Cloud SQL for MySQL disaster recovery: A complete failover and fallback process

Note: This document or section includes references to one or more terms that Google considers disrespectful or offensive. The terms are used because they are keywords in the software that's described in the document.

This tutorial describes a complete disaster recovery (DR) failover and fallback process in Cloud SQL for MySQL (https://cloud.google.com/sql/docs/mysql?authuser=0) by using cross-region read replicas (https://cloud.google.com/sql/docs/mysql/replication/cross-region-replicas?authuser=0).

In this tutorial, you set up a high availability (HA) Cloud SQL for MySQL instance for DR and simulate an outage. Then you step through the DR process to recover your initial deployment after the outage is resolved.

This tutorial is intended for database architects, administrators, and engineers.

To read an overview of how SQL disaster recovery works, see Introduction to Cloud SQL disaster recovery (https://cloud.google.com/architecture/intro-to-cloud-sql-disaster-recovery?authuser=0).

Objectives

- Create an HA Cloud SQL for MySQL instance.
- Deploy a cross-region read replica on Google Cloud using Cloud SQL for MySQL.
- Simulate a disaster and failover with Cloud SQL for MySQL.
- Understand the steps to recover your initial deployment by using a fallback with Cloud SQL for MySQL.

This document focuses only on cross-region DR failover and fallback processes. For information about a single-region HA failover process, see Overview of the high availability configuration (https://cloud.google.com/sql/docs/mysql/high-availability?authuser=0).

Costs

This tutorial uses the following billable components of Google Cloud:

- Cloud SQL (https://cloud.google.com/sql/pricing?authuser=0)

To generate a cost estimate based on your projected usage, use the pricing calculator (https://cloud.google.com/products/calculator?authuser=0).

When you finish this tutorial, you can avoid continued billing by deleting the resources you created. For more information, see Cleaning up (#clean-up).

Before you begin

1. In the Google Cloud Console, on the project selector page, select or create a Google Cloud project.

★ **Note**: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

Go to project selector (https://console.cloud.google.com/projectselector2/home/dashboard?authus

2. Make sure that billing is enabled for your Cloud project. Learn how to confirm that billing is enabled for your project (https://cloud.google.com/billing/docs/how-to/modify-project?authuser=0).
3. In the Cloud Console, activate Cloud Shell.

Activate Cloud Shell (https://console.cloud.google.com/?cloudshell=true&authuser=0)

Phase 1: Setting up an HA database instance for DR

The following phases (1-3) guide you through a complete failover and fallback process. You run all the commands by using the `gcloud` command in Cloud Shell. To simplify the process,

the tutorial uses default settings when possible (for example, the default Cloud SQL version). In your production environment, you might add other configurations.

Set environment variables

This section provides examples of environment variables that define the various names and regions that are required for the commands that you run in this tutorial. You can adjust these example variables to fit your needs.

The following tables describe instance names, their roles, and their deployment regions for each phase of the DR and fallback process in this tutorial. You can also provide your own names and regions.

Initial phase

Instance name	Role	Region
`instance-1`	Primary	`us-west1`
`instance-2`	Standby	`us-west1`
`instance-3`	Cross-region read replica	`us-west2`

Disaster phase

Instance name	Role	Region
`instance-3`	Primary	`us-west2`
`instance-4`	Standby	`us-west2`
`instance-5`	Cross-region read replica	`us-west3`
`instance-6`	Cross-region read replica	`us-west1`

Fallback (final) phase

Instance name	Role	Region
`instance-6`	Primary	`us-west1`
`instance-7`	Standby	`us-west1`

Fallback (final) phase		
`instance-8`	Cross-region read replica	`us-west2`

The instance names in the preceding tables aren't encoded with their roles. In a DR situation, the function of an instance might change—for example, a replica might become the primary. If the name of the new primary contains the word `replica`, confusion and conflicts might arise. Therefore, we recommend not encoding instance names with the function or role that they perform.

The preceding tables list the names of standby instances. Even though this tutorial doesn't exercise an HA failover, the tutorial includes the names of standby instances for completeness.

The fallback phase recreates the original deployment of the initial phase in the same original regions. However, in a fallback, the names of the instances must change because the original names aren't immediately available even after the original instance is deleted. To support the speedy creation of instances in the fallback phase, you should use instance names that don't match the names used in the initial phase.

- In Cloud Shell, set environment variables that are based on the specifications in the preceding tables:

```
export primary_name=instance-1
export primary_tier=db-n1-standard-2
export primary_region=us-west1
export primary_root_password=my-root-password
export primary_backup_start_time=22:00
export cross_region_replica_name=instance-3
export cross_region_replica_region=us-west2
```

  If you want to use a different tier for your primary instance, list the tiers that are available to you, and then assign a different value to the primary_tier:

```
gcloud sql tiers list
```

For a list of regions where you can deploy Cloud SQL, see Instance settings (https://cloud.google.com/sql/docs/mysql/instance-settings?authuser=0#region-values).

Create a primary database instance

1. In Cloud Shell, create a single instance (https://cloud.google.com/sql/docs/mysql/create-instance?authuser=0) of Cloud SQL:

```
gcloud sql instances create $primary_name \
    --tier=$primary_tier \
    --region=$primary_region
```

The **gcloud** command pauses until the instance is created.

2. Set the root password:

```
gcloud sql users set-password root \
    --host=% \
    --instance $primary_name \
    --password $primary_root_password
```

Create a primary database

1. In Cloud Shell, log in to the MySQL shell and enter the root password at the prompt:

```
gcloud sql connect $primary_name --user=root
```

2. At the MySQL prompt, create a database and upload test data (https://cloud.google.com/sql/docs/mysql/quickstart?authuser=0#create_a_database_and_upload_data):

```
CREATE DATABASE guestbook;

USE guestbook;
```

```
CREATE TABLE entries (guestName VARCHAR(255), content VARCHAR(255), entryID

INSERT INTO entries (guestName, content) values ("first guest", "I got here
INSERT INTO entries (guestName, content) values ("second guest", "Me too!")
```

3. Check that the data was successfully committed:

```
SELECT * FROM entries;
```

   Verify that two rows of data are returned.

4. Exit the MySQL shell:

```
exit;
```

At this point, you have a single database that includes a table and some test data.

Change the primary instance to an HA database instance

You can only configure Cloud SQL as a regional HA system, not as a cross-regional system. (Setting up a cross-region read replica is different than configuring Cloud SQL as a cross-regional system.) For more information, see Enabling and disabling high availability on an instance (https://cloud.google.com/sql/docs/mysql/configure-ha?authuser=0).

- In Cloud Shell, create an HA-enabled Cloud SQL instance:

```
gcloud sql instances patch $primary_name \
    --availability-type REGIONAL \
    --enable-bin-log \
    --backup-start-time=$primary_backup_start_time
```

Add a cross-region read replica for DR with automatic update

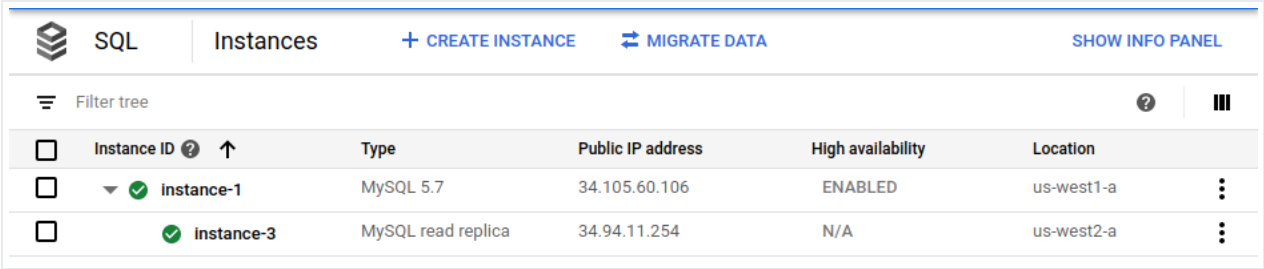
The following steps are sufficient to create a cross-region read replica for this tutorial:

1. In Cloud Shell, set up a cross-region read replica:

```
gcloud sql instances create $cross_region_replica_name \
    --master-instance-name=$primary_name \
    --region=$cross_region_replica_region
```

2. (Optional) To check that the database was replicated, in the Cloud Console, go to the Cloud SQL **Instances** page.

   Go to Instances (https://console.cloud.google.com/sql/instances?authuser=0)



   The Cloud Console shows that the primary instance (**`instance-1`**) is enabled for HA and that a cross-region read replica (**`instance-3`**) exists.

3. Using the same root password for the primary, log in to the cross-region read replica:

```
gcloud sql connect $cross_region_replica_name --user=root
```

4. At the MySQL prompt, select the data to ensure that replication is working:

```
USE guestbook;

SELECT * FROM entries;
```

5. Exit the MySQL shell:

```
exit;
```

For details on how to set up a full cross-region read replica, see the Cloud SQL documentation (https://cloud.google.com/sql/docs/mysql/replication/create-replica?authuser=0)

For large databases in a production environment, we recommend that you back up the primary database and create the cross-region read replica from the backup. This step helps reduce the time it takes for the read replica to synchronize with the primary database. This process is described in the next section. However, you can choose to skip this step and continue with Phase 2 (#phase-2).

Add a cross-region read replica based on a dump file

One way to optimize the creation of a cross-region read replica is to synchronize the replica from an earlier, consistent primary database state instead of synchronizing at the point of accessing the new primary. This optimization requires creating a dump file that the replica uses as the starting state.

For the steps to create a replica based on a dump file, see Replicating from an external server to Cloud SQL (v1.1) (https://cloud.google.com/sql/docs/mysql/replication/replication-from-external?authuser=0#online-+-managed-dump)
. This approach can be helpful for large production databases. However, this tutorial skips this step because the test dataset is small enough for a complete replication.

Phase 2: Simulating a disaster (region outage)

In this phase, you will simulate the outage of a primary region in a production setting by making the primary database unavailable.

Check for cross-region read replica lag

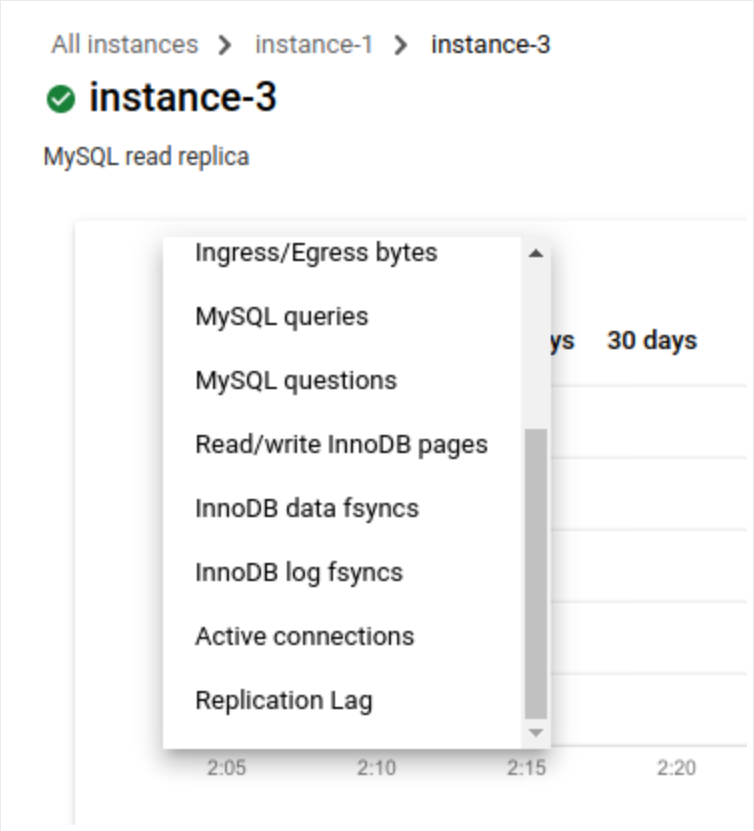
In the following steps, you determine the replication lag of the cross-region read replica:

1. In the Cloud Console, go to the Cloud SQL **Instances** page.

   Go to Instances (https://console.cloud.google.com/sql/instances?authuser=0)

2. Click the read replica (**instance-3**).

3. In the metrics drop-down list, click **Replication Lag**:



The metric changes to **Replication Lag**. The graph shows no delay:

Ideally, the replication lag is zero when a primary region outage occurs, as a delay of zero ensures that all transactions are replicated. If it's not zero, some transactions might not be replicated. In this case, the cross-region read replica won't contain all the transactions that were committed on the primary.

Make the primary instance unavailable

In the following steps, you simulate a disaster by stopping the primary. If a cross-region read replica is attached to the primary, you must first detach the replica, otherwise you can't stop the Cloud SQL instance.

1. In Cloud Shell, remove the cross-region read replica from the primary:

```
gcloud sql instances patch $cross_region_replica_name \
    --no-enable-database-replication
```

   When you're prompted, accept the option to continue.

2. Stop the primary database instance:

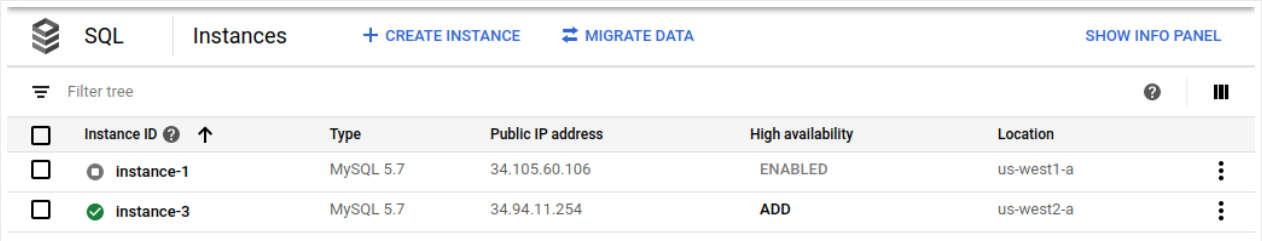
```
gcloud sql instances patch $primary_name --activation-policy NEVER
```

Implement DR

1. In Cloud Shell, promote the cross-region read replica to a standalone instance:

   ```
   gcloud sql instances promote-replica $cross_region_replica_name
   ```

   When you're prompted, accept the option to continue. The Cloud SQL **Instances** page shows the former cross-region read replica (**`instance-3`**) as the new primary, and the former primary (**`instance-1`**) as stopped:

   After you promote the cross-region read replica as the new primary, you enable it for HA. As a best practice, you should update the environment variables with proper naming.

2. Update the environment variables:

   ```
   export former_primary_name=$primary_name
   export primary_name=$cross_region_replica_name
   export primary_tier=db-n1-standard-2
   export primary_region=$cross_region_replica_region
   export primary_root_password=my-root-password
   export primary_backup_start_time=22:00
   export cross_region_replica_name=instance-5
   export cross_region_replica_region=us-west3
   ```

3. Start the new primary:

   ```
   gcloud sql instances patch $primary_name --activation-policy ALWAYS
   ```

4. Enable the new primary as an HA regional instance:

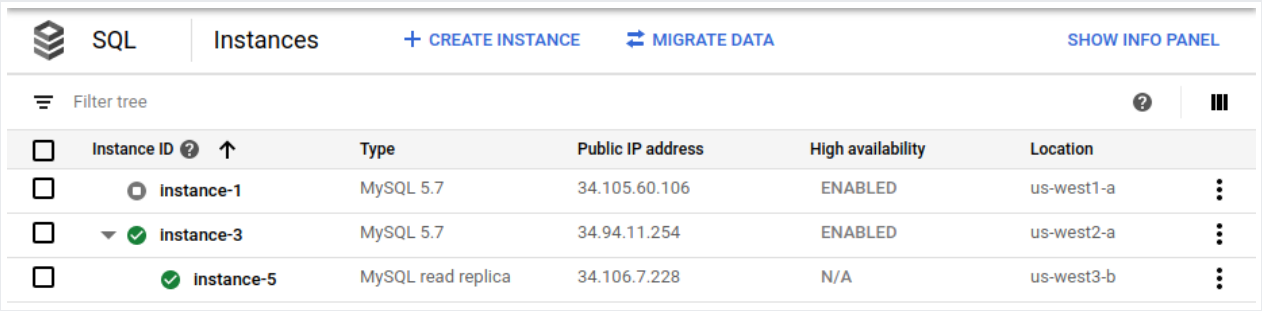
```
gcloud sql instances patch $primary_name \
    --availability-type REGIONAL \
    --enable-bin-log \
    --backup-start-time=$backup_start_time
```

5. Create a cross-region read replica in a third region:

```
gcloud sql instances create $cross_region_replica_name \
    --master-instance-name=$primary_name \
    --region=$cross_region_replica_region
```

   In an earlier step, you set the `cross_region_replica_region` environment variable to `us-west3`.

   After the failover completes, the Cloud SQL **Instances** page in the Cloud Console shows that the new primary (**`instance-3`**) is enabled as HA and has a cross-region read replica (**`instance-5`**):



6. (Optional) If you have regular backups, follow the process described earlier (#add-replica-dump) to synchronize the new primary with the latest backup version.

7. (Optional) If you're using a Cloud SQL proxy, configure the proxy (https://cloud.google.com/sql/docs/mysql/connect-admin-proxy?authuser=0) to use the new primary in order to resume the application processing.

Handle a short-lived region outage

It's possible that the outage that triggers a failover is resolved before the failover completes. In this case, it might make sense to cancel the failover process and continue using the original primary Cloud SQL instance in the region where the outage occurred.

Depending on the specific state of the failover process, the cross-region read replica might have been promoted already. In this case, you must delete it and re-create a cross-region read replica.

Delete the original primary to avoid a split-brain situation

To avoid a split-brain situation, you need to delete the original primary (or make it inaccessible to database clients).

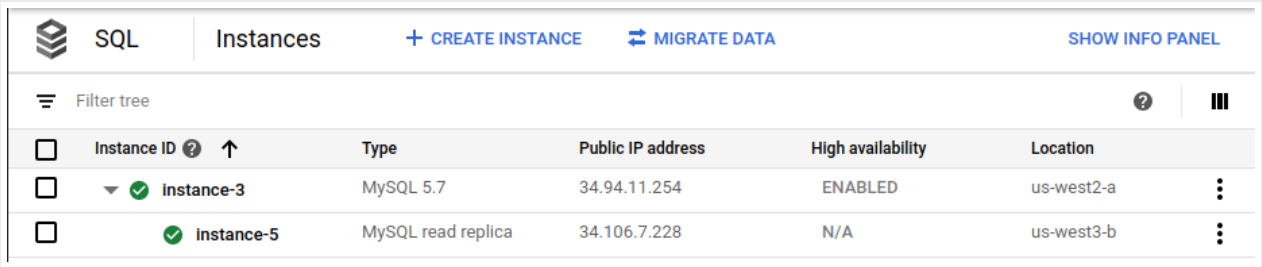
After a failover, a split-brain situation can occur when clients write to the original primary database and the new primary database at the same time. In this case, the content of the two databases is inconsistent. After a failover, the original primary database is outdated and must not receive any read or write traffic.

- In Cloud Shell, delete the original primary:

  ```
  gcloud sql instances delete $former_primary_name
  ```

  When you're prompted, accept the option to continue.

  In the Cloud Console, the Cloud SQL **Instances** page no longer shows the original primary instance (**`instance-1`**) as part of the deployment:



Phase 3: Implementing a fallback

To implement a fallback to your original region (R1) after it becomes available, you follow the same process that is described in Phase 2. That process is summarized as follows:

1. Create a second cross-region read replica in the original region (R1). At this point, the primary has two cross-region read replicas, one in region R3, and one in region R1.
2. Promote the cross-region read replica in R1 as the final primary.
3. Enable HA for the final primary.
4. Create a cross-region read replica of the final primary in `us-west2`.
5. To avoid a split-brain situation, delete all instances that are no longer required (the original primary and the cross-region read replica in R3).

As discussed earlier, it's a best practice to create an initial backup that contains the defined start state for the new primary database.

The final deployment now has an HA primary (with the name `instance-6`) and a cross-region read replica (with the name `instance-8`).

Comparing advantages and disadvantages of a manual versus automatic DR

The following table discusses the advantages and disadvantages of implementing a DR process either manually or automatically. The goal isn't to determine a correct versus incorrect approach but to provide criteria to help you determine the best approach for your needs.

Manual execution	Automatic execution
Advantages:	**Advantages:**
• You have tight control over every step. • You can immediately see, address, and document any issue in the process. • You can see and review every process step during a failover.	• You can implement and test failover processes. • Automation offers the quickest implementation and minimizes delays. • Implementation is independent of human operators, their knowledge, and their availability.

Manual execution	Automatic execution
Disadvantages: • Manually implementing process steps slows down the process. • Human typing errors can introduce issues. • Testing the process typically involves several roles and time, which might discourage regular testing.	**Disadvantages:** • If an unforeseen error occurs, you have to debug during your production failover. • If you encounter errors during the process, you need scripts to pick up (recover) where the process left off. • Sufficient knowledge of the script and its implementation is required to understand the script's behavior, especially in error situations.

As a best practice, we recommend that you start with a manual implementation. Then, voluntarily run the implementation regularly (preferably in production) to ensure that the manual process works and that all team members know their roles and responsibilities. We recommend that you define your manual process in a step-by-step process document. After every implementation, you should confirm or refine the process document.

After you fine-tune the process and are confident that it's reliable, you then determine whether to automate the process. If you select and implement an automated process, you need to test the process regularly in production to ensure that you can implement it reliably.

Cleaning up

To avoid incurring charges to your Google Cloud account for the resources used in this tutorial, you can delete the Cloud project that you created for this tutorial.

Delete the project

Caution: Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such

as an **`appspot.com`** URL, delete selected resources inside the project instead of deleting the whole project.

If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

1. In the Cloud Console, go to the **Manage resources** page.

   Go to Manage resources (https://console.cloud.google.com/iam-admin/projects?authuser=0)

2. In the project list, select the project that you want to delete, and then click **Delete**.
3. In the dialog, type the project ID, and then click **Shut down** to delete the project.

What's next

- Read about Cloud SQL disaster recovery (https://cloud.google.com/architecture/intro-to-cloud-sql-disaster-recovery?authuser=0).
- Read about disaster recovery for MySQL on Compute Engine (https://cloud.google.com/blog/products/databases/disaster-recovery-for-mysql?authuser=0).
- Learn about disaster recovery architectures for cloud infrastructure outages (https://cloud.google.com/solutions/disaster-recovery/architecture?authuser=0).
- Explore reference architectures, diagrams, tutorials, and best practices about Google Cloud. Take a look at our Cloud Architecture Center (https://cloud.google.com/architecture?authuser=0).

Last updated 2021-05-17 UTC.