**Project Setup and Configuration Guide**

**Introduction**

This guide provides step-by-step instructions for setting up and running a Docker container for a Python web application using a base Python 3 Linux image. The project focuses on containerization principles, Docker image creation, and running a Python application within a containerized environment.

**Project Structure**

1. **Create a Project Directory:**

bashCopy code

mkdir my\_docker\_project cd my\_docker\_project

1. **Create Dockerfile:** Create a file named **Dockerfile** in the project directory.

DockerfileCopy code

FROM python:3 WORKDIR /app COPY requirements.txt /app/ RUN pip install --no-cache-dir -r requirements.txt COPY . /app/ ENV PYTHONUNBUFFERED 1 CMD ["python", "your\_app.py"]

1. **Application Dependencies:** If your application has dependencies, list them in a **requirements.txt** file.

textCopy code

Flask==2.0.1

1. **Build Docker Image:** Build the Docker image using the following command:

bashCopy code

docker build -t your\_username/your\_image\_name:tag .

1. **Run Docker Container:** Run the Docker container based on the created image:

bashCopy code

docker run -p 8080:8080 -e ENV\_VARIABLE=value your\_username/your\_image\_name:tag

Adjust the port and environment variables as needed.

1. **Test the Container:** Access your web application at [http://localhost:8080](http://localhost:8080/) (or the specified port). Ensure the application functions correctly.

**Challenges Faced and Lessons Learned**

**Port Binding Issues:**

If you encounter port binding issues, check for existing processes on the specified port. Use **netstat** or **docker ps** to identify and terminate conflicting processes.

bashCopy code

netstat -ano | findstr "8080" taskkill /F /PID <PID>

**ImportError with Werkzeug:**

If you face import errors related to Werkzeug, ensure compatibility between your application's requirements and the installed Werkzeug version in the Docker container. Update the **requirements.txt** file and rebuild the Docker image.

**Access Denied when Terminating Processes:**

If **taskkill** encounters "Access is denied," run the Command Prompt as Administrator and retry the command.

**Address Assignment Error in Flask:**

If Flask shows "Cannot assign requested address," ensure your Flask app binds to **'0.0.0.0'** in the development server. Adjust your Flask app's run command:

pythonCopy code

if \_\_name\_\_ == '\_\_main\_\_': app.run(host='0.0.0.0', port=8080)

**Conclusion**

This project provides hands-on experience with Docker and containerization concepts. It covers Dockerfile creation, image building, and running containerized applications. Addressing challenges enhances troubleshooting skills, and understanding the lessons learned ensures a smoother development process.