

Create Your First Django Model



Estimated time needed: 15 minutes

Learning Objectives

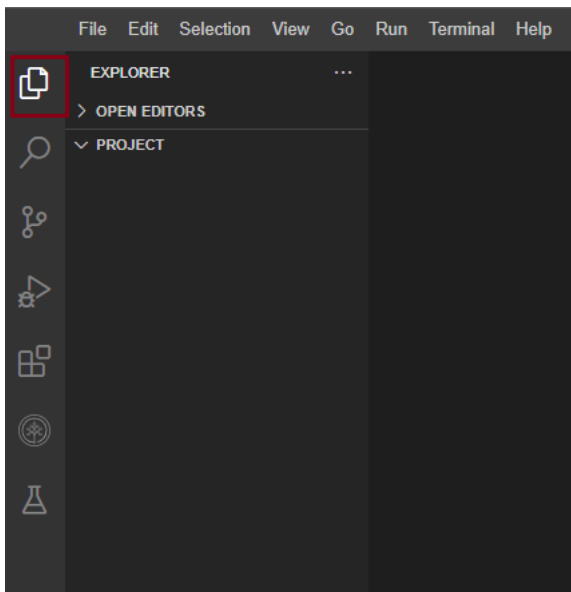
After completing this lab, you will be able to:

- Get familiar with Theia IDE and Django
- Setup a standalone Django ORM app
- Create your first Django project and application
- Create and test your first Django model

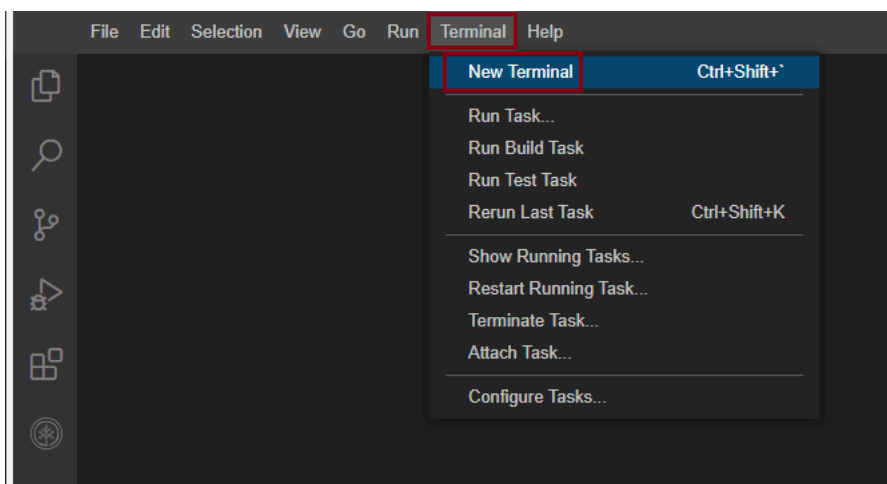
Working with files in Cloud IDE

If you are new to Cloud IDE, this section will show you how to create and edit files, which are part of your project, in Cloud IDE.

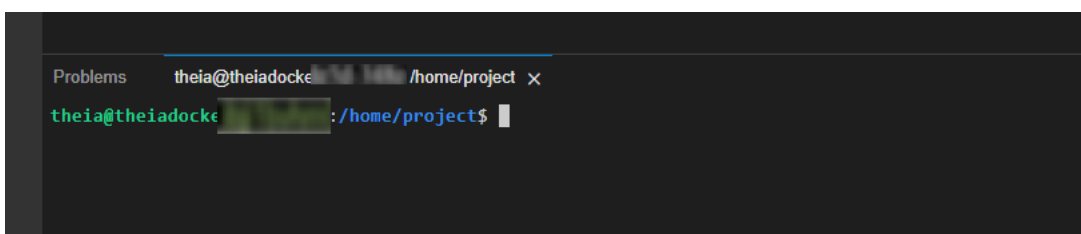
To view your files and directories inside Cloud IDE, click on this files icon to reveal it.



Click on New, and then New Terminal.



This will open a new terminal where you can run your commands.



Concepts covered in the lab

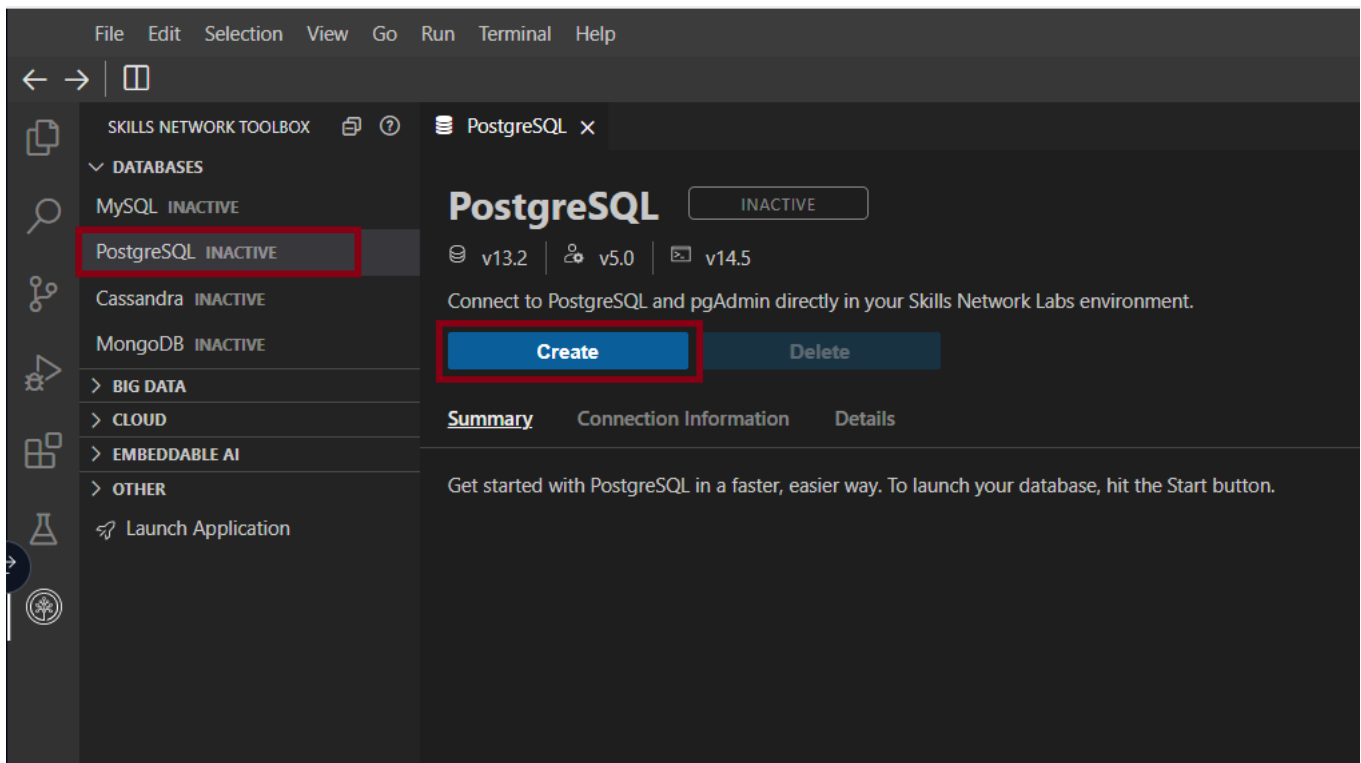
1. Django ORM: A Python ORM component of the Django web application framework, where each Django model maps to a database table.
2. PostgreSQL or Postgres: An open-source relational database management system used by Django.
3. Psycopg: An interface used by Django for working with PostgreSQL.
4. PGAdmin: An admin and development tool for managing PostgreSQL server.
5. Database migrations: The management of version-controlled, incremental and reversible changes to relational database schemas to update or revert the schema to a newer or older version.

Start PostgreSQL in Theia

PostgreSQL, also known as Postgres, is an open-source relational database management system and it is one of the main databases Django uses.

If you are using the Theia environment hosted by [Skills Network Labs](#), a pre-installed PostgreSQL instance is provided for you.

You can start PostgreSQL from UI by finding the SkillsNetwork icon on the left menu bar and selecting PostgreSQL from the DATABASES menu item:



Once the PostgreSQL has been started, you can check the server connection information from the UI.

Please mark down the connection information such as generated username, password, and host, etc, which will be used to configure Django app to connect to this database.

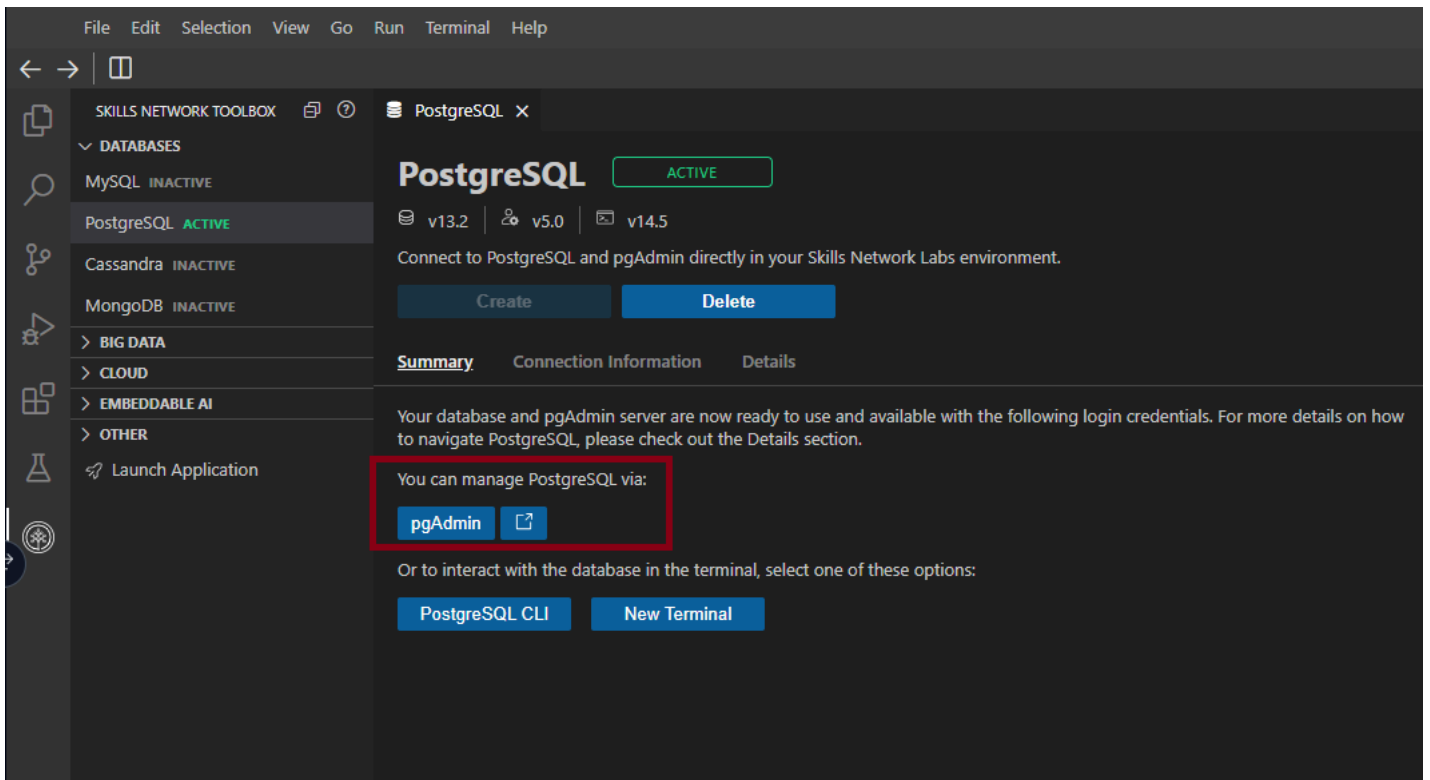
The screenshot shows the Skills Network Lab interface for a PostgreSQL instance. The left sidebar lists various databases: MySQL (inactive), PostgreSQL (active), Cassandra (inactive), and MongoDB (inactive). Below these are categories like BIG DATA, CLOUD, EMBEDDABLE AI, and OTHER. The main panel displays the PostgreSQL instance configuration under the 'Connection Information' tab. The instance is named 'PostgreSQL' and is in an 'ACTIVE' state. It is version v13.2, v5.0, and v14.5. The configuration details include:

- POSTGRES_USERNAME:** postgres
- POSTGRES_HOST:** 172.21.244.157
- POSTGRES_PORT:** 5432
- URL:** https://labs-postgres-shaggy-prehistoric-knife.postgres.databases.labs.skills.network
- POSTGRES_URL:** https://labs-postgres-shaggy-prehistoric-knife.postgres.databases.labs.skills.network
- PostgreSQL CLI Command:** export PGPASSWORD=pJDH4iXaVvFhH105j5EEAUP8; psql --host 172.21.244.157 -p 5432 -U postgres
- POSTGRES_COMMAND:** export PGPASSWORD=pJDH4iXaVvFhH105j5EEAUP8; psql --host 172.21.244.157 -p 5432 -U postgres
- POSTGRES_PASSWORD:** pJDH4iXaVvFhH105j5EEAUP8
- POSTGRES_TITLE:** Postgres Database
- POSTGRES_ID:** labs-postgres-shaggy-prehistoric-knife

- Install these must-have packages before you setup the environment to access postgres.

```
pip install --upgrade distro-info
pip3 install --upgrade pip==23.2.1
```

- Also, a pgAdmin instance is installed and started for you. It is a popular PostgreSQL admin and development tool for you to manage your PostgreSQL server interactively.



Setup Your First Django App

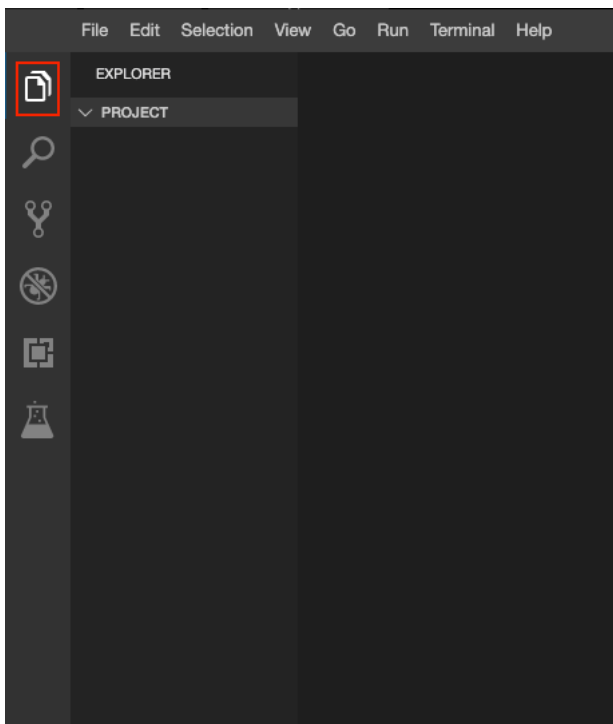
If the terminal was not open, go to Terminal > New Terminal and make sure your current Theia directory is /home/project.

- Run the following command-lines to download a code template for this lab.

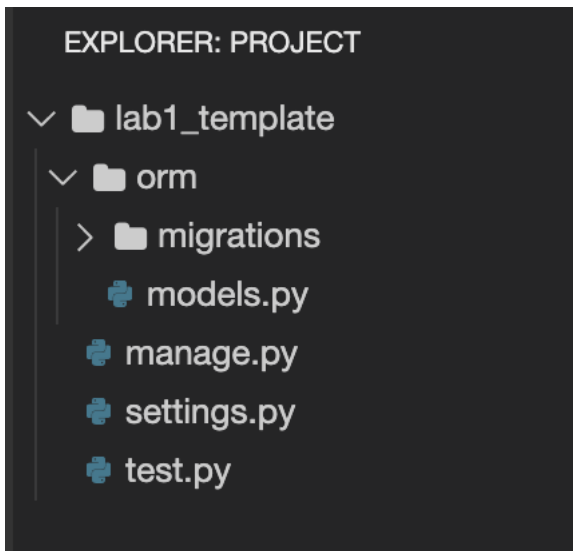
```
wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m3_django_orm/lab1_template.zip"
unzip lab1_template.zip
rm lab1_template.zip
```

You may need to press Enter key to run the last rm command.

After downloading and unzipping is done, click the Explorer on the left menu (the first button).



Your first Django project should look like the following:



Next, we need to set up a proper runtime environment for Django app.

- If the terminal was not open, go to Terminal > New Terminal and cd to the project folder

```
cd lab1_template
```

Let's set up a virtual environment which to contain all the packages we need.

```
pip install virtualenv
virtualenv djangoenv
source djangoenv/bin/activate

pip install -r requirements.txt
```

The requirements.txt contains all necessary Python packages for you to run this lab.

In the created project, you could find some important project files:

- manage.py is a command-line interface that allows you to interact with and manage your Django project
- settings.py contains setting information about this project such as databases or installed Django apps
- orm folder is a container for a standalone Django ORM app
- orm/models.py contains model definitions

You will learn more details about these files in subsequent learning modules and labs.

Next let's connect our Django project to the PostgreSQL we started.

- Open settings.py and scroll to DATABASES section.

```
# Postgre SQL
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'postgres',
        'USER': 'postgres',
        'PASSWORD': 'Place it with your password saved in Step 1',
        'HOST': 'postgres',
        'PORT': '5432',
    }
}
```

- Replace the value of PASSWORD to be the generated PostgreSQL password generated in Step 1.

After that, your settings.py file should look like the following:

```
# Postgre SQL
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'postgres',
        'USER': 'postgres',
        'PASSWORD': 'MjY2NjkteWx1by0y',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}

INSTALLED_APPS = (
    'orm',
)

SECRET_KEY = 'SECRET KEY for this Django Project'
```

- OK now your first Django orm app is ready and you can start defining your first Django model in the next step

Define Your First Django Model

- Open `orm/models.py` (under `lab1_template/orm/` folder) and copy / paste the following snippet under comment

Define your first model from here: to define a User model

```
class User(models.Model):
    # CharField for user's first name
    first_name = models.CharField(null=False, max_length=30, default='john')
    # CharField for user's last name
    last_name = models.CharField(null=False, max_length=30, default='doe')
    # CharField for user's date for birth
    dob = models.DateField(null=True)
```

Now you have defined a very simple User model that only contains `first_name` and `last_name` as `CharField` and `dob` as `DateField`

Activate the User Model

After the User model is defined, Django will be creating a corresponding database table called `orm_user`. The first part `orm` is your app name and the second part `user` is the model name.

Whenever you make changes to your models such as creating new models or modifying existing models, you need to perform database migrations. Django provides utils via `manage.py` interface to help you perform migrations.

- If your current working directory is not `/home/project/lab1_template`, cd to the project folder

```
cd /home/project/lab1_template
```

- First, you will need to generate migration scripts for `orm` app

```
python3 manage.py makemigrations orm
```

and you should see the following result in the terminal:

```
Migrations for 'orm':
  orm/migrations/0001_initial.py
  - Create model User
```

- `orm/migrations` folder is where Django stores the changes to your models and you may wonder what SQL statements Django has created for your model migrations.

You can check the SQL statements by running:

```
python3 manage.py sqlmigrate orm 0001
```

It prints the `orm_user` table creation SQL statement for you.

```
BEGIN;
--
-- Create model User
--
CREATE TABLE "orm_user" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "first_name" varchar(30) NOT NULL, "last_name" varchar(30) NOT NULL, "
COMMIT;
```

In most cases, with Django Model works as ORM component, you don't need to worry about the SQL part at all. You can just use Django Model APIs provided to query/manipulate data in databases.

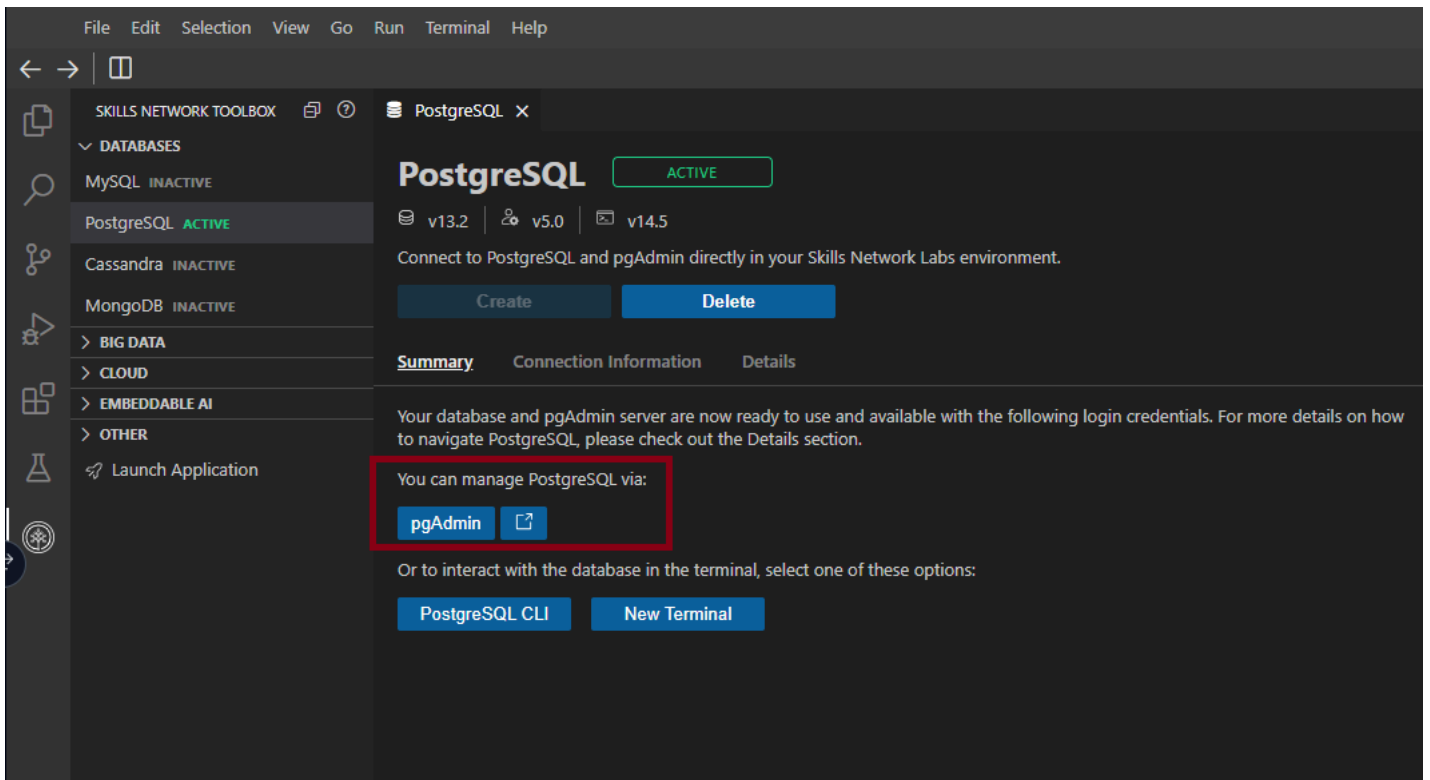
- Next, you can perform the migration to create `orm_user` table by running:

```
python3 manage.py migrate
```

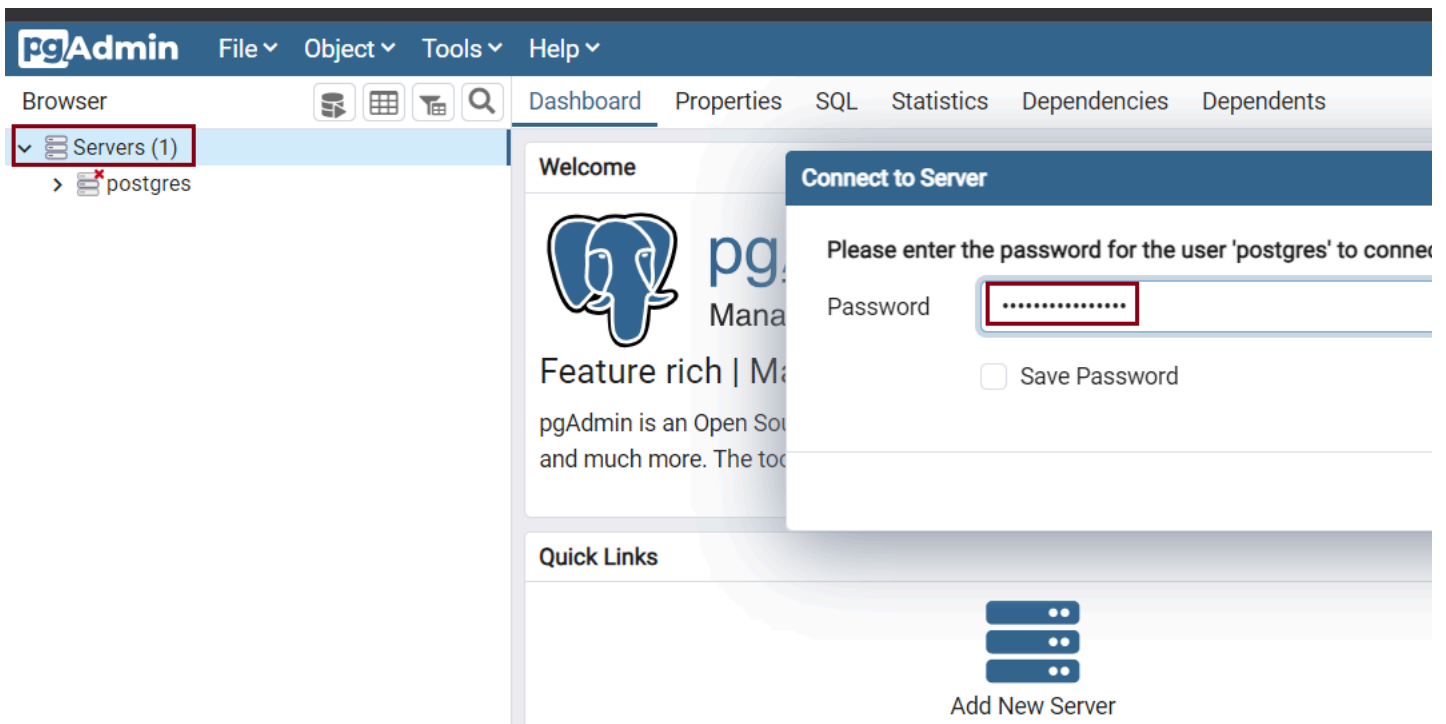
Django will perform migrations for all the installed apps including `orm` app.

```
Operations to perform:
  Apply all migrations: orm
Running migrations:
  Applying orm.0001_initial... OK
```

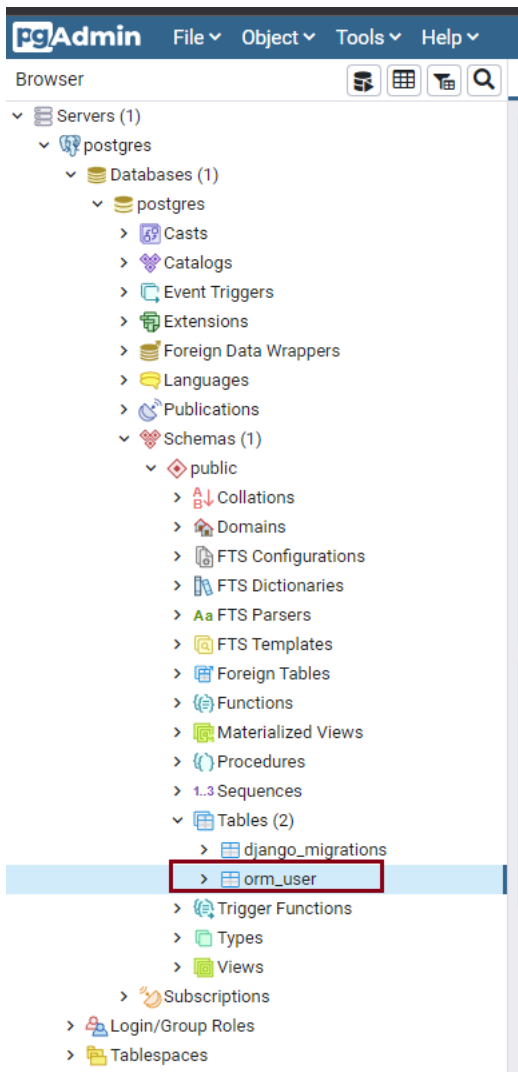
- Click on the Skills Network button on the left to open the **Skills Network Toolbox** Then, click on **Database**, followed by **PostgreSQL**, and then click **pgAdmin**. It will launch in your browser.
Note: Make sure PostgreSQL is already in Active state before proceeding.



- Once pgAdmin is started in a new browser tab, choose Servers, enter the password generated from Step 1 and click OK.



- Then expand Databases->postgres->Schemas->public->Tables, you should see the orm_user table created in previous migration step.



Test the Model

- Open `test.py` and you can find a `test_setup()` method to save a mockup user object and try to check if the user object was saved successfully.
- You could run the `test.py` to test your model:

```
python3 test.py
```

You should see a message that reads,

Django Model setup completed.

That's it, you have set up your first Django ORM app with first Django model.

Summary

In this lab, you have set up your first Django project and application. You have also created and tested your first Django model.

In the next lab, you will try to create some more real Django Models and perform Create, Read, Update, and Delete (CRUD) operations on them.

Author(s)

[Yan Luo](#)

© IBM Corporation. All rights reserved.