

# Script to Dynamically Generate Docker Compose YAML Files

---

## Table of Contents

---

- [Description](#)
- [Problem Statement](#)
- [Prerequisites](#)
  - [Software Requirement](#)
  - [Hardware Requirement](#)
- [Implementation Steps](#)
  - [Step-1: Write a Python Script to Generate Docker Compose YAML](#)
  - [Step-2: Customize the Script for Dynamic Generation](#)
  - [Step-3: Execute the Script to Generate YAML](#)
- [References](#)

## Description

---

This document explains how to write a **Python script** that dynamically generates Docker Compose YAML files. The script allows you to define services, networks, and volumes in a programmatic way, making it easy to manage and scale multi-container applications like **TodoApp**.

## Problem Statement

---

Manually writing Docker Compose files can be time-consuming, especially when managing multiple services and environments. Dynamically generating YAML files through a script ensures consistency, flexibility, and automation in your infrastructure.

## Prerequisites

---

Completion of all previous lab guides (up to Lab Guide-08) is required before proceeding with Lab Guide-09.

### Software Required

- **Docker Desktop**: Installed on your Windows machine.
- **Python 3.x**: Installed on your system to run the script.
- **PyYAML Library**: Required for generating YAML files in Python.
- **TodoAPP\_MYSQL**: To download the source folder [click here](#)

### Hardware Requirement

- Minimum of 4 GB RAM
- At least 2 cores in the processor

# Implementation Steps

## Step-1: Write a Python Script to Generate Docker Compose YAML

First, install the required Python package **PyYAML** to manage YAML files:

```
pip install pyyaml
```

```
Administrator@8385453e15576 MINGW64 ~/Downloads/ToDoApp_MySQL-main/ToDoApp_MySQL-main
$ pip install pyyaml
Collecting pyyaml
  Obtaining dependency information for pyyaml from https://files.pythonhosted.org/packages/ed/23/8da0bbe2ab9dcdd11f4f4557ccaf95c10b9811b13ecced089d43ce59c3c8/PyYAML-6.0.2-cp311-cp311-win_amd64.whl.metadata
  Downloading PyYAML-6.0.2-cp311-cp311-win_amd64.whl.metadata (2.1 kB)
  Downloading PyYAML-6.0.2-cp311-cp311-win_amd64.whl (161 kB)
    162.0/162.0 kB 1.4 MB/s eta 0:00:00
Installing collected packages: pyyaml
Successfully installed pyyaml-6.0.2

[notice] A new release of pip is available: 23.2.1 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Next, create a Python script called `generate_compose.py` to dynamically generate a Docker Compose YAML file for your **ToDoApp**.

Note: Make sure to build that the docker image exists

### Sample Python Script

```
import yaml

def generate_compose():
    # Define the structure of the Docker Compose file
    compose_content = {
        'version': '3',
        'services': {
            'web': {
                'image': 'my_todoapp',
                'ports': ['8081:8081'],
                'networks': ['todoapp_network'],
            },
            'db': {
                'image': 'mysql:8.0',
                'environment': {
                    'MYSQL_ROOT_PASSWORD': 'rootpassword',
                    'MYSQL_DATABASE': 'todoapp_db',
                    'MYSQL_USER': 'root',
                    'MYSQL_PASSWORD': 'P@ssw0rd'
                },
                'networks': ['todoapp_network'],
                'volumes': ['db_data:/var/lib/mysql']
            },
        },
        'networks': {
            'todoapp_network': {
                'driver': 'bridge'
            }
        }
    }
```

```

        }
    },
    'volumes': {
        'db_data': {}
    }
}

# Write the Docker Compose content to a YAML file
with open('docker-compose.yml', 'w') as compose_file:
    yaml.dump(compose_content, compose_file, default_flow_style=False)

print("docker-compose.yml generated successfully!")

if __name__ == '__main__':
    generate_compose()

```

### Explanation:

- **version**: Defines the version of Docker Compose (3 in this case).
- **services**: Specifies the services (containers) to be deployed (**web** and **db** in this case).
  - **web**: Runs the **ToDoApp** on port **8081**.
  - **db**: Runs a MySQL container, providing the environment variables for database setup.
- **networks**: Defines a custom bridge network (**todoapp\_network**) to connect the containers.
- **volumes**: Creates a named volume (**db\_data**) to persist MySQL data.

---

## Step-2: Customize the Script for Dynamic Generation

You can modify the script to accept **dynamic inputs** such as service names, images, and ports. This allows you to customize the Docker Compose YAML file based on user input or environment variables.

### Example of Dynamic Customization:

```

import yaml

def generate_compose(app_name='my_todoapp', app_port=8081, db_name='mysql',
                    db_port=3306):
    compose_content = {
        'version': '3',
        'services': {
            'web': {
                'image': app_name,
                'ports': [f'{app_port}:{app_port}'],
                'networks': ['todoapp_network'],
            },
            'db': {
                'image': db_name,
                'environment': {
                    'MYSQL_ROOT_PASSWORD': 'P@ssw0rd',

```

```

        'MYSQL_DATABASE': 'todoapp_db',
        'MYSQL_USER': 'root',
        'MYSQL_PASSWORD': 'P@ssw0rd'
    },
    'networks': ['todoapp_network'],
    'volumes': ['db_data:/var/lib/mysql'],
    'ports': [f'{db_port}:{db_port}']
}

'networks': {
    'todoapp_network': {
        'driver': 'bridge'
    }
},
'volumes': {
    'db_data': {}
}
}

with open('docker-compose.yml', 'w') as compose_file:
    yaml.dump(compose_content, compose_file, default_flow_style=False)

print(f'docker-compose.yml generated with {app_name} and {db_name}!')

if __name__ == '__main__':
    generate_compose(app_name='my_todoapp', app_port=8081, db_name='mysql',
db_port=3306)

```

This version of the script allows you to specify the **application name**, **application port**, **database name**, and **database port** dynamically.

---

### Step-3: Execute the Script to Generate YAML

Once you've written the script, execute it to generate the **docker-compose.yml** file:

```
python generate_compose.py
```

```

Administrator@8385453e5e15576 MINGW64 ~/Downloads/ToDoApp_MySQL-main/ToDoApp_MySQL-main
$ python generate_compose.py
docker-compose.yml generated with my_todoapp and mysql!

Administrator@8385453e5e15576 MINGW64 ~/Downloads/ToDoApp_MySQL-main/ToDoApp_MySQL-main
$

```

This will generate a **docker-compose.yml** file in the current directory with the structure defined in the script. You can now use this file to manage your multi-container **ToDoApp** with Docker Compose.

---

### Sample Output (docker-compose.yml):

```

version: '3'
services:
  web:
    image: my_todoapp
    ports:
      - "8081:8081"
    networks:
      - todoapp_network
  db:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: P@ssw0rd
      MYSQL_DATABASE: todoapp_db
      MYSQL_USER: root
      MYSQL_PASSWORD: P@ssw0rd
    networks:
      - todoapp_network
    volumes:
      - db_data:/var/lib/mysql
    ports:
      - "3306:3306"

networks:
  todoapp_network:
    driver: bridge

volumes:
  db_data: {}

```

You can now run the following command to launch the application:

```
docker-compose up
```

```

Administrator@8305453e5e15576 MINGW64 ~/Downloads/ToDoApp_MySQL-main
$ cd ToDoApp_MySQL-main

Administrator@8305453e5e15576 MINGW64 ~/Downloads/ToDoApp_MySQL-main/ToDoApp_MySQL-main
$ python generate_compose.py
docker-compose.yml generated with my_todoapp and mysql!

Administrator@8305453e5e15576 MINGW64 ~/Downloads/ToDoApp_MySQL-main/ToDoApp_MySQL-main
$ docker-compose up
[+] Building 0.0s (0/0)
[+] Running 3/3
✓ Network todoapp_mysql-main_todoapp_network Created
✓ Container todoapp_mysql-main-web-1 Created
✓ Container todoapp_mysql-main-db-1 Created
Attaching to todoapp_mysql-main-db-1, todoapp_mysql-main-web-1
todoapp_mysql-main-db-1 | 2024-10-25 04:23:23+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 9.1.0-1.el9 started.
todoapp_mysql-main-db-1 | 2024-10-25 04:23:24+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
todoapp_mysql-main-db-1 | 2024-10-25 04:23:24+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 9.1.0-1.el9 started.
todoapp_mysql-main-db-1 | 2024-10-25 04:23:25+00:00 [ERROR] [Entrypoint]: MYSQL_USER="root", MYSQL_USER and MYSQL_PASSWORD are for configuring a regular
the root user
todoapp_mysql-main-db-1 | Remove MYSQL_USER="root" and use one of the following to control the root user password:
todoapp_mysql-main-db-1 | - MYSQL_ROOT_PASSWORD
todoapp_mysql-main-db-1 | - MYSQL_ALLOW_EMPTY_PASSWORD
todoapp_mysql-main-db-1 | - MYSQL_RANDOM_ROOT_PASSWORD
todoapp_mysql-main-web-1 |
todoapp_mysql-main-web-1 |
todoapp_mysql-main-web-1 |
todoapp_mysql-main-web-1 |
todoapp_mysql-main-web-1 |
todoapp_mysql-main-web-1 |
todoapp_mysql-main-web-1 |
todoapp_mysql-main-web-1 |

```

## References

- Docker Compose Documentation: <https://docs.docker.com/compose/>
- PyYAML Documentation: <https://pyyaml.org/>