Deploy Django

Ubuntu 18.04 - Apache - Django - MySQL

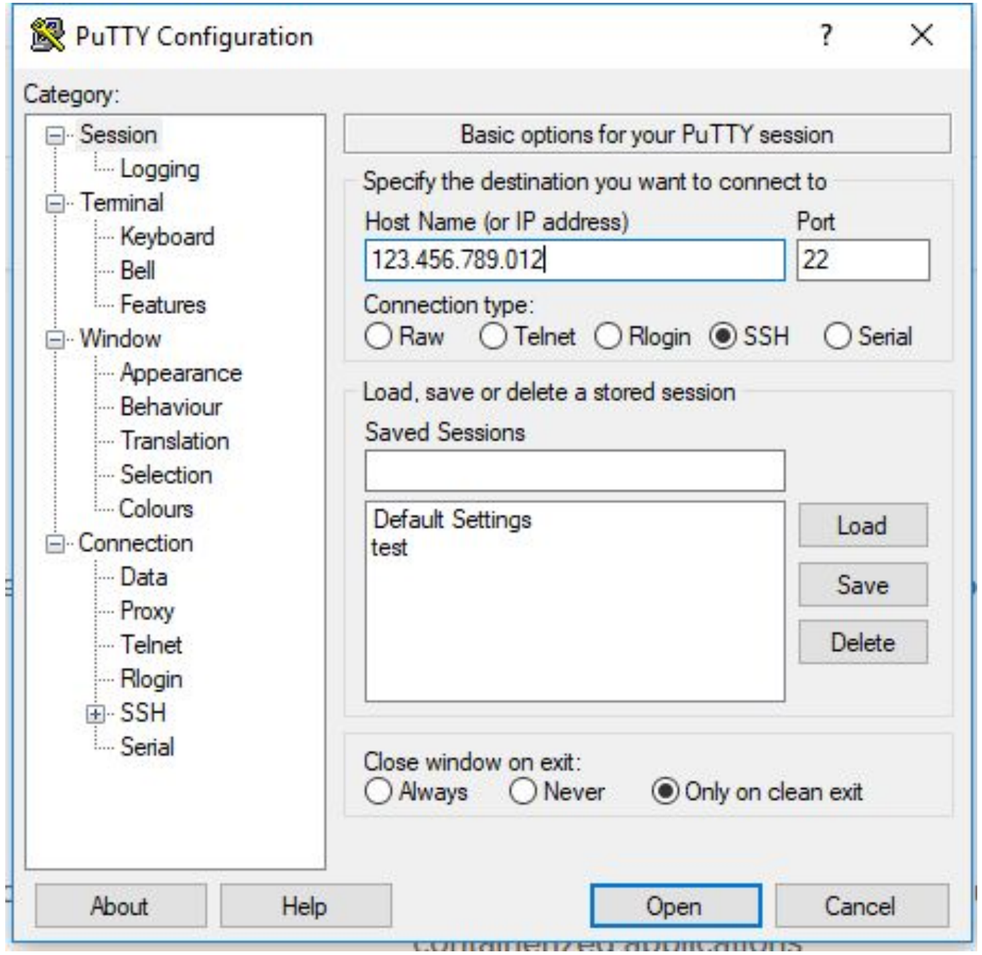
Learn Full Stack Web Development
www.projectfullstack.com

Step 1: Setup Server

- Using Digital Ocean as a VPS provider:
  - Use this link (also in description) for $100 in Digital Ocean credits
    - https://m.do.co/c/b9a816fb103e
  - Register / Login
  - Create a new droplet
    - Distribution: 18.04
    - Size: $5/month
      - 1 GB RAM
      - 1 CPU
      - 25GB SSD Disk
    - Datacenter: The closest one to you
    - Hostname: An easily identifiable name

Step 2: Connect To Server

- Using Linux / Mac:
  - SSH
    - $ ssh root@<your_ip>
    - Example: ssh root@123.456.789.012
    - Select yes to add the server to your known hosts
    - Enter password
- Using Windows:
  - PuTTY
    - Select Session from the left sidebar
    - Enter your ip address in the Host Name input
    - Click the “Open” button
    - Select “Yes” at the popup
    - Enter “root” in the login input
    - Enter password



Step 3: Apply Software Updates

- Update the system to make sure we have the most up to date software
- $ apt update && apt upgrade

Step 4: Create directory structure

- CD to the root of your system
  - $ cd /
- Create the following directory structure
  - project-name
    - site
      - logs
      - public
    - django
    - auth

Step 5: Install PIP & Setup VirtualEnv

- PIP is a package manager we use to install python packages
  - Django is a python package!
- Virtualenv is used to manage Python packages for different projects.
  - Using virtualenv allows you to avoid installing Python packages globally which could break system tools or other projects. You can install virtualenv using pip.
- $ sudo apt install python3-pip
- $ sudo pip3 install virtualenv
- Create a virtualenv
  - Make sure you are in /project-name/ directory
  - $ virtualenv venv
  - $ source venv/bin/activate
  - $ pip install django

Step 6: Create Django Project

- Make sure your venv is activated and django is installed
  - $ pip freeze
- $ cd /project-name/django
- $ django-admin startproject some-project-name
- Change directory into the created project where manage.py is
- Add your server's IP address to settings.py Allowed Host constant
- Run the django development server
  - python manage.py runserver 0.0.0.0:8000
- Open a web browser, goto <your-ip-address>:8000
  - Example: 123.456.789.010:8000
- You should see the default django welcome page

Step 7: Install MySQL

- $ sudo apt install mysql-server
- $ sudo mysql_secure_installation
  - Yes to everything!
- Creating a user and a database
  - $ mysql
  - mysql> CREATE USER ‘username’@’localhost’ IDENTIFIED BY ‘password’;
  - mysql> CREATE DATABASE djangoproject;
  - mysql> GRANT ALL PRIVILEGES ON djangoproject.* to ‘username’@’localhost’;
  - mysql> FLUSH PRIVILEGES

Step 8a: Connect MySQL and Django

- Install “mysqlclient” python package
  - $ sudo apt install python3-dev
  - $ sudo apt install libmysqlclient-dev
  - $ pip install mysqlclient
- Add the following to /project-name/auth/mysql.cnf
  - [client]
    database = ‘your-database’
    user = 'your-mysql-username'
    password = 'your-mysql-username-password'
    default-character-set = 'utf8'
- Restart MySQL
  - $ sudo systemctl restart mysql

Step 8b: Connect MySQL and Django

- Add the following to settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'OPTIONS': {
            'read_default_file': '/tutorial/auth/mysql.cnf',
        },
    }
```

- Check django, create superuser, make migrations, runserver
  - $ python manage.py check
  - $ python manage.py migrate
  - $ python manage.py createsuperuser
  - $ python manage.py runserver 0.0.0.0:8000
- In a browser go to your-ip-address:8000/admin and log in with the superuser credentials

Step 9a: Install & Configure Apache2

- $ sudo apt install apache2 libapache2-mod-wsgi-py3
- Check apache installation
  - In a browser go to your-ip-address
  - You should see the Apache2 default page
- CD to /etc/apache2/sites-available/
  - Change 000-default.conf to the following:
    - See next slide

Step 9b: Install & Configure Apache2

- CD to /etc/apache2/sites-available/
  - Change 000-default.conf to the following:

```
<VirtualHost *:80>

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog /project_name/site/logs/error.log
        CustomLog /project_name/site/access.log combined



        <Directory /<path to directory containing wsgi.py/>
                <Files wsgi.py>
                        Require all granted
                </Files>
        </Directory>


        WSGIDaemonProcess projectname python-path=<abs path to directory containing manage.py is> py
thon-home=/project_name/venv
        WSGIProcessGroup projectname
        WSGIScriptAlias / /<path to directory containing wsgi.py/wsgi.py

</VirtualHost>
```

Step 9c: Install & Configure Apache2

- Example 000-default.conf:

```
<VirtualHost *:80>

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog /tutorial/site/logs/error.log
        CustomLog /tutorial/site/access.log combined

        <Directory /tutorial/django/tutorial/tutorial/>
                <Files wsgi.py>
                        Require all granted
                </Files>
        </Directory>

        WSGIDaemonProcess tutorial python-path=/tutorial/django/tutorial python-home=/tutorial/venv
        WSGIProcessGroup tutorial
        WSGIScriptAlias / /tutorial/django/tutorial/tutorial/wsgi.py

</VirtualHost>
```

Step 9d: Install & Configure Apache2

- Make sure your syntax is correct for the 000-default.conf
  - $ sudo apachectl configtest
- Restart Apache2 for changes to take effect
  - $ sudo service apache2 restart
- In a browser visit your ip address

Step 10: Configuring Static Files

- In a browser, visit your-ip-address/admin
  - Notice the static files (css/javascript/etc) are not loading
- First we need a place for static files to live on our server
  - $ mkdir /project_name/site/public/static
- Add the following to settings.py:
  - STATIC_ROOT = ‘project_name/site/public/static
- Now that django knows where to put static files, we have to tell it to do so:
  - python manage.py collectstatic
- Finally we have to tell Apache to look in this directory for static files
  - Add the following to 000-default.conf
  - alias /static /project_name/site/public/static
    <Directory /project_name/site/public/static>
    Require all granted
    </Directory>
- In a browser, visit your-ip-address/admin. You should now have static files being served