CLOUD SECURITY & MANAGEMENT LAB

Name: Harsh Ranjan

SAP ID: 500097019

Roll no: R2142211262

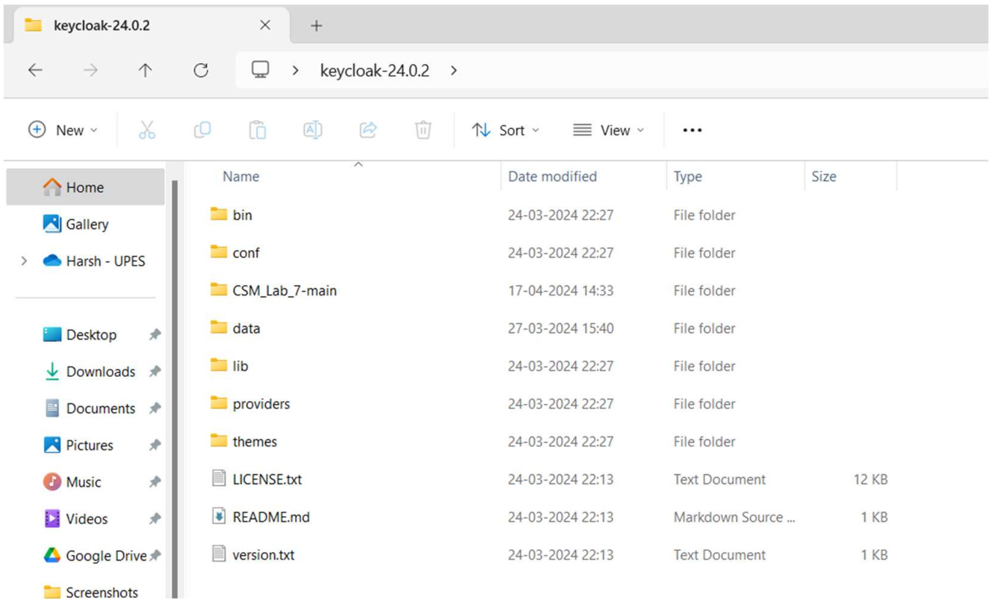
Batch :B7

SUBMISSION TO:

Ms. Avita Katal

Experiment 7a) Understanding Keycloak-Open Source identity and access management

STEP 1: Download Keycloak Package , and Extract the Package from https://www.keycloak.org/downloads

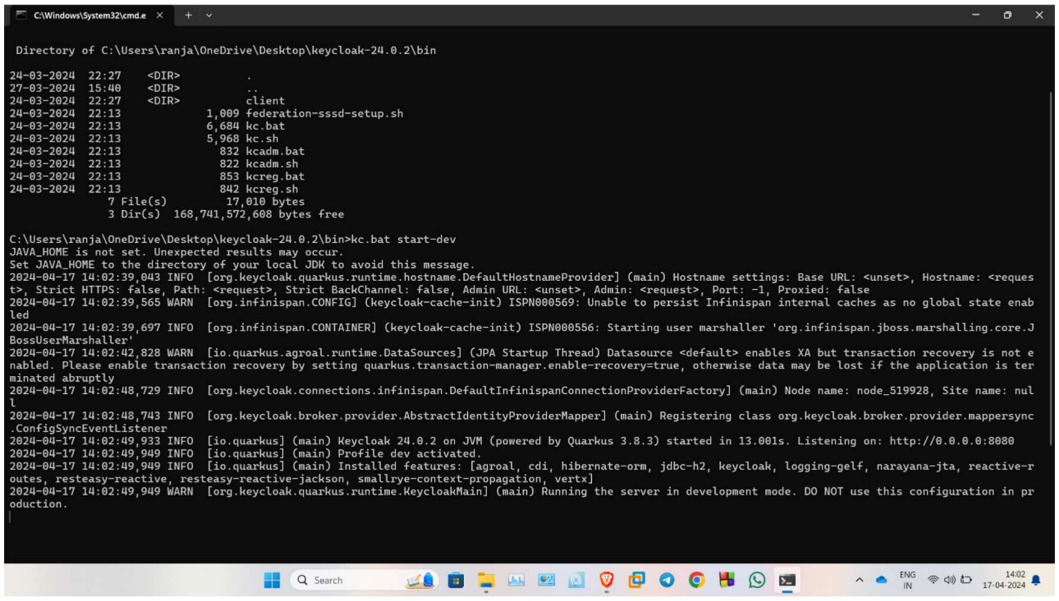


Name	Date modified	Type	Size
bin	24-03-2024 22:27	File folder	
conf	24-03-2024 22:27	File folder	
CSM_Lab_7-main	17-04-2024 14:33	File folder	
data	27-03-2024 15:40	File folder	
lib	24-03-2024 22:27	File folder	
providers	24-03-2024 22:27	File folder	
themes	24-03-2024 22:27	File folder	
LICENSE.txt	24-03-2024 22:13	Text Document	12 KB
README.md	24-03-2024 22:13	Markdown Source ...	1 KB
version.txt	24-03-2024 22:13	Text Document	1 KB

STEP 2: Open the Command Line inside the bin folder

cd bin

then run this command: “kc.bat start-dev”

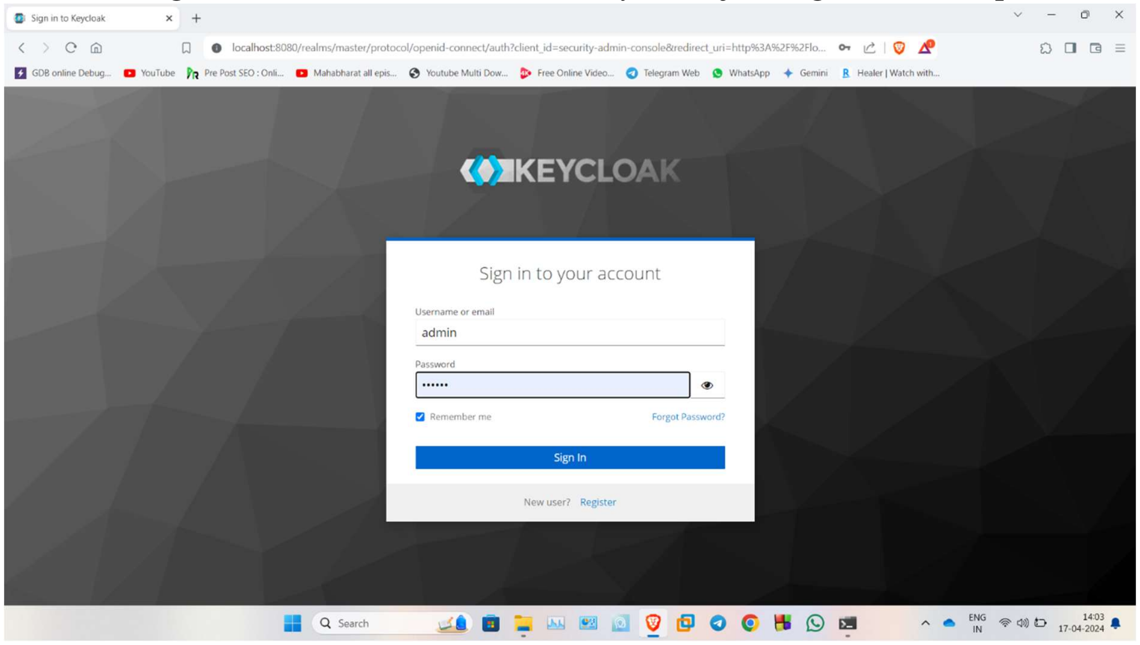


```
 Directory of C:\Users\ranja\OneDrive\Desktop\keycloak-24.0.2\bin

24-03-2024  22:27    <DIR>          .
27-03-2024  15:40    <DIR>          ..
24-03-2024  22:27    <DIR>          client
24-03-2024  22:13             1,009 federation-sssd-setup.sh
24-03-2024  22:13             6,684 kc.bat
24-03-2024  22:13             5,968 kc.sh
24-03-2024  22:13               832 kcadm.bat
24-03-2024  22:13               822 kcadm.sh
24-03-2024  22:13               853 kcreg.bat
24-03-2024  22:13               842 kcreg.sh
               7 File(s)         17,010 bytes
               3 Dir(s)  168,741,572,608 bytes free

C:\Users\ranja\OneDrive\Desktop\keycloak-24.0.2\bin>kc.bat start-dev
JAVA_HOME is not set. Unexpected results may occur.
Set JAVA_HOME to the directory of your local JDK to avoid this message.
2024-04-17 14:02:39,043 INFO  [org.keycloak.quarkus.runtime.hostname.DefaultHostnameProvider] (main) Hostname settings: Base URL: <unset>, Hostname: <reques
t>, Strict HTTPS: false, Path: <request>, Strict BackChannel: false, Admin URL: <unset>, Admin: <request>, Port: -1, Proxied: false
2024-04-17 14:02:39,565 WARN  [org.infinispan.CONFIG] (keycloak-cache-init) ISPN000569: Unable to persist Infinispan internal caches as no global state enab
led
2024-04-17 14:02:39,697 INFO  [org.infinispan.CONTAINER] (keycloak-cache-init) ISPN000556: Starting user marshaller 'org.infinispan.jboss.marshalling.core.J
BossUserMarshaller'
2024-04-17 14:02:42,828 WARN  [io.quarkus.agroal.runtime.DataSources] (JPA Startup Thread) Datasource <default> enables XA but transaction recovery is not e
nabled. Please enable transaction recovery by setting quarkus.transaction-manager.enable-recovery=true, otherwise data may be lost if the application is ter
minated abruptly
2024-04-17 14:02:48,729 INFO  [org.keycloak.connections.infinispan.DefaultInfinispanConnectionProviderFactory] (main) Node name: node_519928, Site name: nul
l
2024-04-17 14:02:48,743 INFO  [org.keycloak.broker.provider.AbstractIdentityProviderMapper] (main) Registering class org.keycloak.broker.provider.mappersync
.ConfigSyncEventListener
2024-04-17 14:02:49,933 INFO  [io.quarkus] (main) Keycloak 24.0.2 on JVM (powered by Quarkus 3.8.3) started in 13.001s. Listening on: http://0.0.0.0:8080
2024-04-17 14:02:49,949 INFO  [io.quarkus] (main) Profile dev activated.
2024-04-17 14:02:49,949 INFO  [io.quarkus] (main) Installed features: [agroal, cdi, hibernate-orm, jdbc-h2, keycloak, logging-gelf, narayana-jta, reactive-r
outes, resteasy-reactive, resteasy-reactive-jackson, smallrye-context-propagation, vertx]
2024-04-17 14:02:49,949 WARN  [org.keycloak.quarkus.runtime.KeycloakMain] (main) Running the server in development mode. DO NOT use this configuration in pr
oduction.
```

STEP 3: Open the web browser and open the following link:

localhost:8080 or depending upon your port number

STEP 4: Register for admin role or if already done just login to admin portal.


Sign in to Keycloak
localhost:8080/realms/master/protocol/openid-connect/auth?client_id=security-admin-console&redirect_uri=http%3A%2F%2Flo...
KEYCLOAK
Sign In
New user? Register
Forgot Password?

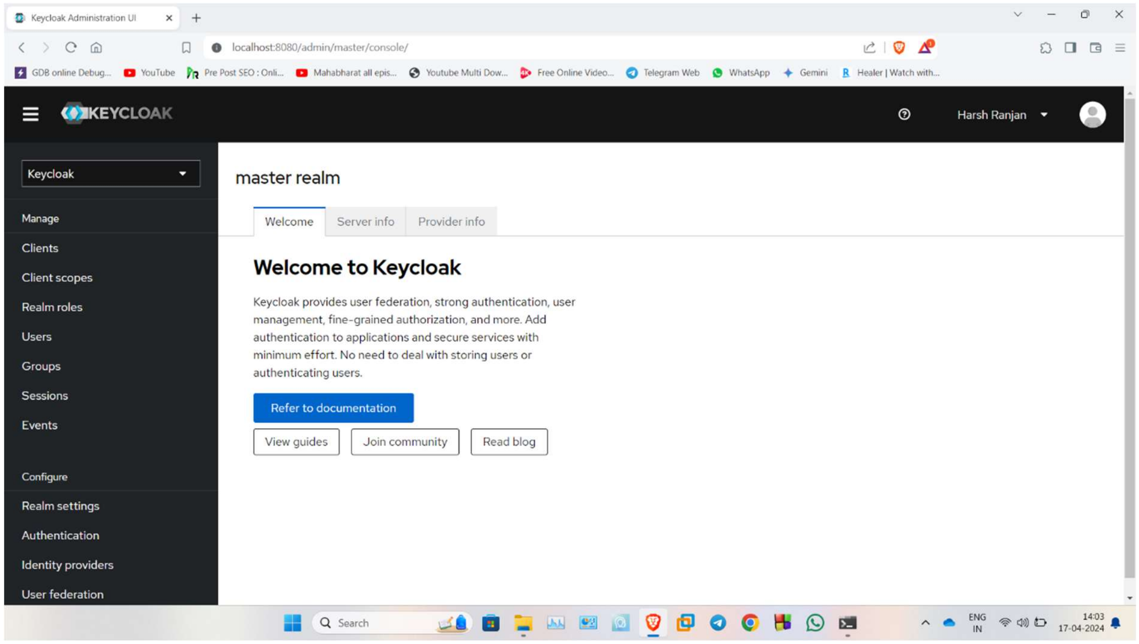
Sign in to your account

Username or email: admin

Password: ••••••

Remember me ✓

STEP 5: Now you can access Key-Cloak and create single signon and perform other operations.

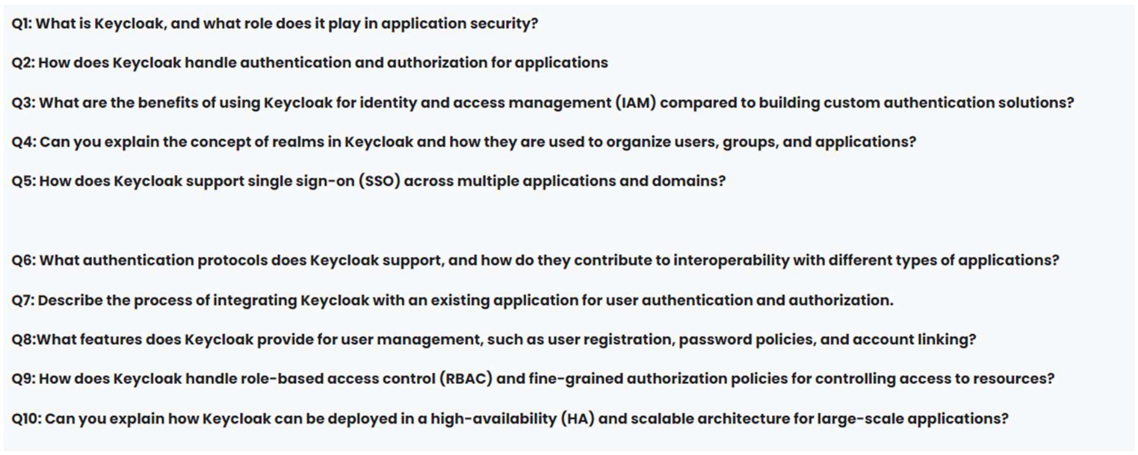


master realm

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.

QUESTIONS:



Q1: What is Keycloak, and what role does it play in application security?

Q2: How does Keycloak handle authentication and authorization for applications

Q3: What are the benefits of using Keycloak for identity and access management (IAM) compared to building custom authentication solutions?

Q4: Can you explain the concept of realms in Keycloak and how they are used to organize users, groups, and applications?

Q5: How does Keycloak support single sign-on (SSO) across multiple applications and domains?

Q6: What authentication protocols does Keycloak support, and how do they contribute to interoperability with different types of applications?

Q7: Describe the process of integrating Keycloak with an existing application for user authentication and authorization.

Q8:What features does Keycloak provide for user management, such as user registration, password policies, and account linking?

Q9: How does Keycloak handle role-based access control (RBAC) and fine-grained authorization policies for controlling access to resources?

Q10: Can you explain how Keycloak can be deployed in a high-availability (HA) and scalable architecture for large-scale applications?

Q1: What is Keycloak, and what role does it play in application security?

ANSWER: Keycloak is a powerful open-source Identity and Access Management (IAM) solution developed by Red Hat. It serves as a centralized authentication and authorization service, offering a range of features to secure applications and APIs. Keycloak plays a pivotal role in application security by providing a comprehensive suite of tools for managing user identities, enforcing access controls, and ensuring secure communication between applications and users.

Q2: How does Keycloak handle authentication and authorization for applications?

ANSWER: Keycloak handles authentication by supporting various industry-standard protocols such as OpenID Connect, OAuth 2.0, and SAML. It facilitates authentication through mechanisms like username/password authentication, social login (e.g., Google, Facebook), and multi-factor authentication. For authorization, Keycloak employs role-based access control (RBAC), allowing administrators to define roles and permissions and assign them to users or groups. Additionally, Keycloak enables fine-grained authorization policies based on attributes or context, ensuring precise control over access to resources.

Q3: What are the benefits of using Keycloak for identity and access management (IAM) compared to building custom authentication solutions?

ANSWER : Utilizing Keycloak for IAM offers several advantages over building custom authentication solutions. Firstly, Keycloak accelerates development by providing out-of-the-box support for authentication and authorization, reducing time-to-market for applications. Secondly, Keycloak enhances security through standardized protocols and best practices, reducing the risk of vulnerabilities inherent in custom implementations. Thirdly, Keycloak centralizes identity management, simplifying administration and enabling consistent user experiences across applications. Lastly, Keycloak is highly scalable and extensible, capable of supporting large user bases and evolving security requirements.

Q4: Can you explain the concept of realms in Keycloak and how they are used to organize users, groups, and applications?

ANSWER: Realms in Keycloak are logical partitions that isolate and organize different sets of users, groups, and applications. Each realm acts as an independent security domain with its own user database, authentication settings, and authorization policies. Administrators can create multiple realms to segment users and applications based on organizational units, departments, or functional requirements. Within a realm, administrators can manage users, define roles and permissions, configure authentication flows, and register applications, ensuring a clear separation of security concerns and administrative boundaries.

Q5: How does Keycloak support single sign-on (SSO) across multiple applications and domains?

ANSWER : Keycloak enables Single Sign-On (SSO) across multiple applications and domains through its session management capabilities. Once a user successfully authenticates with Keycloak in one application, Keycloak issues a session token. Subsequent requests to other applications within the same realm can be authenticated using this session token, eliminating the need for users to log in again. Keycloak manages the user's session state and provides mechanisms for securely propagating authentication tokens between applications, ensuring a seamless and secure SSO experience for users.

Q6: What authentication protocols does Keycloak support, and how do they contribute to interoperability with different types of applications?

ANSWER : Keycloak supports a variety of authentication protocols, including OpenID Connect, OAuth 2.0, and SAML. These protocols are widely adopted standards in the industry and are supported by a vast ecosystem of libraries and frameworks. By supporting these protocols, Keycloak enhances interoperability with different types of applications, including web, mobile, and single-page applications. Developers can leverage Keycloak's authentication capabilities without being tied to specific technologies or platforms, ensuring flexibility and compatibility across diverse application architectures.

Q7: Describe the process of integrating Keycloak with an existing application for user authentication and authorization.

ANSWER : Integrating Keycloak with an existing application involves several steps:

1. Configure Keycloak Realm: Create a realm in Keycloak and define the necessary settings such as authentication methods, client applications, roles, and users.
2. Set Up Client in Keycloak: Register the existing application as a client in the Keycloak realm, specifying the appropriate redirect URIs and authentication settings.
3. Integrate Keycloak Adapter: Install and configure the Keycloak adapter/library for the chosen platform or framework (e.g., Java, Node.js, .NET). This adapter handles interactions with Keycloak, including authentication and token validation.
4. Implement Authentication Flows: Implement the necessary authentication flows in the application to redirect users to Keycloak for authentication and handle the authentication callback to obtain tokens.
5. Authorize Access: Implement authorization logic in the application based on the user's roles and permissions obtained from Keycloak tokens.
6. Secure Communication: Ensure secure communication between the application and Keycloak by using HTTPS and appropriate security measures.

7. Test Integration: Thoroughly test the integration to ensure that user authentication and authorization work as expected across different scenarios.

Q8: What features does Keycloak provide for user management, such as user registration, password policies, and account linking?

ANSWER : Keycloak offers comprehensive user management features, including:

- User Registration: Keycloak provides customizable registration forms and email verification workflows, allowing users to sign up for accounts securely.
- Password Policies: Administrators can define password complexity rules, expiration periods, and password history requirements to enforce strong password policies.
- Account Linking: Keycloak supports linking user accounts from multiple identity providers (e.g., LDAP, Active Directory, social logins) to a single Keycloak account, enabling seamless authentication across different systems.
- User Profile Management: Users can manage their profile information, such as email address, display name, and profile picture, through Keycloak's user interface.

Q9: How does Keycloak handle role-based access control (RBAC) and fine-grained authorization policies for controlling access to resources?

ANSWER : Keycloak implements role-based access control (RBAC) by allowing administrators to define roles and assign them to users or groups. Users inherit permissions associated with their assigned roles, enabling coarse-grained access control. Additionally, Keycloak supports fine-grained authorization policies through its Authorization Services, which allow administrators to define complex access control rules based on attributes, context, or custom policies. These policies enable precise control over access to resources, including data fields, API endpoints, or application functionalities.

Q10: Can you explain how Keycloak can be deployed in a high-availability (HA) and scalable architecture for large-scale applications?

ANSWER : Keycloak can be deployed in a high-availability (HA) and scalable architecture by following best practices such as:

- Clustering: Deploy multiple Keycloak instances in a clustered configuration to distribute load and ensure redundancy. Use load balancers to evenly distribute incoming traffic across the cluster.
- Shared Databases: Configure Keycloak instances to use shared databases for session replication and data synchronization. This ensures consistency and reliability across the cluster.
- Distributed Caching: Utilize distributed caching solutions to improve performance and scalability by caching frequently accessed data such as user sessions, tokens, and authorization policies.
- Horizontal Scaling: Scale Keycloak horizontally by adding more instances or nodes as demand increases. Ensure that the architecture is designed to handle peak loads and accommodate future growth.
- Monitoring and Maintenance: Implement monitoring tools to monitor Keycloak instances, detect performance bottlenecks, and proactively address issues. Regularly update Keycloak and its dependencies to patch security vulnerabilities and ensure stability.

By adopting these strategies, organizations can deploy Keycloak in a resilient and scalable architecture capable