Thursday: Deploying to Heroku

Deploy Database apps to Heroku

Deploy Configuration Variables

We need to now configure our Heroku server for deployment.We have made a lot of changes since we last deployed. First we need to set configuration variables for our email username and password.

```
(virtual)$ heroku config:set MAIL_USERNAME=<YOUR EMAIL ADDRESS>
(virtual)$ heroku config:set MAIL_PASSWORD=<YOUR EMAIL PASSWORD>
```

Set up Postgres on Heroku

Next we want to set up a Postgres database on our Heroku server.

```
heroku addons:create heroku-postgresql
```

This database's URI will be stored in Heroku in the configuration variable `DATABASE_URL`. We then need to define this new Database URI inside our *config.py*

config.py

```
class ProdConfig(Config):
    SQLALCHEMY_DATABASE_URI = os.environ.get("DATABASE_URL")
```

We update the `ProdConfig` class by placing the production `SQLALCHEMY_DATABASE_URI` and setting it to the environment variable `DATABASE_URL`.

We then need to update the `create_app` function inside our *manage.py* file to access the new database URI.

Updating application for production

manage.py

```
..........
app = create_app('production')
..........
```

We pass in the `production` option from our `config_options` dictionary. This will change our app's configurations to the `ProdConfig` class.

Next we need to update our requirements.txt file to pick the new extensions and modules we have used.

```
(virtual)$ pip freeze > requirements.txt
```

Remember to remove the `pkg-resources==0.0.0` line to prevent throwing an error when deploying.

Pushing To Heroku

We can now add and commit our changes and push the application to Heroku.

```
(virtual)$ git push heroku master
```

We now need to define our database schema on our Heroku database.

```
(virtual)$ heroku run python3.6 manage.py db upgrade
```

This will create a database schema on Heroku. We can now visit the website to make sure everything works.