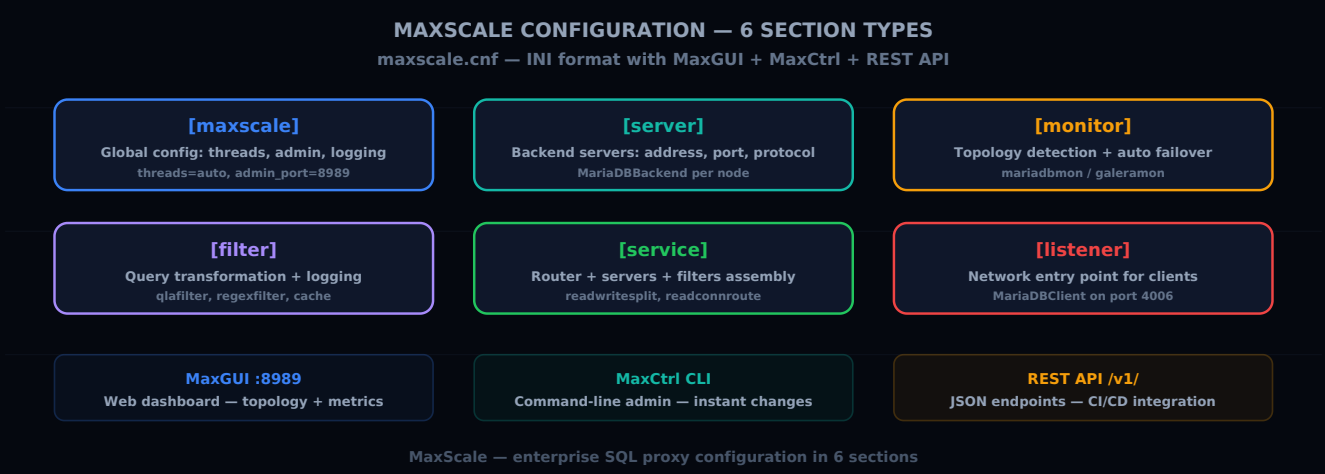
MaxScale: гораздо больше, чем reverse SQL proxy (часть 2)

Sylvain ARBAUDIE · September 1, 2025

MAXSCALE MARIADB PROXY CONFIGURATION



От теории к практике

В первой части мы рассмотрели архитектурные возможности MaxScale: его протоколы, маршрутизаторы, мониторы и фильтры. Эта вторая часть — практическое руководство: как установить, настроить и администрировать MaxScale в повседневной работе.

Установка

MaxScale доступен через официальный репозиторий MariaDB. На Debian/Ubuntu:

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup \
  | sudo bash -s -- --mariadb-maxscale-version=24.02

sudo apt-get install maxscale
```

Основной файл конфигурации находится в `/etc/maxscale.cnf`. Это файл в формате INI, организованный в чётко определённые секции.

Анатомия конфигурационного файла

Файл `maxscale.cnf` организован в шесть типов секций, каждая из которых идентифицируется заголовком в квадратных скобках.

Секция [maxscale] — Глобальная конфигурация

```
[maxscale]
threads        = auto
log_info       = false
admin_host     = 0.0.0.0
admin_port     = 8989
admin_secure_gui = true
```

- **threads=auto**: MaxScale автоматически определяет количество доступных ядер CPU.
- **admin_host/admin_port**: Активирует веб-интерфейс MaxGUI и REST API.
- **admin_secure_gui**: Включает HTTPS для интерфейса администрирования.

Секция [server] — Определение серверов

Каждый бэкенд-сервер MariaDB / MySQL объявляется отдельно:

```
[db-master]
type     = server
address  = 10.0.1.10
port     = 3306
protocol = MariaDBBackend

[db-slave1]
type     = server
address  = 10.0.1.11
port     = 3306
protocol = MariaDBBackend

[db-slave2]
type     = server
address  = 10.0.1.12
port     = 3306
protocol = MariaDBBackend
```

Каждая секция `[server]` носит логическое имя (здесь `db-master`, `db-slave1`, `db-slave2`), на которое ссылаются мониторы и сервисы.

Секция [monitor] — Мониторинг топологии

Монитор — это компонент, автоматически обнаруживающий топологию репликации и состояние каждого узла:

```
[replication-monitor]
type            = monitor
module          = mariadbmon
servers         = db-master, db-slave1, db-slave2
user            = maxscale_monitor
password        = encrypted_password_here
monitor_interval = 2000
auto_failover    = true
auto_rejoin      = true
```

- **mariadbmon**: Стандартный монитор для топологий master-slave MariaDB / MySQL.
- **auto_failover**: В случае потери мастера MaxScale автоматически повышает слейв.
- **auto_rejoin**: Бывший мастер, вернувшийся в строй, автоматически перенастраивается как слейв.

Для Galera вместо `mariadbmon` используется `galeramon`.

Секция [filter] — Трансформация запросов

Фильтры перехватывают и модифицируют SQL-поток:

```
[query-log]
type   = filter
module = qlafilter
filebase = /var/log/maxscale/queries
flush   = true

[regex-rewrite]
type    = filter
module  = regexfilter
match   = SELECT.*FROM\s+legacy_table
replace = SELECT * FROM new_table
```

`qlafilter` записывает все запросы (полезно для аудита). `regexfilter` позволяет переписывать запросы на лету, без модификации приложения.

Секция [service] — Сборка компонентов

Сервис связывает серверы, мониторы, фильтры и маршрутизатор:

```
[rw-service]
type     = service
router   = readwritesplit
servers  = db-master, db-slave1, db-slave2
user     = maxscale_service
password = encrypted_password_here
filters  = query-log | regex-rewrite

max_slave_connections    = 100%
max_slave_replication_lag = 5s
```

- **readwritesplit**: Записи идут на мастер, чтения распределяются по слейвам.
- **max_slave_replication_lag**: Слейв с задержкой более 5 секунд временно исключается из маршрутизации.
- **filters**: Фильтры объединяются через pipe `|`.

Секция [listener] — Сетевая точка входа

Listener предоставляет сервис на сетевом порту:

```
[rw-listener]
type     = listener
service  = rw-service
protocol = MariaDBClient
port     = 4006
address  = 0.0.0.0
```

Приложение подключается к MaxScale на порту 4006, как если бы это был стандартный сервер MariaDB / MySQL. MaxScale прозрачен с точки зрения протокола.

MaxGUI: веб-интерфейс

Начиная с MaxScale 2.5, веб-интерфейс MaxGUI предоставляет полную графическую картину инфраструктуры. Доступный по адресу `https://maxscale-host:8989`, он позволяет:

- Визуализировать топологию в реальном времени (мастер, слейвы, состояние соединений)
- Просматривать метрики производительности (запросов/секунду, латентность, активные соединения)
- Управлять серверами (drain, maintenance, добавление)
- Просматривать логи и оповещения

MaxGUI — инструмент мониторинга, а не конфигурации. Изменения конфигурации выполняются через файл `maxscale.cnf` или через REST API.

MaxCtrl: инструмент командной строки

MaxCtrl — официальный CLI MaxScale. Он взаимодействует с REST API и предлагает интуитивный синтаксис:

```
# Список серверов и их состояние
maxctrl list servers

# Перевести сервер в режим обслуживания
maxctrl set server db-slave1 maintenance

# Снять режим обслуживания
maxctrl clear server db-slave1 maintenance

# Список сервисов и их статистика
maxctrl list services

# Посмотреть топологию репликации
maxctrl show monitor replication-monitor

# Создать фильтр динамически
maxctrl create filter my-filter regexfilter \
  match="SELECT 1" replace="SELECT 2"
```

Преимущество MaxCtrl перед прямым редактированием файла — немедленное применение изменений без перезапуска.

REST API: автоматизация

REST API MaxScale — основа автоматизации. Он предоставляет все функции через HTTP/JSON-эндпоинты:

```
# Получить список серверов
curl -s -u admin:password https://maxscale:8989/v1/servers | jq

# Ручной failover
curl -X POST -u admin:password \
  https://maxscale:8989/v1/monitors/replication-monitor/failover

# Получить метрики сервиса
curl -s -u admin:password \
  https://maxscale:8989/v1/services/rw-service | jq '.data.attributes.statistics'
```

REST API позволяет интегрировать MaxScale в CI/CD-конвейеры, системы мониторинга (Prometheus, Grafana) и инструменты оркестрации (Ansible, Terraform).

Лучшие практики конфигурации

Вот несколько рекомендаций из практического опыта:

1. **Всегда используйте шифрованные пароли** в файле конфигурации. MaxScale предоставляет утилиту `maxkeys` для генерации ключей шифрования.
2. **Включайте автоматический failover** с `auto_failover=true`, но регулярно тестируйте его в пре-продакшн среде.
3. **Настраивайте потоки** согласно нагрузке. `threads=auto` — хорошее значение по умолчанию, но для очень больших объёмов (>50 000 запросов/секунду) может потребоваться ручная настройка.
4. **Мониторьте лаг репликации** через `max_slave_replication_lag`. Слишком большое значение отправляет чтения на отстающие слейвы. Слишком маленькое концентрирует

все чтения на мастере.

5. **Используйте фильтры экономно**. Каждый фильтр добавляет слой обработки. `regexfilter` в частности может значительно влиять на производительность при сложных регулярных выражениях.

Типичная полная конфигурация

Для обобщения, вот типичный файл конфигурации для топологии master + 2 slaves:

```
[maxscale]
threads = auto
admin_host = 0.0.0.0
admin_port = 8989

[db-master]
type = server
address = 10.0.1.10
port = 3306
protocol = MariaDBBackend

[db-slave1]
type = server
address = 10.0.1.11
port = 3306
protocol = MariaDBBackend

[db-slave2]
type = server
address = 10.0.1.12
port = 3306
protocol = MariaDBBackend

[replication-monitor]
type = monitor
module = mariadbmon
servers = db-master, db-slave1, db-slave2
user = maxscale_monitor
password = <encrypted>
monitor_interval = 2000
```

```
auto_failover = true
auto_rejoin = true

[rw-service]
type = service
router = readwritesplit
servers = db-master, db-slave1, db-slave2
user = maxscale_service
password = <encrypted>
max_slave_replication_lag = 5s

[rw-listener]
type = listener
service = rw-service
protocol = MariaDBClient
port = 4006
```

Заключение

MaxScale настраивается с помощью ясного и структурированного INI-файла. Шести типов секций достаточно для описания полной инфраструктуры: глобальная конфигурация, серверы, монитор, фильтры, сервис и listener.

Повседневное администрирование осуществляется тремя взаимодополняющими инструментами: MaxGUI для визуального мониторинга, MaxCtrl для операций командной строки и REST API для автоматизации. Вместе они формируют полную экосистему для управления SQL-прокси корпоративного уровня.

Эта статья была первоначально опубликована на Medium.