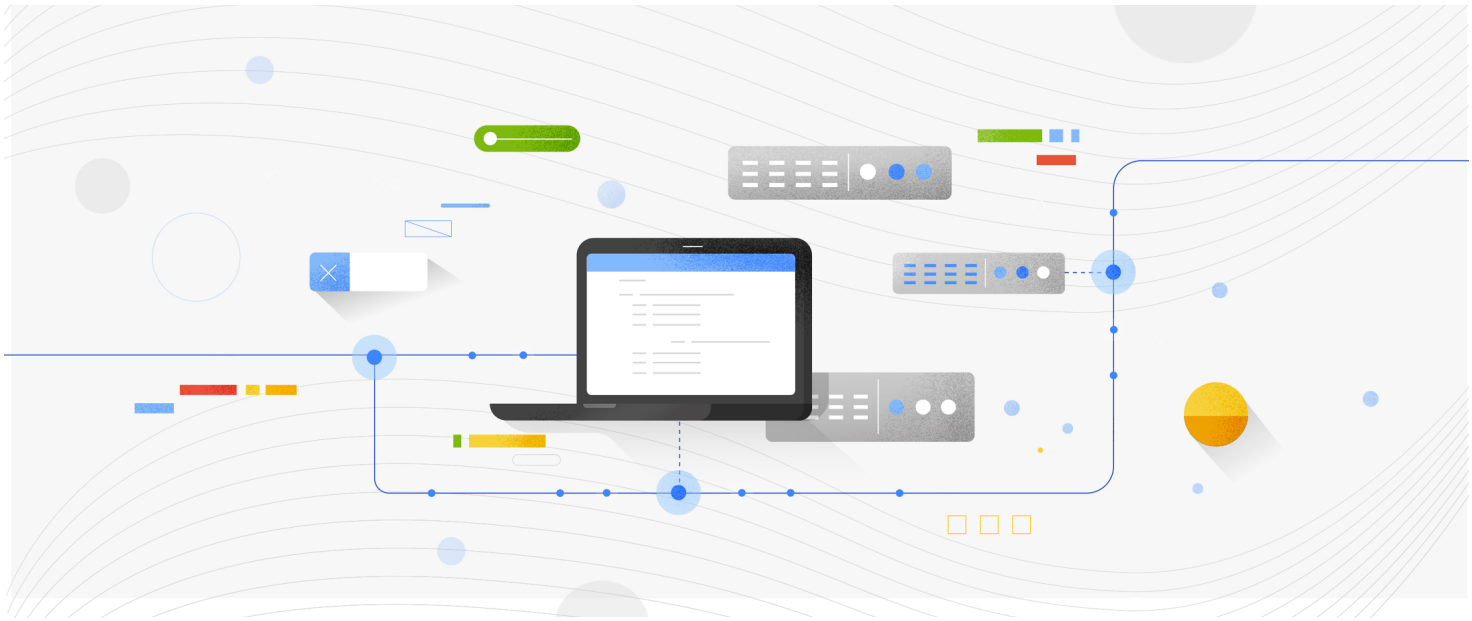
DATABASES

Architecting multi-region database disaster recovery for MySQL



Shashank Agarwal
Database Migration Engineer

Christoph Bussler
Solutions Architect, Google Cloud

March 6, 2020

Enterprises expect extreme reliability of the database infrastructure that's accessed by their applications. Despite your best intentions and careful engineering, database errors happen, whether that's machine crashes or network partitioning. Good planning can help you stay ahead of problems and recover more quickly when issues do occur.

This blog shows one approach of deploying a database architecture that implements high availability and disaster recovery for MySQL on Compute Engine, using regional disks as well as load balancers.

Any database architecture must provide approaches to tolerate errors and recover from those errors quickly without losing data. These approaches are expressed in RTO (recovery time objective) and RPO (recovery point objective), which offer ways to set and then measure how long a service can be unavailable, and how far back data should be saved.

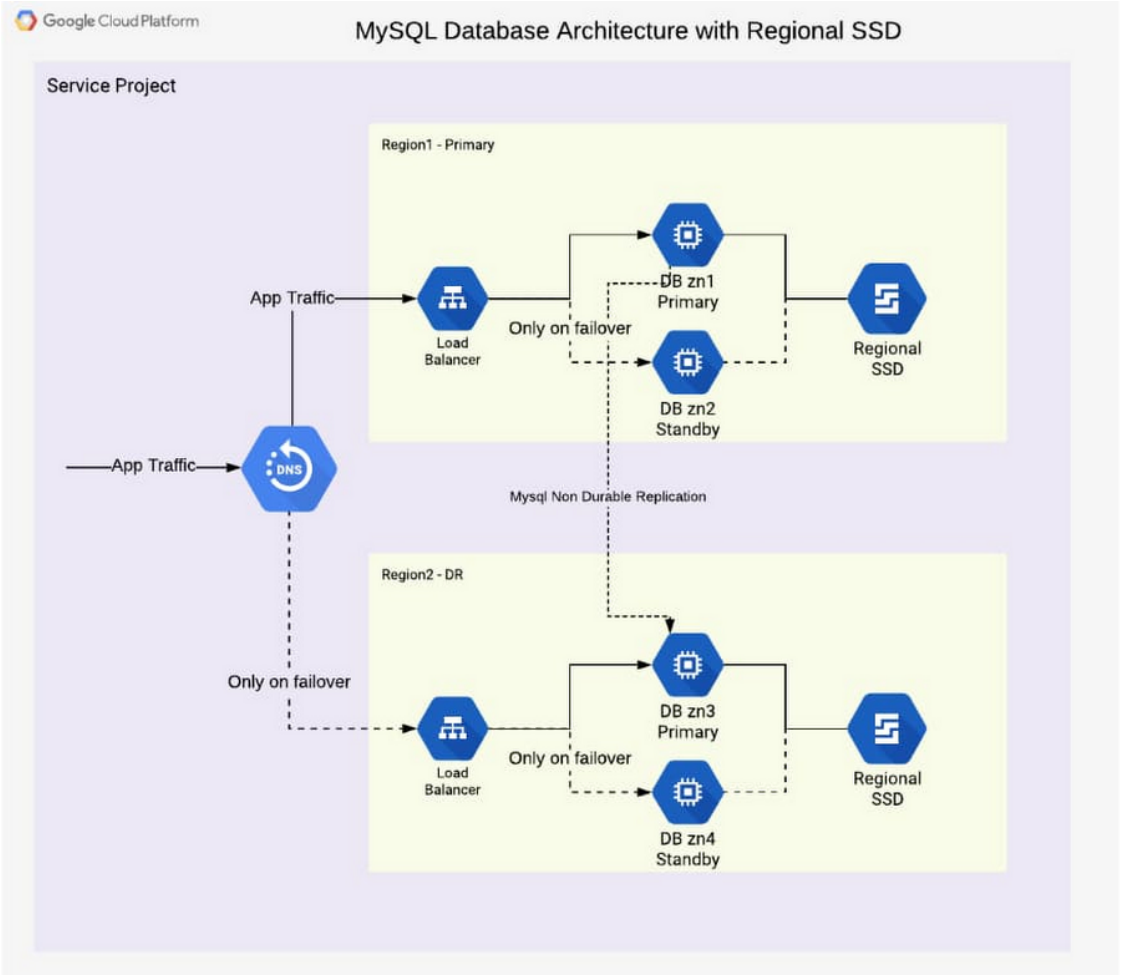
After a database error, a database must recover as fast as possible with an RTO as small as possible, ideally in seconds. There must be as little data loss as possible—ideally, none at all. The desired RPO is the last consistent database state.

From a database architecture and deployment viewpoint, this can be accomplished with two distinct concepts: high availability and disaster recovery. Use both at the same time in order to achieve an architecture that's prepared for the widest range of errors or incidents.

Creating a resilient database architecture

A high-availability database architecture has database instances in two or more zones. If a server on a zone fails, or the zone becomes inaccessible, the instances in other zones are available to continue the processing. The figure below shows two instances, one in zone zn1, and one in zone zn2. The load balancer in front supports directing traffic to a healthy database instance available for read and write queries.

A disaster recovery architecture adds a second high-availability database setup in a second region. If one of the regions becomes inaccessible or fails, the other region takes over. The figure below shows two regions, primary and DR. Data is replicated from the primary to the DR region so that the DR region can take over from the latest consistent database state. The load balancer in front of the regions directs traffic to the region in charge of the read and write traffic. Here's how this architecture looks:



In addition to the database instance setup, a regional disk is deployed so that data is written simultaneously in two zones, proving fail-safe in the event of zone failure. This is a huge advantage of Google Cloud, allowing you to skip MySQL-level replication within a region. Each write operation to disk is done in two zones synchronously. When the primary instance fails, a standby instance is mounted with regional persistent disk(s), and the database service (MySQL) is then started using the same. This brings the peace of mind of not worrying about replication lag or database state for high availability.

From a disaster recovery process view, the following happens over time during a failure situation:

- Normal steady state database operation
- A failure happens and a region becomes unavailable or the database instance inaccessible
- A decision must be made to fail over or not (in case there is the expectation that the region becomes available soon enough or the instance becomes responsive again)
- DNS is updated manually, therefore it redirects application traffic to a second region

- Fallback to the primary region after it becomes available again is optional, as the second region is a fully built-out deployment

From a high-availability process view, the following happens over time during a failure situation:

- Normal steady state database operation
- Database instance fails or becomes unavailable
- Launch the standby instance
- Mount regional SSD and start database
- Automatic redirection of application traffic to the standby via load balancer
- After the failed or unavailable instance becomes available again, a fallback can take place or not

The database architecture shown demonstrates a highly available architecture supporting disaster recovery. With regional disks and load balancers, it is straightforward to provide a resilient database deployment.

Find out more about load balancers and regional disks. Check out general HA and DR processes and detailed steps in the initial part of the reference guide. Try it out to become familiar with the architecture as well as the two major failover processes.

POSTED IN: DATABASESGOOGLE CLOUD PLATFORM

RELATED ARTICLES

·ate your Microsoft SQL Server ‹loads to Google Cloud

Introducing more maintenance controls for Cloud SQL