**Task 1: Environment setup**

**Set up initial files and folders, and virtual environment**

1. **Create a folder structure:**
   * Create a subfolder inside the 3-DevOps/week1/ directory named workshop1.
   * Open this folder in **VS Code**.
2. **Create the virtual environment:**
   * In the terminal, navigate to the workshop1/ folder.
   * Run the command to create a virtual environment:
   * python3 -m venv venv
   * A venv/ folder will be created inside the workshop1/ folder.
3. **Activate the virtual environment:**
   * **Windows:**
   * .\venv\Scripts\activate
   * **macOS/Linux:**
   * source venv/bin/activate

You should see (venv) in your command prompt, indicating that the virtual environment is active.

1. **Upgrade pip:**
   * Run the following command to upgrade pip to version 21.1.2:
   * pip install --upgrade pip==21.1.2
2. **Create the app folder:**
   * Inside the workshop1/ folder, create a new folder named app.
   * Navigate to the app/ folder:
   * cd app
3. **Create requirements.txt file:**
   * Inside the app/ folder, create a file named requirements.txt.
   * Add the following contents to the file:
   * django==3.2.2
   * gunicorn==20.0.4
   * djangorestframework==3.12.4
   * python-decouple==3.4
   * psycopg2-binary==2.9.9
4. **Install required packages:**
   * Run the following command to install the packages from requirements.txt:
   * pip install -r requirements.txt
5. **Verify installation:**
   * Run the following command to list installed packages:
   * pip list
   * Ensure that all the packages listed in requirements.txt are installed.

**Stop and remove all existing containers**

1. **Using Docker CLI:**
   * List all Docker containers, including stopped ones:
   * docker ps -a
   * If there are any running containers, stop them:
   * docker stop <container\_id>
   * Remove the containers:
   * docker rm <container\_id>
2. **Using Docker Desktop:**
   * Open Docker Desktop and navigate to the **Containers/Apps** page.
   * Remove all containers using the trash can icon.

**Task 2: Use django\_admin to scaffold out the Django project**

1. **Navigate to the app/ folder:**
2. cd workshop1/app
3. **Run django-admin to create the Django project:**
4. django-admin startproject nc\_tutorials .
5. **Check the project structure:**
   * Run the following command to list the files:
   * ls

You should see the following files:

manage.py nc\_tutorials/

1. **Disable database connection:**
   * Open nc\_tutorials/settings.py and find the section labeled # Database.
   * Comment out the entire DATABASES object.
2. **Run the Django development server:**
3. python manage.py runserver 8000
   * You should see output similar to:
   * Starting development server at http://127.0.0.1:8000/
4. **Verify the server is running:**
   * Open a web browser and go to http://127.0.0.1:8000.
   * You should see the Django welcome page.
5. **Stop the server:**
   * Press Ctrl + C in the terminal to stop the server.

**Task 3: Docker image and container**

**Add Docker files and configure Dockerfile**

1. **Open the app/ folder in VS Code.**
2. **Generate Dockerfile:**
   * In VS Code, use the **Command Palette** to select **Add Docker Files to Workspace**.
   * Follow the same settings as you did in the first hellodjango exercise.
3. **Modify Dockerfile:**
   * Open the Dockerfile and change the base image to python:3.9-slim.
   * Update the last instruction to:
   * CMD ["gunicorn", "--bind", "0.0.0.0:8000", "nc\_tutorials.wsgi"]

**Build the Docker image and run it in a container**

1. **Build the Docker image:**
   * Run the following command in the app/ folder:
   * docker build -t django-app .
2. **Run the Docker container:**
   * Run the container:
   * docker run -p 8000:8000 django-app
3. **Verify the app in the container:**
   * Open a web browser and go to http://127.0.0.1:8000.
   * You should see the same Django welcome page.
4. **Stop and remove the container:**
   * In the terminal, stop the container using Ctrl + C.
   * Remove the container:
   * docker ps -a
   * docker rm <container\_id>

Confirm in the browser that the app is no longer running.

This completes the setup, project creation, and Dockerization tasks for your Django application.

To guide you through the tasks you outlined, I'll break them down step by step, providing you with the necessary instructions.

**Task 4: Docker Compose, PostgreSQL, pgAdmin**

**1. Creating Directories and SQL File:**

1. Open the workshop1/ folder in VS Code.
2. Create the folder structure: data/misc/.
3. Inside misc/, create a file named django\_init.sql and add the following code:

DROP DATABASE IF EXISTS nc\_tutorials\_db;

CREATE DATABASE nc\_tutorials\_db;

\c nc\_tutorials\_db

SET statement\_timeout = 0;

SET lock\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SET check\_function\_bodies = false;

SET client\_min\_messages = warning;

SET default\_tablespace = '';

SET default\_with\_oids = false;

**2. Creating docker-compose.yml file:**

1. In the workshop1/ folder, create a file named docker-compose.yml.
2. Add the following content:

version: "3.8"

services:

web:

build: ./app

command: python manage.py runserver 0.0.0.0:8000

volumes:

- ./app/:/usr/src/app/

ports:

- 8000:8000

environment:

- DB\_NAME=nc\_tutorials\_db

- DB\_USER=postgres

- DB\_PASSWORD=admin123

- DB\_HOST=pg

- DB\_PORT=5432

- DATABASE=postgres

depends\_on:

- pg

pg:

container\_name: pg\_container

image: postgres:13.2-alpine

restart: always

environment:

POSTGRES\_USER: postgres

POSTGRES\_PASSWORD: admin123

POSTGRES\_DB: nc\_tutorials\_db

POSTGRES\_HOST\_AUTH\_METHOD: trust

ports:

- "5432:5432"

volumes:

- ./data:/data

- ./data/misc/django\_init.sql:/docker-entrypoint-initdb.d/1-django-init.sql

pgadmin:

container\_name: pgadmin\_container

image: dpage/pgadmin4:5.2

restart: always

environment:

PGADMIN\_DEFAULT\_EMAIL: admin@example.com

PGADMIN\_DEFAULT\_PASSWORD: admin123

PGADMIN\_LISTEN\_PORT: 5433

PGADMIN\_CONFIG\_SERVER\_MODE: "False"

PGADMIN\_CONFIG\_MASTER\_PASSWORD\_REQUIRED: "False"

ports:

- "5433:5433"

depends\_on:

- pg

**3. Update Dockerfile:**

1. In the workshop1/app/ folder, open the Dockerfile.
2. Comment out the following lines as they are no longer needed with Docker Compose:

RUN adduser --disabled-password --gecos "" appuser

USER appuser

CMD ["gunicorn", "--bind", "0.0.0.0:8000", "nc\_tutorials.wsgi"]

**4. Bringing up Containers with Docker Compose:**

1. Use the following command to start the containers in detached mode:

docker compose up -d

1. Confirm that all services are running with:

docker compose ps

1. Open pgAdmin in your browser at http://localhost:5433. Log in using the credentials admin@example.com and password admin123.
2. Create a new server configuration in pgAdmin:
   * **Server Name:** nucamp (or any name you prefer).
   * **Hostname/address:** pg.
   * **Password:** admin123.
3. Connect to the server and confirm that nc\_tutorials\_db is listed under Databases.
4. After confirming everything is working, stop and remove the containers:

docker compose down --rmi all

**Task 5: Implement Password Protection**

**1. Creating .env File:**

1. In the app/ folder, create a file named .env.
2. Add the following content to store your database settings:

# Database Settings

DB\_NAME=nc\_tutorials\_db

DB\_USER=postgres

DB\_PASSWORD=admin123

DB\_HOST=127.0.0.1

DB\_PORT=5432

**2. Update settings.py:**

1. In settings.py, add the following import after the pathlib import:

from decouple import config

1. Scroll down to the DATABASES section and replace the commented-out block with:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql\_psycopg2',

'NAME': config('DB\_NAME'),

'USER': config('DB\_USER'),

'PASSWORD': config('DB\_PASSWORD'),

'HOST': config('DB\_HOST'),

'PORT': config('DB\_PORT'),

}

}

This setup will read your database credentials securely from the .env file using python-decouple.

**3. Bring up Containers:**

1. Use the following command to bring the containers up again:

docker compose up -d

This ensures your Django app connects securely to the PostgreSQL database using the credentials from the .env file.

It looks like you're progressing well with setting up Docker Compose, PostgreSQL, pgAdmin, and Django! Here's a summary of the steps in your current task:

**Task 6: Perform Migrations to Generate Database Tables**

1. **Ensure Containers are Running**: Make sure that your Docker Compose containers (web, pg, pgadmin) are up and running.
2. **Run Migrations**: Use the following command to generate the default Django tables:

docker compose exec web python manage.py migrate --noinput

1. **Confirm Tables**:
   * After running the migration, open pgAdmin.
   * Refresh the page and reconnect to the nc\_tutorials\_db database.
   * You should see tables related to Django such as auth\_\* and django\_\*.
2. **Stop and Remove Containers**: Once confirmed, stop and remove the containers:
3. docker compose down --rmi all

**Task 7: Create Tutorials and Users Sub-Apps**

1. **Create Sub-Apps**: Inside the Django project, create two sub-apps:
2. python manage.py startapp tutorials
3. python manage.py startapp users
4. **Add Apps to INSTALLED\_APPS**: In settings.py, add the apps:
5. INSTALLED\_APPS = [
6. 'tutorials',
7. 'users',
8. # other apps
9. ]
10. **Extract and Add Files**:
    * Unzip tutorials.zip and users.zip and add their contents to the tutorials/ and users/ directories, respectively.
11. **Include App URLs**: In urls.py of the main Django project (nc\_tutorials/):
12. from django.urls import path, include
13. urlpatterns = [
14. path('', include('tutorials.urls')),
15. path('', include('users.urls')),
16. # other paths
17. ]
18. **Run Containers**: Start the containers in detached mode:
19. docker compose up -d
20. **Confirm Web App Running**: Visit [http://127.0.0.1:8000](http://127.0.0.1:8000/) and you should see a Django error page, which is expected at this stage.

**Task 7: Creating Tutorials and Users Sub-Apps**:

### ****Step 1: Open Bash Terminal****

1. Navigate to the **workshop1/** folder in your terminal.
2. cd path\_to\_your\_project/workshop1
3. Activate the virtual environment if it's not already activated:
4. source venv/bin/activate

(Windows users may need to use *venv\Scripts\activate* instead.)

1. Use cd to navigate to the project root folder (**app/**):
2. cd app

### ****Step 2: Create Sub-Apps****

Run the following commands to create the tutorials and users sub-apps:

python manage.py startapp tutorials

python manage.py startapp users

You should now see two new folders inside the **app/** folder: tutorials/ and users/.

### ****Step 3: Update**** INSTALLED\_APPS ****in**** settings.py

1. Open the nc\_tutorials/settings.py file.
2. Locate the INSTALLED\_APPS list.
3. Add the following lines to the end of the list:
4. 'tutorials',
5. 'users',

### ****Step 4: Download and Replace Files****

1. **Download the provided tutorials.zip and users.zip files.**
2. Extract their contents.
3. Replace the contents of:
   * workshop1/app/tutorials/ with the files from tutorials.zip.
   * workshop1/app/users/ with the files from users.zip.
4. Ensure that the contents of these folders match the provided structure.

### ****Step 5: Update the Main URL Configuration****

1. Open the urls.py file in the nc\_tutorials/ folder.
2. Locate the from django.urls import path line and update it to:
3. from django.urls import path, include
4. Add the following lines to the urlpatterns list:
5. path('', include('tutorials.urls')),
6. path('', include('users.urls')),

### ****Step 6: Bring Up the Containers****

1. Use Docker Compose to start all containers in detached mode:
2. docker compose up -d
3. Confirm that the Django app is running:
   * Visit: [http://127.0.0.1:8000](http://127.0.0.1:8000/).
   * You should see a **Django error page** with a yellow background.

### ****Debugging Tips****

* If you see a **generic browser error** instead of a Django server error:
  1. Verify that your containers are running:
  2. docker ps
  3. Restart the containers:
  4. docker compose up --build -d
  5. Check the logs for the web container:
  6. docker logs workshop1\_web\_1
* In **Docker Desktop**, expand the workshop1 app in the **Containers/Apps** panel. Click on workshop1\_web\_1 to view its logs.

### ****What’s Next?****

If the Django error page is visible, proceed to **Task 8**. If not, resolve any issues first before continuing. Let me know if you need help with debugging!

**Task 8: Re-Generate Tables, Now Including tutorials\_tutorial Table**

1. **Generate Migrations**: Generate migration files for your models (the Tutorial class):
2. docker compose exec web python manage.py makemigrations --noinput
3. **Apply Migrations**: Apply the migrations, which will also create the tutorials\_tutorial table:
4. docker compose exec web python manage.py migrate --noinput
5. **Check in pgAdmin**:
   * Refresh the table list in pgAdmin for the nc\_tutorials\_db database.
   * You should see the tutorials\_tutorial table.
6. **Check Django Web App**: Go to the Django web app at [http://127.0.0.1:8000](http://127.0.0.1:8000/) to confirm the application is up and running.

**Debugging Tips:**

* **Docker Logs**: If you encounter issues with the containers, use Docker Desktop's console or logs to troubleshoot.
* **Container Access**: You can access the container filesystem using:
* docker compose exec web ls tutorials/migrations

Once these tasks are complete, you should have a working Django application with PostgreSQL backend, and the ability to manage tutorials and users.

Let me know if you need help with any specific steps!

**Task 9: Use Insomnia to Post New Records**

1. **Launch Insomnia**:
   * Open the Insomnia application.
2. **Create a Request Collection**:
   * Name the request collection: YourFirstName YourLastName Django Tutorials (use your actual first and last name).
3. **Create a New POST Request**:
   * Name the request: Add Tutorial.
   * Select the **POST** HTTP method.
   * Set the **Body** type to JSON.
4. **Set the URL**:
   * The URL for the request is: http://127.0.0.1:8000/api/tutorials/.
5. **Add JSON Body**: In the JSON body, paste the following:
6. {
7. "title": "Introduction to Django",
8. "tutorial\_url": "https://www.djangoproject.com",
9. "image\_path": "../static/images/tutorials/introDjango.png",
10. "description": "A tutorial about Django",
11. "published": true
12. }
13. **Send the Request**:
    * After sending the request, you should see a **201 Created** message in the Preview panel.
    * The response should also include an auto-generated id (likely 1).
14. **Take a Screenshot**:
    * Take a screenshot that shows the **201 Created** message and the name of the request collection (with your full name) visible.
15. **Confirm in Browser**:
    * Go to your browser and visit: [http://127.0.0.1:8000](http://127.0.0.1:8000/).
    * You should see the posted tutorial listed.
16. **Visit API URL**:
    * Go to <http://127.0.0.1:8000/api/tutorials/> to confirm the tutorial is listed.
17. **Edit and Resend the POST Request**:
    * Modify the JSON body as follows:
18. {
19. "title": "Introduction to Docker",
20. "tutorial\_url": "https://www.docker.com",
21. "image\_path": "../static/images/tutorials/introDocker.png",
22. "description": "A tutorial about Docker",
23. "published": false
24. }
25. **Send the Updated Request**:
    * Send the updated POST request, and confirm you get a **201 Created** message again.
    * The new auto-generated id should likely be 2.
26. **Take Another Screenshot**:
    * Take a screenshot of the new **201 Created** message with your full name visible.
27. **Confirm in Browser**:
    * Visit [http://127.0.0.1:8000](http://127.0.0.1:8000/) again, and you should see both tutorials listed.
28. **Test the URLs**:
    * Try out the following URLs:
      + <http://127.0.0.1:8000/api/tutorials/1>
      + <http://127.0.0.1:8000/api/tutorials/2>
      + <http://127.0.0.1:8000/api/tutorials/published>
    * Observe the responses from Django.

**Task 10: Use Python to Test and Grade the Assignment**

1. **Download the Python Script**:
   * Download the autograder.py script to your 3-DEVOPS folder.
   * If needed, right-click on the link to download the file.
2. **Verify Script Location**:
   * Open a new terminal in the 3-DEVOPS folder and check if the script is present:
   * ls
3. **Run the Script**:
   * Run the script using Python:
   * python autograder.py
4. **Follow the Instructions**:
   * The script will provide instructions for the assignment grading process. You can accept the default values for workshop, transport, host, and port unless otherwise specified.
5. **Note the Outfile Location**:
   * After running the script, make note of the location where the output file is saved.
6. **Fix Issues and Re-run (if necessary)**:
   * If the script detects issues, address them and run the script again until everything passes.

These steps will complete your tasks, and you'll have verified your Django application with Insomnia and the grading script. Let me know if you run into any issues!