Github & Linux Basic commands

Two ways

Terminal

Github

Use case #1: Maintain source code for personal use

Remote

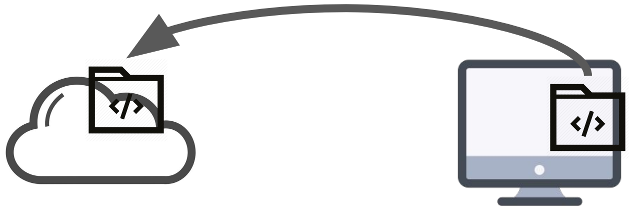
Local

Use case #1: Maintain source code for personal use

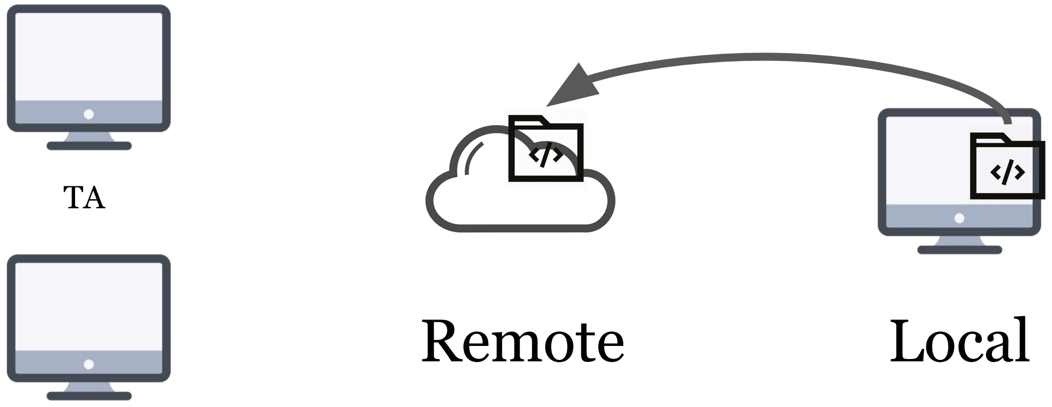
Remote

Local

Use case #1: Maintain source code for personal use

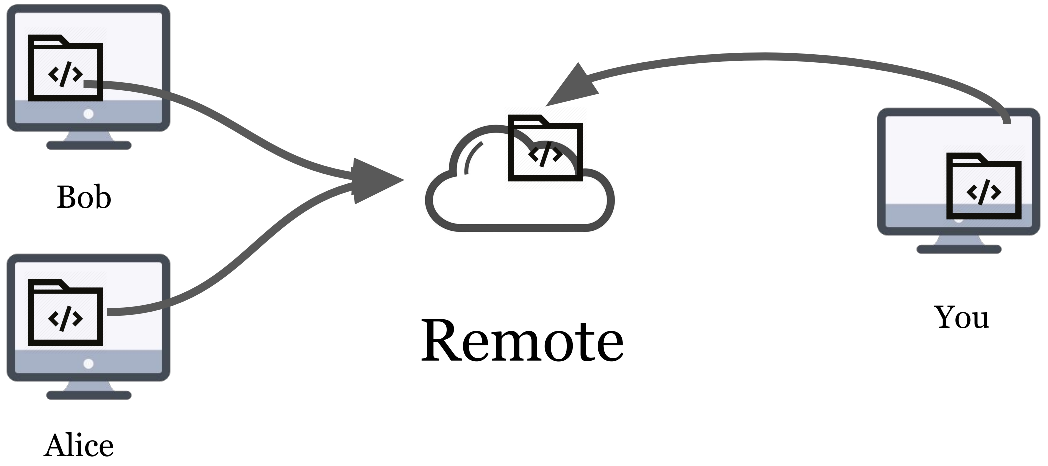


Use case #1: Maintain source code for personal use



Others can view it if they have access to the code

Use case #2: Team collaboration



Need to have version control

Version control tool

Web based hosting service for Git

git ≠ GitHub

What will be covered today

create a new repository from your computer

create a new repository from Github

checkout a repository from Github

working directory vs. HEAD vs. Index

add & commit

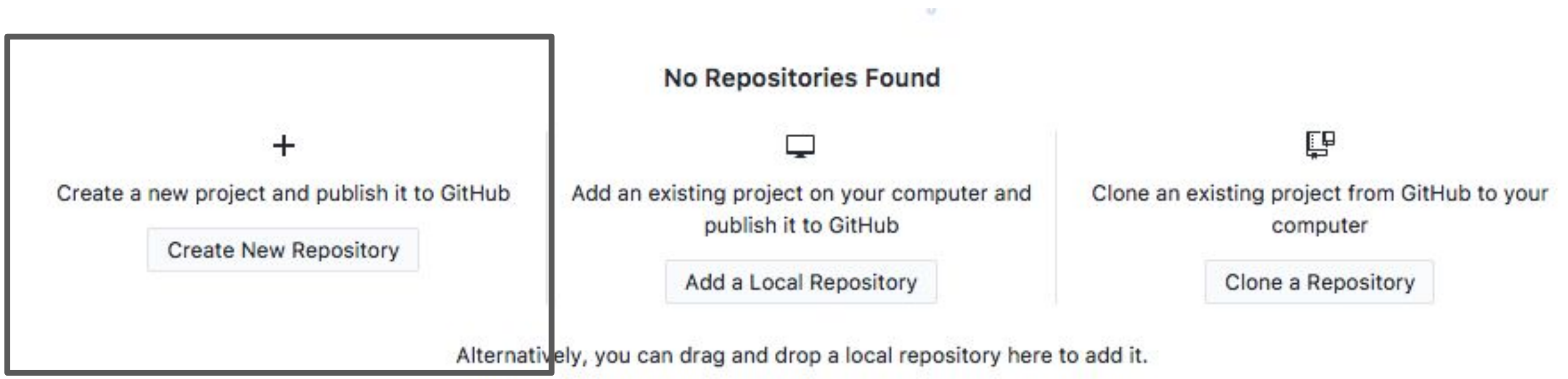
pushing changes

update and merge

create a new repository

(from your computer)

Github Desktop



No Repositories Found

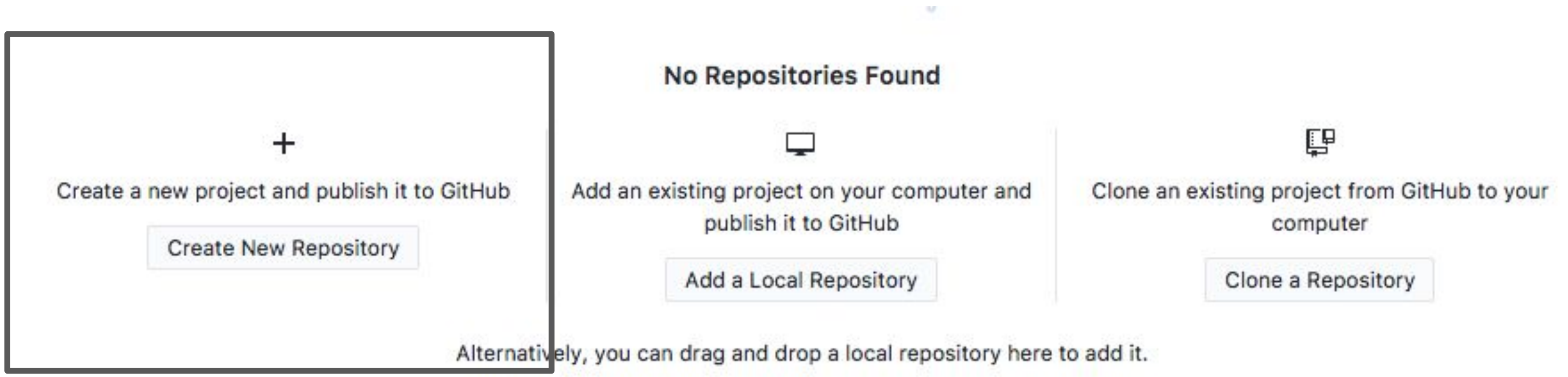
Create a new project and publish it to GitHub
Create New Repository
Add an existing project on your computer and publish it to GitHub
Add a Local Repository
Clone an existing project from GitHub to your computer
Clone a Repository

Alternatively, you can drag and drop a local repository here to add it.

create a new repository

(from your computer)

Github Desktop



No Repositories Found

Create a new project and publish it to GitHub

Create New Repository

Add an existing project on your computer and publish it to GitHub

Add a Local Repository

Clone an existing project from GitHub to your computer

Clone a Repository

Alternatively, you can drag and drop a local repository here to add it.

everything is still local

create a new repository

(from your computer)

1. create a new directory

```
mkdir folder_name
```

2. go to this directory

```
cd folder_name
```

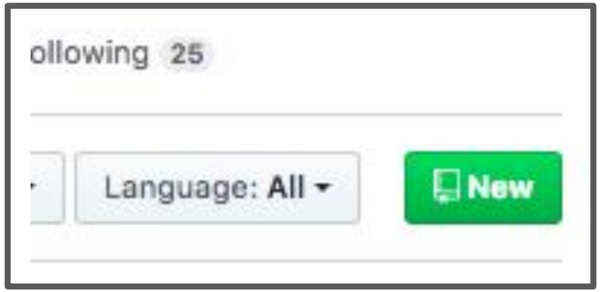
3. create a new git repository

```
git init
```

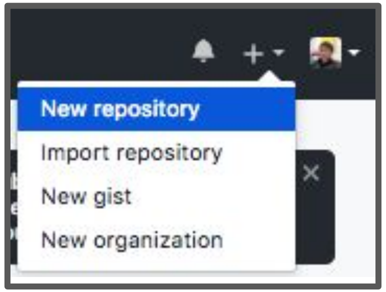
everything is still local

create a new repository

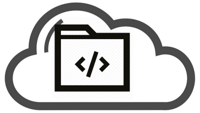
(from the github website)



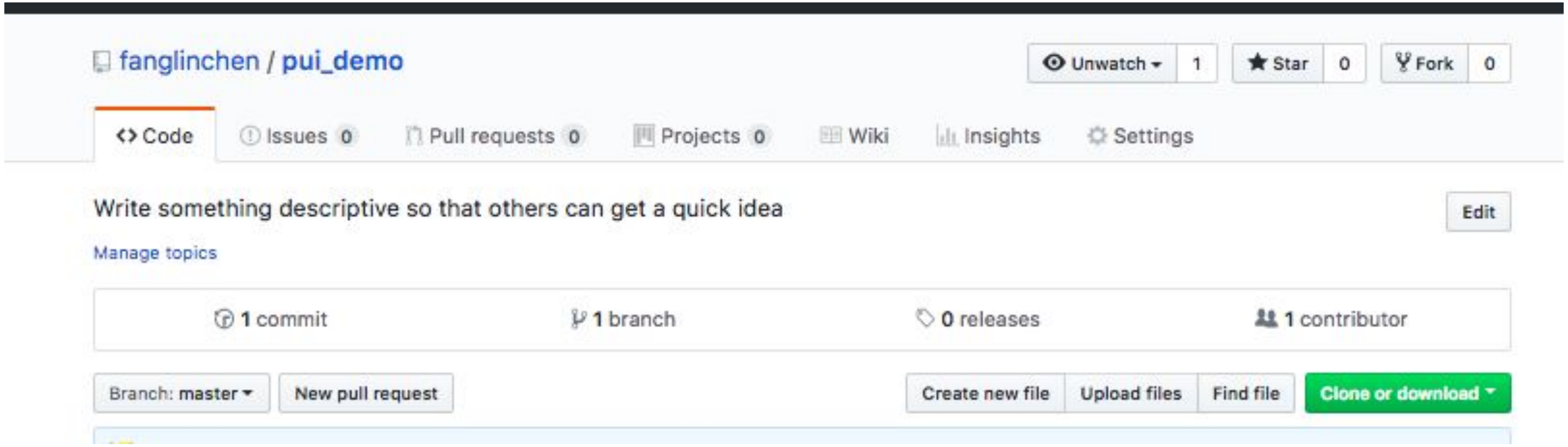
or



nothing is local yet



checkout a repository



fanglinchen / pui_demo

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Write something descriptive so that others can get a quick idea

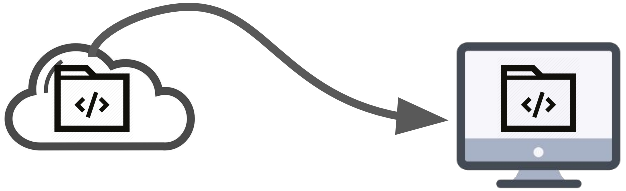
Edit

Manage topics

1 commit | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

only needed when you don't have a local copy of this repo.



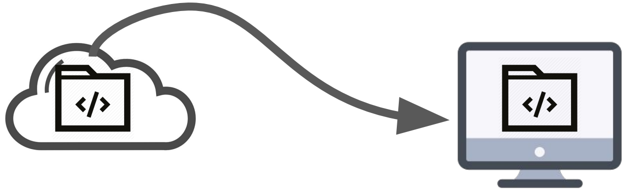
checkout a repository

create a working copy of a local repository

```
git clone /path/to/repository
```

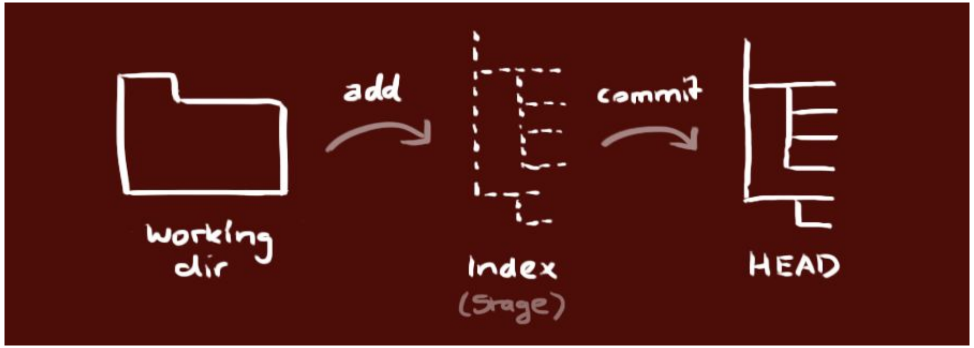
only needed when you don't have a local copy of this repo.

The cloned repository is under the directory where you run this command.



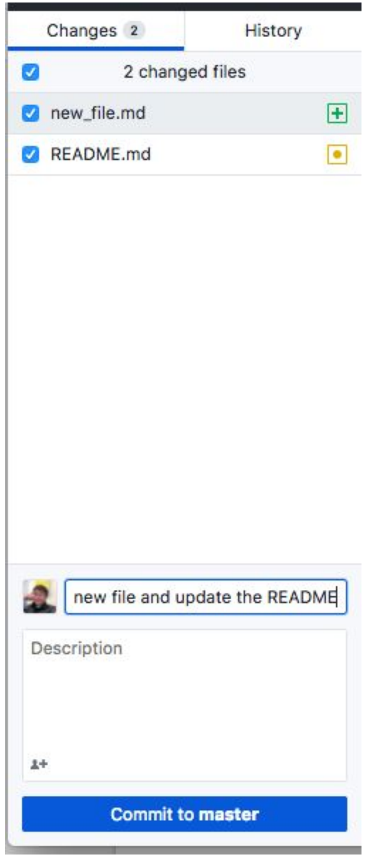
working directory vs. HEAD vs. Index

Your local repository consists of three "trees" maintained by git.
Working Directory holds the actual files.
Index acts as a staging area (or cache).
HEAD points to the last commit you've made.



Reference: http://rogerdudler.github.io/git-guide/

add & commit



Write down commit messages

add & commit

propose changes (add it to Index)

```
git add <filename>
git add .
```

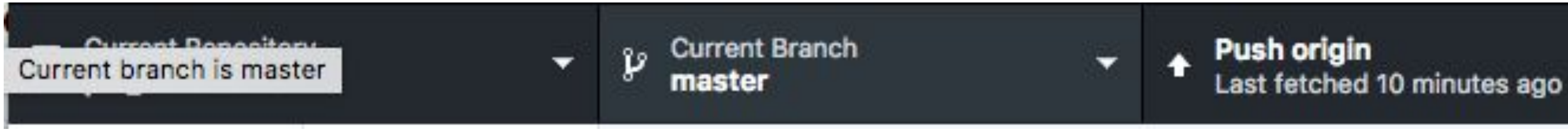
commit these changes

```
git commit -m "Commit message"
```

now the file is committed to the HEAD, but not in your remote repository yet.

+

pushing changes



pushing changes

Send those changes to your remote repository

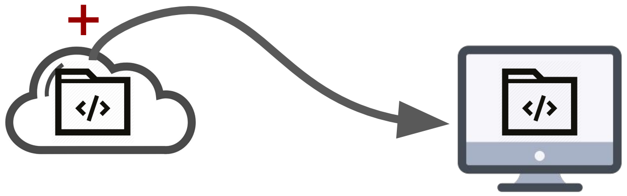
```
git push origin master
```

Change master to whatever branch you want to push your changes to.

update & merge

update your local repository to the newest commit

```
git pull
```



update & merge

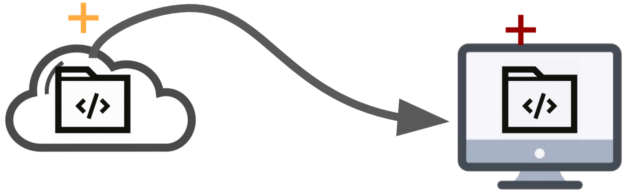
update your local repository to the newest commit

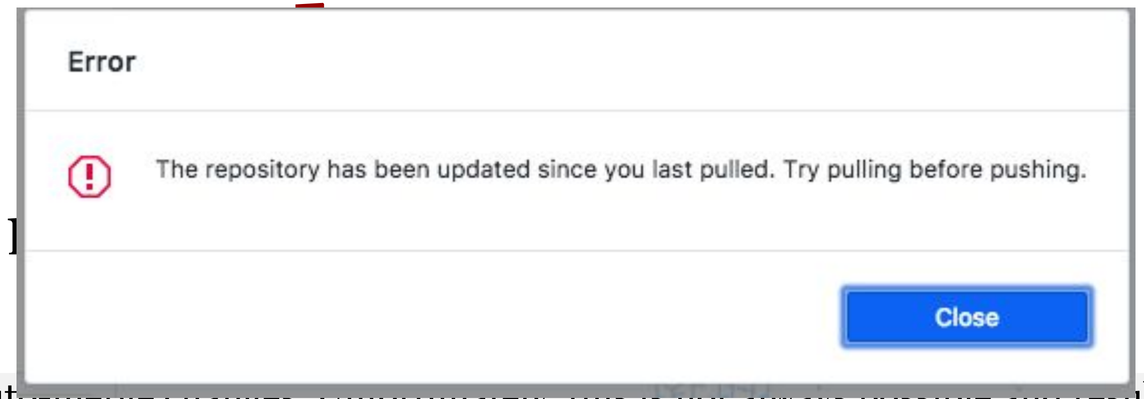
```
git pull
```

Git tries to auto-merge changes. Unfortunately, this is not always possible and results in *conflicts*.

You are responsible to merge those *conflicts* manually by editing the files shown by git.

Conflicts are often found in team collaboration, when there are changes from multiple sources.





Error

The repository has been updated since you last pulled. Try pulling before pushing.

update your l

`git pull`

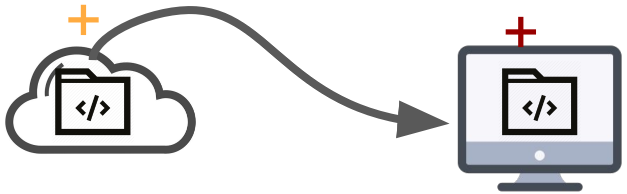
Git tries to auto-merge changes. Unfortunately, this is not always possible and results in *conflicts*.

You are responsible to merge those *conflicts* manually by editing the files shown by git.

For the files in conflict, after manually fixing the conflict

`git add .`

Conflicts are often found in team collaboration, when there are changes from multiple sources.



update & merge

update your local repository to the newest commit

```
git pull
```

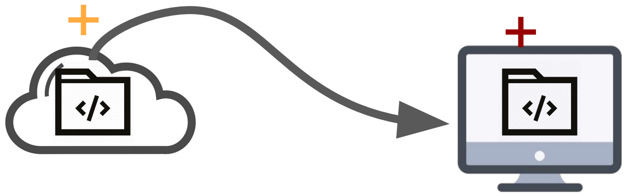
Git tries to auto-merge changes. Unfortunately, this is not always possible and results in *conflicts*.

You are responsible to merge those *conflicts* manually by editing the files shown by git.

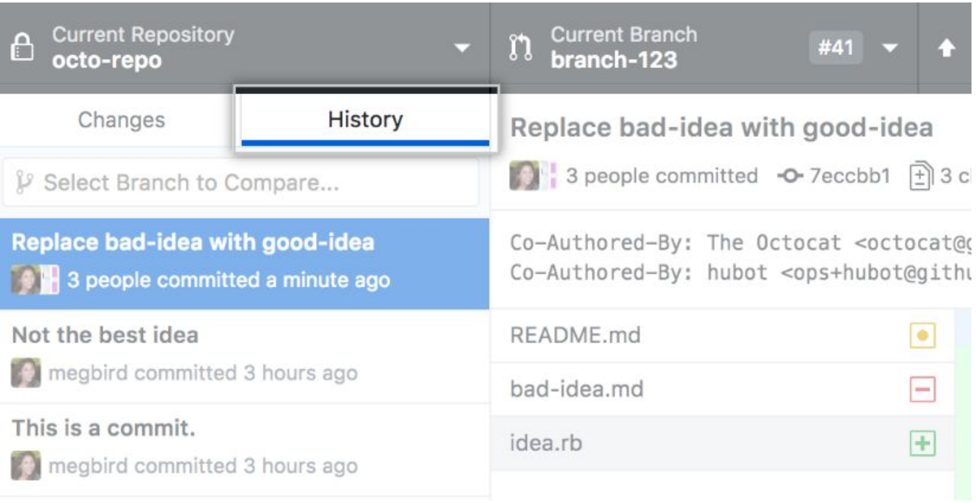
For the files in conflict, after manually fixing the conflict

```
git add .
```

Conflicts are often found in team collaboration, when there are changes from multiple sources.



replace local changes



Current Repository
octo-repo

Current Branch
branch-123

#41

Changes

History

Select Branch to Compare...

Replace bad-idea with good-idea
3 people committed a minute ago

Not the best idea
megbird committed 3 hours ago

This is a commit.
megbird committed 3 hours ago

Replace bad-idea with good-idea

3 people committed 7eccbb1 3 c

```
Co-Authored-By: The Octocat <octocat@g
Co-Authored-By: hubot <ops+hubot@githu
```

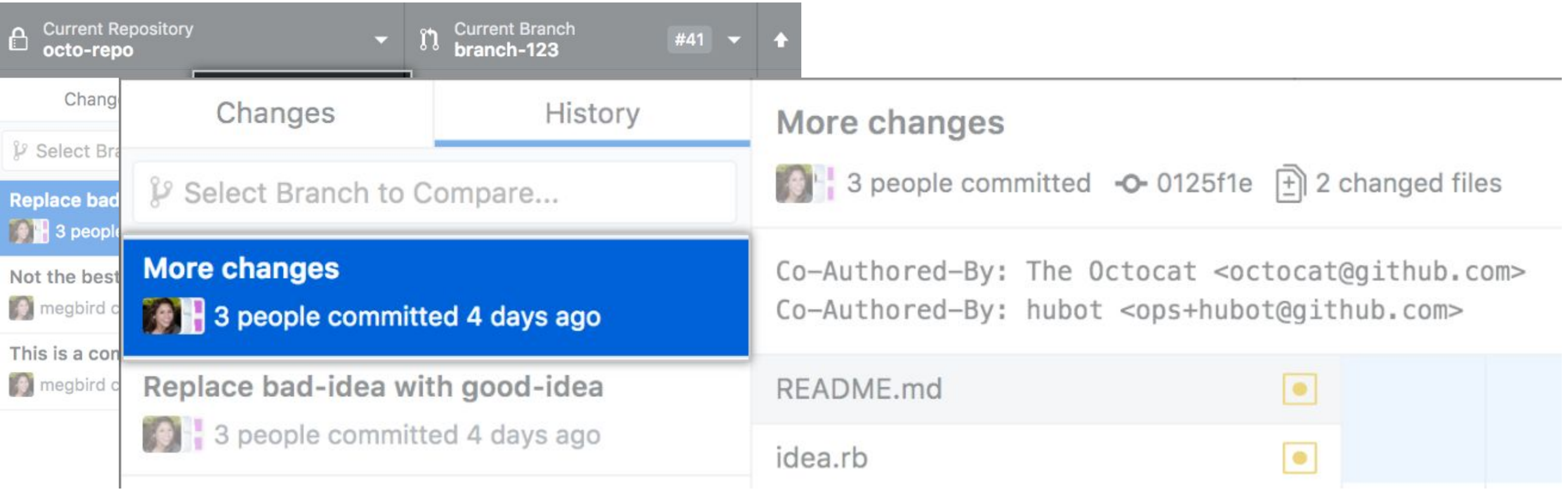
README.md

bad-idea.md

idea.rb

Reference:
https://help.github.com/desktop/guides/contributing-to-projects/reverting-a-commit/

replace local changes



More changes

3 people committed 4 days ago

Replace bad-idea with good-idea

3 people committed 4 days ago

More changes

3 people committed 0125f1e 2 changed files

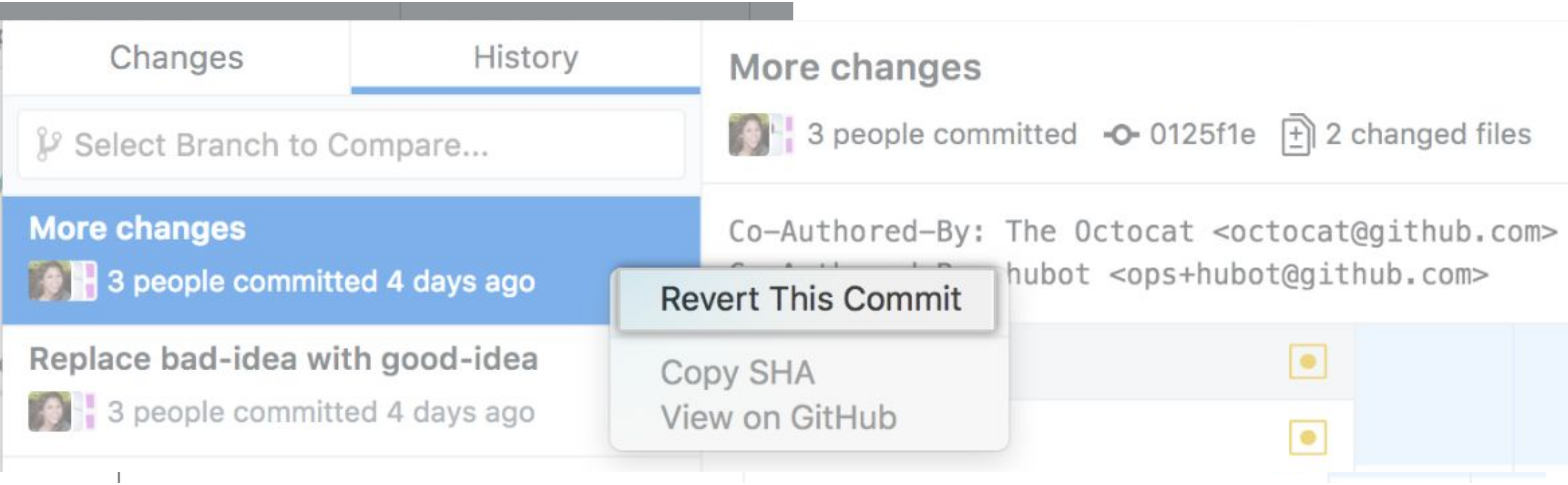
```
Co-Authored-By: The Octocat <octocat@github.com>
Co-Authored-By: hubot <ops+hubot@github.com>
```

README.md

idea.rb

Reference:
https://help.github.com/desktop/guides/contributing-to-projects/reverting-a-commit/

replace local changes



More changes
3 people committed 4 days ago

Replace bad-idea with good-idea
3 people committed 4 days ago

More changes
3 people committed 0125f1e 2 changed files

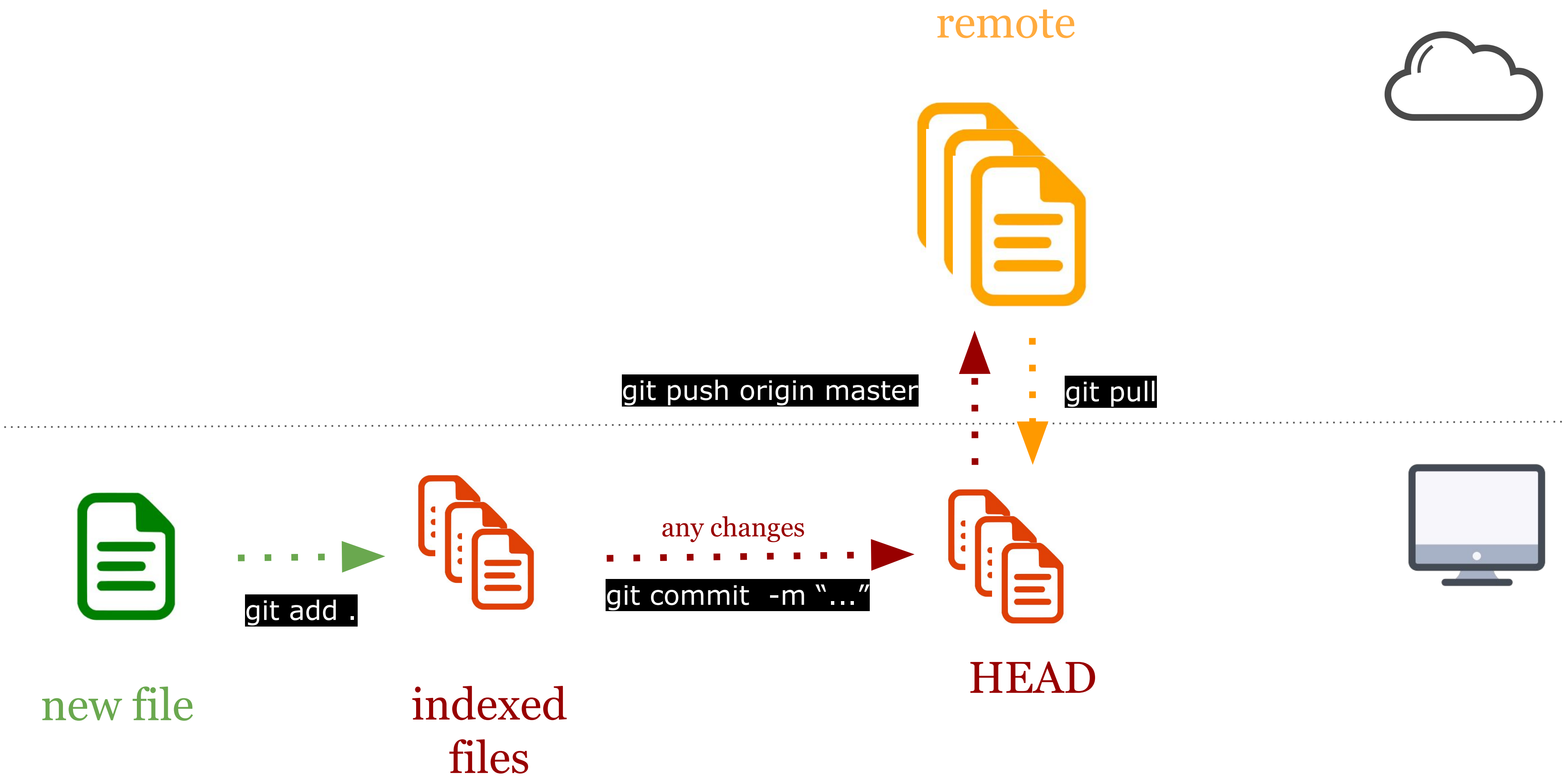
```
Co-Authored-By: The Octocat <octocat@github.com>
Co-Authored-By: hubot <ops+hubot@github.com>
```

Reference:
https://help.github.com/desktop/guides/contributing-to-projects/reverting-a-commit/

replace local changes

go back to a previous version	`git checkout [revision] .`
replace local changes	`git checkout -- <filename>`

Reference:
https://help.github.com/desktop/guides/contributing-to-projects/reverting-a-commit/


remote
git push origin master
git pull
new file
git add .
indexed files
any changes
git commit -m "..."
HEAD

Javascript Basics

Javascript Types

Dynamically typed: you don’t have to specify a type, Javascript will infer it.

var x = 3; // x now has type Number

Basic (Primitive) Types:

- Boolean: **true** or **false**
- Number: **1** or **1.0** // Javascript doesn’t distinguish between integers and floats
- String: **“this is a string”**
- Null: **null** // often used to represent ‘no value’
- Undefined: **undefined** // if a variable has never been assigned something

Functions

JS functions are reusable blocks of code, similar to functions in other languages.

```
function add(a,b) {
  return a + b;
}
```

Call like this: add(2,3) // would return 5

Objects

- An object is a collection of properties and values.
- In JS, this takes the role of classes, dictionaries, etc.

Looks like this: **{ username: “cole”, age: 25, loggedIn: true }**

The values can be anything, including functions or other objects.

Arrays

- An array is an ordered collection of things.
- Elements do not need to be same type.

Define like this: **var myList = [“apple”, “banana”, “citrus”];**

Access an element: **myList[1] // returns “banana”**

Get number of elements: **myList.length**

Add elements: **myList.push(“date”)**

etc.

Control flow

Javascript has a few classic control mechanisms.

```
if (condition) {
  // do something
} else {
  // otherwise
}
```

Control flow

Javascript has a few classic control mechanisms.

```
for (var i = 0; i < 100; i++) {
  // do something for each value of i
}
```

Control flow

Javascript has a few classic control mechanisms.

```
var myList = [“apple”, “banana”, “citrus”];

for (var i = 0; i < myList.length; i++) {

  console.log(myList[i]);

}
```

Control flow

Javascript has a few classic control mechanisms.

```
while (condition) {

  // do something until condition is false

}
```

Browser Javascript

Traditionally, JS has been used in the browser, but today it is used for a ton of stuff.

When you see things like **document** or **window** these are browser-specific Javascript objects. So are elements you get back from their functions.

Want to know what properties they have? Use your Developer console!

Let’s try . . .

Debugging Tips

1. Open up the Developer Tools console and see if there are errors.
2. Is your code being called? Put a **console.log("in X function")** before it.
3. **console.log(object)** to see what objects you are passing to your functions, etc. Make sure they are what you expect!

Demo time

Let’s build something real quick.

Questions?

We will try to take JS or git questions and answer/demo them live.