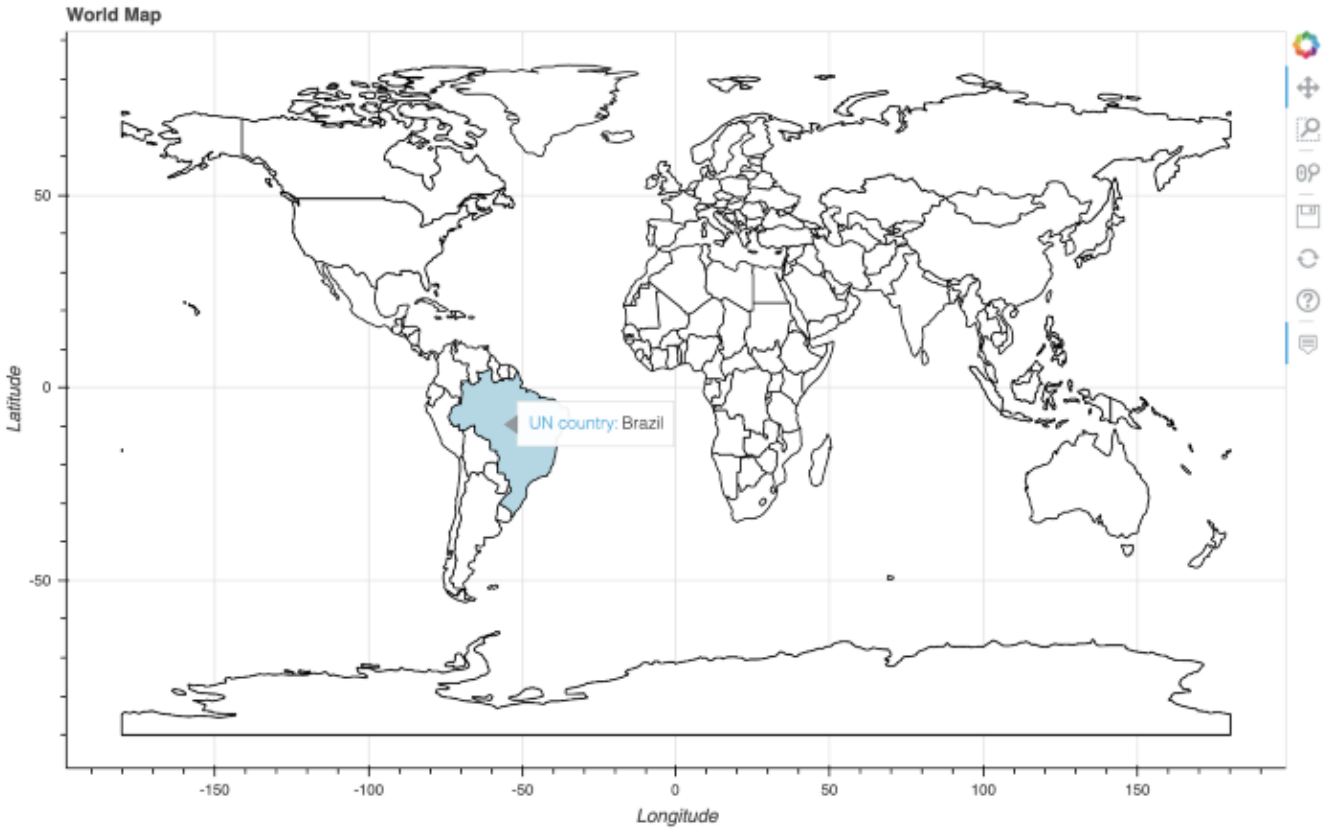
How to deploy a Bokeh app on Heroku

Jo Dorning
Mar 26 · 6 min read



Snapshot of our Bokeh web app

Deploying a data visualisation project to the web might seem daunting, especially to those without any prior experience of deploying apps. Documentation can be sparse and confusing, and researching tools and troubleshooting commands can take hours, if not days. We've been through the process, we know it's frustrating. So, to help the community avoid repeating the same learning curve over and over, we have created this simple, step by step tutorial. We hope this makes it easier for data scientists and others to share their visualisations — so they can spend more time creating, and less time frustrating!

Here we explain in simple terms how to deploy a Bokeh app on the Heroku platform. Bokeh, because it creates beautiful visualisations, and Heroku, because it's simple and free. Our very basic app is an interactive world map with a hover tool that shows

country names based on their United Nations (UN) definition. The code and data are available on Github and our deployed app can be viewed here.

As a bonus, we end with a short guide on how to embed an app into Medium.

Let's get started!

What is Bokeh?

Bokeh is an awesome data visualisation Python package that enables the creation of very elegant interactive graphics. For example, check out this TED Talk based visualisation demo.

What is Heroku?

Heroku is one of the easiest platforms for deploying web applications, entirely in the cloud. It is free to deploy simple demo applications and fairly cheap in general. It became popular within the Python community due to its simplicity. For instance, to deploy common frameworks like Flask, you only need to add one file to your project and use Heroku's simple command-line interface to deploy your application.

. . .

Setup

First we need to set up our app environment.

The file structure

Several files are required to ensure that Bokeh, Python and Heroku can communicate with each other and deploy the app. These are your data, a Python script containing the app, a Procfile, and a requirements file. A sensible file structure is shown below.

```
└── myapp_directory
    ├── data
    │   └── ...
    ├── myapp.py
    ├── Procfile
    └── requirements.txt
```

The App data

If you're deploying your own app you can skip this part, but to follow along with this tutorial you will need to download some data.

Our demo app is an interactive world map. We downloaded an open source geospatial dataset from Natural Earth to plot our map. In this dataset, the country names loosely match the United Nations' list of world regions. For this reason, in our visualisation, we label countries by their UN name.

Natural Earth data are available in two resolutions, 10m and 110m. We used the 110m dataset, because Bokeh was not able to render data at the finer scale of 10m.

You can download the 110m resolution data from here. Under the section titled `Admin 0 – Countries`, click `Download countries`. We used version 4.1.0, which was last updated on 2018–05–21 (according to the Changelog on Git).

The downloaded folder is a Shapefile and contains several component files. Each one contains a different layer of geographic information, and Bokeh needs them all to draw the world map.

The downloaded data should look like this:

Geospatial data files

The App Python file

Below is the code for our demo app, myapp.py. The code is also available on Github.

The basic structure is the same as any Python script, but rather than the command `show(p)` that is typically used to display Bokeh plots, your final command should be `curdoc().add_root(column(p))`. This will launch your plot in the browser.

The Procfile

This file tells Heroku where to source and host the app. It should have the following structure, where myapp is the custom app name.

```
web: bokeh serve — port=$PORT — allow-websocket-
origin=myapp.herokuapp.com — address=0.0.0.0 — use-xheaders myapp.py
```

`--port` specifies the port that the Bokeh server will listen on; `$PORT` is set by Heroku.

`--allow-websocket-origin=myapp.herokuapp.com` specifies your app as the host name.

`--address=0.0.0.0` tells Bokeh to figure out which IP address it will be on.

`--use-xheaders` tells Bokeh to override the remote IP and URI scheme/protocol.

The Procfile code snippet and argument definitions shown here are adapted from this helpful StackOverflow question.

The requirements file

The easiest way to make a requirements file is to set up a virtual environment in your app directory, install the packages you need to get your app running locally, and then use `pip freeze` to save the packages and their versions into a file.

To test your app locally, use the following terminal command from inside your app directory:

```
bokeh serve — show myapp.py
```

If successful, this will deploy the app to your localhost at http://localhost:5006/myapp.

To create your requirements file, use the following terminal command from inside your app directory:

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

You can also create a requirements file by hand, but it is much slower!

How to deploy the app

Now we have all the files we need, it's time to deploy! Here we show each step to execute. Make sure to `cd` into your app directory first.

1. Install requirements — if you haven't already.

```
pip install -r requirements.txt
```

2. Test the app locally.

```
bokeh serve — show myapp.py
```

3. Install the Heroku CLI. You can download it from https://devcenter.heroku.com/articles/heroku-cli, or use brew if on a Mac:

```
brew tap heroku/brew && brew install heroku
```

4. Go to the Heroku site and create a new account, if you don't have one already.

5. On Heroku, log in, go to the dashboard and create a new project. For new accounts you may also need to create a new SSH public key — just follow the prompts.

Then simply follow the instructions on your new project page to deploy your app. For convenience, these instructions are also shown below.

6. Connect your terminal to Heroku (this will open the browser for you to click login).

```
heroku login
```

7. Use `git clone` to add the Heroku source code for your project to your local machine.

```
heroku git:clone -a your-project
```

8. Add and commit your local code.

```
cd your-project
git add .
git commit -am "First commit"
```

9. Push your code to the Heroku remote (note, this is different to your Git remote). It will first install the requirements, refresh any code changes and finally build and deploy your app.

```
git push heroku master
```

Expected terminal output once the app is deployed

Success!

Once deployed successfully you will see a link to your app in the terminal. On click, this link will open the app in your browser.

Congratulations, you have just deployed your Bokeh app! From this point on, your app is accessible to anyone on the web — just share the herokuapp.com link!

You can view our deployed demo app here. Alternatively, scroll to the bottom of the page to view the embedded version inline.

Final notes

If you want to make changes to your app or refresh it with new data, just re-deploy it to Heroku using the same set of Heroku terminal commands:

```
heroku login
heroku git:clone -a your-project
cd your-project
git add .
git commit -am 'added new data'
git push heroku master
```

This update process could, potentially, be automated to refresh the app once a day, or each time new data are added to your data folder.

. . .

Thanks for reading!

I hope this tutorial helped simplify the deployment process and made it easier to share your beautiful Bokeh visualisations with the world. Please comment and share links to your apps!

. . .

Resources

Here are the key resources used to produce this tutorial.

- Github demo code
- Bokeh server documentation
- Heroku
- Natural Earth

. . .

Bonus content! How to include a Bokeh plot in a Medium post

We also found a way to embed interactive Bokeh plots into Medium. Note that this is only possible for static plots that don't require data refresh. If you want your plot to update when your code changes or you get new data, then you're better off deploying your app with Heroku.

Embedding a static plot doesn't require Git or Heroku. You simply save your Bokeh plot as a HTML file and embed a link created from the HTML code.

To create the embeddable link we use a handy tool called Codepen. Sign up and click Create Pen. Paste the HTML code for your plot into the HTML section and wait for your plot to render at the bottom of the page. Then copy the URL in the browser and embed it into your Medium post.

Here's ours — click Run Pen to reveal the plot.

Data Science Bokeh Deployment Web Applications Heroku