**Task 1: Create a Dockerfile for a Python web application.**

**Requirements:** Code for the application: robindarby/web-server.py The provided Python web application uses the “Flask” framework; it should have a single endpoint ( HTTP://localhost:8080 ) that displays a "Hello World!" message when accessed through a web browser.

1. Write a Dockerfile with steps to containerize the Python web application.

2. The Dockerfile should use an official Python image as the base image.

3. Install any required dependencies for the application within the Docker image.

4. Set the working directory within the container to the application's directory.

5. Expose the port on which the application runs within the Docker container ( 8080 ).

6. Specify the command to run the application when the container starts.

--------------------------

**Solution:**

**Dockerfile**

|  |
| --- |
| # Python image as the base image  FROM python:3.9-slim  # environment variables  ENV PYTHONDONTWRITEBYTECODE 1  ENV PYTHONUNBUFFERED 1  # Install dependencies  RUN pip install Flask  # Set the working directory within the container to the application's directory  WORKDIR /app  # Copy the application code into the container  COPY robindarby/web-server.py /app  # Expose the port on which the application runs within the Docker container (8080)  EXPOSE 8080  # Specify the command to run the application when the container starts  CMD ["python", "web-server.py"] |

**Few notes:**

1. Above Dockerfile used official Python image with the tag 3.9-slim.
2. Installs Flask, as it required dependency for the application.
3. Sets the working directory to /app within the container.
4. Copies the web-server.py file from robindarby directory to the /app directory in the container.
5. Exposes port 8080 to allow connections from outside the container.
6. Specifies the command to run the Python application (web-server.py) when the container starts