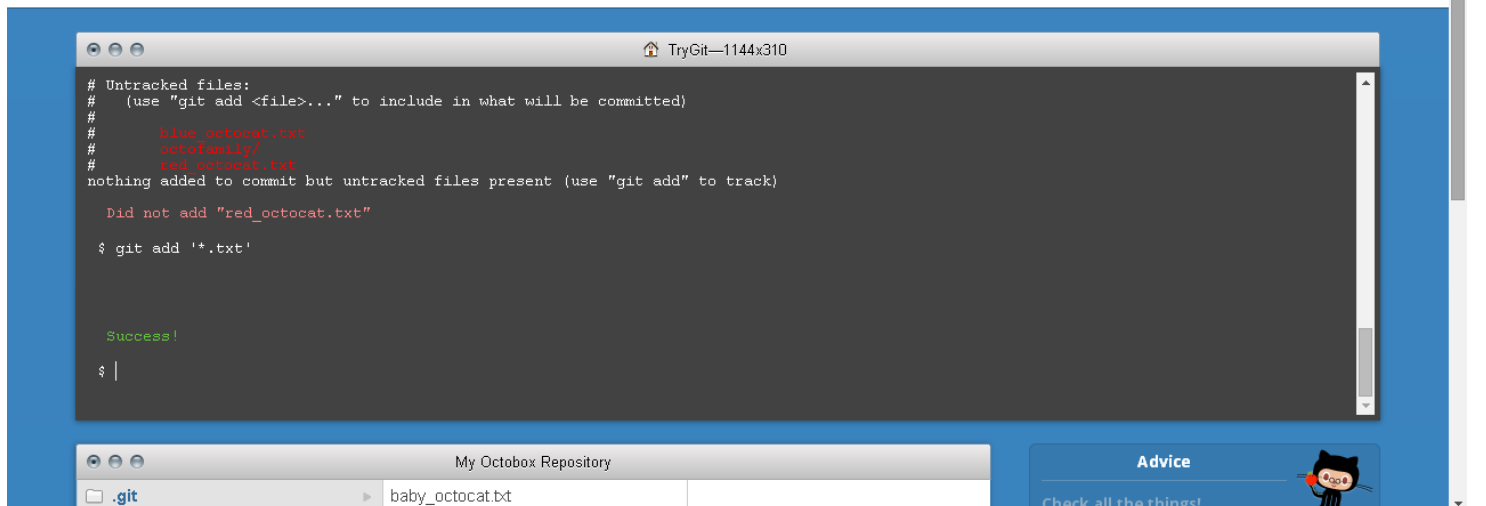


1.8 · Committing All Changes

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'**



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—1144x310

$ git log

[master 3c8baaf] añadir todos los archivos de texto Octocat
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!

$
```

My Octobox Repository

.git baby_octocat.txt

Advice

More useful logs:

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—1144x310

Add all the octocat txt files
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

commit b652edfd080cd3d5e7feb057d0dabc5a0feb5e28
Author: Try Git <try_git@github.com>
Date: Sat Oct 10 08:30:00 2020 -0500

    Added cute octocat story

create mode 100644 octocat.txt

Success!

$
```

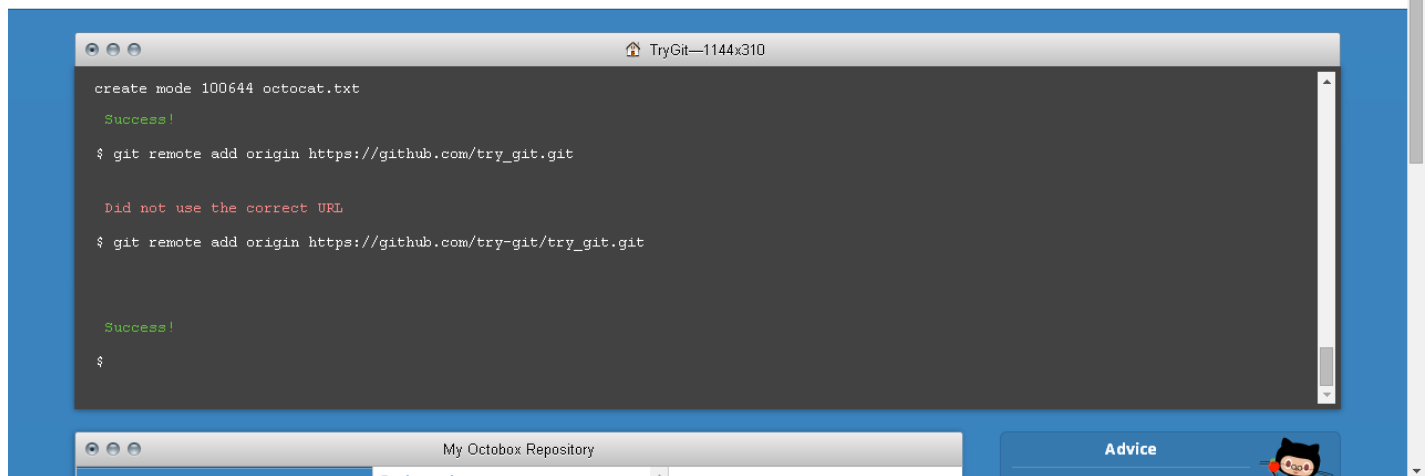


1.11 · Pushing Remotely

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**

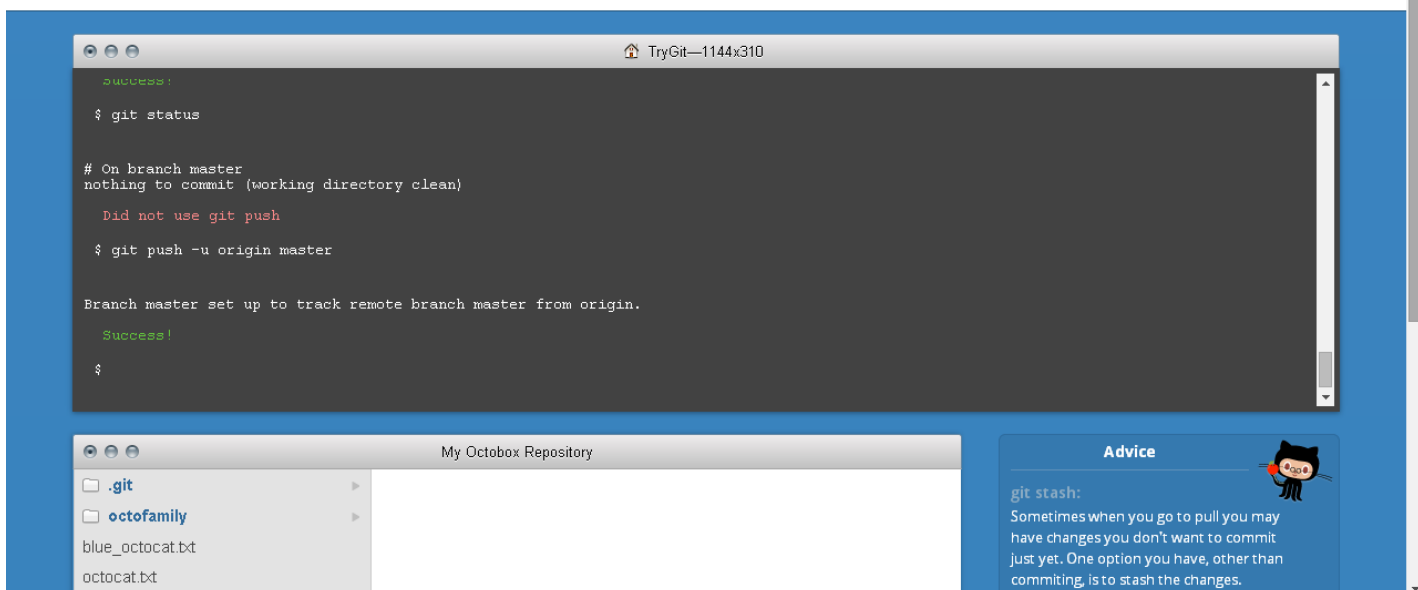


1.12 · Pulling Remotely

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**





1.13 · Differences

Uh oh, looks like there have 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—1144x310

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!

\$ |

My Octobox Repository


.git

octofamily

blue_octocat.txt

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



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—1144x310

"no changes added to commit (use "git add" and/or "git commit -a")"

Did not use git diff

\$ git diff HEAD

diff --git a/octocat.txt b/octocat.txt
index 7d8d808..e725ef6 100644
--- a/octocat.txt
+++ b/octocat.txt
@@ -1,1 @@
-A Tale of Two Octocats
+A Tale of Two Octocats and an Octodog

Success!

\$ |

My Octobox Repository


.git

octofamily

blue_octocat.txt

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

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**



This block contains two screenshots from a tutorial window. The top screenshot is a terminal window titled "TryGit—1144x310" showing the output of `git diff --staged`. It displays the diff for `octocat.txt` and `octodog.txt`, showing that `octodog.txt` is a new file. The bottom screenshot is a file explorer window titled "My Octobox Repository" showing the file structure: `.git`, `octofamily` (selected), `blue_octocat.txt`, `octocat.txt`, `baby_octocat.txt`, `momma_octocat.txt`, and `octodog.txt`. To the right of the file explorer is a blue "Advice" box with the title "Commit Etiquette:" and text explaining that `git diff` shows changes and that related changes should be committed separately.

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**



This block contains two screenshots from a tutorial window. The top screenshot is a terminal window titled "TryGit—1144x310" showing the output of `git diff --staged` after running `git reset octofamily/octodog.txt`. It shows that `octodog.txt` is no longer staged. The bottom screenshot is a file explorer window titled "My Octobox Repository" showing the file structure: `.git`, `octofamily` (selected), `blue_octocat.txt`, `octocat.txt`, `baby_octocat.txt`, `momma_octocat.txt`, and `octodog.txt`. To the right of the file explorer is a blue "Advice" box with the title "Commit Etiquette:" and text explaining that `git diff` shows changes and that related changes should be committed separately.

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**

The screenshot shows a terminal window titled "TryGit-1144x310" with the following output:

```
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 @@
+octodog

Success!

$ git reset octofamily/octodog.txt

Success!

$
```

Below the terminal is a file explorer window titled "My Octobox Repository" showing the following structure:

- .git
- octofamily
 - blue_octocat.txt

On the right, an "Advice" box contains the text: "The '--' So you may be wondering, why do I have to use this '--' thing? git checkout".

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**

The screenshot shows a terminal window titled "TryGit-1144x310" with the following output:

```
Success!

$ git reset octofamily/octodog.txt

Success!

$ git checkout -- octocat.txt

Success!

$ |
```

Below the terminal is a file explorer window titled "My Octobox Repository" showing the following structure:

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

On the right, an "Advice" box contains the text: "Branching Branches are what naturally happens when you want to work on multiple features at the same time. You wouldn't want to end".

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**

A screenshot of a terminal window titled "TryGit—1144x310". The terminal output shows the following:

```
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       octofamily/octocat.txt
nothing added to commit but untracked files present (use "git add" to track)
Use git branch to create a branch named 'clean_up'
$ git branch clean_up

Success!
$ |
```

 Below the terminal is a file explorer window titled "My Octobox Repository" showing a directory structure with ".git", "octofamily", "blue_octocat.txt", and "octocat.txt". To the right is an "Advice" panel with a small cat icon, stating "All at Once" and "You can use: git checkout -b 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'**

A screenshot of a terminal window titled "TryGit—1144x310". The terminal output shows the following:

```
Success!
$ git status

# On branch master
nothing to commit (working directory clean)
Use 'git checkout' to switch to the 'clean_up' branch
$ git checkout clean_up

Switched to branch 'clean_up'
Success!
$
```

 Below the terminal is a file explorer window titled "My Octobox Repository" showing a directory structure with ".git", "octofamily", "blue_octocat.txt", and "octocat.txt". To the right is an "Advice" panel with a small cat icon, stating "Remove all the things!" and "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>".

1.21 · Committing 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—1144x310

# On branch clean_up
nothing to commit (working directory clean)

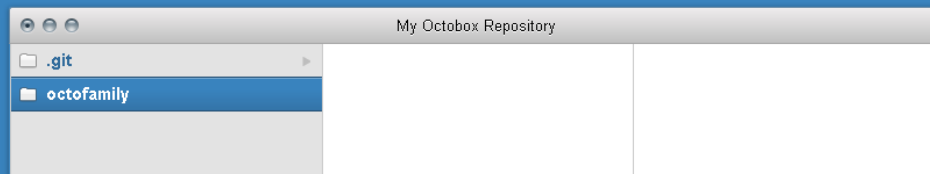
Use 'git rm' to remove and stage the octocats

$ 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!

$ |
```



Advice

The '-a' option

If you happen to delete a file without using 'git rm' you'll find that you still have to use 'git rm' the deleted files from the working tree. You can save this step by

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—1144x310

# octofamily/

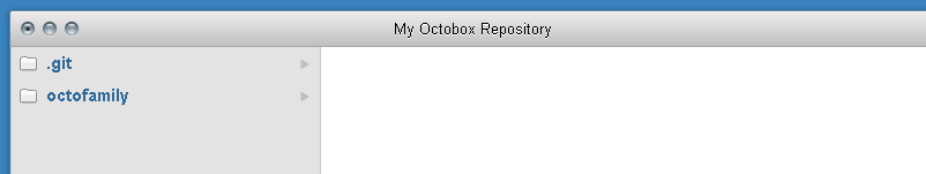
Did not use git commit

$ git commit -m "Removiendo todos los cats"

[clean up 3b4c2eb] Removiendo todos los 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!

$ |
```



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



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



TryGit—1144x310

```
Switched to branch 'master'
Success!

$ git status

# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       octofamily/.gitkeep
nothing added to commit but untracked files present (use "git add" to track)
Use 'git merge' to merge the clean_up branch with master

$ git merge clean_up
```

My Octobox Repository

.git	.gitkeep
octofamily	baby_octocat.txt
blue_octocat.txt	momma_octocat.txt
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



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



TryGit—1144x310

```
Updating 3852b4d..ec6880b
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!

$ |
```

My Octobox Repository

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



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—1144x310

```
Success!

$ git branch -d clean_up

Deleted branch clean_up (was ec6888b).

Success!

$ git status

# On branch master
nothing to commit (working directory clean)

Use 'git push' to push your repository to your remote

$ |
```



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.



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—1144x310

```
$ git status

# On branch master
nothing to commit (working directory clean)

Use 'git push' to push your repository to your remote

$ git push

To https://github.com/try-git/try_git.git
3e70b0f..cd56414 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.