**RDS Proxy Setup with Aurora PostgreSQL**

**Introduction to RDS Proxy**

* RDS Proxy manages database connections, reducing overhead by pooling and multiplexing.
* Decouples applications from direct database interaction, enhancing scalability and availability.

**Benefits of RDS Proxy**

* **Scalability & Performance**: Efficient connection pooling and multiplexing reduce database overhead.
* **Availability & Resilience**: Up to 66% reduction in failover times due to seamless, transparent handling.
* **Security**: Centralized credential management with AWS Secrets Manager and IAM database authentication.

**Core Concepts**

* **Connection Pooling**: Reuse database connections to minimize overhead.
* **Multiplexing**: Reusing database connections across multiple client requests.
* **Connection Pinning**: Certain stateful actions (e.g., SET commands, temporary tables) pin connections and reduce efficiency.
* **Failover Management**: Proxy transparently manages failover, reducing downtime.

**Client-Side Best Practices (AWS JDBC Driver & HikariCP)**

**AWS JDBC Driver Configuration**

* **Use the AWS JDBC wrapper**:
  + URL Prefix: jdbc:aws-wrapper:postgresql://
  + Example:

pgsql

CopyEdit

jdbc:aws-wrapper:postgresql://proxy-endpoint:5432/mydb?sslmode=verify-full

* **SSL/TLS**:
  + Recommended sslmode: verify-full (most secure).
  + Provide AWS root CA certificate via sslrootcert.
* **IAM Authentication**:
  + Enable IAM plugin via JDBC URL (wrapperPlugins=iam).
  + Configure application roles and ensure IAM permissions (rds-db:connect).
* **Disable JDBC Cluster-Aware Failover**:
  + Set enableClusterAwareFailover=false as RDS Proxy manages failover.
* **JDBC Parameters to Prevent Pinning**:
  + assumeMinServerVersion=9.0
  + Set ApplicationName explicitly.

**HikariCP Configuration**

* **Basic Setup**:
  + Set JDBC URL using AWS JDBC wrapper format.
* **Connection Timeouts**:
  + connectionTimeout: 5000–30000 ms (5–30s recommended)
  + idleTimeout: Shorter than RDS Proxy’s IdleClientTimeout (e.g., 5–10 min)
  + maxLifetime: Less than RDS Proxy’s 24-hour limit; for IAM, set below 15 min (e.g., 10–14 min)
* **Pool Sizing**:
  + maximumPoolSize: Moderate, based on application concurrency (e.g., 10–50)
  + minimumIdle: Match or lower than maximumPoolSize based on usage patterns
* **Additional Settings**:
  + leakDetectionThreshold: 30000–60000 ms
  + initializationFailTimeout: Positive (fast-fail startup, e.g., 1000 ms)

**RDS Proxy Parameter Best Practices**

* **MaxConnectionsPercent**: Typically 70–90%, lower (20–30%) for large databases.
* **MaxIdleConnectionsPercent**: Typically 10–50% (default is 50%).
* **IdleClientTimeout**: Default 30 min; always greater than HikariCP’s idleTimeout.
* **ConnectionBorrowTimeout**: Default 120 seconds, monitor and adjust carefully.
* **RequireTransportLayerSecurity**: Always TRUE.
* **PostgreSQL-specific**:
  + Avoid SessionPinningFilters (unsupported).
  + Use InitQuery sparingly, avoiding session-state changes to prevent pinning.

**Common Pitfalls and Avoidance Strategies**

* **Connection Pinning**:
  + Avoid unnecessary SET commands, temporary tables, large queries (>16KB), and explicit SQL-level prepared statements.
  + Rely on Extended Query Protocol through the JDBC driver.
* **Timeout Misconfigurations**:
  + Align HikariCP idleTimeout and maxLifetime with RDS Proxy settings and IAM token lifespan.
* **Network & IAM Misconfigurations**:
  + Ensure correct VPC/Subnet/Security Group setup.
  + Verify IAM roles, secrets access, and permissions.
* **Resource Limitations**:
  + Monitor IP address availability in subnets.
  + Prevent hitting connection limits by proper sizing and pinning mitigation.

**Monitoring and Troubleshooting**

* **CloudWatch Metrics to Monitor**:
  + DatabaseConnectionsCurrentlySessionPinned (critical for multiplexing effectiveness)
  + ClientConnections, DatabaseConnections, DatabaseConnectionsBorrowLatency
* **Enhanced Logging**: Use temporarily for targeted debugging (pinning issues, auth errors).