Introduction to Git and GitHub

Frank Burkholder

Objectives

- Explain what Git is, and what its common commands do:
  - `clone`
  - `add`
  - `commit`
  - `push`
- Explain what GitHub is
- Be able to add a repo from GalvanizeDataScience to your Github, work on it locally, and save changes back to your Github.
- Be able to add another person's Github repo as a remote to your repo.
- Collaborate on modifying a text file on Github with a partner.

What is Git

Git is open-source, distributed version control software that lets you keep track of file changes in a repository or folder

- It runs locally on your laptop (it's also on the servers hosting Github).
- Checkpoints (commits) keep track of what was changed, and by whom and when.
- You can load an earlier checkpoint to reverse changes.
- The distributed aspect of Git allows teams to collaborate on projects.

History (Wikipedia)

- git development began in 2005, after Linux developers stopped using Bitkeeper, a proprietary source control management system. Wanted an open-source SCMS.
- Linus Torvalds and team developed it in less than a month, though it is still actively maintained.
- 'git' means unpleasant person in British slang.

Common git commands

`$ git init` initializes a repository as a Git tracked repository
don't do this much in the DSI - see cloning below

`$ git clone` makes a copy of a Git repository in a new location
you'll do this ~2x/day in the DSI for each new repo (repository/folder/directory)

`$ git add` adds files & folders to a staging area so their histories & changes are tracked
you'll do this many times every day in the DSI

`$ git commit` incorporates the changes in the staging area into a new checkpoint
you'll do this many times every day in the DSI

`$ git push` Updates a Github repo with the changes you've committed locally
you'll do this many times every day in the DSI

`$` signifies that the command is typed in your Terminal.

Typical git DSI workflow

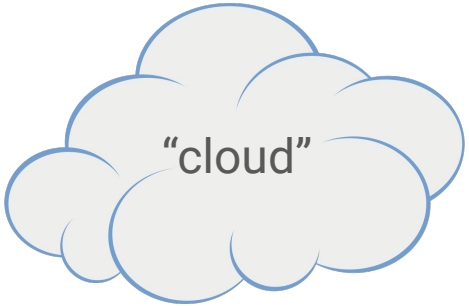
Fork -> **clone** -> modify files -> **add** -> **commit** -> **push**

Github (in your web browser)

integration of git and Github in Terminal

git command in Terminal

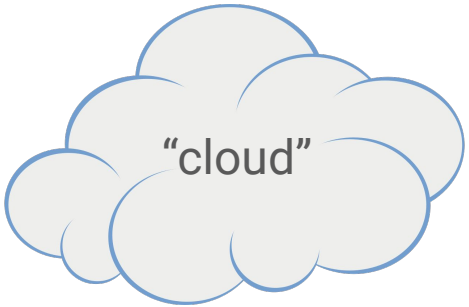
Git and GitHub schematic



local

your laptop

Git and GitHub schematic



GitHub

GDS

(GalvanizeDataScience)

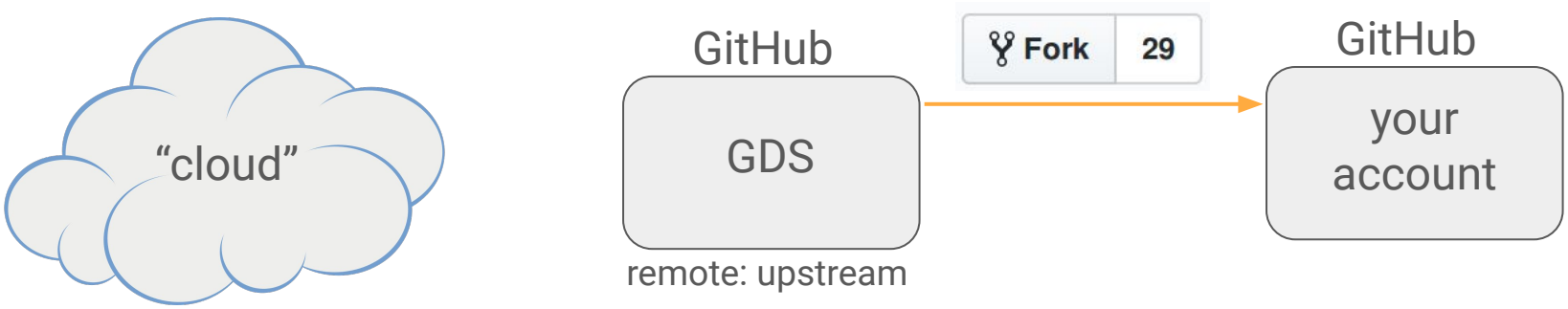
GitHub

your account

local

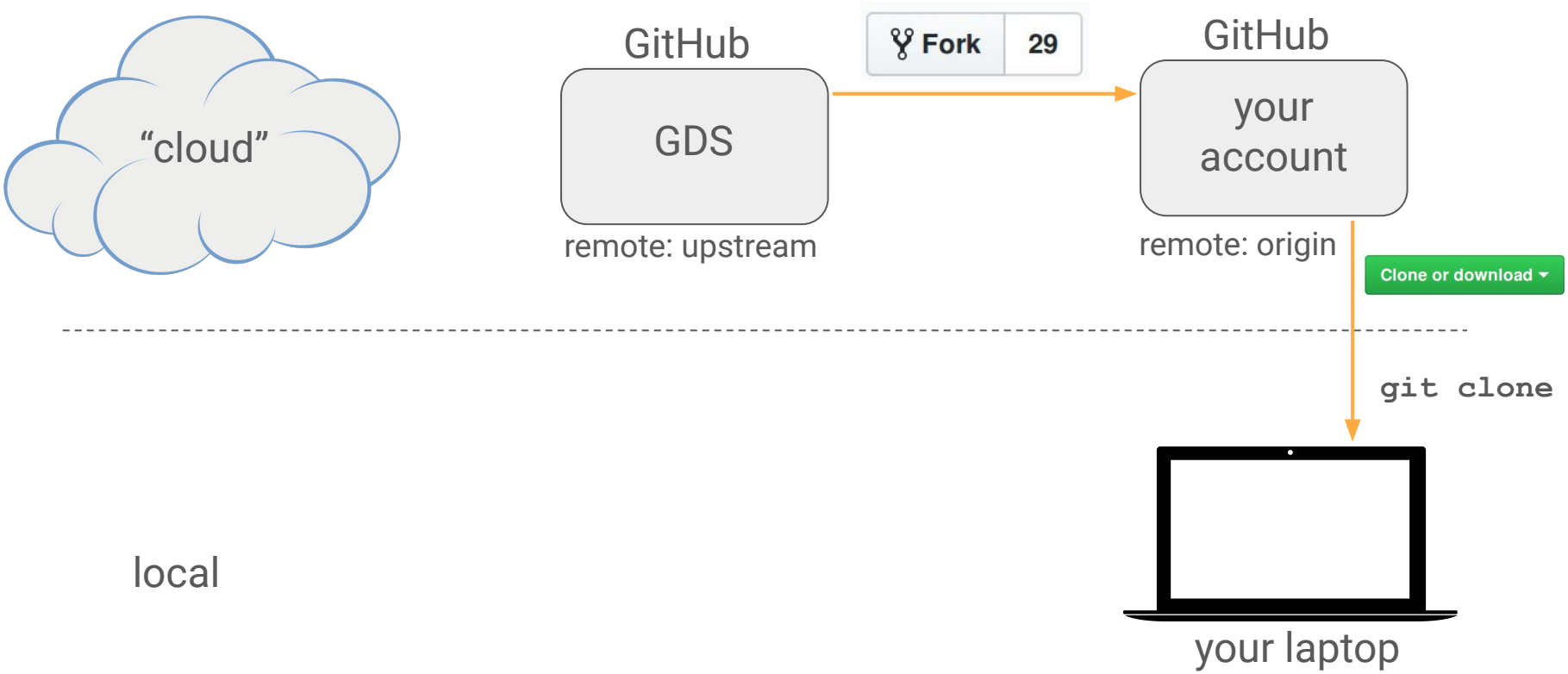
your laptop

Git and GitHub schematic

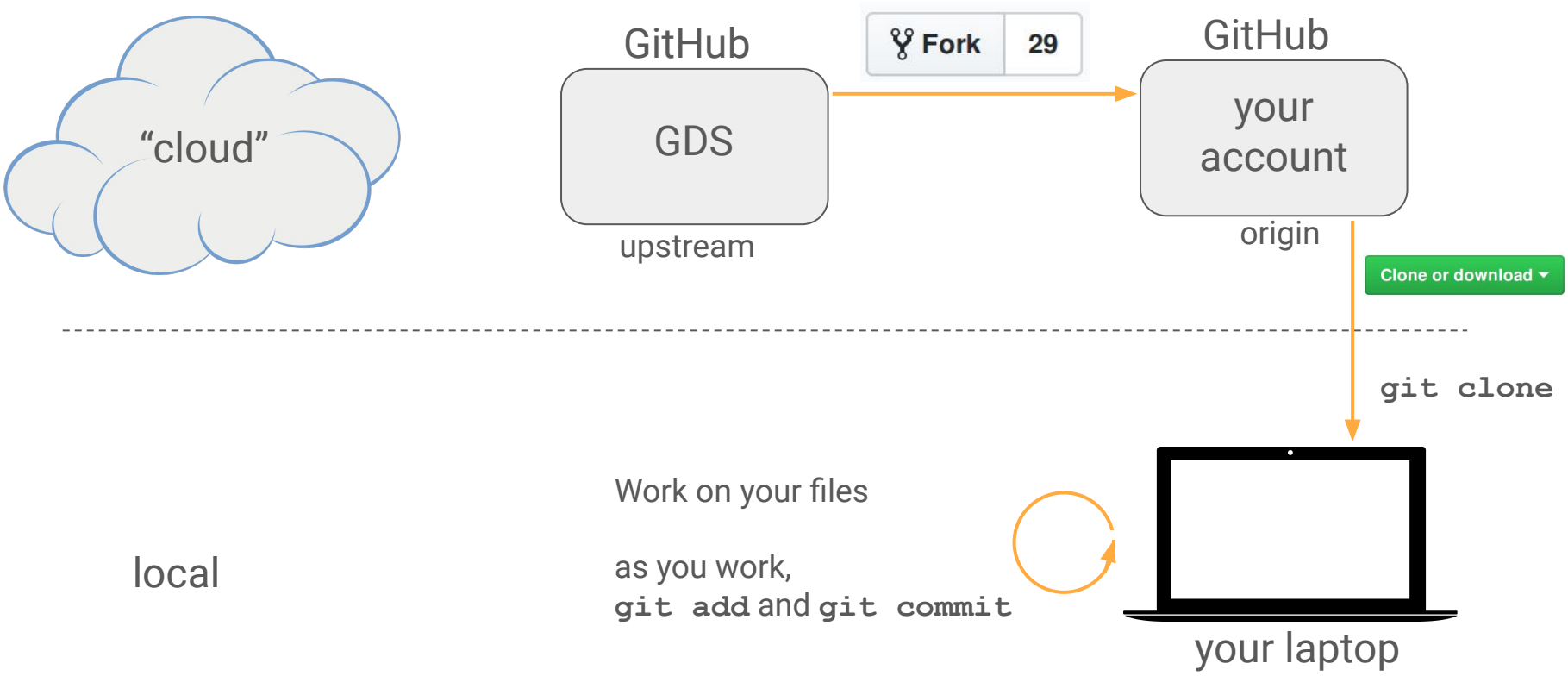


local

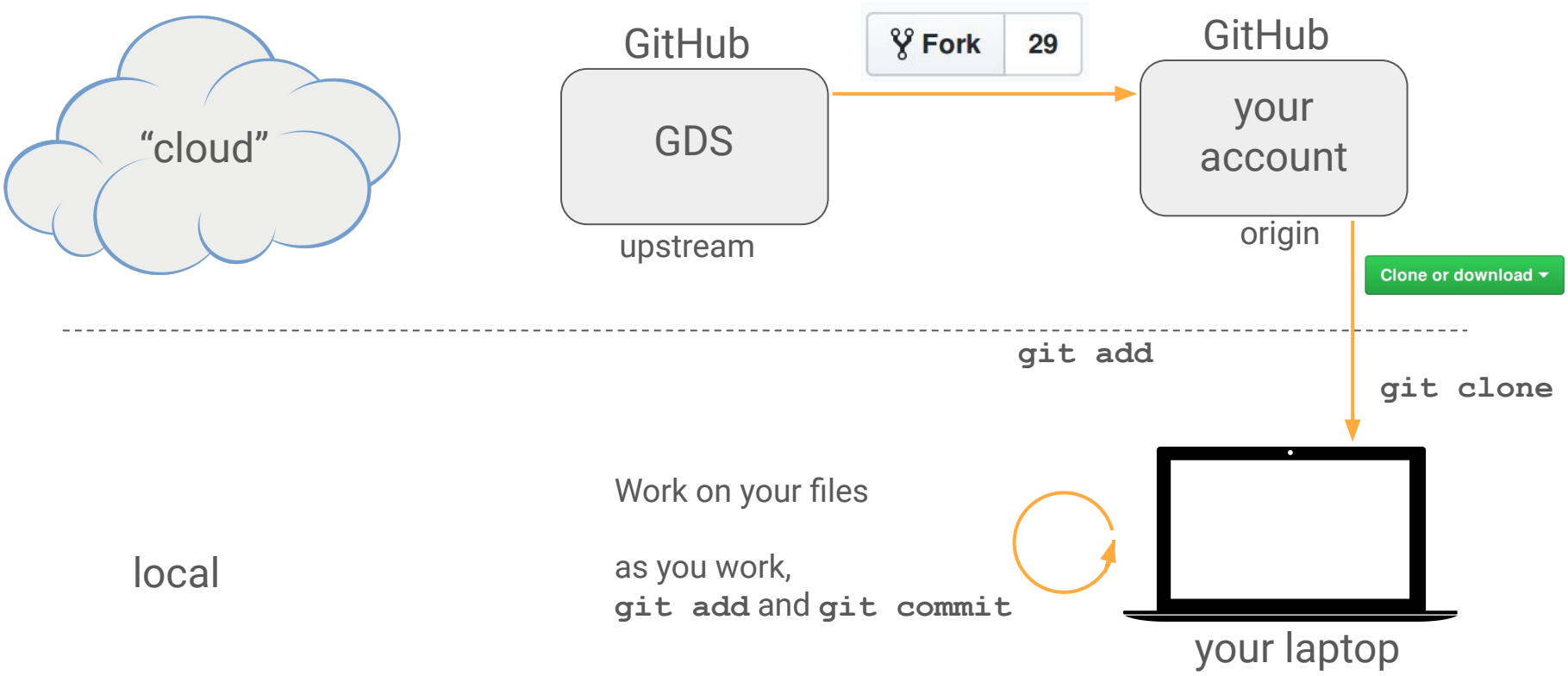
Git and GitHub schematic



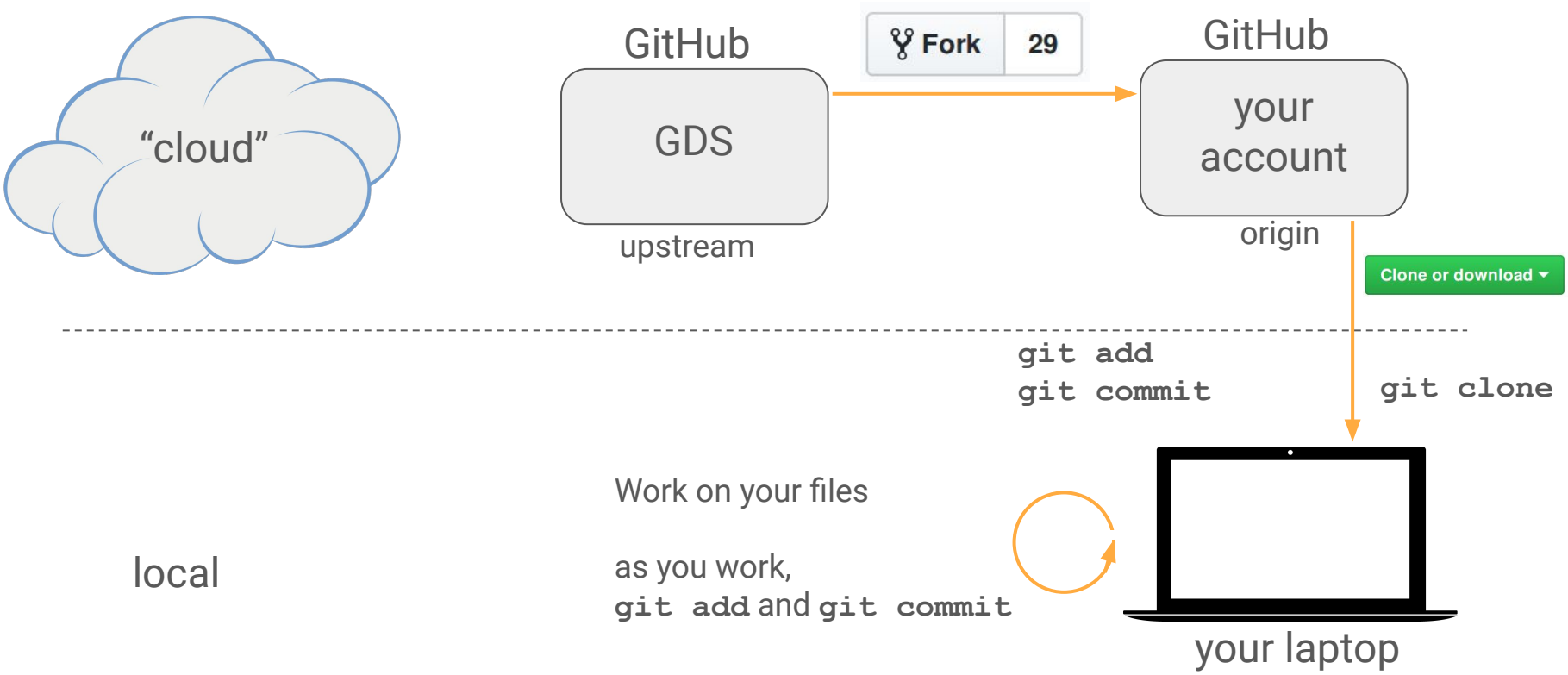
Git and GitHub schematic



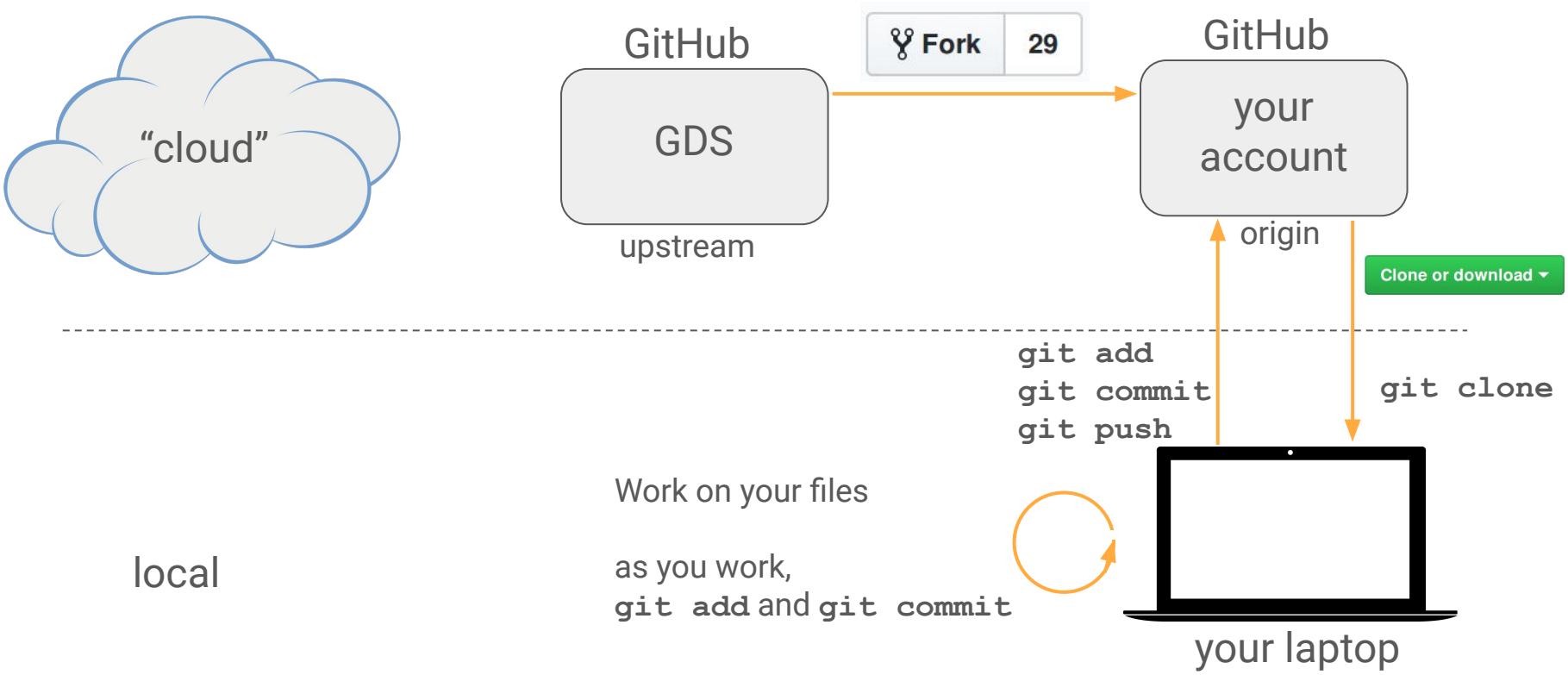
Git and GitHub schematic



Git and GitHub schematic



Git and GitHub schematic



GitHub

GitHub is a web hosting service for Git repositories (repos).

- Github's mascot: the Octocat
- In 2018 became subsidiary of Microsoft
- Git exists independently of GitHub, while the converse is not true.
- GitHub has ~37 million users and 100 million repositories, making it the largest host of source code in the world (Wikipedia, May 2019)
- There are other web hosting services for git repositories (e.g. BitBucket)
- Github repos serve as **remotes**:
    - A remote is a shared Git repository that allows multiple collaborators to work on the same Git project from different locations.

GitHub

GitHub is a web hosting service for Git repositories (repos).

- Galvanize's GalvanizeDataScience (https://github.com/GalvanizeDataScience) is where you'll find DSI repos.
- Public repos are free, and private repos with up to 3 collaborators are also free, but otherwise private repos are monthly charge.
- Any repo that you **fork** from GalvanizeDataScience will remain a private repo (though it will be your repo after your fork it.)
- **Your GitHub account is the public face of your code (it's your portfolio) and is instrumental to finding employment as a Data Scientist.**

GitHub

What you will be doing on GitHub:

`Fork` (button): Creates your own copy (in your GitHub account) of a GitHub repository.

`Clone` (button): Copies the URL of your GitHub repository to the clipboard so that you can paste it into Terminal and `git clone` it with Git. **Clone, don't download.**

Maybe during case studies (depending on workflow you choose):

`pull request` (buttons): Asking permission of the *upstream* repository (where you forked your GH repository, perhaps a classmate) to merge changes that exist in your GH repository.

An example (1 of 3)

Text typed in Terminal

```
(base) frank@peregrine:~/Documents/g/dsi$ git clone https://github.com/Frank-W-B/git-intro.git
Cloning into 'git-intro'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 42 (delta 8), reused 4 (delta 1), pack-reused 24
Unpacking objects: 100% (42/42), done.
(base) frank@peregrine:~/Documents/g/dsi$ cd git-intro/
(base) ✔  ~/Documents/g/dsi/git-intro [master|✔ ]
12:58 $ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
(base) ✔  ~/Documents/g/dsi/git-intro [master|✔ ]
12:58 $ ls
assignment.md  data  haiku.txt  README.md  reference  src
```

A branch in Github is a pointer to a commit. We'll talk about branches more during your first case study.

An example (2 of 3)

Text typed in Terminal

```
(base) ✔  ~/Documents/g/dsi/git-intro [master|✔]
12:59 $ code haiku.txt
```

Text editor VSCode (write haiku then save it):

haiku.txt ●

home > frank > Documents > g > dsi > git-intro > haiku.txt

```
1  Information to
2  Knowledge - the charge of the mind
3  undetermined end
```

```
13:14 $ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   haiku.txt

no changes added to commit (use "git add" and/or "git commit -a")
(base) ✔  ~/Documents/g/dsi/git-intro [master|✚1]
```

An example (3 of 3)

Text typed in Terminal

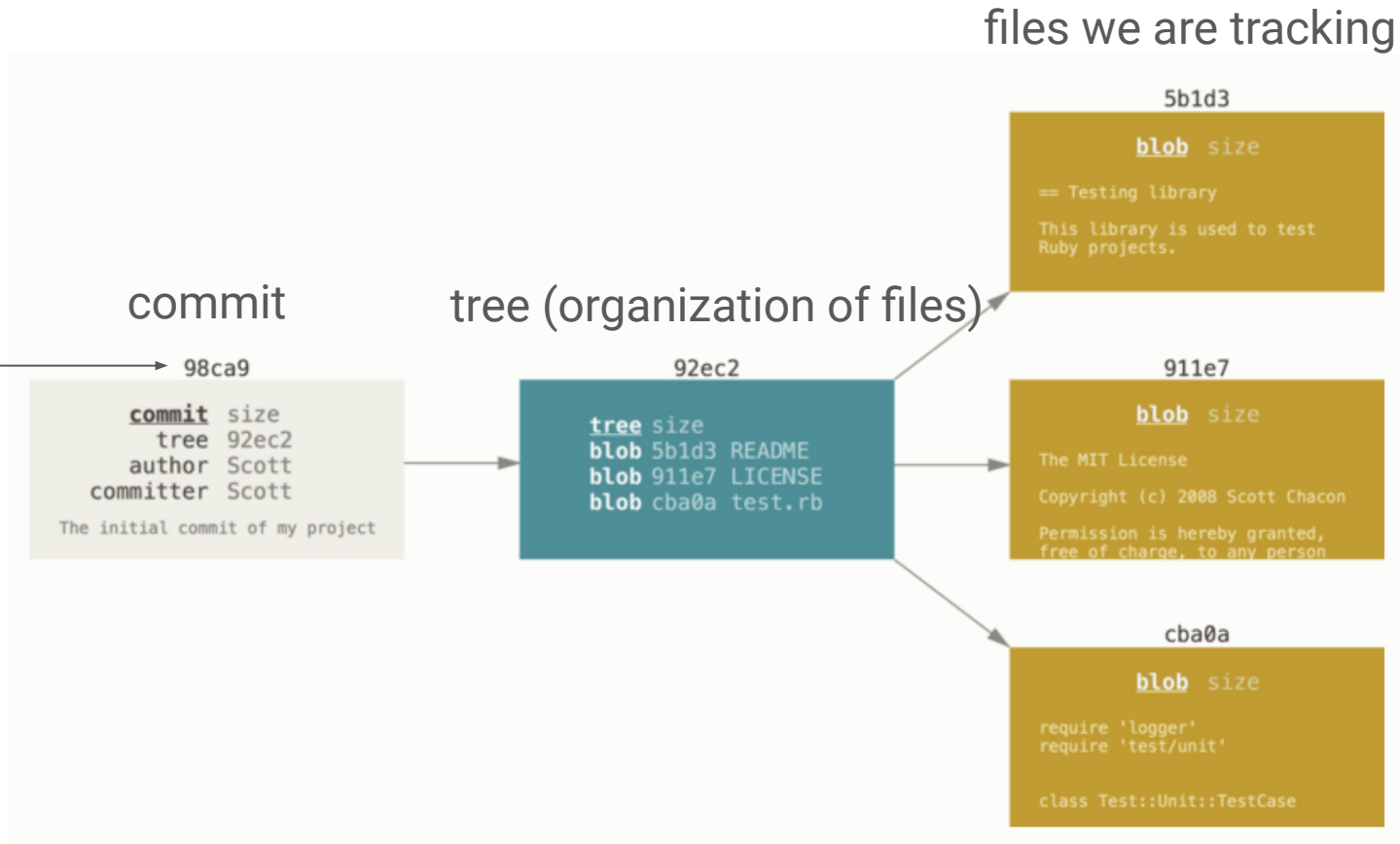
```
(base) ✔ ~/Documents/g/dsi/git-intro [master|✚1]
13:15 $ git add haiku.txt
(base) ✔ ~/Documents/g/dsi/git-intro [master|●1]
13:18 $ git commit -m "First haiku"
[master 83e6d2e] First haiku
 1 file changed, 3 insertions(+), 1 deletion(-)
(base) ✔ ~/Documents/g/dsi/git-intro [master ↑·1|✔]
13:18 $ git push origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 325 bytes | 325.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Frank-W-B/git-intro.git
   da08f65..83e6d2e  master -> master
(base) ✔ ~/Documents/g/dsi/git-intro [master|✔]
```

Breakout - Part I of the assignment

In your web browser, navigate to
https://github.com/GalvanizeDataScience/git-intro
and do Part I of the assignment. Wait do parts II, III, and IV!

(Don’t forget to Fork and Clone the repo, first!)

What git is doing



source

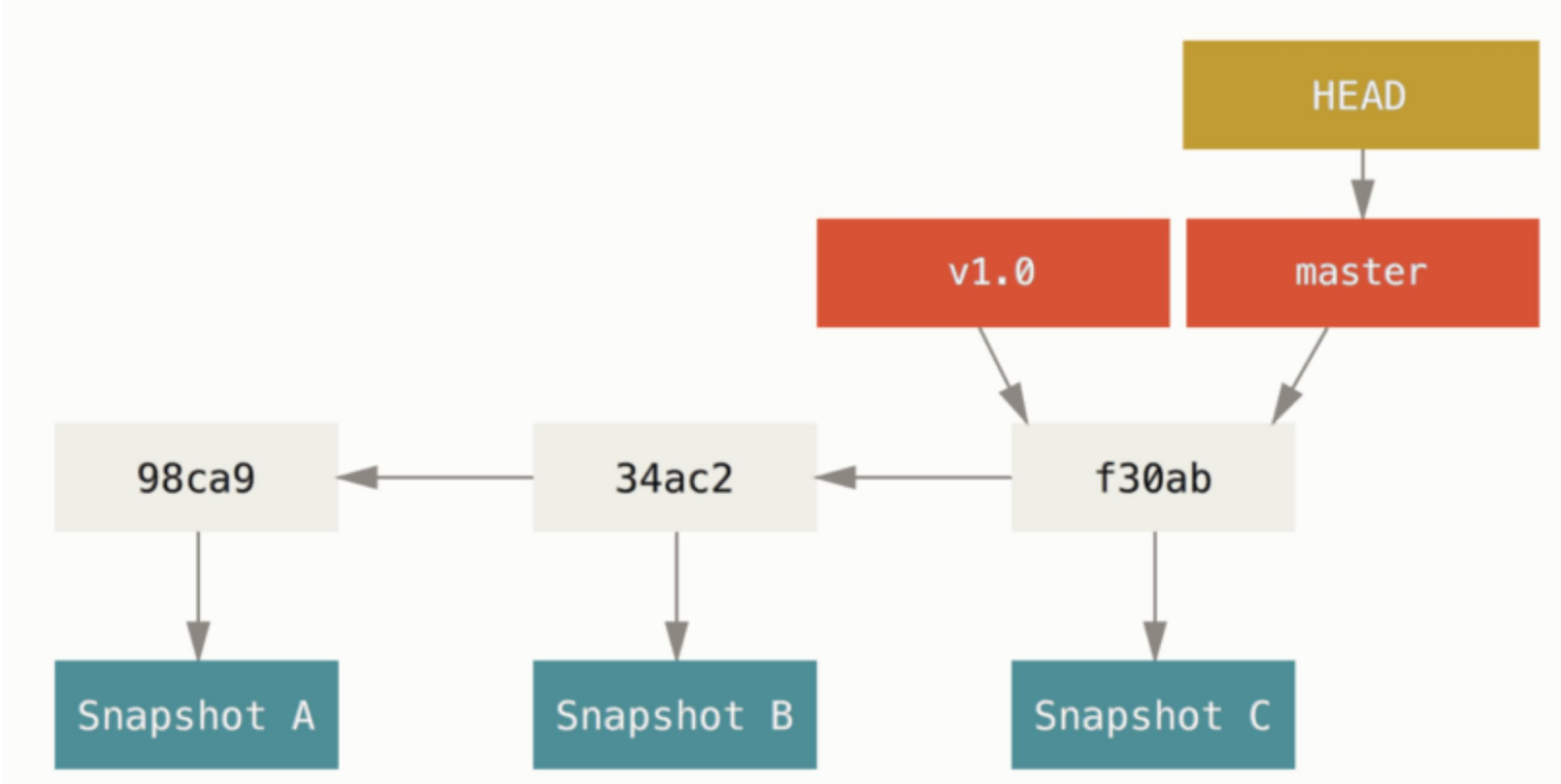
How to see your commits

```
$ git log
$ git log -5                (show last 5)
$ git log --oneline -5      (more concise with relevant hash)
```

Breakout - Part II of the assignment

Please work on Part II. (Wait for parts III and IV)
See your commits, make a new commit, and then revert a commit.

Git branches

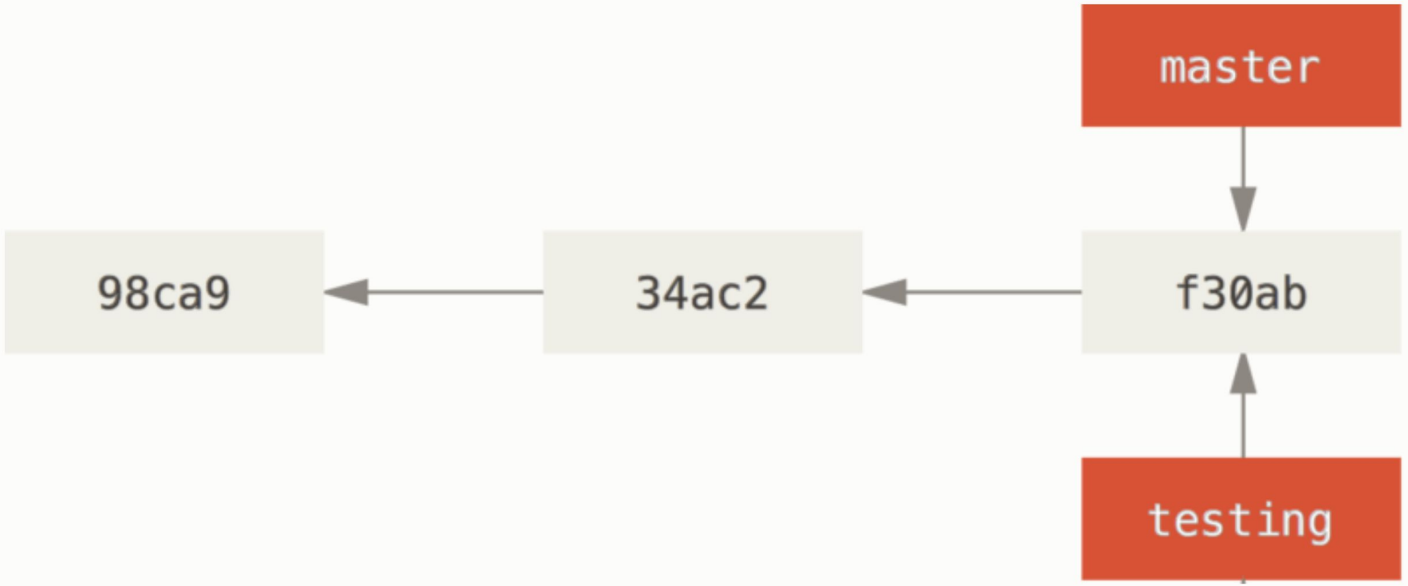


source

Git branches

```
$ git branch testing
```

This creates a new pointer to the same commit you're currently on.

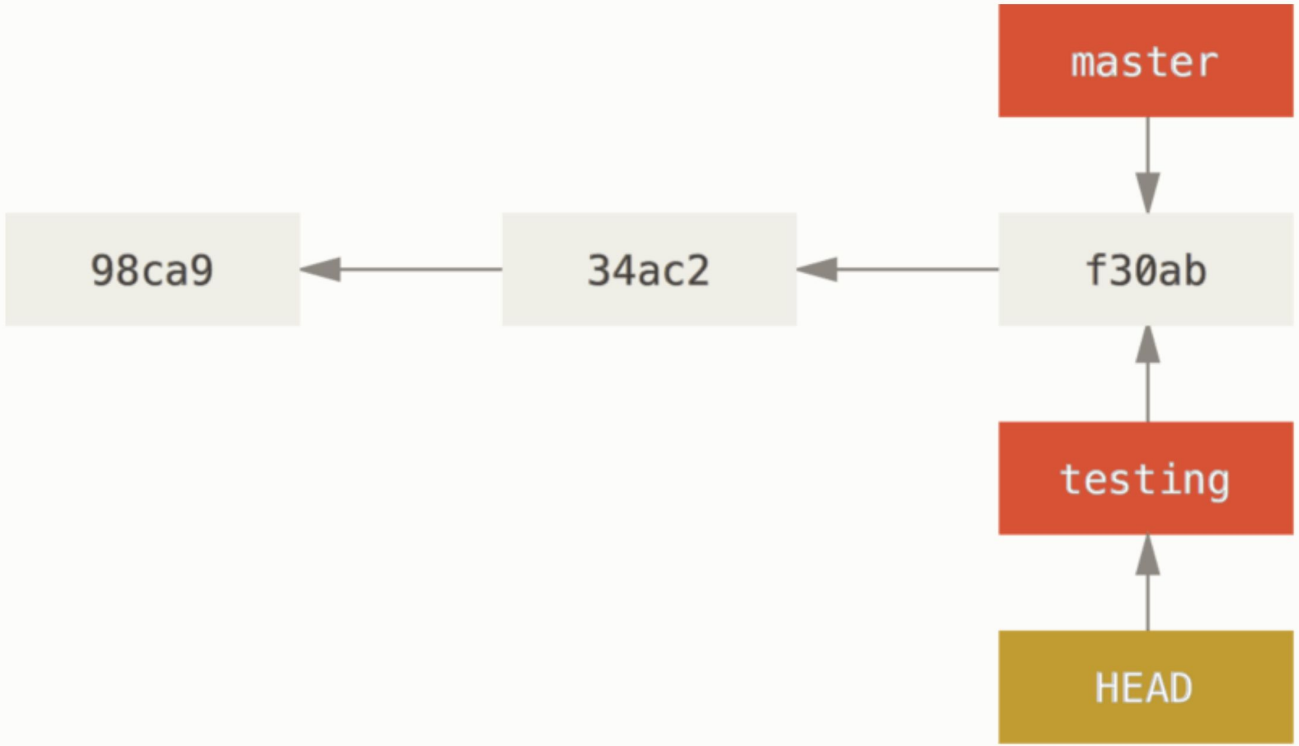


source

Git branches

```
$ git checkout testing
```

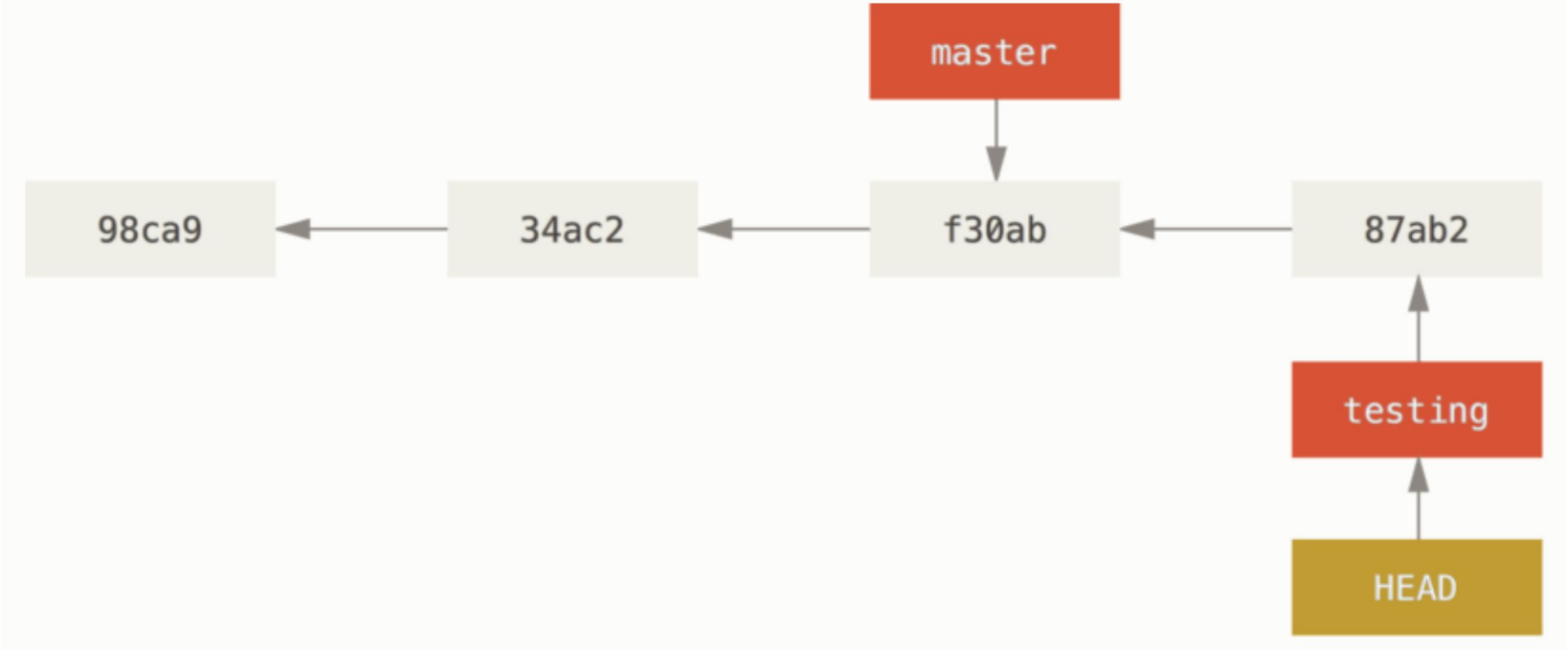
This moves `HEAD` to point to the `testing` branch.



source

Git branches

```
$ vim test.rb
$ git commit -a -m 'made a change'
```

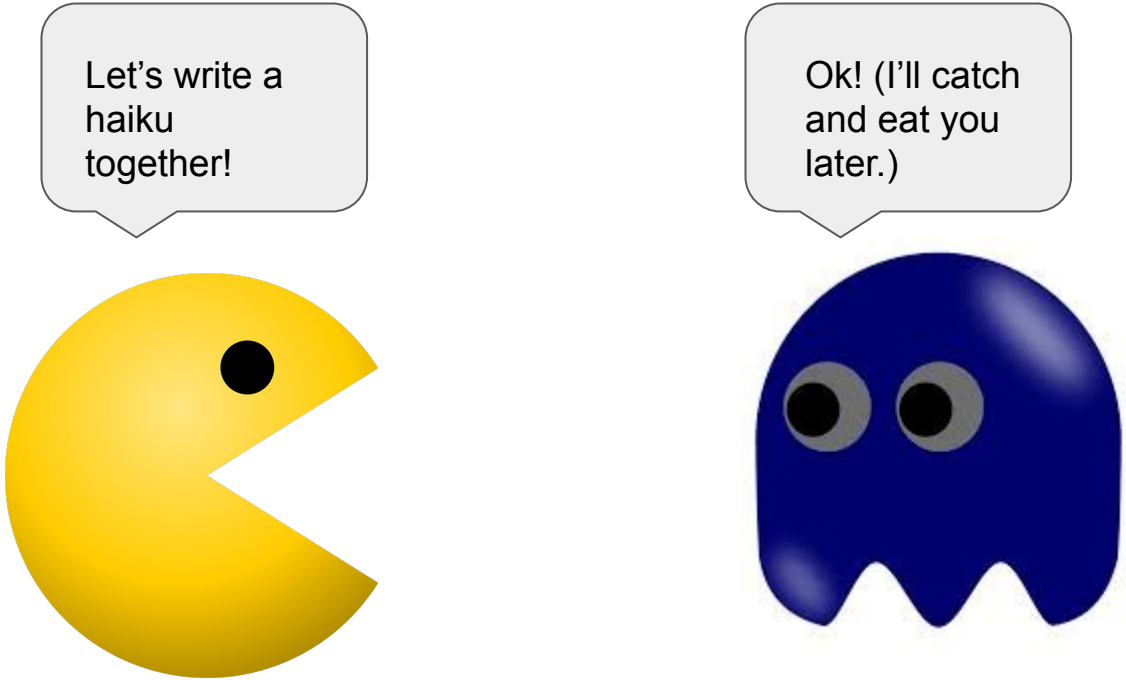


source

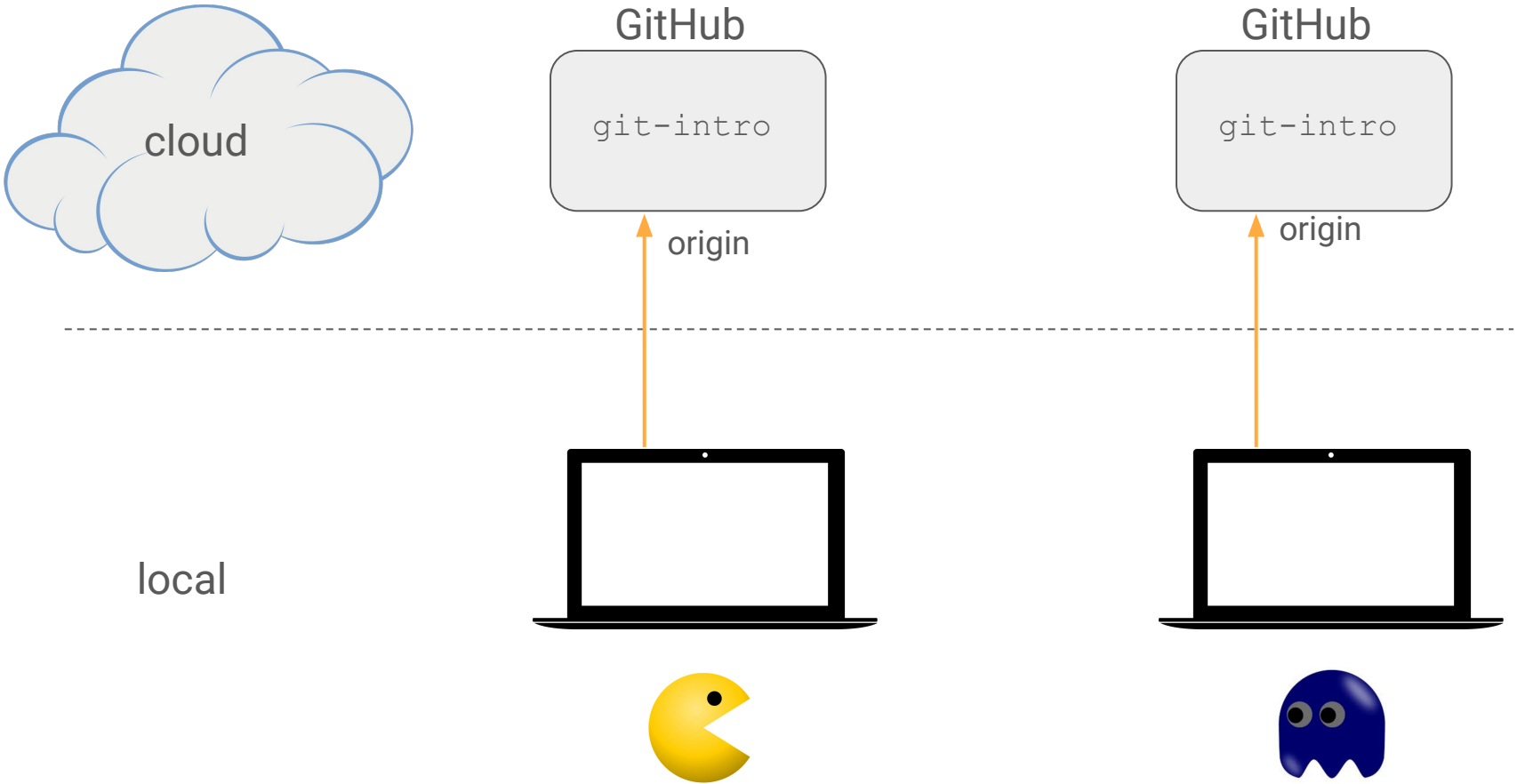
Breakout - Part III of the assignment

Please work on Part III. (Wait for part IV)
Make a branch, work on it, and merge it back in to master.

Well if Git and Github are for collaborating….

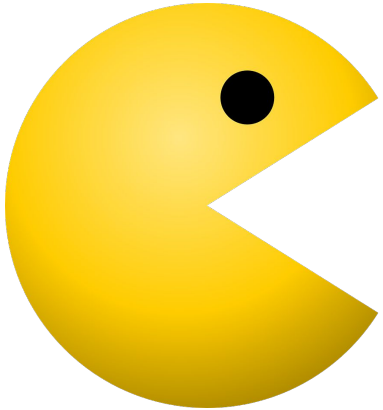


Git and GitHub schematic (before collaborating)



Simple collaborating….

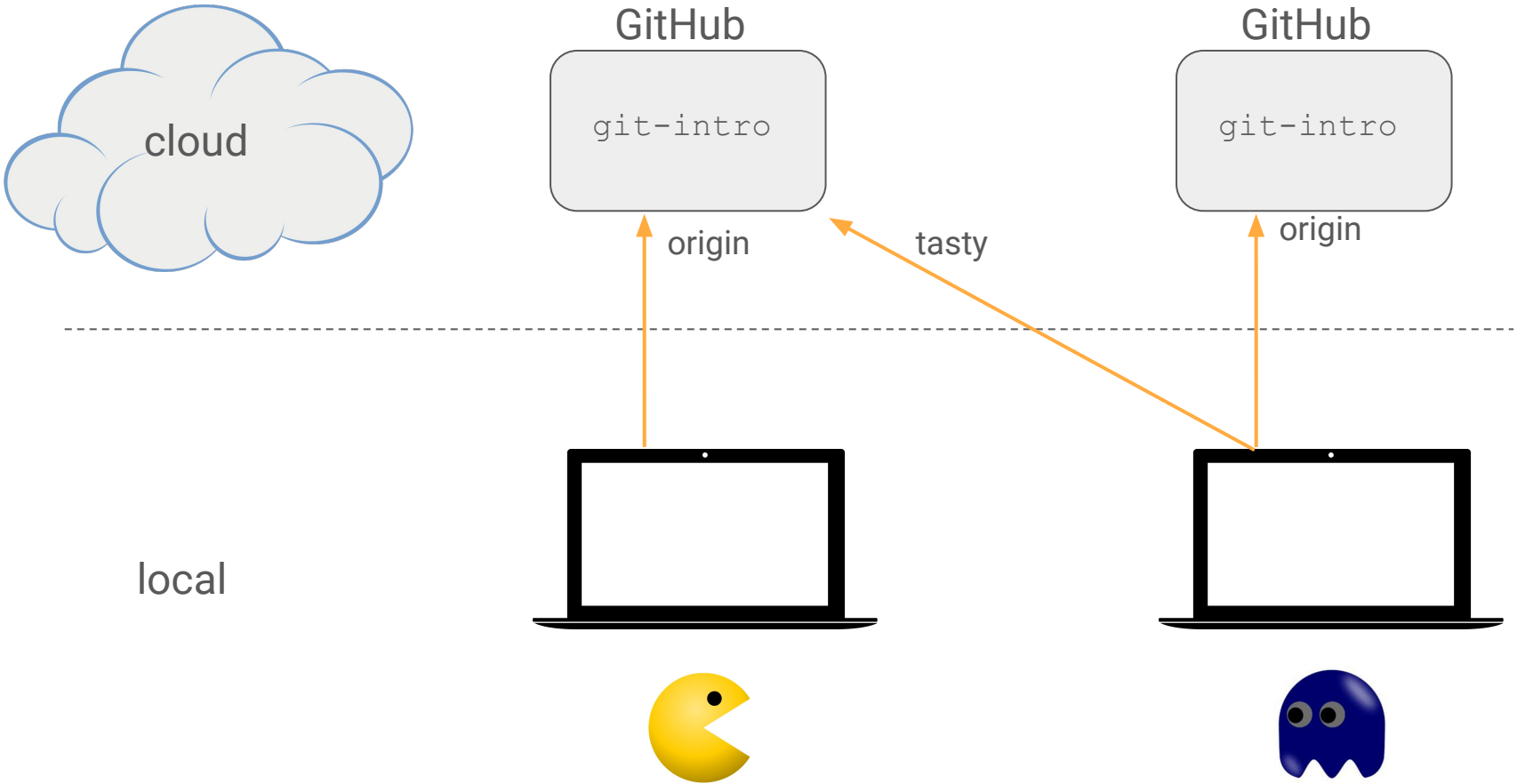
Let's work out of my repo. Add my repo as a remote.



Ok! I'll call your remote "tasty"

In your repo on Github, you'll need to add my Github handle as a collaborator (look in Settings)

Git and GitHub schematic (to collaborate)



How to add a remote to your repo

Frank-W-B is Inky (replace with your GitHub) that wants to add Pac-Man's repo as a remote.

```
(base) ✔  ~/Documents/g/dsi/git-intro [master|✔]
11:39 $ git remote -v
origin  https://github.com/Frank-W-B/git-intro.git (fetch)
origin  https://github.com/Frank-W-B/git-intro.git (push)
(base) ✔  ~/Documents/g/dsi/git-intro [master|✔]
11:48 $ git remote add tasty https://github.com/Pac-Man/git-intro.git
```

Name you want to give remote

URL of remote on Github

```
11:49 $ git remote -v
origin  https://github.com/Frank-W-B/git-intro.git (fetch)
origin  https://github.com/Frank-W-B/git-intro.git (push)
tasty   https://github.com/Pac-Man/git-intro.git (fetch)
tasty   https://github.com/Pac-Man/git-intro.git (push)
```

Now you have two remotes: **origin** (yours) and **tasty** (Pac-Man's)

Working with the new remote

```
(base) ✔  ~/Documents/g/dsi/git-intro [master|✔ ]
11:49 $ git pull tasty master
```

... should pull down whatever additions Pac-Man added ...

... add a line to the `haiku.txt` file using your text editor ...

... `$ git add` ...

... `$ git commit` ...

... now give it back to Pac-Man to add another line to the `haiku.txt`:

```
(base) ✔  ~/Documents/g/dsi/git-intro [master|✔ ]
11:49 $ git push tasty master
```

Working with the new remote

```
(base) ✔  ~/Documents/g/dsi/git-intro [master|✔ ]
11:49 $ git pull tasty master
```

... should pull down whatever additions Pac-Man added ...

... add a line to the `haiku.txt` file using your text editor ...

... `$ git add` ...

... `$ git commit` ...

... now give it back to Pac-Man to add another line to the `haiku.txt`:

```
(base) ✔  ~/Documents/g/dsi/git-intro [master|✔ ]
11:49 $ git push tasty master
```

How does Inky save the new haiku to his Github?

Working with the new remote

```
(base) ✔  ~/Documents/g/dsi/git-intro [master|✔ ]
11:49 $ git pull tasty master
```

... should pull down whatever additions Pac-Man added ...

... add a line to the `haiku.txt` file using your text editor ...

... `$ git add` ...

... `$ git commit` ...

... now give it back to Pac-Man to add another line to the `haiku.txt`:

```
(base) ✔  ~/Documents/g/dsi/git-intro [master|✔ ]
11:49 $ git push tasty master
```

How does Inky save the new haiku to his Github?

```
$ git push origin master
```

Merge conflicts

Merge conflicts occur when two different users make competing changes to a file (like both changing the same line in different ways).

It's not hard to resolve them. Git will tell you which file the conflict occurs in, and give you visual indication in the text file where the problem is.

To solve a Merge conflict:
Open the file with a text editor, select which text you want to keep, save the file, then add and commit it. Voila! Merge conflict over.

Here's more detail:

https://help.github.com/articles/resolving-a-merge-conflict-using-the-command-line/

Branches in git will help avoid situations like this. We'll talk about them later.

Objectives

- Explain what Git is, and what its common commands do:
  - `clone`
  - `add`
  - `commit`
  - `push`
- Explain what GitHub is
- Be able to add a repo from GalvanizeDataScience to your Github, work on it locally, and save changes back to your Github.
- Be able to add another person's Github repo as a remote to your repo.
- Collaborate on modifying a text file on Github with a partner.

Assignment - Pair programming on git

Do Part IV of the assignment