How to Maintain invoicingdb

Rhyannon Joy Rodriguez

Summary

This documentation guides the team through database access and maintenance following the shutdown of the corp replica. We're transitioning to a new model where analytical data comes from the data lake and operational data comes from primary instances via dedicated service accounts. To access production data now, users must temporarily enable self-service grants in our schema management tool, request permissions, use the read-only endpoint for queries, then disable self-service grants. Making database changes requires MySQL 8, AWS CLI, credentials from Kion and AWS, and *critically*, creating snapshots before changes and communicating all work in the team Slack channel. The goal is to restrict production access to *services only* while engineers work primarily in the QA environment. **All proprietary information and links have been removed*.

Frequently Asked Questions

Analytical vs Operational Data and Corp Replica vs Primary Instance

Financial Systems is moving towards the analytical-operational permissions structure currently being enforced for centraldb. Analytical data, like that used for financial reporting, should be sourced from our data lake. Previously requests for analytical data were being routed to our corp replica, but we've turned-off our corp replica, **Generally, corp replicas**:

- Are *read-only* clusters that replicate themselves from a primary cluster using low-level binlog replication
- Are dedicated to *read-only operations by humans* for analytical purposes.
- They should not be used by non-humans, such as daemons or cronjobs
- No production application and or business critical solution should rely on a corp replica
- *Read/write operations should not* be performed against them, as this will break the replication and require the replica to be rebuilt
- Due to the large number of user grants, restoring grants takes a long time and rebuilding can take hours
- Are only supported by us as "best effort" and do not come with SLOs or any other guarantees. We expect them to break and have issues often, and they do.

- We do not recommend or support any important work to be done using them. If a team's capability or process requires a guarantee of availability, it eventually needs to be migrated to a reliable data source, the primary instance, or our data lake

Operational data should be sourced from the primary instance through service accounts. We want to move away from shared service accounts. Service accounts should not be used by multiple applications.For example, we eventually want to remove the money user. Service account requests should include:

- Application or service
- Permissions requested (read or read/write)
- Tables it needs to access along with the permissions (select, insert, update, delete)
- For restricted tables, follow the steps from article "Requesting Access to Restricted Tables"

How do we access our data in PROD now that we don't have the corp replica?

Temporarily set our invoicingdb schema in to accept self-service grants, grant yourself access, reset your password, then **turn-off self-service grants**.

1. Navigate to Schema Management and add invoicingdb to the Schema Name bar
2. Check Allow Self-Service Grants and Save
    - Expect to see "*The schema details have been saved*"
3. Navigate to Permissions → Request (MySQL), add add invoicingdb to the Schema Name bar
4. Select Request Permissions
    - Notice that the host field lists the database cluster **write** endpoint
        - The **only** endpoint onboarded is the **write** endpoint
            1. Use KION → AWS if you require additional details about this internal database's endpoints
    - Expect to see "*Permissions successfully granted*"
5. Navigate to Credentials → Reset Password
    - You may choose to set your own or allow our tools to generate one for you
    - Select Reset Password
    - Use your new password to set-up a new data source in your query tool of choice
6. When setting your datasource properties (in DataGrip or otherwise), **please use** the **read-only** endpoint

- "**ro**" for **read-only**

```
prodinvoicing db.cluster-ro-cr2gqaomp8yh.us-east-2.rds.amazonaws.com
```

- Write access is for services or specific database changes (like adjusting grants)
  - In addition, binary logs (binlogs) are only accessible with specific grants and the write endpoint
  - Read access is for some services and Invoicing engineers
- Previously, the corp-replica was for financial systems analysts and reporters

7. **Do not forget** to turn off self-service grants
   - Return to Schema Management
   - Uncheck Allow Self-Service Grants
   - Select Save
8. *If you attempt to run a query in PROD* and see

```
Connections using insecure transport are prohibited while
--require_secure_transport=ON
```

- Navigate to your data source, right-click and select Properties
  - Navigate to the SSH/SSL tab
  - Check Use SSL box
    1. Select Test Connection
    2. If successful, select Apply then OK

How do we verify the retention period for our database transaction logs (binary logs/"binlogs")?

Binlogs are not enabled by default for clusters. This is something that each database owner should configure based on their specific requirements. For example, we currently do not have retention periods set for our QA database invoicingdbtest binlogs. You must have credentials to access the PROD database invoicingdb, if you don't, see the above section "*How do we access…*" You must have the proper permissions to view binlogs, otherwise you may see `ACCESS DENIED`. If you're conducting a task that requires viewing binlogs, you'll need cluster admin credentials and the write endpoint must be in use. Service users require `REPLICATION CLIENT` privilege to view binlogs, while individual users have not been granted binlog access at this time. Observe that we have enabled binlogs by setting cluster parameters in the our Terraform `invoicingdb.tf` file and in particular these variables:

```
{
     name         = "binlog_format"
     value        = "ROW"
     apply_method = "pending-reboot"
   },
   {
     name         = "enforce_gtid_consistency"
     value        = "ON"
     apply_method = "pending-reboot"
   },
   {
     name         = "gtid-mode"
     value        = "ON"
     apply_method = "pending-reboot"
   }
```

With the correct permissions while using the write endpoint, you should be able to see that with `SHOW VARIABLES LIKE 'log_bin';` the `binlog_backup` is set `ON`. In addition, you should be able to see a list of logs with the command `show binary logs;` in your DataGrip query console (or the tool of your choice). The retention period is configured in the AWS CLI. Verify the configuration with the following:

```
CALL mysql.rds_show_configuration;
```

You may set the retention periods with the following:

```
CALL mysql.rds_set_configuration('binlog retention hours', <some integer
```

```
like 24>);
```

What & Why: Historical Context

Our tables from moneydb were migrated to our own prodinvoicing cluster, but keeping the name moneydb limited our use of our internal privilege management system. We have migrated these tables to invoicingdb, so that we can manage permissions and use our new internal DML and DDL tools.

Our data team is reworking their documentation, but if a doc has a Verified badge, it should be safe to follow. Reference the documentation in the MySQL documentation hub before heading to the Slack channel **#help-mysql**.

During the second migration, existing permissions were copied from moneydb, so your individual permissions should not have been interrupted to moneydb. You'll notice from the MySQL documentation hub that using Terraform is the recommended path to managing new databases, but it is more complex to retroactively implement to an existing environment. We can use AWS CLI for our new databases for select circumstances until advised otherwise.

Once all of our internal databases are onboarded onto our internal permissions system, individuals should be able to self-serve grants. We should aim to limit traffic to the PROD database. Engineers should only be interacting with QA while PROD should only be for services. We have turned-off our corp replica. If for any reason you need admin credentials to make changes not possible through our tools, see the **How-to: Making Database Changes** section below.

db name	cluster	grants	aka
invoicingdbtest	qainvoivingdb	engineers & services	"QA"
invoicingdb	prodinvoicingco	engineers	"corp replica"

	*turned-off 03/25		
invoicingdb	prodinvoicingdb	services only	“PROD”

How-to: Making Database Changes

Document your work on a JIRA ticket. Communicate with the team anytime you’re making a database change so as to not accidentally implement a scream test. Test your changes in QA before moving to PROD. Request that a teammate verifies your changes. Use Slack channel **#invoicing-engineers**.

1. If you need to upgrade to MySQL 8
    - Not all of our databases have been upgraded yet and the `system-setup` script does not enforce MySQL 8.
    - Reference article “How to Upgrade MySQL 5.7 to 8.0 on Mac”
2. If you need to install AWS CLI
    - Reference article “How-to: Setup AWS CLI”
        - Should already be installed and up to date if you’re using the `system-setup` script
        - Check with `brew install awscli`
        - In order to verify your installation (if you installed), or test your connection you’re going to need the next step
3. Generate short-term AWS Credentials
    - Reference article “How do I generate short-lived AWS credentials to use?”
        - Log into Kion using Okta
        - Select the appropriate project, either financial-systems-qa or financial-systems-prod
        - To continue with “How-to: Setup AWS CLI” Step #4
            - Set your profile to `[default]`
            - You don’t need any profile or timezone info in your local `~/.aws/credentials` file
    - To verify that your access to AWS works, Step #7, remove the command’s profile flag:

```
aws ec2 describe-vpcs --dry-run
```

    - Short-term credentials expire after a certain number of hours, make sure they don't stick around somewhere on your local machine
4. Generate AWS Credentials for our clusters
    - Reference article "How do I retrieve the root credentials of my cluster?"
    - Generate cluster credentials with this AWS CLI command

```
aws secretsmanager get-secret-value --region ${us-east-2} --secret-id rds-admin-credentials-${qainvoicingdb}
```

    - Change the cluster to reflect the appropriate environment
        - QA cluster: `qainvoicingdb`
        - PROD cluster: `prodinvoicingdb`
            - Corp Replica cluster was `prodinvoicingco`
            - **Do not attempt to make changes on the replica, it was turned-off 03/25**
            - If we ever restore the corp replica, reference article "How do I rebuild a corp replica of a cluster?"
        - Reference articles "How do I access my cluster?" and or "How do I connect using DataGrip?"
    - Remember that these credentials **DO NOT EXPIRE** by default
        - Set your password to expire
        - Or, delete your admin console when you've completed your database task
5. If there is any chance that your changes could be destructive in PROD, restore a snapshot of the cluster before you begin your maintenance window:
    - Reference article "How do I restore a snapshot of my cluster?"
    - Reference AWS documentation "Restore-db-cluster-from-snapshot"
    - Snapshot commands from the second migration:

```
aws rds create-db-cluster-snapshot --region us-east-2 --db-cluster-identifier prodinvoicingdb --db-cluster-snapshot-identifier corp-replica-backup-20241021
```

6. Execute your changes in stages, verify QA before moving to PROD
7. Document and communicate any issues