Project Name: Keycloak and RabbitMQ for Streaming-Driven IAM

Objective:

Develop and deploy a scalable, secure, multi-tenant identity and access management (IAM) solution using Keycloak, with RabbitMQ as the backbone for targeted streaming and asynchronous messaging. Integrate the system with multiple applications to handle real-time event streaming, automate deployment, and demonstrate robust resilience.

Project Structure

Part 1: High-Availability Keycloak Setup with RabbitMQ for Streaming

1. **Deploy Keycloak as a high-availability cluster**:

   - Use Kubernetes (K3s/K3d) with Helm charts or Kubernetes manifests.
   - Configure persistent storage and external database (e.g., PostgreSQL).

2. **Integrate RabbitMQ for Targeted Streaming**:

   - Deploy RabbitMQ as a high-availability cluster in Kubernetes.

- Configure RabbitMQ to use topic exchanges, enabling messages to be routed to specific consumers or services based on routing keys.
- Ensure that targeted streaming delivers events, such as login activity or user updates, to relevant systems or tenants only.
- Set up message exchanges using RabbitMQ’s topic exchange for real-time, targeted streaming.in **Secure the deployment**:

- Use HTTPS with certificates from Let's Encrypt.
- Enable Keycloak’s security features (e.g., brute-force detection).

Deliverables:

- High-availability Keycloak and RabbitMQ clusters deployed on Kubernetes.
- RabbitMQ configured for streaming events based on topic exchanges and routing keys.
- Secured endpoints for both Keycloak and RabbitMQ.

Part 2: Multi-Tenancy with Streaming Integration

1.

Configure Keycloak for multi-tenancy:

- Create multiple realms:
  - CorporateRealm for internal users.
  - PartnerRealm for external users.
  - CustomerRealm for end-users.

2. **RabbitMQ Integration for Streaming per Realm**:

- Configure RabbitMQ topics for each realm to stream user events and inter-realm notifications.
- Use targeted streaming to ensure events are routed only to the intended tenants or consumers, enhancing efficiency and security.
- Facilitate dynamic stream routing for real-time user activity.

3. **Customize login themes**:

- Develop unique themes for each realm with seamless integration of streaming events (e.g., login activity streams).

4. **Enable cross-realm identity brokering**:

- Use RabbitMQ to synchronize SSO events across realms.

Deliverables:

- Multi-realm Keycloak configuration enhanced by RabbitMQ streaming.
- Customized login themes with streaming-driven activity insights.
- Cross-realm identity brokering with event synchronization via RabbitMQ.

Part 3: Advanced Authentication and Streaming

1. **Set up advanced authentication flows**:

   - Implement MFA using OTP and WebAuthn (FIDO2).
   - Configure conditional authentication policies with streaming event triggers (e.g., notifying other systems of policy matches).

2. **Stream Notifications via RabbitMQ**:

   - Use RabbitMQ for event-driven notification services, such as streaming user login events or MFA triggers to subscribed systems.
   - Leverage targeted streaming to deliver specific notifications to relevant consumers or services, minimizing noise and optimizing performance.

3. **Real-Time Authorization Updates**:

   - Integrate RabbitMQ to dynamically stream authorization changes and policy updates to dependent systems.

Deliverables:

- Real-time event notifications for authentication flows via RabbitMQ.
- Authorization updates broadcast to systems subscribed to RabbitMQ streams.
- Integrated Keycloak-RabbitMQ authentication workflows.

Part 4: DevOps and Automation

1. **Automate Keycloak and RabbitMQ deployment**:

   - Use Ansible, Terraform, or Helm for provisioning infrastructure.
   - Automate RabbitMQ setup for streaming and dynamic queue configuration.

2. **Implement CI/CD pipelines**:

   - Automate deployment of Keycloak themes and RabbitMQ exchange configurations.
   - Include automated tests for real-time streaming and message reliability.

Deliverables:

- Automated provisioning and configuration scripts for Keycloak and RabbitMQ.

- CI/CD pipelines for dynamic deployment of streaming-driven configurations.
- RabbitMQ stream tests validated through CI/CD workflows.

Part 5: Monitoring, Logging, and Resilience

1. **Set up monitoring for streaming metrics**:
   - Use Prometheus and Grafana to monitor RabbitMQ streaming performance (e.g., message rates, queue lengths) and Keycloak metrics (e.g., session activity).
2. **Centralized Logging**:
   - Use the ELK Stack or Loki to aggregate logs from Keycloak and RabbitMQ, focusing on streaming-related events.
3. **Simulate streaming failure scenarios**:
   - Test the system’s resilience to message delivery delays, queue saturation, and node failures.

Deliverables:

- Real-time monitoring dashboards for RabbitMQ streaming and Keycloak activity.
- Centralized log analysis for troubleshooting streaming events.
- Resilience tests validated against simulated failures.

Bonus Part: Advanced Streaming and Federation

1. **Stream Federation Events**:
   - Configure RabbitMQ to stream federated login and identity events to external systems.
   - Ensure real-time updates to federated user activities.

2. **Advanced Streaming Features**:
   - Implement RabbitMQ's delayed message queues for scheduled event streaming.
   - Enhance security using encrypted message streams for sensitive data.

Deliverables:

- Federated user activity streams integrated with RabbitMQ.
- Advanced RabbitMQ streaming features tested and documented.

Submission and Evaluation

Submit the project in a GitHub repository with:

- Deployment scripts for Keycloak and RabbitMQ.
- Configuration files, streaming setups, and documentation.
- Monitoring dashboards and logs demonstrating real-time streaming.

Evaluation will include:

- Live demonstration of Keycloak and RabbitMQ streaming integrations.
- Handling streaming-related failure scenarios.
- Performance monitoring of streaming workloads.