Table of Contents

Deploy Databases

Deploy Neo4j to graphenedb Platform

Prepare Data of Neo4j

- To start neo4j, issue the following command in a system shell and make sure the site http://localhost:7474/browser/ to visit normal:

```
$ neo4j console
```

- To compress database file -- data.db, issue the following command in a system shell:

```
$ tar -cjf data.tar.bz2 %YOUR_NEO4J_DB_PATH%/neo4j-community/data/data.db
```

Import the Compress File to graphededb

- To register a graphededb account by your email
- Click the 'Create database' button
- Check to the 'Admin' tab and click 'Restore database' button, then click 'upload file', choose the compress file

'data.tar.bz', and click 'Restore' waiting for progress 100%

- Check to 'Connection' Tab and you will see the REST URL, REST USERNAME and REST PASSWORD

Deploy Mongodb to mongolab

Prepare an Available mongolab Account

- To register a mongolab account and login in
- Click `Create new` button
- Choose a free cloud provider, fill in the `database name` in the form and cilck the `create new MongoDB deployment` button
- Check to the `Users` tab and click the `Add database user` button, fill in your usename and password

- Check to the `Tools` tab and find which data type you want import, you will see the command follow:

```
 Prepare a available json file
Import collection
% mongoimport -h <Your own mongolab REST URL> -d <databasename> -c <collection> -u <user> -p <passwo
Export collection
% mongoexport -h <Your own mongolab REST URL> -d <databasename> -c <collection> -u <user> -p <passwo
```

```
 Prepare a available csv file
Import collection
% mongoimport -h <Your own mongolab REST URL> -d <databasename> -c <collection> -u <user> -p <passwo
```

```
Export collection
% mongoexport -h <Your own mongolab REST URL> -d <databasename> -c <collection> -u <user> -p <passwo
```

Note: : ds051645.mongolab.com:51645
: the name of database created by step 3
: the name of collection you will create
-u -p : the account of created by step 4
<input file>: the name of import file

Deploy Source Code to Heroku

Set Up

Hang on for a few more minutes to learn how it all works, so you can make the most out of Heroku.
The tutorial assumes that you have:

- Ceate a free Heroku account
- Run the application locally successfully

In this step you will install the Heroku Toolbelt. This provides you access to the Heroku Command Line Interface (CLI), which can be used for managing and scaling your applications and add-ons. A key part of the toolbelt is the heroku local command, which can help in running your applications locally.
Once installed, you can use the heroku command from your command shell.

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:
...
```

Authenticating is required to allow both the heroku and git commands to operate.

Prepare the App

In this step, you will prepare a simple application that can be deployed. There are two ways to structure the application, we would introduce them deperately:

Case1: Clone Heroku Repo

Execute the following commands to clone the sample application：

```
 $ git clone <HEROKU-DEMO-REPO-PATH>
$ cd python-getting-started
```

You now have a functioning git repository that contains a simple application as well as a requirements.txt file.

If you wanna to deploy a python application, should be replaced with the demo application path,for example:https://github.com/heroku/python-getting-started.git, for all optional applications please visit here.

Case2: Use Local Repo

Assume you already have a repo in local and it can ran successfully, what you need is to define you Procfile and bind the heroku path in next step. Heroku would find the Procfile in application's root directory as the boot script.

**Example: Python Application's Configuration: **

/Procfile

```
web: gunicorn run:app --log-file -
```

/run.py

```
from app import app
```

/app/__init__.pypy

```
#...

from flask import Flask
```

```
app = Flask(__name__, static_url_path='')
app.config.from_object('config')

#...
```

*Example: Node.js Application's Configuration: *

/Procfile

```
 web: node web-server.js
```

/web-server.js

```
 var express = require('express');
var app = express();

app.set('port', (process.env.PORT || 8000));
app.use(express.static(__dirname));

app.get('/', function(request, response) {
   response.sendfile('index.html', {root: __dirname })
});

app.listen(app.get('port'), function() {
  console.log("Node app is running at localhost:" + app.get('port'));
```

```
});
```

Note: as the web-server.js deplends on express package, make sure you have config it in your package.json.

Deploy the App

In this step you will deploy the app to Heroku.
Create an app on Heroku, which prepares Heroku to receive your source code:

```
 $  heroku create
Creating lit-bastion-5032 in organization heroku... done, stack is cedar-14
http://lit-bastion-5032.herokuapp.com/ | https://git.heroku.com/lit-bastion-5032.git
Git remote heroku added
```

When you create an app, a git remote (called heroku) is also created and associated with your local git repository. Heroku generates a random name (in this case lit-bastion-5032) for your app, or you can pass a parameter to specify your own app name.
Now deploy your code:

```
 $ git push heroku master
Counting objects: 196, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (99/99), done.
```

```
Writing objects: 100% (196/196), 33.21 KiB | 0 bytes/s, done.
Total 196 (delta 82), reused 196 (delta 82)
Compressing source files... done.
Building source:

-----> Python app detected
-----> Installing runtime (python-2.7.11)
-----> Installing dependencies with pip
   Collecting dj-database-url==0.3.0 (from -r requirements.txt (line 1))
   Downloading dj_database_url-0.3.0-py2.py3-none-any.whl
   Collecting Django==1.9.1 (from -r requirements.txt (line 2))
   Downloading Django-1.9.1-py2.py3-none-any.whl (6.6MB)
   Collecting gunicorn==19.4.5 (from -r requirements.txt (line 3))
   Downloading gunicorn-19.4.5-py2.py3-none-any.whl (112kB)
   Collecting psycopg2==2.6.1 (from -r requirements.txt (line 4))
   Downloading psycopg2-2.6.1.tar.gz (371kB)
   Collecting whitenoise==2.0.6 (from -r requirements.txt (line 5))
   Downloading whitenoise-2.0.6-py2.py3-none-any.whl
   Installing collected packages: dj-database-url, Django, gunicorn, psycopg2, whitenoise
   Running setup.py install for psycopg2
   Successfully installed Django-1.9.1 dj-database-url-0.3.0 gunicorn-19.4.5 psycopg2-2.6.1 whitenoise-2

-----> Preparing static assets
   Running collectstatic...
   ...
   58 static files copied to '/app/staticfiles', 58 post-processed.

-----> Discovering process types
   Procfile declares types -> web

-----> Compressing... done, 44.1MB
-----> Launching...
   Released v4
```

```
    http://lit-bastion-5032.herokuapp.com/ deployed to Heroku

To git@heroku.com:lit-bastion-5032.git
 * [new branch]      master -> master
```

While the deployment is happening, you may see a syntax error during the install for gunicorn about invalid syntax for the line yield from self.wsgi.close(). That error can be ignored.
The application is now deployed. Ensure that at least one instance of the app is running:

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

Now visit the app at the URL generated by its app name. As a handy shortcut, you can open the website as follows:

```
$  heroku open
```

View Logs

Heroku treats logs as streams of time-ordered events aggregated from the output streams of all your app and Heroku components, providing a single channel for all of the events.
View information about your running app using one of the logging commands, heroku logs:

```
 heroku logs --tail
2014-08-15T15:17:55.780361+00:00 app[web.1]: 2014-08-15 15:17:55 [2] [INFO] Listening at: http://0.0.0.0
2014-08-15T15:17:55.780488+00:00 app[web.1]: 2014-08-15 15:17:55 [2] [INFO] Using worker: sync
2014-08-15T15:17:55.830489+00:00 app[web.1]: 2014-08-15 15:17:55 [7] [INFO] Booting worker with pid: 7
2014-08-15T15:17:55.779494+00:00 app[web.1]: 2014-08-15 15:17:55 [2] [INFO] Starting gunicorn 19.0.0
2014-08-15T15:17:56.321151+00:00 heroku[web.1]: State changed from starting to up
2014-08-15T15:17:57.847806+00:00 heroku[router]: at=info method=GET path="/" host=lit-bastion-5032.herok
```

Visit your application in the browser again, and you'll see another log message generated.
Press `Control+C` to stop streaming the logs.

Example: Brandgraph Deploy

Deploy Brandgraph Database: neo4j

Prepare Compress File of Brandgraph Database

- To start neo4j, issue the following command in a system shell and make sure the site http://localhost:7474/browser/ to visit normal:

```
$ neo4j console
```

- To compress database file -- graph.db, issue the following command in a system shell:

```
$ tar -cjf graph.tar.bz2 %YOUR_NEO4J_DB_PATH%/neo4j-community-2.1.6/data/graph.db
```

Note：%YOUR_NEO4J_DB_PATH% is the address of your personal computer

Import the Compress File to graphededb

- Login in the site http://app.graphenedb.com/ by following account：

  username: pwc.aia@gmail.com

  password: Pwcwelcome1

- Click the `Create database` button：
- Check to `Admin` Tab, click `Restore database` button, click `upload file`, choose the above bz2 file and click `Restore` button, waiting for progress 100%
- Check to `Connection` Tab, and you will see the REST URL, REST USERNAME and REST PASSWORD

Deploy Brandgraph API to Heroku

- Make sure have the clone of brandgrph code, else issue the following command in a system shell:

```
$ git clone https://github.com/Analytics-Innovation-Incubator/brand_graph.git
```

- Check to the server folder with the following command：

```
$ cd server
```

- Replace the deployed address of neo4j database in `config.py' file

```
NEO4J_DB_ADDR = "http://BrandGraph:rQTeP3MmZHsoq37MipMw@brandgraph.sb06.stations.graphenedb.com:247
```

- Modify details of requirements.txt, add following script:

```
gunicorn==19.2.1
```

- Define a procfile，and add the following script：

```
web: gunicorn run:app --log-file -
```

Note: run is the name of starting file of API

- Login in heroku with the following account:

```
 $ heroku login
Email: pwc.aia@gmail.com
Password (typing will be hidden): Pwcwelcome1
```

- Create an app on Heroku, which prepares Heroku to receive your source code

```
 $ heroku create %APP_NAME%
```

`%APP_NAME%` can be brandgraphapi or brandgraphapidev

```
 $ git init
$ git remote add heroku %HEROKU_REPO_URL%
```

%HEROKU_REPO_URL% is create by above `https://git.heroku.com/brandgraphapidev.git` or `https://git.heroku.com/brandgraphapidev.git`.

- Deploy API

```
 $ git add .
$ git commit -m 'create remote repository'
$ git push heroku master
```

- Test after deploy

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

view logs if have any issues：

```
 $ heroku logs --tail
```

- Errors may occur

```
  Error: unsupported locale setting
```

Search keywords `locale` by command `COMMOND+SHIFT+F` and remove the it

Deploy Brandgraph Web to Heroku

- Check to the web folder by following command:

```
$ cd web
```

- Define a procfile，and add the following script：

```
web: node web-server.js
```

Note: web-server.js is starting file of web

- modify details of package.json：

```
"express": "4.13.3"
```

- Login in heroku by the following account:

```
 $ heroku login
Email: pwc.aia@gmail.com
Password (typing will be hidden): Pwcwelcome1
```

- Create an app on Heroku, which prepares Heroku to receive your source code:

```
$ heroku create %APP_NAME%
```

`%APP_NAME%` is brandgraph or brandgraphdev

```
$ git init
$ git remote add heroku %HEROKU_REPO_URL%
```

%HEROKU_REPO_URL% is create by above `https://git.heroku.com/brandgraphdev.git` or `https://git.heroku.com/brandgraphdev.git`.

- Deploy

```
$ git add .
$ git commit -m 'create remote repository'
$ git push heroku master
```

- Test URL after deploy

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

Reference

- Getting Started with Python on Heroku
- Getting Started with Node.js on Heroku