Deploy to Production

This part of the tutorial assumes you have a server that you want to deploy your application to. It gives an overview of how to create the distribution file and install it, but won't go into specifics about what server or software to use. You can set up a new environment on your development computer to try out the instructions below, but probably shouldn't use it for hosting a real public application. See Deployment Options for a list of many different ways to host your application.

Build and Install

When you want to deploy your application elsewhere, you build a distribution file. The current standard for Python distribution is the *wheel* format, with the `.whl` extension. Make sure the wheel library is installed first:

```
pip install wheel
```

Running `setup.py` with Python gives you a command line tool to issue build-related commands. The `bdist_wheel` command will build a wheel distribution file.

```
python setup.py bdist_wheel
```

You can find the file in `dist/flaskr-1.0.0-py3-none-any.whl`. The file name is the name of the project, the version, and some tags about the file can install.

Copy this file to another machine, set up a new virtualenv, then install the file with `pip`.

```
pip install flaskr-1.0.0-py3-none-any.whl
```

Pip will install your project along with its dependencies.

Since this is a different machine, you need to run `init-db` again to create the database in the instance folder.

```
export FLASK_APP=flaskr
flask init-db
```

When Flask detects that it's installed (not in editable mode), it uses a different directory for the instance folder. You can find it at `venv/var/flaskr-instance` instead.

Configure the Secret Key

In the beginning of the tutorial that you gave a default value for **`SECRET_KEY`**. This should be changed to some random bytes in production. Otherwise, attackers could use the public `'dev'` key to modify the session cookie, or anything else that uses the secret key.

You can use the following command to output a random secret key:

```
python -c 'import os; print(os.urandom(16))'

b'_5#y2L"F4Q8z\n\xec]/'
```

Create the `config.py` file in the instance folder, which the factory will read from if it exists. Copy the generated value into it.

`venv/var/flaskr-instance/config.py`

```
SECRET_KEY = b'_5#y2L"F4Q8z\n\xec]/'
```

You can also set any other necessary configuration here, although `SECRET_KEY` is the only one needed for Flaskr.

Run with a Production Server

When running publicly rather than in development, you should not use the built-in development server (`flask run`). The development server is provided by Werkzeug for convenience, but is not designed to be particularly efficient, stable, or secure.

Instead, use a production WSGI server. For example, to use Waitress, first install it in the virtual environment:

```
pip install waitress
```

You need to tell Waitress about your application, but it doesn't use `FLASK_APP` like `flask run` does. You need to tell it to import and call the application factory to get an application object.

```
waitress-serve --call 'flaskr:create_app'

Serving on http://0.0.0.0:8080
```

See Deployment Options for a list of many different ways to host your application. Waitress is just an example, chosen for the tutorial because it supports both Windows and Linux. There are many more WSGI servers and deployment options that you may choose for your project.

Continue to Keep Developing!.