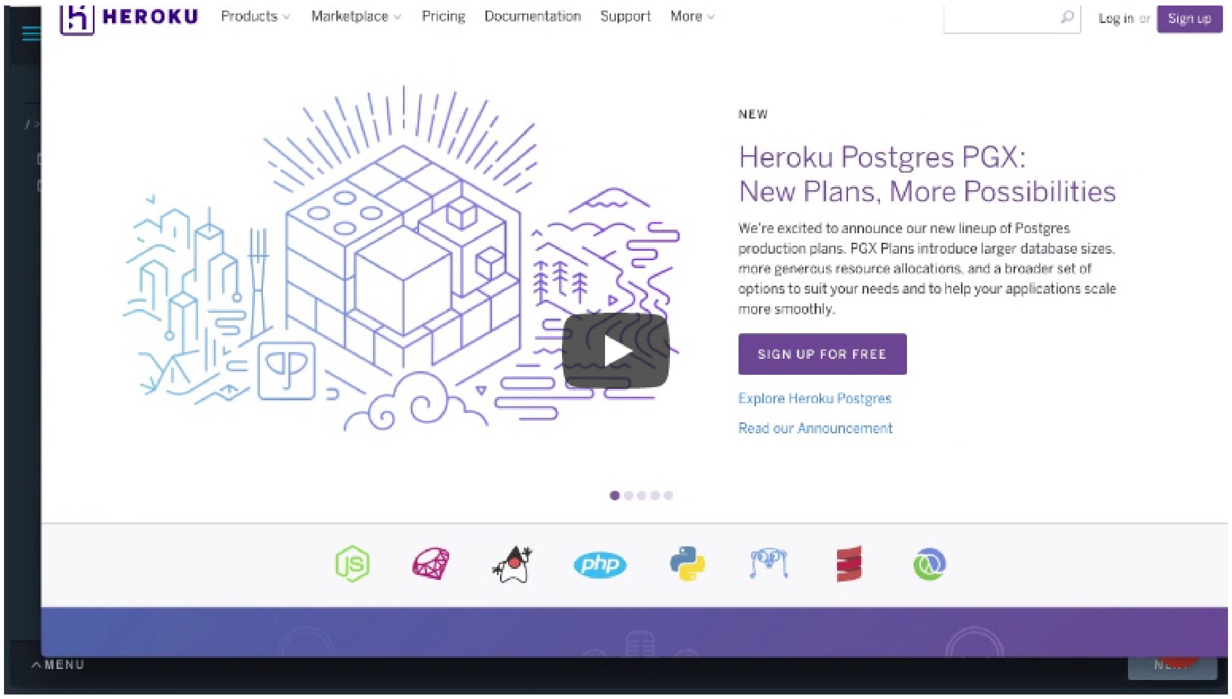
Deployment



NEW

Heroku Postgres PGX:
New Plans, More Possibilities

We're excited to announce our new lineup of Postgres production plans. PGX Plans introduce larger database sizes, more generous resource allocations, and a broader set of options to suit your needs and to help your applications scale more smoothly.

SIGN UP FOR FREE

Explore Heroku Postgres

Read our Announcement

Other Services Besides Heroku

Heroku is just one option of many for deploying a web app, and Heroku is actually owned by Salesforce.com.

The big internet companies offer similar services like Amazon's Lightsail, Microsoft's Azure, Google Cloud, and IBM Cloud (formerly IBM Bluemix). However, these services tend to require more configuration. Most of these also come with either a free tier or a limited free tier that expires after a certain amount of time.

Instructions Deploying from the Classroom

Here is the code used in the screencast to get the web app running:

First, a new folder was created for the web app and all of the web app folders and files were moved into the folder:

```
mkdir web_app
mv -t web_app data worldbankapp wrangling_scripts worldbank.py
```

The next step was to create a virtual environment and then activate the environment:

```
conda update python
python3 -m venv worldbankvenv
source worldbankenv/bin/activate
```

Then, pip install the Python libraries needed for the web app

```
pip install flask pandas plotly gunicorn
```

The next step was to install the heroku command line tools:

```
curl https://cli-assets.heroku.com/install-ubuntu.sh | sh
https://devcenter.heroku.com/articles/heroku-cli#standalone-installation
heroku —-version
```

And then log into heroku with the following command

```
heroku login
```

Heroku asks for your account email address and password, which you type into the terminal and press enter.

The next steps involved some housekeeping:

- remove `app.run()` from worldbank.py
- type `cd web_app` into the Terminal so that you are inside the folder with your web app code.

Then create a proc file, which tells Heroku what to do when starting your web app:

```
touch Procfile
```

Then open the Procfile and type:

```
web gunicorn worldbank:app
```

Next, create a requirements file, which lists all of the Python library that your app depends on:

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

And initialize a git repository and make a commit:

```
git init
git add .
git commit -m ‘first commit’
```

Now, create a heroku app:

```
heroku create my-app-name
```

where my-app-name is a unique name that nobody else on Heroku has already used.

The `heroku create` command should create a git repository on Heroku and a web address for accessing your web app. You can check that a remote repository was added to your git repository with the following terminal command:

```
git remote -v
```

Next, you need to push your git repository to the remote heroku repository with this command:

```
git push heroku master
```

Now, you can type your web app's address in the browser to see the results.

Virtual Environments vs. Anaconda

Virtual environments and Anaconda serve a very similar purpose. Anaconda is a distribution of Python (and the analytics language R) specifically for data science. Anaconda comes installed with a package and environment manager called conda. You can create separate environments using conda. However, these environments automatically come with Python packages meant for data science.

Virtual environments, on the other hand, come with the Python language but do not pre-install other packages.

The classroom workspace has many other Python libraries pre-installed including an installation of Anaconda.

When installing a web app to a server, you should only include the packages that are necessary for running your web app. Otherwise you'd be installing Python packages that you don't need.

To ensure that your app only installs necessary packages, you should create a **virtual Python environment**. A virtual Python environment is a separate Python installation on your computer that you can easily remove and won't interfere with your main Python installation.

There is more than one Python package that can set up virtual environments. In the past, you had to install these packages yourself. With Python 3.6, there is a virtual environment package that comes with the Python installation. The packaged is called venv

However, there is a bug with anaconda's 3.6 Python installation on a Linux system. So in order to use venv in the workspace classroom, you first need to update the Python installation as shown in the instructions above.

Creating a Virtual Environment in the Classroom

Open a terminal window in a workspace and type:

```
conda update python
```

When asked for confirmation, type y and hit enter. Your Python installation should update.

Next, make sure you are in the folder where you want to build your web app. In the classroom, the workspace folder is fine. But on your personal computer, you'll want to make a new folder. For example:

```
mkdir myapp
```

will create a new folder called myapp and `cd myapp` will change your current directory so that you are inside the myapp folder.

Then to create a virtual environment type:

```
python3 -m venv name
```

where name can be anything you want. You'll see a new folder appear in the workspace with your environment name.

Finally, to activate the virtual environment. Type:

```
source name/bin/activate
```

You can tell that your environment is activated because the name will show up in parenthesis on the left side of the terminal.

Creating a Virtual Environment Locally on Your Computer

You can develop your app using the classroom workspace. If you decide to develop your app locally on your computer, you should set up a virtual environment there as well. Different versions of Python have different ways of setting up virtual environments. Assuming you are using Python 3.6 and are on a linux or macOS system, then you should be able to set up a virtual environment on your local machine just by typing:

```
python3 -m venv name
```

and then to activate:

```
source name/bin/activate
```

On Windows, the command is;

```
c:\>c:\Python35\python -m venv c:\path\to\myenv
```

and to activate:

```
C:\> <venv>\Scripts\activate.bat
```

For more information, read through this link.

Databases for Your App

The web app in this lesson does not need a database. All of the data is stored in CSV files; however, it is possible to include a database as part of a Flask app. One common use case would be to store user login information such as username and password.

Flask is database agnostic meaning Flask can work with a number of different database types. If you are interested in learning about how to include a database as part of a Flask app, here are some resources:

- Flask Mega Tutorial
- Heroku - Provision a Database

Deployment

In the next part of the lesson, you'll find a workspace where you can practice deploying the world bank web app. Set up an account on Heroku and then follow the instructions shown in this part of the lesson.

You'll need to use a different name for the web app since the one used in this lesson is already taken.