Ferdowsi University
of Mashhad
1949

Cloud Computing Course

Project – Dockerize, Kubernetes, Micro Services, Services and Cloud

Ferdowsi University of Mashhad (FUM)

Course Professor:

Dr. Saeid Abrishami

Teaching Assistants:

Mobin Tasnimi

Mohsen Gholami

Saeid Rahmani

Cloud Computing – Winter 2025 – CE – FUM

Introduction

This cloud computing project guides students through the essential stages of modern cloud deployment, from containerization to scalable cloud infrastructure. It includes Dockerization, Kubernetes deployment, microservices integration, autoscaling, high availability, and cloud deployment on Arvan Cloud with disaster recovery. The project is worth 10 points, with an additional 1 extra point for further optimizations or enhancements.

Project consists of five phases:

1. **Dockerization (2 point)** – Containerizing the project and managing services using Docker Compose.
2. **Kubernetes (2 points)** – Deploying the project on a Kubernetes cluster and understanding its core concepts.
3. **Adding a Microservice (2 points)** – Developing and integrating a new microservice into the project architecture.
4. **Autoscaling, HA, Load Balancing, Nginx (2 points)** – Implementing automatic scaling, high availability, and load balancing
5. **Deploying to the Cloud (3 points)** – Deploying the project on Arvan Cloud, setting up backup and disaster recovery strategies.

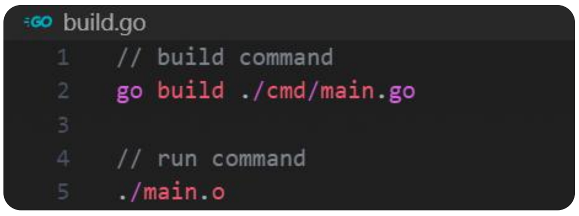
Notification Management System

A notification management and delivery system has been implemented using a microservices architecture, consisting of two main microservices. As a DevOps engineer, you are expected to set up and manage the deployment of this system, ensuring a reliable and scalable deployment environment.

1- Authentication (Auth) Microservice

This microservice handles user registration and authentication. Users can register and obtain an access token through the login service to access other services. Additionally, the token validation service allows retrieving the user ID associated with the token.

The microservice is implemented in Go and requires setting up a MySQL database as the first step. The service can be started in the terminal using the following commands:



build.go
```
// build command
go build ./cmd/main.go

// run command
./main.o
```

This service reads database connection details and JWT secret from *env/dev.env*, After launching the MySQL database, you need to update the database connection parameters (URL_DB) in *env/dev.env*.

Available Routes:

1- POST */user/register* – Registers a new user:
   - Request: JSON containing Email and Password.
   - Response: JSON with user details or 200 to show successful.

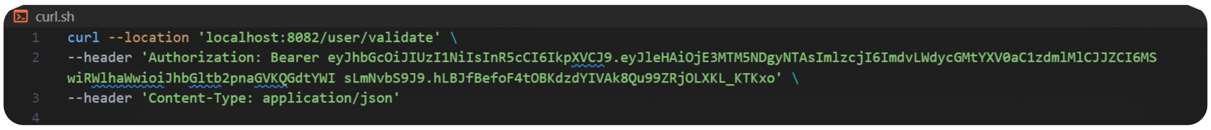
2- POST */user/login* – Authenticates a user and issues an access token.

- Request: JSON containing Email and Password.
- Response: JSON with the generated token and user ID.

3- POST */user/validate* – Validates the provided JWT token.

- Requires Authorization header with the format:
  Authorization: Bearer <Token>
- Response: JSON with the user ID.

Below is an example curl command to test the service:



curl.sh

```
curl --location 'localhost:8082/user/validate' \
--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MTM5NDgyNTAsImlzcjI6ImdvLWdycGMtYXV0aC1zdmlMlCJJZCI6MS
wiRWlhaWwioiJhbGltb2pnaGVKQGdtYWI sLmNvbS9J9.hLBJfBefoF4tOBKdzdYIVAk8Qu99ZRjOLXKL_KTKxo' \
--header 'Content-Type: application/json'
```

You can access the repository of this microservice at the following address:
[fum-course/fum-cloud-notification-auth-2025: Cloud Computing Course - Auth Microservice | Ferdowsi University of Mashhad]

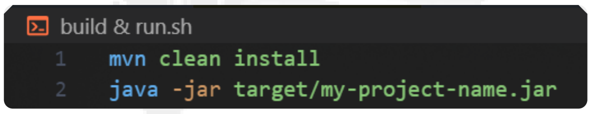
Additionally, the service runs a gRPC server on port 50051, which the core microservice uses to communicate with this service via the gRPC protocol.

2- Core Microservice

The Core Microservice is responsible for managing settings and sending notifications. For simplicity, email notifications will be the only type supported.

This microservice connects to a PostgreSQL database and an ActiveMQ queue. It is developed using the Spring framework and Java programming language.

To run this service, you will need Java 17 and Maven 3.6.3. Once the necessary dependencies are installed, you can run the project using the following commands:



build & run.sh

```
mvn clean install
java -jar target/my-project-name.jar
```

The server configuration files are located in the *resources/main/src* directory. You can configure your connections by modifying the *application.properties* file.

For documentation and to explore the available services and their inputs, you can access the Swagger UI at the following address: http://localhost:8080/docs

To enter the token, you can use the lock icon on the right side of each request in the Swagger interface.

You can access the repository of this microservice at the following address:
[fum-course/fum-cloud-notification-core-2025: Cloud Computing Course - Core Microservice | Ferdowsi University of Mashhad]

Phase 1 – Dockerization (2 point)

In this phase, you will prepare and deploy the project on the Docker platform using *Dockerfiles* and *Docker–Compose*. It is highly recommended to install Docker on *Windows WSL* or a *Linux distribution* for optional performance.

The following images are necessary for the deployment:

- PostgreSQL: *postgres:13.13-bullseye*
- ActiveMQ: *webcenter/activemq*
- MySQL: *mysql:latest*
- Authentication Service: *Golang-based image (preferably Alpine)*
- Core Service: *17-openjdk:3.6.3-maven*

You can find all images in dockerhub (you may need VPN).

Step 1 – Before launching the microservices, ensure that the database containers are properly created and configured. The databases should be persistent to prevent data loss upon container restarts.

Step 2 – Before running the microservices, modify the necessary configuration parameters to establish proper connectivity and functionality.

1- Authentication Service
   - Update the URL_DB parameter to point to the appropriate database.
2- Core Service
   - Database Configuration:
     - *project.spring.datasource.hikari.jdbc-url*
     - *project.spring.datasource.hikari.username*
     - *project.spring.datasource.hikari.password*
     - *project.spring.datasource.hikari.schema*
   - ActiveMQ Configuration:
     - *spring.activemq.broker*
     - *spring.activemq.user*
     - *spring.activemq.password*
   - Authentication Service Configuration:
     - *auth.service.base.url*
     - *auth.grpc.address*
     - *auth.grpc.port*

step 3 – Fallow project and networking rules ensures that the microservices communicate efficiently and securely within the Docker environment:

1- Networking
   a. Each microservices should be contained within its Docker network.
   b. Services must be accessible via localhost within the network.
   c. Port 50051 (Auth gRPC server) should be accessible only to the Core service and must not be exposed externally.
   d. Each microservice should only have access to its respective database, preventing unauthorized inter-service database access.

2- Data Persistence
   a. Databases must be configured to store data persistently, ensuring that no data is lost when containers restart.

3- Security
   a. External access to containers must be restricted to prevent unauthorized access.
   b. Only the necessary ports should be exposed to ensure secure deployment.

It is highly recommended to use multi-stage dockerfile to less image size.

Recommended Resource

- hub.docker.com
- Docker 101 Tutorial | Docker
- Install WSL | Microsoft Learn
- Install Ubuntu on WSL2 - Ubuntu on WSL documentation
- Developing in WSL

Deadline

- 22nd Farvardin

Delivery

- Upload dockerfiles & docker-compose on VU
- Upload a short video (max 8 min) and show project works
    - Test and show project URLs works
    - Restart containers and show data persistency
    - In auth microservice ping all DBs and other microservice
    - In core microservice ping all DBs and other microservice

In-person delivery

- The time of the in-person presentation will be announced.

You are free to use ChatGPT or any other AI, but you are expected to learn.

Phase 2 – Kubernetes (2 point)

In this phase, the project will migrate from Docker Compose to Kubernetes for improved scalability and management. The previous *docker-compose.yml* file is no longer applicable, and all project configurations must be defined within Kubernetes YAML manifests.

Step 1 – Install Kubernetes: For setting up the Kubernetes cluster, Kind is the preferred tool, but alternative solutions are also acceptable.

Step 2 – The following conditions must be met in this phase:

- The Kubernetes cluster must consist of at least three nodes: one master node and two worker nodes.
- Deploy all services and applications using Kubernetes Deployments, Pods, Services, ...
- Each service and its corresponding database must be organized into separate namespaces.
- Utilize ConfigMaps to store and manage environment variables and configuration settings.
- Secure sensitive credentials, such as database usernames and passwords, using Secrets.
- Database access must be restricted to within the Kubernetes cluster.
- Ensure data persistence by preventing database data loss after restarting. This should be achieved by utilizing Volumes, implementing Persistent Volume Claims (PVCs) and Persistent Volumes (PVs) for deployments.
- Set resource limits for each service to control RAM and CPU usage efficiently.
- Each Deployment must be scheduled on a dedicated node, ensuring proper distribution across the cluster.
- All the networking that was implemented in Docker in the previous phase must also be maintained in this phase.
- Do not place all Kubernetes yml configuration in a single file.

Recommended Resource

- kind
- Kubernetes

Deadline

- 15th Ordibehesht

Delivery

- Upload yml files on VU
- Upload a short video (max 8 min) and show project works on VU
  - Test and show project URLs works
  - Show all pods, services, deployments, … by details
  - In auth microservice ping all DBs and other microservice
  - In core microservice ping all DBs and other microservice

In-person delivery

- The time of the in-person presentation will be announced.

You are free to use ChatGPT or any other AI, but you are expected to learn.

Phase 3 – Add your Micro Service (2 point)

In Phase Three, you are required to add your microservice to the project. This microservice must implement a RESTful API in any programming language of your choice. The microservice should manage authentication in a straightforward manner, which means it must include the following endpoints:

1. GET */auth/users*: This endpoint should return to a paginated list of all users who have registered in the system.
2. DELETE */auth/user*: This endpoint should accept either an email or user ID as a parameter and delete the specified user from the database (handle exceptions).

Response must be in JSON format, also your service must have access to the auth database within a separate network and should not be able to access any other services.

Additionally, your service must be deployed alongside the other services of this project in a Kubernetes environment. The API should be accessible from outside the cluster. The choice of programming language, configuration, port, and other settings is at your discretion.

Deadline

- 3rd Khordad

Delivery

- Upload your microservice files (contain new dockerfile & ymls) on VU
- Upload a short video (max 5 min) and show project works
  - Test and show project URLs works
  - Short overview of your microservice
  - Show new pods, services, deployments, … by details
  - In your microservice ping all DBs and other microservices

In-person delivery

- The time of the in-person presentation will be announced.

You are free to use ChatGPT or any other AI, but you are expected to learn.

Phase 4 – HA proxy, Nginx and Auto scalable (2 point)

Step 1 – To ensure that the services can handle high loads and to prevent service failures, implement automatic scaling for your microservices. This means that the services should automatically scale up when they reach a specified high load threshold and scale down when the load decreases (No need to scale the database).

Step 2 – For load distribution among the scaled services and to ensure access to the services, utilize an internal NGINX instance with a custom routing configuration. This will help manage incoming requests and distribute them evenly across the available instances of the services.

Step 3 – To enhance system resilience against increased load or potential failures, increase the number of master nodes to two. Use external HAProxy to manage load balancing and synchronization between the master nodes effectively.

In this phase, the scaling thresholds and custom routing configurations are your responsibility, and you have the discretion to determine the parameters based on your project's needs.

Deadline

- 23rd Khordad

Delivery

- Upload your yml files on VU
- Upload a short video (max 10 min) and show project works
    - Test and show project URLs works
    - Show all pods, services, deployments, nodes, … by details
    - Explain Nginx, HA Proxy and auto scaling configurations

You are free to use ChatGPT or any other AI, but you are expected to learn.

Phase 5 - Migration to Arvan Cloud (3 point)

In this phase, you will migrate all the services that you have implemented in the previous phases to Arvan Cloud.

Step 1 – Migrate all the microservices to Arvan Cloud. Ensure that each service is properly configured and that they function correctly in the new environment. This may involve adjusting configurations, networking settings, and deployment processes to align with Arvan Cloud's infrastructure.

Step 2 – Take a backup of the auth database on one of the Arvan Cloud services. After successfully creating the backup, delete the database and restore the deleted data by replacing it with the backup.

Deadline

- 15th Tir

Delivery

- Upload a video (max 20 min) on VU
- Record a short video demonstrating the successful execution of Step 1 testing of Step 2.
- Show that the services are functioning correctly after migration and verify that the database restoration process works as intended. Submit the video as your delivery.

Your Arvan Cloud account will be charged.