EC2 Django Deployment – Step-by-Step

Jeffrey L. Eppinger
Carnegie Mellon University
Updated February 26, 2018

There are probably many ways to configure an EC2 instance to deploy your Django app, but since many people seem to be having trouble finding a consistent set of tutorials, here's a step-by-step guide, at least of how I do it. In this example, I deploy the `image-example` from class.

Provision the EC2 Instance

Go to `aws.amazon.com`:

- Create account
- Create an EC2 instance
  - Choose Ubuntu (current version is 16.04 as of this writing)
  - T2.micro (whichever size is free)
  - Review & Launch
  - Create Key Pair, download as <name>
- Note: the EC2 console can be used to stop/start your EC2 instance, to find its IP address and its security group, etc.

Connect to your instance with SSH:

- On MAC: `chmod 400 <name>.pem`
- `ssh –i <name>.pem ubuntu@<ip-address>`

Install Django, Test Sample App

In shell on EC2 Instance:

- `sudo apt-get update`
- `sudo apt-get dist-upgrade` ← Type <tab>, <space> to keep local grub-pc
- `sudo apt-get install python3-pip`
- `sudo -H pip3 install --upgrade pip`
- `sudo -H pip3 install django==1.11` ← Note: the double equals
- `sudo reboot`

Allow network access to port 8000

While it's rebooting, in EC2 Console (aws.amazon.com):

- Select your server instance
- Note your instance's security group
- Select Security Groups → <your security group>
    - Click "Inbound" Tab
    - Click Edit → Add Rule
    - Add Custom TCP Rule for Port 8000 (remove IPv6 by deleting "`; ::/0`")
    - Save

Note: it can take several minutes (up to five) for AWS to enable port 8000

Install Test Application

In shell on EC2 Instance:

- `git clone https://github.com/CMU-Web-Application-Development/image-example.git`

- `cd image-example`

(Please note that I pushed an update, since Lecture #10, so that the image example now uses `MEDIA_ROOT` to set the location of where images are stored.)

Then in shell on EC2 Instance, <edit> `/etc/hosts` using `vim` or `Emacs`, etc. The vim editor is already installed. Here's a nice quick reference: http://vim.rtorr.com.

- `python3 manage.py migrate`
- <edit> `webapps/settings.py`
    - Add your IP address to `ALLOWED_HOSTS` list
    - Save the file and exit the editor
- `python3 manage.py runserver 0.0.0.0:8000`

In web browser, visit `http://<ip-address>:8000`

- You should see the class example running, using SQLite for the DB

Install Apache HTTP Server

In shell on EC2 Instance:

- `sudo apt-get install apache2`
- `sudo apt-get install libapache2-mod-wsgi-py3`

In EC2 Console:

- Select your security group
  - Click Edit
  - Change the Custom TCP Rule to HTTP (to enable port 80 instead)
  - Save

In web browser, visit `http://<ip-address>`

- You should see the Apache splash screen

Configure Apache to Serve Django App

In shell on EC2 Instance, edit the Apache config file using emacs or vim or some other editor:

- `sudo <edit> /etc/apache2/apache2.conf`
  - Comment out default mapping for "/" url – it's around line 153:

```
#<Directory />
#   Options FollowSymLinks
#   AllowOverride None
#   Require all denied
#</Directory>
```

  - Insert alias for "/" url:

```
WSGIScriptAlias / /home/ubuntu/image-example/webapps/wsgi.py
WSGIPythonPath /home/ubuntu/image-example
```

  - Add permissions for example project directory:

```
<Directory /home/ubuntu/image-example>
    <Files wsgi.py>
        Require all granted
    </Files>
</Directory>
```

  - Save the file

- In shell on EC2 Instance, fix permissions on the directories and files and restart Apache:
    - `cd ~`
    - `sudo chgrp –R www-data image-example`
    - `chmod –R g+w image-example`
    - `sudo apache2ctl restart`

In web browser, visit `http://<ip-address>`

- You should now see example app running under Apache but static files are not working

A note on file permissions: The Apache server handles requests in processes running under a special user called `www-data`. The `chgrp` command, above, puts all your Django project's files into the `www-data` group, allowing Apache access. The `chmod` command gives write permission to the group. If you did not create an image when testing your app running on port 8000 with the development environment, you will need to create the media folder and then run the `chgrp` and `chmod` commands on the media folder. (The commands above apply to the media folder because the example creates the media folder in the project directory.)

A note on debugging: The Apache server will print errors out in a file called `/var/log/apache2/error.log`. Also, any print statements you've put in your Python code will show up in this error log.

Configure Apache to Serve Static Files

In shell on EC2 Instance, edit the Apache config file using emacs or vim or some other editor:

- `sudo <edit> /etc/apache2/apache2.conf`
    - Add alias and permissions for static folder:

```
Alias /static /home/ubuntu/image-example/picture_list/static

<Directory /home/ubuntu/image-example/picture_list/static>
    Order allow,deny
    Allow from all
</Directory>
```

- `sudo apache2ctl restart`

In web browser, visit `http://<ip-address>`

- Static files should now work

Install and Configure MySQL

In shell on EC2 Instance:

- `sudo apt-get install mysql-server` ← When prompted, set password
- `sudo apt-get install python-dev libmysqlclient-dev`
- `sudo -H pip3 install mysqlclient`

- `mysql -u root -p`
    - `mysql> create database django;`
    - `mysql> quit;`

- `cd image-example`
- `<edit> image-example/webapps/settings.py`
    - Change DB config to use MySQL:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'django',
        'USER': 'root',
        'PASSWORD': '<your password>',
    }
}
```

- `Python3 manage.py migrate`

- `sudo apache2ctl restart`

In web browser, visit `http://<ip-address>`

- Model data should now be stored in the database.

To view the data, in the shell on the EC2 Instance:

- `mysql -u root -p`
    - `mysql> use django;`
    - `mysql> show tables;`
    - `mysql> select * from picture_list_item;`
    - `mysql> quit;`

Additional notes:

Many people have migration problems that they only discover when they try to rebuild their database. You may need to remove your old migrations and re-make new ones. If you have unexplainable error messages from the Django ORM, whether in the cloud or locally, try this sequence of commands to completely start from a clean DB with new migrations is:

- `mysql -u root -p`
    - `mysql> drop database django;`
    - `mysql> create database django;`
    - `mysql> quit;`

- Then in the shell:
    - `cd ~/image-example`
    - `rm picture_list/migrations/0*`
    - `python3 manage.py makemigrations`
    - `python3 manage.py migrate`

Note: If your DB is in SQLite, you can completely reset your migrations this way:

- `cd ~/image-example`
- `rm db.sqlite3`
- `rm picture_list/migrations/0*`
- `python3 manage.py makemigrations`
- `python3 manage.py migrate`