

Project 9 : LDAP deployment

Team

- Arsen Galiev –
- Peter Zavadskii –
- Ilya-Linh Nguen –
- Daniil Tskhe –
- Danila Kochegarov –

Introduction

Modern systems require reliable protection of user data and convenient monitoring. This project provides both, combining authentication with modern observability in a container environment.

Here's the tech stack that makes it work:

- LDAP keeps user management centralized and secure.
- Keycloak steps in for authentication, handling OIDC.
- Kafka acts as streaming logs and metrics in real time.
- Golang powers the backend — fast, efficient, and perfect for microservices.
- Docker & Docker-compose tie it all together in containers and create a network between them to make our system able to work everywhere.

The main goal is to show the implementation of remote authentication using OIDC over the ldap protocol.

Methods

Tools and Technologies:

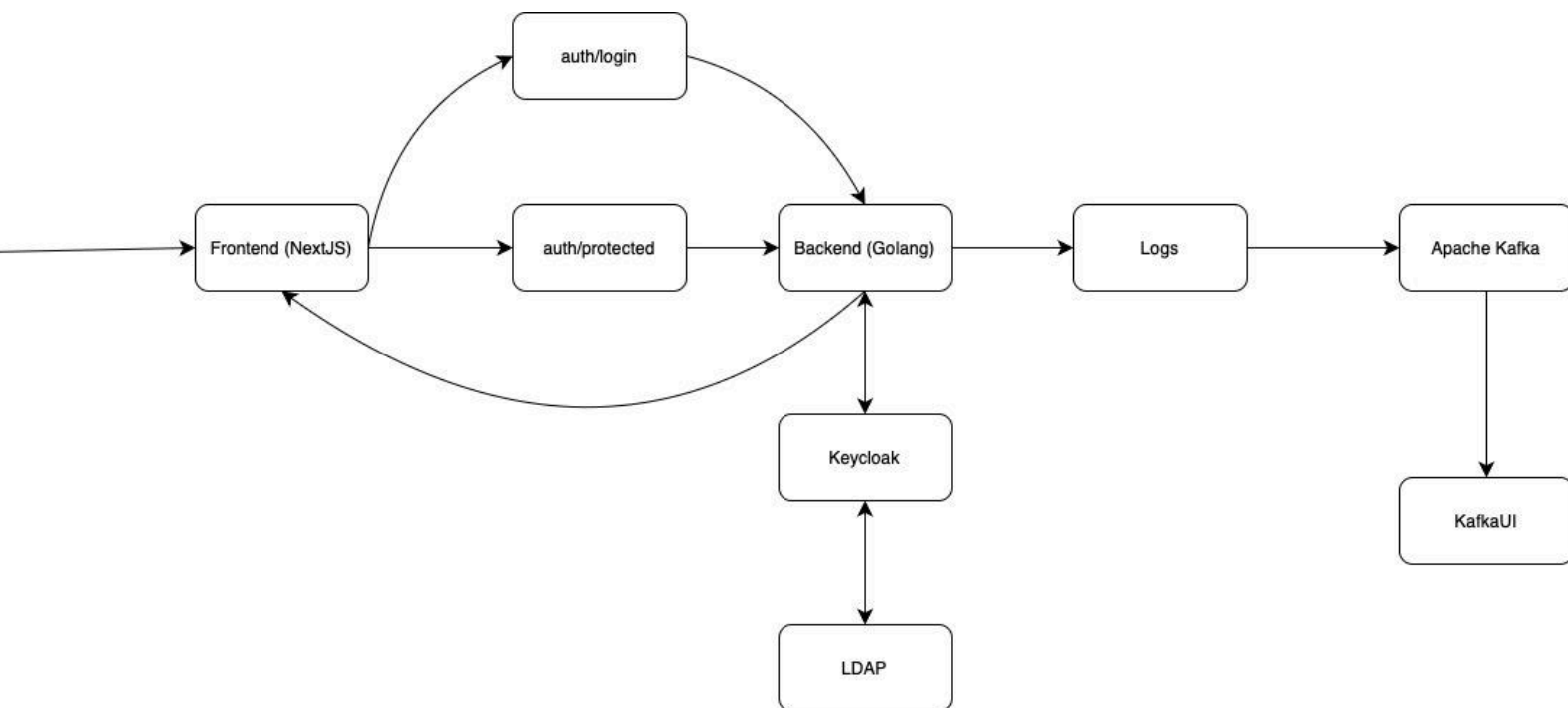
- **Golang**: Backend service
- **NextJS**: Frontend service
- **Ldap**: (Lightweight LDAP implementation) User storage service.
- **Keycloak**: Authentication service (connected to LDAP server)
- **Apache Kafka**: Centralized logging system
- **Kafka UI**: Web interface for Kafka topics
- **Docker/Docker-compose**: Containerization and orchestration

Configurations and Architecture:

- **Keycloak** connects to **LDAP (ldap realization)** using Ldap protocol.

- The **Golang backend** validates user tokens issued by Keycloak.
- Application logs are sent asynchronously to a **Kafka topic**.
- **Kafka UI** provides access to the collected logs.
- All the elements are wrapped in **Docker containers** are orchestrated using **Docker-compose**

Therefore, the total workflow looks like that :



Picture 1: LDAP deployment project – architecture

Results

Our team has successfully implemented the required logic following all the criteria in the task. Users can login using LDAP protocol and obtain the information about him/her including roles, email and username.

For systematization, I'll clarify. We have implemented the following components:

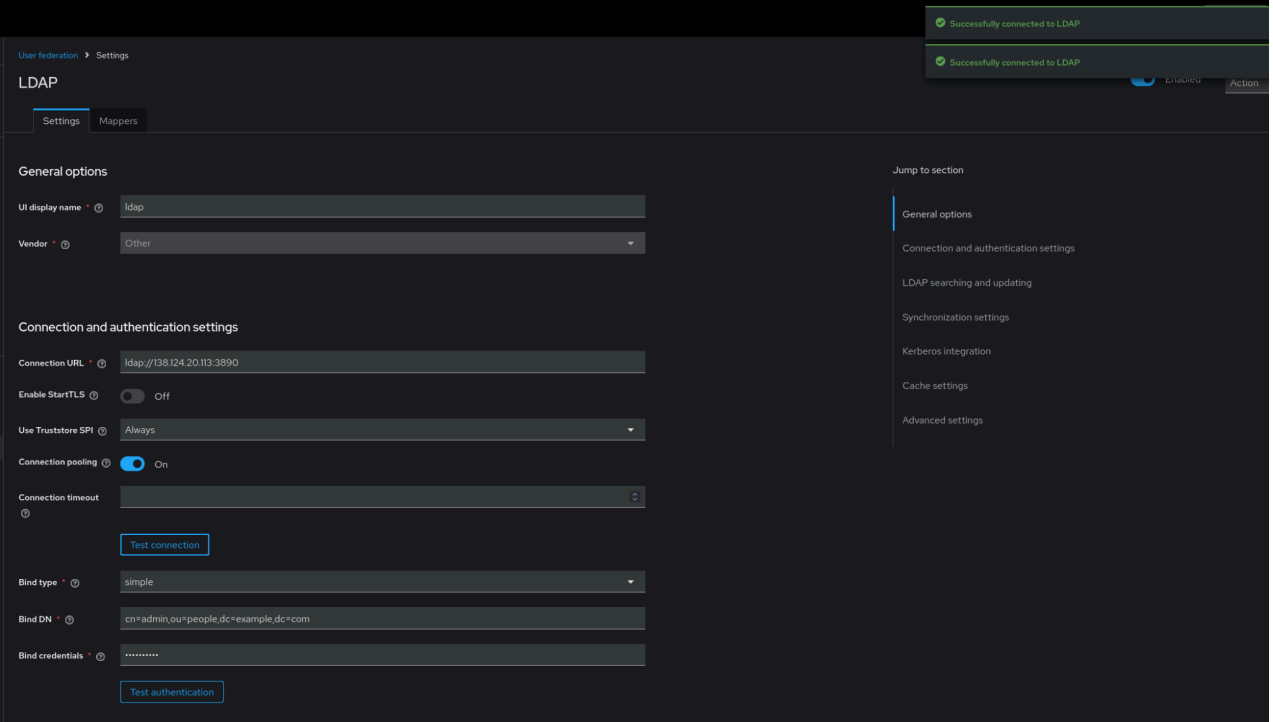
- **Secure authentication** through Keycloak with LDAP backend.

- **Real-time centralized logging** via Kafka.
- **Fully containerized deployment** with Docker/Docker-compose.
- **Frontend-backend communication** via secure APIs.

Performance observations:

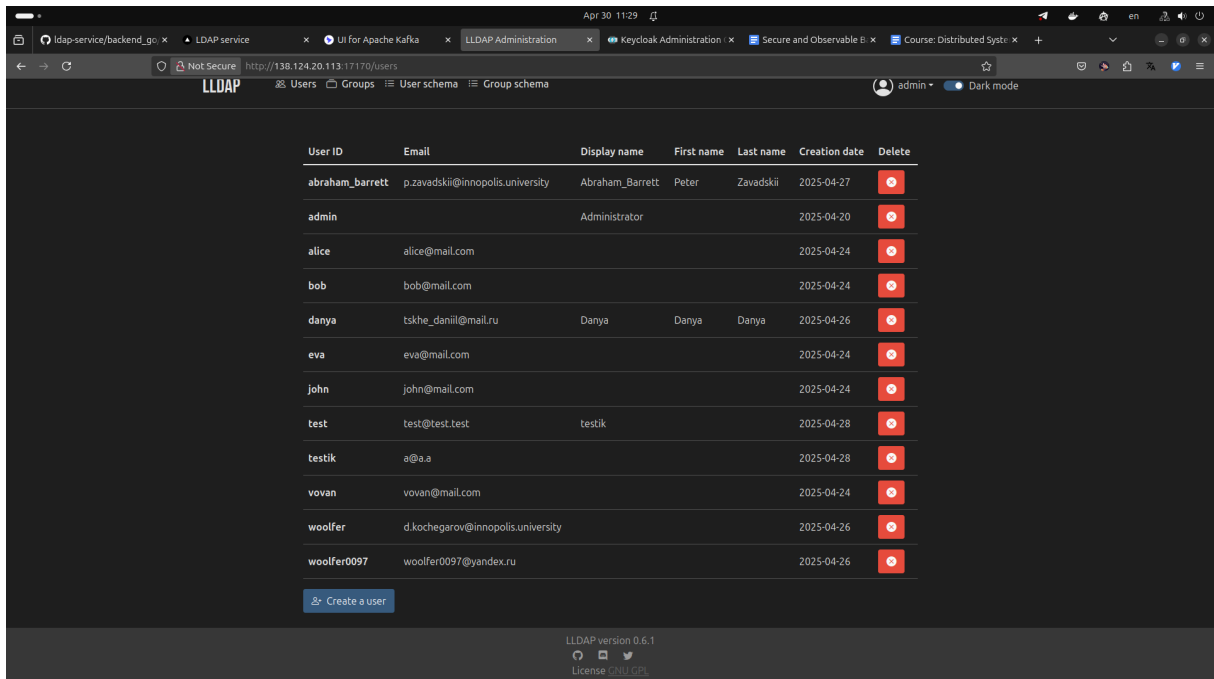
- Authentication latency was measured to be consistently low (below 200ms under nominal load).
- Kafka log publishing remained stable under test conditions (~100 messages/second).

Screenshots of product:

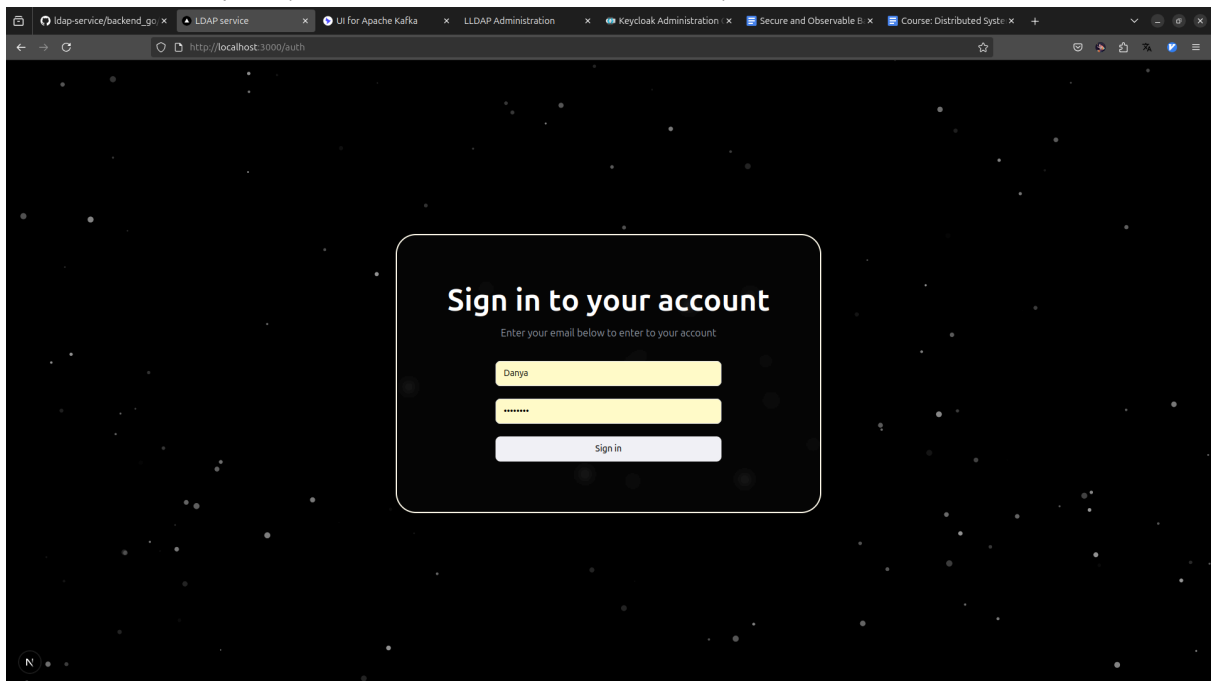


The screenshot displays the Keycloak administration console for LDAP configuration. At the top, a breadcrumb shows 'User federation > Settings'. The main heading is 'LDAP', with tabs for 'Settings' and 'Mappers'. Two green success messages at the top right state 'Successfully connected to LDAP'. The 'General options' section includes 'UI display name' set to 'ldap' and 'Vendor' set to 'Other'. The 'Connection and authentication settings' section contains: 'Connection URL' as 'ldap://138.124.20.113:3890', 'Enable StartTLS' as 'Off', 'Use Truststore SPI' as 'Always', 'Connection pooling' as 'On', a 'Connection timeout' field, a 'Test connection' button, 'Bind type' as 'simple', 'Bind DN' as 'cn=admin,ou=people,dc=example,dc=com', and 'Bind credentials' as masked text with a 'Test authentication' button. A 'Jump to section' sidebar on the right lists: 'General options', 'Connection and authentication settings', 'LDAP searching and updating', 'Synchronization settings', 'Kerberos integration', 'Cache settings', and 'Advanced settings'.

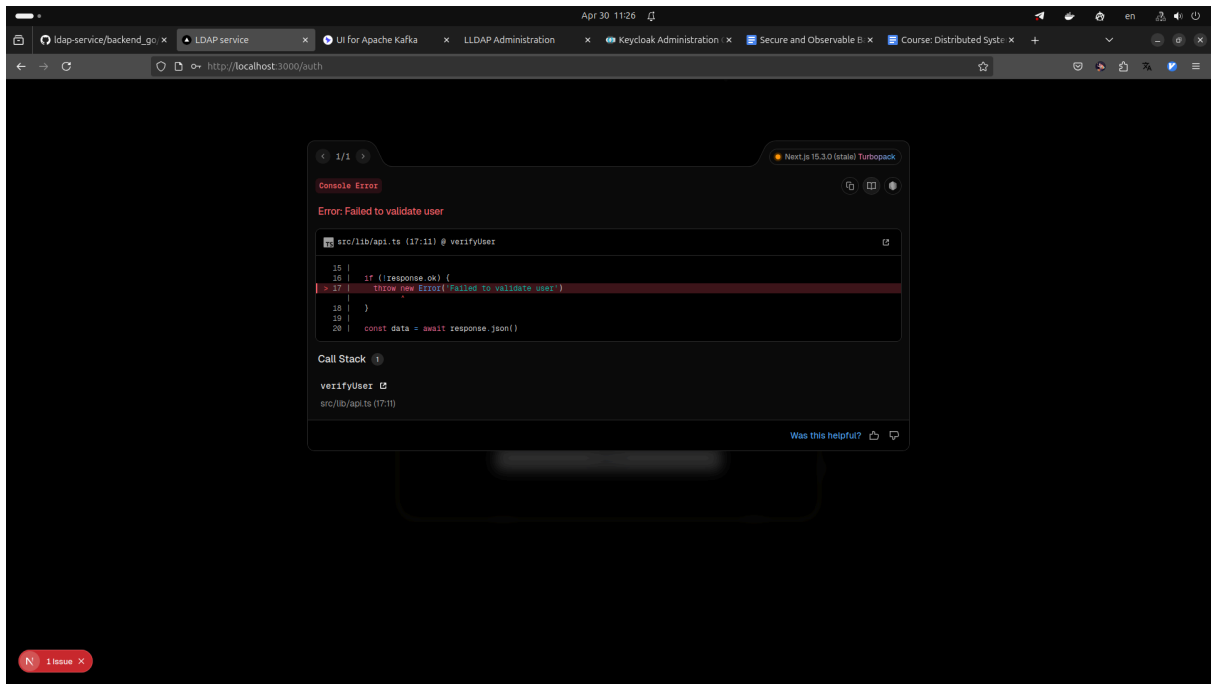
Picture 2: Keycloak configuration to connect to ldap service



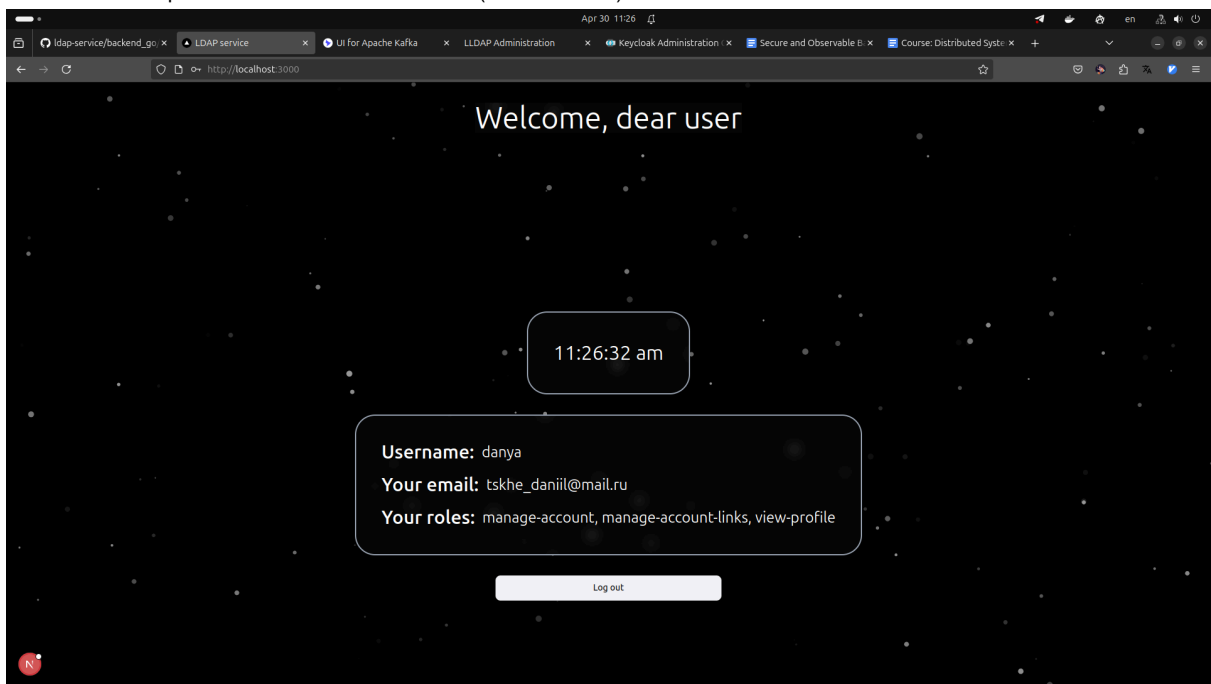
Picture 3: Users on lldap side (satisfies the conditions about 5 or more users)



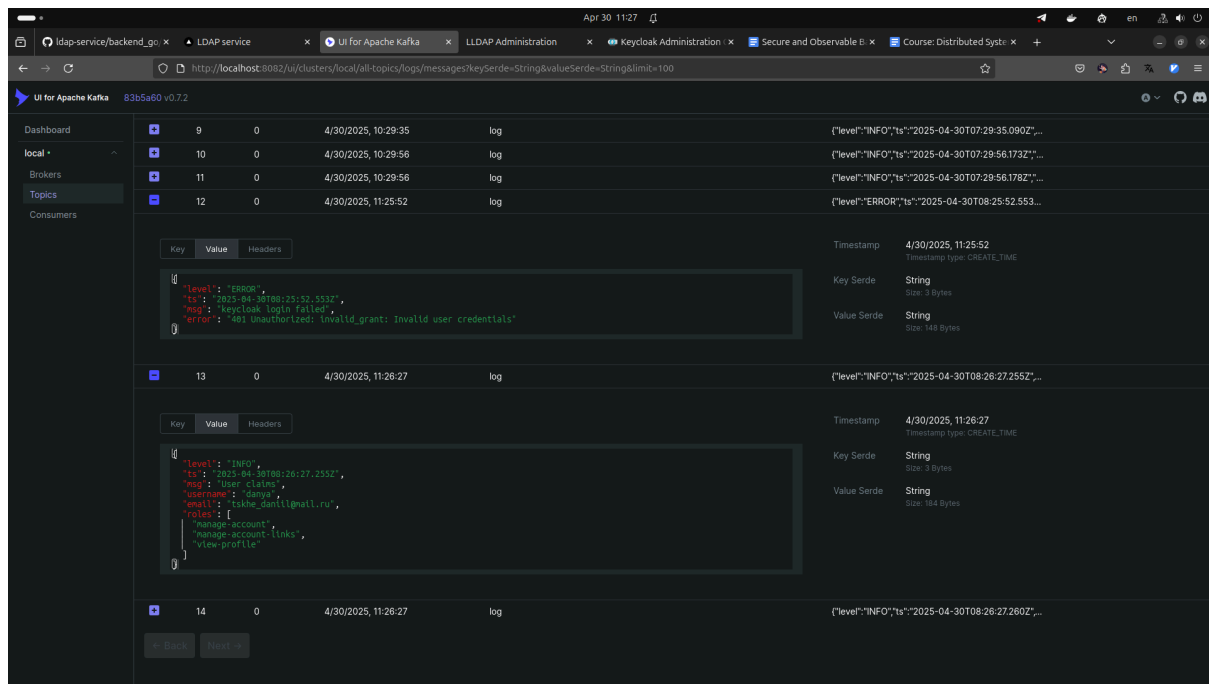
Picture 4: Authorization page with username and password fields



Picture 5: Error if password or username is invalid (task condition)



Picture 6: Page with user's information if username and password are correct



Picture 7: Apache Kafka logs about session and information about user (task condition)

Github repo with project:

<https://github.com/projacktor/ldap-service>

Demo video:

https://drive.google.com/file/d/1dz_RqyCx2glgP2sokr1ZSEDKL8TBukYp/view?usp=sharing

Discussion

During development, we discovered Keycloak and Ldap (Idap protocol). These services have opened up a new opportunity for us to securely store user data and authenticate.

We also encountered some problems during development: keycloak deployment requires https, not http (we solved it using a self-signed certificate and ignoring warnings on the backend about connecting using a self-signed certificate using special parameters). There were also minor problems with the backend and keycloak connection due to ignorance of the effect of the parameters on the connection itself (we decided after a very thorough study of the repository and documentation).

References

- <https://github.com/keycloak/keycloak>
- <https://github.com/lldap/ldap>
- <https://github.com/Nerzal/gocloak>
- <https://nextjs.org/docs>
- <https://medium.com/@allusaiprudhvi999/authentication-and-authorization-in-golang-microservice-using-an-open-source-iam-called-keycloak-46f03a26248f>
- <https://github.com/coreos/go-oidc>
- https://gitlab.com/ALLU999/gomicroservicewithauth/-/tree/main?ref_type=heads
- <https://habr.com/ru/articles/810061/>
- <https://www.youtube.com/watch?v=tgLqnYSTZw4&t=453s>