Project - High Level Design

Deploy Education App To Kubernetes

Course Name: DevOps

Institution Name: Medicaps University – Datagami Skill Based Course

Student Name(s) & Enrolment Number(s):

Sr no	Student Name	Enrolment Number
1.	Uddhav Chourasiya	EN22CS3011042
2.	Tanishq Sharma	EN22CS3011021
3.	Vaishnavi Sarodey	EN22CS3011053
4.	Suman Dangi	EN22CS301998
5.	Sujal Mukati	EN22CS301995

Group Name:- 04D9

Project Number:- 04

Industry Mentor Name:- Mr. Vaibhav Sir

University Mentor Name:- Dr. Ritesh Joshi

Academic Year:2025-26

Table of Contents

1. Introduction :-

1.1 Scope of the Document:-

This document describes the High-Level Design (HLD) for the project “Deploy Education App to Kubernetes”.

The objective of this project is:

1. Containerize an Education (Quiz) Application
2. Deploy it on Kubernetes
3. Implement rolling updates with zero downtime
4. Host it on AWS EC2 infrastructure

The document covers:-

1. Architecture
2. Application Design
3. Kubernetes Deployment
4. Data Design
5. APIs
6. Security
7. Performance aspects

1.2 Intended Audience:-

This document is intended for:-

1. Project mentors
2. Team members
3. DevOps engineers

4. Review committee

It provides a clear understanding of system architecture and deployment strategy.

1.3 System Overview:-

- The system is a Quiz-based Education Application developed using:
- Python Flask (Backend)
- HTML/CSS (Frontend)
- Docker (Containerization)
- Kubernetes (Orchestration)
- AWS EC2 (Infrastructure)
- The application allows users to:
- Login
- Attempt quiz
- Submit answers
- View results

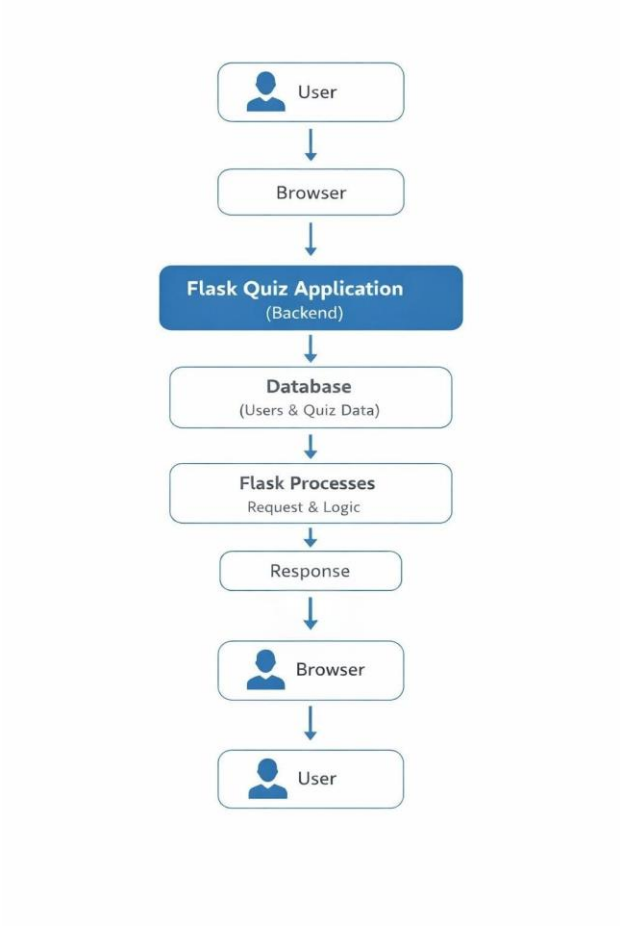
The deployment ensures:

- High availability
- Scalability
- Zero downtime updates

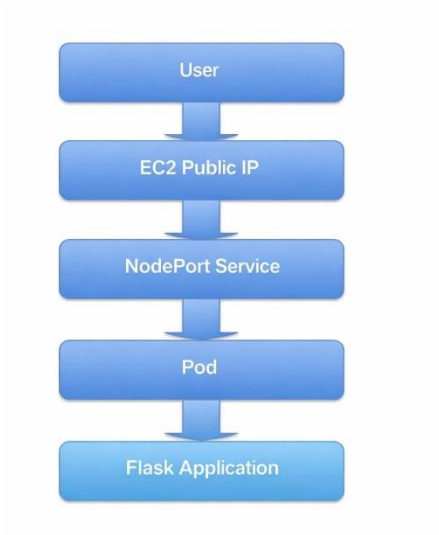
2. System Design :-

2.1 Application Design :-

The application follows a simple web architecture:



In Kubernetes environment:



Main components:-

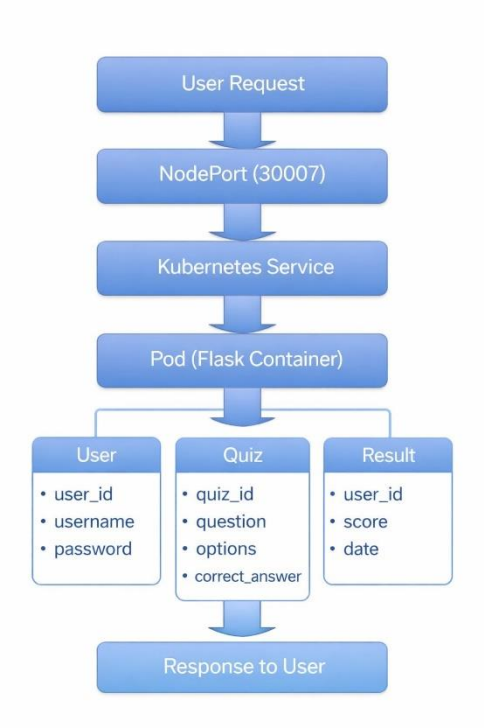
- Flask Backend
- HTML Templates

- Docker Image
- Kubernetes Deployment
- Kubernetes Service

2.2 Process Flow

- Login Flow
- User opens application
- Redirected to login page
- Credentials validated
- Dashboard loaded
- Quiz Flow
- User selects quiz
- Questions loaded
- User submits answers
- Score calculated
- Result displayed

2.3 Information Flow



2.4 Components Design

1. AWS EC2

Provides virtual server to host Kubernetes cluster.

2. Docker

Packages application and dependencies into a container.

3. Kubernetes (K3s)

Manages:

- Pod creation
- Scaling
- Rolling updates
- Service exposure

4. Deployment

Maintains desired number of replicas.

5. Service (NodePort)

Exposes application externally.

2.5 Key Design Considerations

- High Availability using multiple replicas
- Rolling Update strategy
- Free-tier compatible infrastructure
- Port management
- Secure inbound rules
- Lightweight Kubernetes (K3s)

2.6 API Catalogue

1. Login API

POST /login

Authenticates user.

2. Logout API

GET /logout

Logs user out.

3. Home API

GET /

Redirects to login.

4. Quiz API

GET /quiz

Displays questions.

5. Submit API

POST /submit

Submits answers.

6. Result API

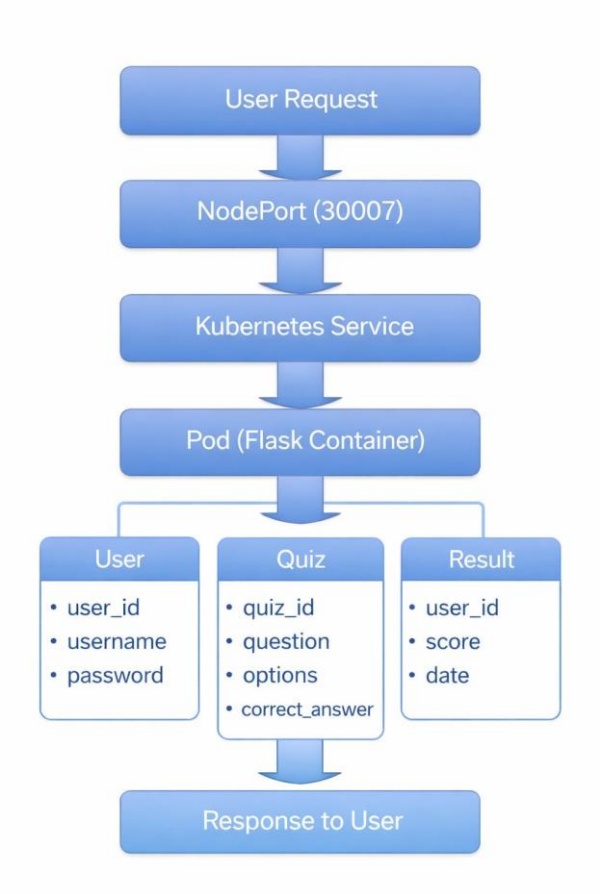
GET /result

Displays final score.

APIs are exposed internally on port 5000 and externally via NodePort 30007.

3. Data Design :-

3.1 Data Model



3.2 Data Access Mechanism :-

- Flask handles database interaction.
- Uses SQL queries.
- Application logic processes data.

3.3 Data Retention Policies :-

- User login data stored securely.
- Quiz attempts stored for evaluation.
- Data stored locally in application database.

3.4 Data Migration

Future scalability:

- Move database to AWS RDS.
- Use persistent volumes in Kubernetes.

4. Interfaces:-

- User Interface
- Web browser interface using HTML/CSS.
- System Interface
- Docker CLI
- kubectl CLI
- AWS Console

5. State and Session Management :-

- Flask session mechanism used.
- Session stored in memory.
- After logout, session destroyed.
- Kubernetes replicas handle stateless design.

6. Caching :-

- Currently minimal caching.
- Future enhancements:
- Redis caching
- Kubernetes horizontal scaling

- Load balancing

7. Non-Functional Requirements:-

7.1 Security Aspects:-

- AWS Security Groups
- Restricted ports (22, 30007)
- Container isolation
- Kubernetes namespace isolation
- No hardcoded credentials
- Future improvements:
- HTTPS

- JWT Authentication
- Secrets management

7.2 Performance Aspects:-

- Multiple replicas for load handling
- Rolling update without downtime
- Lightweight Kubernetes (K3s)
- Efficient container image

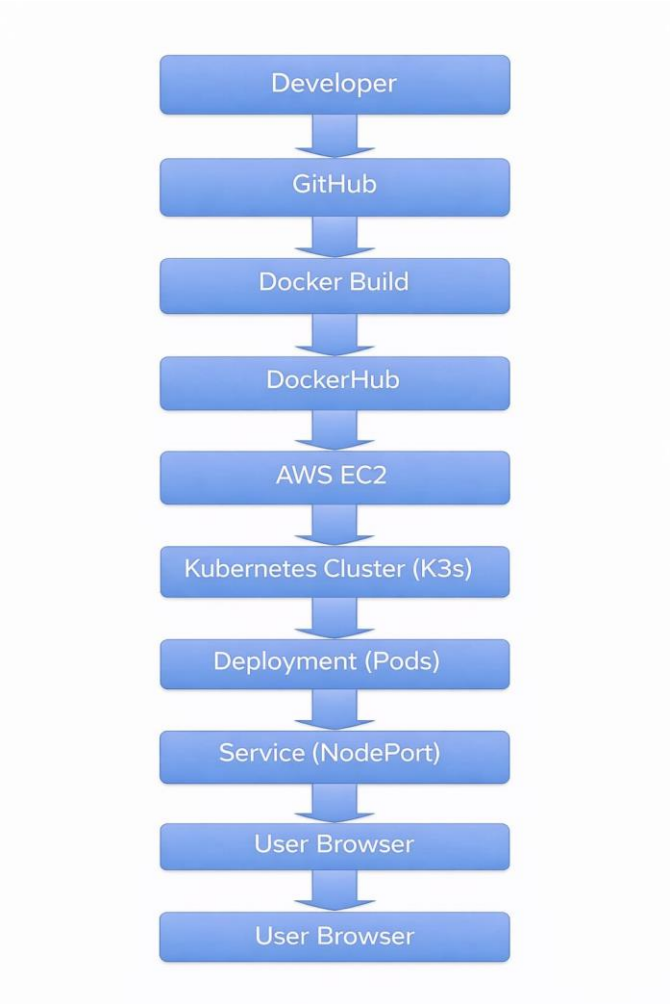
8. Rolling Update Strategy :-

- Rolling Update ensures:
- New version deployed gradually
- Old pods terminated one by one
- At least one pod always running
- Zero downtime

Command used:

kubectl set image deployment/quiz-app quiz-container=image:v2

9. Architecture Diagram (Explain While Drawing):-



10. Tools Used :-

- Git
- Docker
- DockerHub
- AWS EC2
- Kubernetes (K3s)

- kubectl

11. References :-

- Kubernetes Official Documentation
- Docker Documentation
- AWS EC2 Documentation
- Flask Documentation