Git From Scratch

Mohammed Alsobay
Mozn
github.com/alsobay

Hassan Alsibyani
Wasphi
github.com/hsibyani

Version control

It is a way to manage different versions or revisions of the files.

Examples:

- Undo/redo buffers
- Google docs
- Multiple versions

```
sibyani@computer:~$ ls
assignment1-1.go
assignment1-2.go
assignment1-3_work_in_progress.go
```

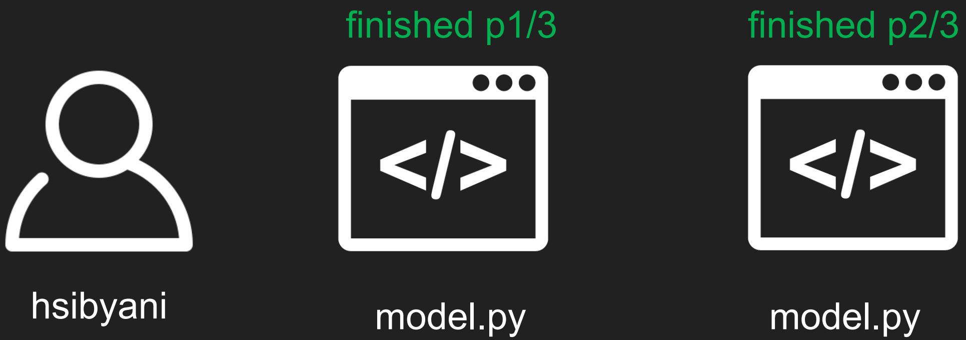
Example:

finished p1/3

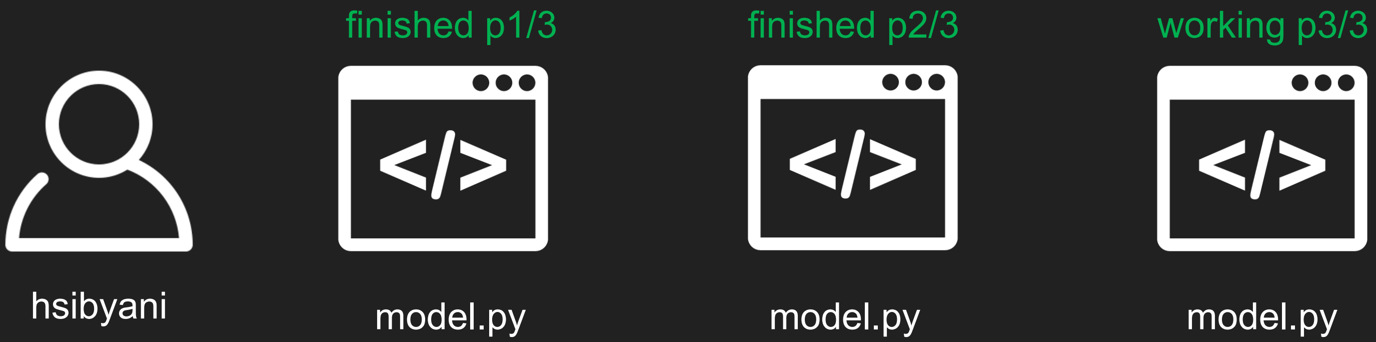
hsibyani

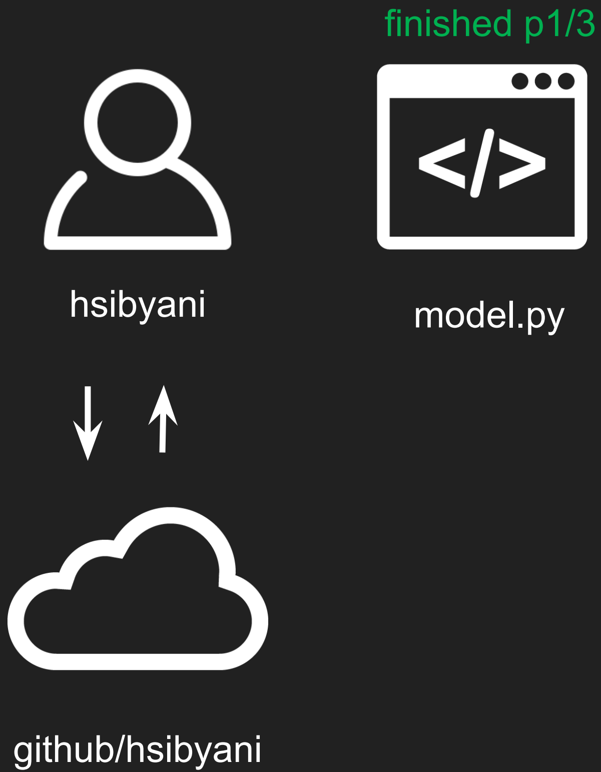
model.py

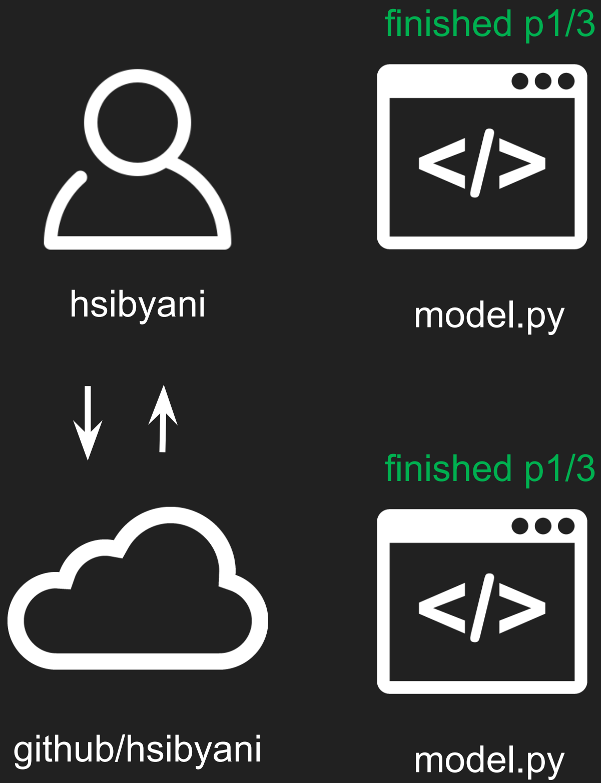
Example:

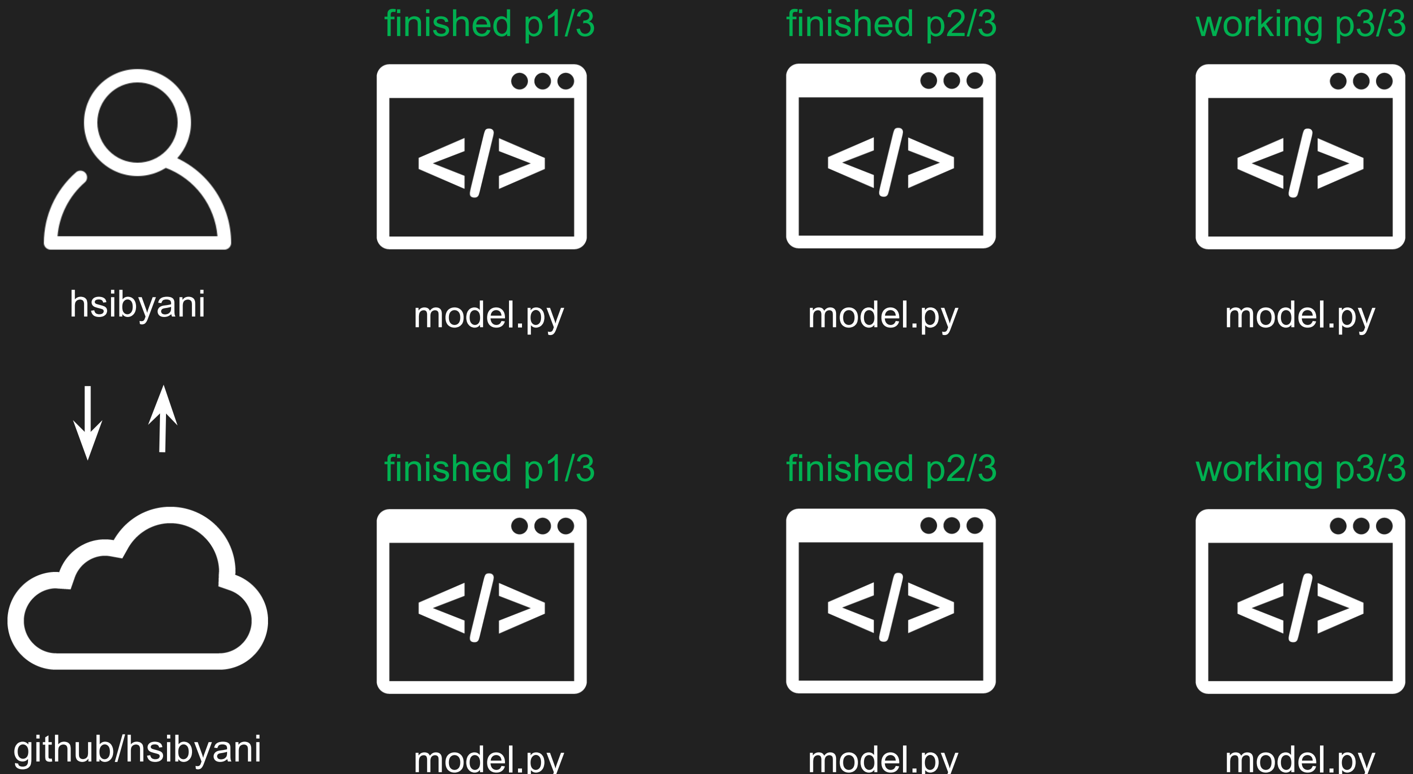


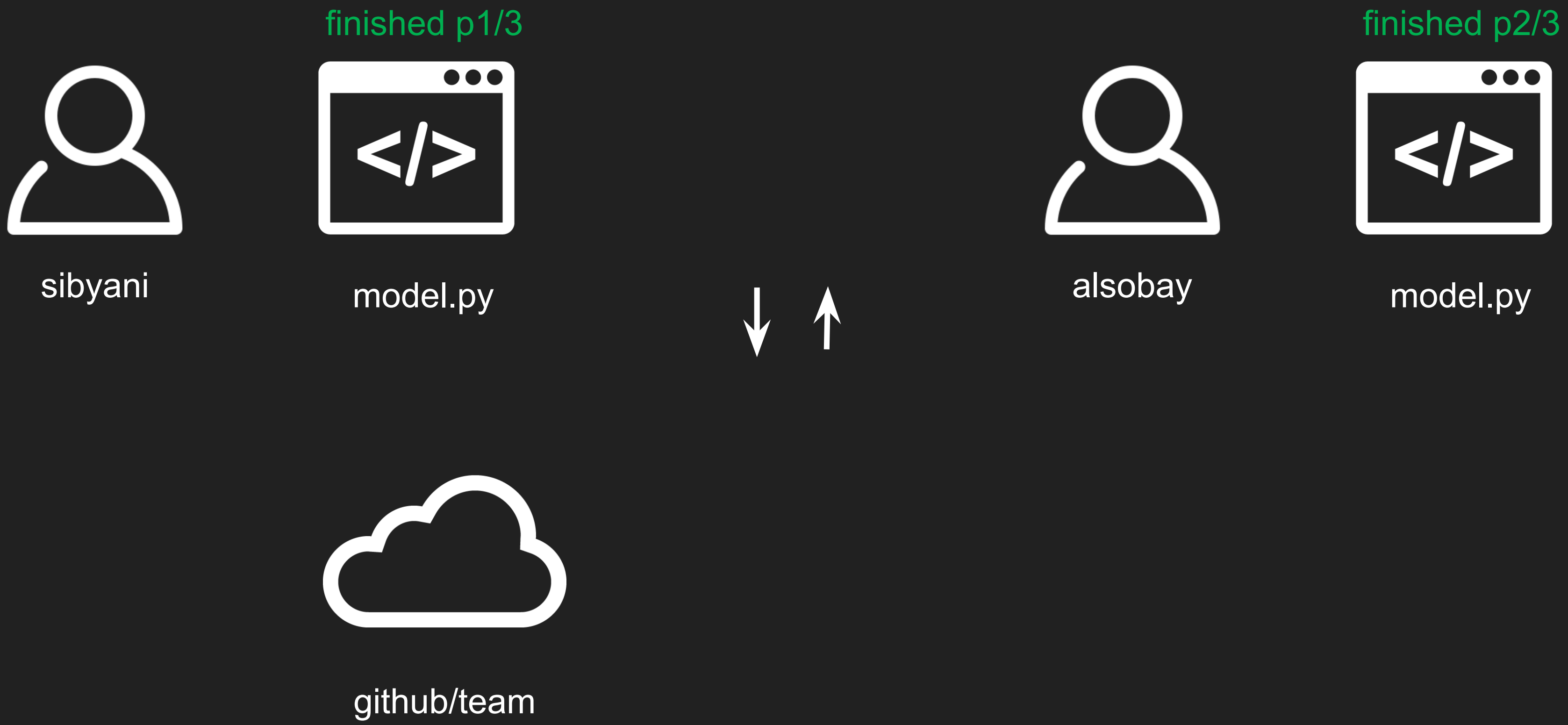
Example:

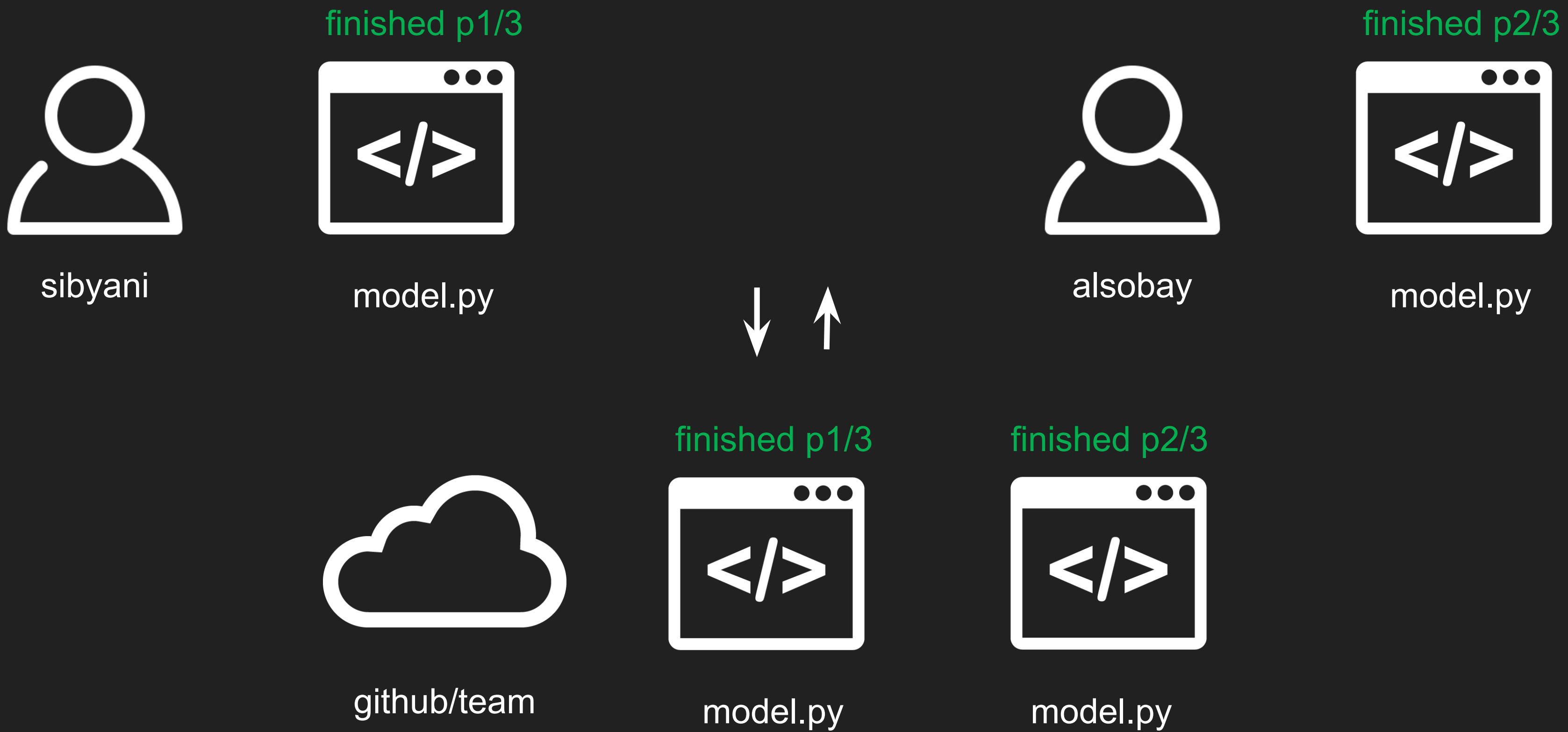

finished p1/3
finished p2/3
working p3/3
hsibyani
model.py
model.py
model.py


finished p1/3
hsibyani
model.py
github/hsibyani


finished p1/3
hsibyani
model.py
finished p1/3
github/hsibyani
model.py


hsibyani
finished p1/3
model.py
finished p2/3
model.py
working p3/3
model.py
github/hsibyani
finished p1/3
model.py
finished p2/3
model.py
working p3/3
model.py


finished p1/3
sibyani
model.py
finished p2/3
alsobay
model.py
github/team


finished p1/3
sibyani
model.py
finished p2/3
alsobay
model.py
finished p1/3
finished p2/3
github/team
model.py
model.py

What is Git, and why should I care?

...and how is it related to Github?

Git is a tool for “version control”, a.k.a. source code management (SCM)

↳ How can you and your team work on the same code without ruining each other’s work?

There are other VC/SCM frameworks!

- MS Team Foundation VC
- Subversion
- Mercurial

Github is a place to host & share your *repositories*. Other options

- GitLab
- Bitbucket

But what’s wrong with naming my files “working”, “workingFinal”, and “workingFinalFinalinshaAllah”?

Using version control, you can roll back to any version you’ve *committed*, and still have a clean folder of *only one copy* of all your files.

PLUS

- Good *commit* messages help you tell a story of how the code was written
- *Branching* is a safe way to collaborate
- Platforms like Github make it easy to add code reviews, cont. integration, etc.

It’s all about repositories.

repository [ri-poz-i-tohr-ee]

a receptacle or place where things are deposited, stored

A repo can be...

- One you started yourself with *git init*
- One you cloned from an existing repository with *git clone [url]*
- A fork of a popular project

Great, but how exactly do my team and I work on the same repo?

Local & Remote Repositories

Working Directory

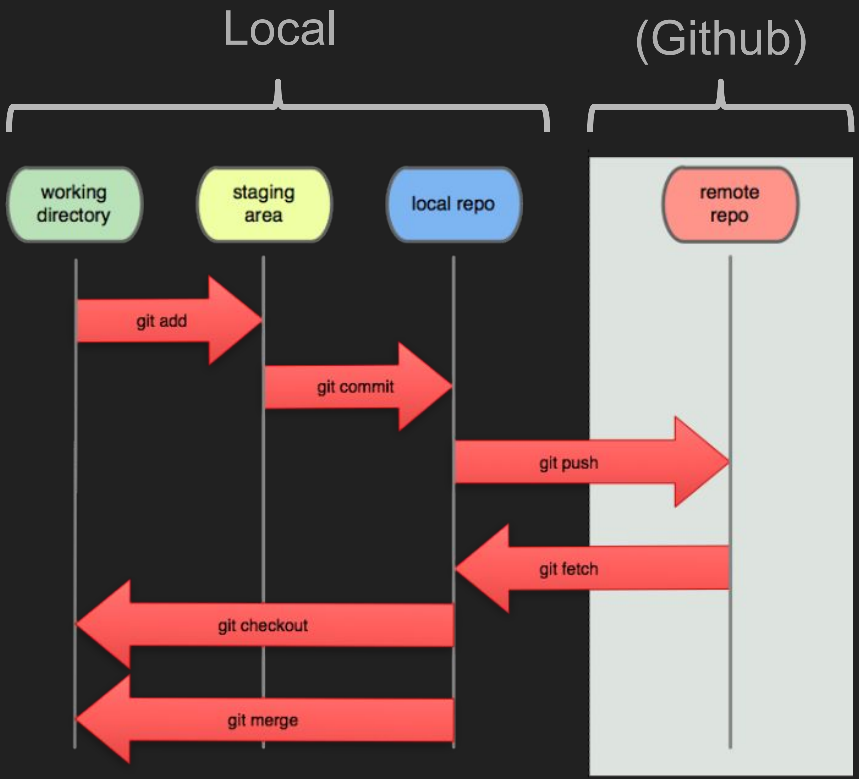
The copy of code you're working on at any time. Changes in this state are not "stored".

Staging Area

This is the collection of files/changes that you have marked to go into the next committed snapshot.

Local Repo (committed)

Once you commit a set of changes, the files in the staging area are stored as a permanent snapshot.



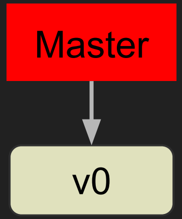
Source: (1) Git, [1.3 Getting Started - Git Basics](), (2) Ido Green, [Git 101 - Useful Commands]()

Non-active

Active

First, start with a new repo

```
>> git init
```

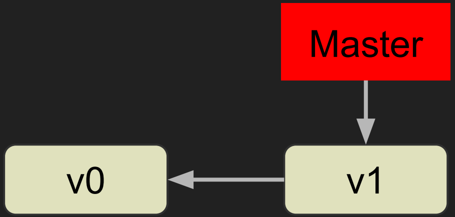


Non-active

Active

Let’s add a feature and commit it

```
*edit file.txt*
>> git add file.txt
>> git commit -m “added website banner”
```



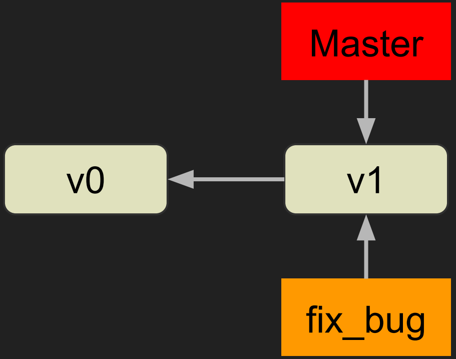
🟧 Non-active

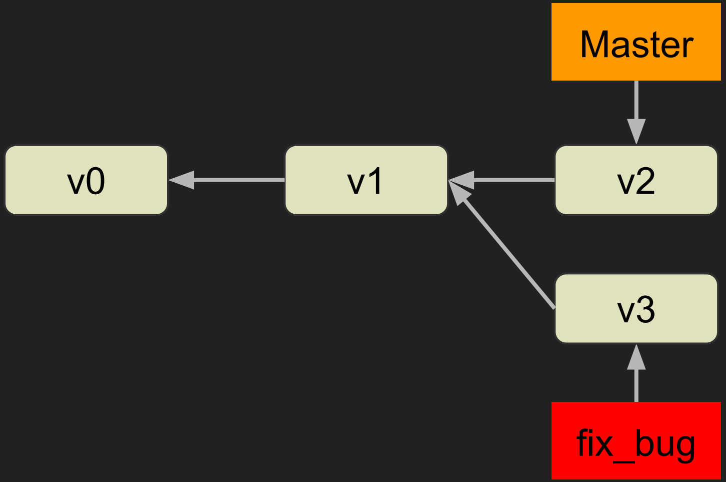
🟥 Active

We want to fix a small bug while development continues, so let's make a branch

```
>> git branch fix_bug
```

```
>> git checkout fix_bug
*make some changes*
>> git add *
>> git commit -m "started bug fix"
```



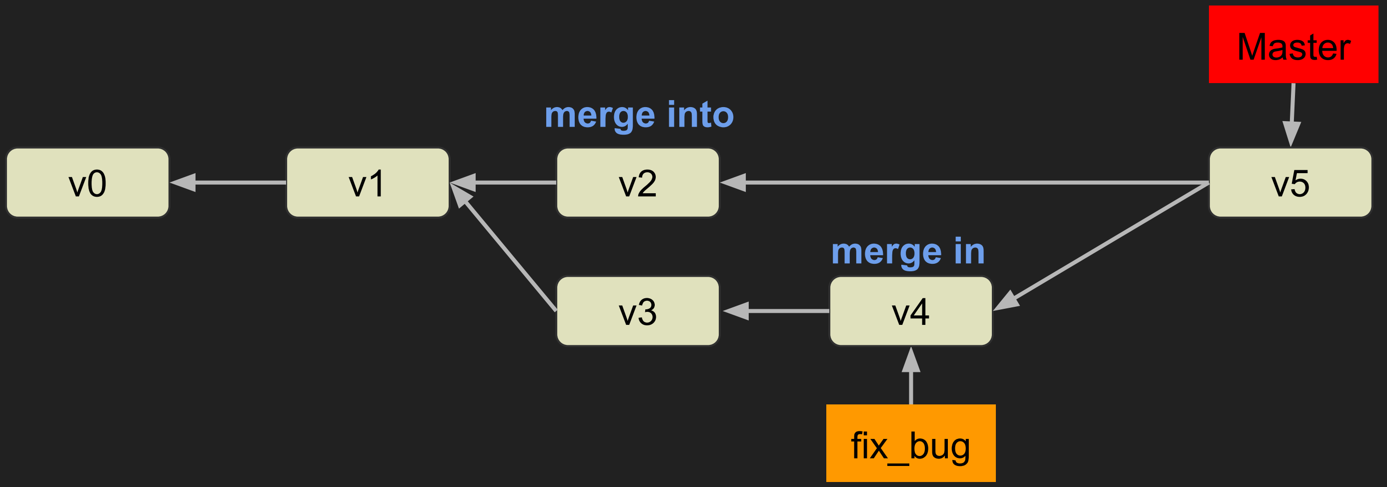


Non-active

Active

Now, let’s bring the bug fix back into our main codebase

```
>> git checkout master
>> git merge fix_bug
```



What if I want to change a commit?

For when you just want to edit the message

```
>> git commit --amend -m “here’s a better commit message”
```

For when you want to edit the snapshot (fix code, forgot a file, etc.)

```
*edit file.txt*
>> git add file.txt
>> git commit --amend --no-edit
```

Now, let's get our hands dirty.

1. Go to github.com/alsobay/git-from-scratch

2. *Fork* the repository from the top right corner Fork

3. Now, clone your fork of the repository (i.e. make a local repo)
   a. git clone https://github.com/[your username]/git-from-scratch.git

   b. You can copy the URL from the repo's site Clone or download

4. Create a file called [firstname] _ [lastname].txt, and write a fun fact about yourself

Bring it all together!

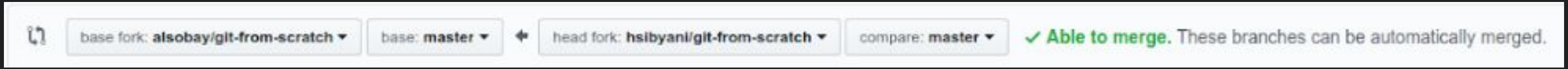
5. Do “git status”. What do you see?
6. Move your txt file to the staging area. [Hint: you’re *adding* it]
7. Do “git status” again. What changed?
8. Commit your changes. Don’t forget to add a helpful message with -m !
9. Create a branch and check it out with “git checkout -b [branch_name]”
10. Push the commit to your remote copy of the repository

Hey, alsobay, add my contribution!

11. Go to your copy of the repository on Github. Notice anything new?

12. Start a *pull request* by clicking Create pull request
    a. Don't forget to add a short message explaining your PR

Notice how the PR says you'll be merging into my master copy of the repo?



base fork: **alsobay/git-from-scratch** ▾ base: **master** ▾ ← head fork: **hsibyani/git-from-scratch** ▾ compare: **master** ▾ ✓ **Able to merge.** These branches can be automatically merged.

Time for me to merge your changes!

Resources

- Learn git in 15 minutes: https://try.github.io/levels/1/challenges/1
- Oh *bleep*, git! For when things go wrong: http://ohshitgit.com/
- Pro Git Book: https://git-scm.com/book