PostgreSQL cluster as Keycloak backend

Overview

The comprehensive document delves into the architecture of our High Availability PostgreSQL setup and the strategic integration with Keycloak for seamless operation in dynamic environments.

Highlights

High Availability PostgreSQL Setup

System is intricately designed to ensure High Availability (HA) through a robust architecture. The deployment leverages PostgreSQL, with a special focus on High Availability and failover management facilitated by repmgr in a docker container. Single Primary and multiple standby(s) are encouraged in the proposed architecture.

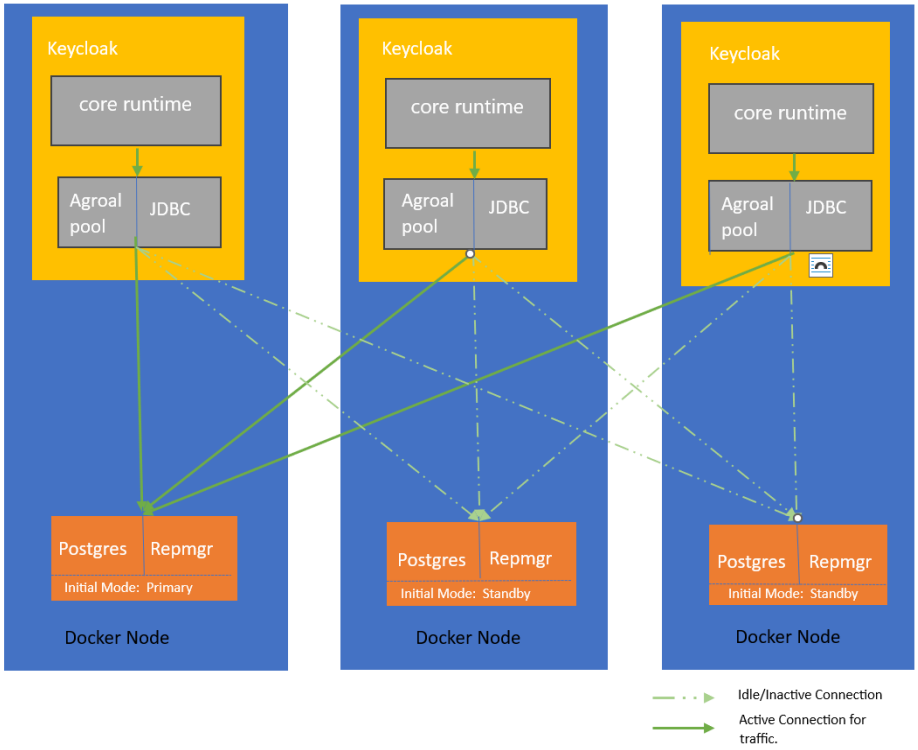
Keycloak Integration

This document sheds light on the integration of Keycloak, with our PostgreSQL setup. Particular attention is given to Keycloak's (Agroal Pool) intelligent detection and utilization of either primary or standby server database connection based on suitable scenario(s).

Optimization Strategies

The proposed usage of preferPrimary strategy within Keycloak (JDBC), ensures utilization of the primary database during routine DB operations. Whenever primary is down, standby server connection will be utilized by Keycloak. A validation query mechanism is introduced, ensuring automatic fallback from current standby to newly repmgr elected primary within minimal time and ensures read operations will be handled during the time of failover. Validation query executes a custom query on DB linked repmgr table [***repmgr_events***] which ensures fallback for desired node.

Overall Architecture



Achieving PostgreSQL High Availability

Repmgr Driven Failover and Fallback Sequencing

1. When a PG node is brought up for the first time, it will query all specified nodes in **REPMGR_PATNER_NODES**, since no one is up, this node will be assumed primary.
2. Upon new PG node coming up, **REPMGR_PATNER_NODES** will be queried, and since a primary is already available, the new node will be designated as slave and will pull the DB files from primary. [Offset of DB files is represented by LSN of WAL, so if any future restart happens for slave and if LSN is not in sync, database files will be updated automatically from Primary]
3. Slaves will do a health check every 5sec if primary is reachable. If it's not reachable, then 3 attempts will be made, so effectively 15 sec will be spent on this process.
   If a slave identifies Primary is no longer available. following is the leader [Primary] election algorithm,
   1. It will check if other slaves are still reachable, and it was following the same older Primary.

2. It then checks if LSN is reported same by each other slave. If LSN is same, then whoever has the lowest repmgr_node_id will be chosen as primary, else the slave with higher LSN will be chosen.
3. Above operations will get executed in parallel across each slave. Trust is also established at Postgres DB table ***repmgr_events***, ensuring election is smooth.
4. Potential primary candidate will initiate process of primary promotion and it will then intimate to another slave to follow them.

4. Now both slaves will be in sync.
5. If older Primary comes up later, it will be attached as standby to current primary and the database files will be reset in accordance with current primary.

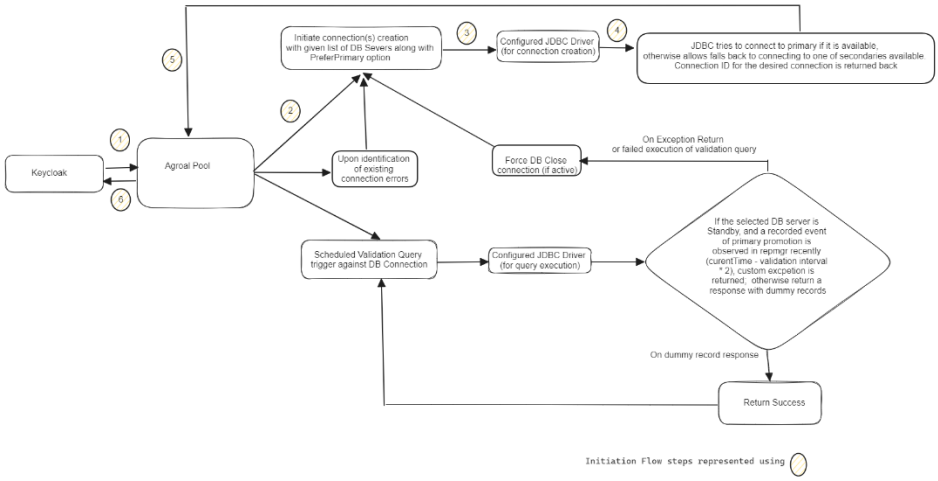
Note: During promotion to primary, if the candidate slave experienced any issues, then the other slaves who was set to get notified upon successful follow-up, won't get that event within 20sec [*degraded_monitoring_timeout*], this leads to restart of docker container, now the leader election happens, and Primary promotion will be ensured

Reliability Insights

- Effectively 20 seconds will be spent for complete failover if Primary is stopped or not reachable. If the PG port is active, but doesn't respond to heartbeat checks, failover will last 30 secs.
- After failover, if older Primary tries to connect with cluster, demotion to standby will be seamless. Fallback duration depends on size of DB files that needs to be restored from primary and network latency involved.
- Query being executed at exact same time of primary failure, won't be retried.

Achieving Keycloak High Availability Backend

Primary Detection and Fallback Sequencing



Achieving Keycloak High Availability Backend

Primary Detection and Fallback Sequencing

*Note***:** Validation query interval is customizable. Current value is 10sec. Based on pool size and KC's DB load, it can be decided. Custom pg-repmgr will have necessary support to facilitate the custom exception.

Reliability Insights

- Once Primary is down, for 30 secs [In accordance with repmgr-driven failover time and validation time] the Write operation(s) will be rejected i.e. until the successful fallback.
- Invalidation of connection using Validation query, will force close a connection, hence active queries and transactions will fail and won't be replayed.
- Read operation(s) will always succeed, except at the exact timestamp(s) of `Primary down` and `fallback to active primary`.