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 by using cross-region read replicas (/sql/docs/mysql/replication/cross-region-replicas).

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.

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 (/sql/docs/mysql/high-availability).

Costs

This tutorial uses the following billable components of Google Cloud:

- Cloud SQL (/sql/pricing)

To generate a cost estimate based on your projected usage, use the pricing calculator (/products/calculator). New Google Cloud users might be eligible for a free trial (/free-trial).

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 the project selector page (https://console.cloud.google.com/projectselector2/home/dashboa

2. Make sure that billing is enabled for your Cloud project. Learn how to confirm that billing is enabled for your project (/billing/docs/how-to/modify-project).
3. In the Cloud Console, activate Cloud Shell.

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

Database DR

In Google Cloud, database DR is about providing continuity of processing, specifically when a region fails or becomes unavailable. Cloud SQL is a regional service (when Cloud SQL is configured for HA (/sql/docs/mysql/high-availability)). Therefore, if the Google Cloud region that hosts a Cloud SQL database becomes unavailable, then the Cloud SQL database also becomes unavailable.

To continue processing, you must make the database available in a secondary region as soon as possible. The DR plan in this tutorial requires you to configure a cross-region read replica in

Cloud SQL. A failover based on export/import is also possible, but that approach takes longer, especially for large databases.

The following business scenarios are examples that warrant a cross-region failover configuration:

- The service level agreement of the business application is greater than the regional Cloud SQL Service Level Agreement (/sql/sla) (99.95% availability). By failing over to another region, you can mitigate an outage.
- All tiers of the business application are already multi-regional and can continue processing when a region outage occurs. The cross-region failover configuration ensures the continued availability of a database.
- The required recovery time objective (RTO) and recovery point objective (RPO) are in minutes rather than in hours. Failing over to another region is faster than recreating a database.

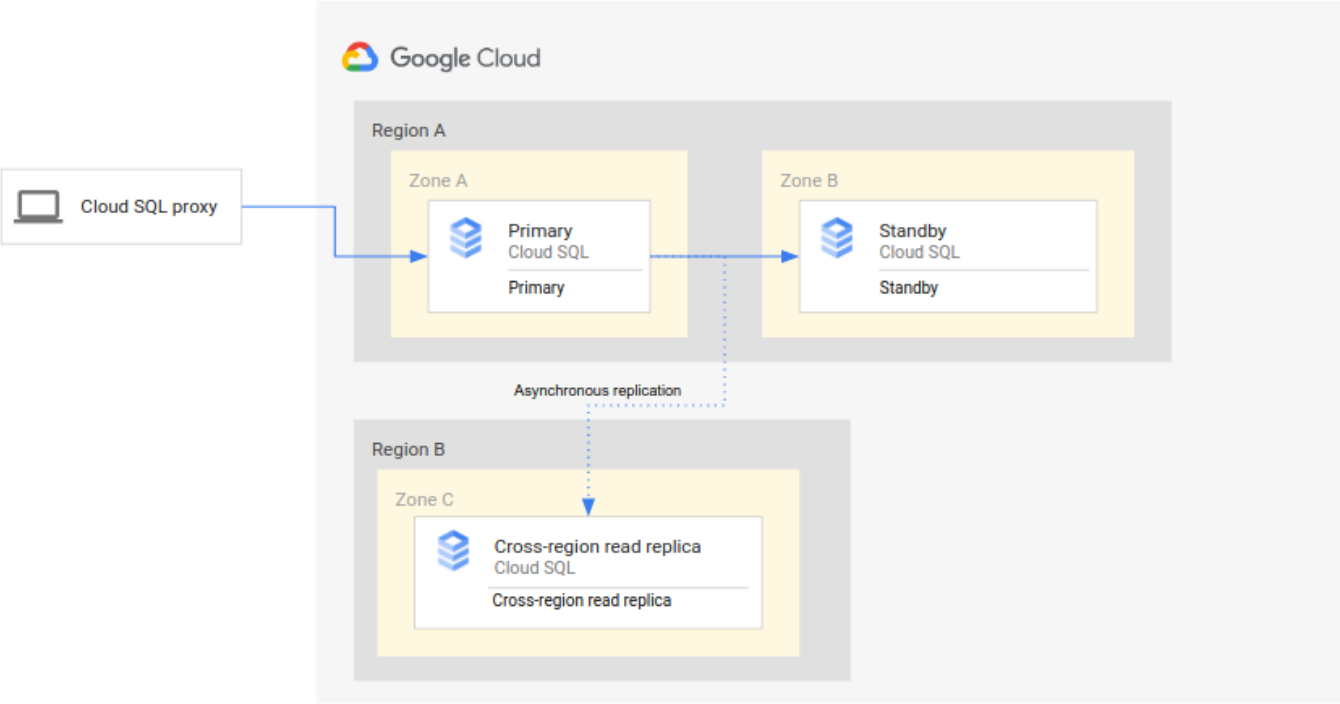
In general, there are two variants for the DR process:

- A database fails over to a secondary region. After the database is ready and used by an application, it becomes the new primary database and remains the primary database.
- A database fails over to a secondary region but falls back to the primary region after the primary region is recovered from its failure.

This tutorial describes the second variant—when a failed database is recovered and falls back to the primary region. This DR process is especially relevant for databases that must run in the primary region because of network latency, or because some resources are available only in the primary region. With this variant, the database runs in the secondary region only for the duration of outage in the primary region.

DR architecture

The following diagram shows the minimal architecture that supports database DR for an HA Cloud SQL instance:



The architecture works as follows:

- Two instances of Cloud SQL (a primary instance and a standby instance) are located in two separate zones within a single region (the primary region). The instances are synchronized by using regional persistent disks.
- One instance of Cloud SQL (the cross-region read replica) is located in a second region (the secondary region). For DR, the cross-region read replica is set up to synchronize (by using asynchronous replication) with the primary instance using a read replica setup.

The primary and standby instances share the same regional disk, and so their states are identical.

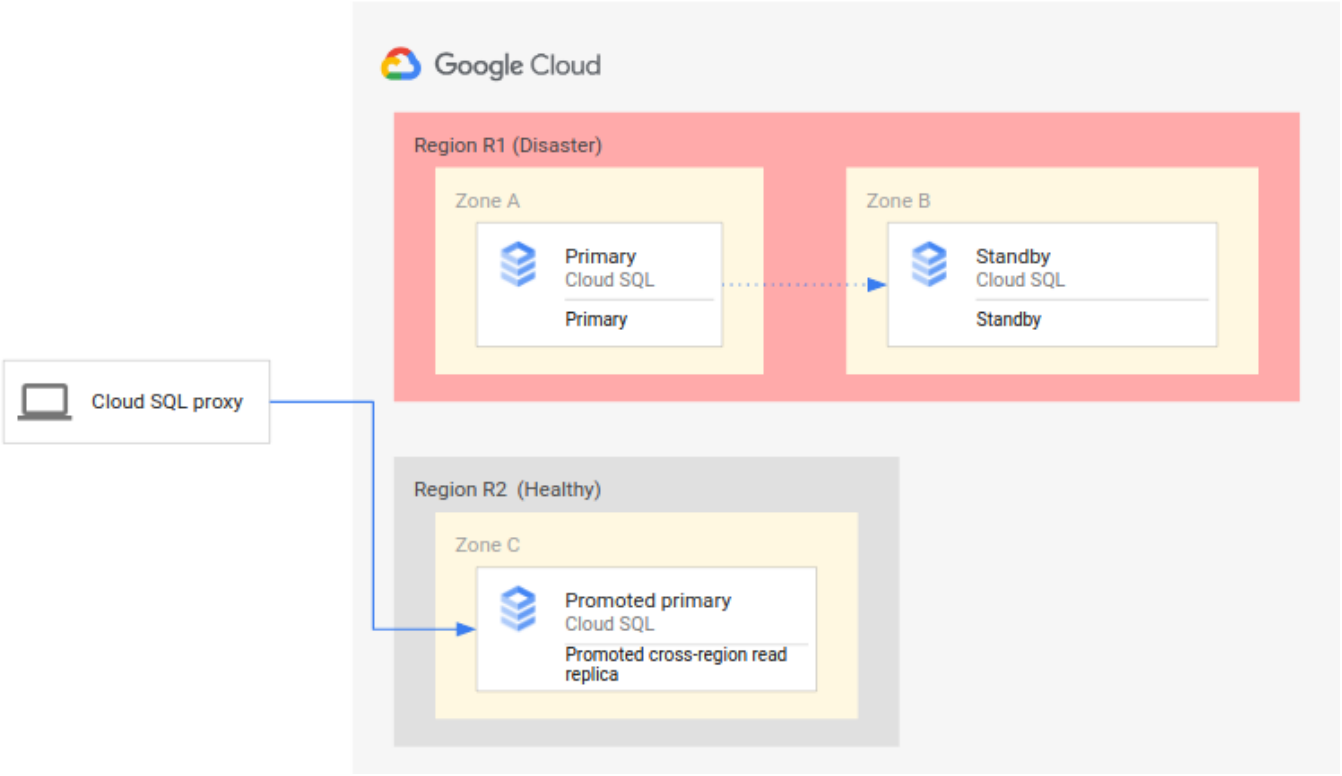
Because this setup uses asynchronous replication, it's possible that the cross-region read replica lags behind the primary instance. When a failover occurs, the cross-region read replica might support an RPO of zero.

Overview of the basic DR process

The basic DR process starts when the primary region becomes unavailable and the primary instance fails over to resume processing in a secondary region. The DR process prescribes the

operational steps that must be performed, either manually or automatically (#comparison), to mitigate the regional failure and establish a running primary instance in a secondary region.

The following diagram shows the DR process:



The preceding DR process consists of the following steps:

1. The primary region (R1), which is running the primary instance, becomes unavailable.
2. The operations team recognizes and formally acknowledges the disaster and decides whether a failover is required.
3. If a failover is required, the cross-region read replica in the secondary region (R2) is made the new primary instance.
4. Client connections are reconfigured to resume processing on the new primary instance and access the primary instance in R2.

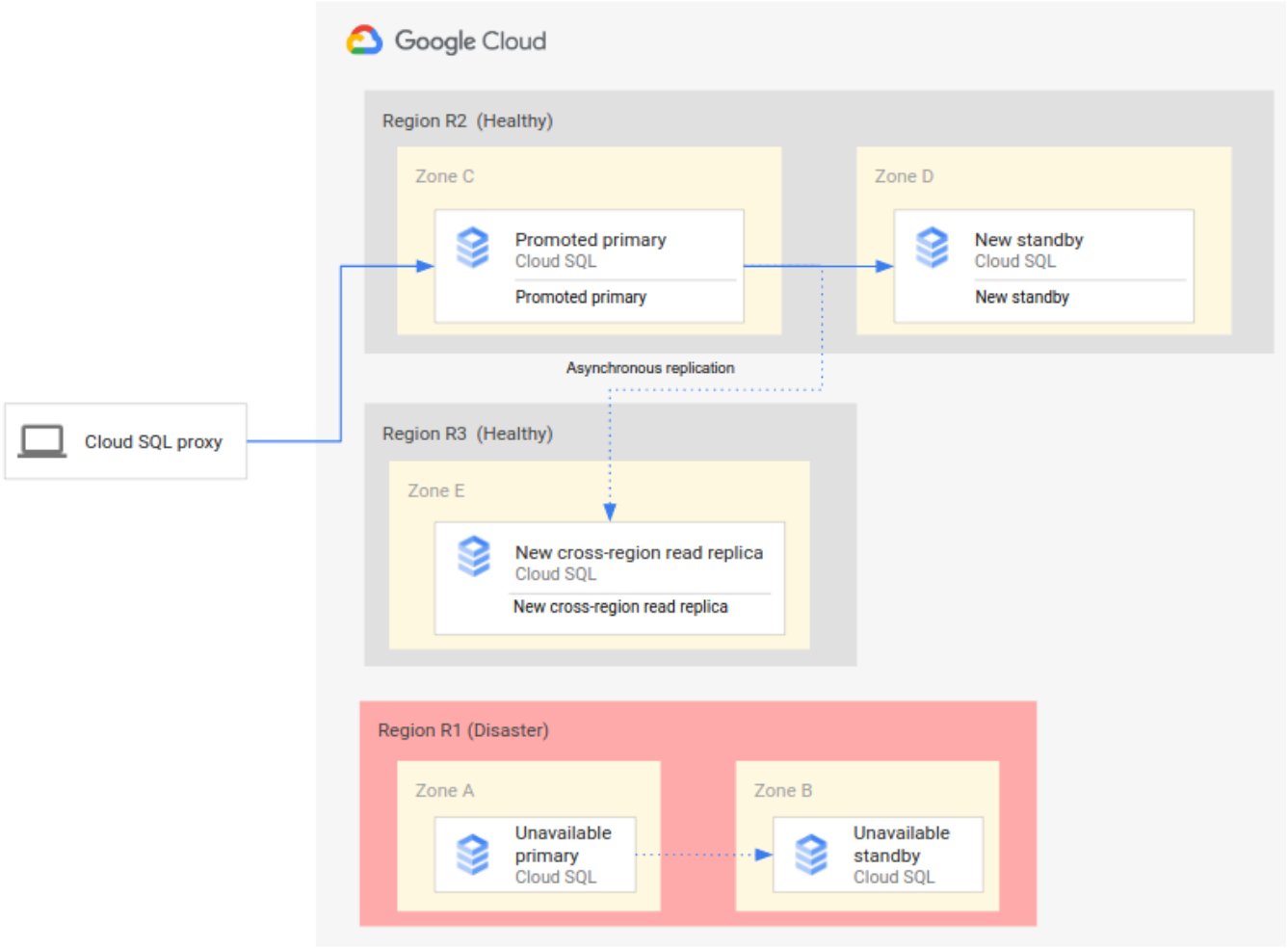
This basic process establishes a working primary database again. However, it doesn't establish a complete DR architecture, where the new primary instance itself has a standby instance and a cross-region read replica.

Overview of the complete DR process

A complete DR process ensures that the single instance, the new primary, is enabled for HA and has a cross-region read replica. A complete DR process also provides a fallback to the original deployment in the original primary region.

Failing over to a secondary region

A complete DR process extends the basic DR process by adding steps to establish a complete DR architecture after failover. The following diagram shows a complete database DR architecture after the failover:



The complete database DR process consists of the following steps:

1. The primary region (R1), which is running the primary database, becomes unavailable.

2. The operations team recognizes and formally acknowledges the disaster and decides whether a failover is required.
3. If a failover is required, the cross-region read replica in the secondary region (R2) is made the new primary instance.
4. Client connections are reconfigured to access and process on the new primary instance (R2).
5. A new standby instance is created and started in R2 and added to the primary instance. The standby instance is in a different zone from the primary instance. The primary instance is now highly available because a standby instance was created for it.
6. In a third region (R3), a new cross-region read replica is created and attached to the primary instance. At this point, a complete disaster recovery architecture is recreated and operational.

If the original primary region (R1) becomes available before step 6 is implemented, the cross-region read replica can be placed in region R1, rather than region R3, right away. In this case, the fallback to the original primary region (R1) is less complex and requires fewer steps.

Avoiding a split-brain state

A failure of the primary region (R1) doesn't mean that the original primary instance and its standby instance are automatically shut down, removed, or otherwise made inaccessible when R1 becomes available again. If R1 becomes available, clients might read and write data (even by accident) on the original primary instance. In this case, a split-brain situation (https://wikipedia.org/wiki/Split-brain_(computing)) can develop, where some clients access stale data in the old primary database, and other clients access fresh data in the new primary database, leading to problems in your business.

To avoid a split-brain situation, you must ensure that clients can no longer access the original primary instance after R1 becomes available. Ideally, you should make the original primary inaccessible before clients start using the new primary instance, then delete the original primary right after you make it inaccessible.

Establishing an initial backup after failover

When you promote the cross-region read replica to be the new primary in a failover, the transactions in the new primary might not be fully synchronized with transactions from the

original primary. Therefore, those transactions are unavailable in the new instance.

As a best practice, we recommend that you immediately back up the new primary instance at the start of the failover and before clients access the database. This backup represents a consistent, known state at the point of the failover. Such backups can be important for regulatory purposes or for recovering to a known state if clients encounter issues when accessing the new primary.

Falling back to an original primary region

As outlined earlier, this tutorial provides the steps to fall back to the original region (R1). There are two different versions of the fallback process.

- If you created the new cross-region read replica in a tertiary region (R3), you must create another (second) cross-region read replica in the primary region (R1).
- If you created the new cross-region read replica in the primary region (R1), you don't need to create another additional cross-region read replica in R1.

After the cross-region read replica in R1 exists, the Cloud SQL instance can fall back to R1. Because this fallback is manually triggered and not based on an outage, you can choose an appropriate day and time for this maintenance activity.

Thus, to achieve a complete DR that has a primary, standby, and cross-region read replica, you need two failovers. The first failover is triggered by the outage (a true failover), and the second failover re-establishes the starting deployment (a fallback or switchover).

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

examples 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`
`instance-8`	Cross-region read replica	`us-west2`

The preceding instance names aren't encoded with their role. In a DR situation, an instance's function 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 primary_tier:

```
gcloud sql tiers list
```

  For a list of regions where you can deploy Cloud SQL, see Instance settings (/sql/docs/mysql/instance-settings#region-values).

Create a primary database instance

1. In Cloud Shell, create a single instance (/sql/docs/mysql/create-instance) 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 (/sql/docs/mysql/quickstart#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 (/sql/docs/mysql/configure-ha).

- 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

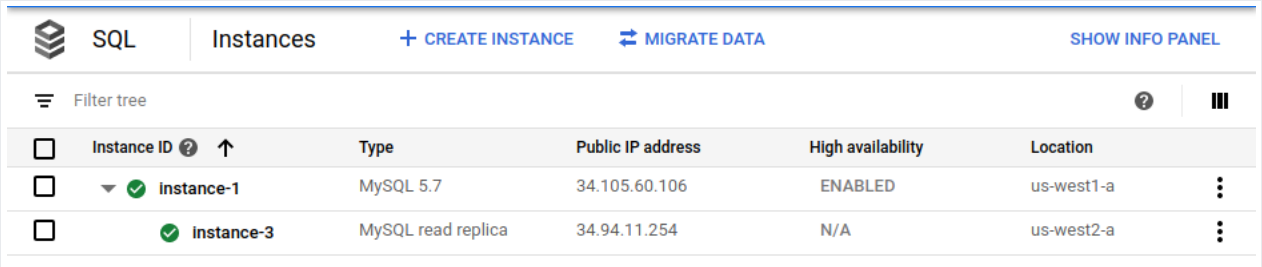
The Cloud SQL documentation (/sql/docs/mysql/replication/create-replica) details how to set up a cross-region read replica, but the following steps are sufficient for this tutorial:

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)



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 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) (/sql/docs/mysql/replication/replication-from-external#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)

This tutorial simulates the outage of a primary region in a production setting by making the primary database unavailable.

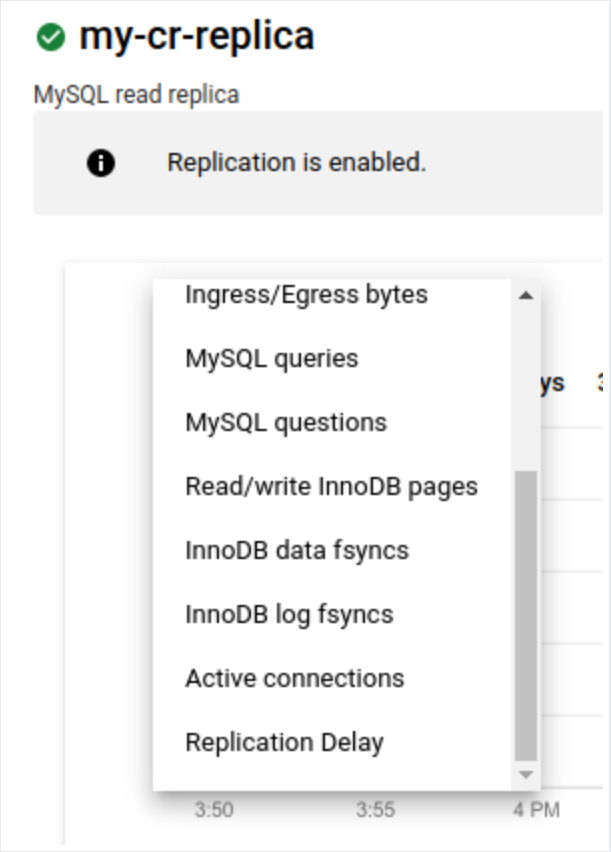
Check for cross-region read replica lag

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

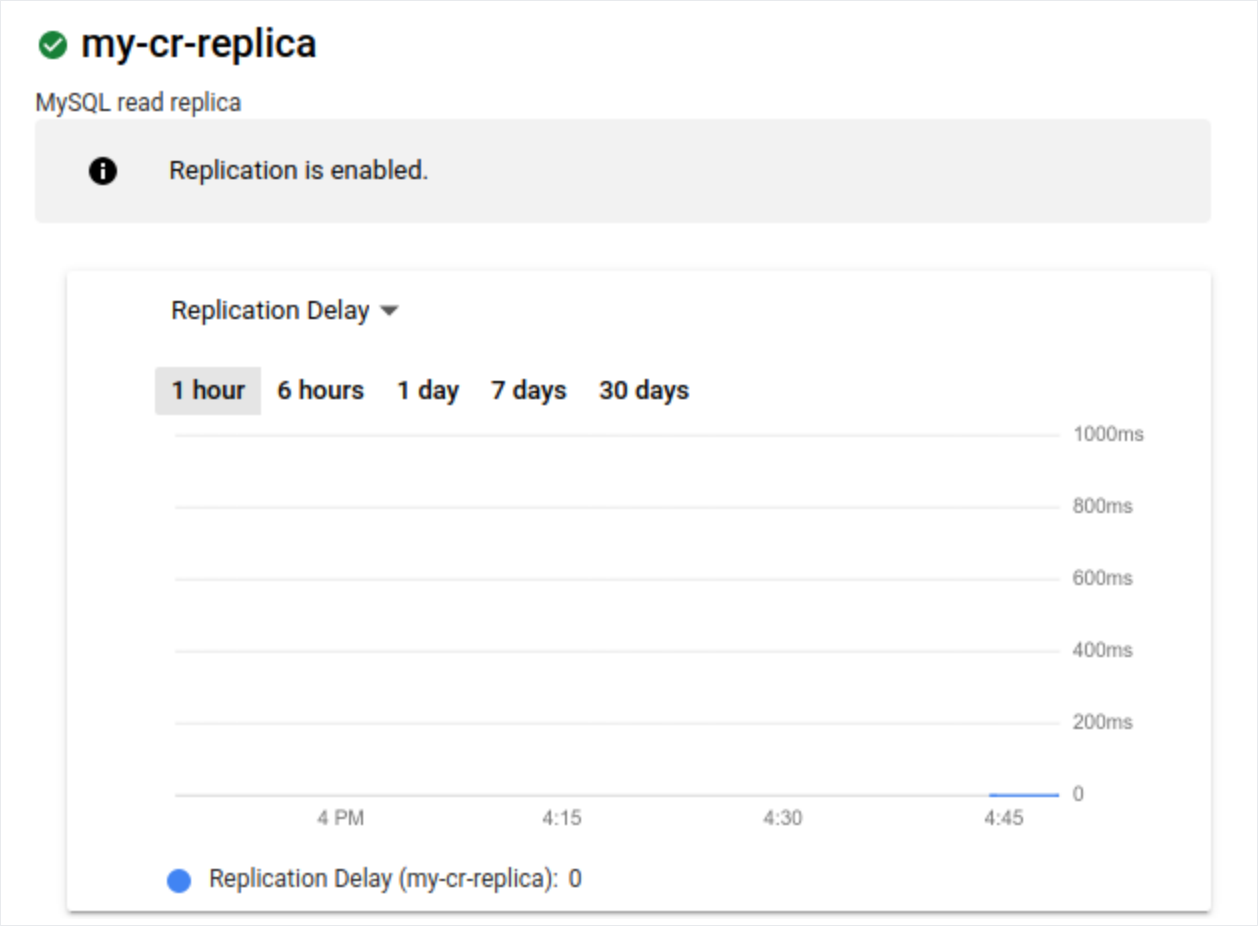
1. In the Cloud Console, go to the Cloud SQL **Instances** page.

   Go to Instances (https://console.cloud.google.com/sql/instances)

2. Click the read replica (**instance-3**).
3. In the metrics drop-down list, click **Replication Delay**:



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



Ideally, the replication delay 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:

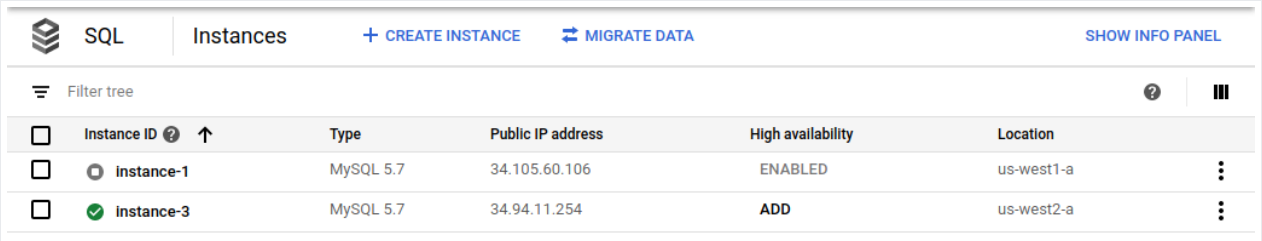
```
gcloud sql instances patch $primary_name --activation-policy NEVER
```

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:

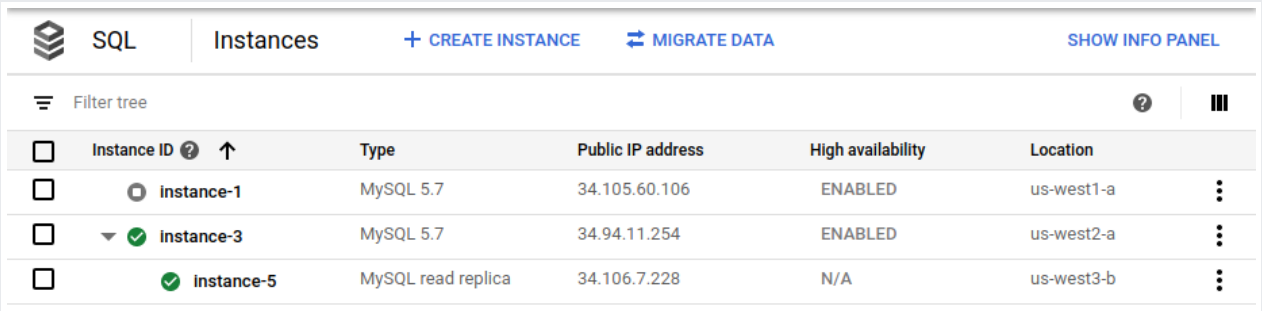
```
gcloud sql instances patch $primary_name \
    --availability-type REGIONAL \
    --enable-bin-log \
    --backup-start-time=$backup_start_time
```

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 (/sql/docs/mysql/connect-admin-proxy) 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).

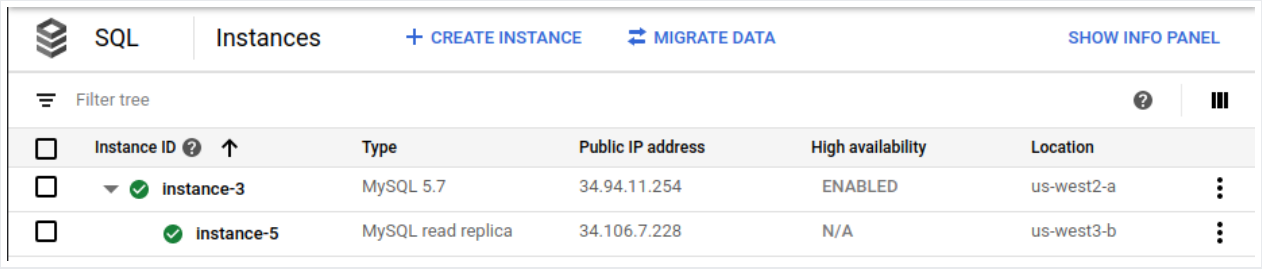
After a failover, a split-brain situation can occur when clients write to the original primary database and the new primary database at the same time. In this case, the content of the two databases is inconsistent. After a failover, the original primary database is outdated and must not receive any read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

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 disaster recovery for MySQL on Compute Engine (/blog/products/databases/disaster-recovery-for-mysql).
- Learn about disaster recovery architectures for cloud infrastructure outages (/solutions/disaster-recovery/architecture).
- Try out other Google Cloud features for yourself. Have a look at our tutorials (/docs/tutorials).

Last updated 2021-01-28 UTC.