3. Design, Deploy and manage a micro services architecture on your local machine using Docker and Docker-compose.

Step 1: Prerequisites

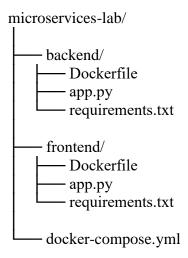
- 1. Install **Docker** on your machine (if not already installed).
 - o Download from: https://www.docker.com/products/docker-desktop
- 2. Install **Docker Compose** (if not already installed).
 - Docker Compose comes bundled with Docker Desktop, so you should have it if Docker is already installed.

Step 2: Create the Project Directory Structure

We will create a project directory that includes two services:

- 1. **BackendService** (Backend Flask service).
- 2. Frontend Service (Flask service).

Here's the directory structure:



Step 3: Define the Backend in Flask

Create the **Dockerfile** for the **backend** (Flask-based).

```
# Use Python 3.9 slim image
FROM python:3.9-slim

# Set the working directory
WORKDIR /app

# Copy requirements file and install dependencies
COPY requirements.txt /app/
RUN pip install --no-cache-dir -r requirements.txt
```

```
# Copy the rest of the application files
COPY . /app/

# Expose port 5000 for the Flask app
EXPOSE 5000

# Run the Flask application
CMD ["python", "app.py"]
```

Create the **requirements.txt** to specify the Python packages for Flask.

```
Flask==2.1.1
Werkzeug==2.0.3
```

Create the **app.py** file for the **backend** (Flask app). This service will simply return a message like "Hello, World!" when accessed.

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World! From Backend!'

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Step 4: Define the Frontend (Flask)

Create the **Dockerfile** for the **Frontend** (Flask-based). This service will make an HTTP request to the **backend** and display the result.

```
# Use Python 3.9 slim image
FROM python:3.9-slim

# Set the working directory
WORKDIR /app

# Copy requirements file and install dependencies
COPY requirements.txt /app/
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application files
COPY . /app/
```

```
# Expose port 5001 for the Flask app
EXPOSE 5001

# Run the Flask application
CMD ["python", "app.py"]
```

Create the **requirements.txt** file to specify the Python packages for Flask.

```
Flask==2.1.1
requests==2.26.0
```

Create the **app.py** file for the **Frontend** (Flask app). This service will call the **Hello Service** API and display the result.

```
from flask import Flask
import requests

app = Flask(__name__)

@app.route('/')
def hello_world():
    response = requests.get('http://hello-service:5000') # Communicate with hello-service
    return f"Frontend says: {response.text}"

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5001)
```

Step 5: Create the docker-compose.yml File

Now, let's define the **Docker Compose** file to manage both services (Backend and Frontend).

docker-compose.yml

```
version: '3.8'

services:

backend:
build:
context: ./backend # Path to the hello-service directory
ports:
- "5000:5000" # Expose the hello-service on port 5000
networks:
- app-network
```

```
frontend:
build:
context: ./frontend # Path to the frontend directory
ports:
- "5001:5001" # Expose the frontend on port 5001
networks:
- app-network
depends_on:
- backend # Ensure hello-service starts first

networks:
app-network:
driver: bridge
```

Step 6: Build and Run the Containers

Now that everything is set up, let's build and run the containers using Docker Compose.

- 1. Navigate to the project directory (microservices-lab).
- 2. Build and start the containers:

```
docker-compose up --build
```

This command will:

- o Build the Docker images for both **Backend** and **Frontend**.
- Start both services and connect them via the app-network.

Step 7: Test the Application

1. Open your browser and go to http://localhost:5001. You should see the message:

Frontend says: Hello, World! from frontend!

This means the **Frontend Service** successfully called the **Hello Service**.

2. If you visit http://localhost:5000 (directly hitting the **Backend**), you will see the message:

Hello, World! from Backend!

Step 8: Clean Up

Once you are done, you can stop the services by running:

```
docker-compose down
```