Name: Minh Hoang
Student number: 152103143

Docker Compose Overview:

Docker Compose is a tool for defining and running multi-container Docker applications. It uses a YAML file (often named docker-compose.yml) to configure the services for your application. With Docker Compose, you can easily define the services, networks, and volumes required for your application to run.

Django Quickstart Guide:

+ Install Docker and Docker Compose:
Ensure that Docker and Docker Compose are installed on your local machine. Docker Compose is often included with Docker Desktop for Windows and macOS, but you may need to install it separately on Linux.

Link to the Docker installation: Install Docker Engine | Docker Docs

+ Understanding the docker-compose.yml fields:

- version: Specifies the version of the Docker Compose file syntax being used.
- services: Defines the different services for your application, such as Django and the database.
- image: Specifies the Docker image to use for each service.
- ports: Maps ports on the container to ports on the host machine, allowing you to access the Django web server.
- volumes: Mounts host directories or named volumes into the container, allowing you to persist data.
- environment: Sets environment variables for the container, such as database credentials or Django settings.
- depends_on: Specifies dependencies between services, ensuring that services are started in the correct order.

+ Running the application:

Once you've defined your docker-compose.yml file, you can start your Django application using the **docker-compose up** command. Docker Compose will read the configuration and start the specified services, including setting up the database and running the Django web server.

+ Accessing the application:

After the services are up and running, you can access your Django application by opening a web browser and navigating to http://localhost. If you've configured a different port in your docker-compose.yml file, you'll need to specify that port in the URL.

Example:

```yaml
version: '3.8'

services:
  db:
    image: postgres:latest
    environment:
      POSTGRES_DB: mydatabase
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
    volumes:
      - postgres_data:/var/lib/postgresql/data

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

volumes:
  postgres_data:
```

Learning Experience:

+ How to define and configure multi-container applications using Docker Compose.

+ How to set up and run a Django application alongside other services like a database.

+ How to manage dependencies between services using Docker Compose's depends_on field.

+ How to access and interact with the Django application running in a Docker container from my local machine.

Conclusion:

Docker Compose simplifies the process of managing complex applications like Django by providing a straightforward way to define and run multi-container environments. By following the quickstart guide, you'll gain practical experience in using Docker Compose with Django, which will be invaluable for developing and deploying containerized Django applications in the future.