
Model Deployment

Deploying the model with Heroku



Democratizing Data Science Learning

Learning Objectives

What is Heroku?

Gunicorn

**Create the required
Heroku files**

**Making a git
repository and
committing
changes**

**Push your app to
the web**

Debugging Errors

At this point, you have a functioning web app running on your local machine. However, it is hard to share this — your machine might be behind a firewall, your IP address might change often or the machine isn't always on.

Luckily, it's possible to deploy Flask apps to an online platform that will make it much easier for people to access your app. We'll do this using Heroku.

What is Heroku?

Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps. The platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market.

Heroku offers a free version to host your app.

Requirements

The minimum things you need to do for the current task:

1. Sign up for a free Heroku account at <https://signup.heroku.com/signup/dc>
2. Make sure you have [git](#) installed, to push your app to Heroku.
3. Install the [Heroku CLI tool](#).

NOTE: If you face a path error after installing the Heroku CLI Tool, you need to add path manually. To do this, you can follow the procedure to add path as shown at 1:20 in the video in the next slide.

Fixing the Path Error in Heroku CLI



Files

The repository consisting of all the required files for the application can be found here:

[https://github.com/dphi-official/Micro-Courses/tree/master/Introduction Model Deployment](https://github.com/dphi-official/Micro-Courses/tree/master/Introduction_Model_Deployment)

Gunicorn: A better web server

The Flask web framework feature convenient built-in web servers that we've been working with till now, but these servers only process a single request at a time. If you deploy with one of these servers on Heroku, your resources will be underutilized and your application will feel unresponsive.

Gunicorn is a pure-Python HTTP server for WSGI applications(i.e Flask). It allows you to run any Python application concurrently by running multiple Python processes within a resource. It provides a perfect balance of performance, flexibility, and configuration simplicity.

In simple terms, Gunicorn is a tool or an interface that takes care of everything that happens in-between the web server and your web application

You can install Gunicorn with pip:

```
$ pip install gunicorn
```


Create the required Heroku files

1. **Procfile** - this tells Heroku what kind of app you are running and how to serve it to users. It is a single line and should look like this:

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

- The first app represents the name of the python file that runs your application or the name of the module it is in. In our case, we have a server.py file inside server folder. So we'll be editing this in the next slide.
- The second app represents your app name i.e from this line in our server.py file:
app = Flask(__name__)

The Procfile is always a simple text file that is named Procfile **without a file extension**. For example, Procfile.txt is not valid.

The Procfile **must live in your app's root directory**. It does not function if placed anywhere else.

Create the required Heroku files

Since our app is inside the server folder, we'll have to modify the Procfile as follows:

```
web: gunicorn server.server:app
```

Create the required Heroku files

2. **requirements.txt** - this tells Heroku which packages to install for your web app.

Instead of manually specifying the packages to be installed, **pipreqs** package can be used to generate a requirements.txt file based on the import statements of the project.

To get setup with the pipreqs, you need to install the pipreqs package using the following command:

```
pip install pipreqs
```

Usage of the pipreqs is very easy. You just need to provide the root location of your project and pipreqs automatically generate a requirements.txt file in root folder.

```
pipreqs /path/to/project
```

Create the required Heroku files

For eg, for creating a requirements.txt file from my folder named House_Price_Application that contains all the files related to this project, I'm doing the following:

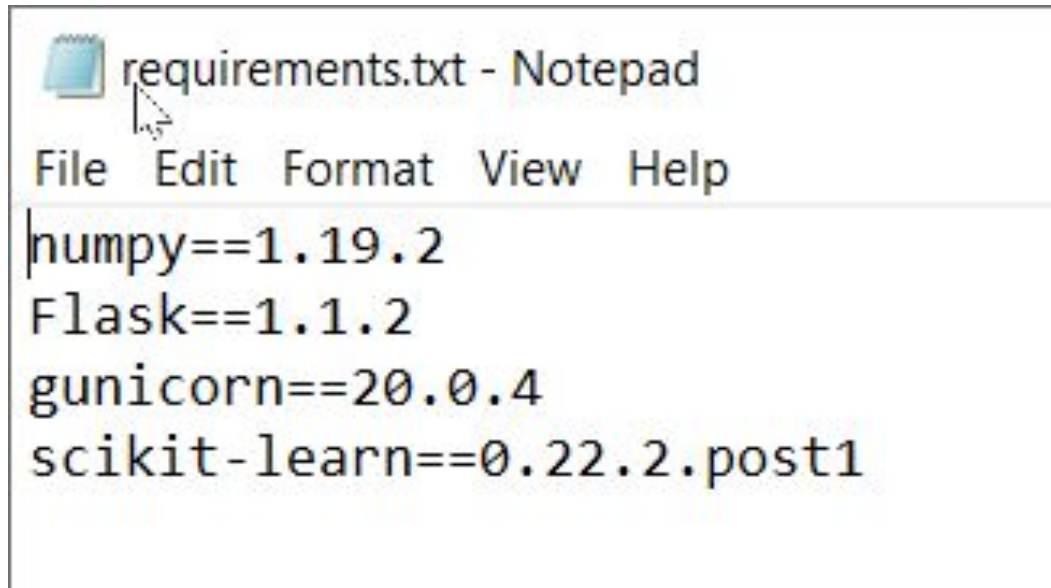
```
Gunnika@LAPTOP-T5HA6JPL MINGW64 ~/Desktop
$ pipreqs "C:\Users\Gunnika\Documents\House_Price_Application"
INFO: Successfully saved requirements file in C:\Users\Gunnika\Documents\House_P
rice_Application\requirements.txt
```

A requirements.txt file now magically appears in my root folder (House_Price_Application)!

PS. There are other methods to create requirements.txt file like pip freeze as well. However, it works better when you have created a virtual environment for your project. Otherwise, it lists all the packages installed in your system.

Create the required Heroku files

These should be the constituents of your requirements.txt file (the versions may be different). If any of these are not added automatically, ensure that you add them.

A screenshot of a Notepad window titled "requirements.txt - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text inside the window is:

```
numpy==1.19.2
Flask==1.1.2
gunicorn==20.0.4
scikit-learn==0.22.2.post1
```

```
requirements.txt - Notepad
File Edit Format View Help
numpy==1.19.2
Flask==1.1.2
gunicorn==20.0.4
scikit-learn==0.22.2.post1
```

Make a git repository for your web app

Inside the root folder, run the following command to create a new repository.

```
git init
```

Add files to repository

While in the root folder, use the following command to add all your web app's files to the git repository:

```
git add .
```

PS. There's a gap between add and .

Commit files

Now, we'll commit all the added files to GitHub. This can be done with the command:

```
git commit -m "initial commit"
```


Authenticate with Heroku

Once this is done, you can log into your Heroku account using the CLI.

```
heroku login
```

It'll display the message:

```
$ heroku login  
heroku: Press any key to open up the browser to login or q to exit: |
```

Press any key, Enter. A new browser window will appear from which you can log in to your Heroku account.

Then come back to the terminal and press Ctrl+C and Y in order to get the terminal back to its normal state.

Create a new Heroku app

You can create a heroku app using the command (don't implement this one):

```
heroku create
```

By default, this will make an app with a random name.

If you want to choose your own name, simply pass it as an argument (implement this). For example:

```
heroku create blr-house-price
```

The above command creates an app with the name blr-house-price.

Note: You need to choose a unique app name that hasn't been used before.

Push your app to the web

Just one more command and your web app will be online. During this process, Heroku will upload your app files, install the packages it needs and start the app running.

```
git push heroku master
```

If everything goes as expected, you'll see output showing things being installed and uploaded.

Check out the deployed app!

You can use the **heroku open** command to open your completed app in the web browser.

Alternatively, just go to <https://blr-house-price.herokuapp.com/> (replacing blr-house-price with your app's name, of course).

You can share that link with whoever you want to see your model.

Important Note

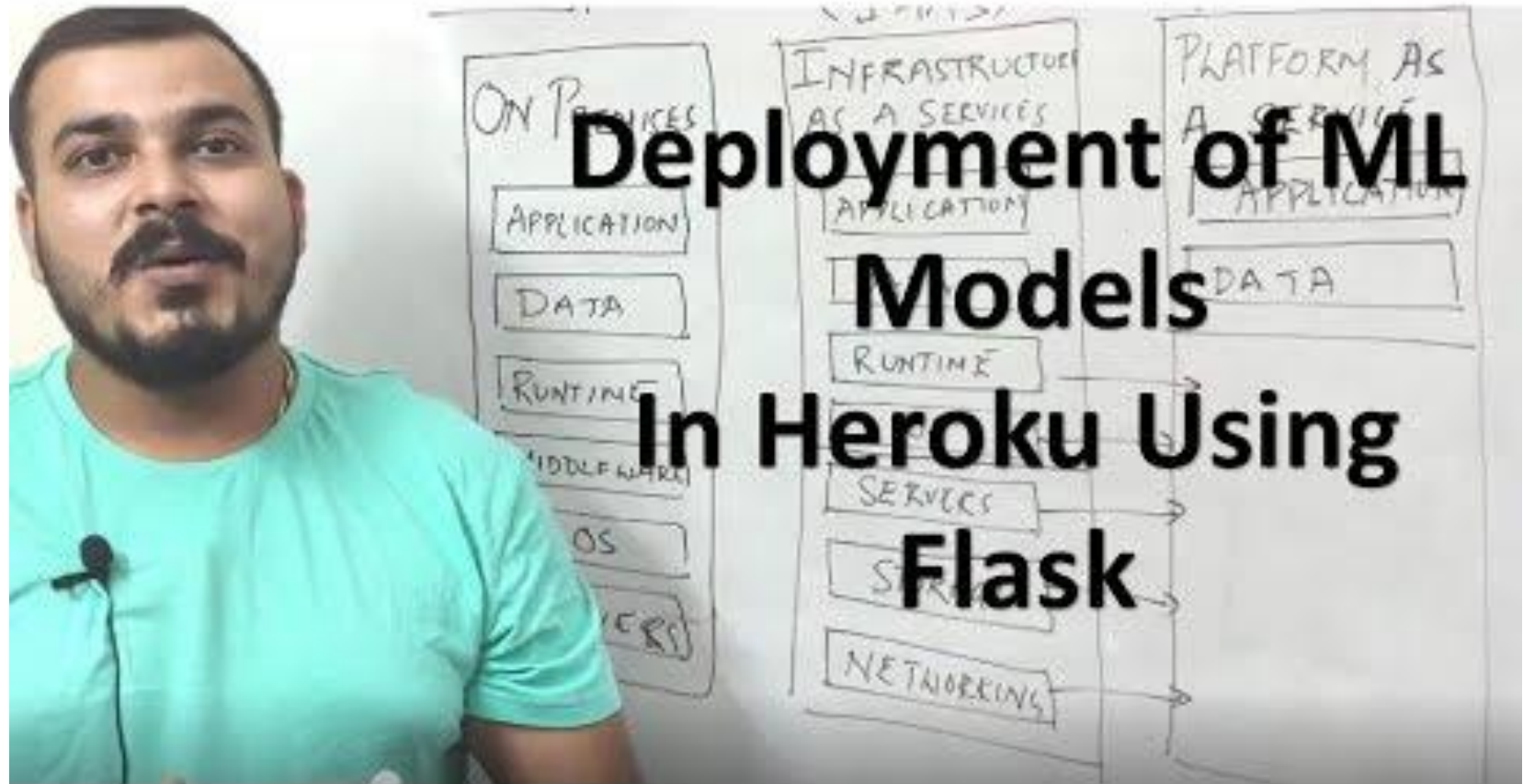
The method of Flask app creation used in the Bengaluru Home Price prediction video series is valid but a little outdated. The series was however more detailed and great for diving into.

The more recent methods make use of Flask's `render_template` functionality to render the HTML file.

Although we have incorporated that in the instructor's code, we would like you to go through the video in the next module that covers the whole process quickly.

When you get a proper understanding from the detailed videos + get updated with the newest method in the short video, you'll be able to create amazing applications on your own!

Walkthrough of the whole process



The image shows a man with a beard and mustache, wearing a green t-shirt, standing in front of a whiteboard. The whiteboard contains a diagram illustrating the layers of cloud services. The diagram is organized into three main columns, each representing a different service model:

- ON PREMISES:** This column lists layers from top to bottom: APPLICATION, DATA, RUNTIME, MIDDLEWARE, OS, and SERVERS.
- INFRASTRUCTURE AS A SERVICE:** This column lists layers from top to bottom: APPLICATION, RUNTIME, SERVICES, STORAGE, and NETWORKING.
- PLATFORM AS A SERVICE:** This column lists layers from top to bottom: APPLICATION and DATA.

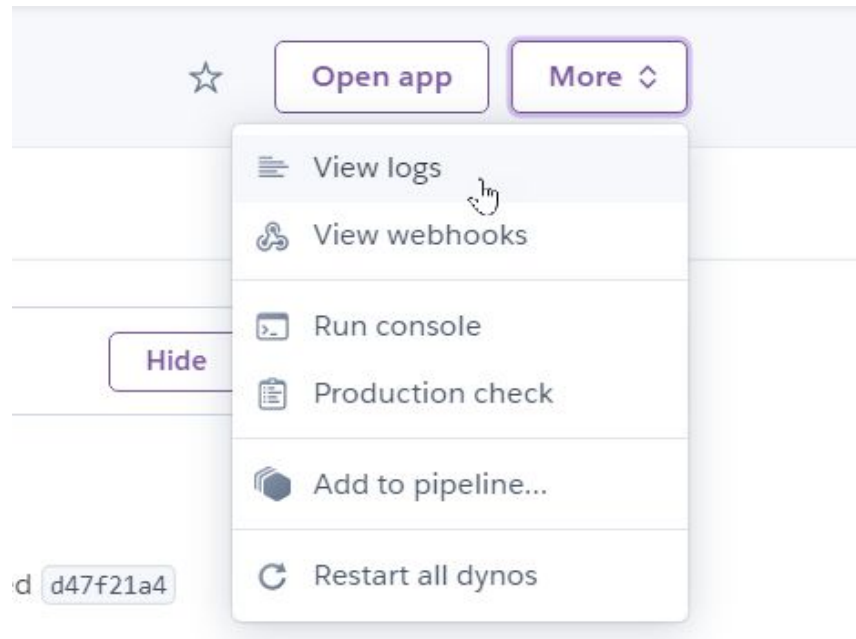
Arrows indicate dependencies or interactions between layers across the different models. For example, the 'APPLICATION' layer in 'INFRASTRUCTURE AS A SERVICE' depends on the 'RUNTIME' layer, which in turn depends on the 'SERVICES' layer, and so on. The 'PLATFORM AS A SERVICE' model shows a direct dependency from the 'APPLICATION' layer to the 'DATA' layer.

Deployment of ML Models In Heroku Using Flask

Debugging Errors

If your app's URL shows that some error has occurred, you can

- go to <https://dashboard.heroku.com/apps>
- Select your app
- Select More at the top right corner and click on View logs.



- Often the logs are helpful enough to debug where the problem lies.

Debugging Errors

If you figure out the error and want to correct it in the files, after correcting, just perform steps from `git add .` until `git push heroku master`.

Your app will get updated and possibly deployed successfully!

Congratulations! :D

Now that you're at the end of this unit, we hope you have actually been able to deploy the app.

What more can you do now?

Maybe try out your hand in CSS and make the app prettier.

Or try out how you can improve your model with hyperparameter tuning.

We've kept the projects by keeping in mind that not all might be familiar with Deep Learning. But Deep Learning + Model Deployment make some excellent projects you can create to make your portfolio shine!

References

- <https://blog.cambridgespark.com/deploying-a-machine-learning-model-to-the-web-725688b851c7>

Slide Download Link

You can download this unit from the below link:

<https://docs.google.com/presentation/d/1JLYDVZTOY15em9SJTv87xgnsvtXNcIXEdNCDLWXsLx0/edit?usp=sharing>

That's it for this unit. Thank you!

Feel free to post any queries on [Discuss](#).