Project: Local Identity & Access Management (IAM) Implementation using Keycloak and Docker

Objective

The primary objective of this project was to set up a complete Identity and Access Management (IAM) system on a local machine using Keycloak. The project involved running the Keycloak server within a Docker container and configuring a full user authentication flow for a sample application, demonstrating a practical understanding of modern IAM concepts.

Technologies Used

- Keycloak: An open-source Identity and Access Management solution.
- Docker: A containerization platform used to run the Keycloak instance locally.
- PowerShell (Windows Terminal): Used to execute Docker commands and manage the container.

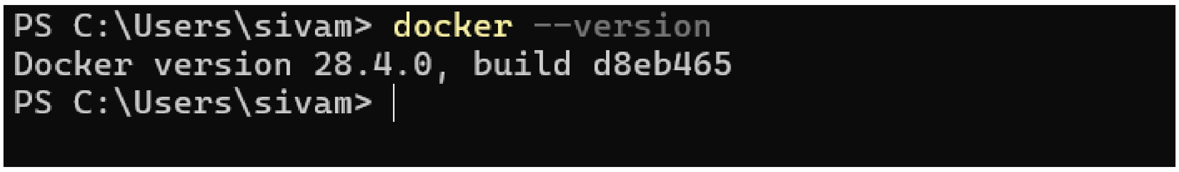
Project Steps

The project was completed by following the steps outlined below. The corresponding screenshots taken at each milestone can be inserted as indicated.

1. Docker Installation

First, Docker Desktop was installed on Windows to provide the necessary containerization environment. The installation was verified by running the following command in the terminal.

```
docker –version
```

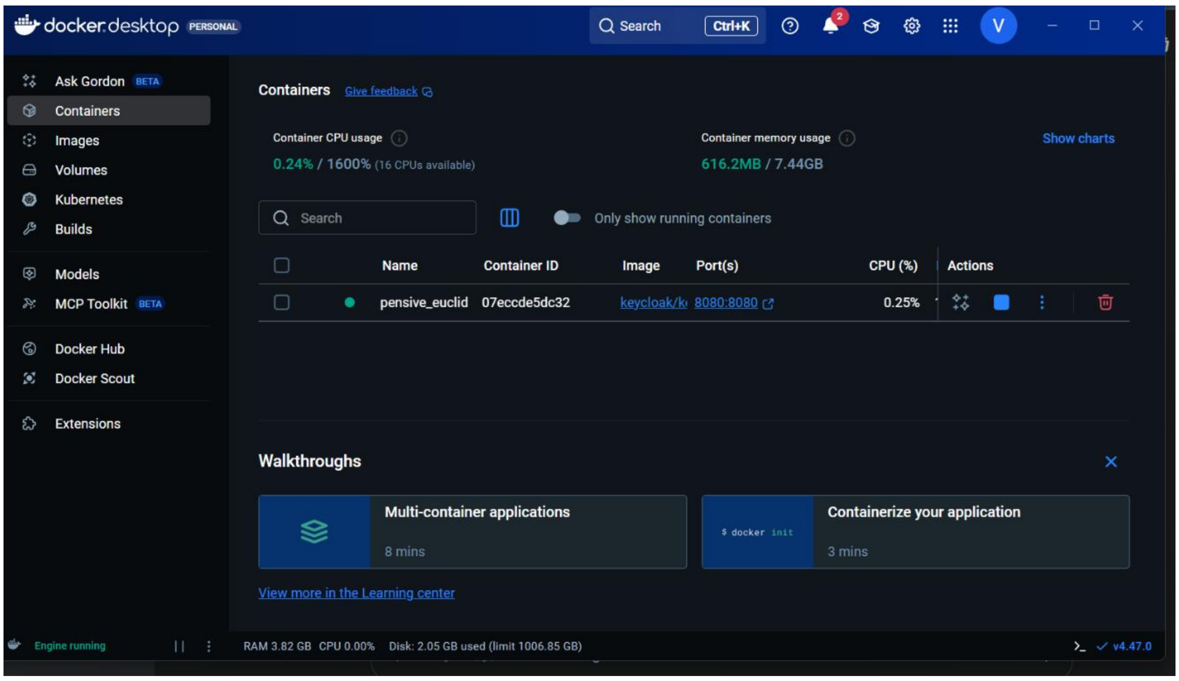


```
PS C:\Users\sivam> docker --version
Docker version 28.4.0, build d8eb465
PS C:\Users\sivam>
```

2. Running the Keycloak Container

Once Docker was ready, the Keycloak server was started in a Docker container using the following command. This command also mapped port 8080 and set the initial administrator credentials (admin/admin).

```
docker run -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:latest start-dev
```


docker desktop PERSONAL

Search Ctrl+K

Ask Gordon BETA
Containers
Images
Volumes
Kubernetes
Builds
Models
MCP Toolkit BETA
Docker Hub
Docker Scout
Extensions

Containers Give feedback

Container CPU usage
0.24% / 1600% (16 CPUs available)

Container memory usage
616.2MB / 7.44GB

Show charts

Search

Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Actions
pensive_euclid	07eccde5dc32	keycloak/k	8080:8080	0.25%	

Walkthroughs

Multi-container applications
8 mins

Containerize your application
3 mins

View more in the Learning center

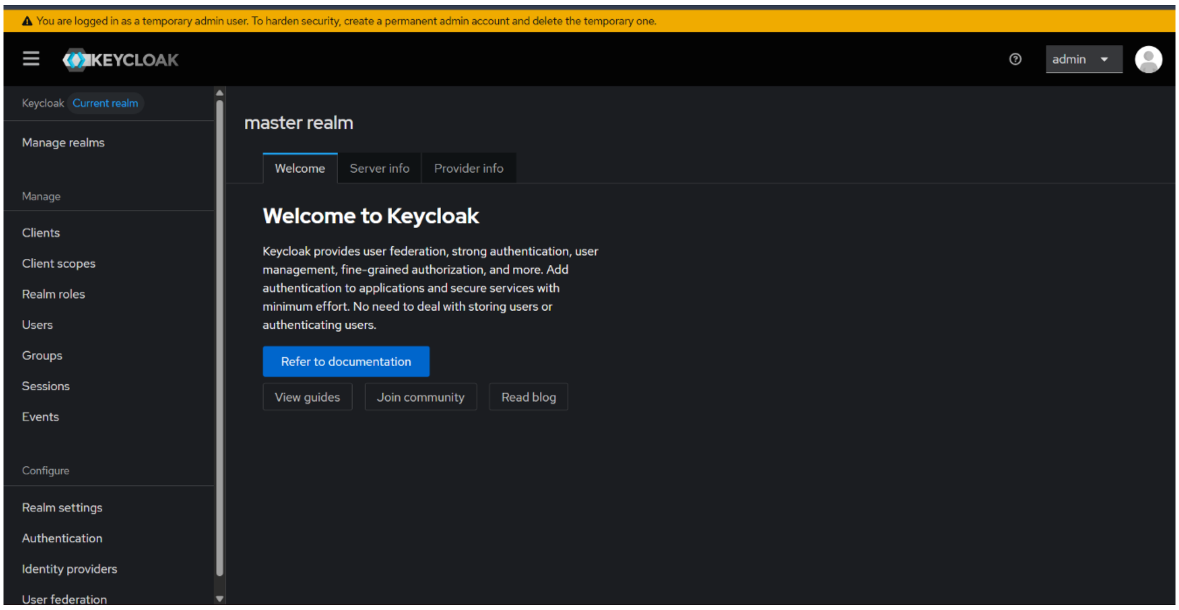
Engine running

RAM 3.82 GB CPU 0.00% Disk: 2.05 GB used (limit 1006.85 GB)

v4.47.0

3. Accessing the Admin Console

With the server running, the Admin Console was accessed at http://localhost:8080/ and logged into using the administrator credentials.


You are logged in as a temporary admin user. To harden security, create a permanent admin account and delete the temporary one.

KEYCLOAK

admin

Keycloak Current realm

Manage realms

Manage

Clients
Client scopes
Realm roles
Users
Groups
Sessions
Events

Configure

Realm settings
Authentication
Identity providers
User federation

master realm

Welcome Server info Provider info

Welcome to Keycloak

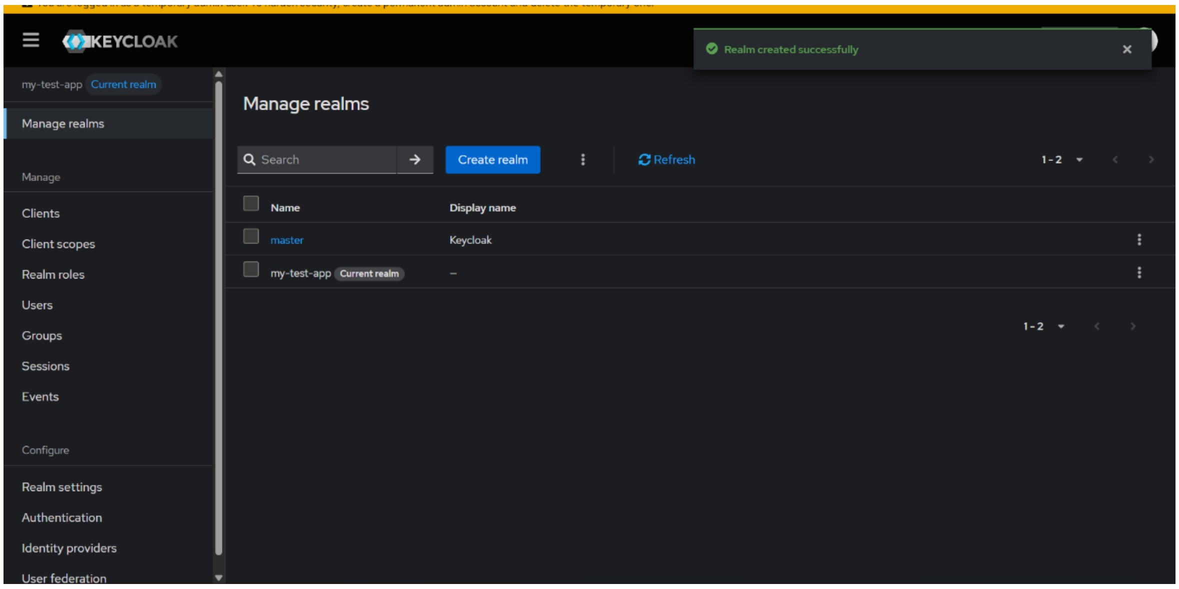
Keycloak provides user federation, strong authentication, user management, fine-grained authorization, and more. Add authentication to applications and secure services with minimum effort. No need to deal with storing users or authenticating users.

Refer to documentation

View guides Join community Read blog

4. Realm Configuration

A new Realm named my-test-app was created to provide a separate, secure space for our application's users and clients, isolating them from the master administration realm.



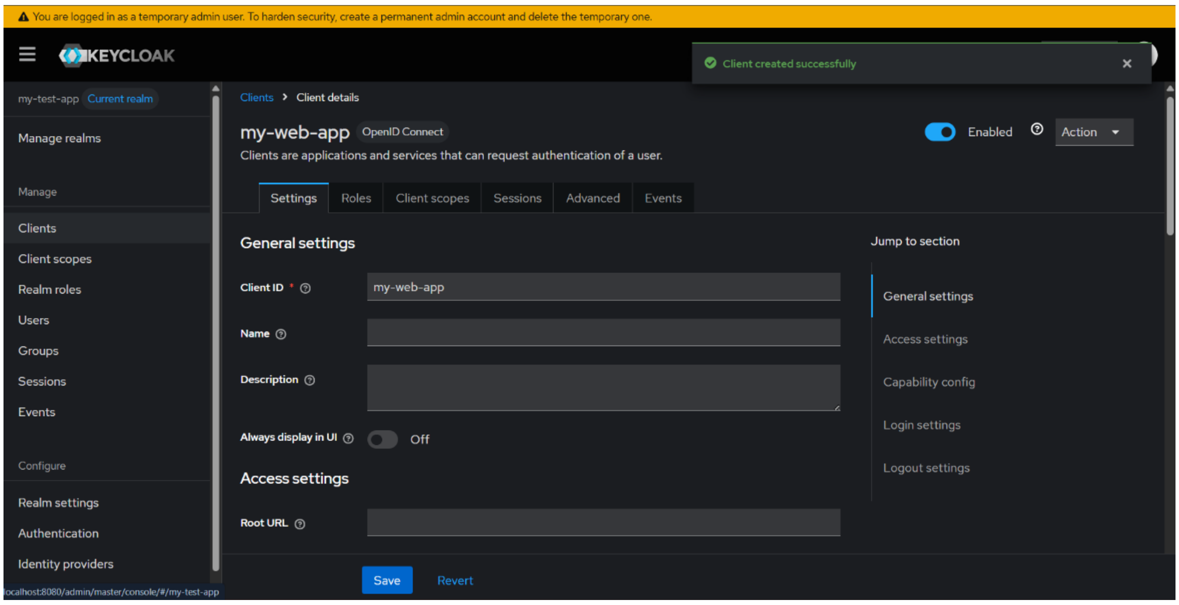
Realm created successfully

Manage realms

Name	Display name
master	Keycloak
my-test-app Current realm	–

5. Client Configuration

Within the new realm, a Client named my-web-app was created to represent our sample application. A crucial step was setting the Valid Redirect URIs to https://www.google.com to define where the user should be sent after a successful login.


You are logged in as a temporary admin user. To harden security, create a permanent admin account and delete the temporary one.

Client created successfully

my-web-app OpenID Connect

Clients are applications and services that can request authentication of a user.

General settings

Client ID * my-web-app

Name

Description

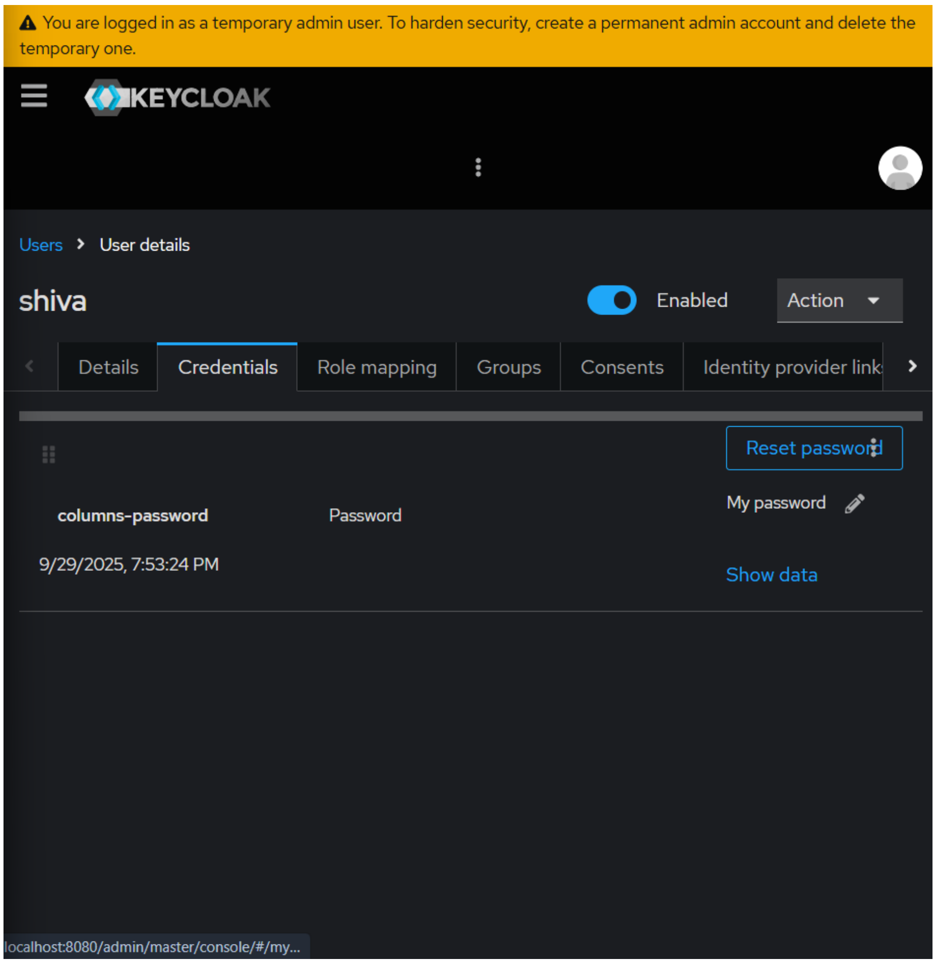
Always display in UI Off

Access settings

Root URL

6. User Creation

To test the login process, a new user named testuser was created, and its password was set to password123.



You are logged in as a temporary admin user. To harden security, create a permanent admin account and delete the temporary one.

Users > User details

shiva

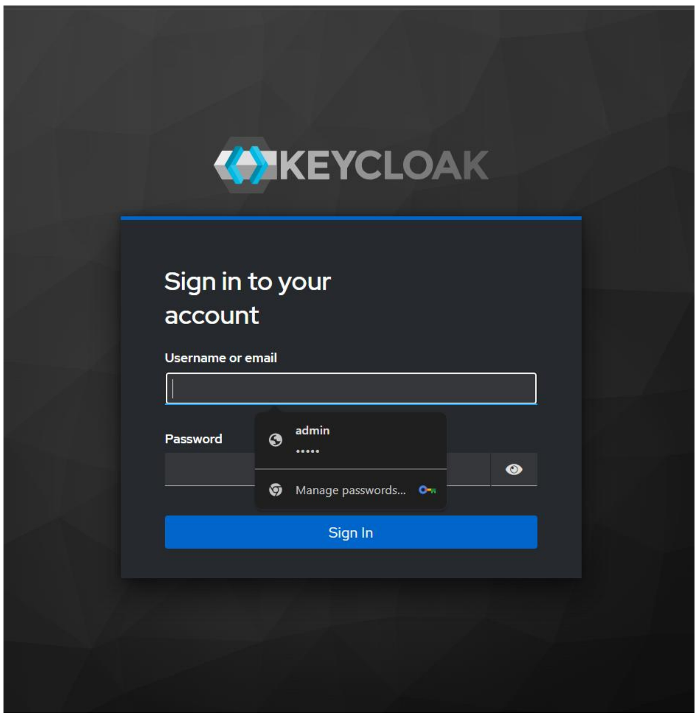
Enabled

columns-password | Password | My password
9/29/2025, 7:53:24 PM

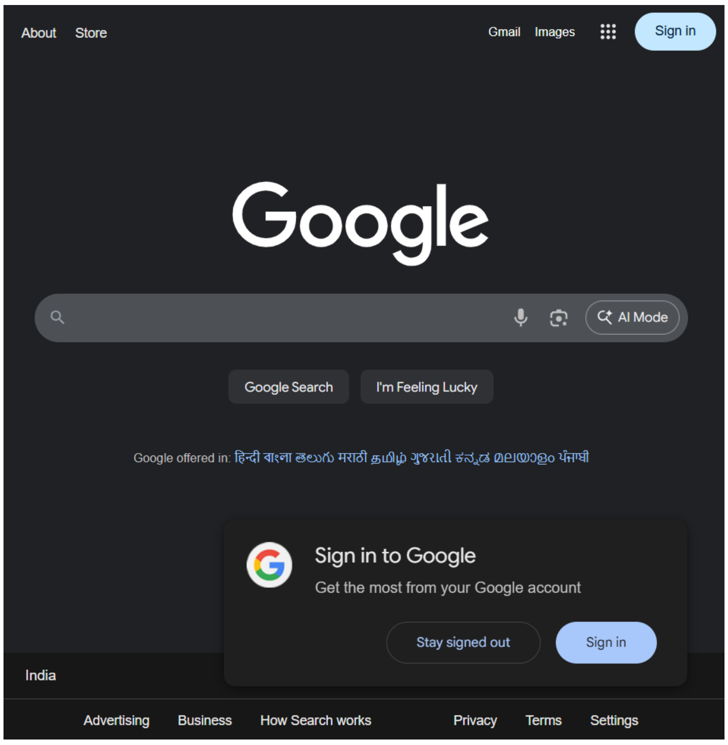
localhost:8080/admin/master/console/#/my...

7. Testing the Authentication Flow

Finally, the end-to-end login process was tested using the following specially constructed URL: http://localhost:8080/realms/my-test-app/protocol/openid-connect/auth?client_id=my-web-app&response;_type=code&scope;=openid&redirect;_uri=https://www.google.com This flow correctly presented the login page.



After authenticating with the testuser credentials and updating the user profile, the system successfully redirected the browser to the Google homepage, confirming the entire flow was working correctly.



Conclusion

This project successfully demonstrated the setup of a basic IAM server using Keycloak and Docker. Despite encountering some technical challenges, they were systematically troubleshooted, leading to a successful implementation of a complete authentication and redirect flow. The project provided valuable hands-on experience with foundational IAM concepts and tools.