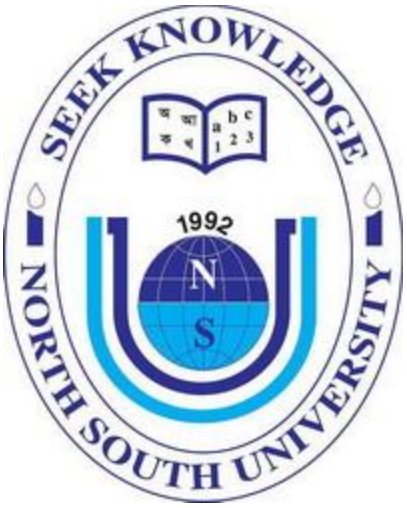


CSE 482

Section 1

Project Deployment Guideline for Online Contest Helper

Summer18

- Osman Ali Chowdhury 1320118042
- Pritom Roy 1430378042
- Mohammad Tanvir Bin Sattar-Mazumder 1321518042

Date: 11/09/18

Let us have an overview of Django and Heroku;

Django:

Django is an extremely popular and fully featured server-side web framework, written in Python. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. A programming framework is a toolkit of components needed to create a web or mobile application. Django is a more fully featured kit than most of other frameworks, it contains everything we need to build an app. Django adheres to D.R.Y. — Don't Repeat Yourself — philosophy. That means that the framework places a premium on getting the absolute most out of very little code. As a result, it supposes less hours to get it working, less code to break, and less to change when you need re-orientation.

Heroku:

Heroku is a cloud application platform, it is basically a Platform-as-a-Service (PaaS). They support several programming languages, including Python. It is very easy to deploy Django applications on Heroku. They also offer a free plan, which is quite limited, but it is great to get started and to host demos of Django applications. It also supports Java, Node.js, Scala, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it lets the developer build, run and scale applications in a similar manner across all the languages.

How deployment works on Heroku:

- The main content of the development are the source code, related dependencies if they exist, and a Procfile for the command.
- The application is sent to Heroku using either of the following: Git, GitHub, Dropbox, or via an API. Application development on Heroku is primarily done through git. The application gets a new git remote typically named as Heroku along with its local git repository where the application was made. Hence to deploy heroku application is similar to using the git push command. There are many other ways of deploying applications too. For example, developers can enable GitHub integration so that each new pull request is associated with its own new application, which enables all sorts of continuous integration scenarios. Deployment then, is about moving the application from a local system to Heroku
- There are packets which take the application along with all the dependencies, and the language runtime, and produce slugs. These are known as build-packs and are the means for the slug compilation process.
- A slug is a combination/bundle of the source code, built dependencies, the runtime, and compiled/generated output of the build system which is ready for execution.
- Next is the Config vars which contain the customizable configuration data that can be changed independently of the source code.
- Add-ons are third party, specialized, value-added cloud services that can be easily attached to an application, extending its functionality.
- A release is a combination of a slug (the application), config vars and add-ons.
- Heroku maintains a log known as the append-only ledger of releases the developer makes.

Deployment procedures:

First we signup in Heroku and download Heroku toolbelt and login via terminal. Code will be like;

```
$ heroku login
Enter your Heroku credentials.
Email: vitor@simpleisbetterthancomplex.com
Password (typing will be hidden):
Authentication successful.
```

- Add a **Procfile** in the project root;

```
web: gunicorn nameofDjangoProject.wsgi --log-file -
```

- Add **requirements.txt** file with all the requirements in the project root; Add **Gunicorn** to **requirements.txt**;

```
pip freeze > requirements.txt
Django==1.9.8
dj-database-url==0.3.0
dj-static==0.0.6
gunicorn==19.6.0
Unipath==1.0
python-decouple==3
Pillow==3.3.0
Markdown==2.6.6
bleach==1.4.3
psycopg2==2.6.1
whitenoise==3.2
```

- Configure the STATIC-related parameters on **settings.py**

```
PROJECT_ROOT = os.path.dirname(os.path.abspath(__file__))

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.9/howto/static-files/
STATIC_ROOT = os.path.join(PROJECT_ROOT, 'staticfiles')
STATIC_URL = '/static/'

# Extra places for collectstatic to find static files.
STATICFILES_DIRS = (
    os.path.join(PROJECT_ROOT, 'static'),
)
```

- Configure **whitenoise** to serve static files

```
pip install whitenoise
import os
from django.core.wsgi import get_wsgi_application
from whitenoise.django import DjangoWhiteNoise

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "bootcamp.settings")

application = get_wsgi_application()
application = DjangoWhiteNoise(application)
STATICFILES_STORAGE = 'whitenoise.django.GzipManifestStaticFilesStorage'
```

Now we have configured the system and proceeding for deployment;

- First, we have to clone the repository we want to deploy

```
git clone https://github.com/vitorfs/bootcamp.git && cd bootcamp
```

- Then login to Heroku using the toolbelt

```
heroku login
```

- Inside the project root, we create a Heroku App

```
heroku create demo-bootcamp
```

- Output would generate at link with its own repo.

```
heroku addons:create heroku-postgresql:hobby-dev
Creating ⬢ demo-bootcamp... done
https://demo-bootcamp.herokuapp.com/ | https://git.heroku.com/demo-
bootcamp.git
```

- Then we add a PostgreSQL database to our app

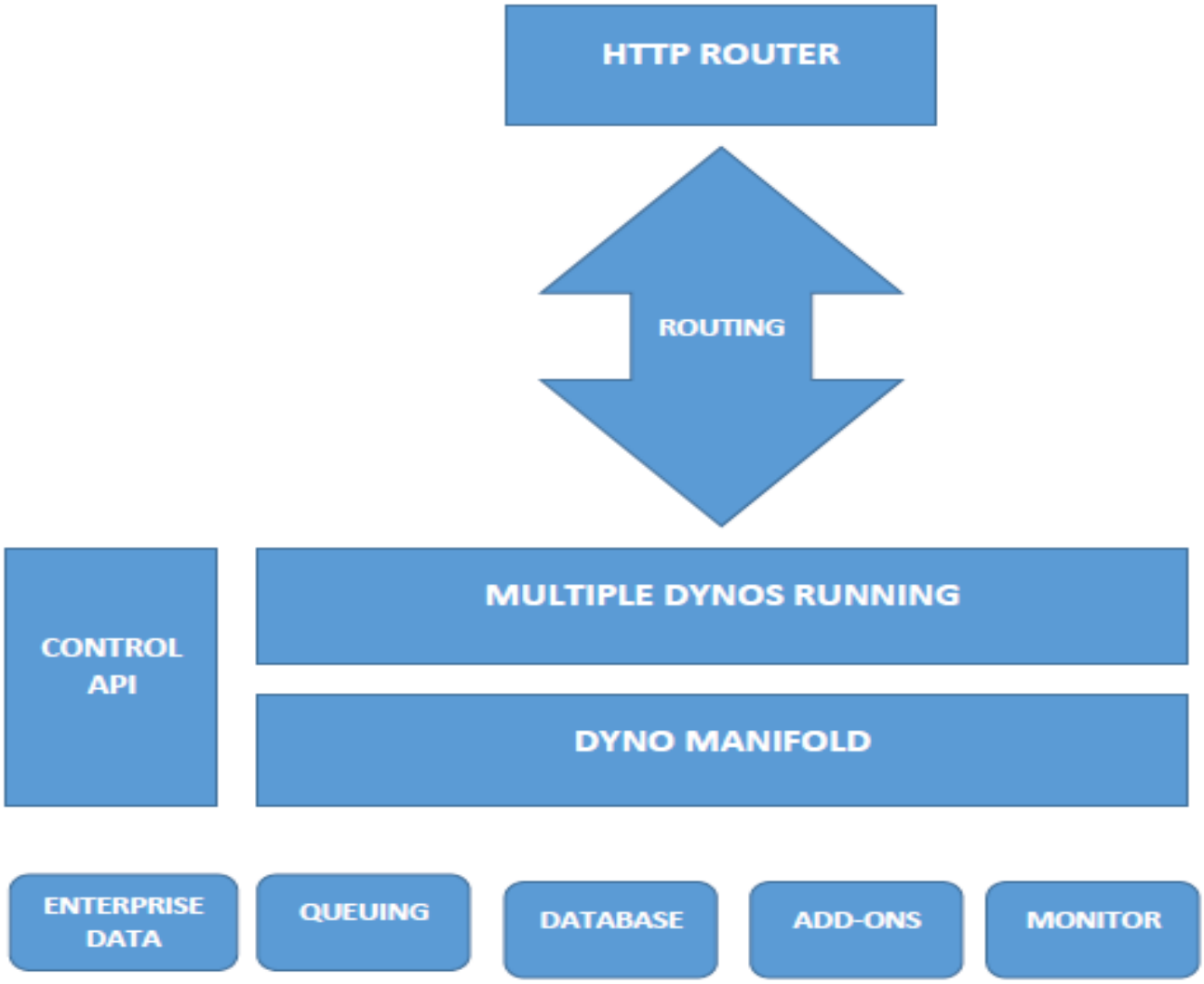
```
Creating postgresql-metric-21979... done, (free)
Adding postgresql-metric-21979 to demo-bootcamp... done
Setting DATABASE_URL and restarting demo-bootcamp... done, v4
Database has been created and is available
```

- Output will create database. We can omit the **app name**, then Heroku will pick a name for us. Better we provide our own app name.

- Now we login to Heroku Dashboard and access our recently created app.
- We click on the **Settings** menu and then on the button **Reveal Config Vars.**
- Now we need to add all the Environment variables. Here is where **python-decouple** and **dj-database-url** are handy.

- The **DEBUG** var defaults to False, and **DATABASE_URL** is automatically added by Heroku upon PostgreSQL database installation. So in this case we will only need to add the **SECRET_KEY** and Push to deploy.
- Finally migrate the database and have output as link given.

Working Flow of Heroku:



------Thank You------