**Deployment Guide – User Service (Docker & Kubernetes)**

**1) Prerequisites**

* Docker 24+
* kubectl 1.27+
* Kubernetes cluster (minikube/kind/EKS/GKE/AKS)
* Optional: NGINX Ingress Controller, Redis (rate limiting), Prometheus/Grafana

**2) Docker**

**2.1 Dockerfile (multi-stage, JDK 17)**

# userservice/Dockerfile

FROM maven:3.9.6-eclipse-temurin-17 AS build

WORKDIR /app

COPY pom.xml .

RUN mvn -q -e -DskipTests dependency:go-offline

COPY src ./src

RUN mvn -q -e -DskipTests package

FROM eclipse-temurin:17-jre

WORKDIR /opt/app

ENV JAVA\_OPTS="-XX:MaxRAMPercentage=75.0 -Djava.security.egd=file:/dev/./urandom"

COPY --from=build /app/target/userservice-\*.jar app.jar

EXPOSE 8080

HEALTHCHECK --interval=30s --timeout=3s --start-period=30s CMD curl -fsS http://localhost:8080/actuator/health || exit 1

ENTRYPOINT ["sh","-c","java $JAVA\_OPTS -jar /opt/app/app.jar"]

**2.2 Local .env (example)**

SPRING\_PROFILES\_ACTIVE=prod

DB\_URL=jdbc:postgresql://postgres:5432/users

DB\_USER=usersvc

DB\_PASSWORD=change\_me

JWT\_SECRET=super\_secret

**2.3 docker-compose.yml (local dev)**

version: "3.9"

services:

postgres:

image: postgres:15

environment:

POSTGRES\_DB: users

POSTGRES\_USER: usersvc

POSTGRES\_PASSWORD: change\_me

ports: ["5432:5432"]

healthcheck:

test: ["CMD-SHELL","pg\_isready -U usersvc -d users"]

interval: 10s

timeout: 3s

retries: 10

redis:

image: redis:7

ports: ["6379:6379"]

userservice:

build: ./userservice

depends\_on:

postgres:

condition: service\_healthy

env\_file: .env

environment:

SPRING\_DATASOURCE\_URL: ${DB\_URL}

SPRING\_DATASOURCE\_USERNAME: ${DB\_USER}

SPRING\_DATASOURCE\_PASSWORD: ${DB\_PASSWORD}

JWT\_SECRET: ${JWT\_SECRET}

SERVER\_PORT: 8080

ports: ["8080:8080"]

gateway:

image: your-org/api-gateway:latest

environment:

REDIS\_HOST: redis

GATEWAY\_JWT\_SECRET: ${JWT\_SECRET}

ports: ["8081:8080"]

depends\_on:

- userservice

- redis

**Build & run:**

docker compose build

docker compose up -d

curl http://localhost:8080/actuator/health

**3) Kubernetes Manifests**

Put these into k8s/ and apply with kubectl apply -f k8s/.

**3.1 Namespace**

# k8s/00-namespace.yaml

apiVersion: v1

kind: Namespace

metadata:

name: user-platform

**3.2 Secrets (DB creds, JWT)**

# k8s/01-secret.yaml

apiVersion: v1

kind: Secret

metadata:

name: userservice-secrets

namespace: user-platform

type: Opaque

stringData:

DB\_URL: jdbc:postgresql://postgres.user-platform.svc.cluster.local:5432/users

DB\_USER: usersvc

DB\_PASSWORD: change\_me

JWT\_SECRET: super\_secret

**3.3 ConfigMap (app config)**

# k8s/02-configmap.yaml

apiVersion: v1

kind: ConfigMap

metadata:

name: userservice-config

namespace: user-platform

data:

SPRING\_PROFILES\_ACTIVE: prod

SERVER\_PORT: "8080"

**3.4 Postgres (dev-only; use managed DB in prod)**

# k8s/03-postgres.yaml

apiVersion: v1

kind: Service

metadata:

name: postgres

namespace: user-platform

spec:

ports:

- port: 5432

selector:

app: postgres

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: postgres

namespace: user-platform

spec:

replicas: 1

selector:

matchLabels: { app: postgres }

template:

metadata:

labels: { app: postgres }

spec:

containers:

- name: postgres

image: postgres:15

env:

- name: POSTGRES\_DB

value: users

- name: POSTGRES\_USER

valueFrom:

secretKeyRef:

name: userservice-secrets

key: DB\_USER

- name: POSTGRES\_PASSWORD

valueFrom:

secretKeyRef:

name: userservice-secrets

key: DB\_PASSWORD

ports:

- containerPort: 5432

readinessProbe:

exec:

command: ["pg\_isready","-U","usersvc","-d","users"]

initialDelaySeconds: 10

periodSeconds: 10

**3.5 User Service – Deployment & Service**

# k8s/10-userservice.yaml

apiVersion: v1

kind: Service

metadata:

name: userservice

namespace: user-platform

spec:

selector: { app: userservice }

ports:

- name: http

port: 8080

targetPort: 8080

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: userservice

namespace: user-platform

spec:

replicas: 2

selector:

matchLabels: { app: userservice }

template:

metadata:

labels: { app: userservice }

spec:

containers:

- name: userservice

image: your-registry/userservice:1.0.0

imagePullPolicy: IfNotPresent

ports:

- containerPort: 8080

envFrom:

- configMapRef:

name: userservice-config

- secretRef:

name: userservice-secrets

env:

- name: SPRING\_DATASOURCE\_URL

valueFrom:

secretKeyRef:

name: userservice-secrets

key: DB\_URL

- name: SPRING\_DATASOURCE\_USERNAME

valueFrom:

secretKeyRef:

name: userservice-secrets

key: DB\_USER

- name: SPRING\_DATASOURCE\_PASSWORD

valueFrom:

secretKeyRef:

name: userservice-secrets

key: DB\_PASSWORD

- name: JWT\_SECRET

valueFrom:

secretKeyRef:

name: userservice-secrets

key: JWT\_SECRET

readinessProbe:

httpGet: { path: /actuator/health/readiness, port: 8080 }

initialDelaySeconds: 15

periodSeconds: 10

failureThreshold: 6

livenessProbe:

httpGet: { path: /actuator/health/liveness, port: 8080 }

initialDelaySeconds: 30

periodSeconds: 10

failureThreshold: 6

resources:

requests: { cpu: "100m", memory: "256Mi" }

limits: { cpu: "500m", memory: "512Mi" }

**3.6 API Gateway (Spring Cloud Gateway) – with Redis Rate Limiting**

# k8s/20-gateway.yaml

apiVersion: v1

kind: Service

metadata:

name: gateway

namespace: user-platform

spec:

selector: { app: gateway }

ports:

- name: http

port: 8080

targetPort: 8080

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: gateway

namespace: user-platform

spec:

replicas: 2

selector:

matchLabels: { app: gateway }

template:

metadata:

labels: { app: gateway }

spec:

containers:

- name: gateway

image: your-registry/api-gateway:1.0.0

env:

- name: REDIS\_HOST

value: redis

- name: GATEWAY\_JWT\_SECRET

valueFrom:

secretKeyRef:

name: userservice-secrets

key: JWT\_SECRET

ports:

- containerPort: 8080

readinessProbe:

httpGet: { path: /actuator/health, port: 8080 }

initialDelaySeconds: 10

resources:

requests: { cpu: "100m", memory: "128Mi" }

limits: { cpu: "500m", memory: "512Mi" }

---

apiVersion: v1

kind: Service

metadata:

name: redis

namespace: user-platform

spec:

selector: { app: redis }

ports:

- port: 6379

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: redis

namespace: user-platform

spec:

replicas: 1

selector:

matchLabels: { app: redis }

template:

metadata:

labels: { app: redis }

spec:

containers:

- name: redis

image: redis:7

ports: [{ containerPort: 6379 }]

**3.7 Ingress (NGINX)**

# k8s/30-ingress.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: user-platform

namespace: user-platform

annotations:

kubernetes.io/ingress.class: nginx

nginx.ingress.kubernetes.io/proxy-body-size: "10m"

spec:

rules:

- host: users.local

http:

paths:

- path: /user

pathType: Prefix

backend:

service:

name: userservice

port:

number: 8080

- path: /

pathType: Prefix

backend:

service:

name: gateway

port:

number: 8080

**3.8 Autoscaling (HPA)**

# k8s/40-hpa.yaml

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: userservice-hpa

namespace: user-platform

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: userservice

minReplicas: 2

maxReplicas: 6

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 60

**3.9 Network Policy (optional, restrict DB access)**

# k8s/50-networkpolicy.yaml

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: postgres-allow-from-userservice

namespace: user-platform

spec:

podSelector:

matchLabels: { app: postgres }

ingress:

- from:

- podSelector:

matchLabels: { app: userservice }

ports:

- protocol: TCP

port: 5432

**4) CI/CD Notes**

* Build and push images: docker build -t your-registry/userservice:1.0.0 . && docker push ...
* Use image tags or Git SHA for immutable deploys
* Apply manifests in order:

kubectl apply -f k8s/00-namespace.yaml

kubectl apply -f k8s/01-secret.yaml

kubectl apply -f k8s/02-configmap.yaml

kubectl apply -f k8s/03-postgres.yaml

kubectl apply -f k8s/10-userservice.yaml

kubectl apply -f k8s/20-gateway.yaml

kubectl apply -f k8s/30-ingress.yaml

kubectl apply -f k8s/40-hpa.yaml

kubectl apply -f k8s/50-networkpolicy.yaml

**Sanity checks:**

kubectl -n user-platform get pods,svc,ingress

kubectl -n user-platform logs deploy/userservice

kubectl -n user-platform port-forward svc/userservice 8080:8080

curl http://localhost:8080/actuator/health

**5) Production Hardening Checklist**

* Use managed Postgres (AWS RDS/Aurora, Cloud SQL)
* Enable HTTPS (cert-manager + Let’s Encrypt) on Ingress
* Externalize secrets (Sealed Secrets / External Secrets)
* Resource requests/limits tuned via load tests
* Rolling updates, PodDisruptionBudgets, anti-affinity
* Structured logs, tracing (OpenTelemetry), metrics (Actuator + Prometheus)

**6) Optional: Helm/Kustomize**

If you prefer packaging:

* **Helm**: create a chart with values for image tag, env, secrets
* **Kustomize**: overlays for dev, staging, prod

**Dockerfile**

# Use OpenJDK 21 base image

FROM openjdk:21-jdk-slim

# Set working directory

WORKDIR /app

# Copy built jar file

COPY target/tuserservice-0.0.1-SNAPSHOT.jar app.jar

# Expose port

EXPOSE 8080

# Run application

ENTRYPOINT ["java", "-jar", "app.jar"]

**Build & Run with Docker**

# Build image

docker build -t tuserservice:latest .

# Run container

docker run -d -p 8080:8080 --name tuserservice tuserservice:latest

**2. Kubernetes Deployment**

**Deployment Manifest**

apiVersion: apps/v1

kind: Deployment

metadata:

name: tuserservice-deployment

labels:

app: tuserservice

spec:

replicas: 3

selector:

matchLabels:

app: tuserservice

template:

metadata:

labels:

app: tuserservice

spec:

containers:

- name: tuserservice

image: tuserservice:latest

imagePullPolicy: IfNotPresent

ports:

- containerPort: 8080

env:

- name: SPRING\_PROFILES\_ACTIVE

value: "prod"

- name: DB\_HOST

value: "postgres-service"

- name: DB\_PORT

value: "5432"

- name: DB\_USER

valueFrom:

secretKeyRef:

name: userservice-secrets

key: db-username

- name: DB\_PASSWORD

valueFrom:

secretKeyRef:

name: userservice-secrets

key: db-password

**Service Manifest**

apiVersion: v1

kind: Service

metadata:

name: tuserservice-service

spec:

selector:

app: tuserservice

ports:

- protocol: TCP

port: 80

targetPort: 8080

type: ClusterIP

**Ingress (Optional – If using API Gateway like Nginx or APISIX)**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: tuserservice-ingress

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- host: tuserservice.local

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: tuserservice-service

port:

number: 80

**3. Secrets & Config**

**Secret for DB Credentials**

apiVersion: v1

kind: Secret

metadata:

name: userservice-secrets

type: Opaque

data:

db-username: dXNlcm5hbWU= # base64("username")

db-password: cGFzc3dvcmQ= # base64("password")

**4. Deployment Steps**

1. **Build & Push Docker Image**
2. docker build -t <registry>/tuserservice:latest .
3. docker push <registry>/tuserservice:latest
4. **Apply Kubernetes Manifests**
5. kubectl apply -f k8s/deployment.yaml
6. kubectl apply -f k8s/service.yaml
7. kubectl apply -f k8s/ingress.yaml
8. **Verify Deployment**
9. kubectl get pods

kubectl get svc