**Kubernetes Deployment Guide for Flask App and Logger**

**Scenario**

A company is develop-ing a **Medical Billing System** that exposes REST APIs to manage billing records. The system consists of two components:

1. **Flask App (app.py)** – Provides REST API endpoints for billing management.
2. **Logger Service (logger.py)** – Monitors application logs in real time.

To ensure **scalability, reliability, and automation**, the application needs to be deployed on **Kubernetes**. The deployment should allow for:

* **Containerization**: Package both the Flask app and logger into Docker containers.
* **Private Docker Registry**: Store and manage container images in a local registry.
* **Deployment in Kubernetes**: Deploy and manage the containers in a Kubernetes cluster.
* **Networking**: Expose the Flask app as a service while keeping the logger internal.
* **Scalability**: Allow the system to scale by increasing replicas.
* **Logging**: Enable centralized logging using a shared volume for logs.
* **Service Exposure**: Allow external access to the Flask app through a Kubernetes NodePort service.

**Solution Approach**

**Step 1: Setup a Local Private Docker Registry**

1. **Start a local registry container** to store images:
2. docker run -d -p 5000:5000 --name registry registry:2
3. **Verify the registry is running:**
4. docker ps | grep registry

**Step 2: Containerizing the Applications**

**Create a Dockerfile for Flask App (app.py)**

# Use official Python image

FROM python:3.9

# Set working directory

WORKDIR /app

# Copy application files

COPY app.py /app/

# Install dependencies

RUN pip install flask

# Create a log directory

RUN mkdir -p /logs && chmod 777 /logs

# Expose application port

EXPOSE 80

# Run Flask application

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

**Create a Dockerfile for the Logger (logger.py)**

# Use a minimal Python image

FROM python:3.9

# Set working directory

WORKDIR /logger

# Copy the logger script

COPY logger.py /logger/

# Create a log directory

RUN mkdir -p /logs && chmod 777 /logs

# Run log monitoring script

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

**Step 3: Build and Push Docker Images to Local Registry**

1. **Build Docker Images:**

docker build -t flask-app -f Dockerfile .

docker build -t log-monitor -f Dockerfile.logger .

1. **Tag Images for Local Registry:**

docker tag flask-app localhost:5000/flask-app

docker tag log-monitor localhost:5000/log-monitor

1. **Push Images to Local Registry:**

docker push localhost:5000/flask-app

docker push localhost:5000/log-monitor

1. **Verify Images in Registry:**

curl http://localhost:5000/v2/\_catalog

**Step 4: Kubernetes Deployment YAML Files**

**Deploy Flask Application and Logger (fapp-and-logger.yaml)**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: flask-app

spec:

  replicas: 1

  selector:

    matchLabels:

      app: flask-app

  template:

    metadata:

      labels:

        app: flask-app

    spec:

      containers:

      - name: flask-app

        image: localhost:5000/flask-app:latest  # Pull from local registry

        ports:

        - containerPort: 80

        volumeMounts:

        - name: log-volume

          mountPath: /logs  # Shared log directory

      - name: logger

        image: localhost:5000/log-monitor:latest  # Pull from local registry

        volumeMounts:

        - name: log-volume

          mountPath: /logs  # Shared log directory

      volumes:

      - name: log-volume

        emptyDir: {}  # Shared volume for logs

**Expose Flask App as a Service (flask-app-service.yaml)**

apiVersion: v1

kind: Service

metadata:

name: flask-service

spec:

selector:

app: flask-app

ports:

- protocol: TCP

port: 80

targetPort: 80

nodePort: 30001

type: NodePort

**Step 5: Deploying on Kubernetes**

1. **Apply the Deployments & Services**

kubectl apply -f fapp-and-logger.yaml

kubectl apply -f flask-app-service.yaml

1. **Verify Pods and Services**

kubectl get pods

kubectl get services

1. **Access the Flask App**

kubectl get services

Otherwise, access at:

http://<Node-IP>:30001

**Step 6: Monitoring and Cleanup (Optional)**

* **Monitor Logs from Logger**

kubectl logs -f deployment/logger

* **Scale Up the Flask App**

kubectl scale deployment flask-app --replicas=3

* **Delete the Deployment**

kubectl delete -f flask-app.yaml

kubectl delete -f logger.yaml

* **Delete All Resources**

kubectl delete all --all