Project Containerization and Deployment Documentation

Task 1: Containerize your Project

1.1 Dockerization

1.1.1 Dockerfile

Create a Dockerfile in the root directory of your project to containerize it. The Dockerfile should include the necessary steps to build and run your microservices using Flask or Express.

dockerfileCopy code

```
# Dockerfile for Flask-based microservice FROM python:3.9-slim as builder WORKDIR /app COPY
requirements.txt . RUN python -m venv venv && \ . venv/bin/activate && \ pip install -r
requirements.txt COPY . . # Development stage FROM builder as development CMD ["python", "app.py"]
# Production stage FROM python:3.9-slim as production WORKDIR /app COPY --from=builder /app /app
CMD ["python", "app.py"]
```

1.1.2 Docker Compose

Create a basic **docker-compose.yml** file to orchestrate the services locally.

yamlCopy code

```
version: '3' services: api-py: build: context: . target: development ports: - "5000:5000" volumes: - .:/app
api-ts: build: context: ./api-ts ports: - "3000:3000" volumes: - ./api-ts:/app
```

Task 2: Deploy project on Kubernetes

2.1 Helm Charts

Create basic Helm charts to facilitate the deployment of your microservices on Kubernetes.

2.1.1 Helm Chart for api-py

- Create **api-py** directory in the root of your project.
- Inside **api-py** directory, create necessary Helm chart files (**Chart.yaml**, **values.yaml**, **deployment.yaml**, etc.).

2.1.2 Helm Chart for api-ts

- Create **api-ts** directory in the root of your project.
- Inside **api-ts** directory, create necessary Helm chart files (**Chart.yaml**, **values.yaml**, **deployment.yaml**, etc.).

2.2 Minikube

Use Minikube to test the deployment locally.

bashCopy code

minikube start minikube kubectl -- apply -f path/to/api-py/helm-chart minikube kubectl -- apply -f path/to/api-ts/helm-chart

Task 3: Deploy project on any Cloud

3.1 Terraform

Use Terraform to create a basic Kubernetes cluster on a cloud provider.

- Create a **terraform** directory.
- Inside the **terraform** directory, define necessary Terraform files (**main.tf**, **variables.tf**, **outputs.tf**, etc.).

3.2 Deploy on Cloud

Deploy your application on the cloud using the Helm charts created in Task 2.

Task 4: Set up CI/CD Jobs

4.1 CI/CD Tool

Choose a CI/CD tool (e.g., GitHub Actions, Jenkins).

4.2 Job Configuration

Configure jobs in your chosen CI/CD tool to perform the following tasks:

- Build your project
- Perform static analysis using SAST/linter tool (e.g., using semgrep)
- Build Dockerfile
- Deploy to the cloud Kubernetes cluster created in Task 3

This documentation provides a step-by-step guide for containerizing, deploying, and setting up CI/CD for your microservices-based project. Follow the instructions outlined in each section to achieve a successful implementation.