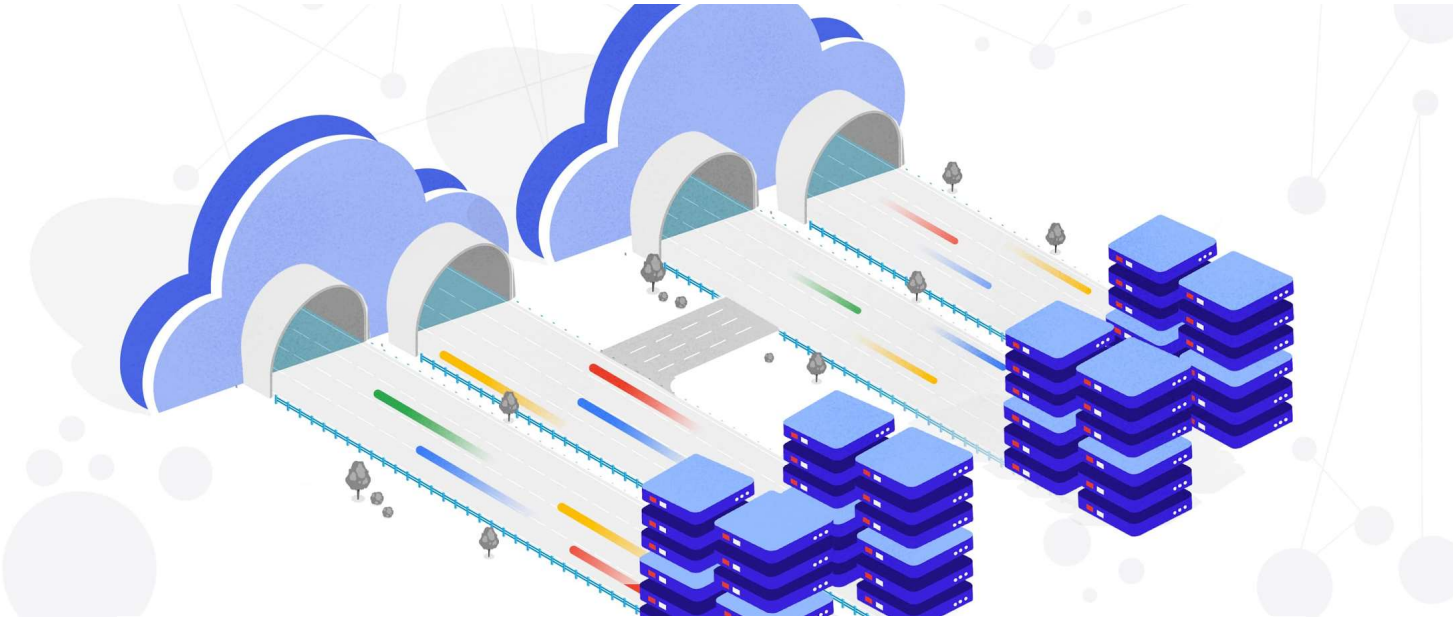
DATABASES

Best practices for homogeneous database migrations

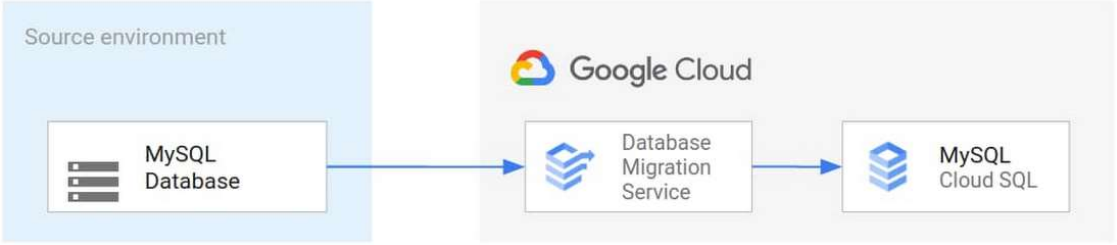


Christoph Bussler
Solutions Architect, Google Cloud

November 12, 2020

Migrating applications to Google Cloud is most effective when you're migrating their backing database or databases to Google Cloud as well. The result is improved performance, lowered cost, and ease of management and operations.

To make these migrations easier, we've announced the new Database Migration Service(DMS), an easy-to-use, serverless migration tool that provides minimal downtime database migration to Cloud SQL for MySQL (in preview) and Cloud SQL for PostgreSQL(in private preview—sign up here).



DMS currently focuses on homogeneous migrations—that is, migrations across compatible database engines, which don't require transformations such as schema

conversion from source to destination. In this case, the goal of the migration is for the migrated database to be as close as possible to a copy of the source database, available in the Cloud SQL destination.

This blog outlines the best practices of homogeneous database migration to Cloud SQL for MySQL, and how DMS enables a secure, straightforward database migration experience.

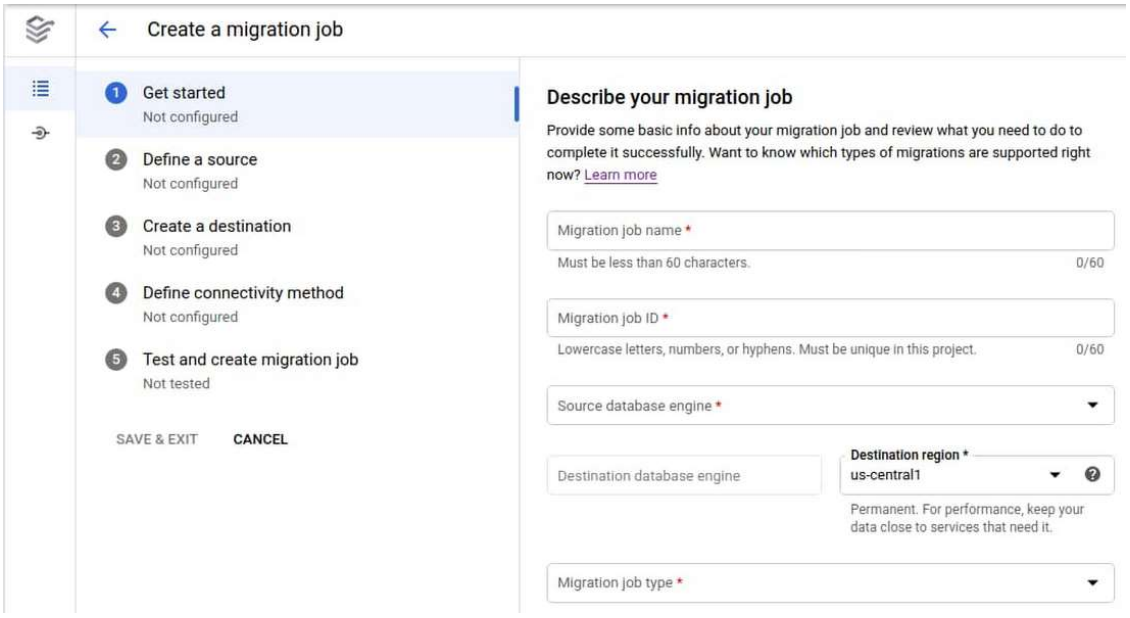
The Database Migration Service advantage

DMS supports homogeneous database migrations as a serverless service. By utilizing the database engine's own native replication technology, DMS supports continuous replication from source to destination. This ensures the databases are constantly in sync, with maximum fidelity of data being transferred. You can cut over to use your application with your new Cloud SQL instance with minimal downtime.

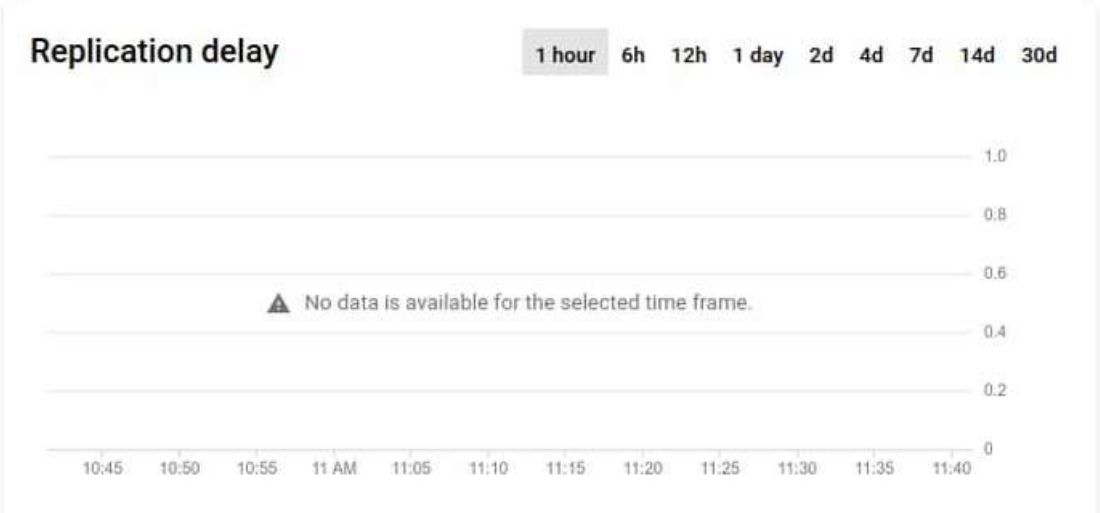
DMS makes database migration simple and reliable with a few key capabilities:

- **Guided, validated setup flow.** The migration job creation flow uniquely features built-in source configuration guidance and secure connectivity support (see screenshot below), to ease the highest complexity portions of migration setup. Within the flow, setup and configuration are validated to ensure that the database migration is set up to succeed.
- **Modularity and reuse of connection profile.** The connection to the source database is specified separately, so you can reuse it throughout the definition, testing, and execution phases without requiring the user to re-enter configuration values. This also enables separation of ownership between teams, separating who defines the connection and who executes the migration.
- **Monitored, native migrations.** DMS utilizes the open source database's own native replication technologies to ensure a reliable, high-fidelity migration. Running migrations can be monitored via UI and API, including tracking any migration delay (see second screenshot below).
- **Automatic management of migration resources.** As a serverless service, any required migration resources are automatically managed by DMS. No resources have to be provisioned or managed by the user, and migration-specific resources never need to be monitored.

DMS's user interface provides a structured process for migration job creation:



DMS provides status and monitoring visibility:



Using Database Migration Service for your migration journey

Here, we'll go through the phases of an overall database migration journey, with guidance on how to leverage DMS in the process.

Assessment and planning

The goal of the assessment phase of a database migration is to collect and review the source databases that have been identified as candidates for migration to Google Cloud. The subsequent planning phase then creates an overall migration plan, including tasks for the implementation of migration jobs, their testing, and the actual migration of the production databases, including database promotion and application cutover.

For this post, we'll focus on migrating the database with DMS, and not the applications that access it. Find more details on application migrations in the Migration to Google Cloud Solution.

Source database assessment

In the common case of database migration, several databases are migrated in a coordinated wave, since applications can depend on more than one data source, or they may be interrelated. The first step for such a process is to collect all databases that are in the scope of a wave of database migrations. For each database, decide if it's a homogeneous migration to the same, compatible database engine in Google Cloud, and therefore a good fit for the current capabilities of DMS.

The most important aspects for analysis are:

- **Prerequisites.** To migrate a database, it has to fulfill specific prerequisites: Namely, specific source database configuration needs to be performed (for example, enabling binary logging), and preparing network connectivity that suits the security posture and requirements of the organization. You can meet these requirements by changing the source configuration and network setup from within the Database Migration Service in order to streamline the process and simplify the migration setup.
- **Size.** Determine the database size, since this will provide input to planning the migration timeline: The larger the database, the more time it will take to migrate the initial snapshot and test the migration as part of the move to Google Cloud in production.

The following discussion focuses on a single database for simplicity. In the case of migrating several databases, all migration-related tasks can be performed in parallel for each of the databases.

Database migration planning

The goal of planning a database migration is to create a list of all the necessary tasks that will ensure a successful migration and production promotion. A timeline-based project plan will indicate the time and the order of the various tasks. Their duration often

depends on the size of the database, especially in the case of testing and migration tasks, as well as other factors like team availability and application usage.

If multiple databases are migrated in waves, a staggered overall migration plan is a good approach. In order to gain experience with DMS and the database migration process, it is a good practice to start with smaller, less mission-critical databases.

The basic elements of a migration plan for a single database are:

1. **Migration timeline.** A timeline with start and projected end dates will specify the overall duration. It contains all the tasks that have to be accomplished.
2. **Preparation tasks.** Preparation tasks determine the size of the database and confirm that all prerequisites are met (as indicated above). This should also include any changes that have to be made to the source database in preparation for migration.
3. **Execution tasks.** These tasks implement the DMS migration job. Information about preparation details as well as migration job creation details are provided in the user interface as a one-stop shop for all required knowledge and background.
4. **Testing.** One of the most important tasks is to test the migration in context of a proof of concept. This can be done only for the initial databases as you gain migration experience, or for every migrated database. A test migrates the database to Google Cloud completely and performs validation, while not yet moving the production application workload to Google Cloud. The goal of testing is to verify that the migration of the database and moving production to Google Cloud will be successful. The application is thoroughly tested against the migrated database. In addition, it's frequently part of the process to spot-test expected queries against the migrated database to ensure consistency.
5. **Final migration and promotion.** The date and time has to be set and communicated for the production migration, generally when the application usage is low, for the application to experience downtime. At that point, the DMS migration job is executed. Once the continuous migration has caught up so that the lag between the source and Cloud SQL is minimal, the database can be promoted and the application can be cut over. The application is shut down, and any pending changes are migrated by the DMS to Cloud SQL. Promotion of the Cloud SQL instance is initiated, any outstanding validation is performed, and the application is cut over and restarted to run against the new Cloud SQL instance.
6. **Database tuning.** Once the application is running in Google Cloud and working against the new Cloud SQL instance, you can tune the database to further improve performance.

Migration planning is a detailed and multi-step process. While most frequently a migration will run without a hitch, it's generally good practice to plan for contingencies in case of additional time required for debugging (such as for establishing connectivity) or if a migration may need to be restarted.

Implementation, testing, execution, cutover

After assessment and planning is completed, implementation, testing migration and cutover can commence.

Implementation

The implementation consists of three resources that correspond to the systems involved.

- **Source connection profile.** Define a connection profile that represents the connectivity info of the source database, which will be used in the migration job. Note that migrations are frequently initiated directly against the primary database, but in the cases where the primary is load-sensitive, or many DDLs run on it, it's preferable to connect to a read replica.
- **Destination database.** The destination Cloud SQL instance is created during the flow of migration job creation, and a connection profile is automatically created for it in the back end when the instance is created to provide the destination of the migration job.
- **Migration job.** The migration job is specified either through the user interface (see screenshot above for an overview of the user interface flow) or using the API, utilizing the created connection profiles. If you use the user interface, you can copy the configuration values that you entered in case you need those again for another migration job specification. Most importantly, use the job testing feature as part of migration job creation to ensure a complete and consistent migration job implementation.
- **Limitations**: Currently the Database Migration Service does not migrate MySQL user management and permission management to the destination database. Users and permissions need to be manually set in the new Cloud SQL instance, and this can be done as soon as the destination database has been created. Learn more about migration limitations.

After the implementation is completed, you can begin testing.

Migration testing

Testing is a very important aspect of database migration to ensure that all aspects of the migration are taken care of, including application migration and application testing.

The best practice is to begin by running a migration job entirely for testing purposes. Start a migration job, and after the migration job enters the continuous replication (CDC) phase with minimal lag, promote the destination database and use it for testing the application in Google Cloud to ensure expected performance and results.

If any error occurs during the migration testing, analyze it and make the required changes, either to the source database or to the migration job. If a change was made, run a complete test again to ensure expected results.

Production migration

Once testing is completed, you can migrate the production database and application. At this point you need to finalize the day and time of production migration. Ideally, there is low application use at this time. In addition, all stakeholders that need to be involved should be available and ready.

Once the production migration starts, it requires close monitoring to ensure that it goes smoothly. The monitoring user interface in DMS is important during this phase to ensure replication lag is low at the time of promotion.

Once the migration is completed, validate that the destination database is complete and consistent in order to support the application.

Database and application cutover

It is a best practice to create a backup of the destination database as a consistent starting point for the new primary database before connecting any application.

Once you take the backup, promote the Cloud SQL database to be the new primary going forward. Cut over all dependent applications to access the new primary database, and open up the applications for production usage.

Once the application starts running on Cloud SQL, monitor the database performance closely to see if performance tuning is required. Since the application has never run before on Cloud SQL, there may be tuning options available that could optimize application performance, as shown here and here.

What's next

- Review the DMS Documentation

Try out DMS in the Google Cloud console. It's available at no additional charge for native lift-and-shift migrations to Cloud SQL