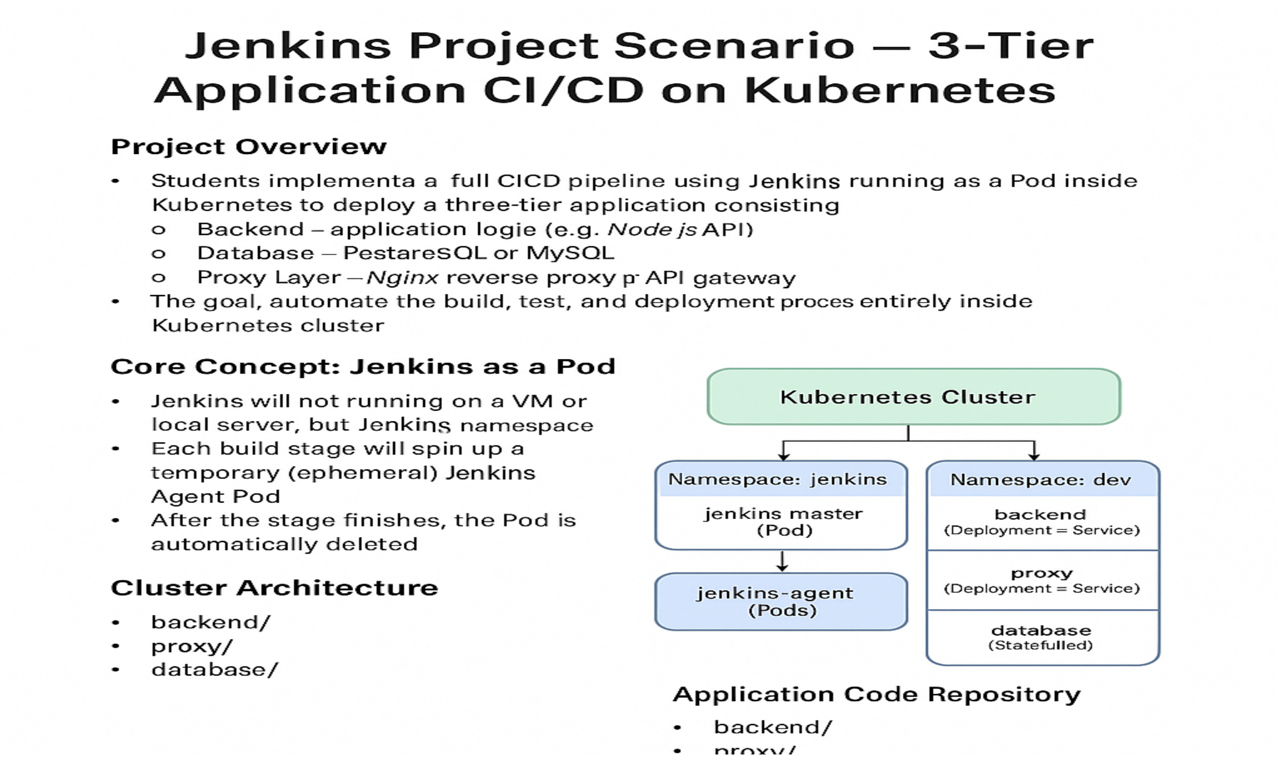
Jenkins Project Scenario – 3-Tier Application CI/CD on Kubernetes

Application Source Code Repository:
The full application source code for this project is available here:
https://github.com/abdelrahmanonline4/deploy-tier-application-backend-Database-proxy-

This repository contains three main components:
• **backend/** – Application logic and API code
• **proxy/** – Nginx configuration for routing
• **database/** – Database initialization and scripts

Core Concept: Jenkins as a Pod
Jenkins runs as a Pod inside the Kubernetes cluster in the 'jenkins' namespace. Each pipeline stage triggers a temporary Jenkins Agent Pod to perform tasks like building, testing, or deploying. Once the stage is complete, the agent Pod is deleted automatically.

Updated Architecture Diagram



Jenkins Project Scenario — 3-Tier Application CI/CD on Kubernetes

Project Overview

- Students implementa a full CICD pipeline using Jenkins running as a Pod inside Kubernetes to deploy a three-tier application consisting
  - Backend – application logie (e.g. *Node js* API)
  - Database – PestaresQL or MySQL
  - Proxy Layer – *Nginx* reverse proxy p API gateway
- The goal, automate the build, test, and deployment proces entirely inside Kubernetes cluster

Core Concept: Jenkins as a Pod

- Jenkins will not running on a VM or local server, but Jenkins namespace
- Each build stage will spin up a temporary (ephemeral) Jenkins Agent Pod
- After the stage finishes, the Pod is automatically deleted

Cluster Architecture

- backend/
- proxy/
- database/

Application Code Repository

- backend/
- proxy/

Pipeline Flow
1. **Source Stage:** Jenkins pulls the latest code from the GitHub repository.
2. **Build Stage:** Jenkins builds Docker images for backend, proxy, and database, tagging each with ${BUILD_NUMBER}.
3. **Push Stage:** The new images are pushed to a Docker registry (e.g., DockerHub).
4. **Deploy Stage:** Jenkins uses Helm or kubectl to deploy the updated images to the Kubernetes cluster under the 'dev' namespace.
5. **Smoke Test:** Jenkins performs a health check on the /health endpoint to confirm successful deployment.
6. **Notification:** The pipeline outputs success or failure results.

Educational Outcomes

By completing this project, students learn to:

- Run Jenkins as a Pod inside Kubernetes.
- Configure dynamic agent Pods.
- Build, push, and deploy multi-tier applications using Jenkins pipelines.
- Work with Docker, Helm, and Kubernetes namespaces.
- Integrate GitHub and container registries in CI/CD automation.

Real-World Relevance

This setup represents how modern DevOps teams operate:

- Jenkins operates natively on Kubernetes.
- Each job runs on isolated, disposable Pods for efficiency.
- The CI/CD pipeline is scalable, cloud-native, and production-ready.