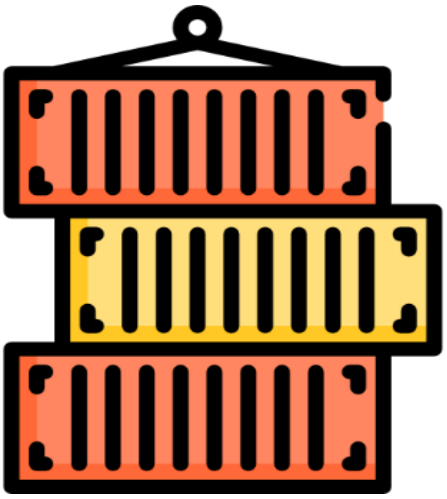

Containerized web application Deployment with Docker

Prepared By:
Mahmoud Alsalloum

Table of contents

Contents

Technology Stack

Ubuntu Server – Lightweight and stable base OS for server deployment

Python 3.12 – Runtime environment for MkDocs and Material theme

MkDocs – Static site generator for project and code documentation using Markdown

MkDocs Material Theme – Modern, responsive design with built-in light/dark mode support

Docker – Containerization of the web application for consistent and portable deployment

Systemd – Service manager to automatically start and monitor the Docker container on server boot

Project Overview

Welcome to my project!

I'm honored that you're visiting my portfolio website and taking the time to explore my work. In this project, I containerized an MkDocs-based documentation website using Docker, hosted on an Ubuntu Server. The goal was to create a consistent, portable, and easily maintainable environment for displaying my IT-related projects and code. To enhance usability and visual presentation, I integrated the Material theme, which supports responsive design and both light and dark modes. The application is currently accessible within my local network for development and personal use. To ensure that the service runs automatically after system restarts, I configured it with a Docker restart either policy or systemd integration. This setup enables me to continuously maintain and expand my documentation site with minimal manual effort, while laying the groundwork for future remote production deployments.

Implementation Steps

After setting up the Ubuntu Server environment, I began preparing the system for hosting the documentation platform. The first step was to ensure that pipx was installed, as shown in Figure 1.

```
~$ sudo apt install pipx
```

Figure 1: pipx installation

With the base tools ready, I used pipx to install MkDocs, a static site generator written in Python. Using pipx ensured that MkDocs and its dependencies were isolated from the system Python environment. See Figure 2.

```
mahmoud@ubuntu-server:~$ pipx install mkdocs
```

Figure 2: Install MkDocs

I then created a new MkDocs project using the mkdocs new command. This generated the basic folder structure, including the docs/ directory and the mkdocs.yml configuration file. See Figure 3.

```
mahmoud@ubuntu-server:~/forum-docs$ ls
docs  mkdocs.yml
```

Figure 3: Create mkdocs project

To enable Dark Mode switching, I edited the mkdocs.yml file and customized the color palette settings. This added a toggle button in the site's UI, allowing users to switch between light and dark themes dynamically. See Figure 4.

```
 GNU nano 7.2                                  mkdocs.yml *
site_name: My Forum Project
theme:
  name: material
  palette:
    - scheme: default
      primary: indigo
      accent: indigo
      toggle:
        icon: material/weather-night
        name: Switch to dark mode
    - scheme: slate
      primary: indigo
      accent: indigo
      toggle:
        icon: material/weather-sunny
        name: Switch to light mode
```

Figure 4: Configuration of Material theme with dark/light mode toggle

Once the MkDocs site was functional locally using mkdocs serve, I proceeded to containerize the application with Docker. First, I created a Docker file that uses a slim Python base image and installs MkDocs along with the Material theme. See Figure 5.

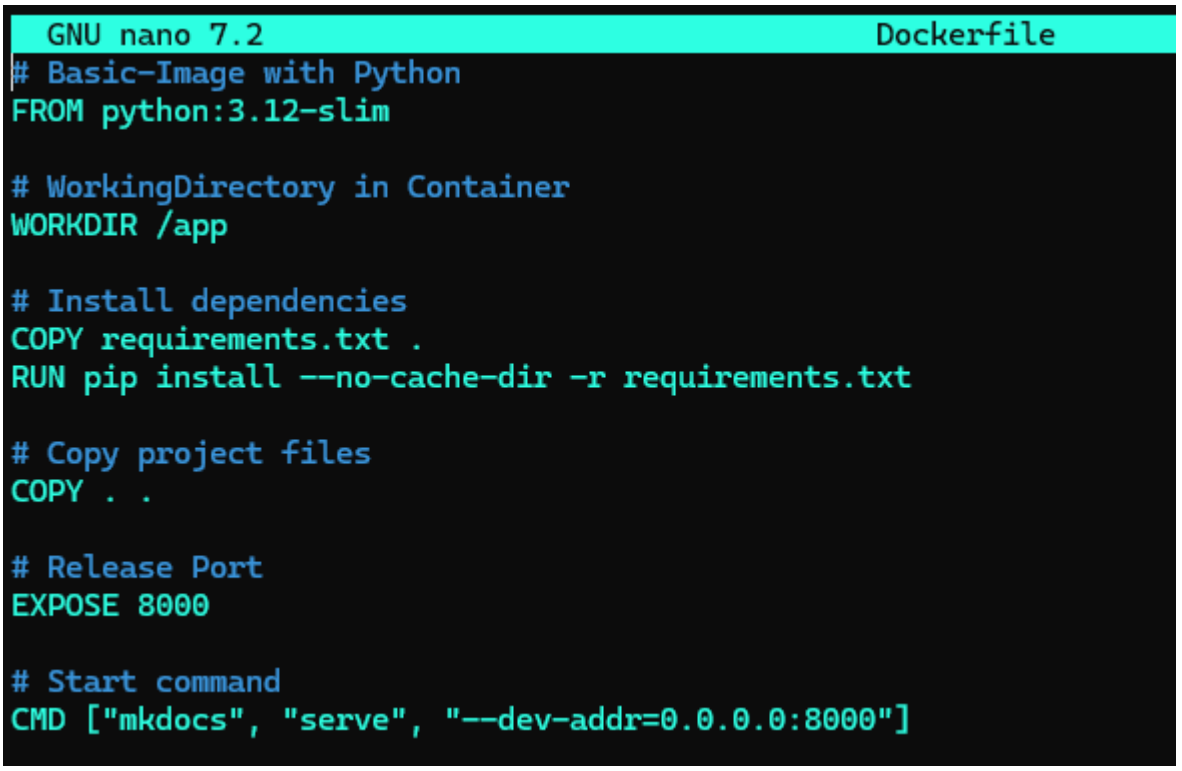

Figure 5: Docker file used for containerizing MkDocs

I also created a requirements.txt file to declare the dependencies explicitly for the Docker image. The Docker image was then built using docker build and tested with docker run. See Figure 6, 7 and 8.

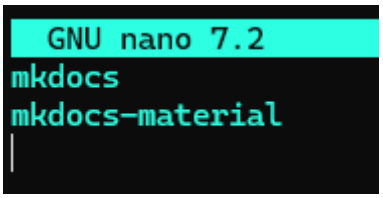

Figure 6: requirements.txt containing MkDocs dependencies

Figure 7: Building Docker container

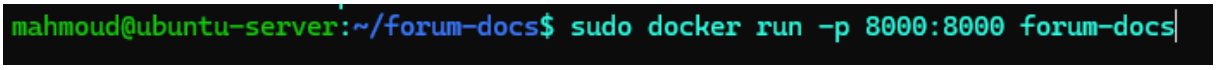

Figure 8: Running Docker container

To ensure that the container starts automatically after a reboot, I created a systemd service file under /etc/systemd/system/forum-docs.service. This service starts the Docker container on boot and monitors it continuously. See Figure 9.

```
  GNU nano 7.2                    /etc/systemd/system/forum-docs.service *
[Unit]
Description=Forum Docs Docker Container
After=network.target docker.service
Requires=docker.service

[Service]
Restart=always
ExecStart=/usr/bin/docker run --rm -p 8000:8000 --name forum-docs forum-docs
ExecStop=/usr/bin/docker stop forum-docs

[Install]
WantedBy=multi-user.target
```

Figure 9: Systemd service file for automatic container startup

After creating the systemd service file at /etc/systemd/system/forum-docs.service, I executed the necessary systemd commands to register, enable, and start the service. Specifically, I ran sudo systemctl daemon-reload to reload the systemd manager configuration, sudo systemctl enable forum-docs.service to enable automatic startup on boot, and sudo systemctl start forum-docs.service to launch the container immediately. This setup ensures that the documentation site is always available after a reboot without manual intervention.
See Figure 10.

```
mahmoud@ubuntu-server:~/forum-docs$ sudo systemctl daemon-reload
mahmoud@ubuntu-server:~/forum-docs$ sudo systemctl enable forum-docs.service
Created symlink /etc/systemd/system/multi-user.target.wants/forum-docs.service → /etc/systemd/system/f
orum-docs.service.
mahmoud@ubuntu-server:~/forum-docs$ sudo systemctl start forum-docs.service
```

Figure 10: Enabling and starting the MkDocs Docker service using systemd

I tested the site by entering the server ‘s IP address along with port 8000 in the browser: http://192.168.178.40:8000. See Figure 11.

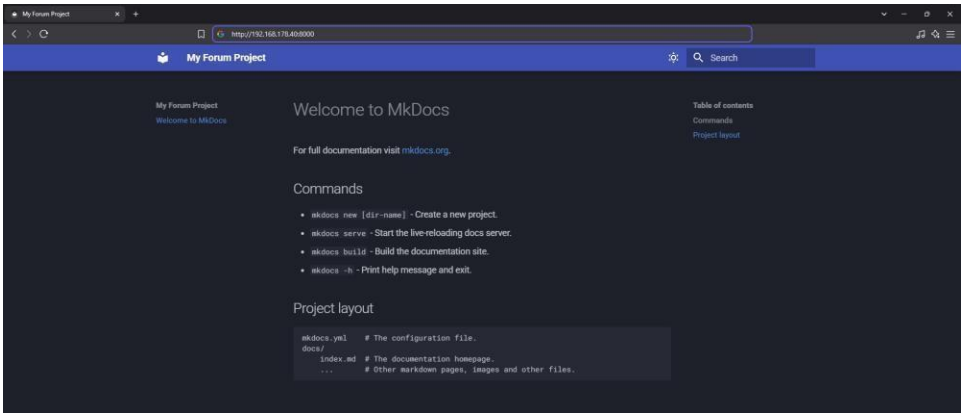

Figure 11: Accessing the MkDocs site via the server's IP address and port 8000