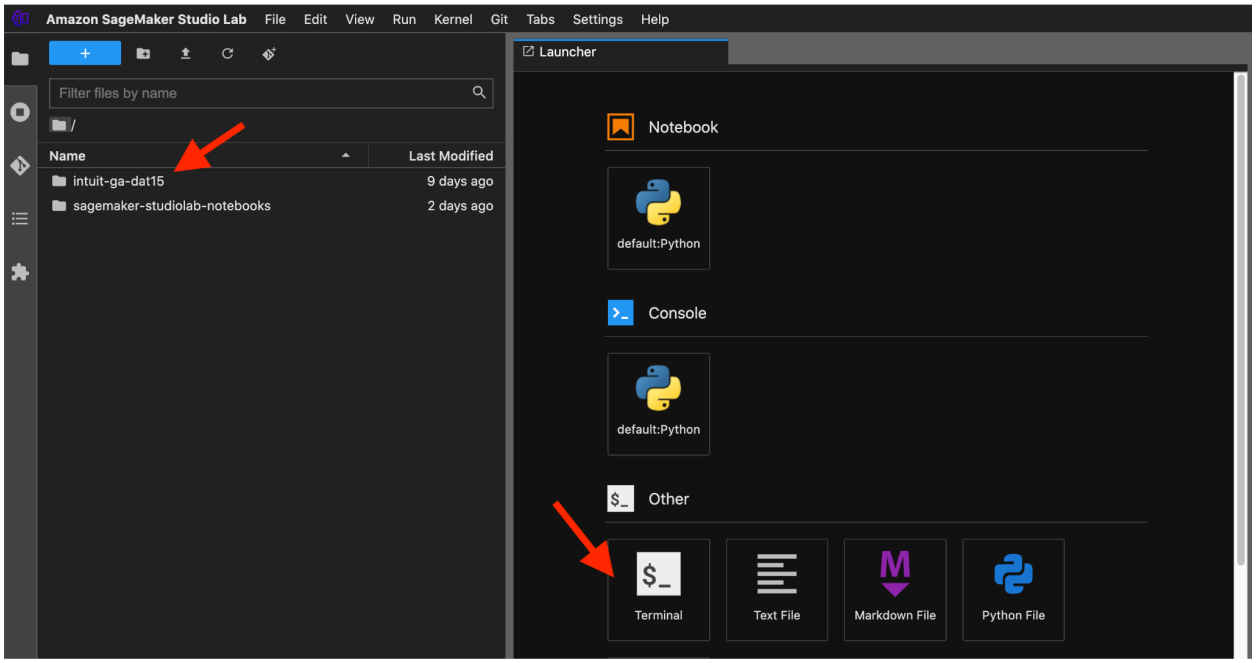
GA Data Science
Intuit Cohort 15
Lesson 3: Project Hour
March 2, 2022

1. Clone the lecture repository to Sagemaker

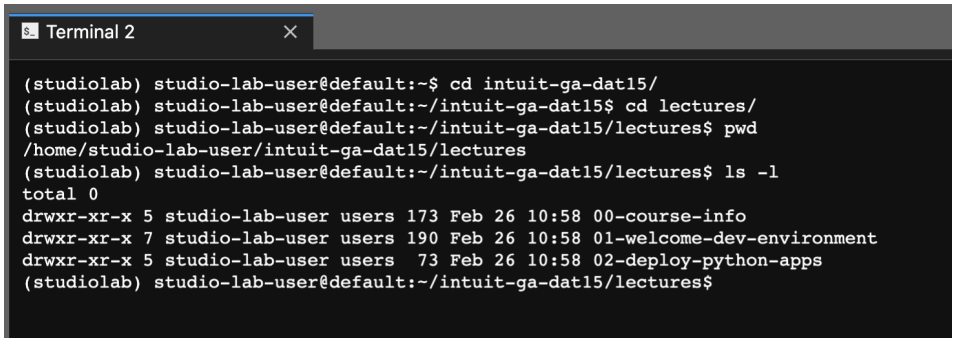
Navigate to Sagemaker Studio. You should see the following. Remember that all of our work will be saved inside the folder “intuit-ga-dat15” that we created last time. If you wanted to, you could use the buttons on the far left side of the screen to do things like create new folders and clone git repositories. But that would be too easy (and a crutch). Instead, we’re going to do it the hard way, using terminal.



Open a new terminal session, and try the following commands.

```
cd intuit-ga-dat15
cd lectures
pwd
ls -l
```

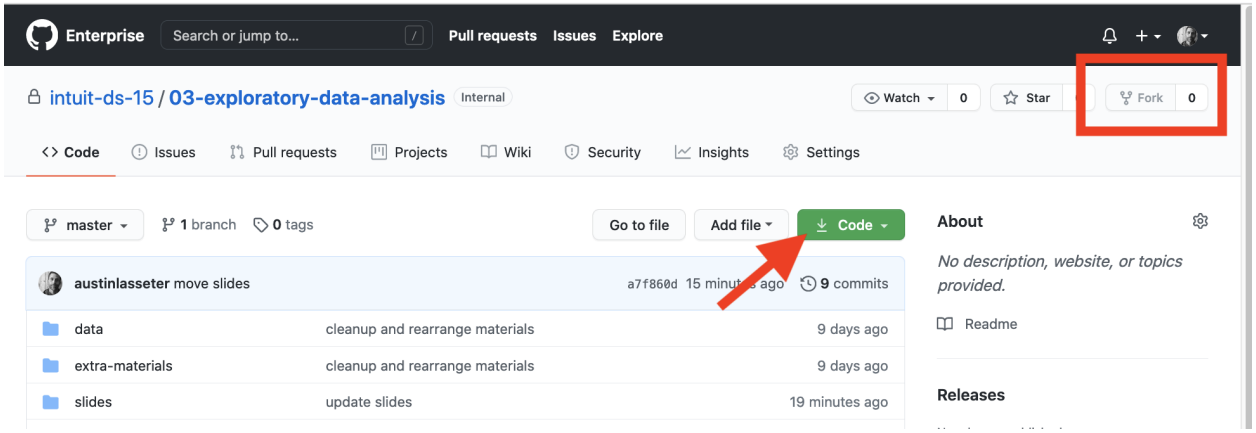
You should see three repositories that we cloned the previous lessons: repos 00, 01, and 02. If the repos aren’t there, you probably missed this last class. But don’t worry! You can just use the same steps that we’re going to use for repo 03, and go back and get the previous repos as well.



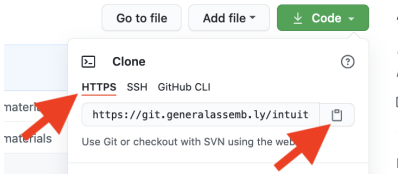
Now navigate to our Data Science org in Github Enterprise.
https://git.generalassemb.ly/intuit-ds-15

Find today's repository.
https://git.generalassemb.ly/intuit-ds-15/03-exploratory-data-analysis

Notice that the option to "Fork" has been disabled - this is by design. We don't want you to fork this repo (because only the instructor should be modifying the content here).
Click on the green "Code" button.



A dialogue window will open. Choose "https" – do *not* choose "SSH" or "Github CLI". Then copy the resulting URL using the clipboard icon to the right.



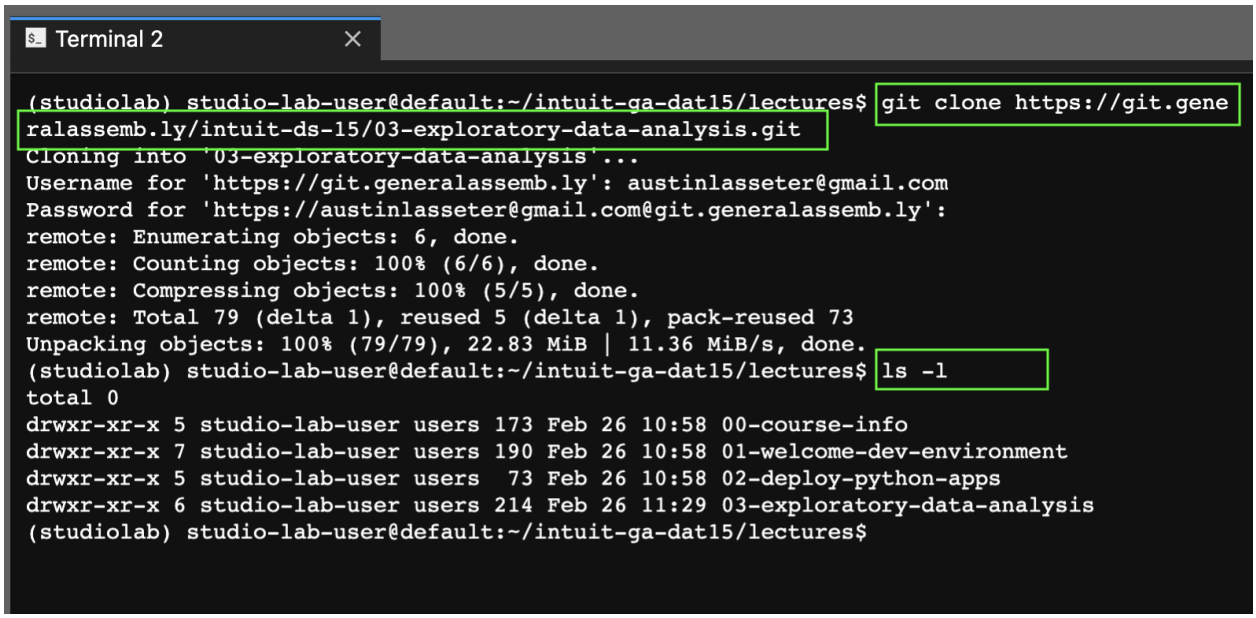
Now that you've copied the URL, go back to the Sagemaker console and your terminal there. Before we do the next steps, use the command **pwd** (print working directory) to make sure that you're inside the right directory. When you clone a repo, it's very important to keep track of the folder structure of your directories - you never want to accidentally clone a git repo inside of another repo! You should be here:

`/home/studio-lab-user/intuit-ga-dat15/lectures`

Use the following command to clone the repo to Sagemaker. After typing the words “git clone” you can just use ctrl+v to paste the URL that you copied in the previous step.
`git clone https://git.generalassemb.ly/intuit-ds-15/03-exploratory-data-analysis.git`

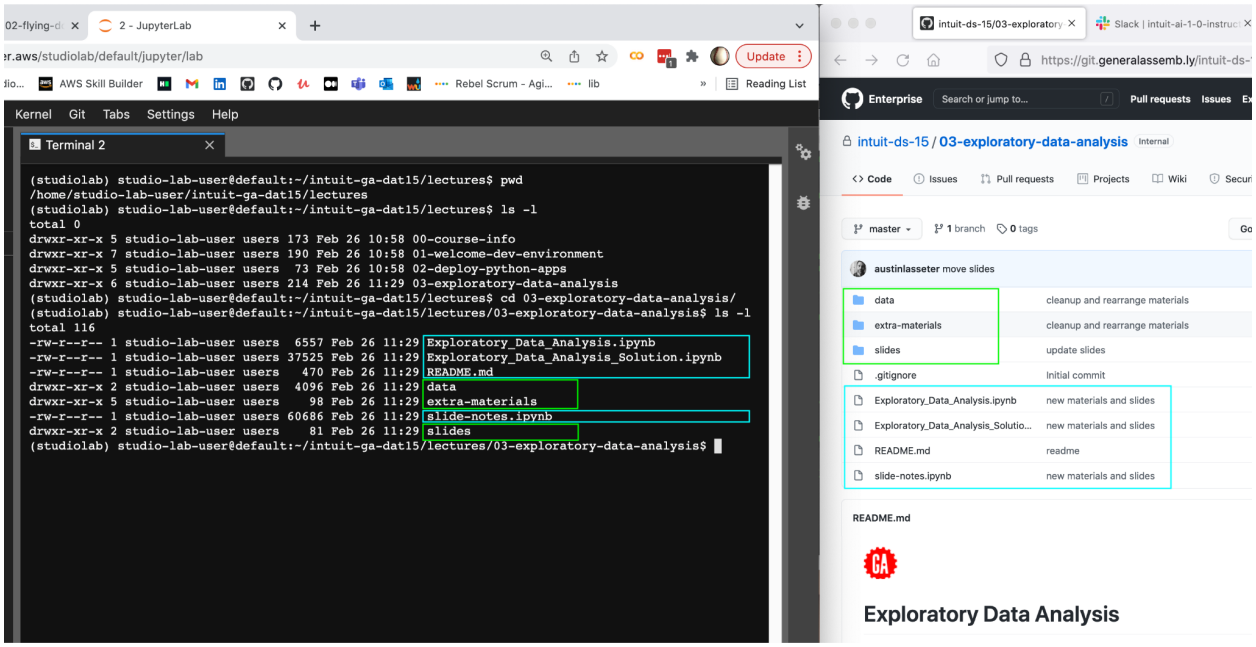
Git will ask you for your username and your password. Note that github enterprise still uses a password, not a personal access token (but github public is the other way around). Also, when you try to type your password (or paste it using ctrl+V) the cursor in terminal doesn’t move at all - this is a security feature to protect your password, but it can be a little confusing the first time it happens. Just press enter after typing the password and all should be well.

You can then use the “list” command ls -l to show the files in your directory – the new 03 repo should now be present.



You can also navigate inside the 03 repo. Type “cd” for “change directory” and begin typing “03” then use the tab button to autocomplete. Check out the files there using the command “ls -l” and you’ll see that they are exact duplicates of the files listed in the remote origin.

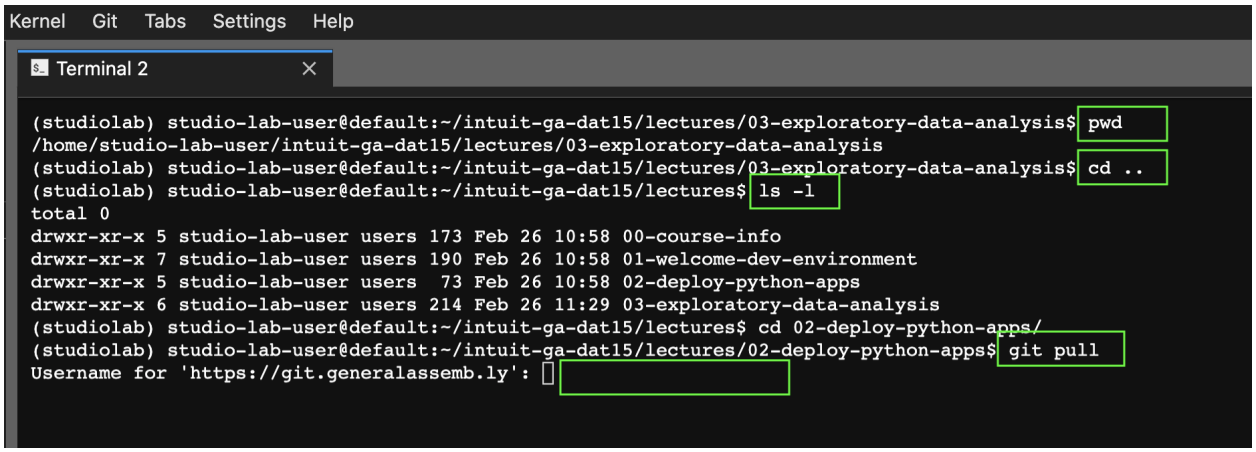
As you can see in the screenshot below, the same files and folders are listed - but not in the same order. Terminal lists them alphabetically, while github lists folders at the top, files at the bottom. There’s also a hidden file (.gitignore) – you can show this in terminal with the command “ls -a” if you want. The .gitignore is sort of a trash bin that prevents garbage files on your machine (like .pyc files or .DS_Store) from ending up in github.



Now your 03 repo is ready to go!

But we also want to go into the repos from last week – 00, 01, and 02 – and update the files there. This is because the instructor (yours truly) has made some changes to the remote origin and you want to update your clone with the latest version. We'll do this for one repo (02) and you can apply the same changes to the others.

To get out of the 03 directory, use the "cd" command (change directory) and two dots (which means go up one directory) like this: `cd ..`
Then navigate into the 02 repository using "cd" and begin typing "02" then use the tab button to autocomplete. Once inside, use the command "git pull" to obtain the latest version from the remote origin. It will once again ask for username and password.



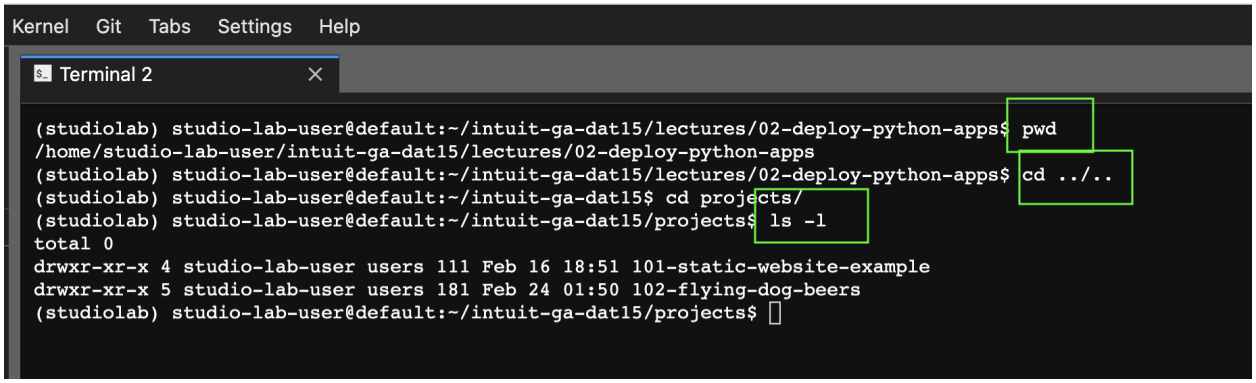
Do the same for the other folders, 00 and 01. This same routine (git clone, git pull) should become a habit for you at the beginning of every lesson in our course.

2. Clone the project repository to Sagemaker

Our course is structured so that we have lectures (which are saved in Github Enterprise) and projects (which are saved in Github Public). We do this because the lecture content is static (only the instructor will change it) but project content is dynamic (we want you to change and update the files as much as you want). This structure mirrors the usual way that most companies maintain their codebase for private and public use.

Some of the steps will be the same here, but some will be different. Get into the habit of following these same steps at the beginning of every lesson.

In Sagemaker, navigate up to the project folder and see what's there. If you successfully cloned the repos last week, you should see folders 101 and 102. If not, that's okay! You can just follow the steps we're about to use for the new repo.



Now navigate to our projects org in Github Public. https://github.com/plotly-dash-apps

This is a good time to point out that throughout this course, we'll be using an open source python library called "plotly dash" – it's a popular tool for building data science applications, and a replacement for the older python app library known as "flask". Read all about it here: https://dash.plotly.com/introduction

I've tried to include a lot of repos inside this org - these are designed to be simple introductory apps for students to play with and learn. The basic structure of the org includes simple and intermediate examples (nothing in this org should be considered "advanced").

- 100 series: simple, static dash applications
- 200 series: simple interactive apps that use a "callback" function
- 300 series: simple apps that incorporate the "pandas" library for interacting with datasets

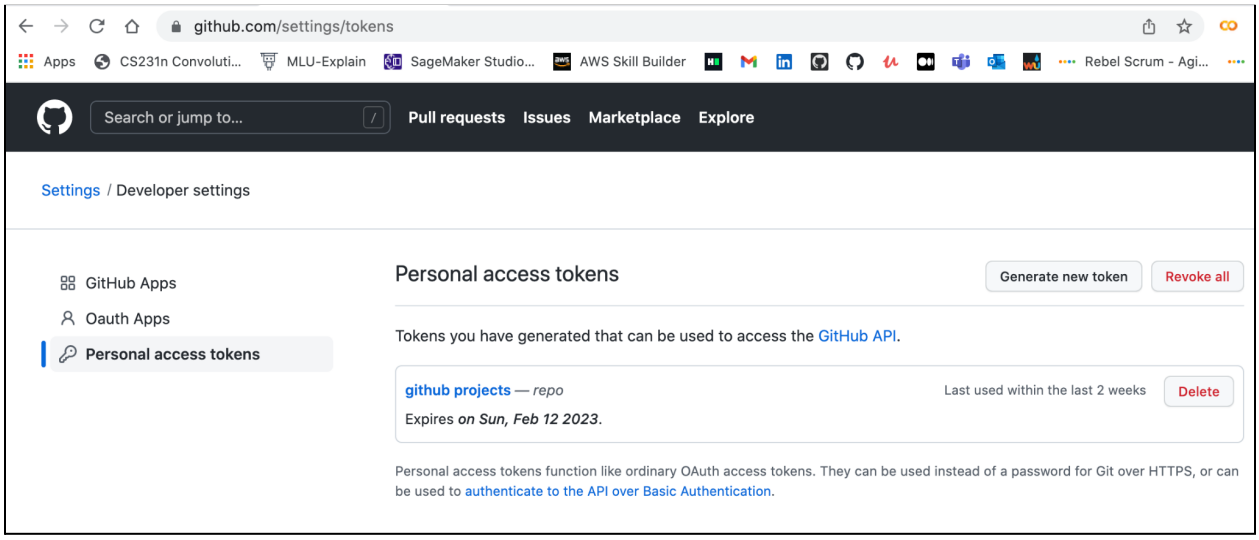
- 400 series: intermediate callback functions without datasets
- 500 series: intermediate apps with datasets and simple machine learning
- 600 series: apps with intermediate machine learning

For today's project, we're going to focus on the following repo:
https://github.com/plotly-dash-apps/203-radio-callbacks

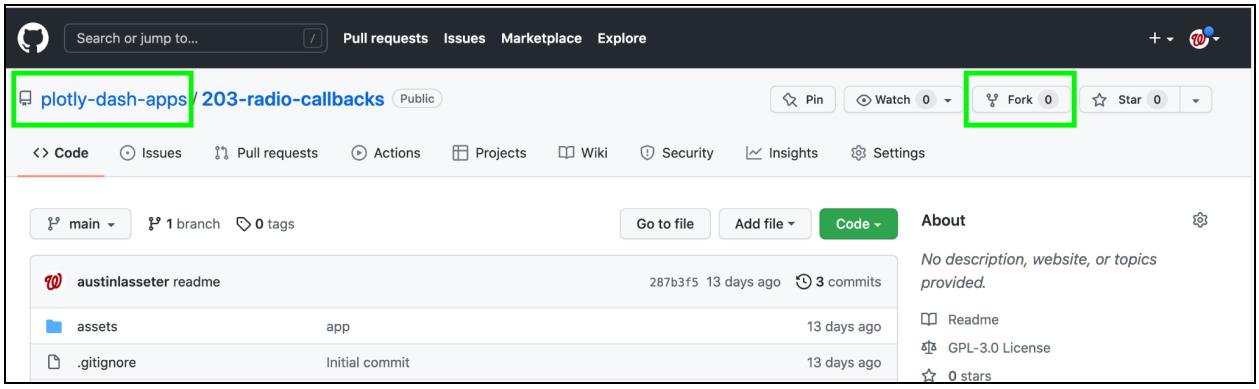
This is what we're going to build in today's project. You can view the finished app here:
https://dash-radio-callback.herokuapp.com/

For now, let's focus on cloning the repo. The first thing you should do is check out your github settings and make sure you have a personal access token - we created this in last week's lesson. If you need a review, you can see those instructions here:
https://git.generalassemb.ly/intuit-ds-15/01-welcome-dev-environment/blob/master/Setting%20up%20AWS%20Sagemaker%20Studio%20Lab.pdf

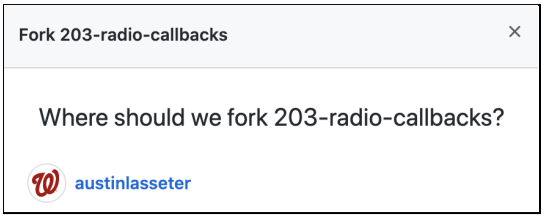
Your can verify that you have a personal access token under github settings/developer settings, but you can't actually see the token itself - you had to copy it to a secure location when you first created it. If you missed that last week, no worries - you can just create a new token now, using the "generate new token" option on the right-hand side.



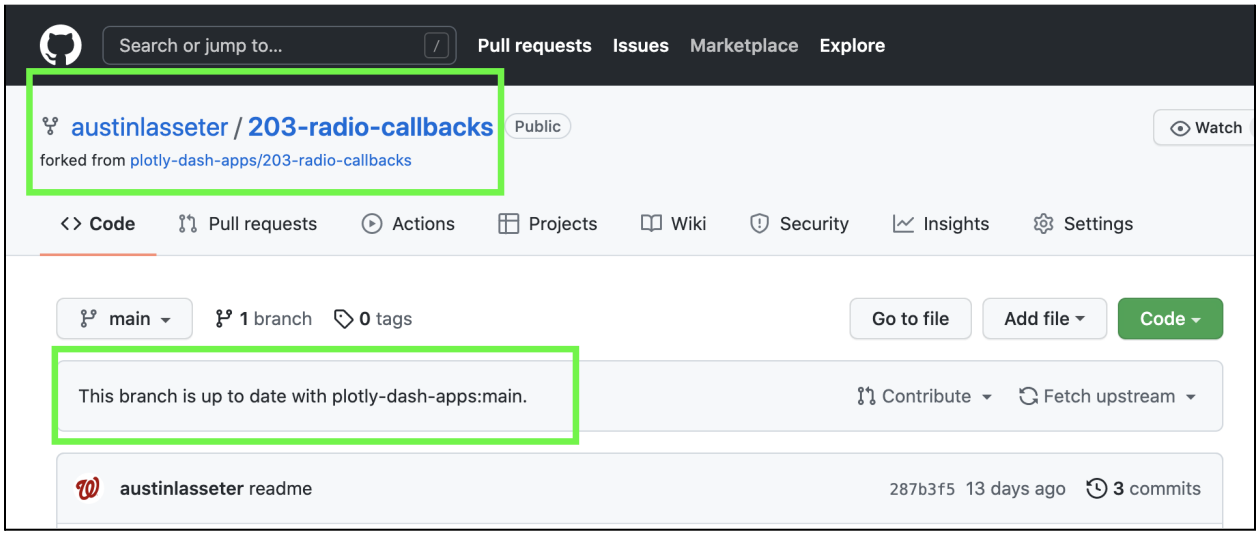
Okay, back to our repo https://github.com/plotly-dash-apps/203-radio-callbacks. Now let's "fork" the repo. We didn't fork the lecture repo in Github Enterprise because you will never modify the lecture materials - only the instructor does that. But you can and should modify the contents of the project folder! Forking is github's way of allowing you to take ownership of the project materials.
First, notice that the name of the repo includes the org "plotly-dash-apps" as a prefix. That's about to change. Look for the "fork" button in the top right corner, and click it.



It will ask you where you want to fork it. Choose your personal account.

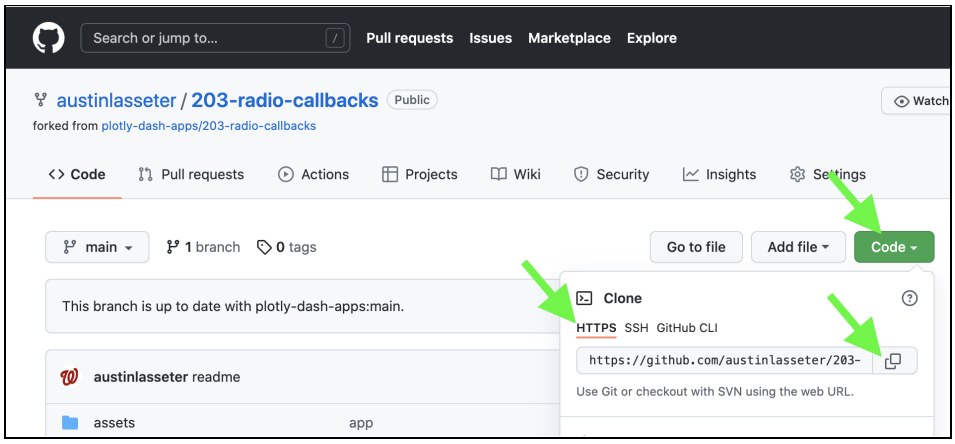


You should now see a new version of the repo, where the prefix is your personal github username, like this:



There's also a message letting you know that your fork is up to date with the "main" branch of the original repo. As time goes on, your fork will begin to diverge from the origin more and more. That's okay. Github has lots of ways to help you keep things more or less in sync (like this), but that's outside the scope of what we want to do today.

Now let's clone to Sagemaker. Click on the green "code" button and choose "https" and then the "copy to clipboard" icon.



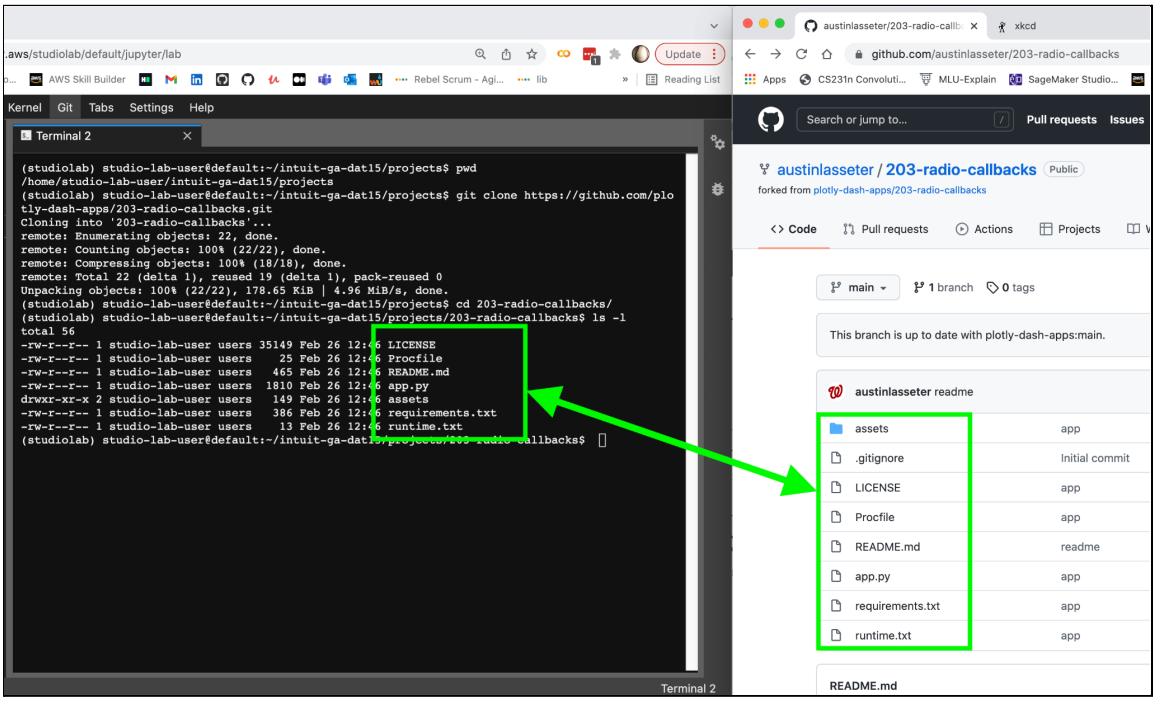
Back in Sagemaker terminal, make sure that you're in the appropriate directory by using the "pwd" command. You should be here:

```
/home/studio-lab-user/intuit-ga-dat15/projects/
```

Clone the repo. Use ctrl+v to paste the url (obviously it will have your name, not mine).

```
git clone https://github.com/austinlasseter/203-radio-callbacks.git
```

Because this is set to be a public repo, git will not ask you for your username or personal access token when you clone the repo - but it will ask you for this information when you try to commit your changes using the "git push" command. More on that in the next section. For now, just "cd" into the repo and check out the files there. They will match what's in the remote origin.



That's it for now! You've successfully cloned the project repo.

3. Setting up your virtual environment with “conda”

When you’re working on an application and you plan to deploy it to the cloud, you need to get in the habit of always working in a python environment. This can be a little tedious at first, but it helps make sure that the list of libraries that are installed on one machine will be identical to the list of libraries installed on another. Failing to do this will result in a lot of conflict down the road.

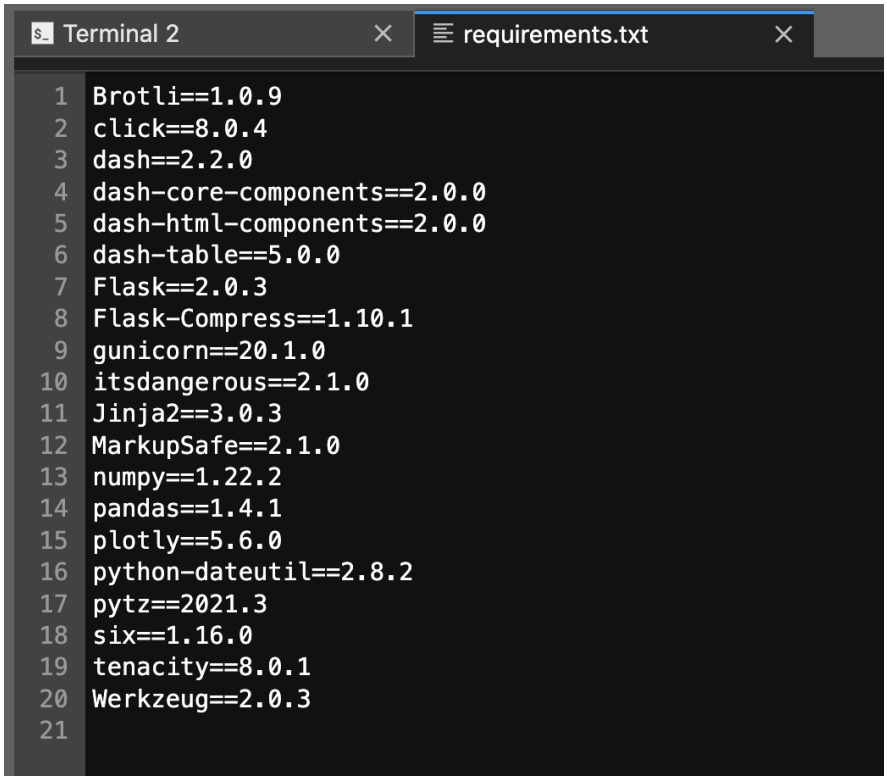
There are two ways to main virtual environments - pip and conda. Sagemaker is built using the Anaconda distribution of python (so it uses conda), but it’s also compatible with pip environments. However, the heroku platform is not compatible with Anaconda environments, and your requirements.txt file will choke on heroku if you create it using conda. So let’s use pip.

Because we’re going to be working a lot with Heroku, we also need to be aware of which python versions Heroku supports (it only supports a few versions). You can read more about that here: https://devcenter.heroku.com/articles/python-support

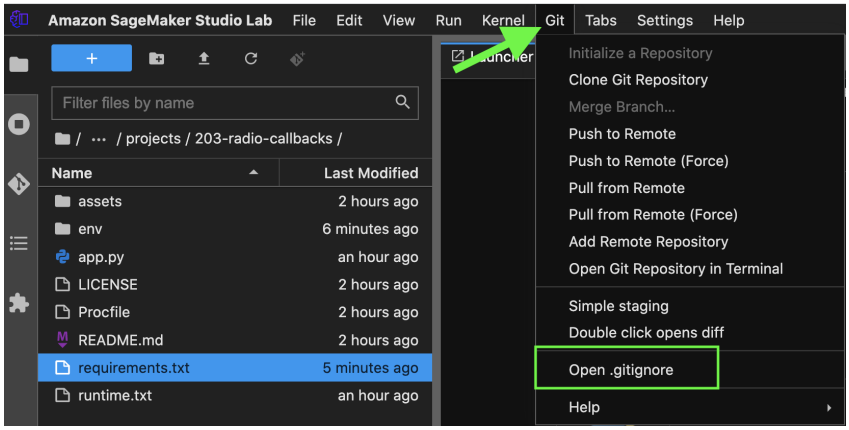
For the purpose of today’s lesson, here’s what you should do.

#	Instructions	Terminal Command
1	Confirm that Sagemaker already has installed python version 3.9.7	`python --version`
2	Create a new environment named simple-dash-env	`python -m venv env`
4	Activate the new environment to use it	`source env/bin/activate`
5	Install the “dash” library	`pip install dash`
6	Install the “pandas” library	`pip install dash`
7	Install the “gunicorn” library	`pip install gunicorn`
8	Create the requirements.txt file	`pip freeze > requirements.txt`

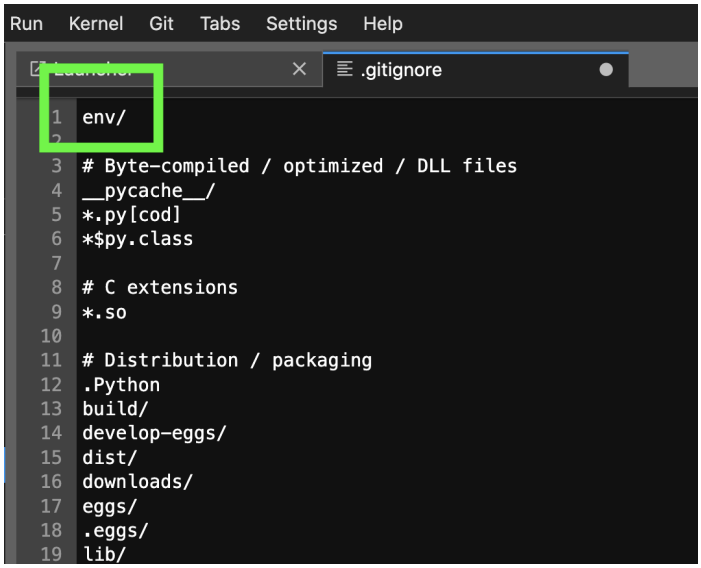
Take a moment to check out the new requirements file that’s been created. It lists all of your python dependencies which are installed in your environment. It will also be used to install these dependencies on heroku. This is a pretty basic list - for more advanced apps, you’ll need to have more libraries installed.



We've also got a new folder called "env" which contains all the binary files used to install these libraries - that's going to slow things down a lot when we try to push to github. To avoid that logjam, we need a hidden file called ".gitignore" which will tell git to "ignore" everything in the /env/ folder. But for reasons no one is sure about, JupyterLab won't display hidden files in the left-hand navigation. Instead you have to open it using the "git" tab in the ribbon.



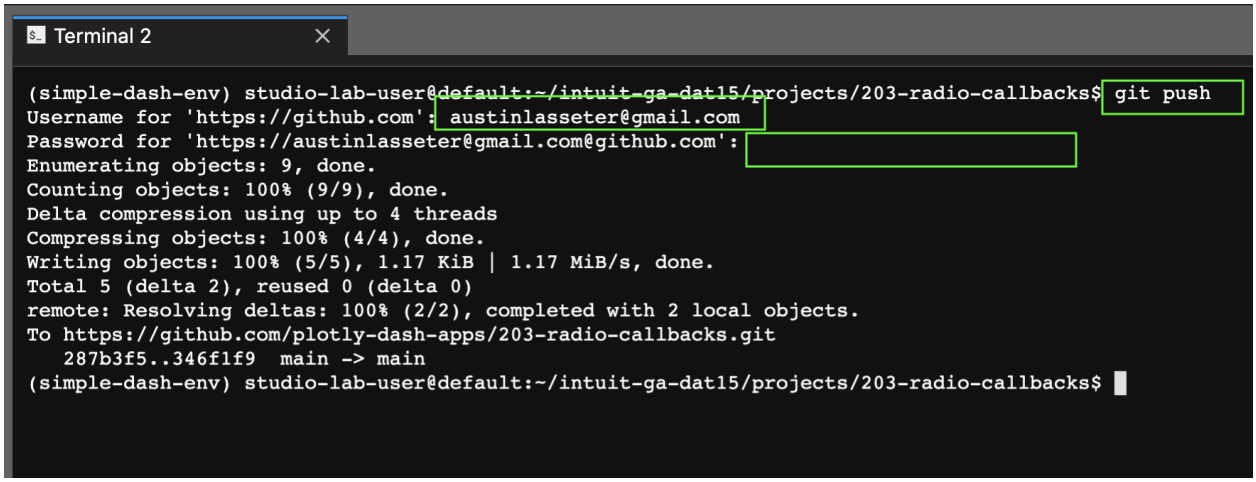
Open up the .gitignore and make sure that /env/ is listed at the top as "env/" – I've provided a template with a lot of other common garbage files as well.



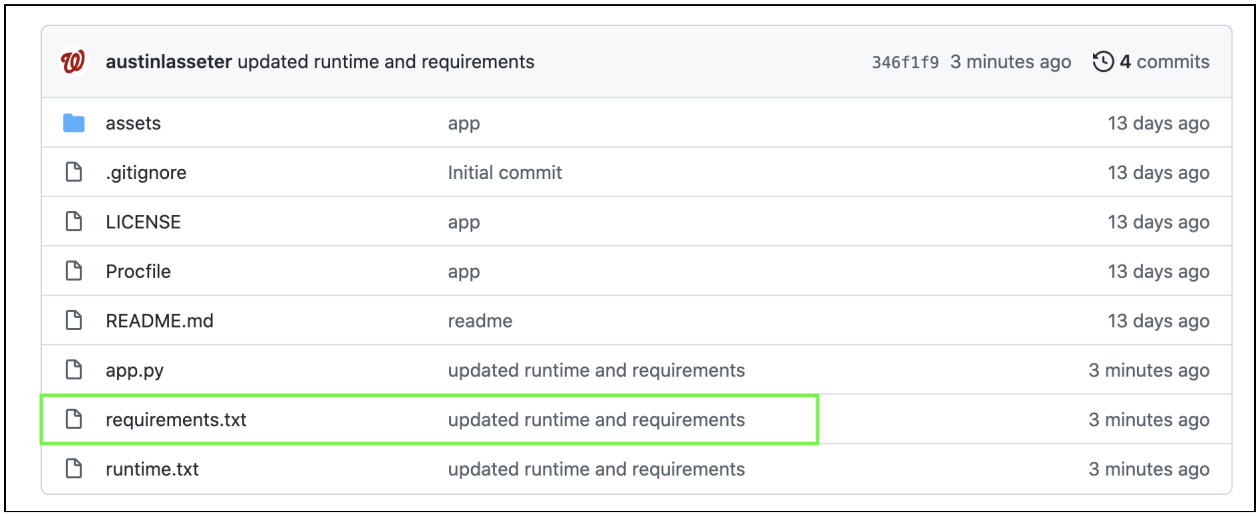
Okay, that was easy. Let's move on.

9	Test out the file "app.py"	`python app.py`
10	Check to see that the following message is displayed in terminal:	`Dash is running on http://127.0.0.1:8050/`
11	Stop running the file "app.py"	`keyboard: ctrl+c`
12	Add your changes to git	`git add .`
13	Write a commit message	`git commit -m "I updated the requirements file"`
14	Push your changes to the remote origin	`git push`

At this point, github will ask you for your username and your "password" (which is actually your personal access token). Remember that the cursor won't move while you type (or paste) the token, so just press "enter" when you're done. It should look like this:



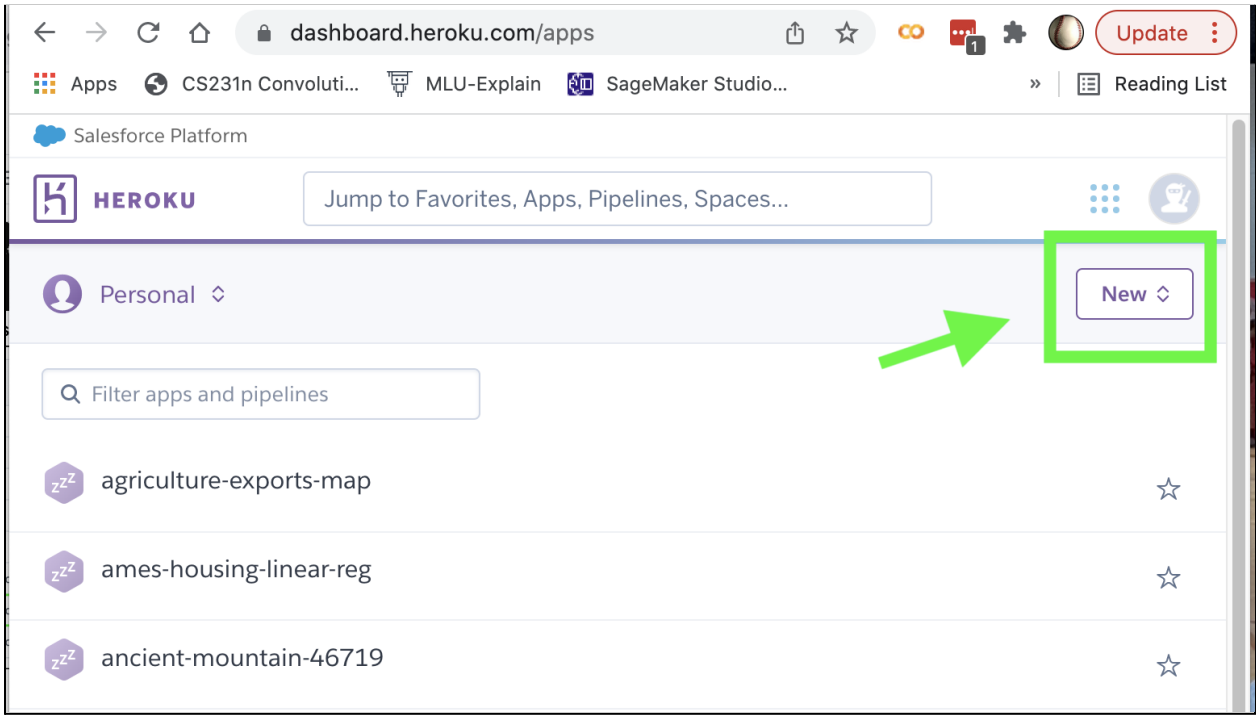
Back in your remote origin repo on github, you should see your commit message similar to this:



Keep in mind that you only have to update the requirements file when you install a new python library into your environment. It's a good idea to use a fresh environment whenever you start a new project, because typically a new project means new dependencies. But you'll get a better idea of that as you try out more and more projects using Plotly Dash.

4. Deploying your app to Heroku

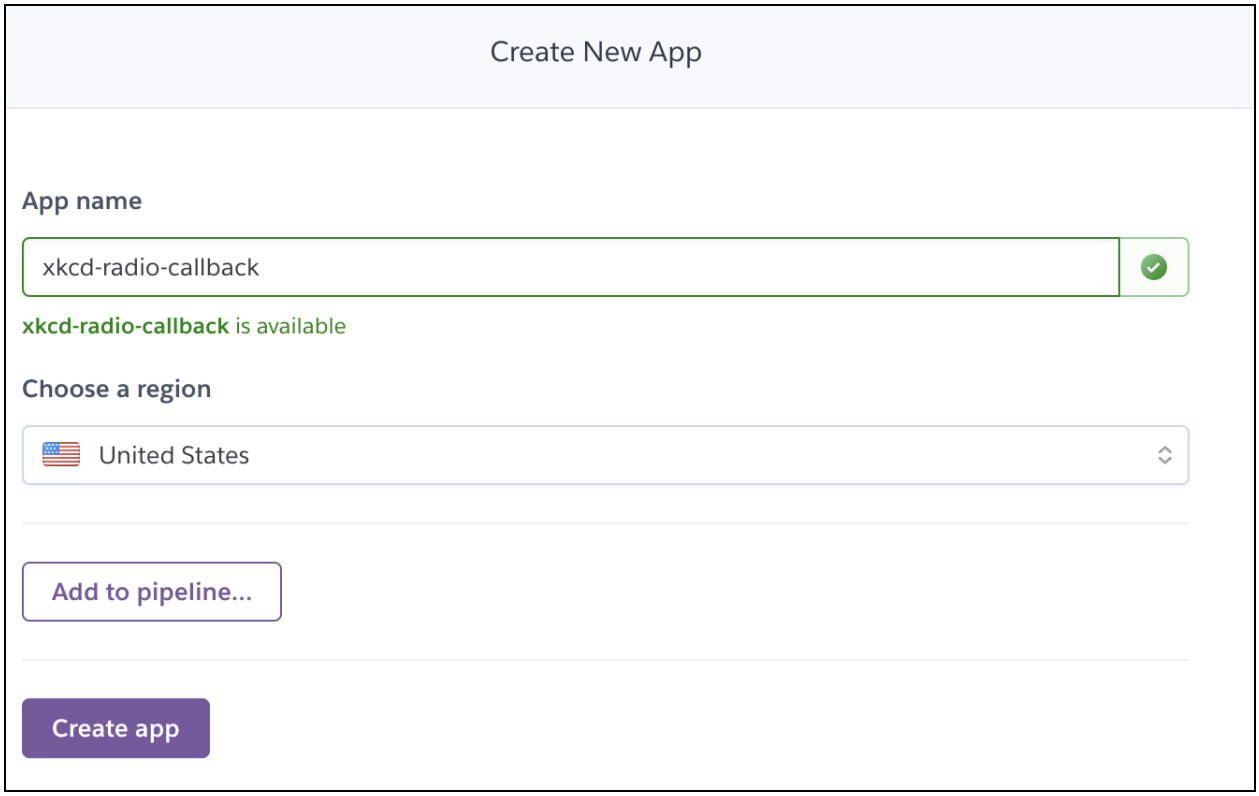
Now let's try deploying our application on heroku. Go to the heroku website and start a new app.



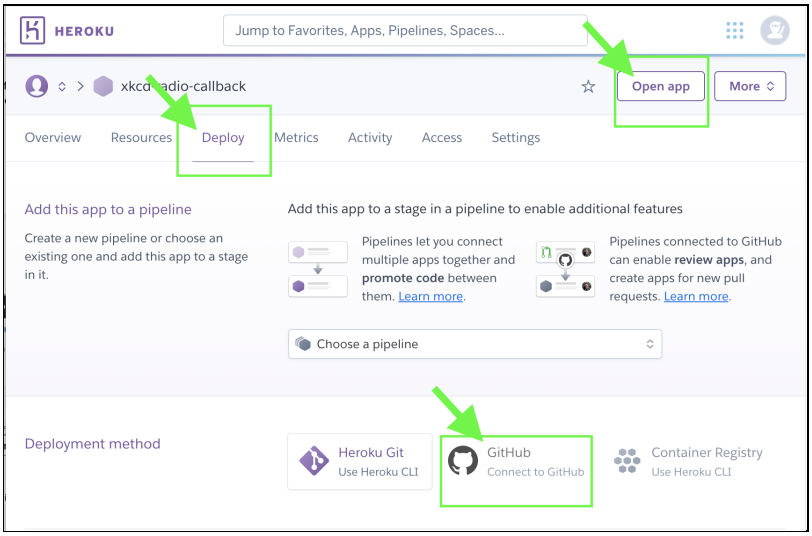
Keep in mind that we're using the free tier so there are some limits. If you are "unverified" (i.e., you did not give them your credit card) then you are limited to 5 apps. If you verify the account by adding a credit card on file, you can have up to 100 apps on the (verified) free plan, and still the 5 app limit for unverified. There is no charge as long as you maintain your apps on the free-tier dynos. You can read more about this here:

- https://stackoverflow.com/questions/48524995/how-many-apps-are-allowed-on-a-hobby-heroku-plan
- https://www.heroku.com/free
- https://devcenter.heroku.com/articles/limits

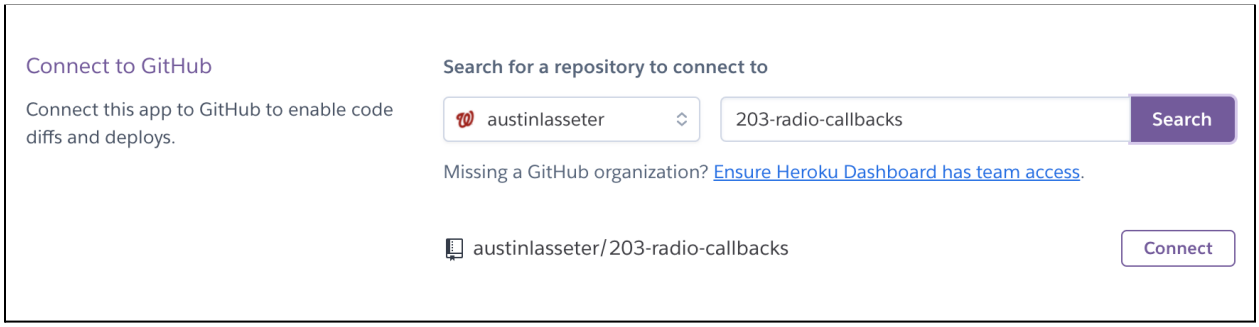
Give your app a unique name. This one is mine.



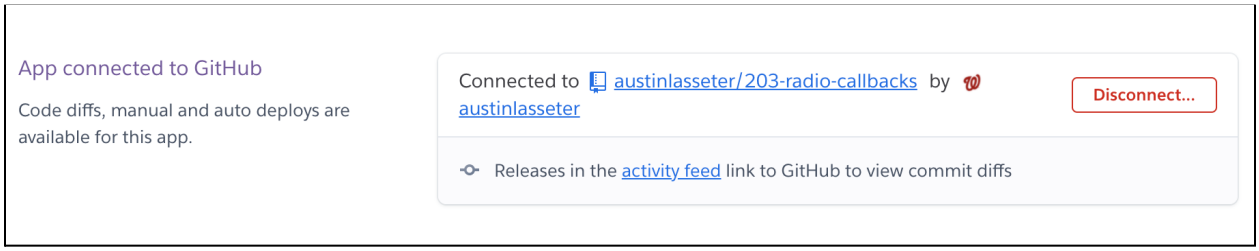
You'll see a lot of options in the next window. Feel free to explore a little. If you choose "open app" you'll see that a placeholder website has been created for you, but nothing is there yet. Under "deploy" choose "github". (There are other ways to deploy to github, like the CLI, but Sagemaker is not compatible with them).



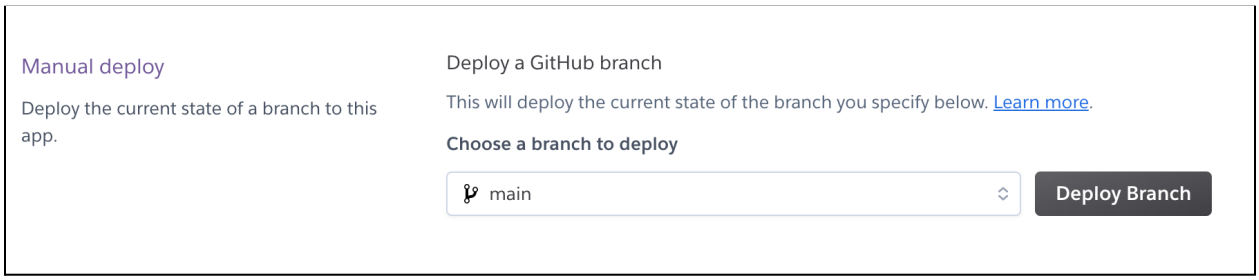
Search for your repo, and choose “connect”. The first time you do this, heroku will ask you for your username and personal access token.



If you authenticate successfully, you will see the following message.

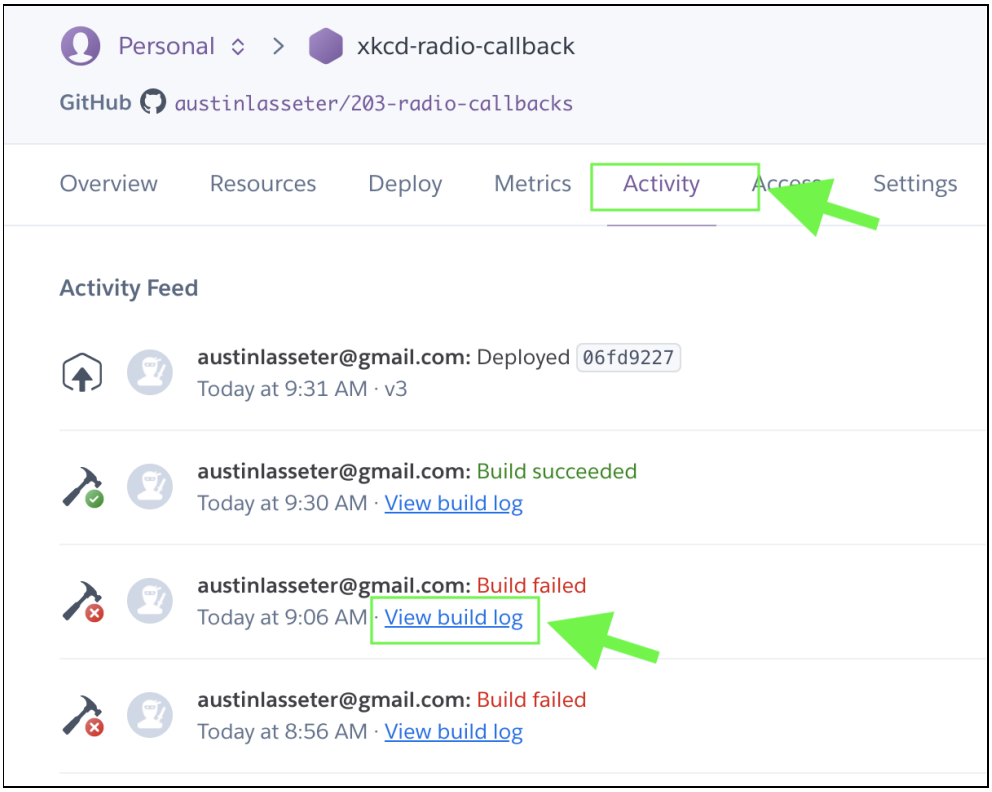


Now scroll down to “manual deploy” and click “deploy branch”.



Wait a couple of minutes for heroku to do its thing. When your app is ready, you should see the option to open your application. My version is deployed here: https://xkcd-radio-callback.herokuapp.com/

Troubleshooting deployment: Sometimes, things do go wrong. You can view your error messages in the log files under “Activity”:



There are usually common problems when you deploy to Heroku:

- There's a syntax error in your python code (and the logs will tell you the line number where you need to make that fix)
- There's a missing dependency in your "requirements.txt" file (so you need to rebuild your virtual environment, and freeze your requirements).

Of course lots of other problems can and will pop up, and we'll try to handle those on a case-by-case basis. Here are some links to help you sort things out:

- Plotly's Dash deployment guide
- Heroku's deployment guide
- Fantastic blog post that dives deep
- Excellent YouTube tutorial

5. Make some changes and deploy again

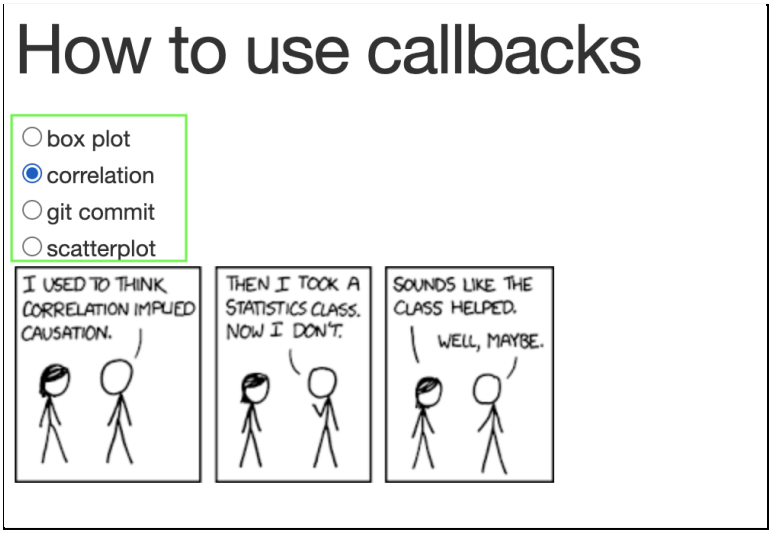
Okay, back in Sagemaker let's make some simple changes to the file "app.py". Because we want to keep things simple today, we are only going to change lines 10-15.

```
7
8  ########### Define your variables ######
9
10 myheading1='How to use callbacks'
11 tabtitle = 'xkcd'
12 list_of_options=['box plot', 'correlation', 'git commit', 'scatterplot']
13 list_of_images=['outlier.png', 'correlation.png', 'gitcommit.jpg', 'scatterplot.png', 'good_code.png']
14 sourceurl = 'https://xkcd.com/'
15 githublink = 'https://github.com/plotly-dash-apps/203-radio-callbacks'
16
```

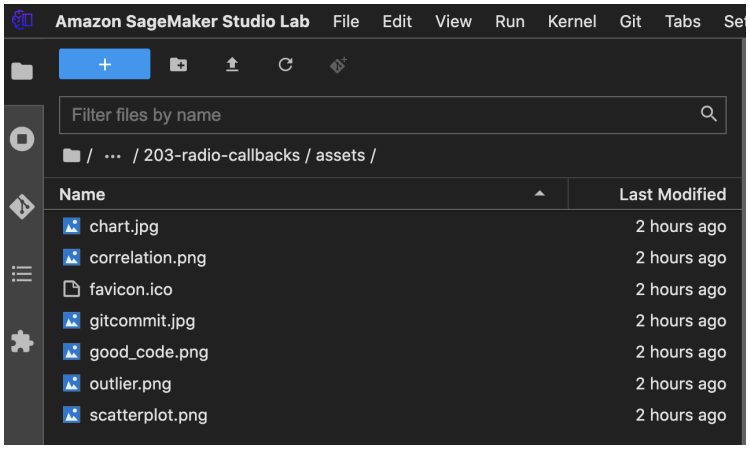
It should be simple to update the variables - just make sure that you keep all your strings in quote marks.

- myheading1
- tabtitle
- sourceurl (replace this with your own data source)
- githublink (replace this with your personal github repo)

The “list of options” is what will appear in your radio-button options. This will get used further down in lines 33-36. Notice that there are exactly four options in line 12, and four images in line 13. Let’s keep the list at exactly four items (no more, no less) for right now.

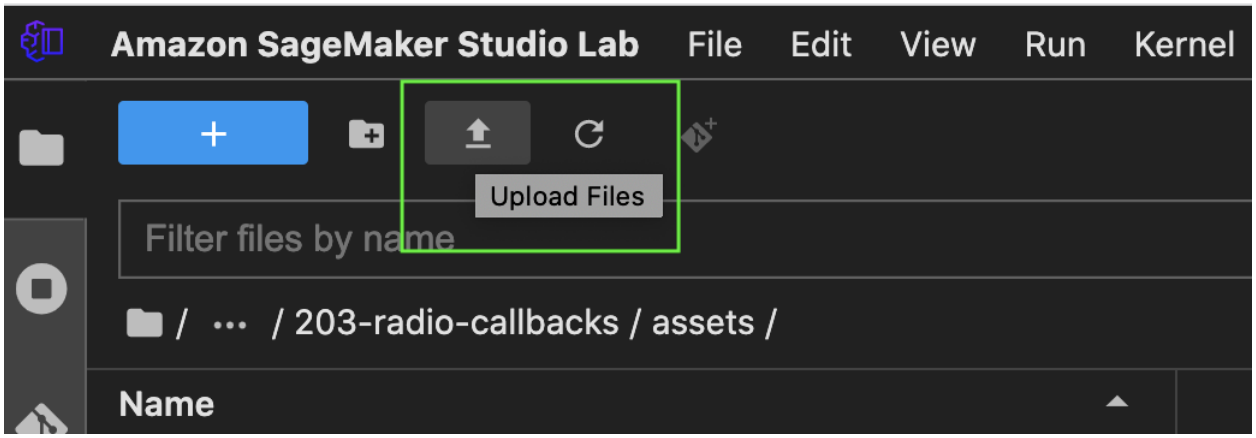


The images are all stored in another folder, called “assets”. Let’s take a look at its contents.



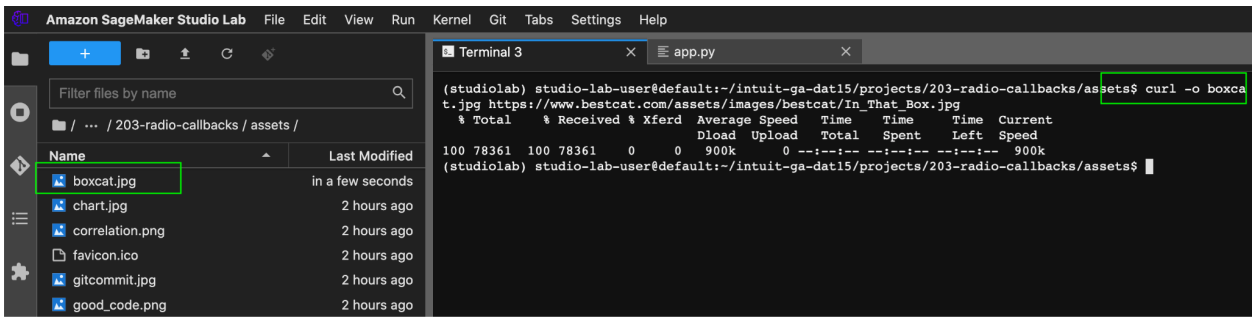
There's a file in here called "favicon" which is the tiny image that displays in your browser tab. You can find cool new favicons on this website: https://www.favicon.cc/

The other jpg and png files correspond to our "list of images" in line 13 of app.py - let's try adding a new file here. Sagemaker has an option to upload files from your laptop:

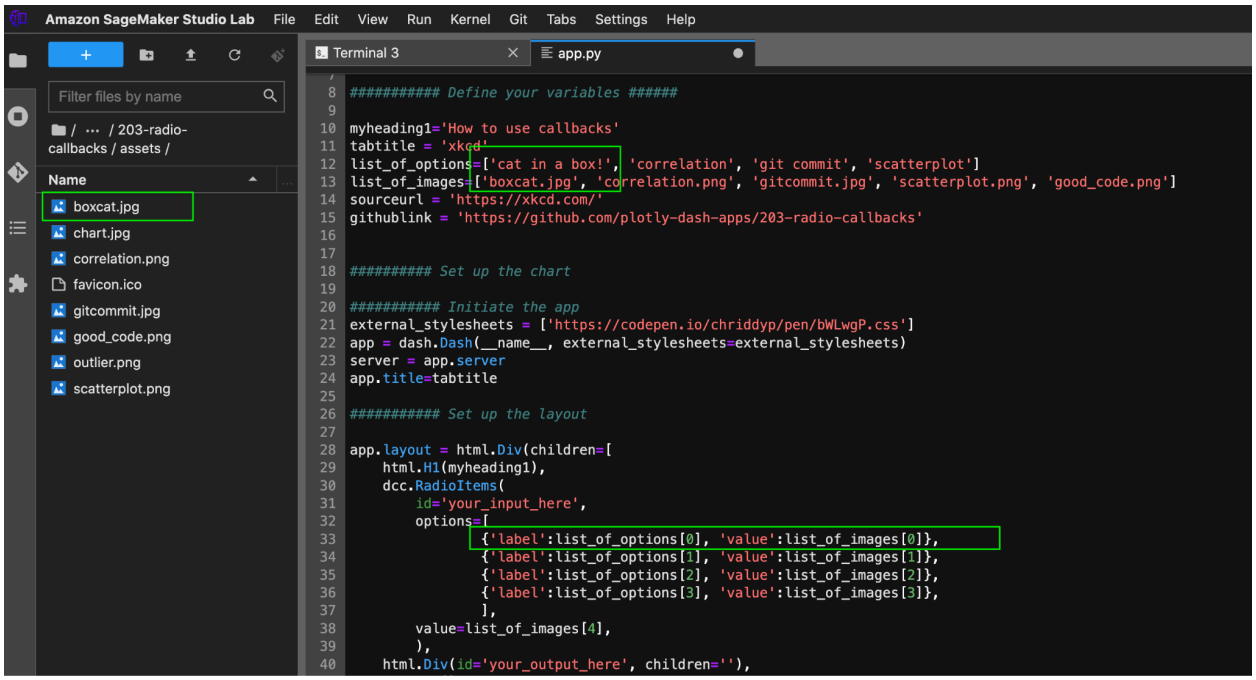


But the "pro" way to do this is in terminal using curl. Make sure you're inside the "assets" folder when you do this.

```
curl -o boxcat.jpg https://www.bestcat.com/assets/images/bestcat/In_That_Box.jpg
```



Now modify "app.py" to use the new file. Make sure that the name of the image in line 13 matches what's in your "assets" folder exactly. Notice that down on line 33, we are using indexing to call the values from our lists - no hard coded values here! - so you don't have to update this line (unless you end up changing the order or number of images in the list).



Now we repeat all the same steps as before.

9	Test out the file “app.py”	`python app.py`
10	Check to see that the following message is displayed in terminal:	`Dash is running on http://127.0.0.1:8050/`
11	Stop running the file “app.py”	`keyboard: ctrl+c`
12	Add your changes to git	`git add .`
13	Write a commit message	`git commit -m "I added a picture of a cat in a box!"`
14	Push your changes to the remote origin	`git push`

At this point, github will ask you for your username and your “password” (which is actually your personal access token).

Check out the remote origin of your repo to make sure that the changes were successful. Over in heroku, go to the “Deploy” tab, scroll down to the bottom, and choose “Deploy Branch”. If everything went smoothly, your app will update with your changes! Note that sometimes the browser on your laptop will cache the previous results - so you may need to open the app in a different browser or in “incognito mode” in order to see the latest version.

When you’re done, post the URL of your app in the #peer-sharing channel so we all celebrate your success!

6. Final thoughts

A lot of you are probably wondering why we are spending so much time on deploying python apps with cat pictures - this is a class about data science and machine learning, right? We need to lay a foundation for deploying simple apps using Sagemaker, Plotly Dash and Heroku. Once you get down the pattern with simple apps, you will be able to build increasingly complex applications and incorporate real machine learning models - but the basic pattern (build, push, deploy) remains the same. Every week during the final hour of class we'll do this same pattern with a new example until it becomes second nature.

In fact, the pattern of deploying to Heroku via github will actually become pretty tedious very quickly - it's a nuisance to have to use github as a middle-man. In addition, it's not possible to view your changes (or your mistakes) until going through this same long process each time. Unfortunately, at this time Sagemaker does not support two key tools that developers rely on:

- Downloading and installing the Heroku CLI tool (so you can skip github completely)
- Viewing your app on localhost before you push and deploy

It's outside the scope of our class to do everything, and we really need to have Sagemaker because it provides so many powerful tools for training and testing high-power machine learning models - something that is difficult to do with your typical laptop. So for the purpose of the class, I will continue using only Sagemaker for demos.

However, if you're feeling confident in your skillset, you may choose to start developing your apps on your laptop, even while you continue to train your models in Sagemaker. This will allow you to install the Heroku CLI and also to view your app on localhost prior to deploying. It's up to you (this is not required, and it's also not recommended for those with limited experience).

You can download and install Python to your laptop using Anaconda like this:
https://www.anaconda.com/products/individual

You can download and install the Heroku CLI to your laptop like this:
https://devcenter.heroku.com/articles/heroku-cli

Here's a blog post I wrote, showing how to do all the steps from today's project using your laptop and the CLI.
https://austinlasseter.medium.com/deploy-a-plotly-dash-app-on-heroku-4d2c3224230

Good luck!