

Deployment of Django App with PostgreSQL

virtual environment creation

- first started with Django is to create the virtual environment.

```
sudo pip install virtualenv
```

- Created a project folder called “myproject” in the “/var” directory:

```
mkdir /var/myproject  
  
cd /var/myproject
```

- Then used virtualenv to create environment-specific dependencies to be installed

```
[root@pga /]# virtualenv /var/myproject/myprojectenv  
  
Using base prefix '/opt/rh/rh-python36/root/usr'  
  
New python executable in /var/myproject/myprojectenv/bin/python3  
  
Also creating executable in /var/myproject/myprojectenv/bin/python  
  
Installing setuptools, pip, wheel...done.
```

- Python is now installed and activate the environments

```
[root@pga /]# source /var/myproject/myprojectenv/bin/activate  
  
(myprojectenv) [root@pga /]# python --version  
  
Python 3.6.3
```

Step to Installing Django

In below we are installing Django into that environment

```
pip install django
```

Now, Django is installed and start a new Django project:

```
/var/myproject/myprojectenv/bin  
  
(myprojectenv) [root@pga bin]# django-admin.py startproject myproject
```

I used below code

```
< mycode.py>
```

```
. . .
```

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql_psycopg2',  
        'NAME': '<mydb_name>',  
        'USER': '<username>',  
        'PASSWORD': '<password>',  
        'HOST': '<db_hostname_or_ip>',  
        'PORT': '<db_port>',  
    }  
}
```

- Now Started the application to confirm the app status:

```
(myprojectenv) [root]# python manage.py runserver 0.0.0.0:5000
```

```
Watching for file changes with StatReloader
```

```
Performing system checks...
```

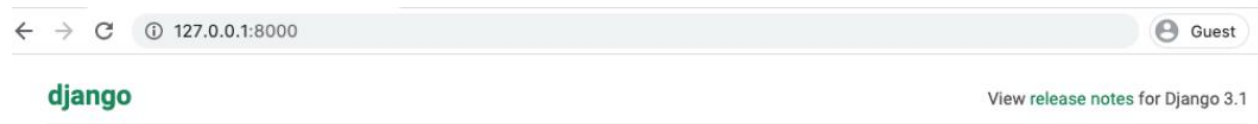
```
System check identified no issues (0 silenced).
```

```
December 06, 2019 - 07:06:22
```

Django version 3.0, using settings 'myproject.settings'

Starting development server at <http://0.0.0.0:5000/>

- Browsed the IPs of local host EC2 for confirm the applicated



Login prompt of Django – Given my admin access

A screenshot of the Django administration login page. The page has a dark blue header with the text 'Django administration'. Below the header, there are three input fields labeled 'Username:', 'Password:', and 'OTP Token:'. At the bottom of the form is a blue button labeled 'Log in'.

Admin view after login on Django

Django administration

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	✎ Change
Users	+ Add	✎ Change

Recent actions

My actions

None available

RDS schemas deployed in an AWS account

- First, I Login AWS Management Console and open the Amazon RDS console
- Then,Amazon RDS console, choose the AWS
- I selected - Create database.
- In Teb,Choose a database creation method, select Standard Create.
- In Engine options, selected the engine type: RDS

Create database

Choose a database creation method [Info](#)

- ☒ **Standard create**
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

- ☐ **Easy create**
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type [Info](#)

- ☐ Amazon Aurora



- ☒ MySQL



- ☐ MariaDB



- ☐ PostgreSQL



- ☐ Oracle



- ☐ Microsoft SQL Server



- For Version, choose the appropriate engine version.
- selected in a later step:
 - Multi-AZ failover option
 - Provisioned IOPS storage option
 - Enable deletion protection option

Created the access

☐ Auto generate a password

Amazon RDS can generate a password for you, or you can specify your own password

Master password [Info](#)

●●●●●●●●

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), "(double quote)

Confirm password [Info](#)

●●●●●●●●

Selected the type of instance type

DB instance class

DB instance class [Info](#)

Choose a DB instance class that meets your processing power and memory requirements. The DB instance class options below are limited to those supported by the engine you selected above.

- ☐ Standard classes (includes m classes)
- ☐ Memory optimized classes (includes r and x classes)
- ☒ Burstable classes (includes t classes)

db.t2.small
1 vCPUs 2 GiB RAM Not EBS Optimized

Defined all the networking details

- VPC
- Security Group
- Additional configuration Database port

Finally- Review the RDS in aws console and clicked to create DB



Your DB instance is being created.

Note: Your instance may take a few minutes to launch.

Connecting to your DB instance

Once Amazon RDS finishes provisioning your DB instance, you can use a SQL client application or utility to connect to the instance.

[Learn about connecting to your DB instance](#)