

DevOps Candidate Practical Assessment

Objective

Demonstrate your ability to automate infrastructure setup, CI/CD, containerization, and basic Kubernetes orchestration, while producing clear documentation suitable for a remote team.

Project Overview

Deploy a small given **Spring Boot + PostgreSQL** application using **Docker** and **Kubernetes**.

Automate build, test, and deployment through a **CI/CD pipeline**.

Document your process clearly so another developer can reproduce it from scratch.

 **Note:** The project is time-boxed (see below). Focus on demonstrating a solid approach, working automation, and clear documentation rather than perfection or completeness.
Do as much as you can within the time, partial but well-explained work is valued more than a complete full setup.

Provided

We have provided

- Spring Boot application in coffeequeue folder
- createSchema.sql to initialise the database.

Tasks

1. Git & CI/CD

- Create a Git repository with a clear branching strategy (main, develop, feature branches).
- Implement an automated pipeline using **GitHub Actions** (or similar) that:
 1. Builds the app with Maven.
 2. Runs unit tests.
 3. Builds and pushes Docker images.

4. Deploys to a local Kubernetes cluster (e.g., Minikube or kind).
 - Provide the workflow file and a screenshot or log of a successful run.

2. Environment Setup (Simulated Ubuntu)

- Supply a single script (`setup.sh`) that installs Docker, `kubectl`, and Minikube on a clean Ubuntu host.
 - It should allow a developer to prepare a working local environment quickly.
(You do not need to provision an actual VM — simulate the installation locally.)
-

3. Containerization

- Dockerize both the Spring Boot app and the PostgreSQL database.
 - Include Dockerfiles and a `docker-compose.yml` to run the stack locally.
 - The app should read database credentials from environment variables.
-

4. Kubernetes Deployment

- Provide YAML manifests for:
 - **Deployment** and **Service** for the app.
 - **Deployment** and **Service** for PostgreSQL.
 - Configure **two replicas** of the app and a **Horizontal Pod Autoscaler** based on CPU usage.
 - Ensure **rolling updates** occur without downtime.
-

5. Database Initialization

- Include the provided `createSchema.sql`.
 - Ensure it runs automatically when the PostgreSQL container starts.
-

6. Monitoring & Health Checks

- Add **liveness** and **readiness** probes to your app deployment.

- Optional: describe how you'd extend this with Prometheus/Grafana or log aggregation.
-

7. Documentation

Provide a clear README.md that covers:

- Tools and versions used.
 - Step-by-step setup and deployment instructions.
 - Explanation of CI/CD flow.
 - Design decisions, trade-offs, and areas you'd improve if you had more time
-

Deliverables

1. Repository containing:
 - Source code, Dockerfiles, and docker-compose.yml
 - CI/CD workflow file(s)
 - Kubernetes manifests
 - setup.sh and SQL scripts
 - Documentation (README.md)
 2. Screenshots or logs showing a successful build, deployment, and scaled pods.
-

Time Box

Approx. 15–20 hours total (over one week).

You are not expected to deliver a production-grade system.

Focus on automation, correctness, and clear reasoning.

If the time frame is too small, **complete as much as you can and document what remains** — your approach will still be evaluated.

Optional Extensions (for senior applicants)

If time allows, you may include:

- Automated rollback in CI/CD on failed deploy.
- Basic resource monitoring (kubectl top pods, Prometheus, etc.).
- Use of Kubernetes secrets for database credentials.