Installation & Lifecycle

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

SERVER LIFECYCLE

Install Paths → **Configure** Scope → **Operate** Diagnose → **Upgrade** Recover

01 Installation, configuration and lifecycle

Install and initialize

- **basedir** identifies the installation root; **datadir** holds instance data. RPM installations on Oracle Linux commonly use /var/lib/mysql; tarball installations require explicit path and ownership planning.
- **mysqld --initialize** creates the data directory and a temporary, expired root password, reported to the error output or configured error log. **--initialize-insecure** creates root without a password.
- **RPM packages** split server, client and supporting functionality. **MySQL Installer** manages Windows product installation and upgrades; its workflow is not a cross-platform Linux installer.
- **Multiple instances** need separate data directories, socket paths, PID files and logs. Separate TCP endpoints by port or bound IP address; use distinct identities for replicated instances.

Option files and variable scope

- **mysqld --verbose --help** reports option-file locations and search order. Use **[mysqld]** for server options and **[client]** or program-specific groups for client options.
- **SET SESSION** changes one connection. **SET GLOBAL** changes runtime global state; for variables with session scope, existing sessions usually retain their values while new sessions inherit the new default.
- **SET PERSIST max_connections=200** changes runtime state and saves the setting in **mysqld-auto.cnf**. **SET PERSIST_ONLY** saves a value for startup without applying it immediately.
- **mysqld-auto.cnf** is server-managed JSON in datadir, processed after ordinary option files. **RESET PERSIST** removes persisted entries. Editing my.cnf or my.ini ordinarily takes effect at restart.

Connections and server lifecycle

- **TCP** supports remote connections across platforms. **SOCKET** uses a local Unix socket; Windows can use local **PIPE** or **MEMORY** when the corresponding server transport is enabled.
- **mysql -h host -P port -u user -p database** selects endpoint, account and default schema. On Unix, localhost normally selects a socket; use **--protocol=TCP** to force TCP.
- **mysqladmin shutdown**, SQL **SHUTDOWN**, a service-manager stop and **SIGTERM** request orderly shutdown. **SIGKILL** bypasses cleanup and requires crash recovery on restart.
- For **startup failures**, inspect the error log, effective datadir, ownership, missing files, incompatible configuration and competing processes. Never initialize over an existing data directory to repair it.

Installation & Lifecycle / Accounts & Security

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

SECURITY LAYERS

- **Host** — Permissions
- **Transport** — TLS
- **Account** — Identity
- **SQL** — Privileges

Upgrade readiness and rollback

- **MySQL Shell upgrade checker** reports compatibility problems before upgrade; it does not automatically fix them. Test application drivers, authentication support, SQL behavior and restore procedures.
- **Physical upgrade** reuses existing files with supported newer binaries. **Logical upgrade** exports and reloads data, rebuilding structures and often reclaiming unused space at greater time cost.
- Later **MySQL 8.0** releases perform dictionary and system-table upgrade tasks in the server. Upgrade controls such as **--upgrade=FORCE** do not replace compatibility checks or backups.
- A **5.7-to-8.0 rollback** requires a tested recovery plan, normally restoring the pre-upgrade 5.7 backup. Switching old binaries onto an upgraded 8.0 data dictionary is unsupported.

02 Accounts, privileges and security

Account identity and authentication

- **user@host** identifies an account; host matches the connecting client. **USER()** shows the connection identity, while **CURRENT_USER()** shows the account used for privilege checking, including proxy contexts.
- **caching_sha2_password** is the MySQL 8.0 default authentication plugin. Older clients may lack support; update drivers before considering a compatible per-account plugin.
- **auth_socket** authenticates local Unix-socket clients using OS identity. Enterprise **PAM** and simple **LDAP** authentication can require the cleartext client plugin; protect password exchange with TLS.
- **ALTER USER ... IDENTIFIED BY** and **SET PASSWORD** change passwords. An expired password restricts normal operations until reset; expiration and **ACCOUNT LOCK** have different effects.

Least privilege and account control

- **GRANT** adds privileges; **REVOKE** removes grants at their assigned scope. Removing a database grant does not remove access inherited through a global grant or active role.
- **SELECT** covers read columns; **UPDATE** covers changed columns. An UPDATE may also need SELECT on columns used in expressions or predicates. Grant only required tables, columns and operations.
- **CREATE TEMPORARY TABLES** is granted at database scope, for example **GRANT CREATE TEMPORARY TABLES ON sales.* TO jane**. It is not a table-level privilege.
- **ALTER USER ... ACCOUNT LOCK** prevents new logins but leaves existing sessions active. Investigate and terminate compromised sessions separately; revoking data privileges alone does not disable authentication.
- Use **CREATE USER**, **ALTER USER**, **GRANT** and **REVOKE** for account administration; they do not require FLUSH PRIVILEGES. Table and column grant changes affect subsequent requests in existing sessions.

Accounts & Security

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

SECURITY LAYERS

- **Host** — Permissions
- **Transport** — TLS
- **Account** — Identity
- **SQL** — Privileges

Roles and activation

- **CREATE ROLE** creates a named privilege collection implemented as an account locked by default. Grant object privileges to the role, then grant the role to users or other roles.
- **Granted** and **active** roles differ. **SET ROLE** activates granted roles for a session; **SET DEFAULT ROLE** configures login defaults. **activate_all_roles_on_login** activates all granted roles at login.
- **WITH ADMIN OPTION** permits granting or revoking the role. **WITH GRANT OPTION** concerns delegation of privileges; role administration does not itself grant authority to drop the role.
- **DROP ROLE** removes roles and their grants. Mandatory roles cannot be dropped while listed in **mandatory_roles**. A role host component is part of its name, not a client-location restriction.

TLS, credentials and host protection

- **PREFERRED** permits fallback; **REQUIRED** demands encryption; **VERIFY_CA** also validates the certificate chain; **VERIFY_IDENTITY** additionally checks the server hostname. Configure a trusted CA for verification.
- **TLS encryption** and **certificate trust** are separate properties. A trusted private CA can support verified internal connections; possession of an encrypted channel alone does not authenticate the intended server.
- **mysql_config_editor** stores named login paths in **.mylogin.cnf**. Its obfuscation is not strong protection against a determined local attacker; restrict file access. Ordinary option files store readable credentials.
- Run mysqld as a **dedicated OS user**. Restrict datadir and private-key permissions, administrative OS access and network reachability. Public certificates and public keys do not require private-key secrecy.
- **mysql_secure_installation** helps remove anonymous accounts and test access and secure root access. Use firewalls and application-host allowlists; changing port numbers is not access control.

Injection defenses and enterprise controls

- **Parameterized prepared statements** separate SQL structure from data. Avoid SQL concatenation; allowlist dynamic identifiers. Stored procedures remain vulnerable when they build unsafe dynamic SQL.
- **Connection Control** delays repeated failed logins; it does not inspect SQL for injection. Keep its minimum delay no greater than its maximum, adjusting the maximum first when raising both.
- **validate_password** checks password policy. Dictionary-file checks require the applicable strong policy; the dictionary is not a universal exact-password denylist under every policy setting.

Accounts & Security / InnoDB & Transactions

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

INNODB RESPONSIBILITIES

- **Dictionary** — Metadata
- **Tablespaces** — Storage
- **Undo** — Consistency
- **Locks** — Concurrency

- **Enterprise Firewall** learns normalized allowed statements in RECORDING mode, blocks unapproved statements in PROTECTING mode and reports suspicious statements in DETECTING mode. RESET clears the account profile and disables its protection.
- **Enterprise Audit** uses filters assigned to users to select events. Its installation script installs supporting objects; **FORCE_PLUS_PERMANENT** requires successful plugin startup and prevents runtime removal.

03 InnoDB storage, metadata and transactions

Dictionary and system schemas

- **MySQL 8.0 data dictionary** stores transactional metadata for tables, views, stored programs and other objects, backing INFORMATION_SCHEMA. Query-plan caches and live performance metrics are separate concerns.
- **SDI** is JSON serialized dictionary information. InnoDB stores SDI within tablespaces; engines such as MyISAM use .sdi files. Copying a schema directory does not register its objects automatically.
- **mysql** holds system data such as grant tables, plugins, help and time-zone definitions. **INFORMATION_SCHEMA** exposes metadata; **Performance Schema** instruments activity; **sys** provides useful diagnostic views.
- **SHOW CREATE TABLE** reports a table definition; **mysqldump --no-data schema table** exports definition SQL. **SHOW INDEX** and INFORMATION_SCHEMA.STATISTICS report indexes; DESCRIBE is not a full DDL export.

Tablespaces and file placement

- **File-per-table** stores one table and its indexes in an .ibd file. **General tablespaces** can share one data file among multiple tables; create tables in them or move eligible tables with ALTER TABLE.
- **System tablespace** contains structures such as the change buffer. Modern 8.0 stores dictionary tables in **mysql.ibd** and undo in separate tablespaces; doublewrite storage changed to separate files in 8.0.20.
- **innodb_data_file_path** configures system-tablespace files; only the final file can autoextend. Shared system or general tablespace free space is normally reused internally rather than returned by dropping one table.
- **Undo tablespaces** support rollback and consistent reads. **Temporary tablespaces** hold temporary data; distinguish explicit temporary tables, optimizer internal tables and TempTable spill files when investigating disk use.
- **innodb_directories** supplies additional directories to scan for InnoDB tablespaces during discovery. It does not move data files or replace datadir; temporary paths are configured separately.

InnoDB & Transactions

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

INNODB RESPONSIBILITIES

- **Dictionary** — Metadata
- **Tablespaces** — Storage
- **Undo** — Consistency
- **Locks** — Concurrency

Transactions, MVCC and locks

- **InnoDB** supports transactions, crash recovery and **MVCC**. Ordinary consistent SELECT reads generally avoid blocking row updates; **FOR UPDATE** takes locking reads and requires transaction-boundary awareness.
- **Record locks** protect index records; **gap** and **next-key locks** can protect ranges. **Intention locks** announce row-lock intent at table level; they are distinct from exclusive whole-table locks.
- **Metadata locks** protect object definitions and can make DDL wait for an open transaction. Inspect Performance Schema.metadata_locks and map its owner thread through Performance Schema.threads.
- **MyISAM** is nontransactional and uses table-level locks. ROLLBACK cannot undo its changes. **BLACKHOLE** discards stored rows; **ARCHIVE** is not a transactional substitute for InnoDB.
- **DDL** such as TRUNCATE commonly causes implicit commits. Atomic DDL protects supported DDL against partial failure; it does not make schema changes reversible with a user ROLLBACK.

Deadlocks and table maintenance

- **Deadlock** is a cycle of incompatible lock waits; InnoDB normally detects it and rolls back a victim transaction. A **lock wait timeout** is a different condition and normally rolls back the waiting statement.
- **SHOW ENGINE INNODB STATUS** includes the latest detected deadlock. **innodb_print_all_deadlocks=ON** logs all detected deadlocks; **sys.innodb_lock_waits** and Performance Schema data locks show current contention.
- **ANALYZE TABLE** refreshes optimizer statistics; **CHECK TABLE** checks supported structures; **OPTIMIZE TABLE** can rebuild InnoDB storage. Rebuilding a file-per-table table can reclaim space after large deletes.
- **mysqlcheck** sends maintenance statements to a running server. REPAIR TABLE does not repair InnoDB corruption. Plan rebuild space, lock effects and transaction completion before maintenance.
- **ALTER TABLE t DROP PRIMARY KEY, ADD PRIMARY KEY(id)** replaces a primary key when the new column meets uniqueness and nullability rules. InnoDB clusters table data by the primary key, so replacement rebuilds storage.

InnoDB & Transactions / Monitoring & Performance

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

TUNE FROM EVIDENCE

Measure Workload → **Explain** Plan → **Improve** Access → **Compare** Results

Encryption and key recovery

- **ALTER TABLE t ENCRYPTION='Y'** enables encryption for an eligible InnoDB table. A configured **keyring** protects master keys used to wrap tablespace keys.
- **Data-at-rest encryption** protects disk content, including indexes, while pages are decrypted in memory. TLS protects network traffic separately; encryption does not remove SQL authorization requirements.
- **ALTER INSTANCE ROTATE INNODB MASTER KEY** rotates the wrapping key. Back up keyring material with the recovery plan; losing required keys can make encrypted data unrecoverable.
- **binlog_encryption** can encrypt newly created binary and relay log files using keyring support. Enabling it does not retroactively encrypt old files or enable TLS on replication connections.

04 Monitoring, query tuning and capacity

Choose the right diagnostic source

- **Error log** reports lifecycle and failures; **general query log** records received statements; **slow query log** records qualifying completed statements; **binary log** records changes for replication and recovery.
- **long_query_time**, **min_examined_row_limit** and **log_queries_not_using_indexes** determine slow-log eligibility. **log_throttle_queries_not_using_indexes** limits the rate of no-index logging, not the row threshold.
- **mysqldumpslow -s at** sorts slow-log summaries by average query time. Slow logs and instrumented Performance Schema statement events can expose **rows examined**, unlike the general log.
- **Performance Schema** needs applicable instruments and consumers enabled. Use statement digests for normalized workload totals and thread IDs to correlate waits, memory and connection IDs.
- **variables_by_thread** exposes other threads' session system variables; **user_variables_by_thread** holds SQL user variables. Use memory summary tables to distinguish per-thread, account and global allocation.
- **Error-log rotation** for a file destination can rename the old file, then use **FLUSH ERROR LOGS** to reopen it. Binary-log expiration settings do not rotate the error log.

Monitoring & Performance

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

TUNE FROM EVIDENCE

Measure Workload → **Explain** Plan → **Improve** Access → **Compare** Results

Execution plans and index design

- **EXPLAIN** estimates join order, access paths, rows and filtered percentages. **EXPLAIN ANALYZE** executes the statement and reports actual timings and row counts; estimates are not exact result counts.
- **ALL** denotes a full table scan; **index** denotes a full index scan; **range**, **ref** and **eq_ref** describe progressively different lookup patterns. Interpret each with key, rows and predicates.
- **B-tree indexes** support comparisons, ranges and ordering. **FULLTEXT** serves text search and **SPATIAL** serves spatial data. Match indexes to WHERE, JOIN and ORDER BY expressions.
- **Composite indexes** usually support their leftmost prefixes. A **covering index** contains all needed columns; InnoDB secondary indexes also carry the primary key. Extra indexes increase write and storage cost.
- **Functional indexes** or indexed generated columns can support predicates such as MONTH(birth_date). An index on the raw date alone is not equivalent to an index on that expression.

Statistics and optimizer choices

- **Persistent InnoDB statistics** survive restarts. **ANALYZE TABLE** refreshes them; **innodb_stats_auto_recalc** enables automatic recalculation after sufficient change. More sample pages trade extra work for potentially better estimates.
- **Histograms** describe column value frequencies and help selectivity and join-order estimates, especially for skewed values. Create them with ANALYZE TABLE ... UPDATE HISTOGRAM; they are statistics, not access indexes.
- **Invisible indexes** remain maintained but are normally excluded from optimizer choice. They support testing index removal; explicitly enabling **use_invisible_indexes** allows controlled plan testing.
- **Hash joins** do not require hash indexes. They can avoid expensive unindexed joins; **join_buffer_size** influences memory, and larger operations can spill to disk. Eligibility varies across 8.0 releases.

Memory, I/O and durability

- **innodb_buffer_pool_size** caches InnoDB pages. Budget RAM for the OS, other global structures and concurrent per-session allocations; avoid swapping. More buffer-pool instances can reduce contention but are not universally faster.
- **key_buffer_size** caches MyISAM indexes; **table_open_cache** is shared. Sort, join and read buffers are allocated per relevant operation or session; raising them with max_connections can multiply memory demand.

Monitoring & Performance / Backup & Recovery

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

TUNE FROM EVIDENCE

Measure Workload → **Explain** Plan → **Improve** Access → **Compare** Results

- **innodb_flush_log_at_trx_commit=1** and **sync_binlog=1** favor durable commits. Relaxing flushing can reduce I/O while increasing crash-loss exposure; disabling doublewrite removes protection against torn page writes.
- **Redo capacity** absorbs write bursts and checkpoint pressure. Earlier 8.0 uses innodb_log_file_size and file count; newer 8.0 supports innodb_redo_log_capacity. More capacity also affects recovery and storage needs.
- Use **status counter deltas** over an interval to measure rates and cache pressure. More max_connections does not fix slow queries; investigate table-open churn, disk latency, CPU, contention and workload changes.

Workload priorities and growth

- **Enterprise Monitor** tracks availability, advisors, alerts and query performance. Agent-based monitoring adds host OS metrics; agentless collection can obtain database and query data without local host visibility.
- **Query Analyzer** groups normalized statements and supports time-range investigation and statement export. Compare execution count, total latency, per-call latency, rows and response-time distribution; high-frequency moderately slow queries can dominate load.
- **QRTi** summarizes query response-time quality against configured thresholds; lower scores warrant investigation. Performance Schema history consumers record recent events, not a permanent ranked list of longest queries.
- **Capacity planning** combines measured resource trends, peaks, forecast growth and application-team plans. **Scale up** adds CPU, RAM or storage; **scale out** adds servers, read replicas or shards.

05 Backup, restore and point-in-time recovery

Choose a recoverable backup

- **RPO** limits acceptable data loss; **RTO** limits recovery duration. Choose backup frequency, binlog retention, restore granularity and storage from these targets, data size and the permitted lock window.
- **Logical backups** export definitions and rows through a running server, offering portability and selective restore. **Physical backups** copy native files, usually restoring large datasets faster but requiring compatible formats.
- **Cold backups** copy a stopped, cleanly shut-down instance. **Hot backups** allow ongoing access; mixed transactional and nontransactional engines need explicit consistency coordination or a suitable replica backup.

Backup & Recovery

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

RECOVERY SEQUENCE

Backup Base state → **Prepare** Restore → **Binlogs** Replay → **Validate** Resume

- **Filesystem snapshots** shorten the capture window but must cover a consistent set of data and required logs. Coordinate nontransactional tables; InnoDB may need crash recovery after restore.
- **Replica backups** can offload production I/O and locks. Record replication position and account for lag. Replication propagates mistakes, so it does not replace retained backups and restore tests.

Logical backup tools and switches

- **mysqldump --single-transaction** uses a consistent InnoDB snapshot without locking its rows throughout the dump. It does not guarantee consistency for MyISAM; avoid concurrent disruptive DDL.
- **--master-data=2** writes source coordinates as a SQL comment; later releases call it **--source-data**. With single-transaction, a brief global read lock establishes coordinates; it is not a fully lock-free operation.
- **--no-data** exports definitions; **--no-create-info** suppresses CREATE TABLE statements. **--routines** and **--events** include those objects; plan account and privilege exports explicitly.
- **mysqlpump** supports parallel logical dumping and include/exclude patterns. **--exclude-databases=% --users** exports accounts as SQL; internal and virtual schemas have special selection rules.
- **--set-gtid-purged=OFF** suppresses GTID_PURGED output when appropriate, such as importing application data into an existing replication group. A dump's GTID set can cover more than its selected tables.
- **--flush-logs** starts a new binary log. **--delete-master-logs** purges older source logs after dumping; it does not archive their contents and can strand replicas that still need them.

Physical backup and restore

- **MySQL Enterprise Backup (mysqlbackup)** supports hot InnoDB, incremental and compressed backups, validation and image-based output. Non-InnoDB content can require locks; physical backup is not an editable SQL dump.
- **apply-log** prepares a physical backup using captured redo. **backup-and-apply-log** combines capture and preparation; **copy-back** restores files from the backup into server locations.
- **backup-image** selects a single-file image; **backup-dir** holds working files and metadata. Restore compressed images with the required extraction, decompression and preparation steps before startup.
- **Optimistic backup** copies quieter tables before busy tables to shorten the period of redo collection. Busy-table classification helps control redo volume; it does not eliminate the need for recovery logs.

Backup & Recovery / Replication & GTIDs

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

RECOVERY SEQUENCE

Backup Base state → **Prepare** Restore → **Binlogs** Replay → **Validate** Resume

- Restore with the server stopped, correct ownership and required keys. **InnoDB .ibd copying alone** is insufficient: use supported transportable-tablespace or backup procedures, or restore separately and logically export one table.
- **MyISAM physical files** include .MYD data, .MYI indexes and .sdi metadata. MySQL 8.0 dictionary registration needs the supported SDI import process; copying files alone does not recreate a dropped dictionary entry.

Binary logs and recovery boundaries

- **Point-in-time recovery** restores a base backup, then replays subsequent binary logs in order. **mysqlbinlog binlog.000003 binlog.000004 | mysql** converts and applies logged events.
- **--start-position**, **--stop-position** and datetime bounds select recovery ranges. Stop before the damaging transaction and preserve transaction boundaries; binary logs do not automatically undo an accidental DROP.
- **mysqlbinlog --base64-output=DECODE-ROWS -v** displays row events as annotated pseudo-SQL for inspection. Human-readable decoding is distinct from producing a replay stream.
- **--rewrite-db='old->new'** rewrites database names in supported events. Statement-based rewriting has default-database limitations; it is not a general text replacement for every qualified identifier.
- **FLUSH BINARY LOGS** rotates logs; max_binlog_size can trigger rotation between transactions. **PURGE BINARY LOGS** and expiration manage retention; rotation alone does not delete history needed by replicas or recovery.

06 Asynchronous replication and GTIDs

Replication pipeline and modes

- **Source binary log -> dump thread -> replica receiver -> relay log -> applier** is the asynchronous pipeline. The receiver connects and requests events; the applier executes received events.
- **Unique server_id** values identify topology members. Sources need binary logging and replication credentials; **log_replica_updates** (formerly log_slave_updates) records applied events for downstream replication and promotion.
- **ROW** logging records row changes; **STATEMENT** records SQL; **MIXED** selects between formats. Replicas connect over TCP, and a replication account needs replication privileges rather than SELECT on every table.

Replication & GTIDs

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

REPLICATION DATA PATH

Source Binary log → **Receiver** Transfer → **Relay log** Queue → **Applier** Execute

- **Asynchronous replication** supports read scaling, offloaded backups and remote copies, but lag and failover data loss are possible. Circular replication does not automatically resolve write conflicts.
- **Semisynchronous replication** waits for configured replica acknowledgments of received, durably logged events, not completed apply. Timeout can fall back to asynchronous mode; promotion and routing remain separate tasks.

GTID identity and provisioning

- **GTID = server UUID:transaction number** identifies a transaction across replication. **gtid_executed** tracks executed GTIDs; **gtid_purged** tracks executed GTIDs no longer represented in available binary logs.
- **gtid_mode=ON**, **enforce_gtid_consistency=ON** and compatible binary logging establish GTID operation. **SOURCE_AUTO_POSITION=1**, formerly MASTER_AUTO_POSITION=1, lets a replica request missing GTIDs.
- **GTID_SUBSET(a,b)** tests containment; **GTID_SUBTRACT(a,b)** finds transactions in a but absent from b. Compare sets by source UUID, not merely the largest numeric suffix.
- If required GTIDs have been **purged on the source**, auto-position cannot recreate their events. Provision from a suitable backup or donor; resetting replica connection metadata does not restore missing data.
- **Errant transactions** originate outside the intended source stream on a replica. An empty GTID transaction can mark an event executed without applying its data; neither technique is a substitute for repairing consistency.

Lag, filters and multi-source channels

- **SHOW REPLICA STATUS**, formerly SHOW SLAVE STATUS, reports receiver/applier health, errors, positions and GTID sets. **Seconds_Behind_Source** is a lag indicator, not proof that all source data has arrived.
- **Growing lag** can result from serial apply, lock contention, large transactions, slow storage or row searches without useful keys. More parallel workers help only when dependencies and resources allow.
- **Source-side binlog filters** can reduce transferred events but also remove recovery and replication history. **Replica-side filters** act after receipt. Statement and row formats interpret database filters differently.
- **Multi-source replication** uses one named channel per source, commonly consolidating independent datasets. It supports GTID auto-positioning but provides no automatic conflict detection or resolution between sources.
- **relay_log_recovery** supports crash recovery of relay-log state when properly configured. Recover from known executed coordinates and intact source logs; blindly deleting relay files can break replication metadata.

InnoDB Cluster

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

CLUSTER RESPONSIBILITIES

- **Group** — Replicate
- **Quorum** — Progress
- **Shell** — Administer
- **Router** — Connect

07 Group Replication and InnoDB Cluster

Group membership and consistency

- **Group Replication** combines membership, transaction ordering and conflict certification. Use InnoDB tables with a primary key or suitable non-null unique key, ROW logging, GTIDs and replicated-update logging.
- **Single-primary** mode elects one writer; **multi-primary** permits writes on multiple members with certification constraints. More writers do not remove conflicts or increase quorum tolerance.
- **Majority quorum** permits progress during unexpected failures: three members tolerate one failure; five tolerate two. A minority partition cannot independently continue committing normal group writes.
- **memberWeight** influences primary election among eligible members. **BEFORE_ON_PRIMARY_FAILOVER** holds new-primary work until its backlog is applied, reducing stale-read exposure after failover.
- **Group communication TLS** and **distributed recovery TLS** are separate channels with their own settings. Plan memberSslMode and recovery credentials together rather than assuming one encrypts every path.

Shell, Router and provisioning

- **InnoDB Cluster** combines Group Replication, **MySQL Shell AdminAPI** and **MySQL Router**. Shell configures and manages topology; Router discovers metadata and routes client connections.
- **dba.checkInstanceConfiguration()** validates readiness; **dba.configureInstance()** prepares an instance. **dba.createCluster()** creates cluster metadata and **cluster.addInstance()** joins another instance.
- **mysqlrouter --bootstrap user@host:port --directory=path** reads cluster metadata and writes Router configuration and supporting files. It does not create the database cluster or reconfigure every server.
- **Clone recovery** provisions an existing target with physical InnoDB data from a donor; incremental recovery applies missing transactions. Select based on data state, available logs and compatibility.
- **Clone plugin** privileges include BACKUP_ADMIN for the donor account and CLONE_ADMIN for the recipient operator. Plan recipient data replacement, filesystem permissions and restart; cloning does not install server binaries.

InnoDB Cluster

MySQL 8.0 Database Administrator

KNOWLEDGE CHEATSHEET

CLUSTER RESPONSIBILITIES

- **Group** — Replicate
- **Quorum** — Progress
- **Shell** — Administer
- **Router** — Connect

Failure diagnosis and recovery

- **cluster.status()** distinguishes ONLINE members, unavailable members, quorum and remaining failure tolerance. A group can have quorum with no capacity to tolerate another unexpected failure.
- **Primary failure** triggers election when the surviving group has quorum. Router refreshes topology for routing; applications must reconnect and handle interrupted transactions.
- **cluster.rejoinInstance()** returns an existing member when appropriate. **forceQuorumUsingPartitionOf()** is for lost quorum; isolate competing partitions before forcing membership.
- **dba.rebootClusterFromCompleteOutage()** restores a stopped cluster. Compare GTID sets and use the most up-to-date valid seed; stale-seed recovery can discard transactions.
- **dba.dropMetadataSchema()** removes AdminAPI metadata without itself stopping Group Replication, disrupting metadata-dependent management and Router discovery. **killSandboxInstance()** simulates abrupt failure; stopSandboxInstance() is graceful.