ObjectRocket a rackspace technology company

Contents

Introduction
Prerequisites
Enter the psql console
Create database and user
Install Django in Virtual Environment
Start the Django project
Test the Django application
Conclusion

Resources > PostgreSQL > The Django Tutorial and PostgreSQL database

The Django Tutorial and PostgreSQL database

Written by Data Pilot
January 09, 2020

DJANGO
POSTGRESQL
PSYCOPG2

Subscribe
Like

Have a Database Problem? Speak with an Expert for Free
Get Started >>

Introduction

Django is a fast, high level developmental framework that allows for developing a web application in Python with a lot less hassle than some other frameworks. While Django automatically stores data in a SQLite database, this Django tutorial for postgresql will explain how to store data in Python with the Django framework using the PostgreSQL database cluster.

Prerequisites

- It must be confirmed whether the PostgreSQL database is installed on the local machine, as it will require the credentials that will be set to the Django application. Execute the `sudo systemctl status postgresql` command in a Systemd distro of Linux terminal to confirm the system is running, or just use the `pg_ctl status` command.
- For MacOS, start the Homebrew PostgreSQL service with the `brew services start postgresql` command and then execute the `brew info postgres` command.
- Python must be installed and working properly. Python 3 is recommended as Python 2.7 is being deprecated and losing support.

Enter the psql console

Execute the following command to enter the interactive PostgreSQL terminal that corresponds to the access of the PostgreSQL administrative user:

```
sudo su - postgres
```

Now enter the sudo password or the root authentication for the device and press the -kbd-ENTER-/kbd- key.

When inside the PostgreSQL shell session, execute the following command to enter the console:

```
psql
```

Create database and user

After entering the psql command-line console, a database and user must be created that can be set for the Django application. Execute the following command to create the database:

```
CREATE DATABASE djangodb;
```

Now execute the following command to create a user login and password that will grant privileges to the newly created database:

```
CREATE USER orkb WITH PASSWORD 'mypass123';
```

Next, modify the parameters for the connection to the Django framework. This is needed to set up the operations for the value queries when a connection is established.

Execute the following `ALTER ROLE` SQL statement to modify the parameters:

```
ALTER ROLE orkb SET client_encoding TO 'utf-8';
```

NOTE: This operation will set the encoding to UTF-8, the default expected in Django framework.

Execute the following command to the default transaction isolation:

```
ALTER ROLE orkb SET default_transaction_isolation TO 'read committed';
```

NOTE: This will set the default transaction isolation each time access is granted to make a query that avoids the transactions that are not committed.

Execute the following command to set the timezone:

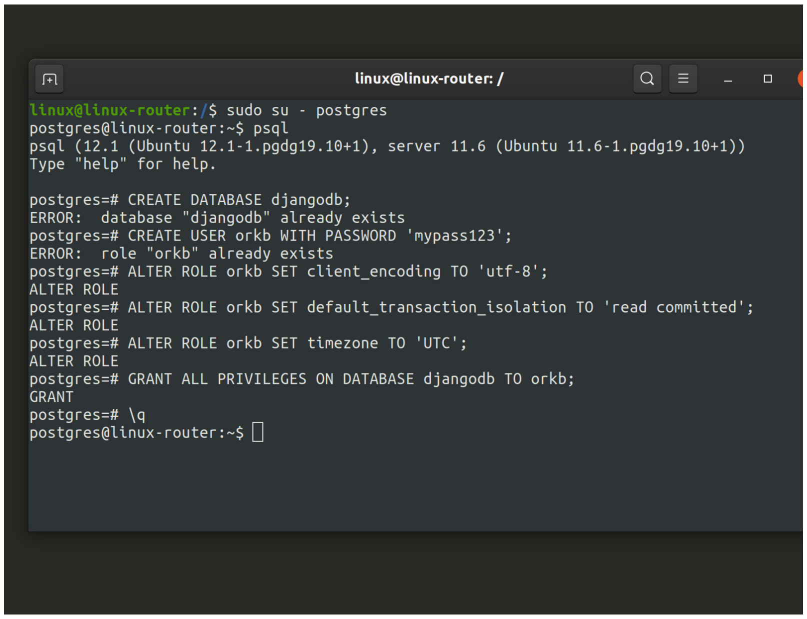
```
ALTER ROLE orkb SET timezone TO 'UTC';
```

NOTE: Django typically uses the UTC, by default, for the timezone.

Finally, execute the following command to grant the user access privileges to the database:

```
GRANT ALL PRIVILEGES ON DATABASE djangodb TO orkb;
```

The results should resemble the following image:



```
linux@linux-router:/$ sudo su - postgres
postgres@linux-router:~$ psql
psql (12.1 (Ubuntu 12.1-1.pgdg19.10+1), server 11.6 (Ubuntu 11.6-1.pgdg19.10+1))
Type "help" for help.

postgres=# CREATE DATABASE djangodb;
ERROR:  database "djangodb" already exists
postgres=# CREATE USER orkb WITH PASSWORD 'mypass123';
ERROR:  role "orkb" already exists
postgres=# ALTER ROLE orkb SET client_encoding TO 'utf-8';
ALTER ROLE
postgres=# ALTER ROLE orkb SET default_transaction_isolation TO 'read committed';
ALTER ROLE
postgres=# ALTER ROLE orkb SET timezone TO 'UTC';
ALTER ROLE
postgres=# GRANT ALL PRIVILEGES ON DATABASE djangodb TO orkb;
GRANT
postgres=# \q
postgres@linux-router:~$
```

Install Django in Virtual Environment

A virtual environment is used to manage the dependencies of different projects by creating a separation of Python isolation to the virtual environments.

Using the terminal, execute the following command to create a directory in the Django framework to store the project:

```
mkdir ~/orkb
```

```
cd ~/orkb
```

Now execute the following command to create a virtual environment that will store the Django project in Python.

```
virtualenv venv
```

NOTE: The `venv` command is now a built-in feature of Python since version 3.6. If using a Python version older than 3.6, the virtual environment package for Python must be installed and activated before installing the Django framework.

Execute the following command to activate the virtual environment:

```
source venv/bin/activate
```

With the venv activated, install the Django framework using the following pip3 command:

```
pip3 install Django
```

Now execute the following command to install the psycopg2 to allow configuring of PostgreSQL:

```
pip3 install psycopg2
```

Start the Django project

Using the below command will allow for creating a folder to hold the codes within the project directory. Note that the manage.py is created after putting a dot at the end of the command to set the project correctly.

Execute the following command to start the Django project:

```
django-admin startproject orkb .
```

With the project now created, execute the following command to configure the project to set the PostgreSQL database credentials:

```
nano venv/settings.py
```

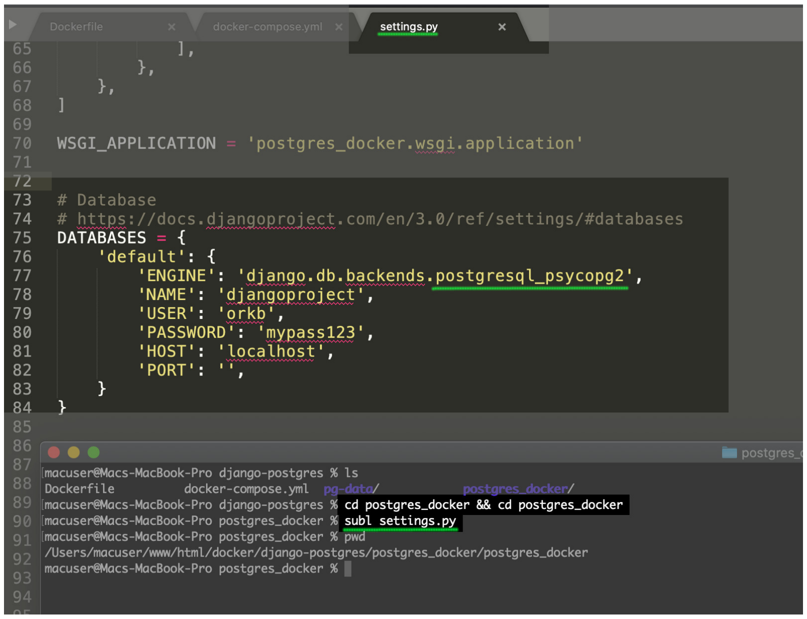
Look for the default configured database as SQLite and set it for the database credentials in PostgreSQL as follows:

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.postgresql_psycopg2',
    'NAME': 'djangoproject',
    'USER': 'orkb',
    'PASSWORD': 'mypass123',
    'HOST': 'localhost',
    'PORT': '',
  }
}
```

Change the `ALLOWED_HOSTS` Python list of allowable domains, as shown below, and add the domain's IP address or `localhost` for local development:

```
ALLOWED_HOSTS = ['localhost', '127.0.0.1']
```

The results should resemble the following:



```
            ],
        },
    },
]

WSGI_APPLICATION = 'postgres_docker.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'djangoproject',
        'USER': 'orkb',
        'PASSWORD': 'mypass123',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

```
macuser@Macs-MacBook-Pro django-postgres % ls
Dockerfile          docker-compose.yml  pg-data/            postgres_docker/
macuser@Macs-MacBook-Pro django-postgres % cd postgres_docker && cd postgres_docker
macuser@Macs-MacBook-Pro postgres_docker % subl settings.py
macuser@Macs-MacBook-Pro postgres_docker % pwd
/Users/macuser/www/html/docker/django-postgres/postgres_docker/postgres_docker
macuser@Macs-MacBook-Pro postgres_docker %
```

Test the Django application

With the configuration finished, the structures of data must be migrated to the database to test the application for the Django server.

Execute the following command to access the project directory:

```
1    cd ~/orkb
```

Now add the following code to create a database structure:

```
1    python3 manage.py makemigrations
```

```
1    python3 manage.py migrate
```

Finally, execute the following code to create an administrative account for the Django application:

```
1    python3 manage.py createsuperuser
2    Username (leave blank to use 'linux'): orkb
3    Email address: orkb@email.com
4    Password:
5    Password (again):
6    Superuser created successfully.
```

NOTE: A current admin account can be used to set up a superuser account for the Django application.

Now execute the following command to test the Django application and confirm it is working properly:

```
python3 manage.py runserver 127.0.0.1:8000
```

When the system displays a message that the application is working, add `/admin` in the URL and enter the previously set up username and password.

Conclusion

This Django tutorial for postgresql explained how to store data in Python with the Django framework using PostgreSQL. The tutorial explained how to create a database and user, grant the user access to the database, install Django in a virtual environment and activate the virtual environment. The Django tutorial for postgresql also explained how to start the Django project, create an administrative account and finally test the Django application. Remember that the virtual environment package for Python must be installed and activated separately, before installing the Django framework, if using a version of Python prior to 3.6.

Related Topics:

- PostgreSQL SELECT First Record on an ObjectRocket Instance
- PostgreSQL Insert for an ObjectRocket Instance
- How to Use the Postgres COUNT on an ObjectRocket Instance
- PostgreSQL UPSERT for an ObjectRocket Instance

Additional Information:

- How to use PostgreSQL UPDATE WHERE on an ObjectRocket Instance
- How to Perform the PostgreSQL Coalesce in ObjectRocket Instance
- How to Use the Postgres Similar To Operator on an ObjectRocket instance
- How to Use the PostgreSQL in Docker in ObjectRocket Instance

Email Address*

SUBMIT

protected by **reCAPTCHA**
Privacy - Terms

SERVICES

Services
MongoDB
Elasticsearch
Redis
Database Migration

PLATFORM

Pricing
Cost of Ownership

COMPANY

About ObjectRocket
Careers
Contact Us

RESOURCES

Resources
Blog
Case Studies
Documentation
EBooks
White Papers

SUPPORT

Support
Submit Ticket
System Status ●
Documentation

Rocket