EXP – 6: Demonstrate Containerization with Docker.

Step 1: Install Docker Desktop

1. **Download Docker Desktop**:
   - Visit the Docker Desktop download page and download the installer for Windows.
2. **Run the Installer**:
   - Double-click the downloaded installer file and follow the setup wizard.
   - During installation, ensure that "Use the WSL 2-based engine" is selected.
3. **Restart Your System**:
   - After installation, restart your system if prompted.
4. **Enable WSL 2 (if not already enabled)**:
   - Open PowerShell as Administrator and run:

     ```
     wsl --install
     ```

   - If WSL 2 is already installed, make sure it's set as the default version:

     ```
     wsl --set-default-version 2
     ```

5. **Launch Docker Desktop**:
   - Open Docker Desktop and wait for it to start. Confirm that it's running by checking the Docker status in the taskbar or running:

     ```
     docker --version
     ```

Step 2: Verify Docker Installation

1. Open a terminal (PowerShell or Command Prompt).
2. Run:

   ```
   docker --version
   ```

   - This should return the Docker version installed.
3. Test Docker with:

   ```
   docker run hello-world
   ```

   - This downloads a test image and runs a container, verifying that Docker is working.

Step 3: Prepare Your Python Program

1. **Create a Python Script**:
   - Create a directory for your project, e.g., `my_python_app`.
   - Inside this directory, create your Python script, e.g., `app.py`.
2. **Create a `requirements.txt` File**:
   - List all Python dependencies in a file named `requirements.txt`. Example:

```
flask
requests
```

Step 4: Create a Dockerfile

1. Inside your project directory, create a file named `Dockerfile` (no file extension).
2. Add the following content:

```
Dockerfile

Copy code
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 5000 available to the world outside this container
EXPOSE 5000

# Define environment variable
ENV PYTHONUNBUFFERED=1

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Step 5: Build the Docker Image

1. Open a terminal in the project directory.
2. Run:

```
docker build -t my-python-app .
```

   - This creates a Docker image named `my-python-app`.

Step 6: Run the Docker Container

1. Start the container with:

   ```
   docker run -p 5000:5000 my-python-app
   ```

   - This maps port 5000 in the container to port 5000 on your machine.
2. Open your browser or use a tool like Postman to access the app:

   ```
   http://localhost:5000
   ```

Step 7: Manage Docker Containers

1. List all running containers:

   ```
   docker ps
   ```

2. Stop a container:

   ```
   docker stop <container_id>
   ```

3. Remove a container:

   ```
   docker rm <container_id>
   ```

Optional: Save and Share Your Image

1. Tag your image:

   ```
   docker tag my-python-app your-dockerhub-username/my-python-app
   ```

2. Push to Docker Hub:

   ```
   docker push your-dockerhub-username/my-python-app
   ```

You can now use this image on any machine with Docker installed.