Step-by-Step: FastAPI on Kubernetes

Step 1: Create a minimal FastAPI app

app/main.py

```
from fastapi import FastAPI, Request

app = FastAPI()

@app.get("/health")
def health():
    return {"status": "ok"}

@app.post("/minio-event")
async def minio_event(request: Request):
    event = await request.json()
    print(event)  # replace with real logic
    return {"status": "received"}
```

requirements.txt

```
fastapi
uvicorn[standard]
```

Step 2: Create Dockerfile

Dockerfile

```
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
```

```
RUN pip install --no-cache-dir -r requirements.txt

COPY app/ app/

EXPOSE 8000

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Build & push:

```
docker build -t myrepo/fastapi-minio-webhook:1.0 .
docker push myrepo/fastapi-minio-webhook:1.0
```

Step 3: Kubernetes Deployment

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: fastapi-webhook
spec:
  replicas: 2
  selector:
    matchLabels:
      app: fastapi-webhook
  template:
    metadata:
      labels:
        app: fastapi-webhook
    spec:
      containers:
      - name: fastapi
        image: myrepo/fastapi-minio-webhook:1.0
        ports:
        - containerPort: 8000
        resources:
```

```
          requests:
            cpu: "100m"
            memory: "128Mi"
          limits:
            cpu: "500m"
            memory: "512Mi"
        readinessProbe:
          httpGet:
            path: /health
            port: 8000
          initialDelaySeconds: 5
          periodSeconds: 10
        livenessProbe:
          httpGet:
            path: /health
            port: 8000
          initialDelaySeconds: 15
          periodSeconds: 20
```

Apply:

```
kubectl apply -f deployment.yaml
```

Step 4: Kubernetes Service (ClusterIP)

MinIO will call this **inside the cluster**.

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: fastapi-webhook
spec:
  selector:
    app: fastapi-webhook
  ports:
```

```
  - port: 8000
    targetPort: 8000
  type: ClusterIP
```

Apply:

```
kubectl apply -f service.yaml
```

Service DNS (important 👇):

```
http://fastapi-webhook.default.svc.cluster.local:8000
```

Step 5: Test inside the cluster

```
kubectl run curltest --rm -it \
  --image=curlimages/curl -- sh

curl http://fastapi-webhook:8000/health
```

Expected:

```
{"status":"ok"}
```

Step 6: Configure MinIO webhook endpoint

In MinIO config:

```
endpoint=http://fastapi-webhook:8000/minio-event
```

If MinIO is in a different namespace:

```
http://fastapi-webhook.<namespace>.svc.cluster.local:8000/minio-event
```

Step 7: Logs & Debugging

```
kubectl get pods
kubectl logs deploy/fastapi-webhook
```

Optional (Highly Recommended in Prod)

Add Auth Header Validation

```
from fastapi import Header, HTTPException

@app.post("/minio-event")
async def minio_event(
    request: Request,
    x_token: str = Header(None)
):
    if x_token != "super-secret":
        raise HTTPException(status_code=401)
```

Then set header in MinIO webhook config.

Scale Automatically (HPA)

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: fastapi-webhook-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: fastapi-webhook
  minReplicas: 2
  maxReplicas: 10
```

```
metrics:
- type: Resource
  resource:
    name: cpu
    target:
      type: Utilization
      averageUtilization: 70
```

Expose Externally (only if needed)

- Ingress (NGINX / Traefik)
- NodePort / LoadBalancer (less common for webhooks)
