MySQL Goes to 8!

Geir Høydalsvik, MySQL Engineering
FOSDEM 2020, Database Track
February 2nd, 2020, Brussels
Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, timing, and pricing of any features or functionality described for Oracle’s products may change and remains at the sole discretion of Oracle Corporation.
WHAT IS MySQL?

• Relational Database
  • Transactional, ACID
  • InnoDB storage engine: ARIES, MVCC
  • OLTP: low latency, high throughput

• Replication
  • Read scale out, High Availability

• Simple, Solid, Secure
  • Easy to use, Proven at scale
LAST 10 YEARS

• Major investments
  • Reengineering
  • Features
  • Quality
• Major releases
  • MySQL 5.5
  • MySQL 5.6
  • MySQL 5.7
  • MySQL 8.0
MySQL 8 - IS LIGHT YEARS AWAY FROM 5.X

@MySQL 8 is light years away from 5.x versions. You now have:

- CTE and Recursive CTE
- Window Functions
- SKIP LOCKED, NO WAIT
- Hash Joins (Coming in 8.0.18)
- Explain Analyze giving you the Actual Plan (Coming in 8.0.18)
The Basics

SQL, JSON, GIS, Character Sets, Collations, Functions
MySQL 8 - OPTIMIZER

Parse phase
- Parse step
- Contextualization step
- Abstract Syntax Tree

Optimize phase
- Range optimizer
- Join optimizer
- Physical plan

Prepare phase
- Resolve step
- Transform step
- Logical Plan

Execute phase
- Produce iterator tree
- Execute iterator
- Resultset
MySQL 8 - HISTOGRAM

- Provides the optimizer with information about column value distribution

```
ANALYZE TABLE table UPDATE HISTOGRAM ON column WITH n BUCKETS;
```

- Table sampling for efficiency
MySQL 8 – ITERATOR EXECUTOR

- Each operation is an iterator
- Execution loop reads from root node
  Row by row
  May trigger multiple read calls further down
- Common interface
  Init()
  Read()

```
SELECT * FROM t1 JOIN t2 ON t1.a = t2.a;
```
MySQL 8 - HASH JOIN

- “Just another iterator”
- Faster than Block Nested Loop
- In-memory if possible
- Spill to disk if necessary
- Used for inner equi-joins in 8.0.18
  - And also for outer, semi and anti joins in 8.0.20
- Hash Join replaces Block Nested Loop

```
SELECT * FROM t1 JOIN t2 ON t1.a = t2.a;
```
MySQL 8 - EXPLAIN ANALYZE

- Wrap iterators in instrumentation nodes
- Measurements
  Time (in ms) to first row
  Time (in ms) to last row
  Number of rows
  Number of loops
- Execute the query and dump the stats
- Built on EXPLAIN FORMAT=TREE

-> Inner hash join (t2.a = t1.a) (cost=0.70 rows=1) (actual time=0.441..0.441 rows=0 loops=1)
  -> Table scan on t2 (cost=0.35 rows=1) (never executed)
  -> Hash
    -> Table scan on t1 (cost=0.35 rows=1) (actual time=0.220..0.220 rows=0 loops=1)
MySQL 8 – CHARACTER SET AND COLLATIONS

- MySQL 8 defaults to UTF-8
- Emoji, CJK characters, ...
- Unicode 9.0 collations with accent, case, and kana sensitivity
- Unicode support for REGEXP
MySQL 8 - Common Table Expression (WITH clause)

• Non-recursive

WITH cte AS (subquery)
SELECT ... FROM cte, t1...

• Recursive

WITH RECURSIVE cte AS
( SELECT ... FROM table_name
  UNION [DISTINCT|ALL]
  SELECT ... FROM cte, table_name )
SELECT ... FROM cte;

A Common Table Expression (CTE) is just like a derived table, but its declaration is put before the query block instead of in the FROM clause.

• Better readability
• Can be referenced multiple times
• Can refer to other CTEs
• Improved performance
A window function performs a calculation across a set of rows that are related to the current row, similar to an aggregate function.

But unlike aggregate functions, a window function does not cause rows to become grouped into a single output row.

Window functions can access values of other rows “in the vicinity” of the current row.
MySQL 8 - Window Functions (OVER clause)

Sum up total salary for each department:

```
SELECT name, dept_id, salary, 
    SUM(salary) OVER (PARTITION BY dept_id) AS dept_total 
FROM employee 
ORDER BY dept_id, name;
```

**PARTITION** == disjoint set of rows in result set
MySQL 8 – LATERAL DERIVED TABLES

• Can refer to other tables in the same FROM clause
• Sometimes referred to as the SQL «for each» equivalent

SELECT ... FROM t1, LATERAL (SELECT ... FROM ... WHERE ... = t1.col) AS derived, t2 ...
MySQL 8.0 - FUNCTIONAL INDEXES

• Index over an expression

```sql
CREATE TABLE t1 (col1 INT, col2 INT);
CREATE INDEX idx1 ON t1 ((col1 + col2), (col1 - col2), col1);
```

• Document content, e.g. JSON array

```sql
CREATE TABLE lottery (data JSON);
CREATE INDEX ticket_idx ON lottery ((CAST(data->'$.lottery_tickets' AS UNSIGNED INT ARRAY)));
```
MySQL 8.0 – INVISIBLE INDEXES

• Indexes are “hidden” to the MySQL Optimizer
  • Not the same as “disabled indexes”
  • Contents are fully up to date and maintained by DML
• Two use cases:
  • Soft Delete: What will happen if I delete this index?
  • Staged Rollout: I will create this index over night and make it visible when I am at work tomorrow
MySQL 8 - CHECK CONSTRAINT

• Standard SQL Syntax

[ CONSTRAINT [symbol] ] CHECK ( condition) [ [ NOT ] ENFORCED ]

• Example

CREATE TABLE t1 (c1 INTEGER CONSTRAINT c1_chk CHECK (c1 > 0) ,
c2 INTEGER CONSTRAINT c2_chk CHECK (c2 > 0) ,
CONSTRAINT c1_c2_chk CHECK (c1 + c2 < 9999) );
MySQL 8 - Expressions as Default Values

- No longer limited to literal values
  
  ```sql
  CREATE TABLE t1 (uuid BINARY DEFAULT (UUID_TO_BIN(UUID())));
  CREATE TABLE t2 (a INT, b INT, c INT DEFAULT (a+b));
  CREATE TABLE t3 (a INT, b INT, c POINT DEFAULT (POINT(0,0)));
  CREATE TABLE t4 (a INT, b INT, c JSON DEFAULT ('[]'));
  ```

- Useful for types without literal values
  - GEOMETRY, POINT, LINESTRING, POLYGON, ...
MySQL 8 - NOWAIT and SKIP LOCKED

SELECT * FROM tickets
WHERE id IN (1,2,3,4)
AND order_id IS NULL
FOR UPDATE
NOWAIT;

SELECT * FROM tickets
WHERE id IN (1,2,3,4)
AND order_id IS NULL
FOR UPDATE
SKIP LOCKED;

Error immediately if a row is already locked

Non deterministically skip over locked rows
MySQL 8.0 - NEW FUNCTIONS

- **REGEXP**
  - REGEXP_INSTR, REGEXP_LIKE, REGEXP_REPLACE, REGEXP_SUBSTR

- **UUID**
  - UUID_TO_BIN, BIN_TO_UUID, IS_UUID

- **STATEMENT_DIGEST**
  - STATEMENT_DIGEST, STATEMENT_DIGEST_TEXT

- **Bit operations are now allowed on all binary data types**
  - BINARY, VARBINARY, BLOB, TINYBLOB, MEDIUMBLOB and LONGBLOB
# MySQL 8 - JSON Functions

<table>
<thead>
<tr>
<th>JSON Function</th>
<th>JSON Function</th>
<th>JSON Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON_ARRAY_APPEND()</td>
<td>JSON_MERGE<a href="">&amp;_PREERVE</a></td>
<td>JSONPRETTY()</td>
</tr>
<tr>
<td>JSON_ARRAY_INSERT()</td>
<td>JSON_OBJECT()</td>
<td>JSON_STORAGE_SIZE()</td>
</tr>
<tr>
<td>JSON_ARRAY()</td>
<td>JSON_QUOTE()</td>
<td>JSON_STORAGE_FREE()</td>
</tr>
<tr>
<td>JSON_CONTAINS_PATH()</td>
<td>JSON_REMOVE()</td>
<td>JSON_ARRAYAGG()</td>
</tr>
<tr>
<td>JSON_CONTAINS()</td>
<td>JSON_REPLACE()</td>
<td>JSON_OBJECTAGG()</td>
</tr>
<tr>
<td>JSON_DEPTH()</td>
<td>JSON_SEARCH()</td>
<td>JSON_MERGE_PATCH()</td>
</tr>
<tr>
<td>JSON_EXTRACT()</td>
<td>JSON_SET()</td>
<td>JSON_TABLE()</td>
</tr>
<tr>
<td>JSON_INSERT()</td>
<td>JSON_TYPE()</td>
<td>JSON_OVERLAPS()</td>
</tr>
<tr>
<td>JSON_KEYS()</td>
<td>JSON_UNQUOTE()</td>
<td>JSON Schema</td>
</tr>
<tr>
<td>JSON_LENGTH()</td>
<td>JSON_VALID()</td>
<td>JSON Array Indexes</td>
</tr>
</tbody>
</table>

23 Copyright © 2020 Oracle and/or its affiliates.
MySQL 8 - JSON_TABLE()
From JSON Document to SQL Table

```sql
mysql> SELECT emps.* FROM JSON_TABLE(@jsonempl, "$[*]" COLUMNS
(id INT PATH ".id", name VARCHAR(45) PATH ".name", age INT
PATH ".age") emps;
+--------------------------+
| id  | name | age |
+--------------------------+
| 1   | John | 34  |
| 2   | Mary | 40  |
| 3   | Mike | 44  |
+--------------------------+
3 rows in set (0,00 sec)
```
MySQL 8 - Full Geography Support

Full Geography Support
• Longitude, Latitude
• Projected – Flat/Across 2 dimensions
• Geographic – Spheroid
• 5107 predefined SRSs from the EPSG Dataset 9.2
• 4628 projected, 479 geographic

Built in and ready to use
MySQL Document Store

SQL + NoSQL = MySQL
MySQL 8 – DOCUMENT STORE

- JSON documents
- Schemaless
- Document collections
  - `db.getCollections()`
- CRUD operations
  - `add()`, `find()`, `modify()`, `remove()`
- Connectors (X DevAPI)
- Asynchronous protocol (X Protocol)
- Server front end (X Plugin)
MySQL 8 – DOCUMENT STORE

- Full Node.js integration
  - Support for “Promises”
- Autocompletion support in IDEs
  - Due to method chaining support
- Intuitive Documentation & Tutorials
  - Example:

**COLLECTION.add Function**
Operations

Secure, Monitor, Manage, and Upgrade
MySQL 8.0 - SECURE BY DEFAULT

- User should not have to “opt in” for security
- Minimize attack surface
- Minimize process permissions
- Minimize file permissions
- Minimize privileges
- Strong authentication
- Strong encryption methods
MySQL 8.0 - AUTHENTICATION

• Strong default authentication
  • caching_sha2_password

• Pluggable authentication
  • Client and server side
  • Support for integration with external authentication systems

• Can use the OS login to authenticate
  • unix sockets
MYSQL 8.0 - PASSWORD MANAGEMENT

• Password rotation policies and enforcement
• Password history and reuse protection
• Password strength evaluation and enforcement
• Password generation
• Two passwords per user
• Brute-force attack protection
MySQL 8.0 - AUTHORIZATION

- Standards compliant user and roles management
  - Users, Roles, and Privileges

- SQL Standard Information Schema views for SQL Roles
  - APPLICABLE_ROLES, ENABLED_ROLES, ROLE_TABLE_GRANTS, ...

- Fine grained permissions management
  - Admin Privileges
MYSQL 8.0 – FOCUS ON OpenSSL

- OpenSSL is linked dynamically
- OpenSSL versions can be patched at OS level
- Support for FIPs compliance
- Can reconfigure certificates without restarting the server
- Encrypt over the wire, TLS 1.3 support
- Encrypt data at rest
MYSQL 8.0 – MONITORING

• Information Schema tables
  • Persistent meta data
  • Views on the data dictionary tables
  • Data dictionary is implemented as system tables in InnoDB
• Performance Schema tables
  • Volatile meta data, lost upon restart
• SYS Schema
  • Stored routines
  • Task oriented reports
MySQL 8.0 - PERFORMANCE SCHEMA

- Statement latency statistics
  - What is the latency distribution for a given SQL statement?
- Data Locks
  - Which user threads are waiting for which locks? Who holds them?
- SQL Errors
  - Which errors have been sent back to clients? When? How often?
- Configuration Variables
  - What is the current value? Who set it? When?
MYSQL 8.0 – MANAGEMENT

- Manage over a user connection or use the MySQL Shell
  - Eliminate the need to access the host machine
  - Eliminate the need to restart the server
- Configuration changes by SQL DDL
  - SET PERSIST (mostly online)
  - RESTART (still required in some cases)
- Auto Upgrade
  - The system reads the «on disk» version
  - Execute upgrade code if needed
Ease of Use
- To a new level
Meet Ada

- Meet Ada, the DevOps
- Ada is smart
- Ada is using the MySQL Shell

Hello!
MySQL Shell: Modern

- Colorful Prompt Themes
- Autocompletion
- Syntax Highlighting
- Context Sensitive Help
- Command History
- Pager, less/more
- Output Formats
MySQL Shell: Flexible

- SQL, JavaScript, Python
- Interactive & Batch
- SQL Client
- Document Store
- InnoDB Cluster Admin
- InnoDB ReplicaSet Admin
MySQL Shell : Extendible

• Utilities
  – upgradeChecker()
  – importJSON()
  – importTable()

• Reporting Framework
  – \show \watch

• User Defined Plugins
  – JS or Python

I adapted it to my prod environment!
MySQL CLONE

Fast instance provisioning
MySQL 8 – CLONE

WHY IS CLONE SUCH A BIG DEAL?

• Puts the power of **fast instance provisioning** into the hands of everybody
• Reduces the complex provisioning procedure to a few simple steps
• Even happens automatically when needed when using **InnoDB Cluster**
• Can be done fully remotely

Ada

CLONE makes my life easy!
MySQL 8 - CLONE Directly from SQL

“User traffic is growing and I need a new read replica”

Provision a new slave (RECIPIENT) from an existing master (DONOR)
MySQL – 8 CLONE Setup the **DONOR**

```sql
mysql> INSTALL PLUGIN CLONE SONAME "mysql_clone.so";
mysql> CREATE USER clone_user IDENTIFIED BY "clone_password";
mysql> GRANT BACKUP_ADMIN ON *.* to clone_user;
```
MySQL 8 - CLONE Setup the RECIPIENT

mysql> INSTALL PLUGIN CLONE SONAME "mysql_clone.so";
mysql> SET GLOBAL clone_valid_donor_list = "donor.host.com:3306";
mysql> CREATE USER clone_user IDENTIFIED BY "clone_password";
mysql> GRANT BACKUP_ADMIN ON *.* to clone_user;
MySQL 8 - Connect to **RECIPIENT** and execute CLONE SQL statement

```
mysql> CLONE INSTANCE
    -> FROM clone_user@donor.host.com:3306
    -> IDENTIFIED BY "clone_password";
```
MySQL 8 - CLONE Check Status

mysql> select STATE, ...
    > from performance_schema.clone_status;

<table>
<thead>
<tr>
<th>STATE</th>
<th>START TIME</th>
<th>DURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>In Progress</td>
<td>2019-07-17 17:23:26</td>
<td>4.84 m</td>
</tr>
</tbody>
</table>
### MySQL 8 - CLONE Check Progress

mysql> select STATE, ...
    > from performance_schema.clone_progress;

<table>
<thead>
<tr>
<th>STAGE</th>
<th>STATE</th>
<th>START TIME</th>
<th>DURATION</th>
<th>Estimate</th>
<th>Done(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DROP DATA</td>
<td>Completed</td>
<td>17:23:26</td>
<td>790.86 ms</td>
<td>0 MB</td>
<td>100%</td>
</tr>
<tr>
<td>FILE COPY</td>
<td>Completed</td>
<td>17:23:27</td>
<td>10.33 m</td>
<td>94,729 MB</td>
<td>100%</td>
</tr>
<tr>
<td>PAGE COPY</td>
<td>Completed</td>
<td>17:33:47</td>
<td>15.91 s</td>
<td>11,885 MB</td>
<td>100%</td>
</tr>
<tr>
<td>REDO COPY</td>
<td>Completed</td>
<td>17:34:03</td>
<td>1.07 s</td>
<td>293 MB</td>
<td>100%</td>
</tr>
<tr>
<td>FILE SYNC</td>
<td>In Progress</td>
<td>17:34:04</td>
<td>51.68 s</td>
<td>0 MB</td>
<td>0%</td>
</tr>
<tr>
<td>RESTART</td>
<td>Not Started</td>
<td>NULL</td>
<td>NULL</td>
<td>0 MB</td>
<td>0%</td>
</tr>
<tr>
<td>RECOVERY</td>
<td>Not Started</td>
<td>NULL</td>
<td>NULL</td>
<td>0 MB</td>
<td>0%</td>
</tr>
</tbody>
</table>
MySQL InnoDB Cluster

High Availability - Out of the Box
MySQL 8 - MySQL InnoDB Cluster

- MySQL Group Replication
  - High Availability
  - Elastic, Fault Tolerant, Self Healing
- MySQL Router
  - Connection Routing, Load Balancing
- MySQL Shell
  - Easy Setup & Administration
MySQL 8 – GROUP REPLICATION

Initialize group
Detect node failure
Reestablish group
Elect new primary
Recover from failure
Rejoin group
Grow and shrink group
Provision new members
Topology Meta-data
Router Notification
Observability
MySQL 8 – REPLICATION TECHNOLOGY

- Multi-threaded Replication Applier With WRITESETs
  - Faster Slaves - higher end-to-end throughput

- Global Transaction Identifiers (GTIDs)
  - Track replication changes seamlessly across replication chains

- Replicated state machines
  - Coordinate, synchronize and execute distributed operations using well known and proven distributed algorithms such as Paxos
MySQL 8 – DISTRIBUTED AND COORDINATED AUTOMATION

- Fault-detection
  - Automatic detection of failed servers in the cluster
- Server fencing
  - Automatic isolation of faulty servers from the app and the cluster
- Data consistency levels
  - Distributed commit protocol enabling reading your own writes
- Distributed recovery
  - Automatic (re)syncing procedure for servers joining a cluster
- Flow control
  - Automatic server throttling preventing unbounded secondary lag
- Membership services
  - Automatic, dynamic list of servers in the cluster and their status
MySQL InnoDB Cluster

Mini-tutorial
MySQL InnoDB Cluster

- configureInstance()
- createCluster()
- addInstance()
- removeInstance()
- rejoinInstance()

And here CLONE is fully automated!
Pre-requisites: Install and start MySQL on 3 servers

Note: mysqld is managed by Linux systemd
mysql-js> dba.configureInstance('clusteradmin@mysql1')

- binlog_checksum = NONE
- enforce_gtid_consistency = ON
- gtid_mode=ON
- server_id= <unique ID>
mysql-js> dba.configureInstance('clusteradmin@mysql2')

binlog_checksum = NONE
enforce_gtid_consistency = ON
gtid_mode=ON
server_id= <unique ID>
mysql-js> dba.configureInstance('clusteradmin@mysql3')

binlog_checksum = NONE
enforce_gtid_consistency = ON
gtid_mode=ON
server_id= <unique ID>
mysql-js> cluster=dba.createCluster('FOSDEM2020')
mysql-js> cluster.status()

```javascript
mysql-js> cluster.status()
{
    "clusterName": "FOSDEM2020",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mysql1:3306",
        "ssl": "REQUIRED",
        "status": "OK_NOT_TOLERANT",
        "statusText": "Cluster is NOT tolerant to any failures."
    },
    "topology": {
        ...
    },
    "topologyMode": "Single Primary"
},
"groupInformationSourceMember": "mysql1:3306"
```

Copyright © 2020 Oracle and/or its affiliates.
mysql-js> cluster.addInstance('clusteradmin@mysql2')
mysql-js> cluster.addInstance('clusteradmin@mysql3')
MySQL Shell: Add Instance with CLONE Progress Reporting

A new instance will be added to the Innodb cluster. Depending on the amount of data on the cluster this may take from a few seconds to several hours.

Adding instance to the cluster...

Monitoring recovery process of the new cluster member. Press q to stop monitoring and let it continue in background.

NOTE: A server restart is expected to happen as part of the clone process. If the server does not support the RESTART command or does not come back after a while, you may need to manually start it back.

* Waiting for clone to finish...

NOTE: 10.6.0.3:3306 is being cloned from 10.6.0.2:3306
** Stage DROP DATA: Completed
** Stage CLONE TRANSFER:
FILE COPY 18056 Completed
PAGE COPY 18056 Completed
REDO COPY 18056 Completed
** Stage RECOVERY:
NOTE: 10.6.0.3:3306 is shutting down...

* Waiting for server restart... ready
10.6.0.3:3306 has restarted, waiting for clone to finish...
* Clone process has finished: 66.63 MB transferred in about 1 second (~66.63 MB/s)

Incremental distributed state recovery is now in progress.

* Waiting for distributed recovery to finish...
NOTE: ‘10.6.0.0:3306’ is being recovered from ‘10.6.0.2:3306’
* Distributed recovery has finished

The instance ‘10.6.0.3’ was successfully added to the cluster.
mysql> cluster.status()

```
{
    "clusterName": "FOSDEM2020",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mysql1:3306",
        "ssl": "REQUIRED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure."
    },
    "topology": {
        ...CUT...
    },
    "topologyMode": "Single Primary"
}
```

MySQL Shell

Status?

FAULT TOLERANT

FOSDEM2020
# mysqlrouter --bootstrap clusteradmin@mysql1 --user=routeradmin
# systemctl start mysqlrouter

Starting mysqlrouter and adding r/w load

MySQL Shell

R/W Load

MySQL Router

FOSDEM2020

MySQL Router

MySQL
mysql1# kill -9 $(pidof mysqld)

Testing... Killing primary mysqld...

MySQL Shell
MySQL: New Primary

OK, mysql1 left the group and mysql2 became the new primary

MySQL Shell

MySQL Router

R/W Load

FOSDEM2020
MySQL: How to regain Fault Tolerance?

1. **Automatic, self healing**
   - Binlog (if GTID available in group)
   - CLONE (otherwise)

2. **Manual fix**
   - Self healing failed, e.g. network failure
   - `rejoinInstance()`
   - CLONE

3. **Replace with new instance (permanent failure)**
   - `removeInstance()`
   - `addInstance()`
   - CLONE
mysql-js> cluster.rejoinInstance('clusteradmin@mysql1')

rejoinInstance()

MySQL Shell

R/W Load

MySQL Router

FOSDEM2020

CLONE
MySQL: Fault Tolerant Again

THANK YOU!

MySQL Shell

R/W Load

MySQL Router

FOSDEM2020
MySQL 8 – CONNECTORS AND DRIVERS

MySQL Engineering
- Node.js Driver (Connector/Node.js)
- Python Driver (Connector/Python)
- C++ Driver (Connector/C++)
- C Driver (Connector/C)
- C API (mysqlclient)
- ADO.NET (Connector/.NET)
- ODBC (Connector/ODBC)
- JDBC (Connector/J)

Community
- PHP Drivers for MySQL
  - (mysqli, ext/mysqli, PDO_MYSQL, PHP_MYSQLND)
- Perl Driver for MySQL (DBD::mysql)
- Ruby Driver for MySQL (ruby-mysql)
- C++ Wrapper for MySQL C API (MySQL++)
- Go MySQL Driver
- NodeJS (mysql, mysql2)
MySQL 8 – SOURCE CODE

• Open Source (GPL)
• GitHub [https://github.com/mysql/mysql-server](https://github.com/mysql/mysql-server)
• Wide platform coverage
• C++ 14 , Use of standard constructs, e.g. std::atomic
• Cleaning up header files dependencies
• Warning-free with GCC 8 and Clang 6
• Asan and Ubsan clean
• [Google C++ Style Guide](https://google.github.io/styleguide/cppguide.html)
• [MySQL Source Code Documentation](https://dev.mysql.com/doc/)
MySQL 8 - The complete list of new features

https://mysqlserverteam.com/the-complete-list-of-new-features-in-mysql-8-0/
MySQL Community on Slack

https://lefred.be/mysql-community-on-slack/

We have 3 nodes A,B,C .. A is primary R/W and at 9:00 AM A went down and B took over and at 11 AM B and C went down .. Last backup was from 11:45 PM from last night

In this scenario we need to merge the writes that happened on A and B to restore until 11 AM

lefred 8:43 PM
you need to restore backup and replay binlogs from B or C

No because B has the writes of A
when A went down, it doesn’t have committed anything that B or C do not have
MySQL on Social Media

https://www.facebook.com/mysql

https://twitter.com/mysql

https://www.linkedin.com/company/mysql