**Simplified Practical Example: Implementing the Model Training Service**

**Step 1: Set Up the Project Structure**

1. **Create a Project Directory**

bash

Copy code

mkdir model-training-service

cd model-training-service

1. **Set Up a Python Virtual Environment**

bash

Copy code

python -m venv venv

source venv/bin/activate # On Windows, use `venv\Scripts\activate`

1. **Install Flask**

bash

Copy code

pip install flask

1. **Create Project Files**
   * Create a file named app.py for the Flask application.

**Step 2: Create the API Endpoint**

1. **Define the Flask App and Endpoint in app.py**

python

Copy code

from flask import Flask, request, jsonify

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

@app.route('/trainModel', methods=['POST'])

def train\_model():

# Extract training data and parameters from request

data = request.json['data']

params = request.json['params']

# Simplified model training logic (dummy implementation)

model = train\_dummy\_model(data, params)

# Return dummy model artifact and metrics

response = {

'model\_artifact': 'dummy\_model\_path',

'training\_metrics': {'accuracy': 0.95}

}

return jsonify(response)

def train\_dummy\_model(data, params):

# Dummy model training logic

print("Training data:", data)

print("Training parameters:", params)

# Here you would add your actual ML training code

return "dummy\_model"

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

app.run(debug=True)

**Step 3: Run the Flask Application**

1. **Start the Flask Server**

bash

Copy code

python app.py

1. **Test the Endpoint**
   * Use a tool like curl or Postman to send a POST request to http://127.0.0.1:5000/trainModel.

**Example POST Request Using curl**

bash

Copy code

curl -X POST http://127.0.0.1:5000/trainModel -H "Content-Type: application/json" -d '{"data": "sample\_data", "params": {"learning\_rate": 0.01}}'

**Expected Response**

json

Copy code

{

"model\_artifact": "dummy\_model\_path",

"training\_metrics": {

"accuracy": 0.95

}

}

**Step 4: Dockerize the Application**

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

dockerfile

Copy code

# Use an official Python runtime as a parent image

FROM python:3.8-slim

# Set the working directory in the container

WORKDIR /app

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

COPY . /app

# Install Flask

RUN pip install flask

# Make port 5000 available to the world outside this container

EXPOSE 5000

# Define environment variable

ENV NAME ModelTrainingService

# Run app.py when the container launches

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

1. **Build and Run the Docker Container**
   * **Build the Docker Image**

bash

Copy code

docker build -t model-training-service .

* + **Run the Docker Container**

bash

Copy code

docker run -p 5000:5000 model-training-service

1. **Test the Dockerized Service**
   * Repeat the POST request to http://127.0.0.1:5000/trainModel to ensure the service works inside the Docker container.

**Step 5: Add Basic Security and Authentication**

1. **Install Flask-HTTPAuth**

bash

Copy code

pip install Flask-HTTPAuth

1. **Modify app.py to Add Authentication**

python

Copy code

from flask import Flask, request, jsonify

from flask\_httpauth import HTTPBasicAuth

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

auth = HTTPBasicAuth()

# Dummy user data for authentication

users = {

"user": "password"

}

@auth.get\_password

def get\_pw(username):

if username in users:

return users.get(username)

return None

@app.route('/trainModel', methods=['POST'])

@auth.login\_required

def train\_model():

data = request.json['data']

params = request.json['params']

model = train\_dummy\_model(data, params)

response = {

'model\_artifact': 'dummy\_model\_path',

'training\_metrics': {'accuracy': 0.95}

}

return jsonify(response)

def train\_dummy\_model(data, params):

print("Training data:", data)

print("Training parameters:", params)

return "dummy\_model"

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

app.run(debug=True)

1. **Test the Authenticated Endpoint**
   * Use curl or Postman with authentication.

bash

Copy code

curl -u user:password -X POST http://127.0.0.1:5000/trainModel -H "Content-Type: application/json" -d '{"data": "sample\_data", "params": {"learning\_rate": 0.01}}'

By following these steps, you will have a basic yet functional Model Training Service implemented with Flask, Dockerized for deployment, and secured with basic authentication. This setup can be extended and integrated with other MLOps components such as model deployment and monitoring services to form a complete SOA-based MLOps solution.