Database Migration

- [Cloning a Service to Another Provider or Region](#)
- [Database Migration Service for MySQL](#)
- [Manual Migration Using mysqldump and mysql](#)

Cloning a Service to Another Provider or Region

Migrating or cloning services across cloud providers or geographic regions is a critical part of modern infrastructure management. Whether you're optimizing for latency, preparing for disaster recovery, meeting regulatory requirements, or simply switching providers, a well-planned migration ensures continuity, performance, and data integrity. This guide outlines a structured methodology for service migration, applicable to most cloud-native environments.

Pre-Migration Preparation

Before initiating a migration, thorough planning and preparation are essential. This helps avoid unplanned downtime, data loss, or misconfiguration during the move:

- **Evaluate the Current Setup**: Begin by documenting the existing service's configuration. This includes runtime environments (container images, platform versions), persistent data (databases, object storage), network rules (ports, firewalls), and application dependencies (APIs, credentials, linked services).
- **Define the Migration Target**: Choose the new cloud provider or region you plan to migrate to. Confirm service compatibility, resource limits, and geographic latency requirements. If you're replicating an existing environment, make sure the target region supports the same compute/storage features and versions.
- **Provision the Target Environment**: Set up the target infrastructure where the service will be cloned. This could involve creating new Kubernetes clusters, VM groups, container registries, databases, or file storage volumes—depending on your stack.
- **Backup the Current Service**: Always create a full backup or snapshot of the current service and its associated data before proceeding. This acts as a rollback point in case of migration issues and ensures recovery in the event of failure.

Cloning Execution

The first step in executing a clone is to replicate the configuration of the original service in the target environment. This involves deploying the same container image or service binary using the same runtime settings. If you're using Kubernetes or container orchestrators, this can be done via Helm charts or declarative manifests. Pay close attention to environment variables, secrets,

mounted paths, storage class definitions, and health check configurations to ensure a consistent runtime environment.

Next, you'll need to migrate any persistent data tied to the service. For file-based storage, tools like rsync or rclone are effective for copying volume contents over SSH or cloud storage backends. It's crucial to verify compatibility across disk formats, database versions, and encoding standards to avoid corruption or mismatched behavior.

After replicating the environment and data, it's important to validate the new service in isolation. This means confirming that all application endpoints respond as expected, background tasks or cron jobs are functioning, and third-party integrations (e.g., payment gateways, S3 buckets) are accessible. You should test authentication flows, data read/write operations, and retry logic to ensure the new service is functionally identical. Use observability tools to monitor resource consumption and application logs during this stage.

Once validation is complete, configure DNS and route traffic to the new environment. This might involve updating DNS A or CNAME records, changing cloud load balancer configurations, or applying new firewall rules. For high-availability setups, consider using health-based routing or weighted DNS to gradually transition traffic from the old instance to the new one.

Post-Migration Validation and Optimization

Once the new environment is live and receiving traffic, focus on optimizing and securing the setup:

- **Validate Application Functionality**: Test all integrations, user workflows, and background jobs to confirm proper behavior. Review logs for silent errors or timeouts. Ensure all applications pointing to the service are updated with the new URL or connection string.
- **Monitor Performance**: Analyze load, CPU, memory, and storage utilization. Scale resources as needed, or optimize runtime settings for the new provider/region. Enable autoscaling where applicable.
- **Secure the Environment**: Implement firewall rules, IP restrictions, and access controls. Rotate secrets and validate that no hardcoded credentials or endpoints point to the old service.
- **Cleanup and Documentation**: Once validated, decommission the old setup safely. Update internal documentation with new deployment details, endpoint addresses, and any configuration changes.

Benefits of Cloning

Cloning a database service, particularly for engines like MySQL offers several operational and strategic advantages. It allows teams to test schema migrations, version upgrades, or major application features in an isolated environment without affecting production. By maintaining a cloned copy, developers and QA teams can work against realistic data without introducing risk.

Cloning also simplifies cross-region redundancy setups. A replica in another region can be promoted quickly if the primary region experiences an outage. For compliance or analytics purposes, cloned databases allow for read-only access to production datasets, enabling safe reporting or data processing without interrupting live traffic.

Additionally, rather than building a new environment from scratch, you can clone the database into another provider, validate it, and cut over with minimal disruption. This helps maintain operational continuity and reduces the effort needed for complex migrations.

Database Migration Service for MySQL

Elestio provides a structured approach for migrating MySQL databases from various environments, such as on-premises systems or other cloud platforms, to its managed services. This process ensures data integrity and minimizes downtime, facilitating a smooth transition to a managed environment.

Key Steps in Migrating to Elestio

Pre-Migration Preparation

Before initiating the migration process, it's essential to undertake thorough preparation to ensure a smooth transition:

- **Create an Elestio Account:** Register on the Elestio platform to access their suite of managed services. This account will serve as the central hub for managing your MySQL instance and related resources.
- **Deploy the Target MySQL Service:** Set up a new MySQL instance on Elestio to serve as the destination for your data. It's crucial to match the software version of your current MySQL database to avoid compatibility issues during data transfer. Detailed prerequisites and guidance can be found in Elestio's migration documentation.

Initiating the Migration Process

With the preparatory steps completed, you can proceed to migrate your MySQL database to Elestio:

1. **Access the Migration Tool:** Navigate to the overview of your MySQL service on the Elestio dashboard. Click on the "Migrate Database" button to initiate the migration process. This tool is designed to facilitate a smooth transition by guiding you through each step.

2. **Configure Migration Settings:** A modal window will open, prompting you to ensure that your target service has sufficient disk space to accommodate your database. Adequate storage is vital to prevent interruptions during data transfer. Once confirmed, click on the "Get started" button to proceed.

3. **Validate Source Database Connection:** Provide the connection details for your existing MySQL database, including:
   - **Hostname:** The address of your current database server.
   - **Port:** The port number on which your MySQL service is running (default is 3306).
   - **Database Name:** The name of the database you intend to migrate.
   - **Username:** The username with access privileges to the database.
   - **Password:** The corresponding password for the user.

   After entering these details, click on "Run Check" to validate the connection. This step ensures that Elestio can securely and accurately access your existing data. You can find these details under Database admin section under your deployed MySQL service.

Database Admin	Display your database credentials	Hide DB Credentials
Host	mysql-320dd1-u7774.vm.elestio.app	
Port	24306	
User	root	
Password	*********	Show password
CLI	mysql --host=mysql-320dd1-u7774.vm.elestio.app --port=24306 --user=root --password=*********	Show password

4. **Execute the Migration:** If all checks pass without errors, initiate the migration by selecting "Start migration." Monitor the progress through the real-time migration logs displayed on the dashboard. This transparency allows for immediate detection and resolution of any issues, ensuring data integrity throughout the process.

Post-Migration Validation and Optimization

After completing the migration, it's crucial to perform validation and optimization tasks to ensure the integrity and performance of your database in the new environment:

- **Verify Data Integrity:** Conduct thorough checks to ensure all data has been accurately transferred. This includes comparing row counts, checksums, and sample data between the source and target databases. Such verification maintains the reliability of your database and ensures that no data was lost or altered during migration.

- **Test Application Functionality:** Ensure that applications interacting with the database function correctly in the new environment. Update connection strings and configurations as necessary to reflect the new database location. This step prevents potential disruptions and ensures seamless operation of dependent systems.
- **Optimize Performance:** Utilize Elestio's managed service features to fine-tune database performance. Set up automated backups to safeguard your data, monitor resource utilization to identify and address bottlenecks, and configure scaling options to accommodate future growth. These actions contribute to improved application responsiveness and overall system efficiency.
- **Implement Security Measures:** Review and configure security settings to protect your data within the Elestio environment. Set up firewalls to control access, manage user access controls to ensure only authorized personnel can interact with the database, and enable encryption where applicable to protect data at rest and in transit. Implementing these security measures safeguards your data against unauthorized access and potential threats.

Benefits of Using Elestio for MySQL

Migrating your MySQL database to Elestio offers several advantages:

- **Simplified Management:** Elestio automates database maintenance tasks, including software updates, backups, and system monitoring, reducing manual work. The platform provides a dashboard with real-time insights into database performance and resource usage. It allows for adjusting service plans, scaling CPU and RAM as needed. Users can modify environment variables and access software information to manage configurations.

- **Security:** Elestio keeps MySQL instances updated with security patches to protect against vulnerabilities. The platform automates backups to ensure data integrity and availability. It provides secure access mechanisms, including randomly generated passwords for database instances, which can be managed through the dashboard.

- **Performance:** Elestio configures MySQL instances for performance based on workload requirements. The platform supports the latest MySQL versions, incorporating updates that improve database operations. Its infrastructure handles different workloads and maintains performance during high usage periods.

- **Scalability:** Elestio's MySQL service allows for scaling database resources to handle growth and changing workloads without major downtime. Users can upgrade or downgrade service plans, adjusting CPU and RAM as needed. The platform supports adding network volumes to increase storage capacity.

Manual Migration Using mysqldump and mysql

Manual migrations using MySQL's built-in tools `mysqldump` and `mysql`, are ideal for users who require full control over data export and import, particularly during transitions between providers, database version upgrades, or importing existing self-managed MySQL datasets into Elestio's managed environment. This guide walks through the process of performing a manual migration to and from Elestio MySQL services using command-line tools, ensuring data portability, consistency, and transparency at every step.

When to Use Manual Migration

Manual migration using mysqldump is well-suited for scenarios that demand complete control over the migration process. It is especially useful when transferring databases from a self-hosted MySQL instance, an on-premises server, or another cloud provider into Elestio's managed MySQL service. This method supports one-time imports without requiring ongoing connections between source and destination systems.

It also provides a reliable approach for performing version upgrades. Because mysqldump creates logical backups, the resulting SQL files can be restored into newer MySQL versions with minimal compatibility issues. When Elestio's built-in tools are not applicable such as in migrations from isolated environments or in selective schema transfers manual migration becomes the preferred option. It also enables offline backup archiving, providing users with transportable and restorable datasets independent of platform-specific backup formats.

Performing the Migration

Prepare the Environments

Before starting the migration, ensure that MySQL is properly installed on both the source system and your Elestio service. The source MySQL server must allow network connections (if remote) and have a user with sufficient privileges to export the database, including read access to all necessary tables, views, stored procedures, and triggers.

On the Elestio side, provision a MySQL service through the dashboard. Once it's active, retrieve the connection credentials from the **Database Info** section this includes host, port, database name, username, and password. Verify that your public IP is allowed under **Cluster Overview >**

Security > Limit access per IP, or the MySQL port will not be reachable.

Create a Dump Using mysqldump

Use mysqldump to export the entire source database into a SQL file. This utility serializes the schema, data, indexes, and routines into a plain-text SQL script.

```
mysqldump -h <source_host> -P <source_port> -u <source_user> -p <source_database> > backup.sql
```

You'll be prompted to enter the source password. Once complete, this produces a portable SQL file named backup.sql. You can also include flags such as `--routines` or `--triggers` if your database uses stored procedures or custom triggers.

To avoid restore-time permission issues, consider adding:

```
--skip-add-drop-table --skip-add-locks --set-gtid-purged=OFF
```

This is especially helpful when importing into a different environment or user setup than the source.

Transfer the Dump File to the Target

If your local system differs from the Elestio service access point, transfer the dump file using a secure file transfer tool:

```
scp backup.sql your_user@your_workstation:/path/to/local/
```

Ensure that the file is accessible on the system you plan to use to run the restore. The dump does not need to be uploaded to the Elestio server — restores are executed via a remote connection using MySQL's native protocol.

Create the Target Database

Elestio provisions a default database during setup. If the dump file references a different database name, you may need to create it manually before restore.

Connect to Elestio's MySQL service using the CLI:

```
mysql -h <elestio_host> -P <elestio_port> -u <elestio_user> -p
```

Once inside the MySQL shell, create the target database:

```
CREATE DATABASE target_database CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

This ensures proper character encoding and collation settings for compatibility with modern applications and multilingual content.

Restore Using MySQL Client

With the dump file in place and the target database created, restore the data:

```
mysql -h <elestio_host> -P <elestio_port> -u <elestio_user> -p target_database < /path/to/backup.sql
```

This command connects to the Elestio-managed MySQL instance and executes every statement from the dump file, recreating tables, inserting data, and applying schema-level objects.

Ensure the target user has privileges to create tables, insert data, and apply any stored routines or triggers included in the dump file.

Validate the Migration

After completing the import, verify that the migration was successful by connecting to the Elestio MySQL instance and running checks against key tables and schema components.

Start by listing the tables and checking row counts:

```
SHOW TABLES;
SELECT COUNT(*) FROM your_important_table;
```

Also review any stored procedures, views, or functions if your application depends on them. Confirm the database schema matches the original setup and that your application is able to connect and operate as expected.

If you've updated environment variables or connection strings, make sure these changes are reflected in your deployment or application config. Consider enabling automated backups via Elestio to protect your imported database going forward.

Benefits of Manual Migration

Manual MySQL migration using mysqldump and mysql offers several important advantages:

- **Portability and Compatibility**: Logical SQL dumps can be restored into any MySQL-compatible instance, whether hosted locally, on another cloud, or in containers.
- **Version Flexibility**: Migrate across MySQL versions without relying on binary replication or platform-specific formats.
- **Offline Storage**: SQL files serve as portable backups that can be stored offline, versioned, or archived for disaster recovery.

- **Platform Independence**: Elestio does not enforce proprietary formats standard MySQL tools give you complete control over the migration and restore process.

This method complements Elestio's automated backup and migration features by enabling custom workflows and one-off imports with full visibility into each stage.