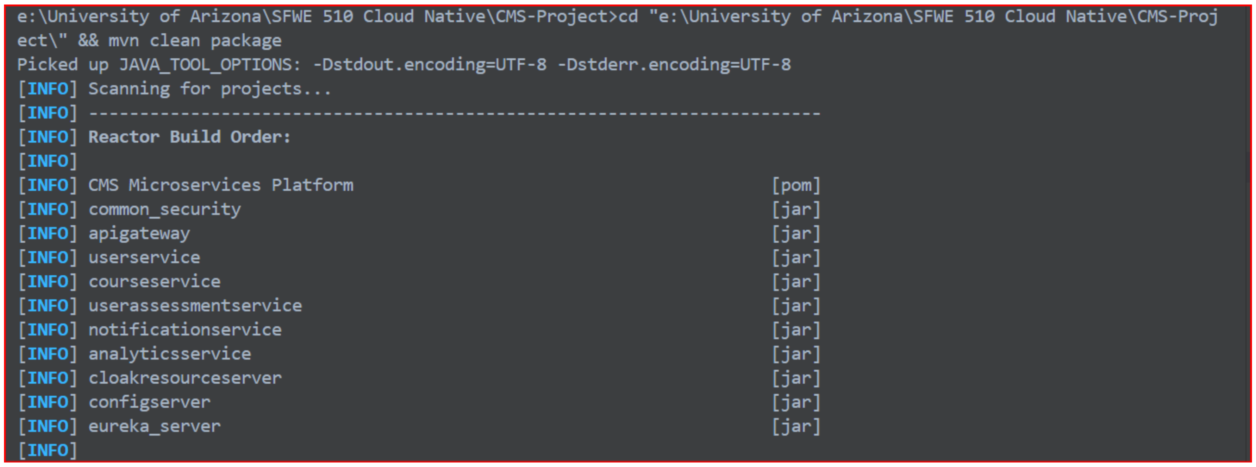
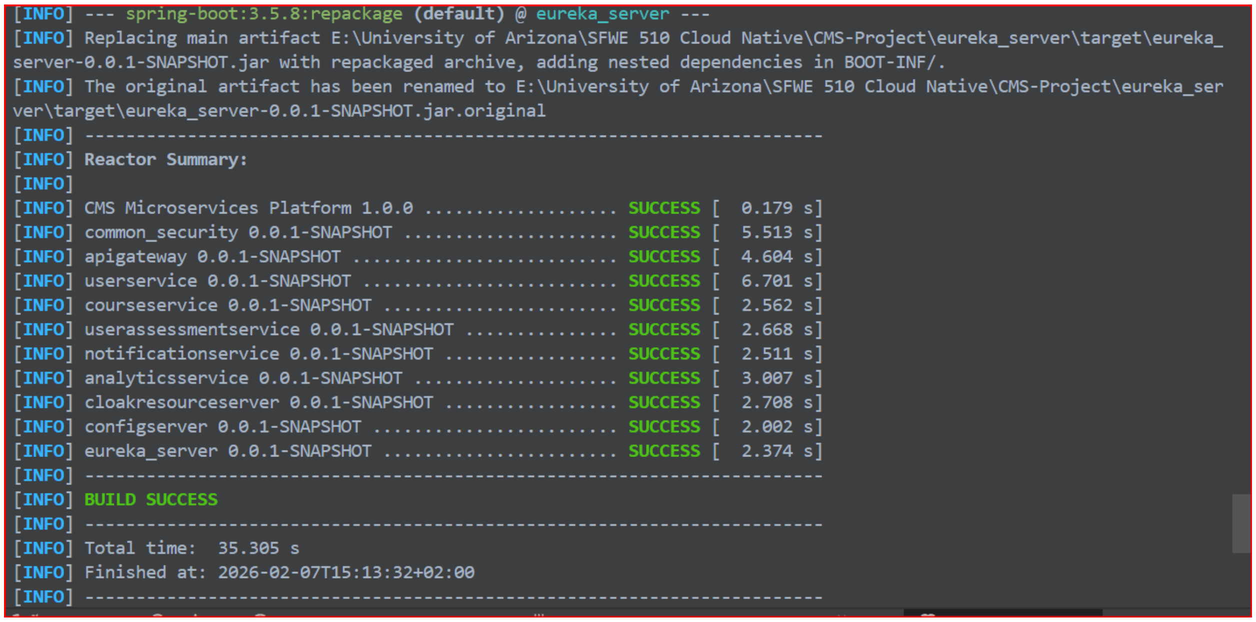
Simon Gawar

CMS-Project Jars Build, Docker, and Kubernetes Deployment Report

1. Build Process

- **Multi-module Maven project**: Each microservice (User, Course, Notification, Analytics, UserAssessment, Cloak Resource Server, API Gateway, Config Server, Eureka Server) built as a Spring Boot application.
- **Optimizations**:
    - Tests skipped (-DskipTests) to avoid context-load failures.
    - Maven Surefire plugin configured with <skipTests>true</skipTests>.
    - Cache mounts used in Dockerfiles for faster dependency resolution.
- **Outcome**: All services produced fat JARs (target/*.jar) ready for containerization. Build logs confirmed **BUILD SUCCESS**.
- 

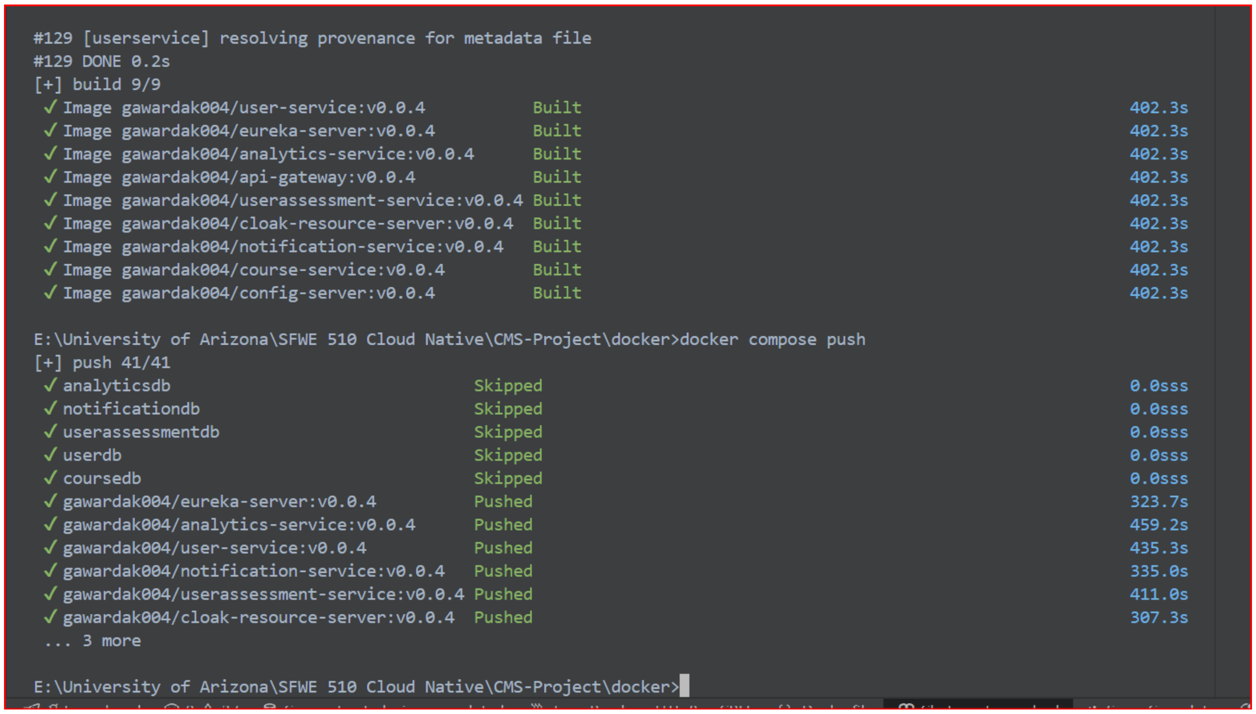
```
e:\University of Arizona\SFWE 510 Cloud Native\CMS-Project>cd "e:\University of Arizona\SFWE 510 Cloud Native\CMS-Proj
ect\" && mvn clean package
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Build Order:
[INFO]
[INFO] CMS Microservices Platform                                         [pom]
[INFO] common_security                                                    [jar]
[INFO] apigateway                                                         [jar]
[INFO] userservice                                                        [jar]
[INFO] courseservice                                                      [jar]
[INFO] userassessmentservice                                              [jar]
[INFO] notificationservice                                                [jar]
[INFO] analyticsservice                                                   [jar]
[INFO] cloakresourceserver                                                [jar]
[INFO] configserver                                                       [jar]
[INFO] eureka_server                                                      [jar]
[INFO]
```



```
[INFO] --- spring-boot:3.5.8:repackage (default) @ eureka_server ---
[INFO] Replacing main artifact E:\University of Arizona\SFWE 510 Cloud Native\CMS-Project\eureka_server\target\eureka_
server-0.0.1-SNAPSHOT.jar with repackaged archive, adding nested dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to E:\University of Arizona\SFWE 510 Cloud Native\CMS-Project\eureka_ser
ver\target\eureka_server-0.0.1-SNAPSHOT.jar.original
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] CMS Microservices Platform 1.0.0 ................... SUCCESS [  0.179 s]
[INFO] common_security 0.0.1-SNAPSHOT ..................... SUCCESS [  5.513 s]
[INFO] apigateway 0.0.1-SNAPSHOT .......................... SUCCESS [  4.604 s]
[INFO] userservice 0.0.1-SNAPSHOT ......................... SUCCESS [  6.701 s]
[INFO] courseservice 0.0.1-SNAPSHOT ....................... SUCCESS [  2.562 s]
[INFO] userassessmentservice 0.0.1-SNAPSHOT ............... SUCCESS [  2.668 s]
[INFO] notificationservice 0.0.1-SNAPSHOT ................. SUCCESS [  2.511 s]
[INFO] analyticsservice 0.0.1-SNAPSHOT .................... SUCCESS [  3.007 s]
[INFO] cloakresourceserver 0.0.1-SNAPSHOT ................. SUCCESS [  2.708 s]
[INFO] configserver 0.0.1-SNAPSHOT ........................ SUCCESS [  2.002 s]
[INFO] eureka_server 0.0.1-SNAPSHOT ....................... SUCCESS [  2.374 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  35.305 s
[INFO] Finished at: 2026-02-07T15:13:32+02:00
[INFO] ------------------------------------------------------------------------
```

2. Dockerization

- **Dockerfiles standardized**: Two-stage builds (Maven build → slim JRE runtime).
- **Security**: Non-root spring user created.
- **JVM tuning**: JAVA_TOOL_OPTIONS="-XX:MaxRAMPercentage=75.0 -Duser.timezone=UTC".
- **Compose orchestration**:
  - Defined all microservices with dependencies on configserver and eurekaserver.
  - Added dedicated MySQL containers (userdb, coursedb, userassessmentdb, notificationdb, analyticsdb) with persistent volumes.
  - Single cms bridge network for service discovery.
- **Outcome**: docker compose up -d --build successfully launched **41 containers** (services + DBs + infra). Images tagged and pushed to Docker Hub as seen below:



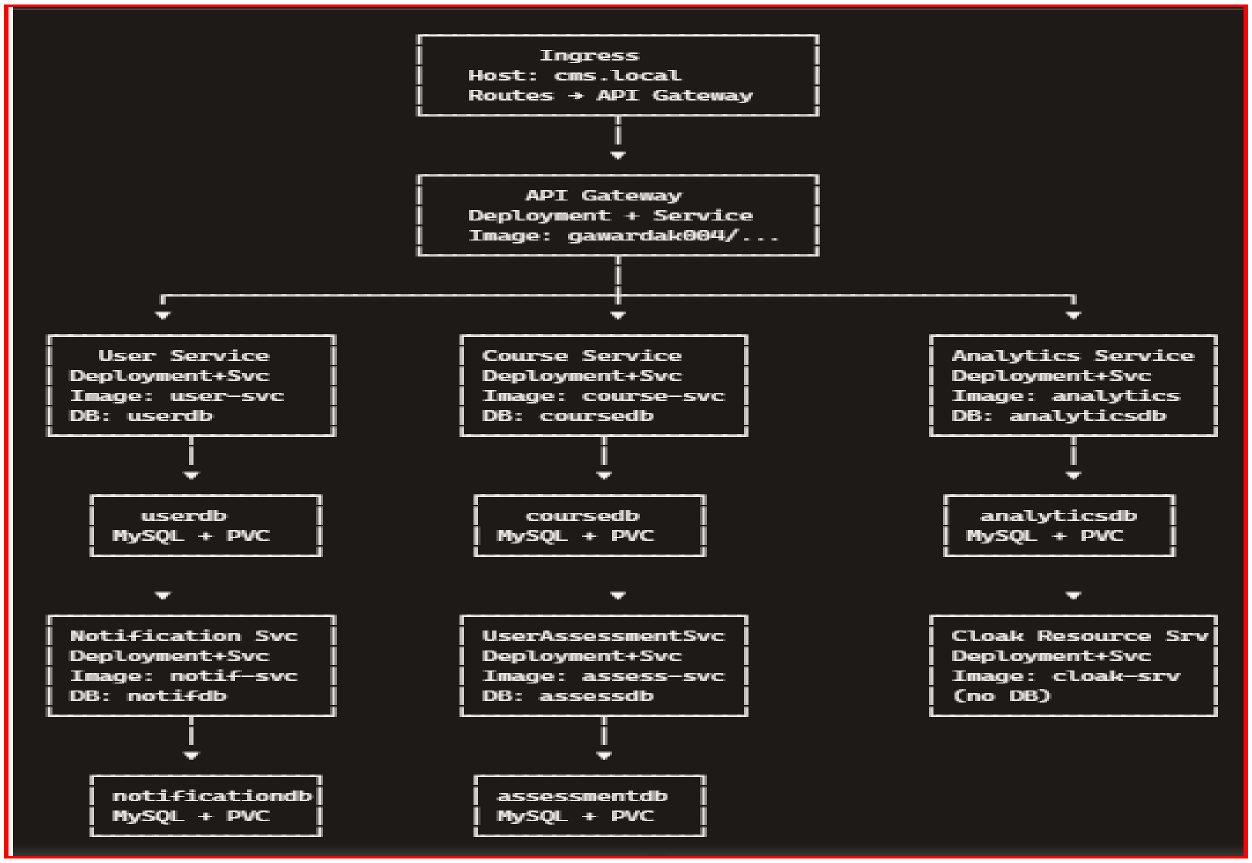
```
#129 [userservice] resolving provenance for metadata file
#129 DONE 0.2s
[+] build 9/9
 ✓ Image gawardak004/user-service:v0.0.4          Built                                                          402.3s
 ✓ Image gawardak004/eureka-server:v0.0.4         Built                                                          402.3s
 ✓ Image gawardak004/analytics-service:v0.0.4     Built                                                          402.3s
 ✓ Image gawardak004/api-gateway:v0.0.4           Built                                                          402.3s
 ✓ Image gawardak004/userassessment-service:v0.0.4 Built                                                         402.3s
 ✓ Image gawardak004/cloak-resource-server:v0.0.4  Built                                                         402.3s
 ✓ Image gawardak004/notification-service:v0.0.4   Built                                                         402.3s
 ✓ Image gawardak004/course-service:v0.0.4         Built                                                         402.3s
 ✓ Image gawardak004/config-server:v0.0.4          Built                                                         402.3s

E:\University of Arizona\SFWE 510 Cloud Native\CMS-Project\docker>docker compose push
[+] push 41/41
 ✓ analyticsdb                                  Skipped                                                          0.0sss
 ✓ notificationdb                               Skipped                                                          0.0sss
 ✓ userassessmentdb                             Skipped                                                          0.0sss
 ✓ userdb                                       Skipped                                                          0.0sss
 ✓ coursedb                                     Skipped                                                          0.0sss
 ✓ gawardak004/eureka-server:v0.0.4             Pushed                                                           323.7s
 ✓ gawardak004/analytics-service:v0.0.4         Pushed                                                           459.2s
 ✓ gawardak004/user-service:v0.0.4              Pushed                                                           435.3s
 ✓ gawardak004/notification-service:v0.0.4      Pushed                                                           335.0s
 ✓ gawardak004/userassessment-service:v0.0.4 Pushed                                                              411.0s
 ✓ gawardak004/cloak-resource-server:v0.0.4  Pushed                                                              307.3s
 ... 3 more

E:\University of Arizona\SFWE 510 Cloud Native\CMS-Project\docker>
```

3. Kubernetes Deployment

This document provides **summary report on all services, servers, and supporting resources** currently defined in your Kubernetes deployment for the CMS project:

- **Deployment Architecture:** Show the communication flow of the services in Kubernetes deployment
- **Namespace isolation**: cms namespace created.
- **Database deployments**: Each DB defined as a Deployment + Service with PVCs.
- **Microservice deployments**: Each Spring Boot service defined as a Deployment + Service.
- **Ingress setup**: API Gateway exposed externally via Ingress (cms.local host).
- **Outcome**: Full stack deployed in Kubernetes. Eureka Server and Config Server orchestrate service registration and configuration. API Gateway provides a single-entry point for external clients.

❖ **Deployment Architecture**



Figgure 1: Kubernetes Deployment

This diagram shows the logical flow: external requests enter through **Ingress → API Gateway**, then fan out to microservices, which rely on **MySQL databases** and are coordinated by **Config Server + Eureka Server**.

❖ Namespace

- **Namespace**: cms is where all resources (services, deployments, secrets, configmaps, ingress, HPA) are scoped under this namespace.

❖ Configuration

- **ConfigMap**: cms-common
  Holds shared configuration values (Spring profiles, Eureka URLs, management endpoints, etc.).
- **Secret**: cms-secrets
  Stores sensitive data such as database URLs, usernames, and passwords.

- **Service → DB Secret Mapping**

Service	Database Name	JDBC URL Secret Key	Username Secret Key	Password Secret Key
User Service	userservicedb	DB_URL_USER	DB_USERNAME_USER	DB_PASSWORD_USER
Course Service	courseservicedb	DB_URL_COURSE	DB_USERNAME_COURSE	DB_PASSWORD_COURSE
User Assessment	userassessmentservicedb	DB_URL_ASSESSMENT	DB_USERNAME_ASSESSMENT	DB_PASSWORD_ASSESSMENT
Notification Service	notificationservicedb	DB_URL_NOTIFICATION	DB_USERNAME_NOTIFICATION	DB_PASSWORD_NOTIFICATION
Analytics Service	analyticsservicedb	DB_URL_ANALYTICS	DB_USERNAME_ANALYTICS	DB_PASSWORD_ANALYTICS

❖ **API Gateway**

- **Deployment**: apigateway (2 replicas)
- **Service**: ClusterIP on port 9091
- **Ingress**: cms-ingress routes external traffic from cms.local → API Gateway
- **Image**: gawardak004/api-gateway:latest
- **Probes**: Readiness and liveness via /actuator/health endpoints.

❖ **Analytics Service**

- **Deployment**: analyticsservice (1 replica)
- **Service**: ClusterIP on port 8086

- **Horizontal Pod Autoscaler**: analyticsservice-hpa (min 2, max 10 replicas, CPU target 70%)
- **Image**: gawardak004/analytics-service:latest
- **Probes**: Readiness and liveness via /actuator/health.

❖ **Cloak Resource Server**

- **Deployment**: cloakresourceserver (1 replica)
- **Service**: ClusterIP on port 8080
- **Image**: gawardak004/cloak-resource-server:latest.

❖ **Config Server**

- **Deployment**: configserver (1 replica)
- **Service**: ClusterIP on port 8071
- **Image**: gawardak004/config-server:latest.

❖ **Course Service**

- **Deployment**: courseservice (1 replica)
- **Service**: ClusterIP on port 8083
- **Horizontal Pod Autoscaler**: courseservice (min/max scaling defined)
- **Image**: gawardak004/course-service:latest.

❖ **Eureka Server**

- **Deployment**: eurekaserver (1 replica)
- **Service**: ClusterIP on port 8761
- **Image**: gawardak004/eureka-server:latest.

❖ **Notification Service**

- **Deployment**: notificationservice (1 replica)
- **Service**: ClusterIP on port 8085
- **Horizontal Pod Autoscaler**: notificationservice-hpa
- **Image**: gawardak004/notification-service:latest.

❖ **User Service**

- **Deployment**: userservice (replica count defined in manifest)
- **Service**: ClusterIP on port 8082 (assumed from your setup)
- **Image**: gawardak004/user-service:latest.

❖ **User Assessment Service**

- **Deployment**: userassessmentservice (replica count defined in manifest)
- **Service**: ClusterIP on port 8084 (assumed from your setup)
- **Image**: gawardak004/userassessment-service:latest.

❖ **Databases (MySQL)**

- **MySQL StatefulSets/Deployments**:
    - userdb → port 3307
    - coursedb → port 3308
    - userassessmentdb → port 3309
    - notificationdb → port 3310
    - analyticsdb → port 3311
- **PersistentVolumeClaims**: Each DB bound to storage (mysql-pvc).
- **Environment Variables**: Must include MYSQL_ROOT_PASSWORD and MYSQL_DATABASE for initialization.

❖ **Network Policy**

- **NetworkPolicy**: Defined to restrict traffic between services, ensuring only allowed pods can communicate (e.g., API Gateway → downstream services, services → databases).
- Helps enforce **zero-trust networking** inside the cluster.

❖ **Outcomes**

- Full stack deployed in Kubernetes where all **Deployments**, **Services**, **Ingress**, **HPAs**, **ConfigMaps**, **Secrets**, and **NetworkPolicies** have been applied successfully under the cms namespace.
- Each microservice points to its Docker Hub image (gawardak004/...:v0.0.4).
- Databases are running with persistent storage and require proper secrets for credentials.
- Autoscaling is enabled for **Analytics Service**, **Course Service**, and **Notification Service**.
- Eureka Server and Config Server orchestrate service registration and configuration. API Gateway provides a single-entry point for external clients
- Config Server and Eureka Server are deployed, providing configuration and service discovery for the stack.
- API Gateway ingress (cms.local) is set up to route external traffic into the cluster.

The last stage was to execute or run:

```
kubectl get all -n cms
```

```
kubectl get hpa -n cms
```

```
kubectl get networkpolicy -n cms
```

This was to confirm pods are **Running**, autoscalers are active, and network policies are enforced.

How to use these

- Public endpoints (like /public/register and /public/ping) can be accessed without a JWT.

- Secured endpoints (/me, /secured/test, /admin/dashboard, /instructor/dashboard, /learner/dashboard) require a valid Keycloak access token in the Authorization: Bearer <token> header.

- All traffic flows through the API Gateway at cms.local, so you don't need to expose each microservice individually.

- **Access Eureka dashboard:** Open http://localhost:8761 (localhost in Bing) in your browser. We now should see the Eureka registry UI.
- **Access API Gateway:** Open http://localhost:9091 (localhost in Bing). Routes should be forward to your microservices.
- **Check Config Server:** Open http://localhost:8071. It should serve configuration properties.
- **Check Keycloak** (cloak-resource-server): We had mapped port 8080, let us now open http://localhost:8080 (localhost in Bing) to confirm the realm is up.

Methods of accessing microservices

There are two ways for accessing the dashboard

1. Microservices External Access via API Gateway

That is using CMS Access links **(HTTP only)** form external**.** Once I have updated the hosts file (172.24.0.2 cms.local), you can reach services directly:

As our gateway runs on http://localhost:9091 and routes are defined with StripPrefix=1, here's how external clients should call each service:

The following are full external access map updated for cms.local, including both the public CRUD endpoints and the secured role-based endpoints from your LoginController:

Service (internal)	**Gateway Route**	**External URL (via API Gateway at cms.local)**
User Service	**/userservice/****	**http://cms.local/userservice/api/users**

Course Service	/courseservice/**	http://cms.local/courseservice/api/courses
Notification Service	/notificationservice/**	http://cms.local/notificationservice/api/notifications
Analytics Service	/analyticsservice/**	http://cms.local/analyticsservice/api/analytics
User Assessment	/userassessmentservice/**	http://cms.local/userassessmentservice/api/assessments
Auth – Current User	/userservice/**	http://cms.local/userservice/api/auth/me
Auth – Public Ping	/userservice/**	http://cms.local/userservice/api/auth/public/ping
Auth – Secured Test	/userservice/**	http://cms.local/userservice/api/auth/secured/test
Auth – Admin Dashboard	/userservice/**	http://cms.local/userservice/api/auth/admin/dashboard
Auth – Instructor Dashboard	/userservice/**	http://cms.local/userservice/api/auth/instructor/dashboard
Auth – Learner Dashboard	/userservice/**	http://cms.local/userservice/api/auth/learner/dashboard

User Registration (public)	**/userservice/****	**http://cms.local/userservice/api/users/public/register**

2. PORT FORWARDING To TEST SERVICES WITH curl:

curl -v http://cms.local/userservice/actuator/health

curl -v http://cms.local/courseservice/actuator/health

curl -v http://cms.local/notificationservice/actuator/health

curl -v http://cms.local/analyticsservice/actuator/health

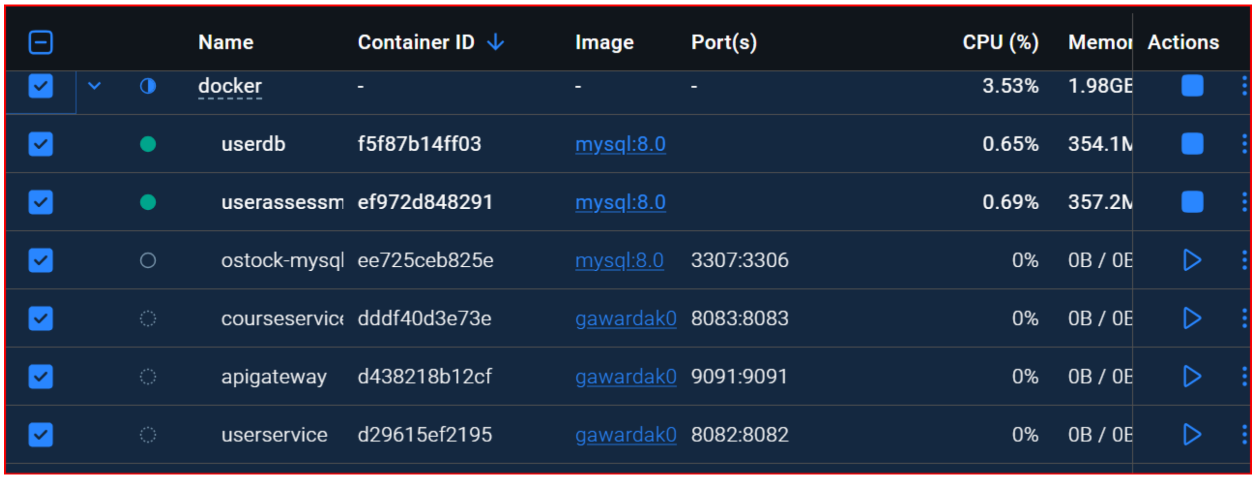
curl -v http://cms.local/userassessmentservice/actuator/health

curl -v http://cms.local/cloakresourceserver/actuator/health

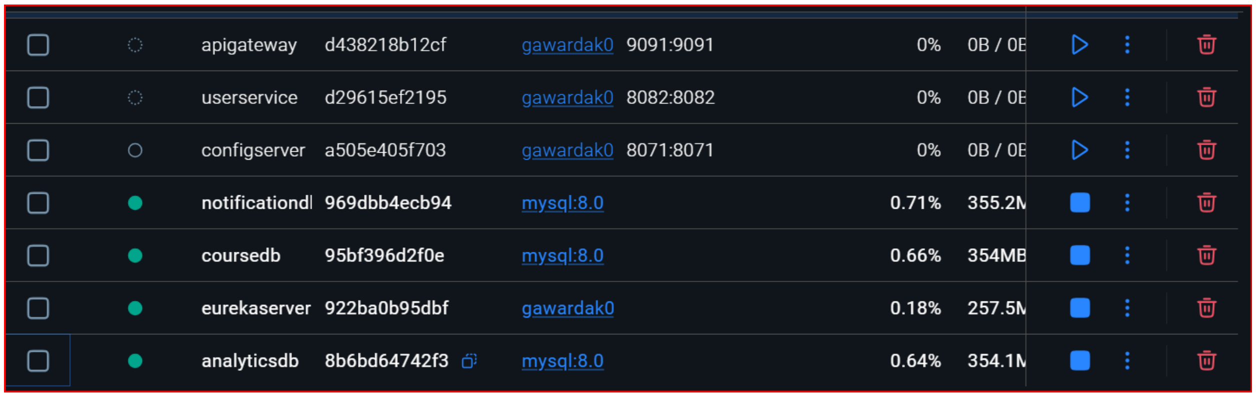
curl -v http://cms.local/eureka.

These are snapshots of the containers, images, and Kubernetes.

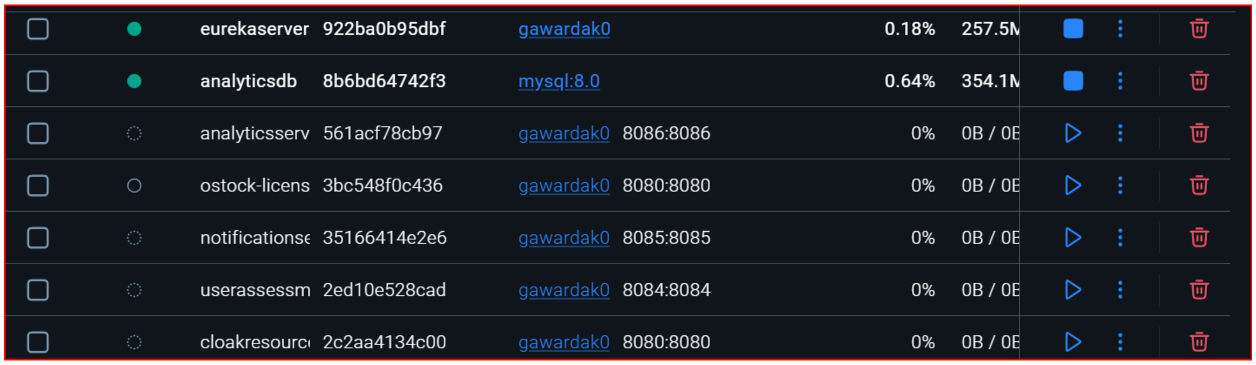
1. Containers



	Name	Container ID	Image	Port(s)	CPU (%)	Memor
◑	docker	-	-	-	3.53%	1.98GE
●	userdb	f5f87b14ff03	mysql:8.0		0.65%	354.1M
●	userassessm	ef972d848291	mysql:8.0		0.69%	357.2M
○	ostock-mysql	ee725ceb825e	mysql:8.0	3307:3306	0%	0B / 0E
◌	courseservic	dddf40d3e73e	gawardak0	8083:8083	0%	0B / 0E
◌	apigateway	d438218b12cf	gawardak0	9091:9091	0%	0B / 0E
◌	userservice	d29615ef2195	gawardak0	8082:8082	0%	0B / 0E



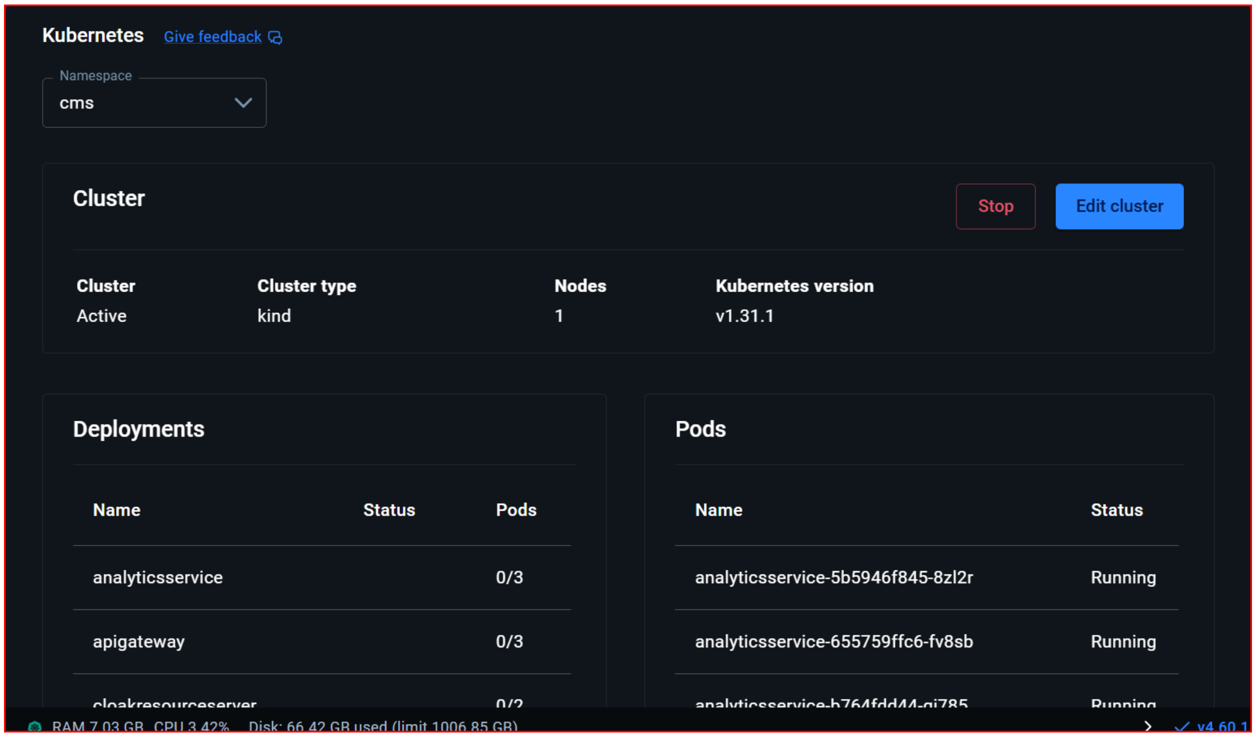
◌	apigateway	d438218b12cf	gawardak0	9091:9091	0%	0B / 0E
◌	userservice	d29615ef2195	gawardak0	8082:8082	0%	0B / 0E
○	configserver	a505e405f703	gawardak0	8071:8071	0%	0B / 0E
●	notificationdl	969dbb4ecb94	mysql:8.0		0.71%	355.2M
●	coursedb	95bf396d2f0e	mysql:8.0		0.66%	354MB
●	eurekaserver	922ba0b95dbf	gawardak0		0.18%	257.5M
●	analyticsdb	8b6bd64742f3	mysql:8.0		0.64%	354.1M



●	eurekaserver	922ba0b95dbf	gawardak0		0.18%	257.5M
●	analyticsdb	8b6bd64742f3	mysql:8.0		0.64%	354.1M
◌	analyticsserv	561acf78cb97	gawardak0	8086:8086	0%	0B / 0E
○	ostock-licens	3bc548f0c436	gawardak0	8080:8080	0%	0B / 0E
◌	notificationse	35166414e2e6	gawardak0	8085:8085	0%	0B / 0E
◌	userassessm	2ed10e528cad	gawardak0	8084:8084	0%	0B / 0E
◌	cloakresourc	2c2aa4134c00	gawardak0	8080:8080	0%	0B / 0E

2.kubernetes

This is the snapshoot of my cms microservices deployed in kurbernetes.



Kubernetes Give feedback

Namespace: cms

Cluster

Stop | Edit cluster

Cluster	Cluster type	Nodes	Kubernetes version
Active	kind	1	v1.31.1

Deployments

Name	Status	Pods
analyticsservice		0/3
apigateway		0/3
cloakresourceserver		0/2

Pods

Name	Status
analyticsservice-5b5946f845-8zl2r	Running
analyticsservice-655759ffc6-fv8sb	Running
analyticsservice-b764fdd44-gj785	Running

RAM 7.03 GB CPU 3.42% Disk: 66.42 GB used (limit 1006.85 GB) v4.60.1