**Deployment**

**Introduction to Deploying Apps**

In this section we’ll look at the final phase of dashboard development - deployment! We show how to deploy your app on Heroku, and how to add a user authentication to your app so that only invited guests can view its contents.

Before deploying your app, you may decide to add user authentication (username and password).

**App Authorization**

From the Dash documentation:

Authentication for dash apps is provided through a separate [**dash-auth**](https://github.com/plotly/dash-auth) package.

**dash-auth** provides two methods of authentication: **HTTP Basic Auth** and **Plotly OAuth**.

**HTTP Basic Auth** is one of the simplest forms of authentication on the web. As a Dash developer, you hardcode a set of usernames and passwords in your code and send those usernames and passwords to your viewers. There are a few limitations to HTTP Basic Auth:

* Users can not log out of applications
* You are responsible for sending the usernames and passwords to your viewers over a secure channel
* Your viewers can not create their own account and cannot change their password
* You are responsible for safely storing the username and password pairs in your code.

**Plotly OAuth** provides authentication through your online Plotly account or through your company's [Plotly On-Premise server](https://plot.ly/products/on-premise/). As a Dash developer, this requires a paid Plotly subscription. Here's where you can [subscribe to Plotly Cloud](https://plot.ly/products/cloud/), and here's where you can contact us about Plotly On-Premise. The viewers of your app will need a Plotly account but they do not need to upgrade to a paid subscription.

Plotly OAuth allows you to share your apps with other users who have Plotly accounts. With Plotly On-Premise, this includes sharing apps through the integrated LDAP system. Apps that you have saved will appear in your list of files at <https://plot.ly/organize> and you can manage the permissions of the apps there. Viewers create and manage their own accounts.

**HTTP Basic Auth** will be sufficient for our purposes. To add authentication to your app, first make sure that both **dash** and **dash-auth** are installed on your system:

$ pip install dash  
 $ pip install dash-auth

Next, pick an app from earlier in the course that you would like to deploy. We’re going to use the solution to our Interactive Components exercise since it’s a fairly short script (it returns the product of two values submitted by a range slider).

Create a new file called **auth1.py** and add the following code:

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

from dash.dependencies import Input, Output

app = dash.Dash()

app.layout = html.Div([

   dcc.RangeSlider(

       id='range-slider',

       min=-5,

       max=6,

       marks={i:str(i) for i in range(-5, 7)},

       value=[-3, 4]

   ),

   html.H1(id='product')  # this is the output

], style={'width':'50%'})

@app.callback(

   Output('product', 'children'),

   [Input('range-slider', 'value')])

def update\_value(value\_list):

   return value\_list[0]\*value\_list[1]

if \_\_name\_\_ == '\_\_main\_\_':

   app.run\_server()

Run the script just to make sure it works, then add the following code (shown in bold):

import dash

**import dash\_auth**

import dash\_core\_components as dcc

import dash\_html\_components as html

from dash.dependencies import Input, Output

**USERNAME\_PASSWORD\_PAIRS = [**

**['JamesBond', '007'],['LouisArmstrong', 'satchmo']**

**]**

app = dash.Dash()

**auth = dash\_auth.BasicAuth(app,USERNAME\_PASSWORD\_PAIRS)**

app.layout = html.Div([

   dcc.RangeSlider(

       id='range-slider',

       min=-5,

       max=6,

       marks={i:str(i) for i in range(-5, 7)},

       value=[-3, 4]

   ),

   html.H1(id='product')  # this is the output

], style={'width':'50%'})

@app.callback(

   Output('product', 'children'),

   [Input('range-slider', 'value')])

def update\_value(value\_list):

   return value\_list[0]\*value\_list[1]

if \_\_name\_\_ == '\_\_main\_\_':

   app.run\_server()

That’s it! Run the script, open a browser to <http://127.0.0.1:8050/> , and you should see be prompted for a username and password before the app will load. We should point out a couple of things:

* The username is case sensitive. JamesBond will work, but jamesbond will not.
* In production, you should store your USERNAME\_PASSWORD\_PAIRS in a separate file or database, and not inside your source code as we have it.
* The field name is arbitrary; we used USERNAME\_PASSWORD\_PAIRS but you can name yours anything you want so long as the same name is passed into **dash\_auth.BasicAuth**.

Resources: <https://dash.plot.ly/authentication>

**Deploying App to Heroku**

Every Dash script so far has used app.run\_server() to launch the app. By default the app runs on **localhost**, and you can only see it on your own machine.

The good news is that Dash uses Flask as its web framework, so anywhere you can deploy Flask, you can deploy Dash. While there are many options out there including Digital Ocean, PythonAnywhere, Google Cloud, Amazon Web Services, Azure, etc., we’ll walk through an app deployment on Heroku.

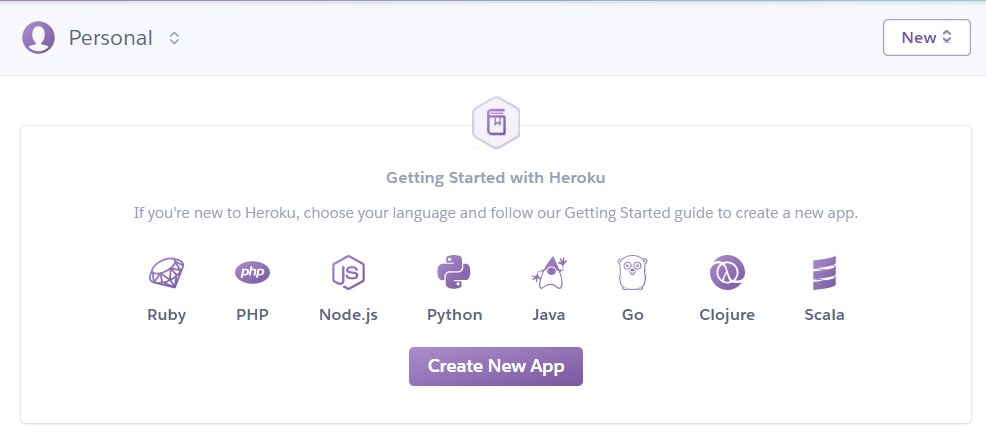
For more information on deploying Flask apps visit <http://flask.pocoo.org/docs/0.12/deploying/>

For more on Heroku visit <https://devcenter.heroku.com/articles/getting-started-with-python#introduction>

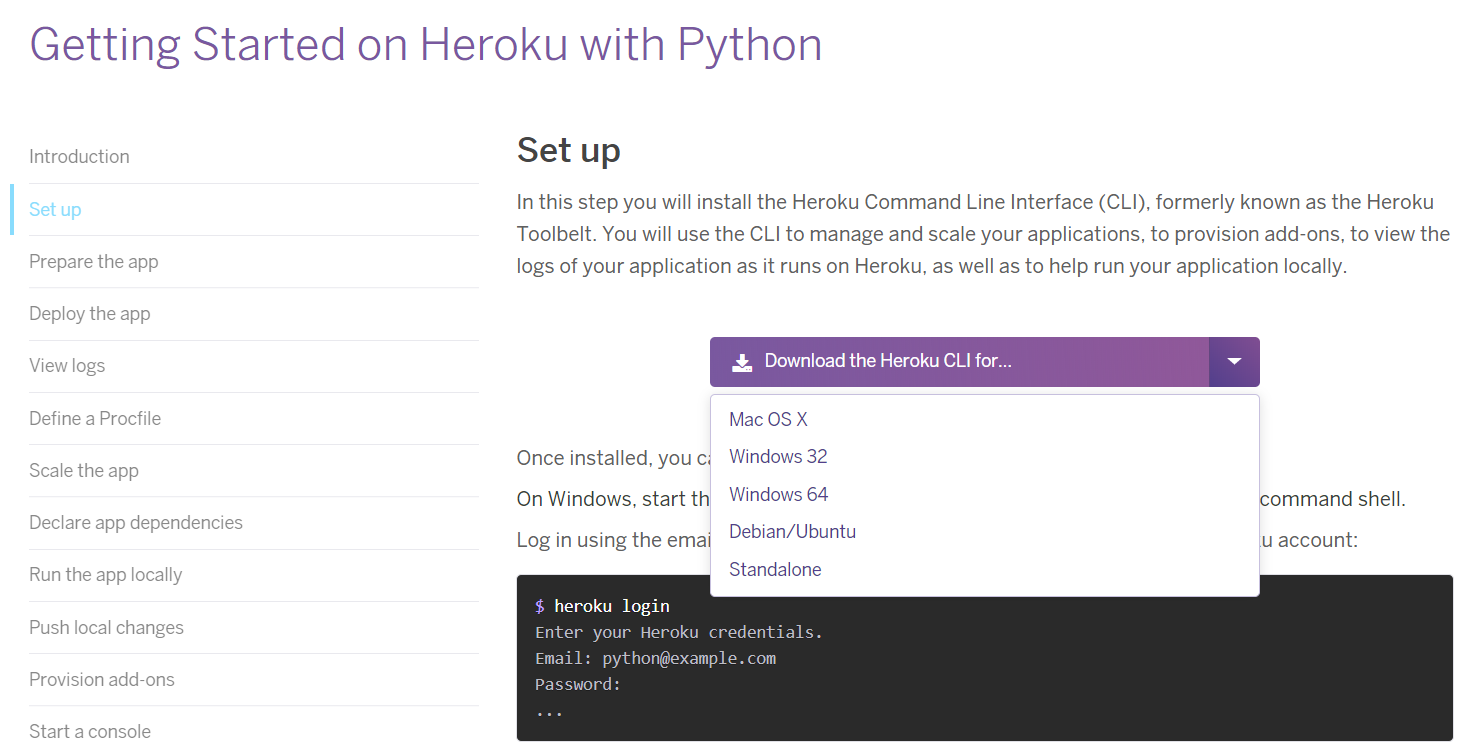
**STEP 1 - Install Heroku and Git**

Heroku is a cloud platform that lets users deploy apps on the web.  
Git is a version control system that will let you keep a local copy of your app for development, and enable you to push changes from your development copy to the deployed version stored at Heroku.

1. Open a **Heroku** account. Free accounts are available at <https://signup.heroku.com/dc>   
   Follow the instructions to obtain a username and password. Write them down!
2. Log into your Heroku account. It should take you to <https://dashboard.heroku.com/apps>



1. Click on **Python**. On the next screen select **Set Up**. An option should appear to download the **Heroku Command Line Interface (CLI)**. Choose your operating system from the dropdown list and follow the instructions to install the utility. You should have the option to install **Git** as well.



1. If **git** was *not* installed with Heroku CLI, you can download it directly from <https://git-scm.com/downloads> and follow the instructions for your operating system.

**STEP 2 - Install virtualenv**

1. Install **virtualenv** if you don’t already have it by typing **pip install virtualenv** at your terminal. Virtualenv allows you to create virtual environments for your app that house Python and all the dependencies your app requires. This includes specific version of plotly, dash, and other libraries that you know will work.   
   As new updates become available, they won’t break your app until you’ve had a chance to test them first!

**STEP 3 - Create a Development Folder**

1. Create a new folder for your project. This will house the “development” copy of your app:  
   C:\>**mkdir my\_dash\_app**  
   C:\>**cd my\_dash\_app**

**STEP 4 - Initialize Git**

1. Initialize an empty git repository:  
   C:\my\_dash\_app>**git init**  
   Initialized empty Git repository in C:/my\_dash\_app/.git/

**STEP 5 (WINDOWS) - Create, Activate and Populate a virtualenv**

**see below for macOS/Linux instructions!**

1. Create a virtual environment. We’re calling ours “venv” but you can use any name you want:  
   C:\my\_dash\_app>**python -m virtualenv venv**
2. Activate the virtual environment:  
   C:\my\_dash\_app>**.\venv\Scripts\activate**
3. Install dash and any desired dependencies into your virtual environment  
   (venv) C:\my\_dash\_app>**pip install dash**(venv) C:\my\_dash\_app>**pip install dash-auth**(venv) C:\my\_dash\_app>**pip install dash-renderer**  
   (venv) C:\my\_dash\_app>**pip install dash-core-components**  
   (venv) C:\my\_dash\_app>**pip install dash-html-components**  
   (venv) C:\my\_dash\_app>**pip install plotly** *(requirement may be satisfied, see below)*

At the time of this writing, **pip install dash** installs:Flask-0.12.2 Jinja2-2.10 MarkupSafe-1.0 Werkzeug-0.14.1 certifi-2018.1.18 chardet-3.0.4 click-6.7 **dash-0.21.0** decorator-4.2.1 flask-compress-1.4.0 idna-2.6 ipython-genutils-0.2.0 itsdangerous-0.24 jsonschema-2.6.0 jupyter-core-4.4.0 nbformat-4.4.0 **plotly-2.5.1** pytz-2018.4 requests-2.18.4 six-1.11.0 traitlets-4.3.2 urllib3-1.22

1. Install a new dependency **gunicorn** for deploying the app:  
   (venv) C:\my\_dash\_app>**pip install gunicorn**

**STEP 5 (macOS/Linux) - Create, Activate and Populate a virtualenv**

1. Create a virtual environment. We’re calling ours “venv” but you can use any name you want:  
   $ **python3 -m python3 -m virtualenv venv**
2. Activate the virtual environment:  
   $ **source venv/bin/activate**
3. Install dash and any desired dependencies into your virtual environment  
   $ **pip install dash**$ **pip install dash-auth**$ **pip install dash-renderer**  
   $ **pip install dash-core-components**  
   $ **pip install dash-html-components**  
   $ **pip install plotly** *(requirement may be satisfied, see above)*
4. Install a new dependency gunicorn for deploying the app:  
   $ **pip install gunicorn**

**STEP 6 - Add Files to the Development Folder**

The following files need to be added:

app1.py a Dash application  
.gitignore used by git, identifies files that *won’t* be pushed to production  
Procfile used for deployment  
requirements.txt describes your Python dependencies, can be created automatically

**app1.py**

Copy the file used in the Basic Authorization section (or any file you’d like to deploy) and add the following code, shown in bold:

import dash

import dash\_auth

import dash\_core\_components as dcc

import dash\_html\_components as html

from dash.dependencies import Input, Output

USERNAME\_PASSWORD\_PAIRS = [

   ['JamesBond', '007'],['LouisArmstrong', 'satchmo']

]

app = dash.Dash()

auth = dash\_auth.BasicAuth(app,USERNAME\_PASSWORD\_PAIRS)

**server = app.server**

app.layout = html.Div([

   dcc.RangeSlider(

       id='range-slider',

       min=-5,

       max=6,

       marks={i:str(i) for i in range(-5, 7)},

       value=[-3, 4]

   ),

   html.H1(id='product')  # this is the output

], style={'width':'50%'})

@app.callback(

   Output('product', 'children'),

   [Input('range-slider', 'value')])

def update\_value(value\_list):

   return value\_list[0]\*value\_list[1]

if \_\_name\_\_ == '\_\_main\_\_':

   app.run\_server()

**.gitignore**

venv  
\*.pyc  
.DS\_Store  
.env

**Procfile**

web: gunicorn app1:server

app1 refers to the filename of our application (app1.py) and server refers to the variable **server** inside that file.

**requirements.txt**

This can be automatically generated by running pip freeze > requirements.txt at the terminal.   
Make sure to do it from inside the development folder with the virtual environment activated.

(venv) C:\my\_dash\_app>**pip freeze > requirements.txt**

Results in a file that looks something like this:

certifi==2018.1.18

chardet==3.0.4

click==6.7

dash==0.21.0

dash-auth==0.1.0

dash-core-components==0.22.1

dash-html-components==0.10.0

dash-renderer==0.12.1

decorator==4.2.1

Flask==0.12.2

Flask-Compress==1.4.0

Flask-SeaSurf==0.2.2

gunicorn==19.7.1

idna==2.6

ipython-genutils==0.2.0

itsdangerous==0.24

Jinja2==2.10

jsonschema==2.6.0

jupyter-core==4.4.0

MarkupSafe==1.0

nbformat==4.4.0

plotly==2.5.1

pytz==2018.4

requests==2.18.4

retrying==1.3.3

six==1.11.0

traitlets==4.3.2

urllib3==1.22

Werkzeug==0.14.1

**STEP 6 - Log onto your Heroku Account**

At the terminal, login using the credentials you established in **STEP1**:

(venv) C:\my\_dash\_app>**heroku login**  
Enter your Heroku credentials:  
Email: my.name@somewhere.com  
Password: \*\*\*\*\*\*\*\*  
Logged in as my.name@somewhere.com

**STEP 7 - Initialize Heroku, add files to Git, and Deploy**

(venv) C:\my\_dash\_app>**heroku create *my-dash-app***

You have to change ***my-dash-app*** to a unique name. The name must start with a letter   
and can only contain lowercase letters, numbers, and dashes.

(venv) C:\my\_dash\_app>**git add .**

Note the period at the end. This adds all files to git (except those listed in .gitignore)

(venv) C:\my\_dash\_app>**git commit -m "Initial launch"**

Every git commit should include a brief descriptive comment. Depending on your operating system, this comment may require double-quotes (not single-quotes).

(venv) C:\my\_dash\_app>**git push heroku master**

This deploys your current code to Heroku. The first time you push may take awhile as it has to set up Python and all your dependencies on the remote server.

(venv) C:\my\_dash\_app>**heroku ps:scale web=1**   
Scaling dynos... done, now running web at 1:Free

This runs the app with a 1 heroku "dyno"

**STEP 8 - Visit Your App on the Web!**

You should be able to view your app at [https://**my-dash-app**.herokuapp.com](https://my-dash-app.herokuapp.com)(changing **my-dash-app** to the name of your app)

**STEP 9 - Update Your App**

Any time you make changes to your app, add new apps to your repo, or install new libraries and/or upgrade existing dependencies in your virtual environment, you want to push the latest updates to Heroku. These are the basic steps:

If installing a new package:

$ **pip install *newdependency***

$ **pip freeze > requirements.txt**

If updating an existing package:

$ **pip install *dependency* --upgrade**

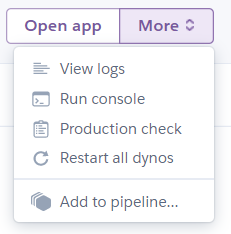
$ **pip freeze > requirements.txt**

In all cases:

$ **git status** *# view the changes (optional)*  
$ **git add .**  *# add all the changes*$ **git commit -m "a description of the changes"**  
$ **git push heroku master**

**TROUBLESHOOTING**

If your app won’t launch on Heroku, follow this checklist:

☐ **app1.py** includes **server = app:server**If not, add this line, save the file, then run Git add/commit/push 

☐ **gunicorn** installed, and included in **requirements.txt**  
If not, run **pip install gunicorn**, then **pip freeze > requirements.txt**, then run Git add/commit/push

☐ If unable to trace locally, visit your Heroku dashboard and click on **More** / **View logs**

Resources: <https://dash.plot.ly/deployment>