**Lesson 07 Demo 04**

**Applying Health Checks and Monitoring to Microservices**

**Objective:** To apply health checks and monitoring to microservices using Docker, Docker Compose, and a Prometheus image for enhanced reliability and observability

**Tools required:** Docker, Docker Compose, Python 3.x, and Flask

**Prerequisites:** None

Steps to be followed:

1. Create microservices
2. Create a requirements file for dependencies
3. Create a Dockerfile for each microservice
4. Create a Docker compose file and run the setup

**Step 1: Create microservices**

1. Switch to the root user using the following command: **sudo su  
     
   A green and black text

   Description automatically generated**
2. Create a directory for the microservices using the following command: **mkdir microservices-demo  
     
   **
3. Navigate inside the created directory using the following command:  
   **cd microservices-demo**  
     
   A black text on a white background

   Description automatically generated
4. Create two directories, **service-a** and **service-b**, for the respective microservices using the following commands: **mkdir service-a**

**mkdir service-b  
  
**

****

1. Create a Python file in the **service-a** directory using the following command:

**vi service-a/app.py**

****

1. Set up a simple Flask web application for **service-a** using the following code:

**from flask import Flask**

**app = Flask(\_\_name\_\_)**

**@app.route('/')**

**def hello\_world():**

**return 'Hello from Service A!'**

**@app.route('/health')**

**def health():**

**return 'OK', 200**

**if \_\_name\_\_ == '\_\_main\_\_':**

**start\_http\_server(8000) # Expose metrics on port 8000 on line 13**

**app.run(host='0.0.0.0', port=5000)**

A computer screen shot of a computer code

Description automatically generated

1. Create a Python file in the **service-b** directory using the following command:

**vi service-b/app.py**



1. Set up a simple Flask web application for **service-b** using the following code:

**from flask import Flask**

**app = Flask(\_\_name\_\_)**

**@app.route('/')**

**def hello\_world():**

**return 'Hello from Service B!'**

**@app.route('/health')**

**def health():**

**return 'OK', 200**

**if \_\_name\_\_ == '\_\_main\_\_':**

**start\_http\_server(8000) # Expose metrics on port 8000 on line 13**

**app.run(host='0.0.0.0', port=5000)**

A computer code with red and blue text

Description automatically generated

**Step 2: Create a requirements file for dependencies**

1. Create a requirements file in the **service-a** directory using the following command:

**vi service-a/requirements.txt**

****

1. Specify the Flask version required for **service-a** using the following code:

**Flask==2.0.1**



1. Create a requirements file in the **service-b** directory using the following command:

**vi service-b/requirements.txt**

****

1. Specify the Flask version required for the **service-b** using the following code:

**Flask==2.0.1**

A white background with black dots

Description automatically generated

**Step 3: Create a Dockerfile for each microservice**

1. Create a Dockerfile in the **service-a** directory using the following command:

**vi service-a/Dockerfile**



1. Enter the following script in the created **Dockerfile** file:

**FROM python:3.8-slim**

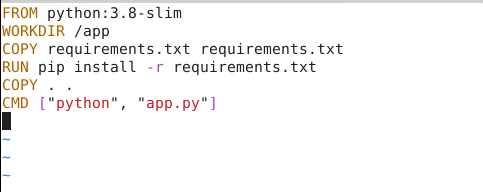
**WORKDIR /app**

**COPY requirements.txt requirements.txt**

**RUN pip install -r requirements.txt**

**COPY . .**

**CMD ["python", "app.py"]**



1. Create a Dockerfile in the **service-b** directory using the following command:

**vi service-b/Dockerfile**



1. Enter the following code in the created **Dockerfile** file:

**FROM python:3.8-slim**

**WORKDIR /app**

**COPY requirements.txt requirements.txt**

**RUN pip install -r requirements.txt**

**COPY . .**

**CMD ["python", "app.py"]**

A screen shot of a computer code

Description automatically generated

**Step 4: Create a Docker compose file and run the setup**

1. Create the **docker-compose** file using the following command:

**vi docker-compose.yml**

****

1. Add the following script in the created **docker-compose** file:

**version: '3.8'**

**services:**

**service-a:**

**build: ./service-a**

**ports:**

**- "5000:5000"**

**healthcheck:**

**test: ["CMD", "curl", "-f", "http://localhost:5000/health"]**

**interval: 30s**

**timeout: 10s**

**retries: 3**

**service-b:**

**build: ./service-b**

**ports:**

**- "5001:5000"**

**healthcheck:**

**test: ["CMD", "curl", "-f", "http://localhost:5001/health"]**

**interval: 30s**

**timeout: 10s**

**retries: 3**

**prometheus:**

**image: prom/prometheus**

**volumes:**

**- ./prometheus/prometheus.yml:/etc/prometheus/prometheus.yml**

**ports:**

**- "9090:9090"**

**grafana:**

**image: grafana/grafana**

**ports:**

**- "3000:3000"**

**volumes:**

**- grafana-data:/var/lib/grafana**

**volumes:**

**grafana-data:**

**A screenshot of a computer program

Description automatically generated**

1. Run the setup by using the following command:

**docker-compose up --build**

A screenshot of a computer

Description automatically generated

A close-up of a text

Description automatically generated

1. Open the following URL in your default browser:

**localhost:9090**

A screenshot of a computer

Description automatically generated

By following these steps, you have successfully applied health checks and monitoring to microservices using Docker, Docker Compose, and Prometheus for enhanced reliability and observability.