Deployment Document

Prakash Rai
Team #02 Unravel

November 20, 2017

Unravel is written in Python, HTML, CSS and JavaScript. It uses Django as app server, nginx as web server and PostgreSQL as database. A starting point for installing and setting up nginx, django and PostgreSQL can be found on `https://nginx.org/en/docs/`, `https://docs.djangoproject.com/en/1.11/topics/install/` and `https://www.postgresql.org/docs/` respectively. Currently, unravel is deployed on Amazon Web Services(AWS). A step by step guide on how to deploy unravel on AWS can be found on `docs.aws.amazon.com/gettingstarted/latest/deploy/overview.html`.

0.1 Configuration

0.1.1 Setting Environmental Variables

For running unravel, it is important to set some environmental variables. They are:

- **PRODUCTION**: This environment variable is used to distinguish between production and development environment. This is useful because production environment needs to be more secure and efficient than development environment. Can take two values, "True" or "False".
- **DB_USER**: This environment variable contains username of database administrator.
- **DB_PASS**: This environment variable contains password to connect to database.
- **DB_NAME**: This environment variable contains name of the database.
- **SECRET_KEY**: This environment variable contains secret key required to generate CSRF tokens in django.

Environment variable in any GNU/Linux OS, which contains Bourne Again Shell (BASH) can be set as follows.

1. Open *.bashrc* file by using *vim*[1] */home/<username>/.bashrc* command.
2. Append export <ENVIRONMENT_VARIABLE_NAME> = <VALUE>
3. Close the file.
4. Restart bash or type *source /home/<username>/.bashrc* in terminal.

[1] other file editors can be used too.

0.1.2 Serving static files

Static files can be served by using Django in-built *serve_static* method, but it is very inefficient and is only meant for development purpose. For serving static files on production server, we will use nginx. A step by step guide for configuring nginx is given below.[2] A screenshot of current nginx configuration is also attached.

1. Inside unravel project directory, run *python manage.py collectstatic* command. It collects all static files used in the project and places them in a folder named *static* in project directory.
2. Go to */etc/nginx/conf.d* directory.
3. Create a file unravel.conf.
4. Configure listen port to 80.
5. Create an alias for */static*. Reroute all requests for */static* to *$PROJECT_DIR/static*
6. Reroute all request for port on which django is running [3] to port 80.
7. Finally, close the file and run *sudo service nginx restart* to restart nginx.

```
1 server {
2     listen 80 default_server;
3
4     server_name 18.217.40.45;
5     location /static {
6         alias  /home/ubuntu/unravel/static;
7     }
8     location / {
9         proxy_pass         http://127.0.0.1:8000;
10    }
11 }
```

Figure 1: Nginx configuration

[2]Assumption: Nginx is installed and running smoothly.
[3]default port is 8000

0.1.3 Running the app after deployment

After nginx is configured to serve static file, only thing left is to configure and run our app server. A step by step guide for running the app server is given below.

- Go to project directory.
- Run *python manage.py makemigrations*. This command scans through whole project and for changes in existing database models or addition of new models. It, then generates a migration file which contains necessary details to modify the DDL of existing schema.
- Run *python manange.py migrate*. This commands reads the migration file created in above step and changes the schema accordingly.
- Run *python manage.py runserver 0.0.0.0:8000*. It is used to run app server on given IP address and port number.

Unravel is up and running. Open your browser and type <IP>:<port> to access the web app.