Project Title: Microservices-Based Learning Platform for Course Content Management and Learner Progress Tracking System

Simon Gawar

Project Phase 3: Deploying the Application

1. Overview of Microservices for deployment

The CMS application now consists of the following microservices and infrastructure components:

- user-service → Port 8082
- course-service → Port 8083
- analytics-service → Port 8086
- notification-service → Port 8085
- userassessment-service → Port 8084
- auth-service → Port 8081
- cloak-resource-server → Port 8087
- eureka-server → Port 8761
- config-server → Port 8071
- api-gateway → Port 9090
- common_security (this is shared module for Keycloak integration)

2. Deployment Steps

Step 1: Package Microservices using Maven:

- mvn clean package

Then each microservice uses a Dockerfile to build its image from the JAR file as named below:

- user_service-0.0.1-SNAPSHOT.jar
- course-service-0.0.1-SNAPSHOT.jar
- analytics-service-0.0.1-SNAPSHOT.jar
- notification-service-0.0.1-SNAPSHOT.jar
- userassessment-service-0.0.1-SNAPSHOT.jar
- eureka-server-001-SNAPSHOT.jar
- config-server-0.0.1-SNAPSHOT.jar
- api-gateway-0.0.1-SNAPSHOT.jar
- common_security (shared module)

Step 2: Dockerization for each microservice:

- Each microservice includes a Dockerfile that copies the JAR file and sets environment variables for DB and Keycloak.

Step 3: Docker Compose orchestration:

- Includes Eureka Server, Config Server, API Gateway, all microservices, and Keycloak + mysql.

Step 4: Keycloak Setup:

- Realm: cms-realm, Client: cms-client, Roles: ADMIN, INSTRUCTOR, LEARNER, Users mapped to roles.

Step 5: Heroku deployment steps (Still not yet finalized):

- Install Heroku CLI and login: heroku login
- Create app: heroku create <app-name>
- Add ClearDB MySQL: heroku addons:create cleardb:ignite
- Configure environment variables for DB connection
- Push Docker image: heroku container:push web -a <app-name>
- Release app: heroku container:release web -a <app-name>

2. Authentication Flow and Sequence Diagram

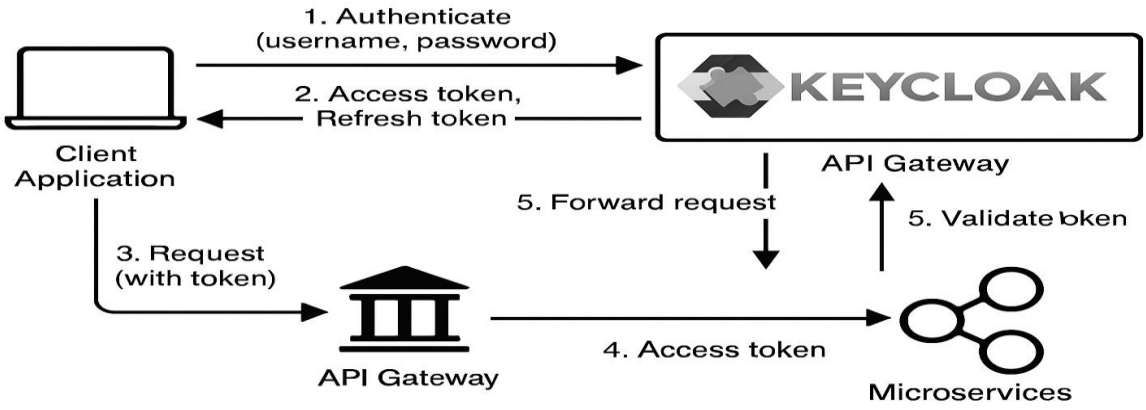


Figure 1: Authentication Flow Diagram

This is authentication flow diagram showing the process with Keycloak integration:

- **Step 1:** Client authenticates with Keycloak using username/password.
- **Step 2:** Keycloak issues access_token and refresh_token.
- **Step 3:** Client sends API request with Authorization: Bearer <access_token> to API Gateway.
- **Step 4:** API Gateway forwards the request to microservices.
- **Step 5:** Token validation occurs via Keycloak JWKS.

Login: POST /realms/cms-realm/protocol/openid-connect/token → returns access_token and refresh_token.
Refresh: POST /token with refresh_token → returns new tokens.
Logout: POST /logout with refresh_token → invalidates session.

API Gateway & Microservices: All requests include Authorization: Bearer <access

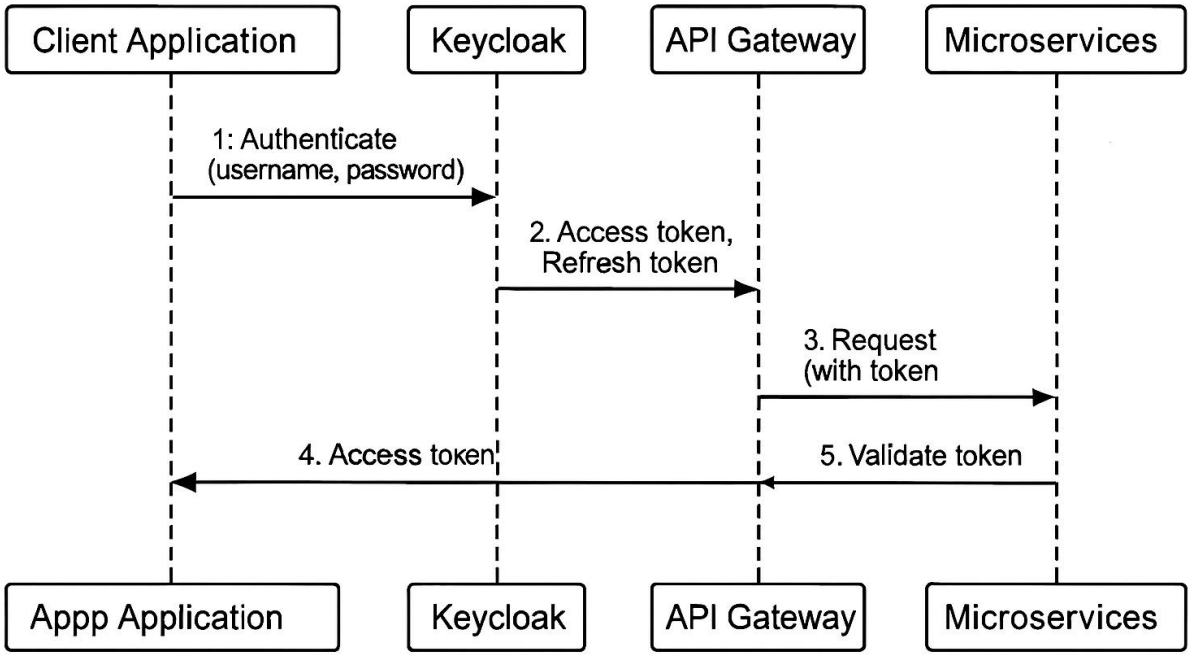


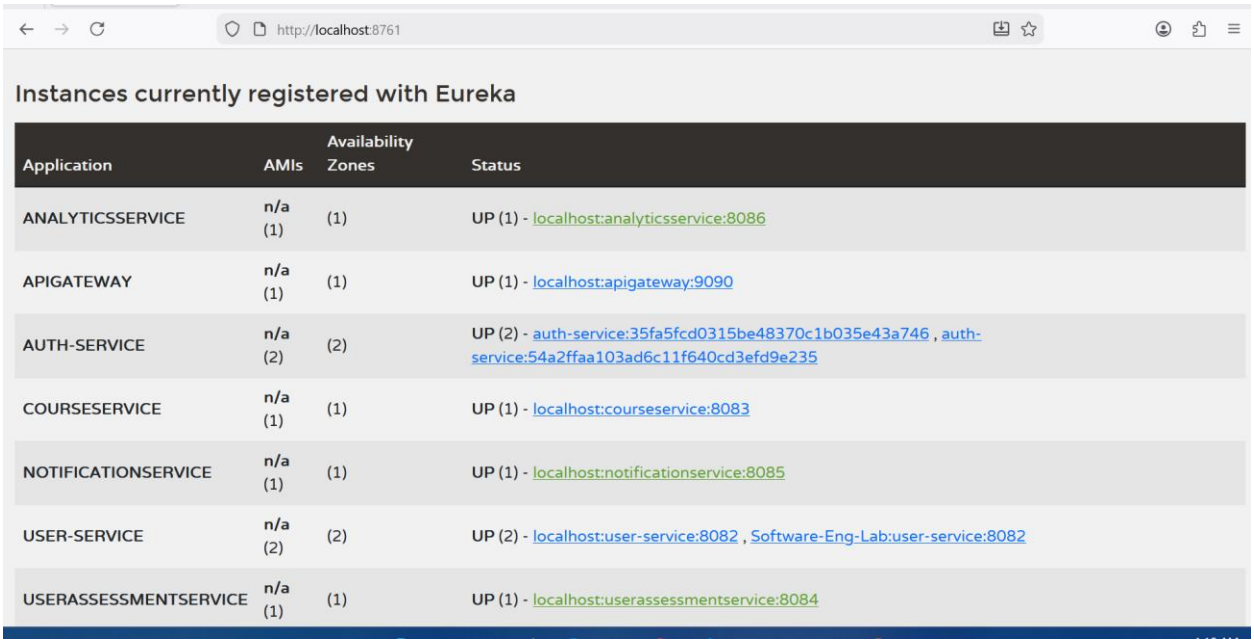
Fig 2: Authentication Sequence Diagram

This is authentication sequence diagram Flow Explanation:

- **Actors**: Here is the **authentication sequence diagram** for Keycloak integration:
- **Actors:**
- Client Application
- Keycloak
- API Gateway
- Microservices
- **Steps:**
- Client authenticates with Keycloak using username/password.
- Keycloak issues access_token and refresh_token.
- Client sends request with token to API Gateway.
- API Gateway forwards request to microservices.
- Token validation occurs via Keycloak JWKS.

3. Eureka server

Eureka is a Service Registry that allows services to register themselves and discover other services dynamically without hardcoding IP addresses as shown below.



4. Environment Variables and Database Configuration

Example .env:

- KEYCLOAK_CLIENT_SECRET=your-secret
- SPRING_PROFILES_ACTIVE=dev
- EUREKA_URL=http://eureka-server:8761/eureka/
- DB_USER=root
- DB_PASSWORD=root@2025db

5. Verification

1. Initialize Git if not done:

git init

git add .

git commit -m "Phase 3 Deployment"

2. Push to Heroku: git push heroku main Or deploy JAR directly: heroku deploy:jar target/<your-app>.jar --app your-app-name

- Access Eureka dashboard: http://localhost:8761
- Test APIs via Postman using Keycloak tokens.
- Validate roles for endpoints: /api/users/admin/** → ADMIN only.
- Open the app:
- heroku open

7. Submission Checklist

- GitHub Repo: Source code + README
- Postman Collection: Updated with Keycloak endpoints
- README: Include Keycloak setup, Docker instructions, and testing guide
- Video Demo: Show Docker Compose deployment + API calls