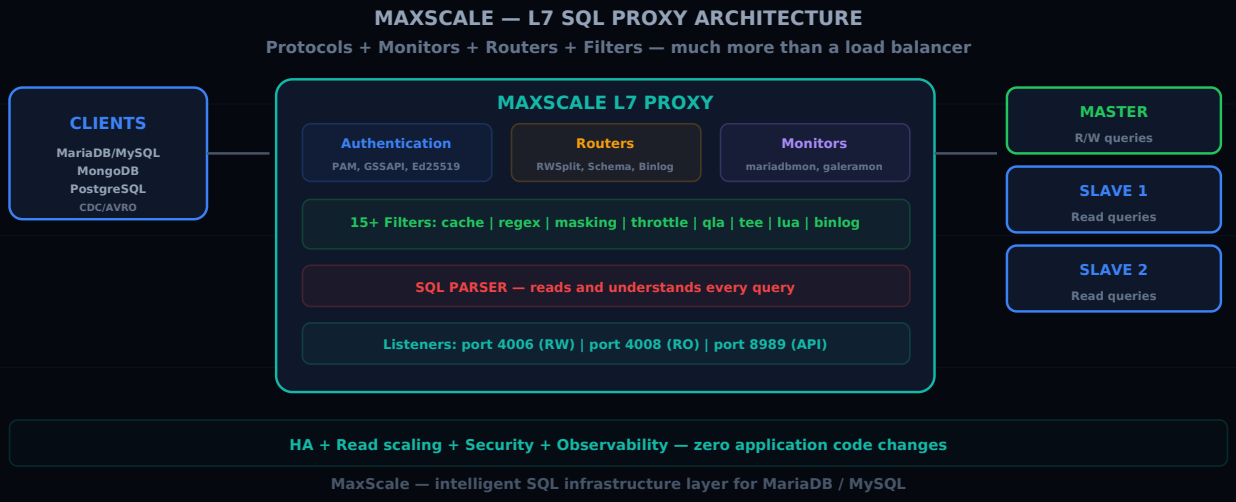
MaxScale: Much More Than a Reverse SQL Proxy (Part 1)

Sylvain ARBAUDIE · July 31, 2025

MAXSCALE MARIADB PROXY ARCHITECTURE



Not a Simple Proxy

When people talk about MaxScale, the reflex is to compare it to HAProxy or a classic load balancer. This is a fundamental error. MaxScale is not a Layer 4 (TCP) proxy. It is a **Layer 7 proxy** that understands the SQL protocol. It parses queries, analyzes their type, and makes intelligent routing decisions.

The difference is comparable to a mail carrier who delivers by address (L4) versus an assistant who reads your emails, sorts them by priority, filters spam, and only shows you what is relevant (L7). MaxScale reads SQL.

Authentication: Beyond Passwords

MaxScale supports a comprehensive range of authentication mechanisms:

- **MariaDBAuth**: native MariaDB / MySQL authentication with local credential caching
- **PAM**: integration with enterprise directories (LDAP, Active Directory) via Pluggable Authentication Modules

- **GSSAPI/Kerberos**: SSO authentication for Windows / Active Directory environments
- **Ed25519**: MariaDB's public key authentication, more secure than classic SHA hashing

Authentication is handled at the listener level. You can therefore have different authentication methods on different ports: PAM for internal applications, Ed25519 for administrative connections, MariaDBAuth for legacy compatibility.

Protocols: Not Just SQL

MaxScale is polyglot. It supports multiple input and output protocols:

MariaDBClient / MariaDBBackend

The native MariaDB / MySQL protocol. This is the primary use case: applications connect to MaxScale as if it were a standard database server.

CDC / AVRO

The Change Data Capture protocol streams binlog changes in AVRO format. It is the ideal tool for building real-time data pipelines, feeding data lakes, or synchronizing external systems.

NoSQL / MongoDB

MaxScale can expose a MongoDB-compatible interface. Applications that speak the MongoDB protocol can interact with a MariaDB database via MaxScale. It is a niche but powerful feature for migrations.

PostgreSQL (Experimental)

PostgreSQL protocol support on the input side is under development. The goal is to allow PostgreSQL applications to connect to MariaDB backends.

Monitors: Topology Intelligence

Monitors are MaxScale's eyes. They regularly query backend servers to automatically detect the topology and health state of the cluster.

mariadbmon

The primary monitor for MariaDB / MySQL replication topologies. It detects the master, slaves, relays, and replication state. It handles automatic failover: if the master goes down, a slave is promoted, and the other slaves are reconfigured to replicate from the new master.

galeramon

The dedicated monitor for Galera clusters. It detects cluster state (Primary, Non-Primary, Disconnected), node count, state UUID, and handles routing accordingly.

Routers: Routing Intelligence

Routers are MaxScale's brain. They decide where to send each query based on its type and the detected topology.

readwritesplit

The most commonly used router. Write queries (`INSERT` , `UPDATE` , `DELETE` , `CREATE` , etc.) go to the master. Read queries (`SELECT`) are distributed across slaves. Transactions are sent entirely to the master.

readconnroute

A simpler router that distributes connections (not queries) across available servers. Useful for read-intensive workloads that do not require query-level read/write separation.

schemarouter

Routes queries to the server hosting the requested schema. Ideal for per-database sharding: `client_europe` on server A, `client_asia` on server B.

binlogrouter

Turns MaxScale into a replication relay. MaxScale behaves as a slave of the master, and the real slaves connect to MaxScale rather than the master. This reduces load on the master and centralizes binlog distribution.

kafkarouter (CDC)

Sends binlog events to Apache Kafka. Each database modification is published as a Kafka message, allowing consumers to react in real time.

Filters: More Than 15 Modules

Filters intercept and transform the SQL stream between client and server. MaxScale offers more than 15:

- **qlafilter**: complete query logging (audit)
- **regexfilter**: query rewriting by regular expression
- **cache**: query cache with automatic invalidation
- **throttlefilter**: per-user query rate limiting
- **masking**: dynamic masking of sensitive data (emails, card numbers)
- **topfilter**: collection of slowest queries
- **commentfilter**: SQL comment injection for tracing
- **tee**: stream duplication to a second backend (shadow testing)
- **namedserverfilter**: routing to a specific server based on rules
- **hintfilter**: interpretation of SQL hints to force routing
- **luafilter**: Lua script execution for custom logic
- **binlogfilter**: binlog event filtering by schema or table

Deployment: Docker and Kubernetes Ready

MaxScale is available as an official Docker image:

```
docker run -d --name maxscale \
  -v /path/to/maxscale.cnf:/etc/maxscale.cnf \
  -p 4006:4006 -p 8989:8989 \
  mariadb/maxscale:latest
```

For Kubernetes, MaxScale deploys as a StatefulSet or Deployment depending on the use case. The REST API facilitates integration with health checks and readiness probes.

Business Value

Beyond technical features, MaxScale delivers concrete business value:

- **High availability**: automatic failover in seconds, transparent to the application
- **Read scalability**: add slaves without modifying application code
- **Security**: SQL filtering, data masking, centralized authentication
- **Observability**: logging, metrics, REST API for monitoring
- **Easier migrations**: multi-protocol support for technology transitions

MaxScale is not a simple reverse proxy. It is an intelligent infrastructure layer that sits between applications and MariaDB / MySQL databases, bringing resilience, security, and flexibility without modifying a single line of application code.

This article was originally published on Medium.