https://www.udacity.com/wiki/ud330/deploy

Deploy Your Flask App to Heroku

Now that you've completed Full-Stack Foundations and Authentication & Authorization you have a web application that is ready to be shared on the Internet for others to see. In this tutorial, you will learn how to use Heroku to deploy your applications to the Internet. Heroku offers free services for deploying a website with basic features.

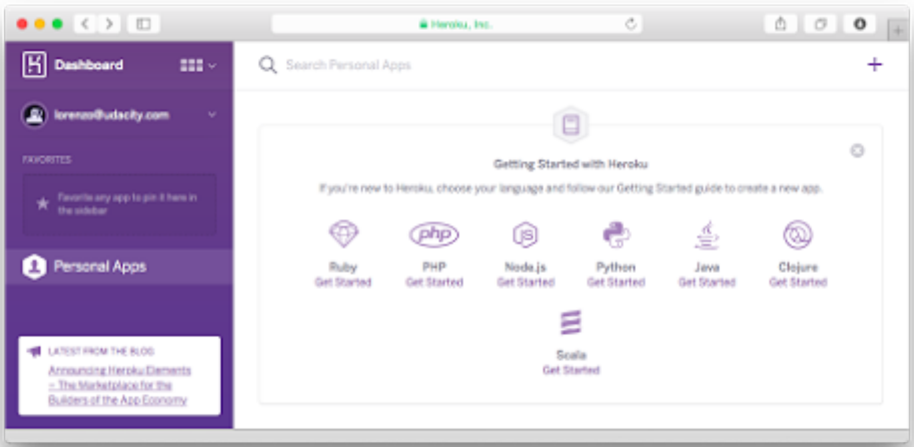
Step 1 - Make a Heroku Account

Go to heroku.com and follow the steps to create a free account.

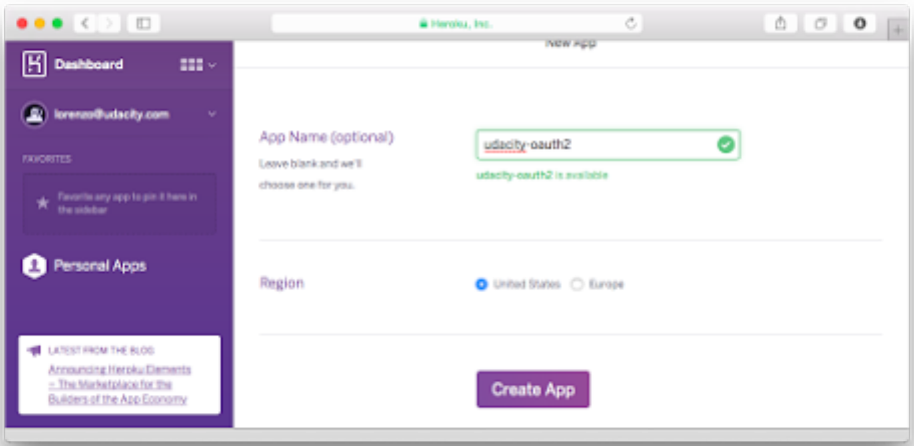
Heroku, Inc.
heroku
Focus on the app
Sign up for free
Powerful platform built by developers for developers
Focus on building great apps →
php
Start-up style app dev fitted for the enterprise
Learn more about Heroku Enterprise →

```
> git push heroku master
```

Once you’re logged in, click on the plus sign in the top right corner of your Heroku dashboard to create a new app.


Heroku, Inc.
Dashboard
lorenzo@udacity.com
FAVORITES
Favorite any app to pin it here in the sidebar
Personal Apps
LATEST FROM THE BLOG
Announcing Heroku Elements – The Marketplace for the Builders of the App Economy
Search Personal Apps
Getting Started with Heroku
If you're new to Heroku, choose your language and follow our Getting Started guide to create a new app.
Ruby Get Started
PHP Get Started
Node.js Get Started
Python Get Started
Java Get Started
Clojure Get Started
Scala Get Started

You can try and give your application a unique name or leave the field blank an Heroku will make an application name for you.


Heroku, Inc.
Dashboard
lorenzo@udacity.com
FAVORITES
Favorite any app to pin it here in the sidebar
Personal Apps
LATEST FROM THE BLOG
Announcing Heroku Elements – The Marketplace for the Builders of the App Economy
New App
Create App

App Name (optional): udacity-oauth2
Leave blank and we'll choose one for you.
udacity-oauth2 is available

Region: [✓ United States] Europe

Step 2 - Fire up your Vagrant VM

On your computer, fire up the vagrant machine you used to build your application using `vagrant up` and `vagrant ssh`. Using vagrant makes this tutorial much easier to follow for users on any operating system.

Step 3 - Install Git, Ruby & the Heroku Toolbelt

Heroku has a suite of features to simplify the deployment process. In order to take advantage of these features, we must install the Heroku toolbelt inside our vagrant environment. Ruby is needed in order to take install the Heroku toolbelt.

Install Ruby in your Vagrant VM in order to use the heroku toolbelt. copy and paste the code below line-by-line into your vagrant machine's terminal:

```
sudo apt-get install git
sudo apt-get install ruby-full
wget -qO- https://toolbelt.heroku.com/install-ubuntu.sh | sh
```

Step 4 - Prepare your App with Foreman

Before you proceed to Step 4, please know that as of 12th August, 2015 Heroku Toolbelt installs will not come with Foreman, but now that you have Ruby installed on your Vagrant VM you can simply run:

```
sudo gem install foreman
```

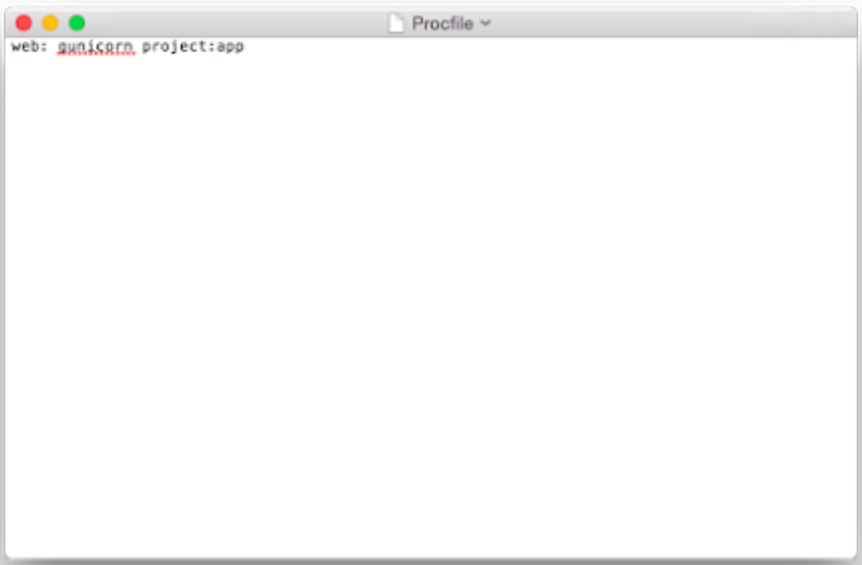
For more information on Foreman visit the Github page: https://github.com/ddollar/foreman

You can read about the Heroku Toolbelt update here - https://devcenter.heroku.com/changelog-items/692

Foreman is a locally-running web server that will find and run your web application almost identically to the way heroku will run your application. A few modifications to the restaurant menu app must be made before it can properly be run. Once you get your app up and running on Foreman, you know it's functionally ready to be deployed.

Making a Procfile

Foreman looks for a procfile in order to know what to run. Make a file called "Procfile" and make sure it contains the following information: `web: gunicorn project:app`



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

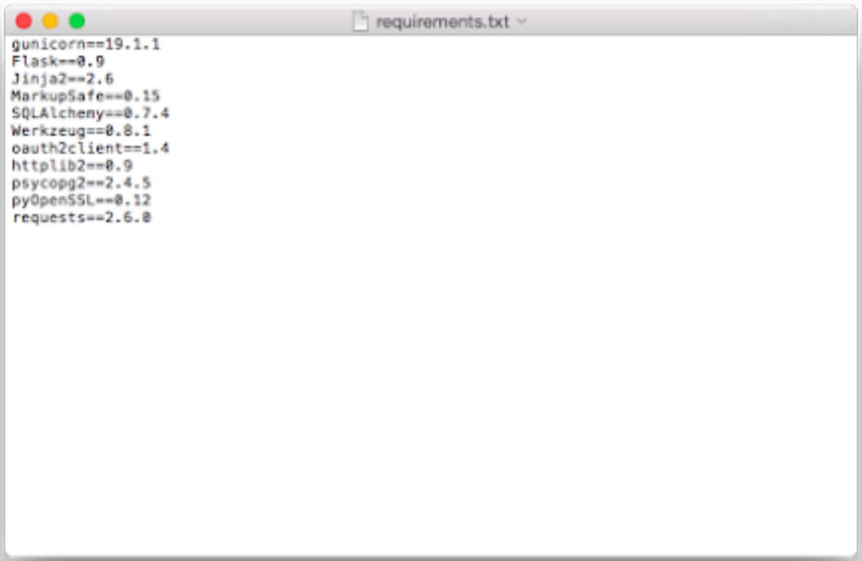
Gunicorn is a python-based web server that Heroku and Foreman will use to run your application. **Error! Filename not specified.**

Making Requirements.txt

Heroku recognizes an app as a Python app by the existence of a requirements.txt file in the root directory. For your own apps, you can use pip freeze to automatically create a requirements.txt file by running:

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

For consistency purposes make sure your requirements.txt file looks something like this if you are deploying the restaurant menu application:



```
gunicorn==19.1.1
Flask==0.9
Jinja2==2.6
MarkupSafe==0.15
SQLAlchemy==0.7.4
Werkzeug==0.8.1
oauth2client==1.4
httplib2==0.9
psycopg2==2.4.5
pyOpenSSL==0.12
requests==2.6.0
```

The requirements.txt file lists the app dependencies together with their versions. When an app is deployed, Heroku reads this file and installs the appropriate Python dependencies.

Run this command in your local directory to install the dependencies, preparing your system for running the app locally:

```
sudo pip install -r requirements.txt --allow-all-external
```

app.run Entry Point

Since Foreman will execute app.run, we no longer need call this explicitly in our code. In the last line of your project.py file, comment out the app.run line. Also comment out the if statement that checks for the __main__ parameter. remove the indentation of app.secret_key and app.debug. Your modified code should now look like this:

```
#if __name__ == '__main__':
app.secret_key = 'super_secret_key'
app.debug = True
#app.run(host = '0.0.0.0', port = 5000)
```

Step 5 - Create a Postgres Database

Since your application will now be open to the public, a more robust database solution is necessary. A free postgres database can be set up with Heroku as well. visit the Heroku add-on marketplace and add a free hobby dev postgres database to your application.

https://devcenter.heroku.com/articles/getting-started-with-python#provision-a-database

Once you have created your database you can select it from the heroku dashboard clicking on your app, then the database option, and then showing the URL.

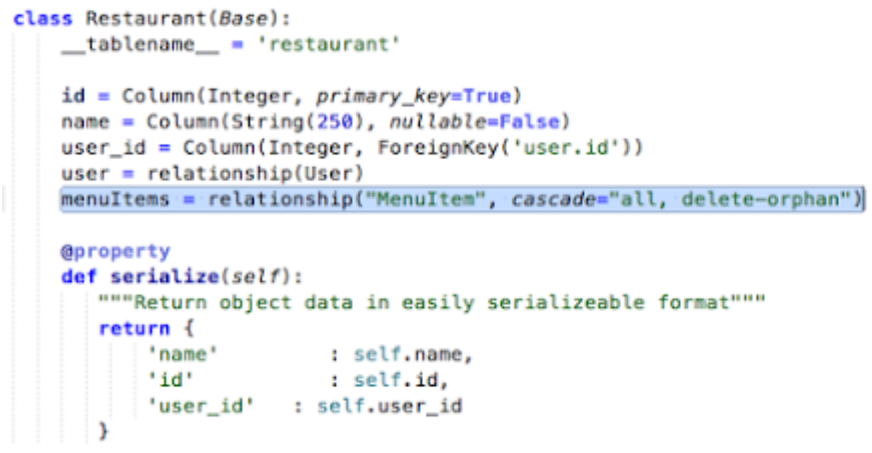
Database setup Copy the URL for you database and paste it everywhere you referred to the sqlite database in your code. This should be in the project.py, database_setup.py, and lotsofmenus.py files.

Error! Filename not specified.

Update database_setup.py, project.py, and lotsofmenus.py such that they point to the new postgres database.

Cascade on Delete

With sqlite you could delete a restaurant even if menu items were still present. Postgres does not allow a parent node to be deleted leaving the children node in the database. Modify your database_setup.py file as so such that if a restaurant is deleted all of its menu items are deleted as well using the cascades feature of SQLAlchemy.



```
class Restaurant(Base):
    __tablename__ = 'restaurant'

    id = Column(Integer, primary_key=True)
    name = Column(String(250), nullable=False)
    user_id = Column(Integer, ForeignKey('user.id'))
    user = relationship(User)
    menuItems = relationship("MenuItem", cascade="all, delete-orphan")

    @property
    def serialize(self):
        """Return object data in easily serializeable format"""
        return {
            'name'          : self.name,
            'id'            : self.id,
            'user_id'    : self.user_id
        }
```

Run database_setup.py

Execute database_setup.py from your command line. If everything worked properly you have successfully connected to your postgres database. Optionally you can execute lotsofmenus.py to populate the database with restaurant menu data.

Once all of these changes have been made, execute `foreman start web` from the vagrant machine to run the web server. Your web process loads on port 5000 because this is what Foreman provides as a default in the $PORT env var. It's important that your web process respect this value, since it's used by the Heroku platform when you deploy.

Open http://localhost:5000 and make sure the app is running properly in your browser.

Once you've tried out your app locally, you are ready to deploy to Heroku

Step 6 - Login to Heroku

Log in using the email address and password you used when creating your Heroku account:

```
$ heroku login

Enter your Heroku credentials.

Email: python@example.com

Password:

Authentication successful.
```

Point Heroku to the app you created.

for example, if you app was called 'hidden-valley-8790' you should write `git remote add heroku git@heroku.com:hidden-valley-8790.git`

Step 7 - Push changes to Heroku

```
git init
git add .
git commit -m“Initial Commit”
git push heroku master
```

(this may take a minute or two)

visit [your-application-name].herokuapp.com and check and see that your application is now live!

For more information:

Check out the getting started guide on Heroku for more information about deploying Python application.

Heroku also has a suite of add-ons that you can use to add more functionality to your application.

Papertrail is a great add-on for logging your application's activity.