Launching a Odoo Container

How to launch your own odoo 13. Follow the step-by-step guide below to set up the necessary prerequisites and deploy the odoo13.

d **by dhruvraj**

Prerequisites

Before you begin, ensure that you have Docker and Docker Compose installed on your system. You can find installation instructions for Docker and docker compose .

Please complete this step before beginning.

1 .Create a Folder: Begin by making a new folder named "odoo_docker" on your desktop.

2. Organize Your Files: Gather all the necessary files such as docker-compose.yaml, odoo.conf, dump.sql, and the addons folder into the "odoo_docker" folder. Keep everything well-organized.

- refer the documentation and install docker and docker compose into your system

This is tested on version

- Docker version : 24.0.6
- Docker-compose : 1.22.0, build f46880fe

install docker :

```
sudo apt update
```

```
sudo apt install docker.io
```

```
sudo systemctl start docker
sudo systemctl enable docker
```

Download the Docker Compose binary:

```
sudo apt install docker-compose
```

Verify Docker Compose installation:

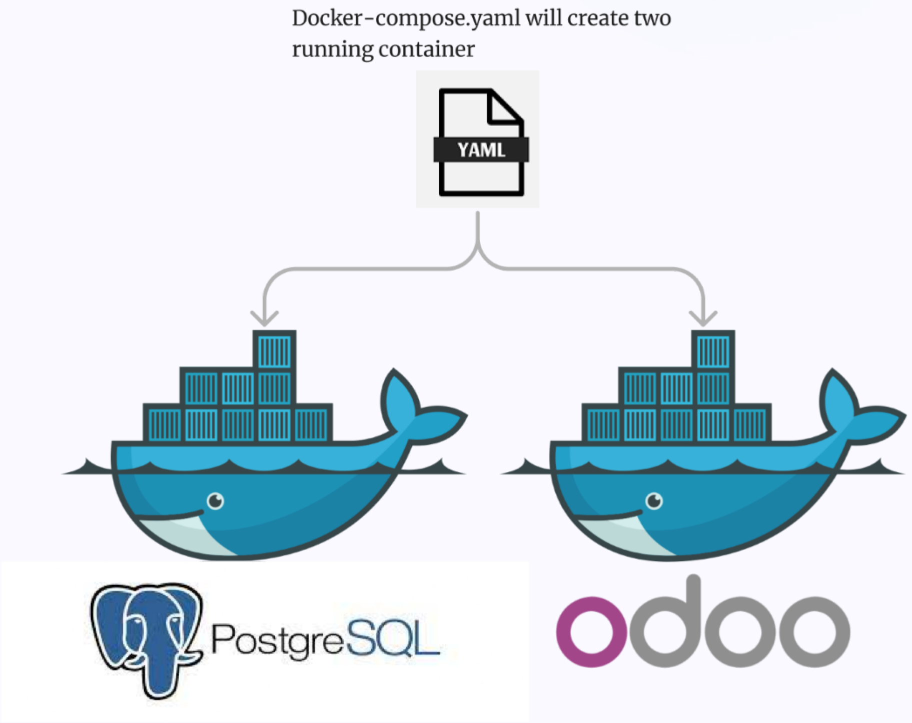
```
docker-compose --version
```

verify docker version :

```
docker --version
```

NOTE : Each project will come with its own docker-compose.yaml file, featuring different images. The provided docker-compose file below serves as a reference for understanding.

Steps



1. Get the docker-compose.yml file form the following source .
2. Place the docker-compose file into the desired directory with all the requirement like odoo addons folder and Postgres databasefile.sql
3. Below is the custom Docker Compose file required to set up Odoo 13 using Docker containers. We will explain each section and line in detail.

Docker-compose.yaml

You can customise the Docker Compose file according to your specific needs and configurations

```
# Use version 3 of Docker Compose syntax for better compatibility and features.
version: '3'

# Define the services (containers) to be run.
services:

 # PostgreSQL database service.
 db:
  image: postgres:14  # Use the PostgreSQL version 14 image from Docker Hub.
  container_name: postgresdb  # Set the container name to "postgresdb".
  environment:
   POSTGRES_USER: odoo  # Set PostgreSQL username to "odoo".
   POSTGRES_PASSWORD: odoo  # Set PostgreSQL password to "odoo".
  restart: always  # Restart the container always if it stops.
  volumes:
   - postgres_data:/var/lib/postgresql/data  # Mount a volume for PostgreSQL data storage.
  ports:
   - "5432:5432"  # Map host machine port 5432 to container port 5432 for PostgreSQL.

 # Odoo application service.
 odoo_homecare:
  image: custom image   # Use the specified image "custom image ".
  depends_on:
   - db  # Ensure the "odoo_homecare" service starts after the "db" service.
  container_name: odoo_custom  # Set the container name to "odoo_custom".
  ports:
   - "8069:8069"  # Map host machine port 8069 to container port 8069 for Odoo.
  environment:
   POSTGRES_USER: odoo       # Set PostgreSQL username to "odoo".
   POSTGRES_PASSWORD: odoo    # Set PostgreSQL password to "odoo".
   POSTGRES_HOST: db      # Set the hostname for the PostgreSQL server.
   POSTGRES_PORT: 5432     # Set the port for the PostgreSQL server.
  command: ["odoo", "-c", "/etc/odoo/odoo.conf"]  # Specify the command to run Odoo with the given configuration file.
  volumes:
   - odoo_data:/var/lib/odoo  # Mount a volume for Odoo data storage.
   - ./addons:/mnt/extra-addons  # Mount a volume for additional Odoo addons.
  restart: always  # Restart the container always if it stops.

# Define named volumes for persistent data storage.
volumes:
 postgres_data:  # Volume for PostgreSQL data.
 odoo_data:  # Volume for Odoo data.

# End of the Docker Compose file.
```

enableStart Your Application

In the same directory as your docker-compose.yml file, run the following command to start your services:

Copy command

```
docker-compose  up
```

After launching containers we will restore database into PostgreSQL container

1. **Copy the SQL Dump File to the Container:**

You need to copy your SQL dump file into the PostgreSQL container. You can use the docker cp command to do this. Replace your_dump_file.sql and container_name_or_id with your actual file and container information:

Copy code and run on terminal :

```
docker cp /path/to/your/database_dump.sql   your_postgres_container:/tmp/
```

In my scenario, since I have kept both my Docker Compose file (docker-compose.yaml) and my Dockerfile in the same directory, I have used the current directory, which is denoted by ./

```
docker cp ./dump.sql postgresdb:/tmp/
```

2.Access the PostgreSQL Container:

Access the PostgreSQL container by running an interactive shell:

Copy code

```
docker exec -it your_postgres_container bash
```

In my case :

```
docker exec -it postgresdb bash
```

3.Restore the Database Using psql:

Once inside the container, you can use the psql command to restore the plain text SQL dump. Replace your_database_name with the actual name of the database you want to restore and /tmp/database_dump.sql with the path to the dump file inside the container:

Copy code:

```
psql -U user_name_here  -d your_database_name -a -f /tmp/database_dump.sql
```

in my case , As i have set the user name to "odoo" in the compose file and the database name is also "odoo"

```
psql -U odoo -d odoo -a -f /tmp/dump.sql
```

In my case Here, -U postgres specifies the PostgreSQL superuser (you may need to change this if you're using a different user), -d your_database_name specifies the target database, -a specifies that the commands should be echoed, and -f specifies the input file to execute.

3.Exit the Container:

After the restoration is complete, exit the container:

Copy code:

```
exit
```

After Database restoration the container should be Restarted

To restart all container we can use this command in the same directory

```
docker-compose restart
```

Access Your Application

Your services are now running. You can access your web service at

```
http://localhost:8069/
```