## [Step 2 – Creating a Database and Database User](https://www.digitalocean.com/community/tutorials/how-to-use-postgresql-with-your-django-application-on-ubuntu-20-04" \l "step-2-creating-a-database-and-database-user)

By default, Postgres uses an authentication scheme called “peer authentication” for local connections. Basically, this means that if the user’s operating system username matches a valid Postgres username, that user can login with no further authentication.

During the Postgres installation, an operating system user named **postgres** was created to correspond to the **postgres** PostgreSQL administrative user. You need to use this user to perform administrative tasks. You can use sudo and pass in the username with the -u option.

Log into an interactive Postgres session by typing:

psql -d postgres -U cashogomez

First, you will create a database for the Django project. Each project should have its own isolated database for security reasons. We will call the database myproject in this guide, but it’s always better to select something more descriptive:

1. CREATE DATABASE myproject;

note

Remember to end all commands at an SQL prompt with a semicolon.

Next, you will create a database user which you will use to connect to and interact with the database. Set the password to something strong and secure:

1. CREATE USER myprojectuser WITH PASSWORD 'password';

Afterwards, you will modify a few of the connection parameters for the user you just created. This will speed up database operations so that the correct values do not have to be queried and set each time a connection is established.

1. ALTER ROLE myprojectuser SET client\_encoding TO 'utf8';
2. ALTER ROLE myprojectuser SET default\_transaction\_isolation TO 'read committed';
3. ALTER ROLE myprojectuser SET timezone TO 'UTC';

You are setting the default encoding to UTF-8, which Django expects. You are also setting the default transaction isolation scheme to “read committed”, which blocks reads from uncommitted transactions. Lastly, you are setting the timezone. By default, your Django projects will be set to use UTC. These are all recommendations from [the Django project itself](https://docs.djangoproject.com/en/1.9/ref/databases/#optimizing-postgresql-s-configuration).

Now, all you need to do is give your database user access rights to the database you created:

1. GRANT ALL PRIVILEGES ON DATABASE myproject TO myprojectuser;

Exit the SQL prompt to get back to the **postgres** user’s shell session:

1. \q
2. pip install Django psycopg2

You can now start a Django project within the myproject directory. This will create a child directory of the same name to hold the code itself, and will create a management script within the current directory. Make sure to add the dot at the end of the command so that this is set up correctly:

1. django-admin startproject myproject .

[**Configure the Django Database Settings**](https://www.digitalocean.com/community/tutorials/how-to-use-postgresql-with-your-django-application-on-ubuntu-20-04#configure-the-django-database-settings)

Now that you have a project, you need to configure it to use the database you created.

Open the main Django project settings file located within the child project directory:

1. nano ~/myproject/myproject/settings.py

Towards the bottom of the file, you will see a DATABASES section that looks like this:

~/myproject/myproject/settings.py

. . .

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

. . .

This is currently configured to use SQLite as a database. You need to change this so that your PostgreSQL database is used instead.

First, change the engine so that it uses the postgresql adaptor instead of the sqlite3 adaptor. For the NAME, use the name of your database (myproject in this example). You also need to add login credentials. You need the username, password, and host to connect to. You will add and leave blank the port option so that the default is selected:

~/myproject/myproject/settings.py

. . .

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'myproject',

'USER': 'myprojectuser',

'PASSWORD': 'password',

'HOST': 'localhost',

'PORT': '',

}

}

. . .

While you are here, you will also need to adjust the ALLOWED\_HOSTS directive. This defines a whitelist of addresses or domain names allowed to connect to the Django instance. Any incoming requests with a **Host** header that is not in this list will raise an exception. Django requires that you set this to prevent a certain class of security vulnerability.

In the snippet below, there are a few commented out examples used to demonstrate:

~/myproject/myproject/settings.py

. . .

# The simplest case: just add the domain name(s) and IP addresses of your Django server

# ALLOWED\_HOSTS = [ 'example.com', '203.0.113.5']

# To respond to 'example.com' and any subdomains, start the domain with a dot

# ALLOWED\_HOSTS = ['.example.com', '203.0.113.5']

ALLOWED\_HOSTS = ['your\_server\_domain\_or\_IP']

In the square brackets, list the IP addresses or domain names that are associated with your Django server. Each item should be listed in quotations with entries separated by a comma. If you wish requests for an entire domain and any subdomains, prepend a period to the beginning of the entry.

When you are finished, save and close the file.

[**Migrate the Database and Test your Project**](https://www.digitalocean.com/community/tutorials/how-to-use-postgresql-with-your-django-application-on-ubuntu-20-04#migrate-the-database-and-test-your-project)

Now that the Django settings are configured, you can migrate your data structures to your database and test out the server.

You can begin by creating and applying migrations to your database. Since you don’t have any actual data yet, this will simply set up the initial database structure:

1. cd ~/myproject
2. python manage.py makemigrations
3. python manage.py migrate

After creating the database structure, you can create an administrative account by typing:

1. python manage.py createsuperuser

You will be asked to select a username, provide an email address, and choose and confirm a password for the account.

If you followed the initial server setup guide, you should have a UFW firewall in place. Before you can access the Django development server to test your database, you need to open the port in your firewall.

Allow external connections to the port by typing:

1. sudo ufw allow 8000

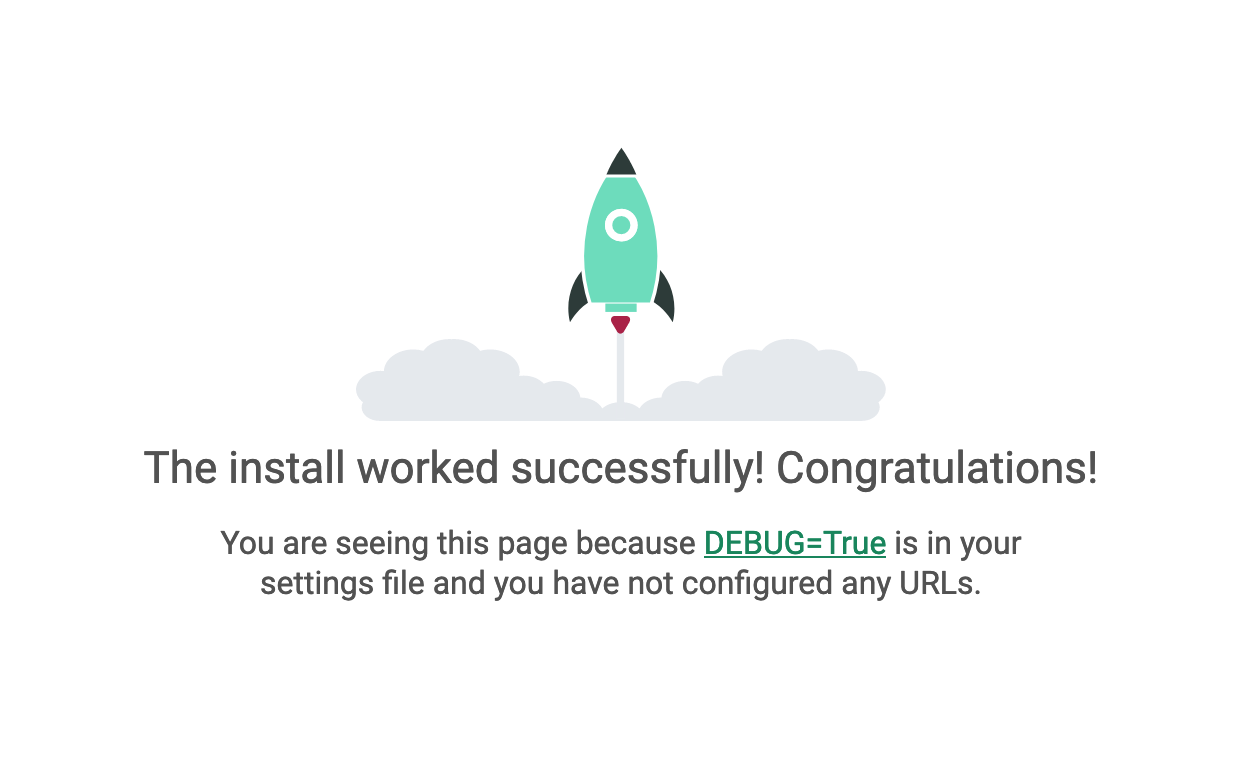
Once you have the port open, you can test that your database is performing correctly by starting up the Django development server:

1. python manage.py runserver 0.0.0.0:8000

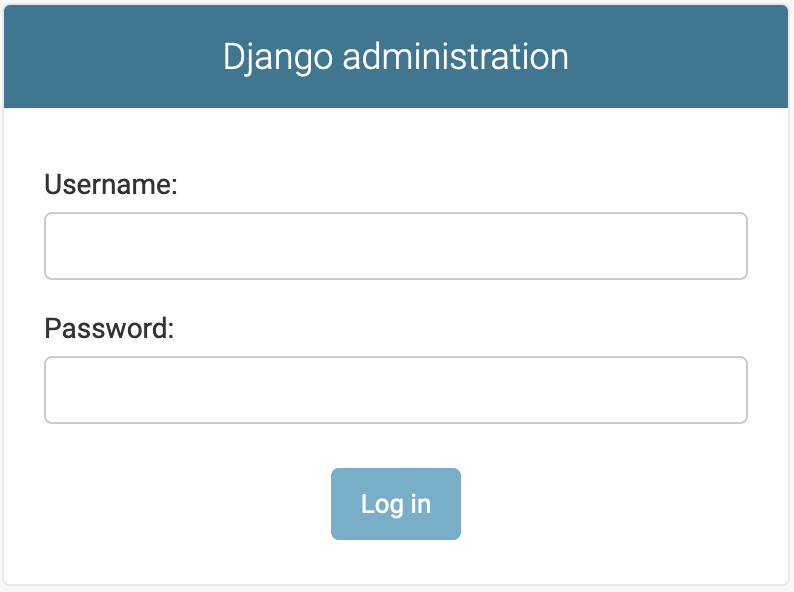
In your web browser, visit your server’s domain name or IP address followed by :8000 to reach default Django root page:

http://server\_domain\_or\_IP:8000

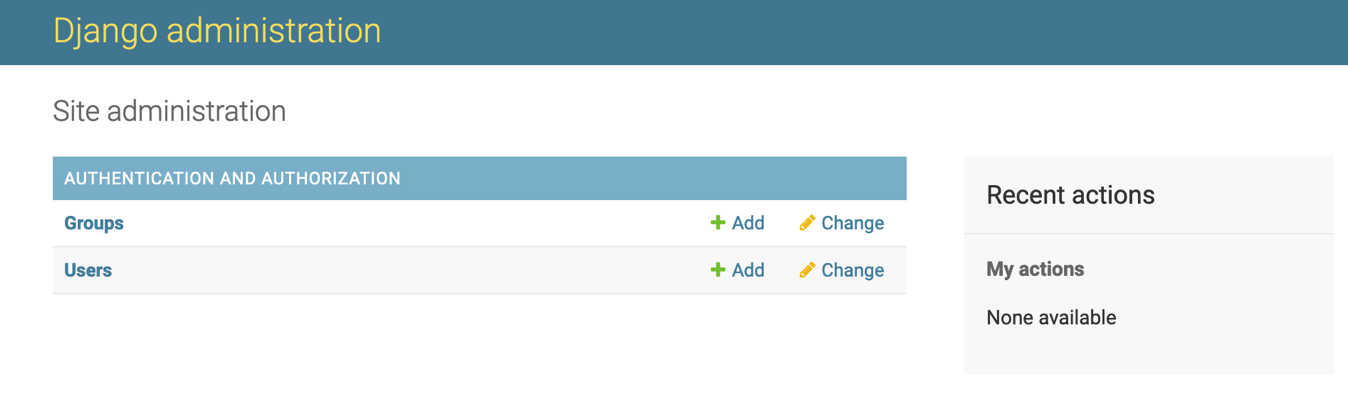
You should see the default index page:



Append /admin to the end of the URL and you should be able to access the login screen to the admin interface:



Enter the username and password you just created with the createsuperuser command. You will then be taken to the admin interface:



When you’re done investigating, you can stop the development server by hitting CTRL-C in your terminal window.

By accessing the admin interface, you have confirmed that you database has stored your user account information and that it can be appropriately accessed.

[**Conclusion**](https://www.digitalocean.com/community/tutorials/how-to-use-postgresql-with-your-django-application-on-ubuntu-20-04#conclusion)

In this guide, you have demonstrated how to install and configure PostgreSQL as the backend database for a Django project. While SQLite can easily handle the load during development and light production use, most projects benefit from implementing a more full-featured database management system.

To take your project even further, see our guide on [How To Set Up Django with Postgres, Nginx, and Gunicorn on Ubuntu 20.04](https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu-20-04).