6.170 Deployment Guide

At some point, you will want to deploy your applications on a server somewhere, so that all the world can access them! We'll ask you to deploy somewhere for testing by the TAs for grading purposes. You're free to deploy anywhere you want that leads to a publicly accessible web site, but we have guides for deploying through Heroku and OpenShift.

We recommend students use Heroku to deploy apps. However, using Heroku with Mongo will require adding a credit card to your account for verification purposes -- you will not be charged for Sandbox versions as explained in this guide. If you do not have a credit card, we recommend you to use OpenShift to deploy. Below are guides for both platforms.

Note: Make sure all your code (package.json, ./bin/www, app.js, etc.) is in the root directory of your project. In other words, your proj3 contents should be tests/, app.js, bin/, routes/, views/, etc., not tests/ and fritter/.

Heroku

Sign up and install Heroku

Create a Heroku account at https://signup.heroku.com/.

You will need to use MongoLabs (an add-on that runs MongoDB on your Heroku server) for Fritter Part 2, which requires adding a credit card to your billing account. Login and add a credit card by opening the dropdown menu next to your email and selecting "Manage Account" > "Billing". Note: You will not be billed for using the Sandbox version of MongoLabs, which is sufficient for the purposes of this class.

Follow the instructions here to install the Heroku CLI, which will allow you to deploy your app from the command line.

Set up your code

Define a Procfile in the root directory of your application. A Procfile is a text file that declares how Heroku should run your app. Make sure to name your Procfile exactly as "Procfile" (without extensions). The Procfile specifies which process type Heroku should run (web, worker, clock, etc.) as well as any commands in the following format:

```
<process type>: <command>
```

We will be using a web process type and want to start the server, which is either in ./bin/www or app.js. This is what your Procfile should contain if your server is started in ./bin/www (as in the
Demo Notes app and generated express app):

```bash
web: node ./bin/www
```

This is what your Procfile should look like if your server is started in `app.js`:

```bash
web: node app.js
```

If you used the Demo Notes app, you need to tell Heroku to use the environmental port, not an arbitrary port like 3000. Change this line (in `./bin/www`)

```javascript
app.listen(process.env.PORT || 3000);
```

For Project 4 (Fritter Part 2) only: Next, set up the MongoLab URI in `app.js`. The MongoLab URI will be stored in an environmental variable. The following code allows the app to connect to MongoDB both on the Heroku server as well as your local machine:

```javascript
var mongoose = require('mongoose');
// Connect to either the MONGOLAB_URI or to the local database.
mongoose.connect(process.env.MONGOLAB_URI || 'mongodb://localhost/mymongodb');
```

Create the Heroku app

Create a Heroku app using your command line. In the following example, `example-6170` is the name of the app. It needs to be a unique name, so try doing something like `-fritter` You need to create the app, add the MongoLab add-on (for Part 2), and add Heroku as a remote origin to your Git repo so you can redeploy.

```
heroku apps:create rswang-fritter
heroku addons:create mongolab    # only for Part 2
heroku git:remote -a rswang-fritter
```

Commit the changes from above.

```
git push heroku master
```
Scale your Heroku app to run on one dyno. **Note:** Scaling to more than one dyno is not free. One dyno will be sufficient for this class.

```
heroku ps:scale web=1
```

**Open your app**

You can open your app by running:

```
heroku open
```

If your app doesn't show and says "Application Error: An error occurred in the application and your page could not be served. Please try again in a few moments.", check the logs first:

```
heroku logs
```

Some common deployment issues are not including a required NPM module in `package.json`.

**Redeploying your app**

Whenever you want to redeploy your app, make sure your changes have been committed to your repo. Then, push the Heroku server, which will automatically deploy your app:

```
heroku push remote master
```

**OpenShift**

**Sign up and install OpenShift**

Register for OpenShift [here](#).

Install the CLI by following the instructions [here](#). Make sure you follow the instructions to set up your SSH key by running `rhc setup`.

**Set up your code**

Edit your `package.json` code to include instructions for OpenShift to run your app. Add the "scripts" object your `package.json` with the appropriate command.
The MongoDB URL, port, and IP address for the OpenShift app are environmental variables. In `./bin/www`, include these environmental variables. The following code allows the app to run both on the OpenShift server as well as your local machine:

```javascript
app.set('port', process.env.OPENSHIFT_NODEJS_PORT || process.env.PORT || 3000);
app.set('ipaddress', process.env.OPENSHIFT_NODEJS_IP || '127.0.0.1');
```

**For Project 4 (Fritter Part 2):** Set up MongoDB URL in `app.js`:

```javascript
mongoose.connect(process.env.OPENSHIFT_MONGODB_DB_URL ||
'mongodb://localhost:27017/mymongodb'); // connect to our database
```

**Create the OpenShift app**

**For Project 3 (Fritter Part 1):** Create a new application with the NodeJS cartridge:

```
rhc app-create example6170 nodejs-0.10
```

**For Project 4 (Fritter Part 2):** Create a new application with the NodeJS and MongoDB cartridges:

```
rhc app-create example6170 nodejs-0.10 mongodb-2.4
```

Upon completion, you should see the following:
Your application 'example6170' is now available.

URL:        http://example6170-jsmith.rhcloud.com/
SSH to:     55f7101c89f5cff153000056@example6170-jsmith.rhcloud.com
Git remote: ssh://55f7101c89f5cff153000056@example6170-jsmith.rhcloud.com/~/git/test3.git/
Cloned to:  /Users/jsmith/Documents/example6170

Copy the Git URL and add a remote repository to your project repo:

```
git remote add openshift ssh://55f70acb2d5271c52e000095@example6170-jsmith.rhcloud.com/~/git/example6170.git/
```

Then, push your project changes to the Openshift repository -- your app will automatically deploy. **Note:** We are force-pushing to overwrite the starter OpenShift code.

```
git push openshift master --force
```

You can view your app in your browser at http://<application>-<namespace>.rhcloud.com.

**Redeploying your app**

Commit your changes to your app and push to the remote origin:

```
git push remote openshift
```

The app will be updated and the OpenShift server will be restarted!