Navigation

Introduction

Release Notes

Lifecycle Policy

Install

Upgrade

- Upgrade

Architecture

- Architecture

Guides

- Create Instance
- Delete Instance
- Access Met
- User Management
- Backup & Restore
- Stop & Star
- Instance Restart
- Parameter Configuration
- Update Spe
- Query Analysis
- Log
- Monitoring
- Scheduling Configuration

HowTo

Introduction

Alauda Database for MySQL is a high availability database solution based on MySQL Group Replication technology and runs on Kubernetes. It provides automated deployment and management capabilities for MySQL Group Replication clusters.

TOC

Key Features

- **High Availability**: Cluster architecture based on MySQL Group Replication, implementing multi-node data automatic synchronization and fault self-healing, with automatic switchover during primary node failure, reducing service interruption risks.
- **Data Consistency**: Distributed authentication based on group replication technology, ensuring data consistency.
- **Automated Management**: Implementing full lifecycle management of cluster deployment, configuration updates, backup recovery, etc., through Operator, systematically reducing operational costs.
- **Elastic Scaling**: Supporting online addition or removal of nodes, automatic global transaction state synchronization, adapting to dynamic business load changes.
- **Read-Write Separation**: Supporting read-write separation through MySQL Router

Usage Limitations

- Group Member Count: The maximum number of MySQL replication group members is 9. At least 3 members are recommended to ensure high availability.
- InnoDB Engine: Relies on the InnoDB storage engine; it is recommended to configure `disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"` to disable unsupported storage engines.
- Explicit Primary Key: Each table to be replicated must have a defined primary key or an equivalent primary key, where equivalents are non-null unique keys. Configure `sql_generate_invisible_primary_key: ON` to support tables without explicitly defined primary keys.
- Network Quality: Designed to be deployed in cluster environments where server instances are located very close to each other, resulting in low network latency. High network latency can lead to replication delays or even replication failures.
- Isolation Level Restrictions: It is recommended to use the READ COMMITTED isolation level; the SERIALIZABLE isolation level is not supported.
- Foreign Key Restrictions: Cascading foreign keys are not recommended in multi-master mode.
- Multi-Master Deadlocks: In multi-master mode, deadlocks may occur when using `SELECT .. FOR UPDATE` statements because locks are not shared among members.
- Concurrent DDL and DML Conflicts: In multi-master mode, DDL and DML operations may cause conflicts; it is recommended to avoid concurrent DDL and DML operations.
- Transaction Size Limitations: Ensure that the transaction size is within the group_replication_transaction_size_limit range (default approximately 143 MB) and can be completed within the group_replication_member_expel_timeout period (default 5s).
- Heterogeneous Architecture Limitations: Due to limitations of the MySQL Clone plugin, heterogeneous scenarios are currently not supported.
- Rolling Update Limitations: MySQL Pods will prioritize restarting secondary members first, with the primary member restarted last. This process may cause brief connection interruptions, requiring application-level retry mechanisms.

Release Notes

TOC

v4.1.3

New and Optimized Features

Improved stability and fixed several security issues.

Fixed Issues

- Previously, MySQL-MGR instances occasionally experienced leader election failures during non-standard group leave/join operations, resulting in the cluster's inability to self-heal. This issue has been resolved in Alauda Database Service for MySQL v4.1.3.

Lifecycle Policy

TOC

Version Lifecycle Timeline

Below is the lifecycle schedule for released versions of the Alauda Database Service for MySQL-MGR:

Alauda Database Service for MySQL-MGR Version	Release Date	End of Support
v4.1.x	2025-08-18	Supported util 1 month post-release of the next version

Release Policy

The Alauda Database Service for MySQL-MGR follows a **rolling support model**:

- The life cycles of different major versions are independent of each other, and there may be multiple major versions in maintenance simultaneously.

- Each minor version remains supported **until the next minor version is officially released plus 1 month**. This is done by releasing patches that apply to each supported minor version.

Maintenance Policy

Alauda provides maintenance support for the Alauda Database Service for MySQL-MGR based on its rolling support model.

During the support period of each version, Alauda provides the following services:

- **Security Updates**: Regular patches and updates in line with Alauda's security standards.
- **Bug Fixes**: Timely fixes for critical and major issues, aligned with upstream MySQL community fixes.
- **Upgrade Assistance**: Guidance and support for smooth upgrades between the Alauda Database Service for MySQL-MGR versions.
- **Regular Maintenance**: Periodic updates to ensure the operator remains stable and compatible with the Kubernetes ecosystem.

Install

TOC

Installing Alauda Container Platform Data Services Essentials

Optional Installation

The plugin serves as a foundational component for Alauda Data Services web console. It is not necessary to install it if the **web console** is not required.

Prerequisites

1. Download Alauda Container Platform Data Services Essentials installation package that matches the platform.
2. Use the platform's application publishing capability to publish Alauda Container Platform Data Services Essentials to the `global` cluster.

Procedure

Web Console

1. In the left navigation bar, go to **Marketplace** > **Cluster Plugins**.
2. Select the `global` cluster.
3. Click the action button next to **Alauda Container Platform Data Services Essentials** plugin > **Install**.

Installing Alauda Container Platform Data Services RDS Framework

Optional Installation

The plugin serves as a foundational component for Alauda Data Services web console. It is not necessary to install it if the web console is not required.

Prerequisites

1. Download Alauda Container Platform Data Services RDS Framework installation package that matches the platform.
2. Use the platform's application publishing capability to publish Alauda Container Platform Data Services RDS Framework to any cluster where Data Services view are desired.

Procedure

Web Console

1. Log in to the platform and navigate to the **Administrator** page.
2. In the left navigation bar, select **Marketplace** -> **OperatorHub** to enter the **OperatorHub** page.
3. Find **Alauda Container Platform Data Services RDS Framework**, click **Install** to access the installation page.

Configuration Parameters:

Parameter	Recommended Configuration
Channel	The default Channel is **stable**.
Version	Select the desired version to install.
Installation Mode	**Cluster**: All namespaces under the cluster share one Operator for creating and managing instances, resulting in lower resource usage.
Installation Location	Choose the **Recommended**: If none exists, it will be created automatically.
Upgrade Strategy	**Manual**: When a new version is available in the OperatorHub, manual confirmation is required to upgrade the Operator to the latest version.

4. On the **Install Operator** page, select **Default Configuration**, then click **Install** to complete the installation of **Alauda Container Platform Data Services RDS Framework**.

Installing Alauda Database for MySQL

Prerequisites

1. Download the plugin installation package that matches the platform.
2. Use the platform's application publishing capability to publish the plugin to any cluster where Alauda Database for MySQL capabilities are desired.

Procedure

Web Console

1. Log in to the platform and navigate to the **Administrator** page.
2. In the left navigation bar, select **Marketplace** -> **OperatorHub** to enter the **OperatorHub** page.
3. Find **Alauda Database Service for MySQL**, click **Install** to access the installation page.

Configuration Parameters:

Parameter	Recommended Configuration
Channel	The default Channel is **alpha**.
Version	Select the desired version to install.
Installation Mode	**Cluster**: All namespaces under the cluster share one Operator for creating and managing instances, resulting in lower resource usage.
Installation Location	Choose the **Recommended**: If none exists, it will be created automatically.
Upgrade Strategy	**Manual**: When a new version is available in the OperatorHub, manual confirmation is required to upgrade the Operator to the latest version.

4. On the **Install Operator** page, select **Default Configuration**, then click **Install** to complete the installation.

Upgrade

> **NOTE**
>
> This document provides the upgrade path principles and supported version compatibility for Alauda Database Service for MySQL-MGR.

TOC

Compatibility Matrix

The table below lists supported versions of the Alauda Database Service for MySQL-MGR and its key components:

Alauda MySQL-MGR package version	MySQL server versions	Kubernetes Versions
v4.1.x	8.0	1.25+

Alauda MySQL-MGR package version	MySQL server versions	Kubernetes Versions
v4.0.x	8.0	1.25+

Please refer to Release Notes for version-specific changes, new features and deprecations information.

Prerequisites

Before initiating an upgrade, please ensure the following:

1. **Version Compatibility**: The current server version can be supported by the upgrade target version of the operator.
2. **Component Health**: MySQL-MGR instances are in a `Ready` state.
3. **Resource Availability**: The cluster has sufficient resources to support the upgrade process.

Upgrade Path Guidelines

Consecutive Minor Upgrade

- **Description**: Upgrade step-by-step through consecutive minor versions.
- **Example**: `4.1.x → 4.2.x → 4.3.x`

Patch-Level Upgrade

- **Description**: Any upgrade between patch versions of the same minor version are supported.
- **Example**: `4.1.3 → 4.1.7`

Upgrade Strategy

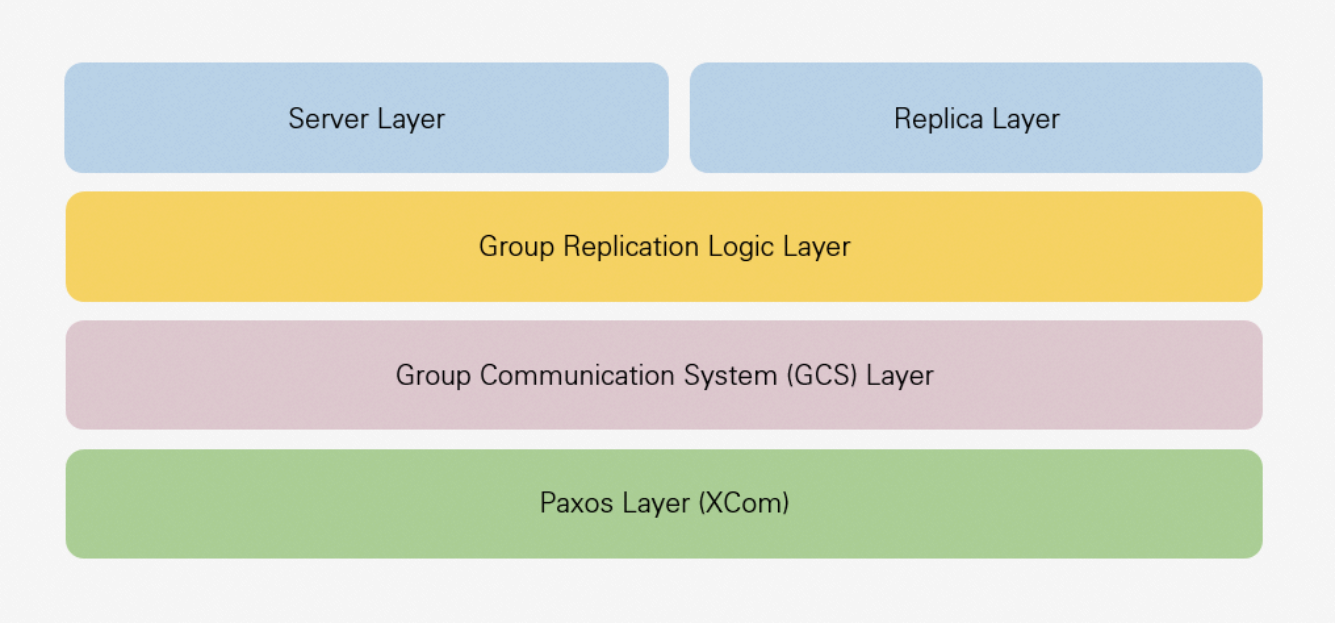
Alauda Database Service for MySQL-MGR will execute upgrades based on the configured upgrade strategy:

- **Automatic** : Auto-upgrades are triggered immediately upon detecting new component versions.
- **Manual** : Requires manual approval before initiating the upgrade process.

Architecture

TOC

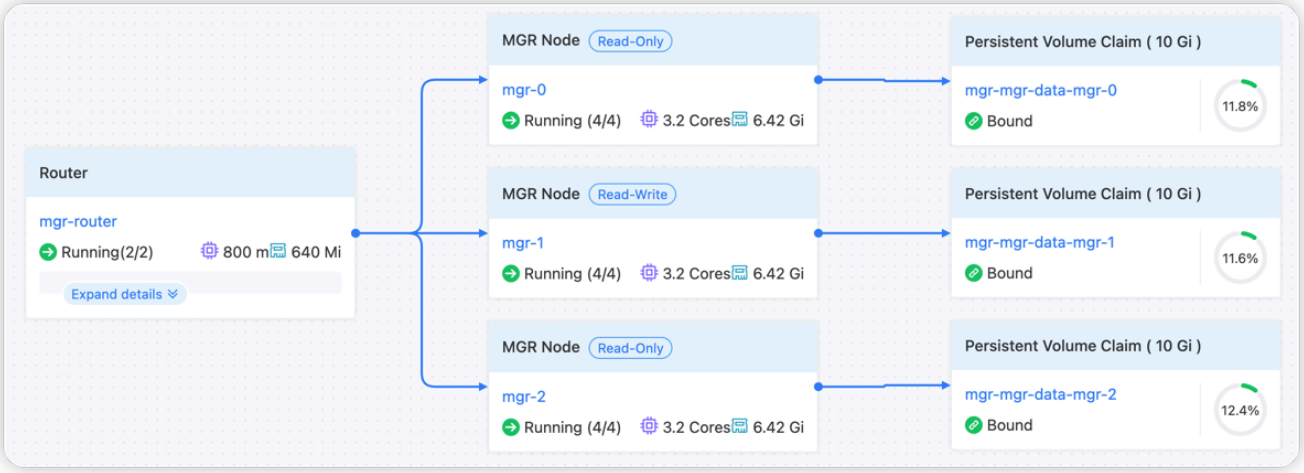
MGR Architecture



Under the Server layer and Replica layer of MySQL, group replication is divided into three layers:

- Group Replication Logic Layer: Responsible for interacting with the Server layer, sending, receiving, and replaying transactions to the Group Communication System Layer.
- Group Communication System Layer: Responsible for message passing, fault detection, and cluster member management.
- Paxos Layer: Implemented based on the Paxos protocol, ensuring data order consistency and majority availability.

MGR Instance Deployment Architecture



Core Components

- Deploys MySQL members managing MGR as StatefulSets, achieving multi-master replication and high availability through Group Replication.
- Deploys MySQL Router as Deployments that connect to MySQL members in the StatefulSet and provide external read-write separation service capabilities.
- Manages MySQL data storage via PVC to ensure data persistence.
- Provides read-write separation services through two different Services: read-write and read-only.

Data Flow

1. Client requests are accessed through the MySQL Router's Service.
2. The Router routes requests to the appropriate MySQL node based on the request type.

3. Write operations are synchronized to all nodes through Group Replication.

4. Read operations can be routed to any available node.

Guides

Create Instance

Function Overview

Prerequisites

Procedure

Delete Instance

Procedure

Access Met

Function Overv

Procedure

User Management

Stop & Start Instance

Feature Overview

Prerequisites

Notes

Procedure

Instance Restart

Procedure

Parameter

Query Analysis

Feature Introduction

Procedure

Monitoring

Function Overview

Scheduling

Precautions

Procedure

Create Instance

TOC

Function Overview

Create and manage MySQL-MGR database instances within a Kubernetes cluster. Quickly deploy a MySQL Group Replication cluster that meets your requirements by configuring resource specifications, parameter templates, account information, and other parameters.

Prerequisites

- Ensure that the MySQL-MGR Operator is installed in the cluster.
- Ensure that the cluster storage class supports dynamic volume provisioning (TopoLVM is recommended).
- Ensure that there are sufficient resource quotas.

Procedure

CLI

1. Create Password

```
kubectl -n ${namespace} create secret generic mgr-${instance_name}-pass
word \
--from-literal=clusterchecker={password} \
--from-literal=exporter=${password} \
--from-literal=manage=${password} \
--from-literal=root=${password}
```

Info

- `${instance_name}` is the instance name, used to identify the instance and should be unique.
- `${namespace}` is the namespace to which the instance belongs.
- `${password}` is the password, where different users can be set with distinct passwords.

2. Create Instance CR

```
kubectl apply -n $namespace -f - <<EOF
apiVersion: middleware.alauda.io/v1
kind: Mysql
metadata:
  labels:
    mysql/arch: mgr
  name: ${instance_name}
spec:
  mgr:
    enableStorage: true
    members: 3
    monitor:
      enable: true
    resources:
      server:
        limits:
          cpu: "2"
          memory: 4Gi
        requests:
          cpu: "2"
          memory: 4Gi
    router:
      replicas: 2
      resources:
        limits:
          cpu: 800m
          memory: 640Mi
        requests:
          cpu: 800m
          memory: 640Mi
      svcRO:
        type: ClusterIP
      svcRW:
        type: ClusterIP
    volumeClaimTemplate:
      spec:
        resources:
          requests:
            storage: 20Gi
        storageClassName: sc-topolvm
  params:
    mysql:
      mysqld:
```

```
binlog_expire_logs_seconds: "604800"
binlog_format: ROW
character_set_server: utf8mb4
default_storage_engine: InnoDB
default_time_zone: +08:00
disabled_storage_engines: MyISAM
event_scheduler: "ON"
general_log: "OFF"
innodb_adaptive_flushing: "ON"
innodb_adaptive_hash_index: "OFF"
innodb_autoinc_lock_mode: "2"
innodb_buffer_pool_chunk_size: "134217728"
innodb_buffer_pool_dump_at_shutdown: "ON"
innodb_buffer_pool_dump_pct: "25"
innodb_buffer_pool_instances: "2"
innodb_buffer_pool_load_abort: "OFF"
innodb_buffer_pool_load_at_startup: "ON"
innodb_buffer_pool_size: 1536M
innodb_deadlock_detect: "ON"
innodb_disable_sort_file_cache: "OFF"
innodb_fast_shutdown: "1"
innodb_file_per_table: "ON"
innodb_flush_log_at_trx_commit: "1"
innodb_flush_method: O_DIRECT_NO_FSYNC
innodb_flush_neighbors: "0"
innodb_io_capacity: "4000"
innodb_io_capacity_max: "4294967295"
innodb_max_purge_lag: "0"
innodb_max_undo_log_size: "1073741824"
innodb_online_alter_log_max_size: "134217728"
innodb_page_cleaners: "4"
innodb_purge_batch_size: "600"
innodb_purge_rseg_truncate_frequency: "128"
innodb_purge_threads: "4"
innodb_read_io_threads: "4"
innodb_redo_log_capacity: 1G
innodb_rollback_segments: "128"
innodb_strict_mode: "ON"
innodb_thread_concurrency: "5"
innodb_undo_log_truncate: "ON"
innodb_write_io_threads: "4"
interactive_timeout: "3600"
log_bin: bin
log_bin_trust_function_creators: "OFF"
```

```
        log_output: FILE
        loose_group_replication_communication_max_message_size: "104857
60"
        loose_group_replication_exit_state_action: READ_ONLY
        loose_group_replication_flow_control_applier_threshold: "1000"
        loose_group_replication_flow_control_certifier_threshold: "100
0"
        loose_group_replication_member_expel_timeout: "5"
        loose_group_replication_message_cache_size: 256M
        loose_group_replication_paxos_single_leader: "ON"
        loose_group_replication_poll_spin_loops: "0"
        loose_group_replication_transaction_size_limit: "150000000"
        loose_group_replication_unreachable_majority_timeout: "0"
        loose_group_replication_xcom_ssl_accept_retries: "10"
        max_allowed_packet: "67108864"
        max_binlog_size: "1073741824"
        max_connect_errors: "1000000"
        max_connections: "512"
        max_heap_table_size: 16M
        max_prepared_stmt_count: "8192"
        performance_schema: "ON"
        replica_parallel_type: LOGICAL_CLOCK
        replica_parallel_workers: "4"
        skip_external_locking: "ON"
        skip_name_resolve: "ON"
        sort_buffer_size: "262144"
        sql_generate_invisible_primary_key: "ON"
        sql_mode: NO_ENGINE_SUBSTITUTION
        sync_binlog: "1"
        table_definition_cache: "2000"
        table_open_cache_instances: "16"
        thread_handling: pool-of-threads
        thread_pool_max_threads: "128"
        thread_pool_oversubscribe: "32"
        thread_pool_size: "2"
        tmp_table_size: 16M
        wait_timeout: "3600"
    router:
      DEFAULT:
        max_total_connections: "300"
      logger:
        level: info
  upgradeOption:
    autoUpgrade: false
```

```
  version: "8.0"
EOF
```

3. Verify Instance Status

```
kubectl get mysql -n {namespace} {instance_name} -w
```

Web Console

1. Select **MySQL-MGR** from the left navigation bar.

2. Enter the target namespace.

3. Click **Create Instance**.

4. Configure Basic Parameters:

   - Instance Name: unique identifier
   - Architecture Configuration: Single Master (default) / Multi-Master

5. Resource Configuration:

   - Number of Replicas: recommended to start from 3 nodes
   - Specifications: CPU/Memory request and limit values for MySQL members
   - Storage Class: select a storage class of type TopoLVM
   - Storage Size: evaluate storage needs based on business requirements

6. Account Resources:

   - Configure root administrator password.

7. Connection Configuration:

   - Access Method: External Access / Internal Access

8. Click **Create** and wait for the status to change to Running.

Delete Instance

Resources to be Deleted

Deleting an instance will automatically remove all safely deletable resources created by the instance, such as StatefulSets, Deployments, ConfigMaps, Secrets, and Services. Special resources that may contain important data need to be deleted manually, such as PersistentVolumeClaims and backup resources associated with MySQL-MGR, specifically mysqlbackups.mysql.middleware.alauda.io and its related actual data.

TOC

Procedure

CLI

1. Delete the instance

```
kubectl delete mysql -n ${namespace} ${name}
```

2. Clean up persistent data

```
kubectl get pvc -n ${namespace} -l v1alpha1.mysql.middleware.alauda.io/
cluster=${name} -o name | xargs kubectl delete -n ${namespace}
```

Web Console

1. Select **MySQL-MGR** from the left navigation bar.
2. Enter the target namespace.
3. Click on ***instance name***.
4. Click **Actions** > **Delete**.
5. Choose deletion options:
   - Optional: Delete PersistentVolumeClaim
6. Confirm deletion:
   - Check associated resources
   - Enter the instance name for final deletion confirmation

Access Methods

TOC

Function Overview

Alauda Database for MySQL provides multiple ways to access MySQL-MGR clusters, supporting both internal and external cluster access. This section describes how to obtain appropriate connection addresses and establish connections to MySQL-MGR instances using the `mysql` client tool both inside and outside the cluster. For detailed connection examples of client libraries, please refer to How to Access MySQL-MGR Instances

Procedure

ClusterIP

CLI

1. Obtain the Service name: Based on the instance name, the corresponding Service names are `${name}-read-write` and `${name}-read-only`.

2. Connect using the MySQL Client. For example, to access the read/write service:

```
mysql -h${instance_name}-read-write.${namespace} -P3306 -u${username} -p${password}
```

3. View member roles:

```
select * from performance_schema.replication_group_members;
```

Web Console

1. Go to **Access Management** on the instance details page.

2. Obtain the access method:

   - Internal access to the cluster: Read/write or read-only internal routing; formats are `${instance name}-read-write` and `${instance name}-read-only`.

3. Obtain the account credentials.

4. Connecting using the `mysql` client tool from outside the cluster or via terminal console. For example, to access the read-write service:

```
mysql -h${instance_name}-read-write.${namespace} -P3306 -u${username} -p${password}
```

5. View member roles:

```
select * from performance_schema.replication_group_members;
```

NodePort

CLI

Confirm the deployed access mode

Ensure the field is configured as **NodePort**.

```
kubectl -n ${namespace} get mysql ${name} -o jsonpath='{.spec.mgr.router.svcRW.type}'
```

1. Obtain the NodePort access port:

```
kubectl get svc -n ${namespace} ${name}-read-write -o jsonpath='{.spec.ports[0].nodePort}'
```

2. Connect using the MySQL Client:

```
mysql -h${node_ip} -P${port} -u${username} -p${password}
```

3. View member roles:

```
select * from performance_schema.replication_group_members;
```

Web Console

Confirm the deployed access mode

Navigate to **Access Management** on the instance details page, and check for **External Access - via NodePort**.

1. Go to **Access Management** on the instance details page.

2. Choose the access method:

- External access - via NodePort: Read/write or read-only Router connection address.

3. Obtain the account credentials.

4. Connect using an external MySQL client tool or terminal console:

```
mysql -h${node_ip} -P${port} -u${username} -p${password}
```

5. View member roles:

```
select * from performance_schema.replication_group_members;
```

User Management

TOC

Feature Overview

Provides complete user lifecycle management, including creation, permission assignment, password modification, and deletion operations.

Procedure

CLI

1. Create User Password

```
kubectl -n ${namespace} create secret generic mgr-${instance_name}-${us
ername}-password \
--from-literal=host="%" \
--from-literal=user=${username} \
--from-literal=password=${password}
```

Info

- `${instance_name}` is the desired MySQL-MGR instance name for the user to be created
- `${namespace}` is the namespace to which the instance belongs
- `${username}` is the desired username to be created
- `${password}` is the desired password to be set

2. Create User Management CR

```
kubectl apply -n ${namespace} -f - <<EOF
apiVersion: middleware.alauda.io/v1
kind: MysqlUser
metadata:
  labels:
    mgr/cluster: mgr
    mysql/arch: mgr
  name: ${name}
  namespace: ${namespace}
spec:
  host: '%'
  mysql: ${instance_name}
  privileges:
    - grants:
        - SELECT
        - INSERT
        - UPDATE
        - DELETE
        - CREATE
        - DROP
        - REFERENCES
        - INDEX
        - ALTER
        - CREATE TEMPORARY TABLES
        - LOCK TABLES
        - CREATE VIEW
        - SHOW VIEW
        - CREATE ROUTINE
        - ALTER ROUTINE
        - EXECUTE
        - EVENT
        - TRIGGER
      targets:
        - ${database}.*
  secretName: ${secret_name}
  user: ${username}
EOF
```

Info

- `${instance_name}` is the desired MySQL-MGR instance name for the user to be created

- `${namespace}` is the namespace to which the instance belongs
- `${username}` is the desired username to be created
- `${secret_name}` is the secret name for carrying the password
- `${database}` is the desired database name for the user to be created

3. Log in to MySQL using the corresponding user

4. Check user permissions, example for user dev:

```
SHOW GRANTS;
```

Result will show:

```
+-----------------------------------------------+
| Grants for dev@%                              |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO `dev`@`%`               |
| GRANT ALL PRIVILEGES ON `dev`.* TO `dev`@`%` |
+-----------------------------------------------+
```

Web Console

1. Enter the instance **User Management** tab

2. Click **Create User**

3. Configure parameters:

- Username: conforms to naming conventions
- Password: meets complexity requirements
- Permissions: database-level permissions

4. Log in to MySQL using the corresponding user

5. Check user permissions, example for user dev:

```
SHOW GRANTS;
```

Result will show:

```
+-----------------------------------------------+
| Grants for dev@%                              |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO `dev`@`%`               |
| GRANT ALL PRIVILEGES ON `dev`.* TO `dev`@`%` |
+-----------------------------------------------+
```

Backup & Restore

TOC

Feature Introduction

Database backup and restore are critical functionalities to ensure data security and business continuity. By regularly backing up databases, it is possible to quickly recover data in the event of data loss, corruption, or inadvertent operations, thereby minimizing business interruptions and data loss.

Prerequisites:

Prior to backup, please prepare an external S3 compatible storage based on the volume of business data.

- External S3 Storage: S3 (Amazon Simple Storage Service) is an object storage service provided by Amazon. If you need to back up data to your own S3 bucket, please confirm with the administrator that the S3 storage has been integrated with your project.

Configure Storage Information

CLI

Create S3 secret

```
kubectl -n ${namespace} create secret generic ${secret_name} --from-literal=AWS_ACCESS_KEY_ID=${access_key} --from-literal=AWS_SECRET_ACCESS_KEY=${secret_key}
```

Web Console

Navigate to **Backup Center** -> **External S3 Storage** page, click create S3 storage, fill in the corresponding information, and create.

Ensure that the connectivity status of the external S3 storage shows as accessible.

Procedure

Configure Automatic Backup

CLI

Add automatic backup configuration information in the spec of the mysqlschedules resource.

```
kubectl patch mysqlschedules -n ${namespace} ${instance_name}-schedule --
type='merge' --patch '
spec:
  expiryDays: 7
  full:
    cron: 0 18 * * 6
    enable: true
  incr:
    cron: 0 19 * * 0,1,2,3,4,5
    enable: true
  storage:
    s3:
      bucket: ${bucket}
      endpoint: ${endpoint}
      region: ${region}
      secret:
        name: ${secretName}
'
```

Parameter	Description
expiryDays	Retention period for backup data
full.cron	Cron expression for full backup
incr.cron	Cron expression for incremental backup
storage.s3.bucket	Bucket name
storage.s3.endpoint	S3 storage endpoint
storage.s3.region	S3 storage region
storage.s3.secret.name	S3 storage secret name

Web Console

1. In the left navigation bar, click **MySQL-MGR**.

2. Click on ***Namespace Name***.

3. Click on ***Instance Name***.

4. In the **Backup & Restore** tab, click edit.

5. Enable **Automatic Backup** and set parameters.

   Note: The **Storage Location** is the S3 storage available for the current project.

6. Click **Update**.

Note:

- The timestamp in the automatic backup name is in UTC time (same as Controller Manager), indicating that this backup started executing at 18:00 on February 2, 2021, Beijing time.
- If it is time for automatic backup but the instance is not in the **Running** state, the platform will wait until the instance is ready to start the backup.

Create Manual Backup

Prerequisite

Ensure the instance status is **Running**

CLI

1. Create a data backup

```
kubectl -n ${namespace} create -f - <<EOF
apiVersion: mysql.middleware.alauda.io/v1
kind: MySQLBackup
metadata:
  name: ${backup_name}
  namespace: ${namespace}
spec:
  cluster:
    name: ${instance_name}
  storage:
    s3:
      bucket: ${bucket}
      endpoint: ${endpoint}
      region: ${region}
      secret:
        name: ${secret_name}
  type: ${type}
EOF
```

Parameter	Description
backup_name	Backup resource name
namespace	The namespace where the instance to be backed up resides
instance_name	The name of the instance to be backed up
storage.s3.bucket	Bucket name
storage.s3.endpoint	S3 storage endpoint
storage.s3.region	S3 storage region
storage.s3.secret.name	S3 storage secret name
type	Backup type: full or increment

2. Wait for the backup to complete and check the backup status

```
kubectl get MySQLBackup -n ${namespace} ${backup_name}
```

Web Console

1. In the left navigation bar, click **MySQL-MGR**.
2. Click on ***Namespace Name***.
3. Click on ***Instance Name***.
4. In the **Backup & Restore** tab, click **Create Backup**.
5. Set parameters and click **Create**. The backup duration depends on the data volume and network conditions, please be patient.

Restore Using Backup

CLI

1. Create MySQLRestore to restore data

```
kubectl -n ${namespace} create -f - <<EOF
apiVersion: middleware.alauda.io/v1
kind: MySQLRestore
metadata:
  labels:
    mysql/arch: mgr
  name: ${restore_name}
  namespace: ${namespace}
spec:
  mgr:
    backup:
      name: ${backup_name}
    cluster:
      name: ${target_instance_name}
    source:
      name: ${source_instance_name}
EOF
```

Parameter	Description
restore_name	Restore resource name
namespace	The namespace where the instance to be restored resides
backup_name	The full backup resource name to be used
target_instance_name	The name of the instance to restore data to
source_instance_name	The instance name from which the backup resource is sourced

2. Wait for the restore to complete and check the restore status

```
kubectl get MySQLRestore -n ${namespace} ${restore_name}
```

Web Console

1. In the left navigation bar, click **MySQL-MGR**.
2. Click on ***Namespace Name***.
3. Click on ***Instance Name***.
4. In the **Backup & Restore** tab, click **Database Restore**.
5. Select the target instance and click **Restore**. Please be patient until the restore record shows as **Restore Successful**.

Delete Backup

CLI

```
kubectl delete MySQLBackup -n ${namespace} ${backup_name}
```

Web Console

1. In the left navigation bar, click **MySQL-MGR**.

2. Click on ***Namespace Name***.

3. Click on ***Instance Name***.

4. In the **Backup & Restore** tab, click **Delete** and confirm.

   Deleting S3 backup data will also remove the data from S3 storage.

Stop & Start Instance

TOC

Feature Overview

Provides the temporary stop and restart functions for instances, applicable for resource scheduling, maintenance, cost-saving, and other scenarios. Stopping an instance will retain all configurations and data but release the compute resources.

Prerequisites

Stopping an instance requires the instance to be in **Processing**, **Running**, **Abnormal**, or **Unknown** status.

Starting an instance requires the instance to be in **Stopping** or **Stopped** status.

Notes

To ensure data safety, it is recommended to perform a full backup before stopping the instance.

1. Stopping an instance will terminate all database connections.
2. PVC containing MySQL data will be retained.
3. After restarting, it is necessary to verify the completeness of the MGR topology.

Procedure

CLI

Stop Instance

```
kubectl patch mysql ${name} -n ${namespace} --type='merge' -p '{"spec":{"pause":true}}'
```

Start Instance

```
kubectl patch mysql ${name} -n ${namespace} --type='merge' -p '{"spec":{"pause":false}}'
```

Web Console

1. Select **MySQL-MGR** from the left navigation bar.
2. Enter the target namespace.
3. Select the target instance.

Stop Instance

1. Click **Actions > Stop**.
2. Confirm the operation and wait for the status to change to "Stopped."

Start Instance

1. In the operation menu of the stopped instance, click **Start**.
2. Wait for the status to change to "Running."

Instance Restart

TOC

Procedure

CTL

Add an annotation to mark the instance for restart

```
kubectl annotate mysql ${name} -n ${namespace} "restartTime=$(date +%s)"
```

Web Console

1. Click on **MySQL-MGR** in the left navigation bar.
2. Click on ***namespace name***.
3. Click on ***instance name***.
4. Click **Actions** > **Restart**, and confirm.

Parameter Configuration

TOC

Overview

Configuring appropriate MySQL parameters based on business scenarios can optimize the database's performance, better supporting business operations. In the MySQL-MGR instance, system parameters provided by MySQL and Percona are used by default. You can conveniently configure parameters through the interface.

The platform provides parameter templates to define a set of desired MySQL parameters. When creating an instance, you can use parameter templates to batch modify the default parameters, and parameters not included in the template will retain their default values. Once the instance is running normally, MySQL parameters can be updated.

For more parameter information, please refer to the MySQL Official Documentation ↗ and Percona Official Documentation ↗.

Notes

Modifying parameter configurations will restart the Pods one by one, which may cause brief interruptions in business operations.

Procedure

CLI

```
kubectl patch mysql -n ${namespace} ${name} --type='merge' --patch '
spec:
  params:
    mysql:
      mysqld:
        max_connections: "1000"
        binlog_expire_logs_seconds: "604800"
'
```

You can modify the parameters by adjusting the spec.params.mysql in the CR, with the corresponding parameter keys being consistent with those in MySQL. As shown above, setting max_connections to 1000 and binlog_expire_logs_seconds to 604800, which equals 7 days.

Web Console

Setting Parameters Using Recommended Templates

Tip: Please refer to [Parameter Templates] to create parameter templates.

- The platform provides a variety of parameter templates for any MySQL-MGR instance in the cluster. Each specification of the parameter template is set with system parameters

such as `loose_group_replication_paxos_single_leader: ON` and `sql_generate_invisible_primary_key: ON` to enhance MySQL usability.

- The MySQL-MGR instance by default applies the parameter template named **ssd-2c4g-mysql-8.0-mgr**. When creating a MySQL-MGR instance, you can select the corresponding parameter template based on your storage medium and instance specifications.

Update Parameters

After updating the existing parameter configuration for an instance, the instance will automatically restart the MySQL replicas sequentially to apply the changes.

Procedure

1. Click **MySQL-MGR** in the left navigation bar.
2. Click ***Namespace Name***.
3. Click ***Instance Name***.
4. On the **Parameter Configuration** tab, click **Update Parameters**.
5. Configure the desired parameters, either individually or through bulk import.

   Note: Ensure that the imported parameters do not include unsupported MySQL parameters, as this could prevent the instance from starting.
6. Click **Update**.

Related Operations

If you want to reuse the parameter configuration of another instance, you can click to export parameters on the **Parameter Configuration** tab of the source instance, and refer to the parameter template to create a custom template for bulk parameter import.

Parameter Support Description

Parameter	default value
ssl_session_cache_mode	ON
ssl_session_cache_timeout	300
performance_schema	ON
performance_schema_instrument	
performance_schema_consumer_events_stages_current	OFF
performance_schema_consumer_events_stages_history	OFF
performance_schema_consumer_events_stages_history_long	OFF
performance_schema_consumer_events_statements_cpu	OFF
performance_schema_consumer_events_statements_current	ON
performance_schema_consumer_events_statements_history	ON
performance_schema_consumer_events_statements_history_long	OFF
performance_schema_consumer_events_transactions_current	ON
performance_schema_consumer_events_transactions_history	ON
performance_schema_consumer_events_transactions_history_long	OFF

Parameter	default value
performance_schema_consumer_events_waits_current	OFF
performance_schema_consumer_events_waits_history	OFF
performance_schema_consumer_events_waits_history_long	OFF
performance_schema_consumer_global_instrumentation	ON
performance_schema_consumer_thread_instrumentation	ON
performance_schema_consumer_statements_digest	ON
performance_schema_events_waits_history_long_size	-1
performance_schema_events_waits_history_size	-1
performance_schema_max_cond_classes	150
performance_schema_max_cond_instances	-1
performance_schema_max_program_instances	-1
performance_schema_max_prepared_statements_instances	-1
performance_schema_max_file_classes	80

Parameter	default value
performance_schema_max_file_handles	32768
performance_schema_max_file_instances	-1
performance_schema_max_socket_instances	-1
performance_schema_max_socket_classes	10
performance_schema_max_mutex_classes	350
performance_schema_max_mutex_instances	-1
performance_schema_max_rwlock_classes	60
performance_schema_max_rwlock_instances	-1
performance_schema_max_table_handles	-1
performance_schema_max_table_instances	-1
performance_schema_max_table_lock_stat	-1
performance_schema_max_index_stat	-1
performance_schema_max_thread_classes	100
performance_schema_max_thread_instances	-1

Parameter	default value
performance_schema_setup_actors_size	-1
performance_schema_setup_objects_size	-1
performance_schema_accounts_size	-1
performance_schema_hosts_size	-1
performance_schema_users_size	-1
performance_schema_max_stage_classes	175
performance_schema_events_stages_history_long_size	-1
performance_schema_events_stages_history_size	-1
performance_schema_max_statement_classes	219
performance_schema_events_statements_history_long_size	-1
performance_schema_events_statements_history_size	-1
performance_schema_max_statement_stack	10
performance_schema_max_memory_classes	450

Parameter	default value
performance_schema_digests_size	-1
performance_schema_events_transactions_history_long_size	-1
performance_schema_events_transactions_history_size	-1
performance_schema_max_digest_length	1024
performance_schema_max_digest_sample_age	60
performance_schema_session_connect_attrs_size	-1
performance_schema_max_metadata_locks	-1
performance_schema_max_sql_text_length	1024
performance_schema_error_size	5242
partial_revokes	OFF
log_error	
log_error_services	

Parameter	default value
log_timestamps	UTC
max_connections	151
max_prepared_stmt_count	16382
open_files_limit	0
table_definition_cache	400
table_open_cache	4000
table_open_cache_instances	16
persist_sensitive_variables_in_plaintext	ON
daemonize	OFF
skip_grant_tables	OFF
help	OFF
verbose	OFF

Parameter	default value
version	
initialize	OFF
initialize_insecure	OFF
keyring_migration_source	
keyring_migration_destination	
keyring_migration_user	
keyring_migration_host	
keyring_migration_password	
keyring_migration_socket	
keyring_migration_port	0
keyring_migration_to_component	OFF
no_dd_upgrade	OFF
validate_config	OFF
abort_slave_event_count	0

Parameter	default value
allow_suspicious_udfs	OFF
ansi	
autocommit	ON
binlog_do_db	
binlog_ignore_db	
character_set_client_handshake	ON
character_set_filesystem	
character_set_server	
chroot	
collation_server	
console	OFF
core_file	
default_storage_engine	
default_tmp_storage_engine	
default_time_zone	

Parameter	default value
disconnect_slave_event_count	0
exit_info	0
external_locking	OFF
gdb	OFF
super_large_pages	OFF
language	
lc_messages	
lc_time_names	
log_bin	
log_bin_index	
relay_log_index	
log_isam	
log_short_format	OFF

Parameter	default value
log_tc	
log_tc_size	24576
master_info_file	
master_retry_count	86400
max_binlog_dump_events	0
memlock	OFF
old_style_user_limits	OFF
port_open_timeout	0
replicate_do_db	

Parameter	default value
replicate_do_table	
replicate_ignore_db	
replicate_ignore_table	
replicate_rewrite_db	
replicate_same_server_id	OFF
replicate_wild_do_table	

Parameter	default value
replicate_wild_ignore_table	
safe_user_create	OFF
show_replica_auth_info	OFF
show_slave_auth_info	OFF
skip_host_cache	
skip_new	
skip_stack_trace	
sporadic_binlog_dump_fail	OFF
ssl	ON

Parameter	default value
admin_ssl	ON
symbolic_links	OFF
sysdate_is_now	OFF
tc_heuristic_recover	OFF
debug_sync_timeout	0
transaction_isolation	REPEATABLE-READ
transaction_read_only	OFF
user	
early_plugin_load	
plugin_load	
plugin_load_add	

Parameter	default value
innodb	
upgrade	AUTO
ssl_ca	
ssl_capath	
tls_version	
ssl_cert	
ssl_cipher	
tls_ciphersuites	
ssl_key	
ssl_crl	
ssl_crlpath	
admin_ssl_ca	

Parameter	default value
admin_ssl_capath	
admin_tls_version	
admin_ssl_cert	
admin_ssl_cipher	
admin_tls_ciphersuites	
admin_ssl_key	
admin_ssl_crl	
admin_ssl_crlpath	
performance_schema_show_processlist	OFF
auto_increment_increment	1
auto_increment_offset	1
windowing_use_high_precision	ON
cte_max_recursion_depth	1000
automatic_sp_privileges	ON

Parameter	default value
back_log	0
basedir	
default_authentication_plugin	
default_password_lifetime	0
bind_address	
admin_address	

Parameter	default value
admin_port	33062
create_admin_listener_thread	OFF
password_require_current	OFF
binlog_cache_size	32768
binlog_stmt_cache_size	32768
binlog_max_flush_queue_time	0
binlog_group_commit_sync_delay	0
binlog_group_commit_sync_no_delay_count	0

Parameter	default value
binlog_format	ROW
binlog_row_image	FULL
binlog_row_metadata	MINIMAL
binlog_transaction_compression	OFF
binlog_transaction_compression_level_zstd	3

Parameter	default value
session_track_gtids	OFF
binlog_direct_non_transactional_updates	OFF
explicit_defaults_for_timestamp	ON
master_info_repository	TABLE
relay_log_info_repository	TABLE
binlog_rows_query_log_events	OFF
binlog_order_commits	ON
bulk_insert_buffer_size	8388608
character_sets_dir	

Parameter	default value
select_into_buffer_size	131072
select_into_disk_sync	OFF
select_into_disk_sync_delay	0
completion_type	NO_CHAIN
concurrent_insert	AUTO
connect_timeout	10
information_schema_stats_expiry	86400
datadir	
debug	
delay_key_write	ON
delayed_insert_limit	100
delayed_insert_timeout	300

Parameter	default value
delayed_queue_size	1000
event_scheduler	ON
expire_logs_days	0
binlog_expire_logs_seconds	2592000
binlog_expire_logs_auto_purge	ON
flush	OFF
flush_time	0
ft_boolean_syntax	

Parameter	default value
ft_max_word_len	84
ft_min_word_len	4
ft_query_expansion_limit	20
ft_stopword_file	
init_connect	
init_file	
init_replica	
init_slave	
interactive_timeout	28800
join_buffer_size	262144
key_buffer_size	8388608
key_cache_block_size	1024
key_cache_division_limit	100
key_cache_age_threshold	300

Parameter	default value
large_pages	OFF
lc_messages_dir	
local_infile	OFF
lock_wait_timeout	31536000
transaction_write_set_extraction	XXHASH64
rpl_stop_replica_timeout	31536000
rpl_stop_slave_timeout	31536000
binlog_error_action	ABORT_SERVER
log_bin_trust_function_creators	OFF
check_proxy_users	OFF

Parameter	default value
mysql_native_password_proxy_users	OFF
sha256_password_proxy_users	OFF
log_bin_use_v1_row_events	OFF
log_error_suppression_list	
log_queries_not_using_indexes	OFF

Parameter	default value
log_slow_admin_statements	OFF
log_slow_replica_statements	OFF
log_slow_slave_statements	OFF
log_throttle_queries_not_using_indexes	0
log_error_verbosity	2
log_statements_unsafe_for_binlog	ON
long_query_time	10
low_priority_updates	OFF
lower_case_table_names	0
max_allowed_packet	67108864

Parameter	default value
replica_max_allowed_packet	1.07E+09
slave_max_allowed_packet	1.07E+09
max_binlog_cache_size	1.84E+19
max_binlog_stmt_cache_size	1.84E+19
max_binlog_size	1.07E+09
max_connect_errors	100
max_digest_length	1024
max_delayed_threads	20
max_error_count	1024
max_heap_table_size	16777216
max_join_size	1.84E+19
max_seeks_for_key	1.84E+19
max_length_for_sort_data	4096

Parameter	default value
max_points_in_geometry	65536
max_relay_log_size	0
max_sort_length	1024
max_sp_recursion_depth	0
max_user_connections	0
max_write_lock_count	1.84E+19
min_examined_row_limit	0
net_buffer_length	16384
net_read_timeout	30
net_write_timeout	60
net_retry_count	10
new	OFF
old	OFF
old_alter_table	OFF

Parameter	default value
optimizer_prune_level	1
optimizer_search_depth	62
optimizer_max_subgraph_pairs	100000
range_optimizer_max_mem_size	8388608
histogram_generation_max_mem_size	20000000
parser_max_mem_size	1.84E+19

Parameter	default value
optimizer_switch	on
global_connection_memory_limit	1.84E+19
connection_memory_limit	1.84E+19
connection_memory_chunk_size	8912
global_connection_memory_tracking	OFF
end_markers_in_json	OFF
optimizer_trace	
optimizer_trace_features	on

Parameter	default value
optimizer_trace_offset	-1
optimizer_trace_limit	1
optimizer_trace_max_mem_size	1048576
pid_file	
plugin_dir	
port	0
preload_buffer_size	32768
read_buffer_size	131072
require_secure_transport	OFF
read_only	OFF
super_read_only	OFF

Parameter	default value
read_rnd_buffer_size	262144
div_precision_increment	4
eq_range_index_dive_limit	200
range_alloc_block_size	4096
query_alloc_block_size	8192
query_prealloc_size	8192
skip_networking	OFF
skip_name_resolve	OFF
skip_show_database	OFF
socket	
thread_stack	1048576
tmpdir	

Parameter	default value
transaction_alloc_block_size	8192
transaction_prealloc_size	4096
thread_handling	one-thread-per-connection
secure_file_priv	
server_id	1
server_id_bits	32
regexp_time_limit	32
regexp_stack_limit	8000000
replica_compressed_protocol	OFF
slave_compressed_protocol	OFF
replica_exec_mode	STRICT
slave_exec_mode	STRICT

Parameter	default value
replica_type_conversions	
slave_type_conversions	
replica_sql_verify_checksum	ON
slave_sql_verify_checksum	ON
slave_rows_search_algorithms	HASH_SCAN
replica_parallel_type	LOGICAL_CLOCK

Parameter	default value
slave_parallel_type	LOGICAL_CLOCK
binlog_transaction_dependency_tracking	COMMIT_ORDER
binlog_transaction_dependency_history_size	25000
replica_preserve_commit_order	ON
slave_preserve_commit_order	ON
binlog_checksum	CRC32
source_verify_checksum	OFF
master_verify_checksum	OFF
slow_launch_time	2

Parameter	default value
sort_buffer_size	262144
sql_mode	NO_ENGINE_SUBSTITUTION
max_execution_time	0
ssl_fips_mode	OFF
auto_generate_certs	ON
updatable_views_with_limit	YES
schema_definition_cache	256
tablespace_definition_cache	256
stored_program_definition_cache	256
thread_cache_size	0
tmp_table_size	16777216

Parameter	default value
wait_timeout	28800
internal_tmp_mem_storage_engine	TempTable
temptable_max_ram	1.07E+09
temptable_max_mmap	1.07E+09
temptable_use_mmap	ON
big_tables	OFF
profiling_history_size	15
default_week_format	0
group_concat_max_len	1024
report_host	

Parameter	default value
report_user	
report_password	
report_port	0
keep_files_on_create	OFF
general_log_file	
slow_query_log_file	
general_log	OFF
log_raw	OFF
slow_query_log	OFF
log_slow_extra	OFF

Parameter	default value
log_output	FILE
log_replica_updates	ON
log_slave_updates	ON
relay_log	
relay_log_info_file	
relay_log_purge	ON
relay_log_recovery	OFF
rpl_read_size	8192
replica_allow_batching	ON
slave_allow_batching	ON
replica_load_tmpdir	

Parameter	default value
slave_load_tmpdir	
replica_net_timeout	60
slave_net_timeout	60
replica_skip_errors	
slave_skip_errors	
relay_log_space_limit	0
sync_relay_log	10000
sync_relay_log_info	10000
replica_checkpoint_period	300
slave_checkpoint_period	300
replica_checkpoint_group	512

Parameter	default value
slave_checkpoint_group	512
sync_binlog	1
sync_source_info	10000
sync_master_info	10000
replica_transaction_retries	10
slave_transaction_retries	10
replica_parallel_workers	4
slave_parallel_workers	4
replica_pending_jobs_size_max	1.34E+08
slave_pending_jobs_size_max	1.34E+08
host_cache_size	128

Parameter	default value
enforce_gtid_consistency	
binlog_gtid_simple_recovery	ON
stored_program_cache	256
gtid_mode	OFF

Parameter	default value
gtid_executed_compression_period	0
disconnect_on_expired_password	ON
validate_user_plugins	ON
block_encryption_mode	aes-128-ecb

Parameter	default value
session_track_system_variables	
session_track_schema	ON
session_track_transaction_info	OFF
session_track_state_change	OFF
offline_mode	OFF
avoid_temporal_upgrade	OFF
show_old_temporals	OFF
disabled_storage_engines	
persisted_globals_load	ON

Parameter	default value
mandatory_roles	
activate_all_roles_on_login	OFF
password_history	0
password_reuse_interval	0
binlog_row_value_options	
show_create_table_verbosity	OFF
secondary_engine_cost_threshold	100000
sql_require_primary_key	OFF

Parameter	default value
sql_generate_invisible_primary_key	OFF
show_gipk_in_create_table_and_information_schema	ON
persist_only_admin_x509_subject	
binlog_row_event_max_size	8192
group_replication_consistency	EVENTUAL
binlog_encryption	OFF
binlog_rotate_encryption_master_key_at_startup	OFF
default_table_encryption	OFF
table_encryption_privilege_check	OFF

Parameter	default value
print_identified_with_as_hex	OFF
generated_random_password_length	20
protocol_compression_algorithms	
replication_optimize_for_static_plugin_config	OFF
replication_sender_observe_commit_only	OFF
skip_replica_start	OFF

Parameter	default value
authentication_policy	
skip_slave_start	OFF
terminology_use_previous	NONE
xa_detach_on_prepare	ON

Parameter	default value
debug_sensitive_session_string	
explain_format	TRADITIONAL
sha256_password_private_key_path	
sha256_password_public_key_path	
sha256_password_auto_generate_rsa_keys	ON
caching_sha2_password_private_key_path	
caching_sha2_password_public_key_path	
caching_sha2_password_auto_generate_rsa_keys	ON
caching_sha2_password_digest_rounds	5000
innodb_api_trx_level	0
innodb_api_bk_commit_interval	5

Parameter	default value
innodb_autoextend_increment	64
innodb_dedicated_server	OFF
innodb_buffer_pool_size	1.34E+08
innodb_buffer_pool_chunk_size	1.34E+08
innodb_buffer_pool_instances	0
innodb_buffer_pool_filename	
innodb_buffer_pool_dump_now	OFF
innodb_buffer_pool_dump_at_shutdown	ON
innodb_buffer_pool_in_core_file	ON
innodb_buffer_pool_dump_pct	25

Parameter	default value
innodb_buffer_pool_evict	
innodb_buffer_pool_load_now	OFF
innodb_buffer_pool_load_abort	OFF
innodb_buffer_pool_load_at_startup	ON
innodb_lru_scan_depth	1024
innodb_flush_neighbors	0
innodb_checksum_algorithm	crc32
innodb_log_checksums	ON
innodb_commit_concurrency	0

Parameter	default value
innodb_concurrency_tickets	5000
innodb_compression_level	6
innodb_ddl_buffer_size	1048576
innodb_ddl_threads	4
innodb_data_file_path	
innodb_temp_data_file_path	
innodb_data_home_dir	
innodb_extend_and_initialize	ON
innodb_doublewrite	ON
innodb_doublewrite_dir	
innodb_doublewrite_batch_size	0
innodb_doublewrite_files	0
innodb_doublewrite_pages	0
innodb_stats_include_delete_marked	OFF
innodb_api_enable_binlog	OFF

Parameter	default value
innodb_api_enable_mdl	OFF
innodb_api_disable_rowlock	OFF
innodb_fast_shutdown	1
innodb_read_io_threads	4
innodb_write_io_threads	4
innodb_file_per_table	ON
innodb_flush_log_at_timeout	1
innodb_flush_log_at_trx_commit	1
innodb_flush_method	fsync
innodb_force_recovery	0
innodb_force_recovery_crash	0
innodb_fill_factor	100
innodb_ft_cache_size	8000000
innodb_ft_total_cache_size	6.4E+08
innodb_ft_result_cache_limit	2E+09

Parameter	default value
innodb_ft_enable_stopword	ON
innodb_ft_max_token_size	84
innodb_ft_min_token_size	3
innodb_ft_num_word_optimize	2000
innodb_ft_sort_pll_degree	2
innodb_force_load_corrupted	OFF
innodb_lock_wait_timeout	50
innodb_deadlock_detect	ON
innodb_page_size	16384
innodb_log_buffer_size	16777216
innodb_log_file_size	50331648
innodb_log_files_in_group	2
innodb_redo_log_capacity	1.05E+08

Parameter	default value
innodb_log_write_ahead_size	8192
innodb_log_group_home_dir	
innodb_log_writer_threads	ON
innodb_log_spin_cpu_abs_lwm	80
innodb_log_spin_cpu_pct_hwm	50
innodb_log_wait_for_flush_spin_hwm	400
innodb_log_compressed_pages	ON
innodb_max_dirty_pages_pct	90
innodb_max_dirty_pages_pct_lwm	10
innodb_adaptive_flushing_lwm	10
innodb_adaptive_flushing	ON
innodb_flush_sync	ON

Parameter	default value
innodb_flushing_avg_loops	30
innodb_max_purge_lag	0
innodb_max_purge_lag_delay	0
innodb_old_blocks_pct	37
innodb_old_blocks_time	1000
innodb_open_files	0
innodb_optimize_fulltext_only	OFF
innodb_rollback_on_timeout	OFF
innodb_ft_aux_table	
innodb_ft_enable_diag_print	OFF
innodb_ft_server_stopword_table	
innodb_ft_user_stopword_table	
innodb_disable_sort_file_cache	OFF
innodb_stats_on_metadata	OFF

Parameter	default value
innodb_stats_transient_sample_pages	8
innodb_stats_persistent	ON
innodb_stats_persistent_sample_pages	20
innodb_stats_auto_recalc	ON
innodb_adaptive_hash_index	ON
innodb_adaptive_hash_index_parts	8
innodb_stats_method	nulls_equal
innodb_replication_delay	0
innodb_status_file	OFF
innodb_strict_mode	ON
innodb_sort_buffer_size	1048576

Parameter	default value
innodb_online_alter_log_max_size	1.34E+08
innodb_directories	
innodb_sync_spin_loops	30
innodb_spin_wait_delay	6
innodb_spin_wait_pause_multiplier	50
innodb_fsync_threshold	0
innodb_table_locks	ON
innodb_thread_concurrency	0
innodb_adaptive_max_sleep_delay	150000
innodb_thread_sleep_delay	10000
innodb_tmpdir	
innodb_autoinc_lock_mode	2

Parameter	default value
innodb_use_native_aio	ON
innodb_change_buffering	all
innodb_change_buffer_max_size	25
innodb_change_buffering_debug	0
innodb_disable_background_merge	OFF
innodb_random_read_ahead	OFF
innodb_read_ahead_threshold	56
innodb_read_only	OFF
innodb_io_capacity	200
innodb_io_capacity_max	4.29E+09
innodb_idle_flush_pct	100
innodb_page_cleaners	4
innodb_monitor_enable	
innodb_monitor_disable	

Parameter	default value
innodb_monitor_reset	
innodb_monitor_reset_all	
innodb_purge_threads	4
innodb_purge_batch_size	300
innodb_background_drop_list_empty	OFF
innodb_purge_run_now	OFF
innodb_purge_stop_now	OFF
innodb_log_flush_now	OFF
innodb_log_checkpoint_now	OFF
innodb_log_checkpoint_fuzzy_now	OFF
innodb_checkpoint_disabled	OFF
innodb_buf_flush_list_now	OFF
innodb_merge_threshold_set_all_debug	50
innodb_semaphore_wait_timeout_debug	600
innodb_page_hash_locks	16
innodb_validate_tablespace_paths	ON

Parameter	default value
innodb_use_fdatasync	OFF
innodb_status_output	OFF
innodb_status_output_locks	OFF
innodb_print_all_deadlocks	OFF
innodb_cmp_per_index_enabled	OFF
innodb_max_undo_log_size	1.07E+09
innodb_purge_rseg_truncate_frequency	128
innodb_undo_log_truncate	ON
innodb_undo_log_encrypt	OFF
innodb_rollback_segments	128
innodb_undo_directory	
innodb_temp_tablespaces_dir	
innodb_sync_array_size	1

Parameter	default value
innodb_compression_failure_threshold_pct	5
innodb_compression_pad_pct_max	50
innodb_default_row_format	dynamic
innodb_redo_log_archive_dirs	
innodb_redo_log_encrypt	OFF
innodb_print_ddl_logs	OFF
innodb_trx_rseg_n_slots_debug	0
innodb_limit_optimistic_insert_debug	0
innodb_trx_purge_view_update_only_debug	OFF
innodb_fil_make_page_dirty_debug	4.29E+09
innodb_saved_page_number_debug	0
innodb_compress_debug	none

Parameter	default value
innodb_page_cleaner_disabled_debug	OFF
innodb_dict_stats_disabled_debug	OFF
innodb_master_thread_disabled_debug	OFF
innodb_sync_debug	OFF
innodb_buffer_pool_debug	OFF
innodb_ddl_log_crash_reset_debug	OFF
innodb_interpreter	
innodb_interpreter_output	
innodb_parallel_read_threads	4
innodb_segment_reserve_factor	12.5
myisam_block_size	1024
myisam_data_pointer_size	6
myisam_max_sort_file_size	9.22E+18
myisam_recover_options	

Parameter	default value
myisam_sort_buffer_size	8388608
myisam_use_mmap	OFF
myisam_mmap_size	1.84E+19
myisam_stats_method	nulls_unequal
archive	ON
blackhole	ON
federated	OFF
ndbcluster	OFF
ndb_extra_logging	1
ndb_wait_connected	120
ndb_wait_setup	120

Parameter	default value
ndb_cluster_connection_pool	1
ndb_cluster_connection_pool_nodeids	
ndb_recv_thread_activation_threshold	8
ndb_recv_thread_cpu_mask	
ndb_report_thresh_binlog_mem_usage	10
ndb_report_thresh_binlog_epoch_slip	10
ndb_eventbuffer_max_alloc	0
ndb_eventbuffer_free_percent	20
ndb_log_update_as_write	ON

Parameter	default value
ndb_log_updated_only	ON
ndb_log_update_minimal	OFF
ndb_log_empty_update	OFF
ndb_log_orig	OFF
ndb_distribution	KEYHASH
ndb_autoincrement_prefetch_sz	512
ndb_force_send	ON
ndb_use_exact_count	OFF
ndb_use_transactions	ON

Parameter	default value
ndb_use_copying_alter_table	OFF
ndb_allow_copying_alter_table	ON
ndb_optimized_node_selection	3
ndb_batch_size	32768
ndb_replica_batch_size	2097152
ndb_optimization_delay	10
ndb_index_stat_enable	ON
ndb_index_stat_option	
ndb_log_bin	OFF
ndb_log_binlog_index	ON
ndb_log_apply_status	OFF
ndb_log_transaction_id	OFF
ndb_log_transaction_compression	OFF

Parameter	default value
ndb_log_transaction_compression_level_zstd	3
ndb_log_fail_terminate	OFF
ndb_clear_apply_status	ON
ndb_schema_dist_upgrade_allowed	ON
ndb_schema_dist_timeout	120
ndb_schema_dist_lock_wait_timeout	30
ndb_connectstring	
ndb_mgmd_host	
ndb_nodeid	0
ndb_blob_read_batch_bytes	65536

Parameter	default value
ndb_blob_write_batch_bytes	65536
ndb_replica_blob_write_batch_bytes	2097152
ndb_deferred_constraints	0
ndb_join_pushdown	ON
ndb_log_exclusive_reads	OFF
ndb_read_backup	ON
ndb_data_node_neighbour	0
ndb_fully_replicated	OFF
ndb_row_checksum	1
ndb_dbg_check_shares	0
ndb_show_foreign_key_mock_tables	OFF

Parameter	default value
ndb_slave_conflict_role	NONE
ndb_applier_conflict_role	NONE
ndb_default_column_format	FIXED
ndb_metadata_check	ON
ndb_metadata_check_interval	60
ndb_metadata_sync	OFF
ndb_applier_allow_skip_epoch	OFF
ndbinfo	OFF
ndbinfo_max_rows	10
ndbinfo_max_bytes	0

Parameter	default value
ndbinfo_show_hidden	OFF
ndb_transid_mysql_connection_map	OFF
ngram	ON
ngram_token_size	2
mysqlx_cache_cleaner	ON
mysqlx	ON
mysqlx_port	33060
mysqlx_max_connections	100
mysqlx_min_worker_threads	2
mysqlx_idle_worker_thread_timeout	60
mysqlx_max_allowed_packet	67108864
mysqlx_connect_timeout	30

Parameter	default value
mysqlx_ssl_key	
mysqlx_ssl_ca	
mysqlx_ssl_capath	
mysqlx_ssl_cert	
mysqlx_ssl_cipher	
mysqlx_ssl_crl	
mysqlx_ssl_crlpath	
mysqlx_socket	
mysqlx_bind_address	
mysqlx_port_open_timeout	0
mysqlx_wait_timeout	28800
mysqlx_interactive_timeout	28800
mysqlx_read_timeout	30
mysqlx_write_timeout	60
mysqlx_document_id_unique_prefix	0

Parameter	default value
mysqlx_enable_hello_notice	ON
mysqlx_compression_algorithms	ZSTD_STREAM
mysqlx_deflate_default_compression_level	3
mysqlx_lz4_default_compression_level	2
mysqlx_zstd_default_compression_level	3
mysqlx_deflate_max_client_compression_level	5
mysqlx_lz4_max_client_compression_level	8
mysqlx_zstd_max_client_compression_level	11
skip_external_locking	OFF
loose_group_replication_flow_control_certifier_threshold	25000
loose_group_replication_transaction_size_limit	1.07E+09
loose_group_replication_flow_control_applier_threshold	25000
loose_group_replication_communication_max_message_size	1.07E+09
loose_group_replication_paxos_single_leader	OFF

Parameter	default value
loose_group_replication_message_cache_size	1.07E+09

Update Specification

As data volume increases, it is crucial to expand the resources promptly before reaching the storage quota to ensure continuous availability of the service. It is important to expand storage in a timely manner before reaching the storage quota, to ensure the continuous availability of the business. The online expansion process will not affect the normal provision of services.

TOC

Restrictions

- When changing CPU and memory resources, please evaluate based on business requirements and resource availability to avoid expansion failures due to insufficient or excessive resources, which may lead to the instance being unable to start normally.
- Group Member Count: The maximum number of MySQL replication group members is 9. At least 3 members are recommended to ensure high availability.

Procedure

CLI

To change the instance specifications, you can control it through the field `spec.mgr.resources` (refer to the API documentation for details).

```
kubectl -n ${namespace} patch mysql ${instance_name} --type=merge --patch
='{"spec":{"mgr": {"resources":{"server":{"limits":{"cpu": "4","memor
y":"8Gi"},"requests":{"cpu":"4", "memory":"8Gi"}}}}}}'
```

Use the following command to expand the storage capacity:

```
kubectl patch mysql -n ${namespace} ${instance_name} --type='json' -p
'[{"op":"replace","path":"/spec/mgr/volumeClaimTemplate/spec/resources/re
quests/storage","value":"${new_storage}"}]'
```

Use the following command to check whether the storage capacity expansion was successful:

```
kubectl get pvc -n ${namespace} -l "v1alpha1.mysql.middleware.alauda.io/c
luster=${instance_name}"
```

Web Console

1. Click on **MySQL-MGR** in the left navigation bar.
2. Click on ***namespace name***.
3. Click on ***instance name***.
4. In the **Details** tab, click on **Update Specification** above the topology.
5. Fill in the desired replica count and new specifications for the corresponding resources.
6. In **Storage Expansion**, modify **New Capacity** to specify the desired increased volume.
7. Click **Update**.

After a moment, you will see the corresponding changes in the **Specifications** in the topology.

Query Analysis

TOC

Feature Introduction

By analyzing SQL query activity, precisely identify the SQL statements that affect database performance, or continuously monitor query activities and perform statement adjustments and optimizations.

Procedure

Web Console

1. In the left navigation bar, click on **MySQL-MGR**.

2. Click on ***Namespace Name***.

3. Click on ***Instance Name***.

4. In the **Query Analysis** tab, set the search criteria.

- To find statements executed within a specific time range: Select a time period at **Time Range**.
- To find specific statements: Enter the statement keyword at **Search by SQL Statement**, then click search.

5. Wait a moment and check the results.

Log

TOC

Feature Introduction

Understand the logs during the instance operation process from the container level, such as MySQL error logs. Proper use of logs can help you quickly locate issues and handle faults and exceptions.

Procedure

CLI

1. View all related pods of the instance, including the mysql pod and mysql-router pod.

```
kubectl get pod -n ${namespace} -l "middleware.instance/name=${instance_name}"
```

2. Query the logs for a specific pod.

```
kubectl logs -n ${namespace} ${pod_name}
```

Web Console

1. In the left navigation bar, click **MySQL-MGR**.

2. Click ***Name of the Namespace***.

3. Click ***Instance Name***.

4. In the **Realtime Log** tab, set the View option to view the corresponding log.

Monitoring

TOC

Function Overview

The monitoring data embedded in the panel can be utilized for MySQL monitoring and alerts from the perspectives of resources, performance, and capacity, and it supports configuring notification strategies.

The intuitively presented monitoring data can provide decision-making support for operations inspection or performance tuning.

Performance Monitoring Panel

The platform defaults to collecting common monitoring indicators related to MySQL's resources, performance, and capacity. In the instance's **Monitoring** tab, real-time monitoring data of the indicators can be viewed.

Indicator	Default Monitoring Indicators
Pod CPU Utilization	Pod's CPU usage. Pods with a CPU usage exceeding 100% will be ignored, because under normal circumstances the CPU usage will not exceed 100%.

Indicator	Default Monitoring Indicators
	If it exceeds 100%, please check whether all containers of the pod have set resource limits.
Pod Memory Utilization	Pod's memory usage. Pods with a memory usage exceeding 100% will be ignored, because under normal circumstances memory usage will not exceed 100%. If it exceeds 100%, please check whether all containers of the pod have set resource limits.
Network Received	The average bytes per second received from the network in the past 5 minutes
Network Transmitted	The average bytes per second sent onto the network in the last 5 minutes
PVC Used	PVC storage usage.
MySQL Connections	Max Connections is the maximum permitted number of simultaneous client connections. By default, this is 151. Increasing this value increases the number of file descriptors that mysqld requires.
MySQL Client Thread Activity	Threads Connected is the number of open connections, while Threads Running is the number of threads that are not sleeping.
MySQL Table Locks	MySQL takes a number of different locks for varying reasons. In this graph, we see how many table-level locks MySQL has requested from the storage engine.
MySQL Sorts	This graph also shows when sorts had to scan a whole table or a specified range of a table in order to return the results that could not have been sorted via an index.
MySQL Slow Queries	Slow queries are defined as queries that take longer than the long_query_time setting.
InnoDB Buffer Pool Requests	InnoDB Buffer Pool Requests refer to the total number of read and write requests made to the InnoDB buffer pool.
TPS	Transactions Per Second (TPS) is a key performance metric that measures the number of transactions MySQL processes per second.

Indicator	Default Monitoring Indicators
Commands	Commands refer to the number of SQL statements executed by MySQL, including queries, inserts, updates, deletes, and other database operations.

Scheduling Configuration

When the node taints change, adding, modifying, or deleting Pod tolerations is supported. Updating the scheduling configuration will not affect the normal provision of services.

TOC

Precautions

- Pod migration is not supported. After changing the Pod tolerations, the selected nodes must include the nodes that were present before the update.
- If you have configured node labels, the number of nodes matching the Pod tolerations and node taints must be greater than or equal to the number of replicas. Additionally, the changed available nodes must either match the previously available nodes or include new nodes added to the original available nodes for the scheduling configuration to update correctly.

Procedure

Web Console

1. In the left navigation bar, click on **MySQL-MGR**.

2. Click on ***namespace name***.

3. Click on ***instance name***.

4. Click on the **Actions** button in the upper right corner > **Update Scheduling Configuration**.

5. Modify the configuration as needed.

6. Click **Update**.

HowTo

Patch Version Upgrade

- Setting Up Patch Version Upgrade Strateg
- Patch Version Upgrade
- Bulk Patch Version Upgrade

How To Access MySQL-MGR Instances

- Applicable scenarios
- Common client programs

Patch Version Upgrade

With the release of vulnerability fixes or new features, Patch version upgrades can occur for the middleware. During patch version upgrades, rolling updates will be triggered. It is recommended to choose an appropriate time window for the upgrade.. When a Patch version is available for upgrade, it is advised to upgrade as soon as possible to apply the latest fixes to the components.

TOC

Setting Up Patch Version Upgrade Strategy

Procedure

Web Console

1. Go to the instance detail page.
2. Click on **Patch Version Upgrade Strategy**.
3. Select **Manual** or **Automatic**.
4. Click on **Update** to complete the setting.

Patch Version Upgrade

Prerequisites

When a Patch version is available, the instance list page will display an upgrade icon, and the instance detail page will show an upgrade prompt.

Procedure

Web Console

1. Go to the instance detail page.
2. Click on **Upgrade Now** next to the upgrade prompt.
3. Read the upgrade prompt to understand the changes in versions of components and plugins.
4. Click on **Upgrade Now**.
5. Wait for the instance status to change to running, indicating a successful upgrade.

Bulk Patch Version Upgrade

Prerequisites

Users must have platform administrator permissions.

Procedure

Web Console

1. In the left navigation bar, click on **Upgrade Management**.
2. Select the required **operator** to enter the operator detail page.
3. Select the instances to be upgraded and click on **Bulk Upgrade**.
4. Read the upgrade prompt to understand the changes in versions of components and plugins.
5. Click on **Upgrade Now**.
6. Wait for the instance status to change to running, indicating a successful upgrade.

How To Access MySQL-MGR Instances

There are multiple ways to connect to a MySQL-MGR instance when accessing it from a business application. This section describes how to choose a connection address to access the MySQL-MGR instance and provides examples for demonstration.

TOC

Applicable scenarios

Access Method	Applicable Scenarios
Access within the Cluster	In single-master or multi-master mode, applications within the cluster access the MGR instance through the port exposed by the SVC.
Access from outside the Cluster	In single-master or multi-master mode, applications outside the cluster access the MGR instance through the NodePort port.

Common client programs

The example demonstrates programs using JDBC and GORM clients. Please refer to the corresponding framework documentation for other clients.

When using the code, please modify the relevant variables to your actual values. The variable descriptions are as follows:

Variable	Description
host	The IP or SVC name of the host for the corresponding access method
port	The port number for the corresponding access method
user	The username for MySQL-MGR
password	The actual password for the MySQL-MGR user
database	The name of the database to be accessed

JDBC

1. Cluster Internal Access

```
import java.sql.*;
public class Main {
    static final String DB_URL = "jdbc:mysql://mgr-0:3306,mgr-1:3306,mg
r-2:3306/mysql";
    static final String USER = "root";
    static final String PASS = "123456";
    public static void main(String[] args) {
        try {
            Connection conn = DriverManager.getConnection(DB_URL,USER,P
ASS);
            Statement statement = conn.createStatement();
        }catch(SQLException se){
            se.printStackTrace();
        }
    }
}
```

2. External Access to the Cluster - via NodePort

```
import java.sql.*;
public class Main {
    static final String DB_URL = "jdbc:mysql://192.168.1.100:32630,192.
168.1.101:32630,192.168.1.102:32630/mysql";
    static final String USER = "root";
    static final String PASS = "123456";
    public static void main(String[] args) {
        try {
            Connection conn = DriverManager.getConnection(DB_URL,USER,P
ASS);
            Statement statement = conn.createStatement();
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
}
```

GORM

1. Cluster Internal Access

```
import (
    "gorm.io/driver/mysql"
    "gorm.io/gorm"
    "gorm.io/plugin/dbresolver"
)

func main() {
    writeDSN:="root:123456@tcp(mgr-read-write:32631)/mysql"
    readDSN := "root:123456@tcp(mgr-read-only:32630)/mysql"
    db, err := gorm.Open(mysql.Open(writeDSN), &gorm.Config{})
    if err != nil {
        panic(err)
    }
    err = db.Use(dbresolver.Register(dbresolver.Config{
        Replicas:          []gorm.Dialector{mysql.Open(readDSN)},
        Policy:            dbresolver.RandomPolicy{},
        TraceResolverMode: true,
    }))
    if err != nil {
        panic(err)
    }
}
```

2. External Access to the Cluster - via NodePort

```go
package main

import (
    "gorm.io/driver/mysql"
    "gorm.io/gorm"
    "gorm.io/plugin/dbresolver"
)

func main() {
    writeDSNs := []string{
        "root:123456@tcp(192.168.18.130:32631)/mysql",
        "root:123456@tcp(192.168.18.131:32631)/mysql",
        "root:123456@tcp(192.168.18.132:32631)/mysql",
    }
    readDSNs := []string{
        "root:123456@tcp(192.168.18.131:32630)/mysql",
        "root:123456@tcp(192.168.18.132:32630)/mysql",
        "root:123456@tcp(192.168.18.133:32630)/mysql",
    }
    db, err := gorm.Open(mysql.Open(writeDSNs[0]), &gorm.Config{})
    if err != nil {
        panic(err)
    }
    sources := make([]gorm.Dialector, 0)
    replicas := make([]gorm.Dialector, 0)
    for _, dsn := range writeDSNs {
        sources = append(sources, mysql.Open(dsn))
    }
    for _, dsn := range readDSNs {
        replicas = append(replicas, mysql.Open(dsn))
    }
    err = db.Use(dbresolver.Register(dbresolver.Config{
        Sources:           sources,
        Replicas:          replicas,
        Policy:            dbresolver.RandomPolicy{},
        TraceResolverMode: true,
    }))
    if err != nil {
        panic(err)
    }
}
```