INTRO TO git

AUGUST GUANG

WHAT IS git?

- open source version control tool written by Linus Torvalds ()
- **version control:** tracks and manages changes to documents, computer programs, and other collections of information

WHY USE GIT?

- Tracking changes over time
- Helps with collaboration on the same software
- Protect stable/production code from bugs

WEBHOSTS

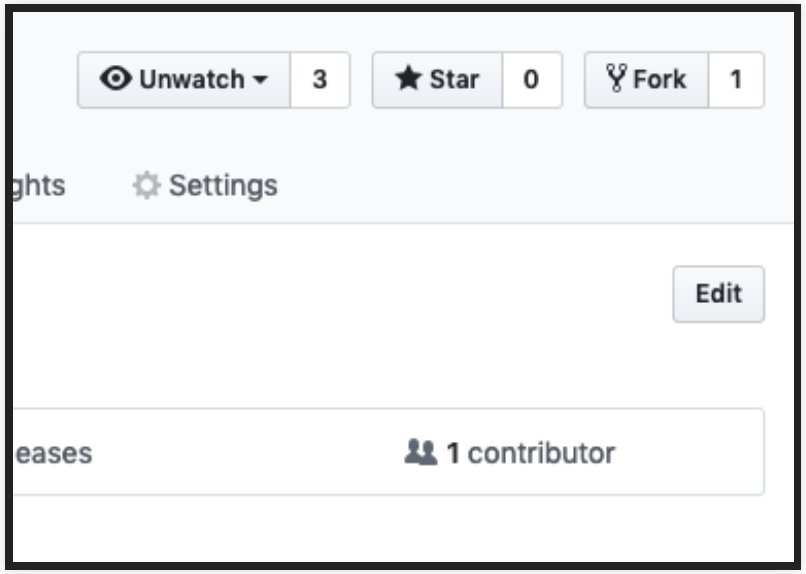
 *Gitlab*  *Github*  *Bitbucket*

git BASICS - FORKING

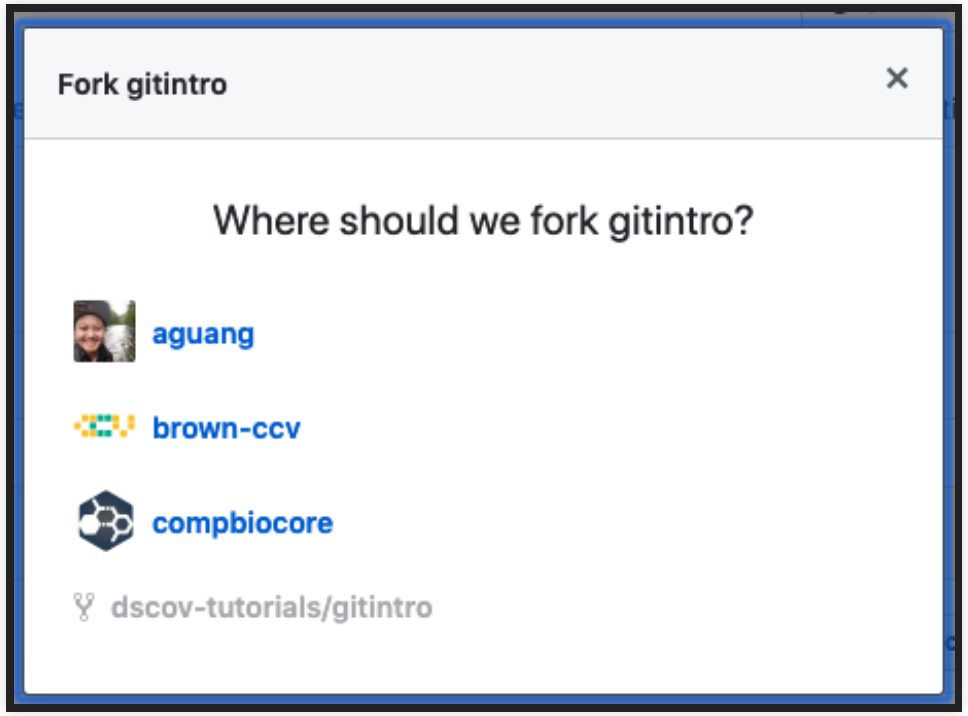
*First, let's create our own copy of the repository this presentation is hosted on by **forking** it.*
Forking a repository is done when you want to a copy of a repository that someone else owns but you want to make your own changes to it for your own purposes.

Go to the Github repo of this tutorial:
https://github.com/dscov-tutorials/gitintro.

Click on the Fork button in the upper right.



It will ask where you want to fork it to. Pick your own account.



After a bit, Github will redirect you to a complete copy of this repo, but now owned by you. You can tell it is a fork because it will say at the top in small font "forked from dscov-tutorials/gitintro".

Go ahead and clone the repository by going to the terminal and typing in the below.

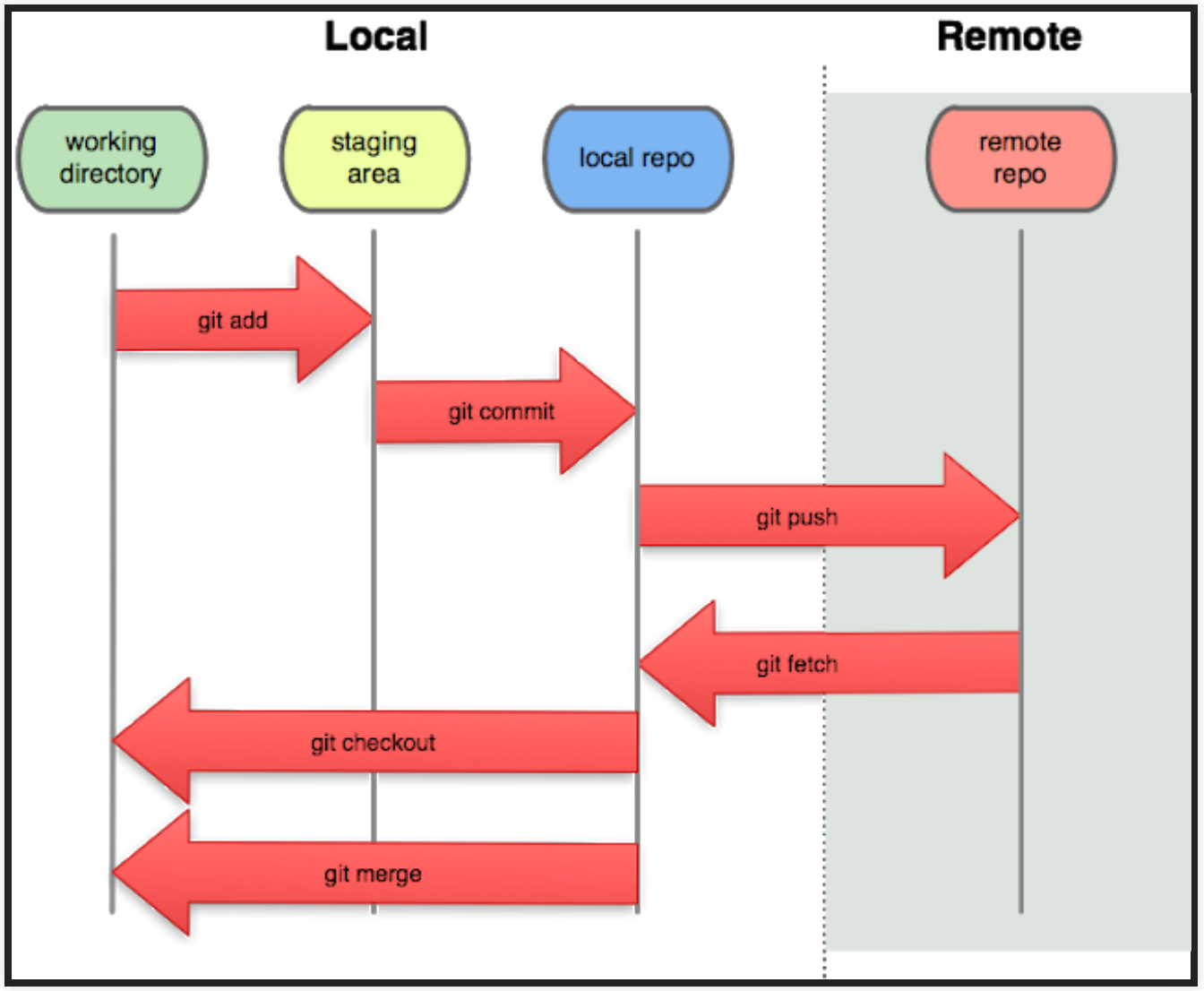
```
git clone https://github.com/$USERNAME/gitintro.git
```

(In general you can get a clone address by clicking the green **Clone or download** icon on your repo's github page)

git BASICS

- **repository or repo:** where documents, software, etc is stored and the changes are tracked

```
gitintro
├── img
│   └── images.png
├── PITCHME.md
└── README.md
```



https://hoantran-it.blogspot.com/2016/06/git-tutorial-

```
# check status of your git repo including what's changed
# and what's not being tracked
git status
# add file contents to be ready to be committed
git add FILE
# commit file contents to the local repository
git commit FILE
# commit all added/modified/deleted file contents with
# specific message
git commit -a -m "commit message"
# push file contents to the remote (i.e. cloud) repository
git push
```

`git status` reveals that everything is up to date.

```
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

```
Working Directory          | Local                      | Remote
 ---                       | ---                        |
gitintro                   | gitintro                   | gitintro
├── img                    | ├── img                    | ├── img
|   └── images.png         | |   └── images.png         | |   └── im
├── PITCHME.md             | ├── PITCHME.md             | ├── PITCHM
└── README.md              | └── README.md              | └── README
```

Let's create a file.

```
echo "test" > test.txt
git status
```

```
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committe

    - test.txt

nothing added to commit but untracked files present (use "git
```

`git add test.txt` adds the file to the staging area.

```
git add test.txt
git status
```

```
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:  test.txt/
```

```
Working Directory          | Local                        | Remote
 ---                       | ---                          |
 gitintro                  |  gitintro                    |  gitintro
 ├── img                   |  ├── img                     |  ├── img
 |   ├── images.png        |  |   ├── images.png          |  |   ├── i
 ├── test.txt              |  └── README.md               |  └── READM
 └── README.md             |                              |
```

`git commit -a -m "test.txt"` actually commits it to local repo.

```
git commit -a -m "test.txt"
git status
```

```
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

```
Working Directory          | Local                      |   Re
 ---                       | ---                        |
 gitintro                  |  gitintro                  |  git
 ├── img                   |  ├── img                   |  ├──
 |   ├── images.png        |  |   ├── images.png        |  |
 ├── test.txt              |  ├── test.txt              |  └──
 └── README.md             |  └── README.md             |
```

`git log` will show a log of everyone's commits and messages.

```
commit 335531d99fd3987a169121307965e28e75de4dbf (HEAD -> maste
Author: August Guang <august.guang@gmail.com>
Date:   Wed Mar 13 13:56:59 2019 -0400

    test.txt

(and so on)
```

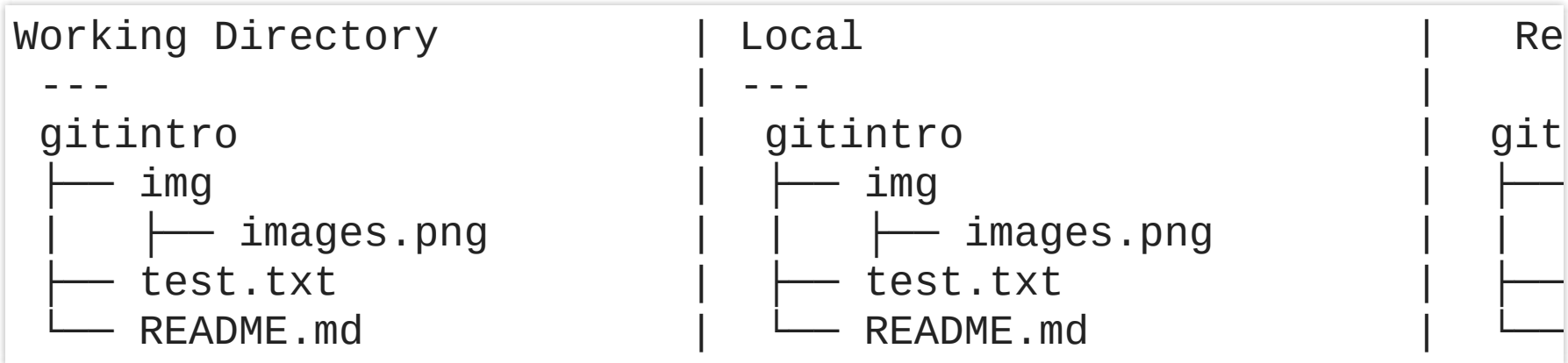
`git push origin master` pushes everything from the local repository to the remote repository.

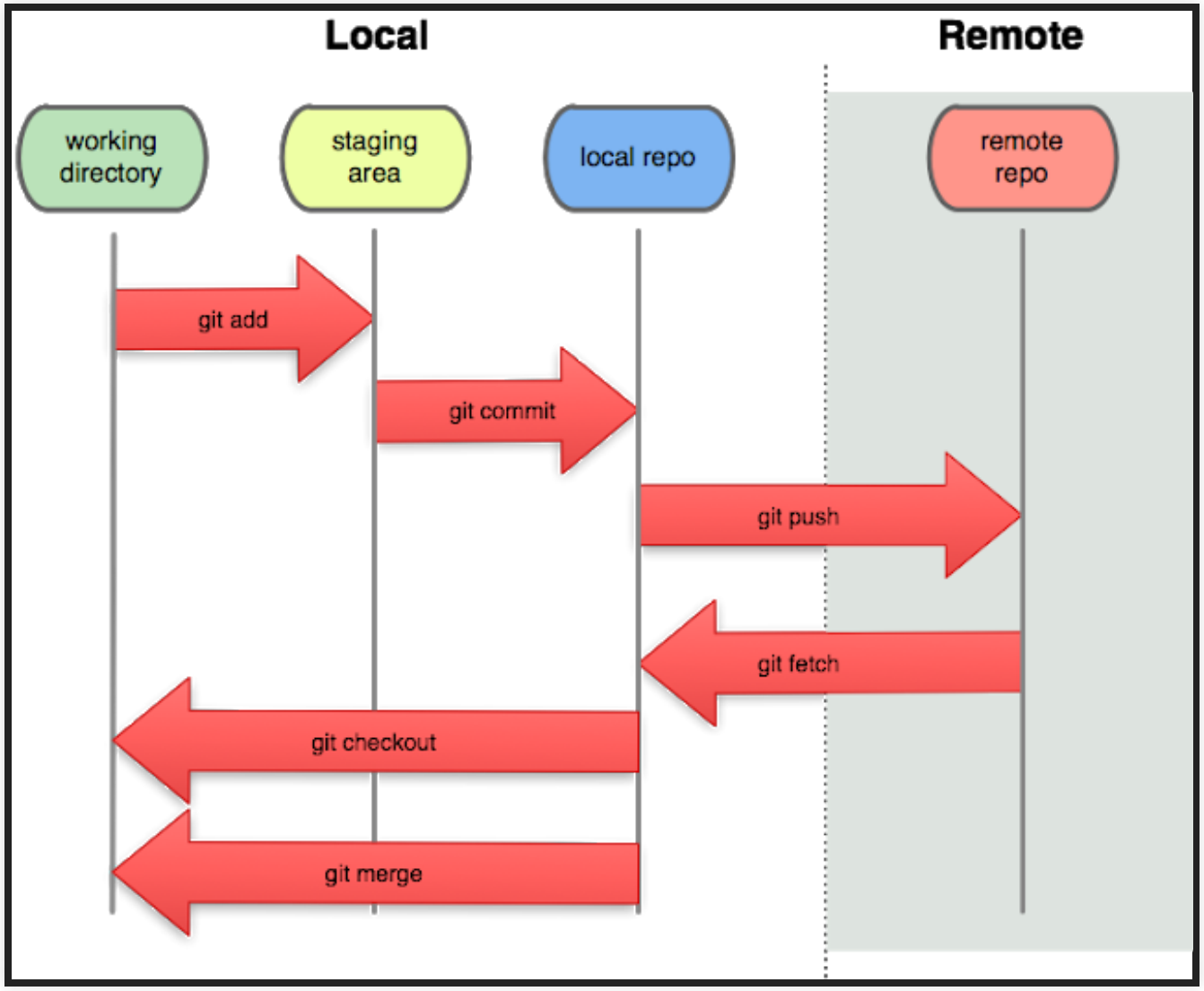
```
git push origin master
```

```
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 274 bytes | 274.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local o
To https://github.com/aguang/gitintro.git
   2fb5c0d..335531d  master -> master
```

```
git status
```

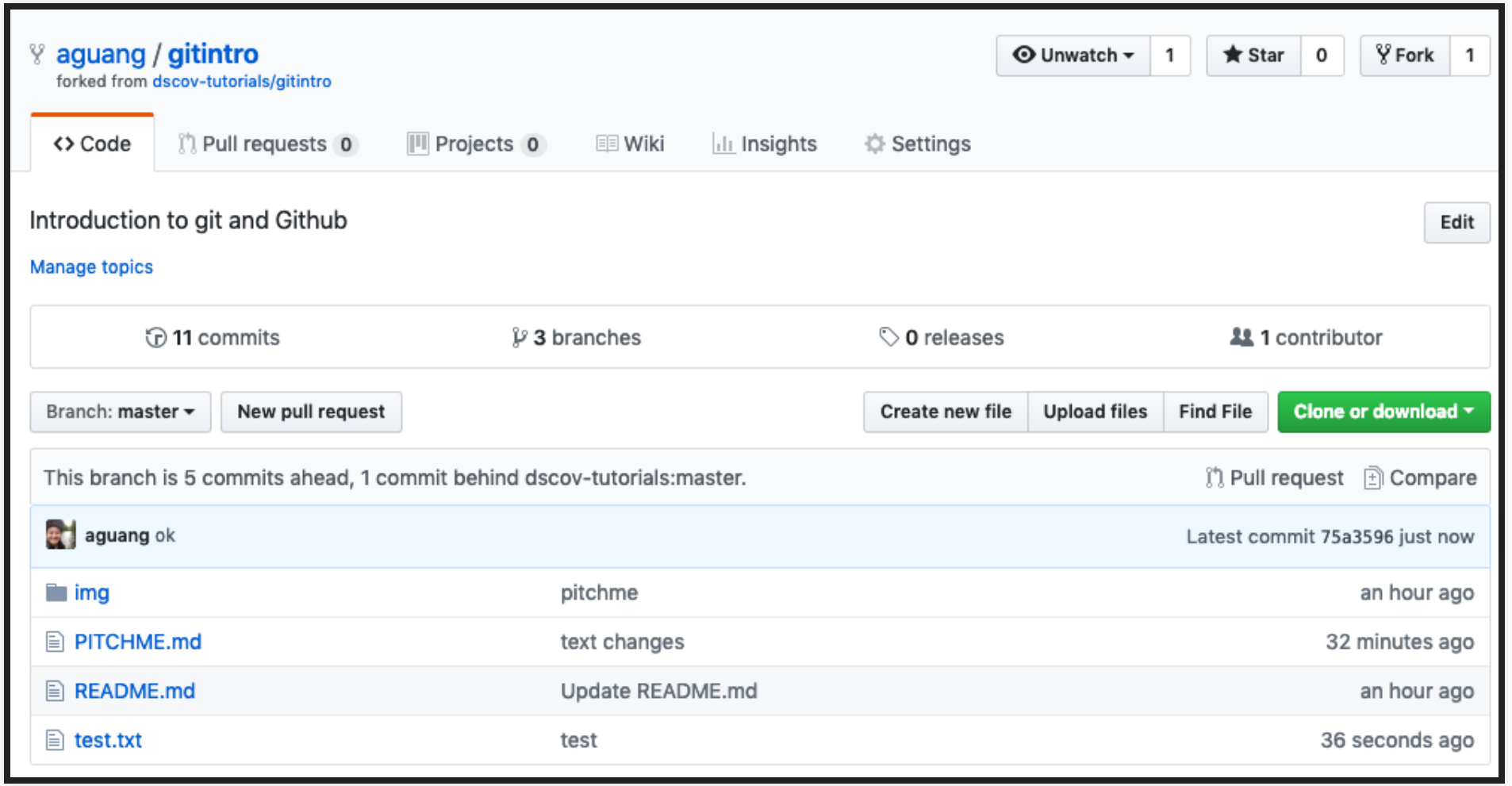
```
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
```


Working Directory
---
gitintro
├── img
│ ├── images.png
├── test.txt
└── README.md
Local
---
gitintro
├── img
│ ├── images.png
├── test.txt
└── README.md
Re
git
├──
│
├──
└──

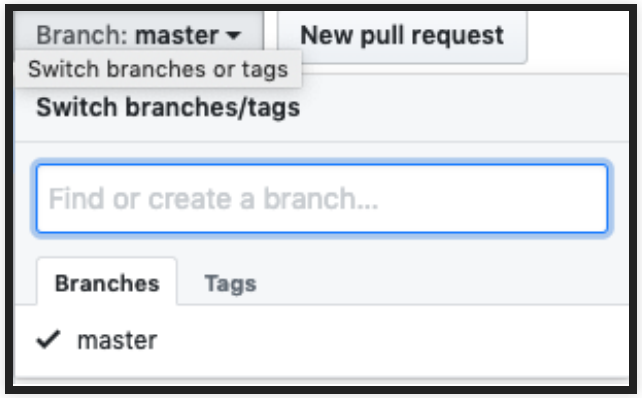

Local
Remote
working directory
staging area
local repo
remote repo
git add
git commit
git push
git fetch
git checkout
git merge

```
# pull data from remote repo into local repo
git fetch
# tells you what branch you have locally and what
# branch your working directory is on
git branch
# tells you what branch you have remotely
git branch -r
# used to navigate between branches on the local repo
# need to run git fetch first to pull in branches
git checkout branch
# merges data from local repo into working directory
git merge
# used to create a new branch
git checkout -b NEW_BRANCH
# combines fetch & merge all at once
```

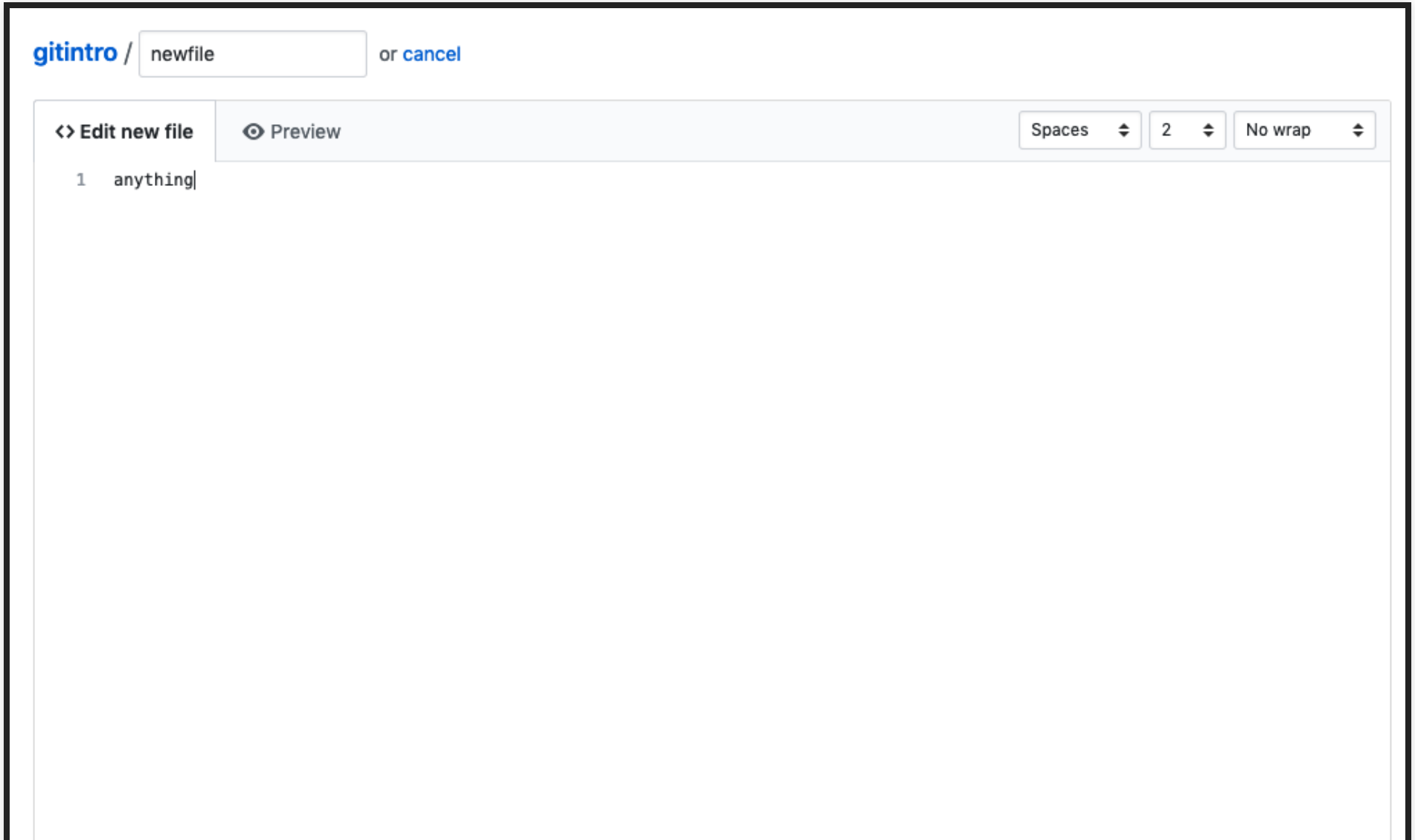
Let's first make a new branch called `example` on Github and add a file `newfile`. Go back to the view of your repository on the browser. It should look like you have a file `test.txt` in it now.



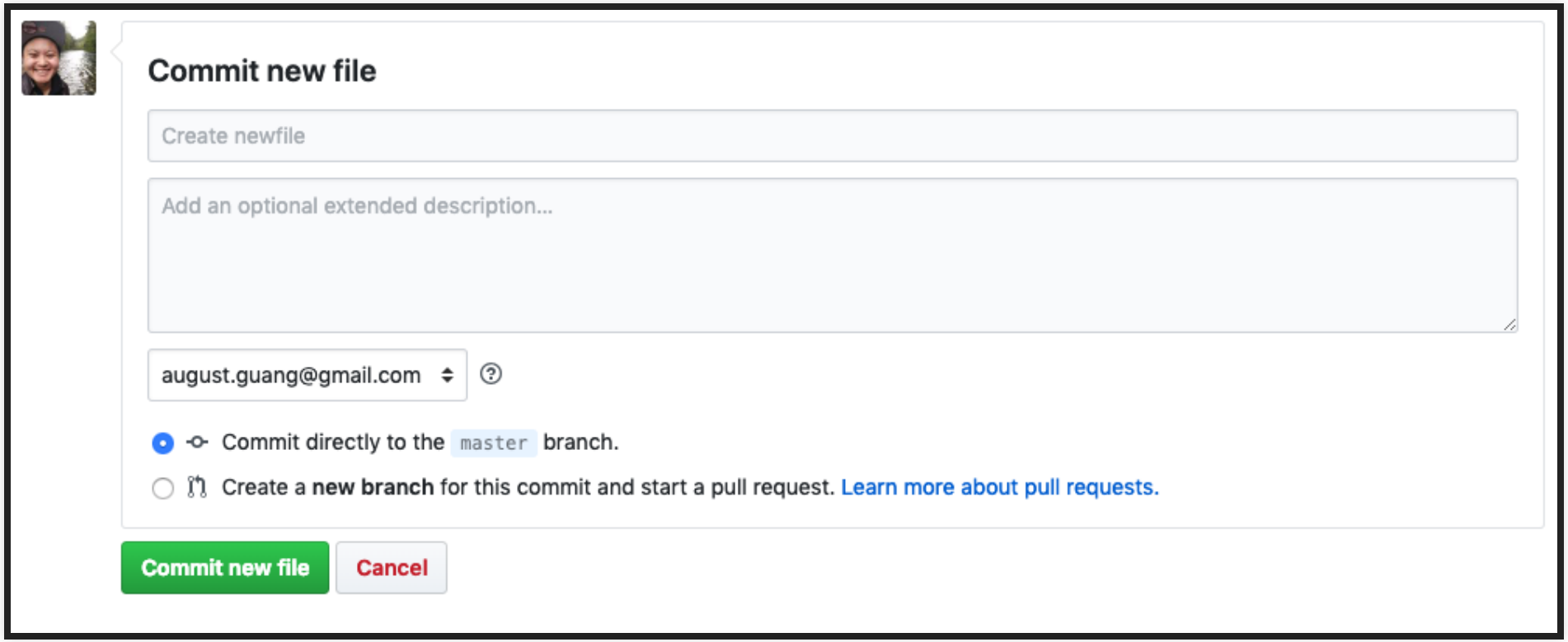
Click on the Branch: **master** icon. A window will appear with the ability to type in a new branch name. Type in `example`.



Now you are on a new branch called `example`. Click on the "Create new file" icon. This will take you to an editing screen. Type in "newfile" inside Name your file... and put whatever you want into the blank text underneath.



Then scroll down and hit the green "Commit new file" button.



You have now created a new file `newfile` inside the branch `example` *remotely*. So how do we get it into local?

Current structure

```
+ remote
master                          | example
 ---                            | ---
 gitintro                       |  gitintro
 ├── img                        |  ├── img
 │   ├── images.png             |  │   ├── images.png
 ├── test.txt                   |  ├── newfile
 └── README.md                  |  └── README.md


- local
master
 ---
 gitintro
 ├── img
```

`git branch` tells you what branch you are on. `git branch -r` tells you what you have in your remote.

```
git branch
git branch -r
```

```
(base) aguang@cis240l0htdh:~/CORE/workshops/dscov/test/gitintr
* master
(base) aguang@cis240l0htdh:~/CORE/workshops/dscov/test/gitintr
  origin/HEAD -> origin/master
  origin/master
```

`git fetch` pulls data from remote repo into local repo.

```
git fetch
```

```
From https://github.com/aguang/gitintro
 * [new branch]      example        -> origin/example
```

```
+ remote
master                    | example
 ---                      | ---
 gitintro                 |  gitintro
 ├── img                  |  ├── img
 |   ├── images.png       |  |   ├── images.png
 ├── test.txt             |  ├── newfile
 └── README.md            |  └── README.md

- local
master                    | example
 ---                      | ---
 gitintro                 |  gitintro
 ├── img                  |  ├── img
```

```
git branch
git branch -r
```

```
(base) aguang@cis240l0htdh:~/CORE/workshops/dscov/test/gitintr
* master
(base) aguang@cis240l0htdh:~/CORE/workshops/dscov/test/gitintr
  origin/HEAD -> origin/master
  origin/example
  origin/master
```

`git checkout example` pulls in an exact copy from local.

```
git checkout example
```

```
Branch 'example' set up to track remote branch 'example' from
Switched to a new branch 'example'
```

```
+ remote
master                    | example
 ---                      | ---
 gitintro                 |  gitintro
 ├── img                  |  ├── img
 |   ├── images.png       |  |   ├── images.png
 ├── test.txt             |  ├── newfile
 └── README.md            |  └── README.md

- local
master                    | example
 ---                      | ---
 gitintro                 |  gitintro
 ├── img                  |  ├── img
```

`git merge $BRANCH` merges everything from local current branch $BRANCH into working directory.

```
+ remote
master                      | example
 ---                        | ---
 gitintro                   |  gitintro
 ├── img                    |  ├── img
 │   ├── images.png         |  │   ├── images.png
 ├── test.txt               |  ├── newfile
 └── README.md              |  └── README.md

- local
master                      | example
 ---                        | ---
 gitintro                   |  gitintro
 ├── img                    |  ├── img
 │   ├── images.png         |  │   ├── images.png
```

If some of your files get overwritten you may get merge conflicts. These you will have to fix file by file and line by line to decide which version you want to keep. The merge conflicts will be marked with text like

```
>>>>>>>>>>>HEAD
code_version1
<<<<<<<<<<<commit
code_version2
```

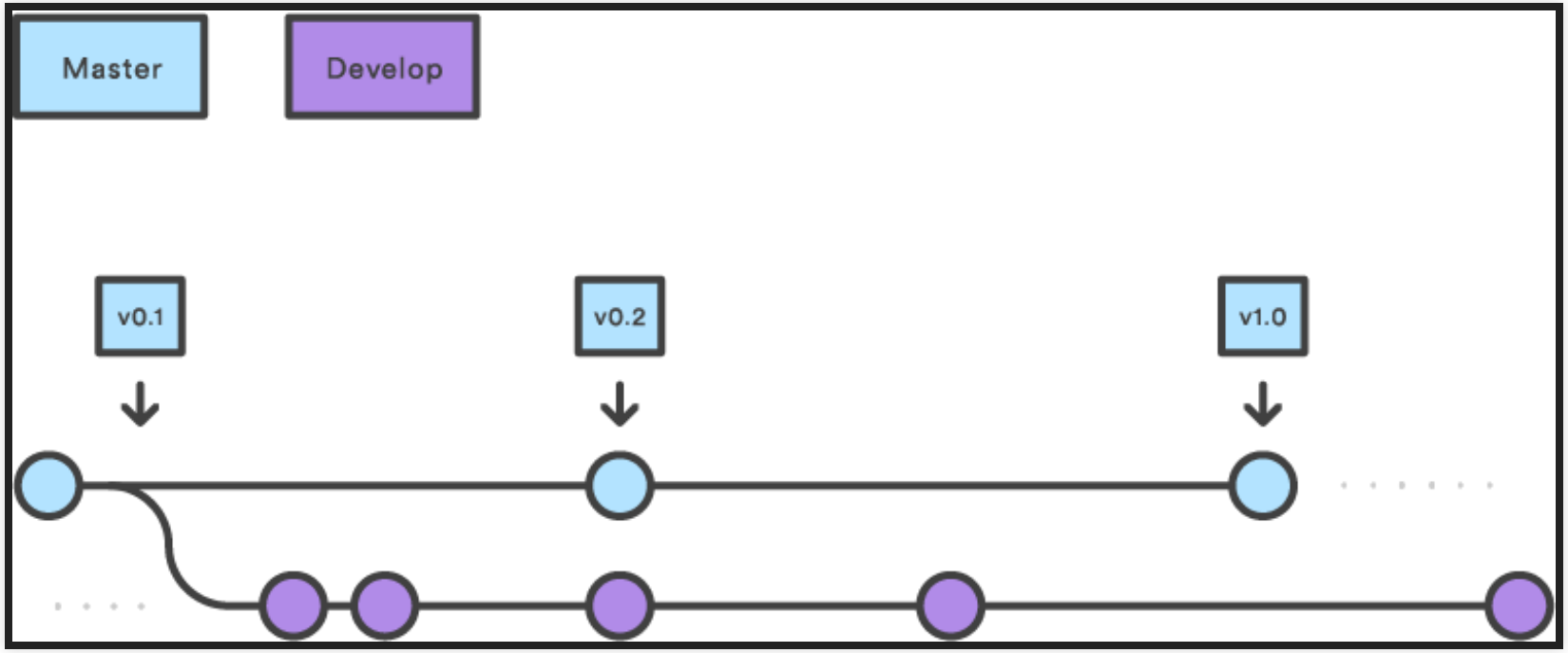
`git checkout -b NEW_BRANCH` creates a new branch locally and switches the working directory over.

`git branch` tells you what branch you are on, this time with NEW_BRANCH added.

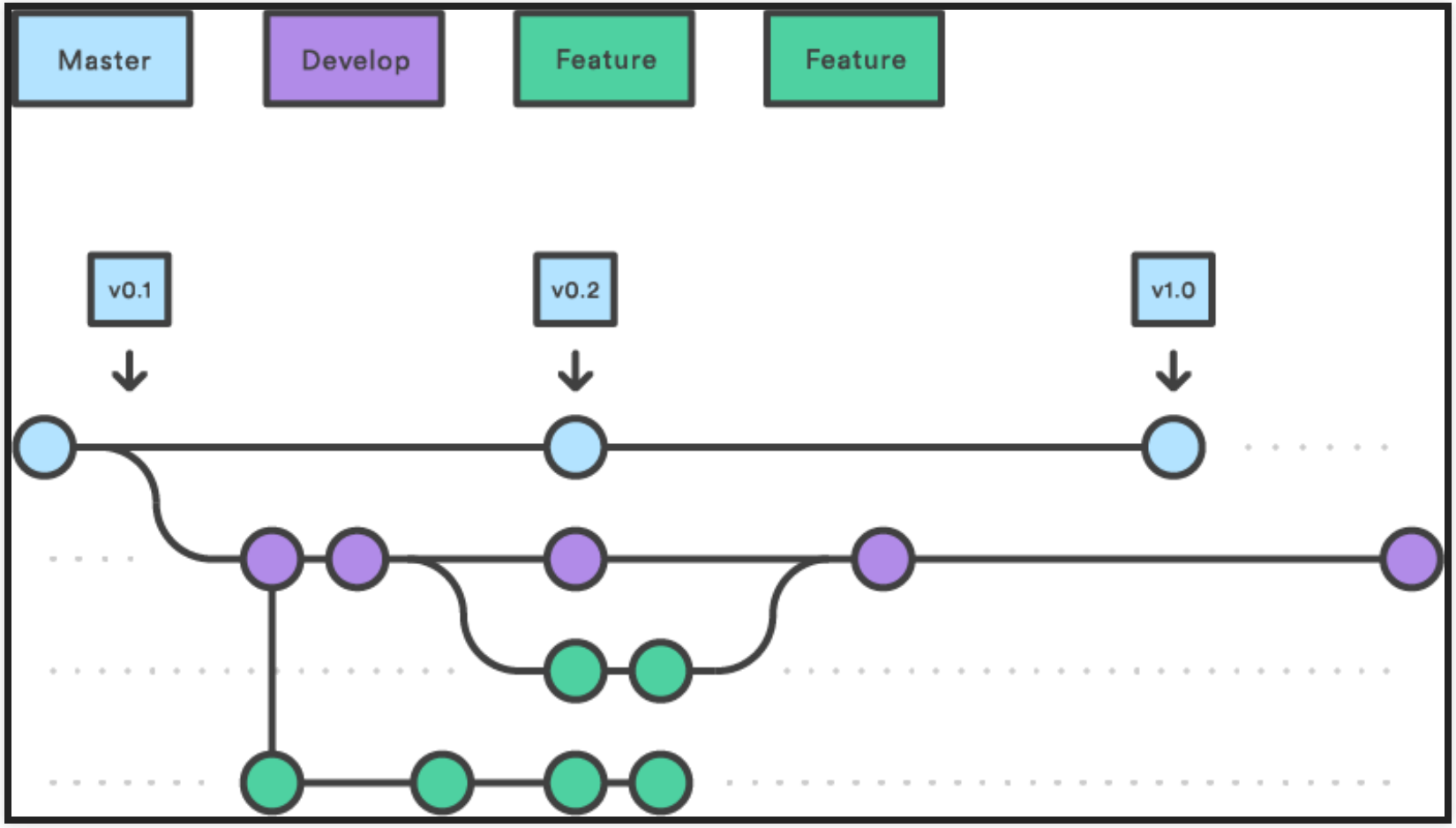
```
git branch
* NEW_BRANCH
  example
  master
```

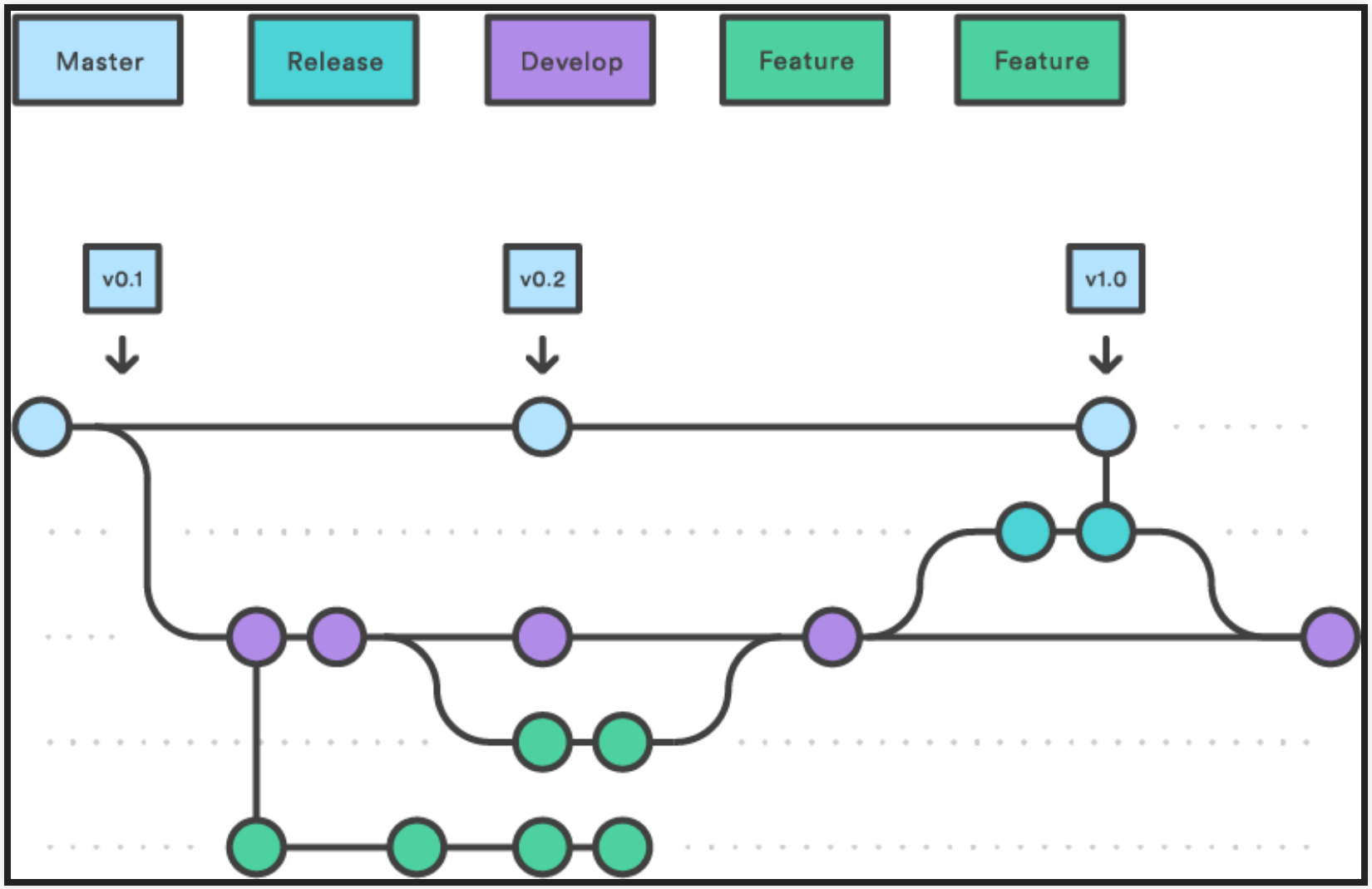
git WORKFLOWS

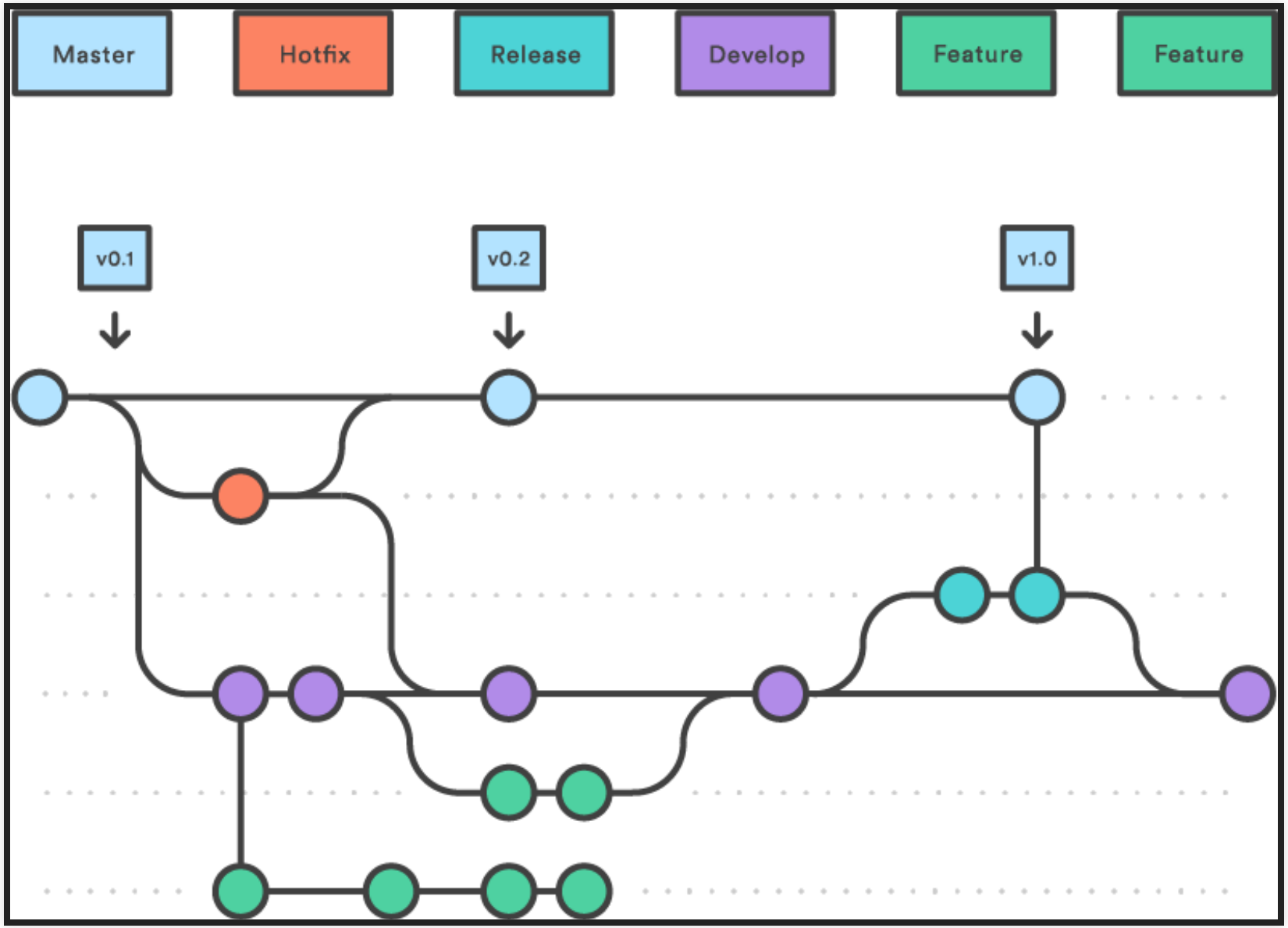
GITFLOW



https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow


Master
Develop
Feature
Feature
v0.1
v0.2
v1.0


Master
Release
Develop
Feature
Feature
v0.1
v0.2
v1.0


Master
Hotfix
Release
Develop
Feature
Feature
v0.1
v0.2
v1.0

GITHUB FLOW

https://guides.github.com/introduction/flow/

FUN THINGS

SLACK INTEGRATION

- Can subscribe a channel to a Github repository so everyone in the channel sees commits, pushes, etc and can comment on them
- Useful for individual projects

CODE BLOCK DELIMITER

src/gitpitch.md

SOURCE FILE NOT FOUND

OTHER INTEGRATIONS: TRAVIS, CODECOV, NOTEBOOKS

http://github.com/aguang/transmissim

LEARN MORE

- Generally  http://www.stackoverflow.com is where I have acquired all of my git knowledge.
- Atlassian also has great explanations of everything