Lesson-End Project

Containerizing a Legacy Application

Project agenda: To containerize a legacy Python application using Docker, integrating diverse storage and volume strategies for ensuring data persistence and optimizing performance

Description: Your company is experiencing the challenge of modernizing legacy applications while ensuring portability and streamlined deployment processes. To address this, you are undertaking a project that aims to containerize legacy applications using Docker, integrating a variety of storage and volume strategies.

Tools required: Docker and Ubuntu OS

Prerequisites: None

Expected deliverables: A containerized legacy application integrated with storage and volume strategies

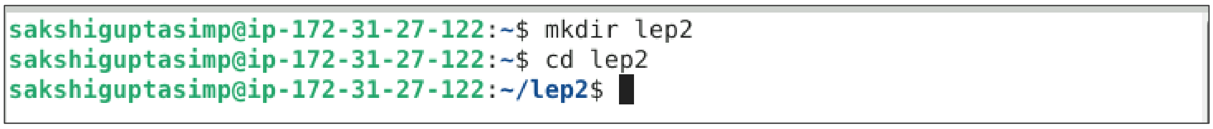
Steps to be followed:
1. Create a Dockerfile and define the Python dependencies
2. Create the Django project using Docker Compose
3. Set up a database connection and configure Docker Compose
4. Change the ownership of files and start the application
5. Verify the setup and clean up the environment

Step 1: Create a Dockerfile and define the Python dependencies

1.1 Create a project folder and navigate to it using the following commands:

mkdir lep2

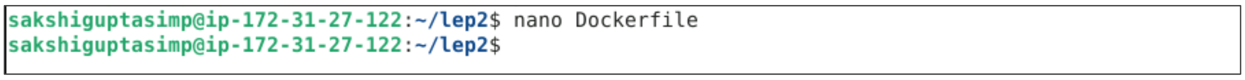
cd lep2



```
sakshiguptasimp@ip-172-31-27-122:~$ mkdir lep2
sakshiguptasimp@ip-172-31-27-122:~$ cd lep2
sakshiguptasimp@ip-172-31-27-122:~/lep2$ 
```

1.2 Create a Dockerfile using the following command:

nano Dockerfile



```
sakshiguptasimp@ip-172-31-27-122:~/lep2$ nano Dockerfile
sakshiguptasimp@ip-172-31-27-122:~/lep2$
```

1.3 Add the following configurations to the Dockerfile:

FROM python:3

ENV PYTHONUNBUFFERED 1

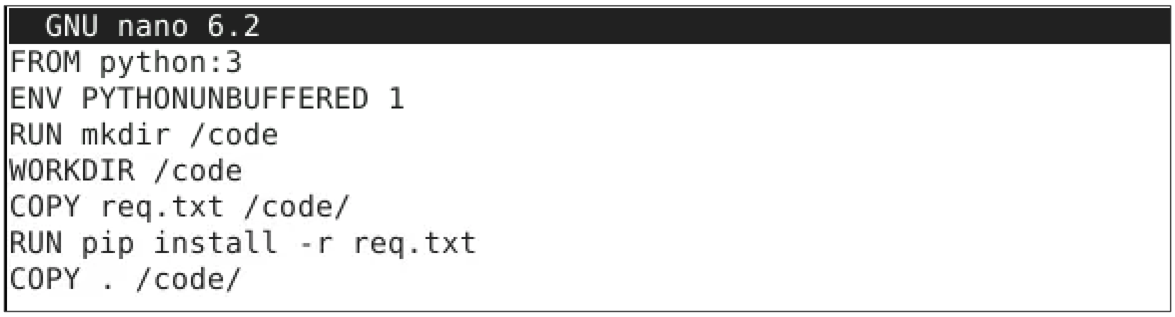
RUN mkdir /code

WORKDIR /code

COPY req.txt /code/

RUN pip install -r req.txt

COPY . /code/



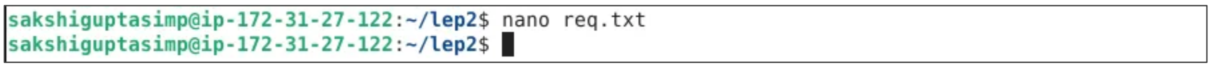
```
  GNU nano 6.2
FROM python:3
ENV PYTHONUNBUFFERED 1
RUN mkdir /code
WORKDIR /code
COPY req.txt /code/
RUN pip install -r req.txt
COPY . /code/
```

This Dockerfile snippet sets up a Python 3 environment, creates a working directory, installs dependencies from a requirements file, and copies the application code into the container.

Note: To save the file press **Ctrl+X**, then **Y**, and finally **Enter**

1.4 Create a Python dependencies file using the following command:

nano req.txt

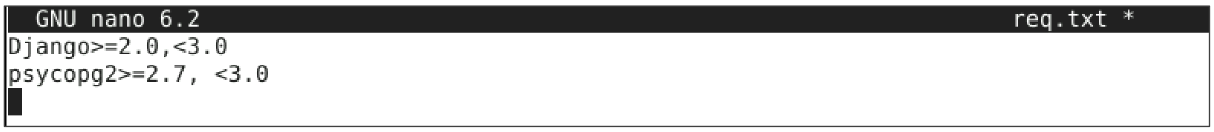


```
sakshiguptasimp@ip-172-31-27-122:~/lep2$ nano req.txt
sakshiguptasimp@ip-172-31-27-122:~/lep2$ █
```

1.5 Add the following dependencies in the req.txt file:

Django>=2.0,<3.0

psycopg2>=2.7, <3.0

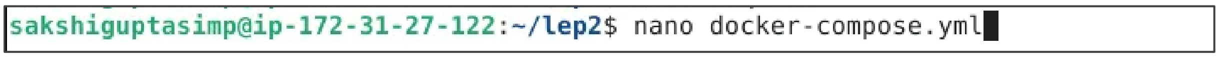


```
 GNU nano 6.2                                              req.txt *
Django>=2.0,<3.0
psycopg2>=2.7, <3.0
```

Note: To save the file press **Ctrl+X**, then **Y**, and finally **Enter**

1.6 Create a **docker-compose.yml** file using the following command:

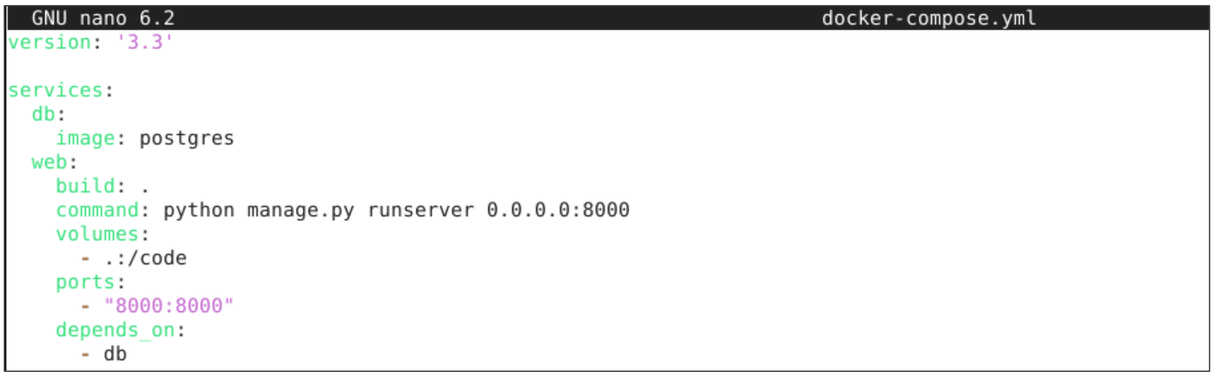
nano docker-compose.yml



```
sakshiguptasimp@ip-172-31-27-122:~/lep2$ nano docker-compose.yml
```

1.7 Define the following services in the **docker-compose.yml** file:

```
version: '3.3'

services:
  db:
    image: postgres
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code   # Implementing bind mount strategy here
    ports:
      - "8000:8000"
    depends_on:
      - db
```

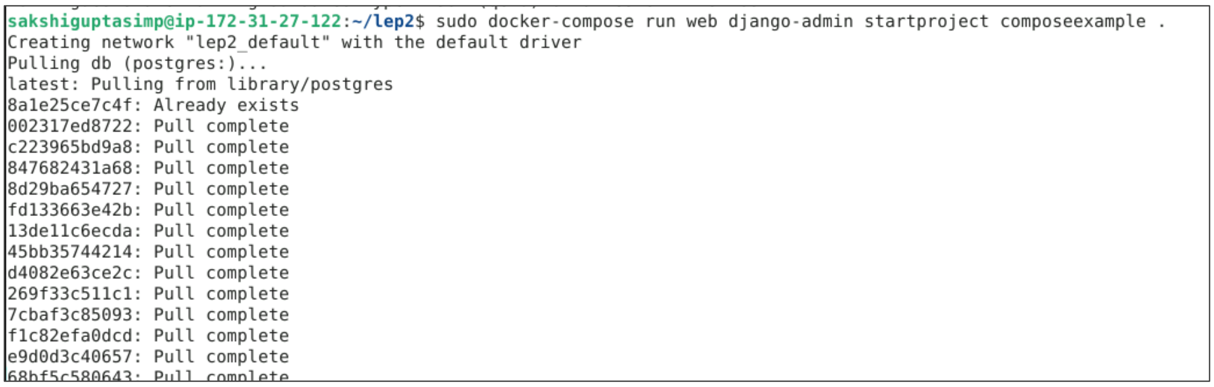


```
 GNU nano 6.2                                    docker-compose.yml
version: '3.3'

services:
  db:
    image: postgres
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    depends_on:
      - db
```

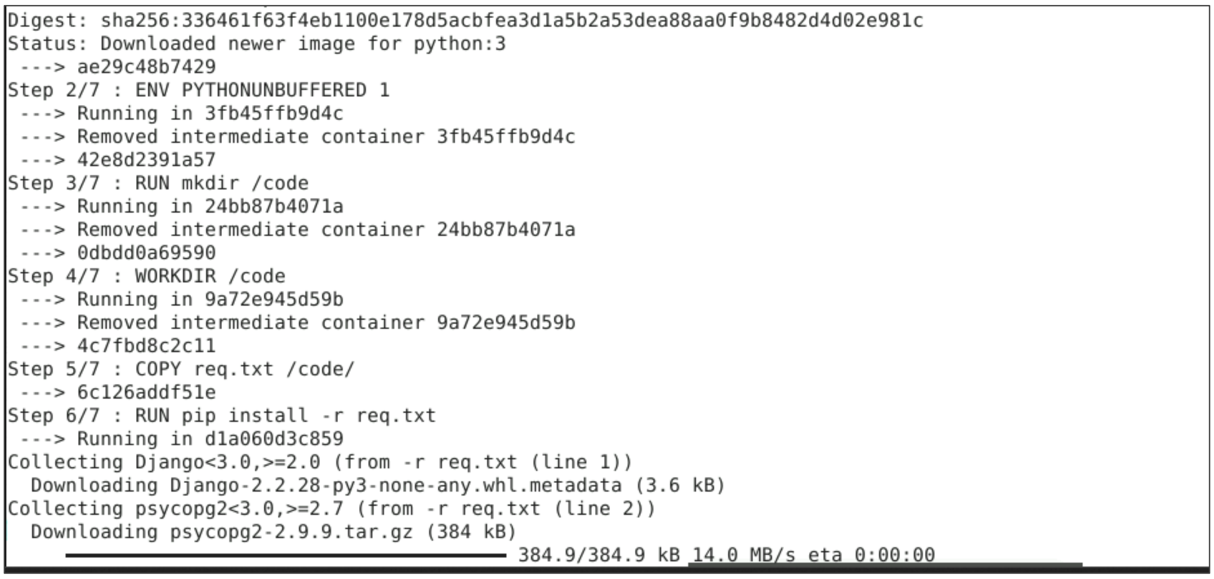
Step 2: Create the Django project using Docker Compose

2.1 Create a Django project with Docker Compose using the following command:

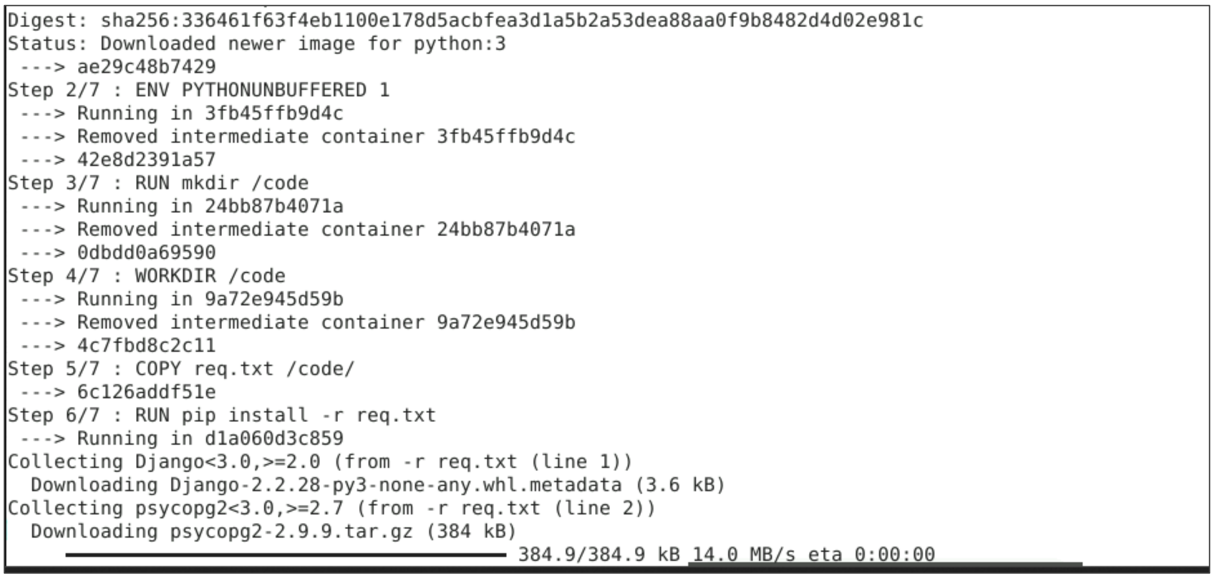
sudo docker-compose run web django-admin startproject composeexample .



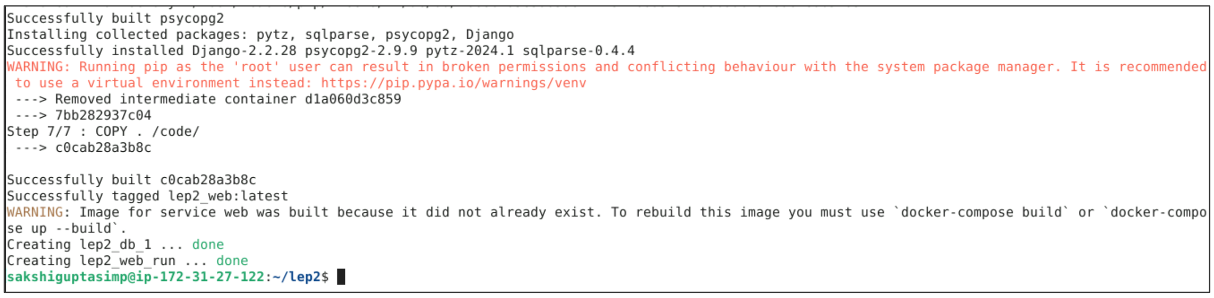
```
sakshiguptasimp@ip-172-31-27-122:~/lep2$ sudo docker-compose run web django-admin startproject composeexample .
Creating network "lep2_default" with the default driver
Pulling db (postgres:)...
latest: Pulling from library/postgres
8a1e25ce7c4f: Already exists
002317ed8722: Pull complete
c223965bd9a8: Pull complete
847682431a68: Pull complete
8d29ba654727: Pull complete
fd133663e42b: Pull complete
13de11c6ecda: Pull complete
45bb35744214: Pull complete
d4082e63ce2c: Pull complete
269f33c511c1: Pull complete
7cbaf3c85093: Pull complete
f1c82efa0dcd: Pull complete
e9d0d3c40657: Pull complete
68bf5c580643: Pull complete
```



```
Digest: sha256:336461f63f4eb1100e178d5acbfea3d1a5b2a53dea88aa0f9b8482d4d02e981c
Status: Downloaded newer image for python:3
 ---> ae29c48b7429
Step 2/7 : ENV PYTHONUNBUFFERED 1
 ---> Running in 3fb45ffb9d4c
 ---> Removed intermediate container 3fb45ffb9d4c
 ---> 42e8d2391a57
Step 3/7 : RUN mkdir /code
 ---> Running in 24bb87b4071a
 ---> Removed intermediate container 24bb87b4071a
 ---> 0dbdd0a69590
Step 4/7 : WORKDIR /code
 ---> Running in 9a72e945d59b
 ---> Removed intermediate container 9a72e945d59b
 ---> 4c7fbd8c2c11
Step 5/7 : COPY req.txt /code/
 ---> 6c126addf51e
Step 6/7 : RUN pip install -r req.txt
 ---> Running in d1a060d3c859
Collecting Django<3.0,>=2.0 (from -r req.txt (line 1))
  Downloading Django-2.2.28-py3-none-any.whl.metadata (3.6 kB)
Collecting psycopg2<3.0,>=2.7 (from -r req.txt (line 2))
  Downloading psycopg2-2.9.9.tar.gz (384 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 384.9/384.9 kB 14.0 MB/s eta 0:00:00
```



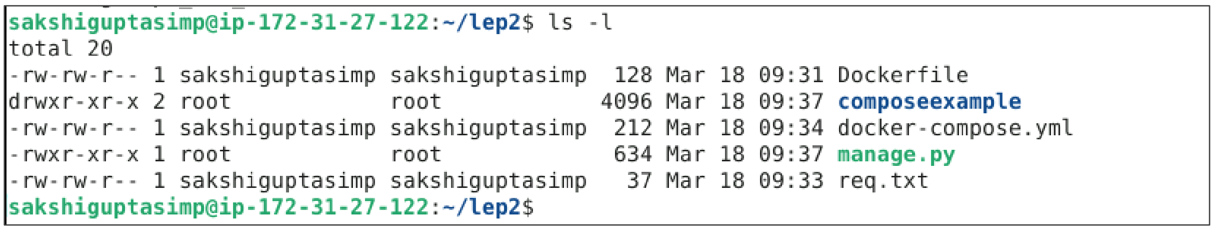
```
Digest: sha256:336461f63f4eb1100e178d5acbfea3d1a5b2a53dea88aa0f9b8482d4d02e981c
Status: Downloaded newer image for python:3
 ---> ae29c48b7429
Step 2/7 : ENV PYTHONUNBUFFERED 1
 ---> Running in 3fb45ffb9d4c
 ---> Removed intermediate container 3fb45ffb9d4c
 ---> 42e8d2391a57
Step 3/7 : RUN mkdir /code
 ---> Running in 24bb87b4071a
 ---> Removed intermediate container 24bb87b4071a
 ---> 0dbdd0a69590
Step 4/7 : WORKDIR /code
 ---> Running in 9a72e945d59b
 ---> Removed intermediate container 9a72e945d59b
 ---> 4c7fbd8c2c11
Step 5/7 : COPY req.txt /code/
 ---> 6c126addf51e
Step 6/7 : RUN pip install -r req.txt
 ---> Running in d1a060d3c859
Collecting Django<3.0,>=2.0 (from -r req.txt (line 1))
  Downloading Django-2.2.28-py3-none-any.whl.metadata (3.6 kB)
Collecting psycopg2<3.0,>=2.7 (from -r req.txt (line 2))
  Downloading psycopg2-2.9.9.tar.gz (384 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 384.9/384.9 kB 14.0 MB/s eta 0:00:00
```



```
Successfully built psycopg2
Installing collected packages: pytz, sqlparse, psycopg2, Django
Successfully installed Django-2.2.28 psycopg2-2.9.9 pytz-2024.1 sqlparse-0.4.4
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended
 to use a virtual environment instead: https://pip.pypa.io/warnings/venv
 ---> Removed intermediate container d1a060d3c859
 ---> 7bb282937c04
Step 7/7 : COPY . /code/
 ---> c0cab28a3b8c

Successfully built c0cab28a3b8c
Successfully tagged lep2_web:latest
WARNING: Image for service web was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compo
se up --build`.
Creating lep2_db_1 ... done
Creating lep2_web_run ... done
sakshiguptasimp@ip-172-31-27-122:~/lep2$ 
```

2.2 List the contents of the Django project directory using the following command:

ls -l

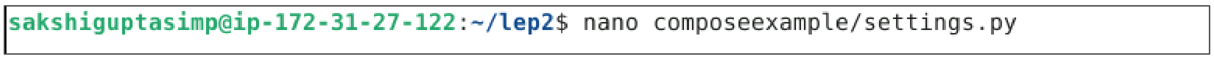


```
sakshiguptasimp@ip-172-31-27-122:~/lep2$ ls -l
total 20
-rw-rw-r-- 1 sakshiguptasimp sakshiguptasimp  128 Mar 18 09:31 Dockerfile
drwxr-xr-x 2 root            root            4096 Mar 18 09:37 composeexample
-rw-rw-r-- 1 sakshiguptasimp sakshiguptasimp  212 Mar 18 09:34 docker-compose.yml
-rwxr-xr-x 1 root            root             634 Mar 18 09:37 manage.py
-rw-rw-r-- 1 sakshiguptasimp sakshiguptasimp   37 Mar 18 09:33 req.txt
sakshiguptasimp@ip-172-31-27-122:~/lep2$
```

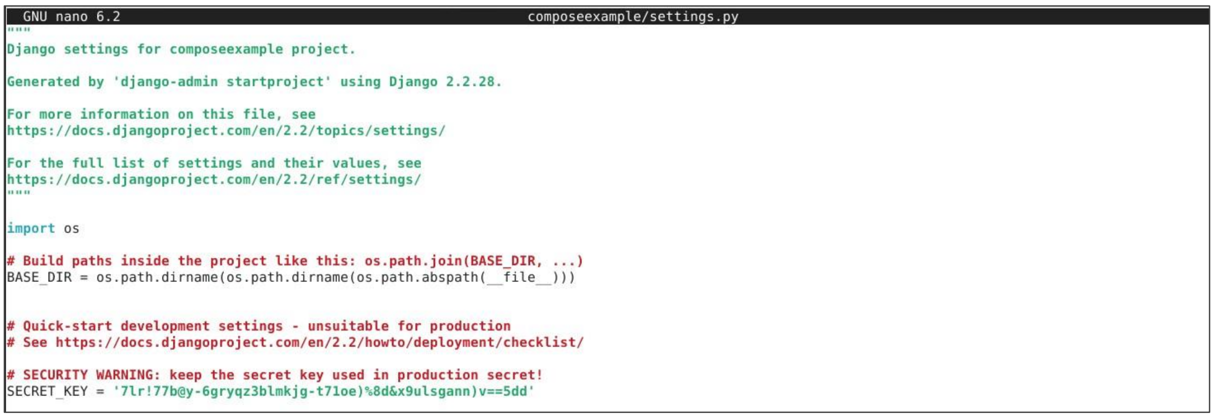
Step 3: Set up a database connection and configure Docker Compose

3.1 Edit the **composeexample/settings.py** file to set up the database connection using the following command:

nano composeexample/settings.py



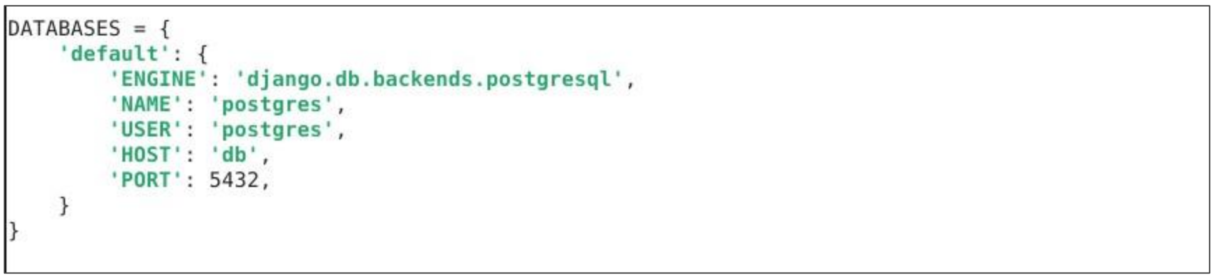
```
sakshiguptasimp@ip-172-31-27-122:~/lep2$ nano composeexample/settings.py
```



```
  GNU nano 6.2                                                       composeexample/settings.py
"""
Django settings for composeexample project.

Generated by 'django-admin startproject' using Django 2.2.28.

For more information on this file, see
https://docs.djangoproject.com/en/2.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.2/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '7lr!77b@y-6gryqz3blmkjg-t7loe)%8d&x9ulsgann)v==5dd'
```

3.2 Replace the DATABASES configuration with the following script:

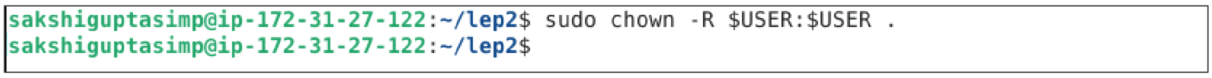
DATABASES = {
'default': {
'ENGINE': 'django.db.backends.postgresql',
'NAME': 'postgres',
'USER': 'postgres',
'HOST': 'db',
'PORT': 5432,
}
}



```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'postgres',
        'USER': 'postgres',
        'HOST': 'db',
        'PORT': 5432,
    }
}
```

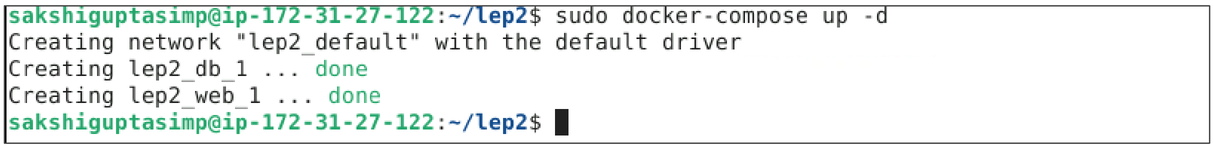
Step 4: Change the ownership of files and start the application

4.1 Change the ownership of new files to the current user using the following command:
sudo chown -R $USER:$USER .



```
sakshiguptasimp@ip-172-31-27-122:~/lep2$ sudo chown -R $USER:$USER .
sakshiguptasimp@ip-172-31-27-122:~/lep2$
```

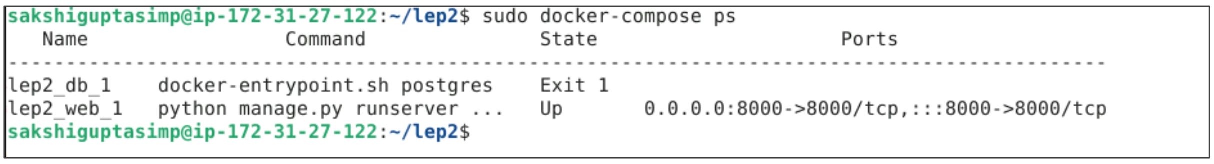
4.2 Run the following command to start the application:
sudo docker-compose up -d



```
sakshiguptasimp@ip-172-31-27-122:~/lep2$ sudo docker-compose up -d
Creating network "lep2_default" with the default driver
Creating lep2_db_1 ... done
Creating lep2_web_1 ... done
sakshiguptasimp@ip-172-31-27-122:~/lep2$
```

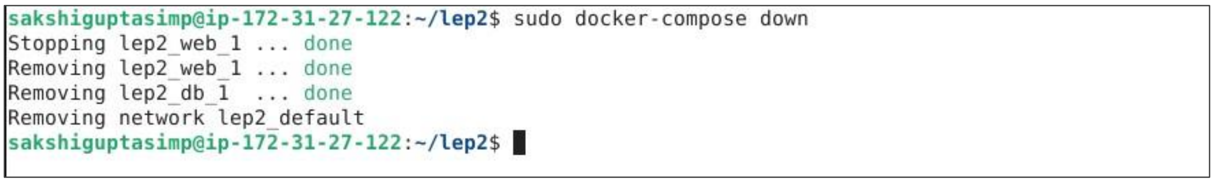
Step 5: Verify the setup and clean up the environment

5.1 List the running containers to verify the setup using the following command:
sudo docker-compose ps



```
sakshiguptasimp@ip-172-31-27-122:~/lep2$ sudo docker-compose ps
   Name                   Command                  State                        Ports
-----------------------------------------------------------------------------------------------
lep2_db_1    docker-entrypoint.sh postgres    Exit 1
lep2_web_1   python manage.py runserver ...   Up       0.0.0.0:8000->8000/tcp,:::8000->8000/tcp
sakshiguptasimp@ip-172-31-27-122:~/lep2$
```

5.2 Run the following command to clean up the resources:
sudo docker-compose down



```
sakshiguptasimp@ip-172-31-27-122:~/lep2$ sudo docker-compose down
Stopping lep2_web_1 ... done
Removing lep2_web_1 ... done
Removing lep2_db_1  ... done
Removing network lep2_default
sakshiguptasimp@ip-172-31-27-122:~/lep2$ 
```

By following these steps, you have successfully containerized a legacy Python application using Docker, integrating diverse storage and volume strategies to ensure data persistence and optimize performance.