MariaDB

MARIADB PLATFORM FOR TRANSACTIONS: PRODUCTION DEPLOYMENT CHECKLIST AND GUIDE

WHITEPAPER

TABLE OF CONTENTS

MARIADB PLATFORM FOR TRANSACTIONS:

Production Deployment Overview

MariaDB Platform is the leading enterprise open source database for transactional, analytical or hybrid transactional/analytical processing at scale. And while it's lightweight, easy to install and runs on any kind of infrastructure, including cloud and containers, deploying a production database to meet the requirements of business-critical, mission-critical applications (high availability, disaster recovery, performance, scalability and security) requires proper planning, configuration and preparation.

The goal of this production deployment checklist and guide is to provide IT, operations and database management teams with the information necessary to ensure a smooth and successful deployment of MariaDB Platform for transaction processing (i.e., OLTP) in staging and production environments, where database requirements are much higher than in development and testing environments. It assumes MariaDB Platform has been deployed in development and test environments, and requirements from development teams (or existing production data) will be used to help with capacity planning and hardware sizing.

The process is comprised of five stages:

- Planning
- Installation
- Configuration
- Preparation
- Testing

The planning, preparation and testing stages are critical to ensuring proper configuration and meeting business and technical requirements, short term *and* long term. While it is possible to change the configuration of production databases (to better meet enterprise requirements), proper planning and preparation not only reduces the need for these changes, but makes it easier to meet future application requirements (e.g., scaling out to support consistent, if not exponential, growth). And testing, of course, ensures that everything came together as expected.

The production deployment checklist and guide includes installation and configuration of not only the database itself, but also of the advanced database proxy – a critical component when it comes to meeting high availability and security requirements in production environments. It is one of many features, including workload-optimized storage, that must be enabled and configured to meet the enterprise requirements of business-critical, mission-critical applications in production environments.

The following checklist will help guide you through each stage of deployment, including links to documentation where it may be helpful. However, you don't have to do it all yourself. MariaDB can provide expert resources, including remote DBAs, enterprise architects and migration managers to help organizations of all sizes plan and execute a production deployment – whether it's to support legacy infrastructure and application modernization, the launch of new applications and services or the migration from a proprietary database like Oracle Database, Microsoft SQL Server or IBM Db2.

MARIADB PLATFORM FOR TRANSACTIONS:

Production Deployment Checklist

Planning	2-4 weeks, 5 months out	
		Complete
Table workloads		☐
Table storage		☐
Table security		☐
Query analysis		☐

Installation	0-1 week, 4 months out	
	Documentation	Complete
MariaDB Server: primary	https://goo.gl/ytS1qM	☐
MariaDB Server: secondaries (2+)		☐
MariaDB Server: shards (optional, 2+)		☐
MariaDB MaxScale: primary	https://goo.gl/fkmhGy	☐
MariaDB MaxScale: secondary		☐

Configuration	6-8 weeks, 4 months out	
HIGH AVAILABILITY		**WEEK 1**
	Documentation	**Complete**
MariaDB Server: replication – or –	https://goo.gl/22WwNs	☐
MariaDB Server: clustering	https://goo.gl/W6pqi8	☐
MariaDB MaxScale: Keepalived	https://goo.gl/ShKULw	☐
MariaDB MaxScale: automatic failover	https://goo.gl/HCr9Ar	☐

Configuration	6-8 weeks, 4 months out	
SCALABILITY		**WEEK 1**
	Documentation	**Complete**
MariaDB Server: sharding (optional)	https://goo.gl/JnaRRJ	☐
MariaDB MaxScale: read/write splitting	https://goo.gl/LgB5sK	☐

Configuration	6-8 weeks, 4 months out	
DISASTER RECOVERY		**WEEK 1**
	Documentation	**Complete**
MariaDB Server: point-in-time rollback	https://goo.gl/jkM3wv	☐

Configuration	6-8 weeks, 4 months out	
SECURITY		**WEEKS 1-2**
	Documentation	**Complete**
MariaDB Server: authentication (PAM)	https://goo.gl/WhHmu5	☐
MariaDB Server: authorization (roles)	https://goo.gl/4NW4Ah	☐
MariaDB Server: auditing	https://goo.gl/aieXAq	☐
MariaDB Server: transparent data encryption	https://goo.gl/DjNG3Y	☐
MariaDB Server: network data encryption	https://goo.gl/X7NEfo	☐
MariaDB MaxScale: authentication (PAM)	https://goo.gl/bE9MeK	☐
MariaDB MaxScale: database firewall	https://goo.gl/v9XMkc	☐
MariaDB MaxScale: data masking	https://goo.gl/FrEuxi	☐
MariaDB MaxScale: result limiting	https://goo.gl/TZZDby	☐

Configuration	6-8 weeks, 4 months out	
PERFORMANCE TUNING AND OPTIMIZATION		**WEEKS 2-3**
	Documentation	**Complete**
MariaDB Server: group commit	https://goo.gl/rmnCwJ	☐
MariaDB Server: parallel replication	https://goo.gl/ZjPcoj	☐
MariaDB Server: binary log compression	https://goo.gl/oRy3F2	☐
MariaDB Server: thread pool		☐
MariaDB Server: buffer pool		☐
MariaDB MaxScale: query result cache	https://goo.gl/J2Mhr5	☐

Preparation	1-2 weeks, 2 months out	
	Documentation	**Complete**
Table storage	https://goo.gl/Ht6C2m	☐
Table partitioning	https://goo.gl/oz9GWT	☐
Table compression	https://goo.gl/QD1xKc	☐
Column compression	https://goo.gl/XHHyw6	☐

Testing	2-4 weeks, 2 months out	
	Documentation	**Complete**
Query performance		☐
Replication performance		☐
Backup and restore	https://goo.gl/g9A4Sy	☐
Point-in-time rollback	https://goo.gl/jkM3wv	☐

MARIADB PLATFORM FOR TRANSACTIONS:

Production Deployment Guide

Planning

It's important to identify the workload and storage requirements of individual tables before creating the schema because MariaDB Server leverages per-table, workload-optimized storage engines.

- **InnoDB** is a general-purpose storage engine for mixed read/write workloads. If read scalability is required, replication can be used.
- **MyRock** is a space- and write-optimized storage engine created by Facebook. MyRocks has better compression and write performance than InnoDB, but less read performance. It is recommended for write-intensive workloads or if especially efficient storage is required – for example, to run on smaller, less expensive solid state drives.
- **Spider** is a storage engine for partitioning and querying data stored on separate database instances (i.e., storage nodes).

If write scalability is required, we recommend MyRocks on the storage nodes. If storage scalability is required, we recommend InnoDB on the storage nodes for mixed read/write workloads, and MyRocks for write-intensive workloads.

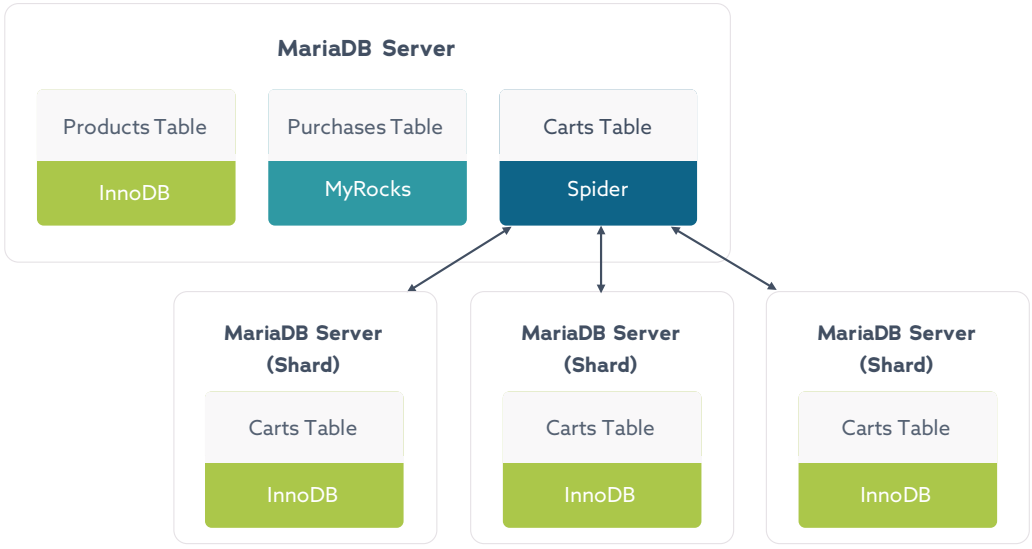


Diagram 1: MariaDB Server tables with different storage engines

Tip

In a replicated environment, the same table can be created with different storage engines on different database instances. For example, a table might be created with MyRocks on the primary for high write performance and InnoDB on the secondary for high read performance.

In addition, it's important to identify tables and columns with personally identifiable and/or sensitive information (PII/SPI). These tables should be noted for encryption and the columns should be noted for data masking. If columns contain medium to large text/binary data and the table will use InnoDB, these columns should be noted for column compression.

Next, we recommend a query analysis to identify how the data will be queried – when, how frequently and by whom. This information will be necessary to properly configure the database firewall and protect the data from unauthorized access.

Finally, high availability and scalability strategies must be selected prior to installation and configuration, and based on workload requirements (e.g., storage); application requirements (e.g., consistency); and service level agreements for availability, durability and performance. The high availability strategy will determine whether replication with automatic failover or clustering is required, while the scalability strategy will determine whether sharding is required.

Tip

The white paper *High availability with MariaDB Platform: The definitive guide* provides detailed information on high availability considerations and strategies.

White paper download: https://goo.gl/9gd1Qi

Configuration

High Availability

MariaDB Platform provides high availability via standard replication with automatic failover or clustering. However, while failover is not an issue with clustering, it is limited to InnoDB. And because replication is synchronous, write performance depends on the number of nodes (i.e., database instances in the cluster) – the more nodes there are, the more impact there is on write performance. We recommend clustering for mixed read/write workloads that require the highest availability with low to moderate read scalability. Standard replication with automatic failover is best for mixed read/write and write-intensive workloads requiring high availability with high write performance and/or the highest read scalability: semi-synchronous for durability, asynchronous for write performance.

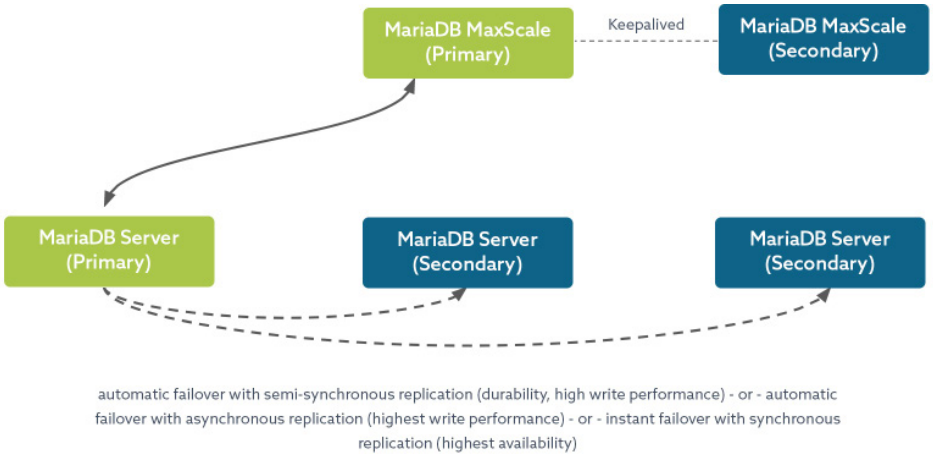


Diagram 2: MariaDB Platform deployed in a high availability topology

Note

If deploying to cloud infrastructure with its own load balancer, and where only one database proxy instance can be active at a time, Corosync and Pacemaker can be used instead of Keepalived. (Refer to Diagram 2.)

MariaDB MaxScale high availability documentation: https://goo.gl/48WGc4

Note

If replication is chosen for high availability and one or more tables use the MyRocks storage engine, the binary log must be set to the row-based format (not mixed or statement based).

Binlog formats documentation: https://goo.gl/X6tSsF

Tip

If consistent reads are required with replication, the Consistent Critical Read filter can be enabled in the database proxy, and with clustering, read queries can be delayed until all replicated transactions have been applied.

Documentation: https://goo.gl/zJorGS

Resources

High availability guide: https://goo.gl/6x2igh
High availability presentation: https://goo.gl/QP8Fdc
High availability documentation: https://goo.gl/QkJDT6

Scalability

MariaDB Platform scales reads, writes and storage – up and out. If moderate read scaling is required, both standard replication and clustering can provide high availability and read scalability. MariaDB MaxScale should be configured for read/write splitting, routing writes to the primary and load-balancing reads across one or more secondaries (if not all). If clustering is used, we recommend read/write splitting because it prevents conflicts when the same data is written to different nodes at the same time.

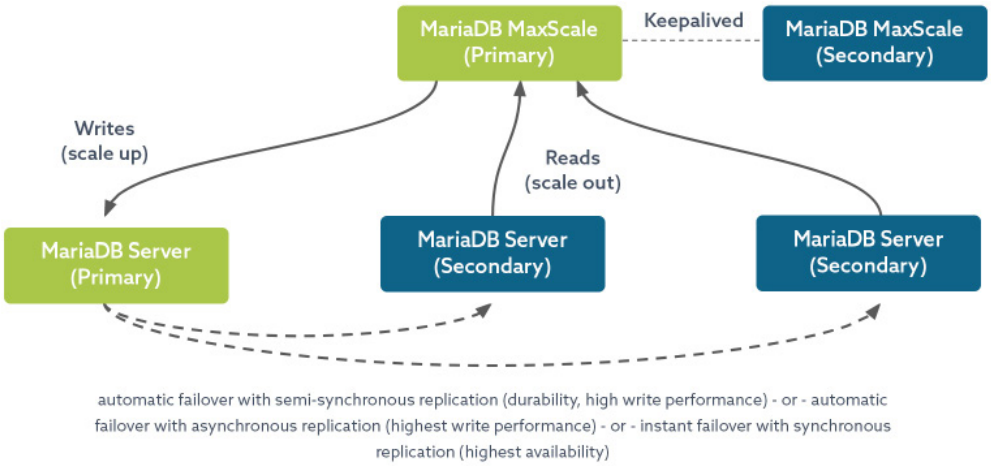


Diagram 3: MariaDB Platform in a high availability topology with read scaling

If high read scaling is required, an instance of MariaDB MaxScale can be deployed and configured as a binlog server (i.e., replication server) between the primary and the secondaries to offload replication from the primary. The secondaries will replicate from the binlog server instead of from the primary database, so its performance will not be impacted by replication from many secondaries.

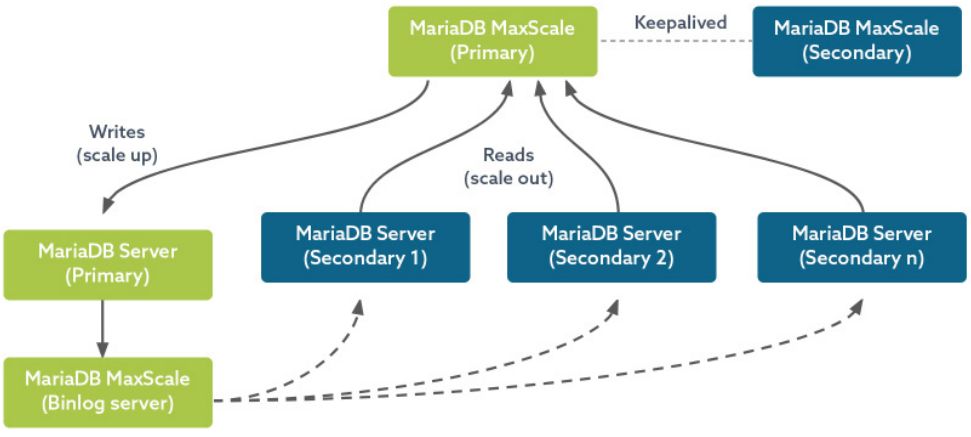


Diagram 4: MariaDB Platform in an extreme read-scaling topology

If write and/or storage scaling is required, tables can be created with the Spider storage engine to leverage transparent sharding across multiple databases. There are two database roles in a Spider topology: the Spider nodes (querying and routing) and the data nodes (querying and storage). The Spider nodes use standard replication or clustering to provide high availability. However, the data nodes use two-phase commit, writing to two or more databases within a single transaction, to provide high availability.

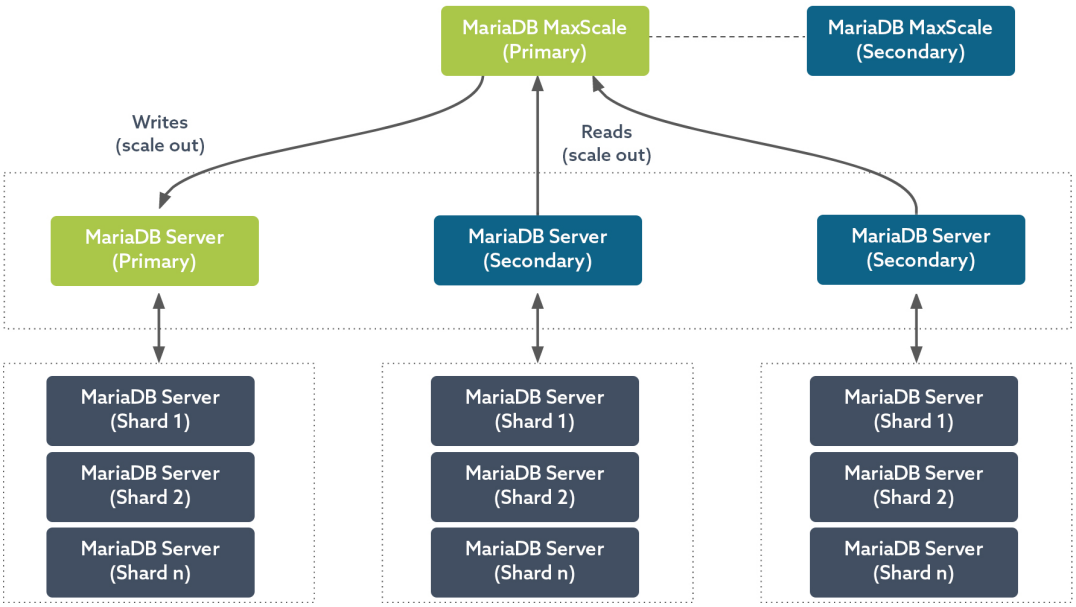


Diagram 5: MariaDB Platform in an high availability topology with scalable storage and writes

Tip

The data nodes can use InnoDB or MyRocks. InnoDB is recommended for mixed read/write workloads. MyRocks is recommended for write-intensive workloads.

Tip

Spider, combined with JSON documents, is an effective alternative to NoSQL databases. Examples include customer profiles and shopping carts stored as JSON documents in InnoDB tables (i.e., scalable mixed read/write workloads), and clickstream events stored as JSON documents in MyRocks tables (i.e., scalable write-intensive workloads).

Resources

Spider presentation: https://goo.gl/qVzJQX
Spider documentation: https://goo.gl/uJi2aK
Binlog server documentation: https://goo.gl/mWpSRk
Read/write splitting documentation: https://goo.gl/n178Vi

Disaster Recovery

MariaDB Platform provides disaster recovery though backup and restore with MariaDB Backup as well as point-in-time rollback with MariaDB Flashback. MariaDB Backup supports both full and incremental backups on InnoDB tables, and full backups on MyRocks tables. MariaDB Backup supports compressed and/or encrypted tables, and can compress and/or encrypt backups.

MariaDB Flashback performs a point-in-time rollback à la Oracle Flashback. It rolls back (or unwinds) a sequence of recent transactions to a previous point in time. In many cases, it is faster to roll back recent transactions than to perform a full restore.

Note

The binary log must be set to the full row image format in order to use MariaDB Flashback.

Binlog documentation: https://goo.gl/oWZgjQ
MariaDB Backup documentation: https://mariadb.com/kb/en/library/mariabackup-overview/
MariaDB Flashback documentation: https://mariadb.com/kb/en/library/flashback/

Note

MariaDB Flashback does not support DDL statements (e.g., DROP TABLE). If schema changes must be recovered, MariaDB Backup is required.

Security

MariaDB Platform includes a number of advanced security features for protecting both the database (e.g., from denial of service attacks) and the data itself (e.g., from data breaches). In a properly secured deployment, all of these features will be enabled and configured:

Resources

PAM authentication: https://goo.gl/AaXZwL
Roles: https://goo.gl/KL8Hbu
Auditing: https://goo.gl/XZE4hr
Transparent data encryption (TDE): https://goo.gl/SB7dm4
Network data encryption (i.e., TLS): https://goo.gl/yEF31y
Database firewall: https://goo.gl/BbbiXv
Data masking: https://goo.gl/BbbiXv
Result limiting: https://goo.gl/YFVso2

The TDE plugin supports external key providers. MariaDB Server includes plugins for the AWS Key Management Service (KMS) and eperi Gateway for Databases. We recommend using one of these key providers for maximum security.

The database firewall is one of the most important features when it comes to protecting your data and preventing data breaches. It can be configured to allow or block queries based on type (e.g., DDL vs. DML), syntax (e.g., missing WHERE clause), table/column, frequency, role and time. The database firewall rules should be created based on the query analysis performed in the planning phase.

In order to prevent data breaches from exposing PII/SPI, the data masking filter must be enabled – and configured with rules created based on the schema analysis performed in the planning phase.

Finally, result limiting should be enabled and configured to protect the database from denial of services attacks (malicious or accidental). If a query tries to return too many rows (or too much data), slowing down the database and limited availability, the results will be empty.

Note

The database firewall must be used with data masking to fully protect personal information. The data masking rules are based on column names. If a query calls a function on a protected column, the results will not be masked. Thus, database firewall rules must be created to block queries with functions on protected columns.

Resources

Encryption key management documentation: https://goo.gl/6xd1yj

Performance Tuning and Optimization

MariaDB Platform includes a number of performance options and features. If standard replication is used for high availability and/or read scalability, parallel replication and group commit can improve replication performance and reduce replication lag. By default, MariaDB Server calls fsync() to flush the binary log writes of a single transaction, and secondaries apply the binary log writes of a single transaction at a time (i.e., sequentially). However, when group commit is enabled, MariaDB Server will flush the binary log writes of multiple transactions with a single fsync() for better performance, and when parallel replication is enabled and configured too, secondaries will apply the binary log writes of multiple transactions at a time (i.e., in parallel).

Further, disk and network IO can be reduced by enabling binary log compression.

MariaDB Server implements a thread pool for connections to improve the performance of concurrent, short-lived queries. However, MariaDB Server can be configured to expose a separate port with a dedicated thread for superusers, ensuring database administrators continue to have access when every available thread in the thread pool is executing a query.

If there will be tables using InnoDB, setting the buffer pool size should be the first optimization made to increase performance. MariaDB recommends setting the buffer pool size to 80% of the available memory. If necessary, the buffer pool size can be increased or decreased based on the performance tuning results captured during the testing phase.

MariaDB MaxScale can further improve performance with query result caching. When query result caching is enabled, query results for all SELECT statements will be cached in MariaDB MaxScale by default (for maximum performance). However, query result caching can be configured to cache the results of specific queries rather than all queries.

Tip

If query result caching must be configured to cache the results of specific queries (e.g., for security reasons), rules based on regular expressions are faster than rules requiring query parsing, and are recommended for better performance.

Resources

Group commit documentation: https://goo.gl/4NhZvP
Parallel replication documentation: https://goo.gl/tZEvR1
Binary log compression documentation: https://goo.gl/Z1h8J6
Thread pool documentation: https://goo.gl/SGJGKw
InnoDB buffer pool documentation: https://goo.gl/sCaHDP
Query result caching documentation: https://goo.gl/NRnJYH

Preparation

It's important to identify the information gathered in the planning phase before creating the schema. In particular:

- Use the correct storage engine when creating a table (ENGINE=InnoDB|RocksDB|Spider)
- Use encryption for tables with personal information (ENCRYPTED=YES)
- If using InnoDB, use compressed columns (COMPRESSED) if there are few text/binary fields
- If using InnoDB, use table page compression (PAGE_COMPRESSION=1) if there are many pages
- If using MyRocks, the data is compressed by default

After the schema is created, the fastest way to load data is with the LOAD DATA INFILE statement.

Tip

If data is being loaded into a MyRocks table, bulk load can be enabled for better performance. If the data is sorted by primary key, drop all of the secondary indexes, load the data and then create the secondary indexes. MyRocks will use Fast Secondary Index Creation for better performance. If the data is not sorted, bulk load must be disabled and unsorted bulk load enabled.

Data loading documentation: https://goo.gl/a15ZCJ

Resources

Compressed columns documentation: https://goo.gl/eDfEPQ

Page compression documentation: https://goo.gl/2JtUff

Testing

It's important to test performance before going live. It not only helps identify potential performance issues; it also helps with resource optimization during the performance testing process, where you can experiment with different settings and options:

- InnoDB buffer pool: increase or decrease
- InnoDB redo log size: increase or decrease
- Number of connections: increase or decrease
- Connection wait time: increase or decrease
  - Check for idle connections
- Thread cache size: increase or decrease
- Temporary table size: increase as needed
- Join buffer size: increase as needed
- Sort buffer size: decrease if possible
- Maximum packet size: increase as needed

During performance testing, important performance and system metrics should be monitored, including CPU, memory, disk and network utilization.

Tip

We recommend enabling slow query monitoring and logging to identify slow queries and improve query performance.

Slow queries parameter documentation: https://mariadb.com/kb/en/library/server-status-variables/#slow_queries
Slow query log documentation: https://mariadb.com/kb/en/library/slow-query-log-overview/

Finally, before going live, it's important to test both automatic failover and disaster recovery, including full backup and restore with MariaDB Backup and point-in-time rollback with MariaDB Flashback.

Resources

Performance tuning presentation: https://goo.gl/A7xikm

Performance tuning documentation: https://goo.gl/niQQvn

MariaDB Backup documentation: https://goo.gl/Xc9MxS

MariaDB Flashback documentation: https://goo.gl/XZxKUA

MARIADB PLATFORM FOR TRANSACTIONS:

Production Deployment Services

To help ensure that MariaDB's customers succeed, we can provide additional expertise and staffing resources to guarantee a smooth and successful production deployment. Our remote DBAs, enterprise architects and migration managers can help with everything from initial planning and configuration to long-term migration and ongoing maintenance.

Remote DBAs

MariaDB has a dedicated team of expert DBAs on staff to augment your database management teams as needed, full-time or on demand. When it comes to production deployments (before or after), a remote DBA can perform an initial assessment of the database and formulate replication configuration and disaster recovery plans before turning to query optimization and performance tuning. And on a quarterly basis, they can perform ongoing security and performance audits.

> *"MariaDB's Remote DBA is absolutely amazing. It's an expert resource on tap – easy to contact, helpful and very responsive to where we are using it. Every dollar we invest translates into great value in helping us to be more agile as we strive to meet our business objectives."*
>
> ***– Steve Sharpe, System Architect, Teleplan***

Enterprise Architects

Deploying MariaDB Platform to cloud/container infrastructure or to support a microservices architecture? MariaDB maintains a team of enterprise architects dedicated to helping customers meet modern business requirements by transforming database and application infrastructure – whether it's breaking down legacy monolithic applications, building a hybrid cloud strategy or reviewing application and database infrastructure to create a long-term roadmap based on business and technology goals.

Migration Managers

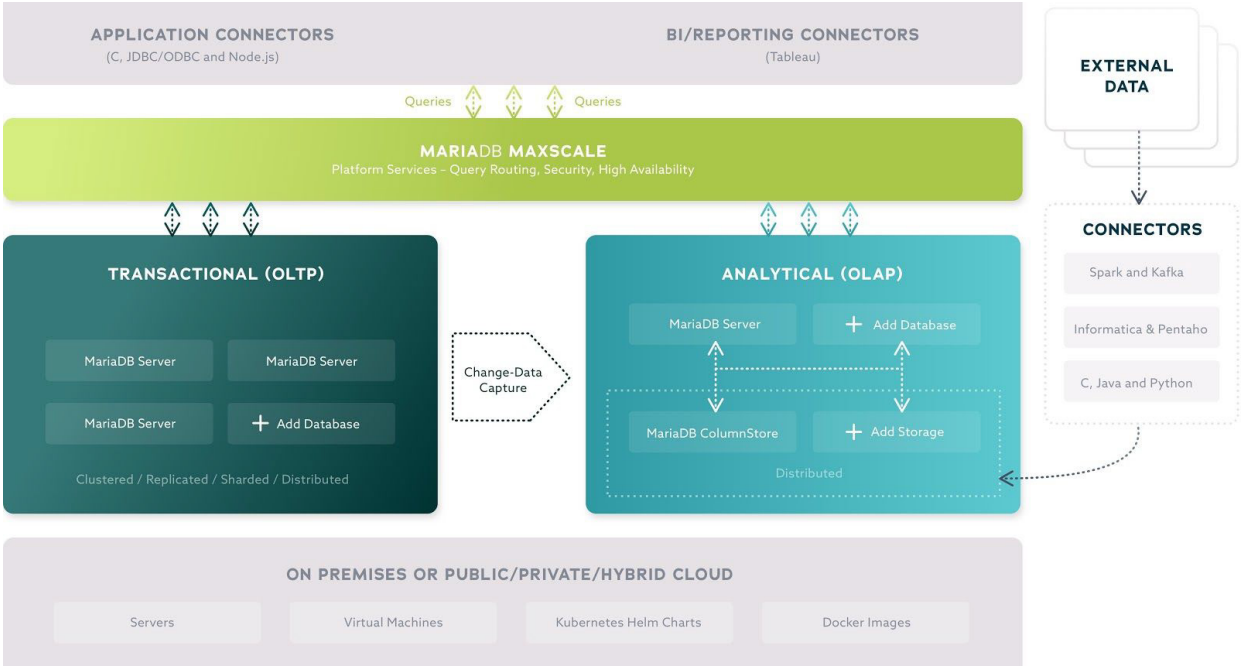
When migrating from a proprietary database – including Oracle Database, Microsoft SQL Server and IBM Db2 – MariaDB's Red Rover Migration Practice provides the expertise and tools you need to ensure a successful migration, from assessment to data and code migration to switchover. Migration Practice managers and architects can create migration project plans, implement and manage them, perform skills-gap and -analysis training and help with business continuity planning and preparation.

It's never too early to engage MariaDB expert resources when migrating from a proprietary database and/or deploying a production database configured to meet the requirements of business-critical, mission-critical applications.

ABOUT MARIADB PLATFORM

Transactions and Analytics, UNITED

MariaDB Platform is an enterprise open source database for transactional, analytical or hybrid transactional/analytical processing at scale. By preserving historical data and optimizing for real-time analytics while continuing to process transactions, MariaDB Platform provides businesses with the means to create competitive advantages and monetize data – everything from providing data-driven customers with actionable insight to empowering them with self-service analytics.



MariaDB Server

MariaDB Server is the foundation of the MariaDB Platform. It is the only open source database with the same enterprise features found in proprietary databases, including Oracle Database compatibility (e.g., PL/SQL compatibility), temporal tables, sharding, point-in-time rollback and transparent data encryption.

MariaDB ColumnStore

MariaDB ColumnStore extends MariaDB Server with distributed, columnar storage and massively parallel processing for ad hoc, interactive analytics on hundreds of billions of rows via standard SQL – with no need to create and maintain indexes, and with 10% of the disk space using high compression.

MariaDB MaxScale

MariaDB MaxScale provides MariaDB Platform with a set of services for modern applications, including transparent query routing and change-data-capture for hybrid transactional/analytical workloads, high availability (e.g., automatic failover) and advanced security (e.g., data masking).