Web Deployment

It’s Launchtime!

Let’s Deploy!

Congrats! We’ve just completed using Flask to create a functioning app.

Let’s Deploy!

Congrats! We’ve just completed using Flask to create a functioning app.

However, we’re only able to run it locally right now...

Let’s Deploy!

Congrats! We’ve just completed using Flask to create a functioning app.

However, we’re only able to run it locally right now…

Let’s see how we can deploy this app on the actual web!

Let’s Deploy!

We’ll be working with Heroku, a platform as a service (PaaS) that enables developers like us to build, run, and operate our apps on the cloud….

Let's Deploy!

We'll be working with Heroku, a platform as a service (PaaS) that enables developers like us to build, run, and operate our apps on the cloud….

In other words, Heroku will allow us to use its servers to publish our anagram app on the web :D

Let’s Deploy!

(We chose Heroku because it supports several languages, one of which is Python! It is also very compatible with Flask, so it’s a great choice to get started.)

Let's Deploy!

The first thing we need to do is tell Heroku which libraries we need to use to run our app. Luckily, we can enter a small command into our terminal to do just that:

```
$ pip freeze > requirements.txt
```

Let’s Deploy!

$ pip freeze > requirements.txt

Running this command creates a file called `requirements.txt` that contains all the libraries we’re using as well as their versions.

Let’s Deploy!

Great! It worked!

Let’s Deploy!

… But let’s check that it worked.

Let's Deploy!

Command line practice: how can we see what's in requirements.txt? (There are a few ways. Some ways require opening the file, others do not. Pick your favorite method!)

Let's Deploy!

Alrighty, the next step is to define a set of processses/commands that Heroku should run before launching our application. We can do this by creating a file called "Procfile."

Let's Deploy!

Alrighty, the next step is to define a set of processses/commands that Heroku should run before launching our application. We can do this by creating a file called "Procfile."

Procfile: A simple *text file* that declares what commands are going to be run by our platform (in this case, Heroku)

Let’s Deploy!

Let’s create this Procfile in our terminal:

Let's Deploy!

Our procfile should read as follows:

`web: gunicorn app:app`

Let’s Deploy!

That’s it!

Let’s Deploy!

But what does it all mean???

Let’s Deploy!

Let’s break it down:

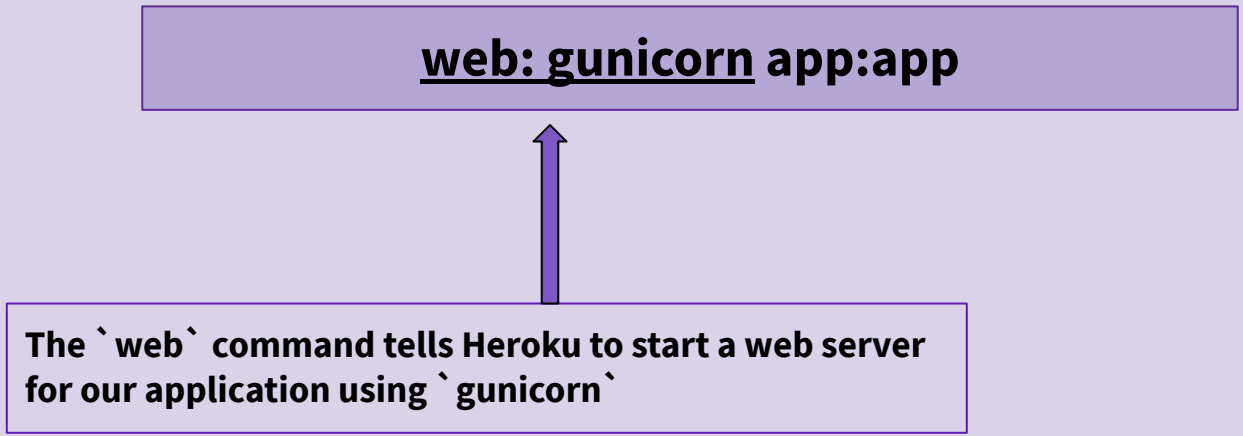
Let’s Deploy!

Let’s break it down:

```
web: gunicorn app:app
```

Let's Deploy!

Let's break it down:

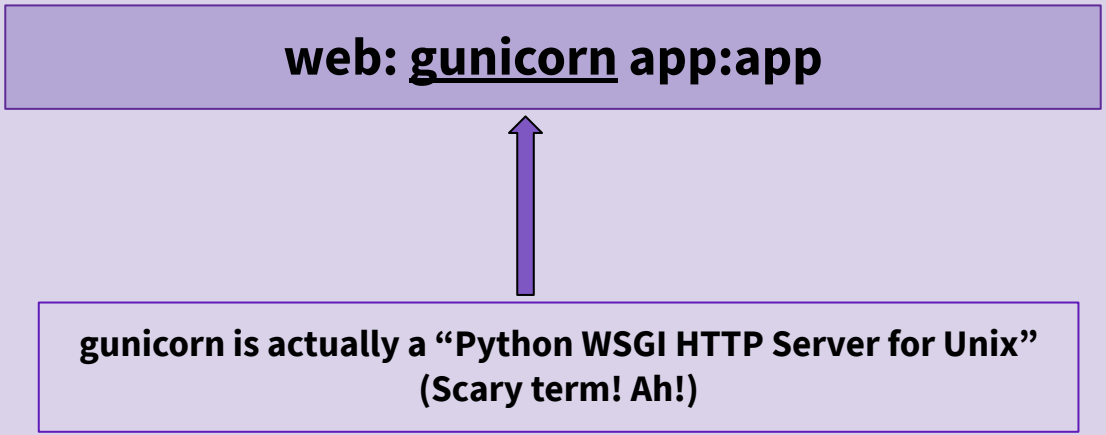


Let's Deploy!

Let's break it down:

Let's Deploy!

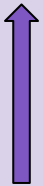
Let's break it down:



Let's Deploy!

Let's break it down:

web: gunicorn app:app



What that means in (very simplified) plain English is that gunicorn will take our code and do a lot of work underneath the hood so that our code will be displayed in any browser, as long as the user enters the correct URL.

Let's Deploy!

Let's break it down:

Let’s Deploy!

Let’s break it down:

web: gunicorn app:app

And that’s it! That’s our Procfile :D

Let’s Deploy!

The next step is to create a Heroku account.

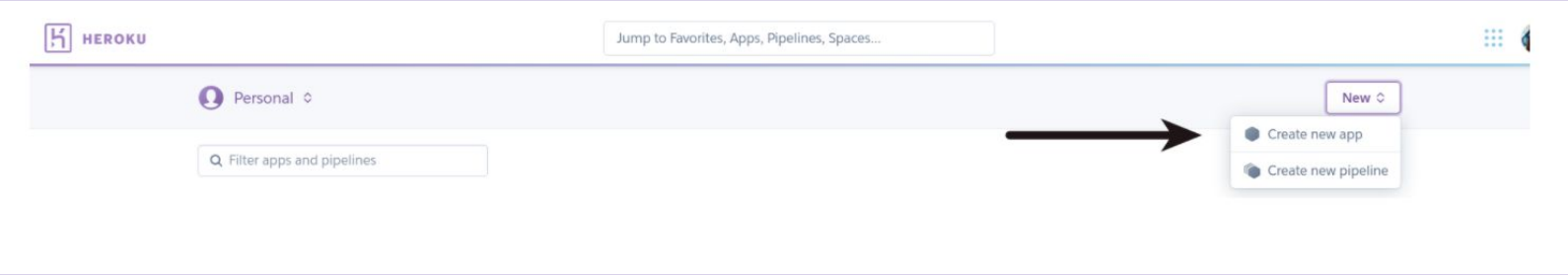
Let's Deploy!

Let's use Unicornelius's credentials:

```
Email: unicornelius.dazzle@gmail.com
Password: st@nfordPyth0ncs41
```

Let's Deploy!

Now let's go to the Heroku website! We'll go to the top right and click New → Create new App.



Let's Deploy!

We can name our app anything we want. This name will be the first part of our app's default URL:

{app-name}.herokuapp.com

Let’s Deploy!

Because we’re hosting section in the US, let’s also host our app on US servers :D

Let’s Deploy!

Because we’re hosting section in the US, let’s also host our app on US servers :D

Let’s Deploy!

Once this app is deployed on Heroku, we’re ready to deploy it online!

Let’s Deploy!

The next steps can all take place in the command line:

Let's Deploy!

The next steps can all take place in the command line:

`$ git init .`

We'll use git to create a repository that will store our code.

Let's Deploy!

The next steps can all take place in the command line:

`$ git init .`

`$ git add app.py Procfile requirements.txt`

Now we'll add the three files we created to that repository:

- **app.py**
- **Procfile**
- **requirements.txt**

Let's Deploy!

The next steps can all take place in the command line:

`$ git init .`

`$ git add app.py Procfile requirements.txt`

`$ git commit -m "first commit"`

Then we'll commit those changes.

Let's Deploy!

`$ heroku login -i`

`$ heroku git:remote -a {app-name}`

Finally, we'll use heroku commands to deploy! (Make sure you have the Heroku CLI installed into your computer or virtual environment!)

Let's Deploy!

`$ heroku login -i`

`$ heroku git:remote -a {app-name}`

Note: Be sure to replace {app-name} with your actual app's name!!

Let's Deploy!

`$ git push heroku master`

Finally, we'll run this last magical command.

Let's Deploy!

A bunch of lines should be appearing in our terminal now...

Let's Deploy!

A bunch of lines should be appearing in our terminal now...

... among the last of those lines is our URL!

Let’s Deploy!

Copy this URL into your browser, and it should take you to our officially deployed app! Congratulations!

Questions?

Sources!

Shoutout to all the links on the right!

- https://stackabuse.com/deploying-a-flask-application-to-heroku/
- https://devcenter.heroku.com/articles/heroku-cli
- https://pythonbasics.org/what-is-flask-python/
- https://www.fullstackpython.com/green-unicorn-gunicorn.html
- https://www.fullstackpython.com/wsgi-servers.html