

Elastically scaling your MySQL environment

This tutorial describes how to vertically scale (scale up and down) a high availability MySQL database cluster deployment (primary and replica database). This process includes scaling up and scaling down Compute Engine instances as well as scaling up their disks.

The usual reason for scaling up disk capacity is to accommodate an increase in data being managed.

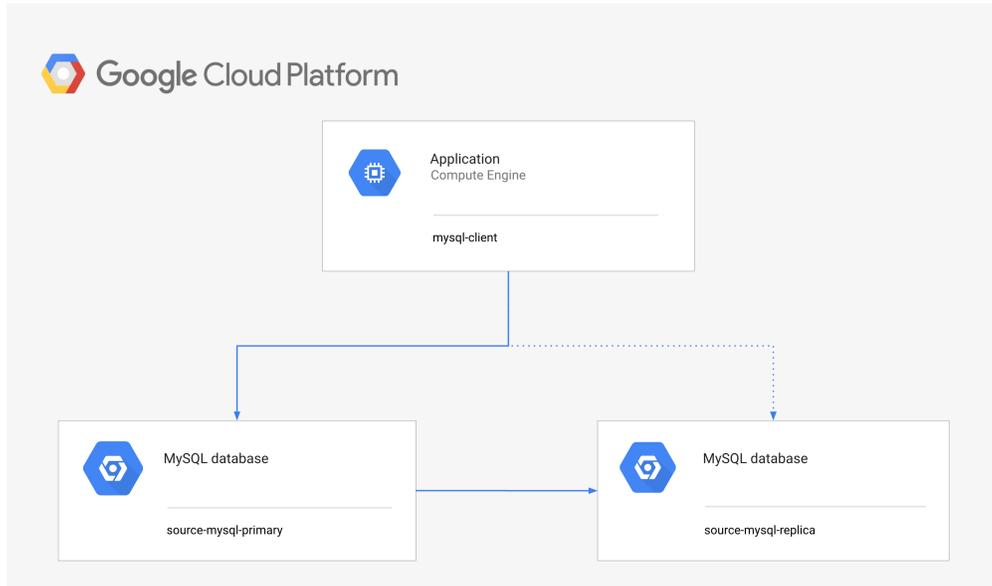
There are several reasons for vertically scaling Compute Engine instances that execute MySQL. Following are some reasons for scaling up (and the opposite reasons for scaling down):

- The system is reaching its write/read throughput performance limit. An increase in the number of CPUs and memory provides more hardware capacity.
- The number of queries is increasing over time, or spikes in the number of queries are expected (for example, during Black Friday or Cyber Monday). An increase in the number of CPUs and memory introduces a reserve.
- The number of concurrent number of queries is increasing—for example, by more clients being added to the system. An increase in the number of CPUs and memory supports a higher level of concurrency.
- Google Cloud might display the recommendation "increase perf" in the list of Compute Engine instances. That recommendation is important if you are reconsidering whether to scale up the Compute Engine instances.

The tutorial is useful for the following roles:

- Cloud architects who are planning the deployment of a MySQL cluster for scalability
- Cloud engineers who are implementing applications by using MySQL clusters
- Cloud operations teams that are managing MySQL clusters
- IT and database administrators who are managing databases in MySQL clusters and who have to execute a vertical scaling process (or execute several over time)

The following diagram shows the overall architecture of a highly available MySQL cluster. The tutorial uses this architecture as a basis for describing the vertical scaling process.



This tutorial assumes you are familiar with the following:

- Setting up and running a MySQL cluster by using Deployment Manager and various command-line tools such as Cloud Shell and `mysql`.
- Compute Engine instance management operations.
- Compute Engine disk management operations.

Objectives

- Set up a MySQL cluster with primary and replica databases.
- Vertically scale up all Compute Engine instances of the MySQL cluster (memory and CPU) by changing their machine type.
- Vertically scale down all Compute Engine instances of the MySQL cluster (memory and CPU) by changing their machine type.
- Increase the size of the Compute Engine instances' disks.

Costs

This tutorial uses the following billable components of Google Cloud:

- [Compute Engine \(/compute/pricing\)](/compute/pricing)
- [Cloud Storage \(/storage/pricing\)](/storage/pricing)

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

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

Before you begin

1. [Sign in \(https://accounts.google.com/Login\)](https://accounts.google.com/Login) to your Google Account.

If you don't already have one, [sign up for a new account \(https://accounts.google.com/SignUp\)](https://accounts.google.com/SignUp).

2. In the Cloud Console, on the project selector page, select or create a 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 \(https://console.cloud.google.com/projectselector2/home/dashboard\)](https://console.cloud.google.com/projectselector2/home/dashboard)

3. Make sure that billing is enabled for your Google Cloud project. [Learn how to confirm billing is enabled for your project \(/billing/docs/how-to/modify-project\)](/billing/docs/how-to/modify-project).

4. Enable the Compute Engine and Cloud Storage APIs.

[Enable the APIs \(https://console.cloud.google.com/flows/enableapi?apiid=compute,storage,deploymentmanager\)](https://console.cloud.google.com/flows/enableapi?apiid=compute,storage,deploymentmanager)

5. [Install and initialize the Cloud SDK \(/sdk/docs\)](/sdk/docs).

Setting up your MySQL cluster

Your first step is to create a running MySQL cluster. You populate this cluster with some data used for illustration and verification. For data verification, the tutorial provides instructions that query the MySQL primary and replica databases.

The following instructions for setting up a MySQL cluster are taken from a related tutorial, [Migrating a MySQL cluster to Compute Engine using HAProxy \(/solutions/migrating-mysql-cluster-compute-engine-haproxy\)](/solutions/migrating-mysql-cluster-compute-engine-haproxy), and are slightly modified here for convenience.

1. In the Cloud Console, open Cloud Shell:

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

2. Set an environment variable for the Cloud Storage bucket name:

```
GCS_BUCKET_NAME=${USER}-mysql-$(date +%s)
echo $GCS_BUCKET_NAME
```

3. Create the Cloud Storage bucket (multi-regional by default):

```
gsutil mb gs://${GCS_BUCKET_NAME}/
```

The bucket will hold creation scripts and startup scripts used for both the MySQL primary and the replica creation.

4. Clone the GitHub repository and retrieve the scripts that you use to set up the environment:

```
git clone https://github.com/GoogleCloudPlatform/solutions-compute-mysql-migration-ha
```



5. From the `mysql-migration` folder, run the initialization script to create a MySQL cluster of primary and replica Compute Engine instances:

```
cd mysql-migration
./run.sh ${DEVHELL_PROJECT_ID} ${GCS_BUCKET_NAME}
```

This script also creates a MySQL client Compute Engine instance.

6. Enable remote root access to the primary instance from the client instance:

- a. In the Cloud Console, go to the ****VM instances** page:

[GO TO VM INSTANCES](https://console.cloud.google.com/compute/instances) (https://console.cloud.google.com/compute/instances)

- b. In the row for the `source-mysql-primary` instance, click **SSH** to connect to a secure shell.

- c. When the secure shell is available, run the following command:

```
mysql -u root -psolution-admin
```

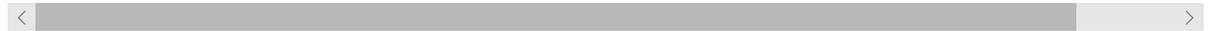
d. When you are logged into `mysql`, issue the following statement:

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'solution-admin';
```

7. Verify that the database is accessible from the client. In the Cloud Console, use **ssh** to connect to the `mysql-client` Compute Engine instance. When the secure shell is available, do the following:

a. Select **Compute Engine > VM Instances**, find the row with an instance called `source-mysql-primary`, and make note of the internal IP address:

```
mysql -u root -psolution-admin -h <var>internal-ip-address-of-source-mysql-primar
```



b. When the `mysql` shell is available, execute the following commands:

```
SHOW databases; # source_db must be present
USE source_db;
SHOW tables; # source_table must be present
SELECT COUNT(*) FROM source_table; # must return 5000
```

You can use the same set of commands to check that the replica contains the same data set: use the internal IP address of `source-mysql-replica` instead.

At this point, three Compute Engine instances are running:

- The client (`mysql-client`).
- The MySQL primary (`source-mysql-primary`).
- The MySQL replica (`source-mysql-replica`). The MySQL primary is replicating to the MySQL replica.

The machine type for each Compute Engine instance is `f1-micro` (1 vCPU, 0.6 GB memory), and the scale up is to the machine type `n1-standard-1` (1 vCPU, 3.75 GB memory). The disks' sizes are 10 GB and are doubled to 20 GB. These selections are examples only and can be changed to the particular needs of a deployment.

Vertically scaling up Compute Engine instances (without failover)

This section describes how to scale up the Compute Engine instances that are running the MySQL primary and replica. You scale up CPU and memory at the same time by changing the machine type of the

Compute Engine instance. In order to change the machine type, you have to stop the Compute Engine instance and, after the change, restart it.

To ensure equal processing capacity, we recommend that you configure both Compute Engine instances to use the same machine type.

The MySQL replica is scaled up first, and any problems discovered will not interrupt the execution of the MySQL primary. If a problem occurs, you can resolve it without primary downtime. Furthermore, you can assess if this problem was temporary or spurious or a general problem that you need to address before the primary database is going to be scaled up.

An alternative approach (that still requires restarting the Compute Engine instances) involves the failover of the primary to the secondary for minimizing downtime. You walk through this approach in the following sections.

Scale up the MySQL replica

First, stop the Compute Engine instance that's running the MySQL replica.

1. In the Cloud Console, go to the **VM instances** page to see the list of Compute Engine instances:

[LIST COMPUTE INSTANCES](https://console.cloud.google.com/compute/instances) (https://console.cloud.google.com/compute/instances)

2. In the row for the `source-mysql-replica` instance, click  (**Settings and utilities**), and then click **Stop**.
3. After the Compute Engine instance is stopped, click `source-mysql-replica`, and then click  **Edit**.
4. Under **Machine type**, select the machine type `n1-standard-1` (1 vCPU, 3.75 GB memory) to scale up to.
5. Click **Save**.
6. After the save completes, click  **Start**.

You can use the previously mentioned verification `mysql` commands to test whether the MySQL replica is up and running again after the scaling operation.

Scale up the MySQL primary

First, stop the Compute Engine instance running the MySQL primary.

1. In the Cloud Console, go to the **VM instances** page to see the list of Compute Engine instances:

[LIST COMPUTE INSTANCES](https://console.cloud.google.com/compute/instances) (https://console.cloud.google.com/compute/instances)

2. In the row for the `source-mysql-primary` instance, click  (**Settings and utilities**), and then click **Stop**.
3. After the Compute Engine instance has stopped, click `source-mysql-primary`, and then click  **Edit**.
4. Under **Machine type**, select the machine type `n1-standard-1` (1 vCPU, 3.75 GB memory) to scale up to. Ensure that this machine type is the same as selected for the MySQL replica.
5. Click **Save**.
6. After the save completes, click  **Start**.

You can use the previously mentioned verification `mysql` commands to test whether the MySQL primary is up and running again after the scaling operation.

Vertically scaling down Compute Engine instances (without failover)

This section describes how to scale down the Compute Engine instances that are running the MySQL primary and replica. You scale down CPU and memory at the same time by changing the machine type of the Compute Engine instance. In order to change the machine type, you have to stop the Compute Engine instance and after the change, restart it.

To ensure equal processing capacity, we recommend that you configure both Compute Engine instances to use the same machine type. The steps are analogous to those of scaling up. However, for the sake of completeness, the next section explicitly states them.

Scale down the MySQL replica

First, stop the Compute Engine instance that's running the MySQL replica.

1. In the Cloud Console, go to the **VM instances** page to see the list of Compute Engine instances:
[LIST COMPUTE INSTANCES](https://console.cloud.google.com/compute/instances) (<https://console.cloud.google.com/compute/instances>)
2. In the row for the `source-mysql-replica` instance, click  (**Settings and utilities**), and then click **Stop**.
3. After the Compute Engine instance has stopped, click `source-mysql-replica`, and then click  **Edit**.
4. Under **Machine type**, select the machine type `f1-micro` (1 vCPU, 0.6 GB memory) to scale down to.
5. Click **Save**.
6. After the save completes, click  **Start**.

You can use the previously mentioned verification `mysql` commands to test whether the MySQL replica is up and running again after the scaling operation.

Scale down the MySQL primary

First, stop the Compute Engine instance that's running the MySQL primary.

1. In the Cloud Console, go to the **VM instances** page to see the list of Compute Engine instances:

[LIST COMPUTE INSTANCES](https://console.cloud.google.com/compute/instances) (https://console.cloud.google.com/compute/instances)

2. In the row for the `source-mysql-primary` instance, click **⋮ (Settings and utilities)**, and then click **Stop**.
3. After the Compute Engine instance has stopped, click `source-mysql-primary`, and then click **Edit**.
4. Under **Machine type**, select the machine type `f1-micro` (1 vCPU, 0.6 GB memory) to scale down to. Ensure that this machine type is the same as the one you previously selected for the MySQL replica.
5. Click **Save**.
6. After the save completes, click **▶ Start**.

You can use the previously mentioned verification `mysql` commands to test whether the MySQL primary is up and running again after the scaling operation.

Vertically scaling up Compute Engine instances (with failover)

Shutting down, scaling up, and restarting a MySQL database might take too long in a production environment. The faster process is based on failover: You scale up the replica first, and as soon as it is up and running again, you stop the existing primary, and the replica becomes the (new) primary. The overall downtime is the time it requires to failover the MySQL database to the scaled-up replica.

On a high level, the process is as follows:

1. Scale up the replica by stopping it, changing its machine type, and restarting it.
2. Wait for the replica to catch up on the changes that took place in the primary during the scale up of the replica.
3. Stop the primary.
4. Wait for the replica to drain the replication log.
5. Make the replica the new primary.

6. Stop the primary (new replica).
7. Scale up the new replica.
8. Make it the new slave of the new primary.

After this process is complete, both MySQL systems are scaled up and are in a primary/replica relationship: the former primary is the new replica, and the former replica is the new primary. The commands are outlined in detail in the following sections.

A fallback is not necessarily required in the general case because both the primary and the replica are the same machine types with the same type and amount of disk space. A fallback would cause a brief outage for the duration of the falling back. However, if the fallback is required, you'd have to execute the failover steps a second time.

Scale up the existing MySQL replica

Scale up the replica as outlined in [Scale up the MySQL replica](#) (#scale-up-mysql-replica). During this time, the primary remains available uninterrupted.

Failover primary to the scaled-up replica

The following commands execute the failover from the primary to the replica.

1. In Cloud Shell, stop the primary so that no further updates are received:

```
gcloud compute instances stop source-mysql-primary --zone=us-east1-b;
```

It's not necessary to wait for the primary to be stopped before you continue with the next steps.

2. In the Cloud Console, go to the **VM instances** page:

[GO TO VM INSTANCES](https://console.cloud.google.com/compute/instances) (https://console.cloud.google.com/compute/instances)

3. In the row of the `source-mysql-replica` instance, click **SSH** to connect to the instance.
4. When the secure shell is available, start the `mysql` shell:

```
mysql -u root -psolution-admin
```

5. Check whether binary logging is enabled on the replica (must be **ON**):

```
SHOW VARIABLES LIKE 'log_bin';
```

6. Check whether log slave updates are disabled (must be OFF):

```
SHOW VARIABLES LIKE 'log_slave%';
```

7. Drain the relay log:

```
STOP SLAVE IO_THREAD;
```

8. Ensure that all processing took place:

```
SHOW PROCESSLIST;
```

The output of this command must show **Slave has read all relay log**. Keep executing the command until you see that result.

9. Stop the replica:

```
STOP SLAVE;
```

10. Change the role of the replica to primary:

```
RESET MASTER;  
GRANT REPLICATION SLAVE ON *.* TO 'sourcereplicator'@'%' IDENTIFIED BY 'solution-admi
```

<  >

The new primary is now in place.

Scale up the new MySQL replica

The former replica is now the primary, and clients can access it for read and write operations.

Scale up the replica (the former primary) by following the instructions as shown previously, and then start up the replica.

Connect the replica to the primary for replication

1. In the Cloud Console, go to the **VM instances** page:

GO TO VM INSTANCES (<https://console.cloud.google.com/compute/instances>)

2. In the row of the `source-mysql-primary` instance, click **SSH** to connect to a secure shell.
3. When the secure shell is available, start the `mysql` shell:

```
mysql -u root -psolution-admin
```

4. Start the replication:

```
CHANGE MASTER TO MASTER_HOST='source-mysql-replica', master_user='sourcereplicator',m
RESET SLAVE;
START SLAVE;
```



The primary MySQL instance is now replicating to its replica.

Test replication from the primary to the replica

The following test adds a row into the table `source_table` on the primary MySQL instance `source-mysql-replica`. You can observe the addition in the replica MySQL instance `source-mysql-primary`.

1. In the `source-mysql-replica` instance, add a row on the new primary. If no row was added before, the count must show `5001`.

```
USE source_db;
INSERT INTO source_table (event_data) VALUES (ROUND(RAND()*15000,2));
SELECT count(*) FROM source_table;
```

2. Observe replication on the replica. The count must show `5001`.

```
USE source_db;
SELECT count(*) FROM source_table;
```

This concludes the procedure required for the failover process. You use the same procedure for scaling down in conjunction with failover.

Increasing the size of the Compute Engine instances' disks

This section describes how to increase the size of a Compute Engine instance disk, both for the Compute Engine instance that's hosting the MySQL primary and for the Compute Engine instance that's hosting the MySQL replica. Disks can only be increased in size, not reduced.

For scaling up disks, there are two approaches, and both are outlined in the sections that follow. The ability to resize disks dynamically is a feature that doesn't require that you re-create the Compute Engine instances. For more details, see [this blog post](#)

([/blog/products/gcp/introducing-google-cloud-persistent-disks-with-non-disruptive-online-resizing](#)). One approach is stopping the Compute Engine instances before increasing the disk size and then restarting them. Restarting the instances automatically resizes the root partition that stores the MySQL data files.

The alternative approach does not require stopping and restarting the Compute Engine instances. Instead, it requires that you execute command-line statements in Cloud Shell and in the secure shells of the instances.

For confirmation, you can use the command `df -h --total` before and after the disk size increase to check the size before and after.

We recommend the best practice of taking a snapshot of each of the disks before you resize them. This precaution ensures that you can resurrect the state of each disk from before the resizing.

Increase the disk size of the MySQL replica (with shutdown)

First, increase the size of the disk of the Compute Engine instance that's hosting the MySQL replica.

1. In the Cloud Console, go to the **VM instances** page to see the list of Compute Engine instances:

[LIST COMPUTE INSTANCES](https://console.cloud.google.com/compute/instances) (<https://console.cloud.google.com/compute/instances>)

2. In the row for the `source-mysql-replica` instance, click **⋮ (Settings and utilities)**, and then click **Stop**.
3. List the disks of the Compute Engine instances:

[LIST DISKS OF INSTANCES](https://console.cloud.google.com/compute/disks) (<https://console.cloud.google.com/compute/disks>)

4. Select `source-mysql-replica`.
5. Click ** Edit**.
6. For **Size**, increase the size to 20 GB.
7. Click **Save** and wait for the save operation to complete.
8. List the Compute Engine instances:

[LIST COMPUTE INSTANCES](https://console.cloud.google.com/compute/instances) (https://console.cloud.google.com/compute/instances)

9. In the row for the `source-mysql-replica` instance, click  (**Settings and utilities**), and then click **Start**.

You can use the previously mentioned verification `mysql` commands to verify whether the MySQL primary is running as expected after the disk size increase.

Increase the disk size of the MySQL primary (with shutdown)

Increase the size of the disk of the Compute Engine that's hosting the MySQL primary.

1. In the Cloud Console, go to the **VM instances** page to see the list of Compute Engine instances:

[LIST COMPUTE INSTANCES](https://console.cloud.google.com/compute/instances) (https://console.cloud.google.com/compute/instances)

2. In the row for the `source-mysql-primary` instance, click  (**Settings and utilities**), and then click **Stop**.
3. List the disks of the Compute Engine instances:

[LIST DISKS OF INSTANCES](https://console.cloud.google.com/compute/disks) (https://console.cloud.google.com/compute/disks)

4. Select `source-mysql-primary`.
5. Click  **Edit**.
6. For **Size**, increase the size to 20 GB.
7. Click **Save** and wait for the save operation to complete.
8. List the Compute Engine instances:

[LIST COMPUTE INSTANCES](https://console.cloud.google.com/compute/instances) (https://console.cloud.google.com/compute/instances)

9. In the row for the `source-mysql-primary` instance, click  (**Settings and utilities**), and then click **Start**.

You can use the previously mentioned verification `mysql` commands to verify whether the MySQL primary is running as expected after the disk size increase.

Increase the disk size of the MySQL replica (dynamically without shutdown)

The following steps show the dynamic disk size increase for the file system `ext4` and a volume with a single partition. Other file system types or partition configurations require different steps in order to accomplish the increase.

As earlier, you first increase the disk size of the Compute Engine instance that's hosting the replica, and then you increase the disk size of the Compute Engine instance that's hosting the primary.

1. In the Cloud Console, go to the **VM instances** page to see the list of Compute Engine instances:

[LIST COMPUTE INSTANCES](https://console.cloud.google.com/compute/instances) (<https://console.cloud.google.com/compute/instances>)

2. Click **SSH** to connect to the `source-mysql-replica` instance.
3. In the secure shell, check the disk and its partitioning, and observe that the disk `sda` has one partition `sda1`:

```
lsblk
```

4. In Cloud Shell, run the following command to increase the disk size. When prompted, respond with `y`.

```
gcloud compute disks resize source-mysql-replica --size=20G --zone=us-east1-c
```

5. In the secure shell, confirm that the disk size increased:

```
lsblk
```

Note also that the partition is still 10 GB in size.

6. In the secure shell, run the following command to see the file systems, their type, and their size.

```
df -Th
```

7. In the secure shell, grow the partition:

```
sudo growpart /dev/sda 1
sudo resize2fs /dev/sda1
lsblk
df -Th
```

The last two commands allow you to see the increase.

Increase the disk size of the MySQL primary (dynamically without shutdown)

The process for dynamically increasing the disk size for the primary is the same as for the replica.

1. In the Cloud Console, go to the **VM instances** page to see the list of Compute Engine instances:

[LIST COMPUTE INSTANCES](https://console.cloud.google.com/compute/instances) (<https://console.cloud.google.com/compute/instances>)

2. Click **SSH** to connect to the `source-mysql-primary` instance.
3. In the secure shell, check the disk and its partitioning, and observe that the disk `sda` has one partition `sda1`:

```
lsblk
```

4. In Cloud Shell, run the following command to increase the disk size. When prompted, respond with `y`.

```
gcloud compute disks resize source-mysql-primary --size=20G --zone=us-east1-b
```

5. In the secure shell, confirm that the disk size increased:

```
lsblk
```

Note also that the partition is still 10 GB in size.

6. In the secure shell, run the following command to see the file systems, their type, and their size.

```
df -Th
```

7. In the secure shell, grow the partition:

```
sudo growpart /dev/sda 1
sudo resize2fs /dev/sda1
lsblk
df -Th
```

The last two commands allow you to see the increase.

Cleaning up

After you've finished the current tutorial, you can clean up the resources that you created on Google Cloud so they won't take up quota and you won't be billed for them in the future. The following sections describe how to delete or turn off these resources.

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.

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

[Go to the Manage resources page](https://console.cloud.google.com/iam-admin/projects) (<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

You can apply what you learned in this tutorial to MySQL clusters in a production environment to establish a process and routine for when scaling is required. In order to practice the content first, clone the production MySQL cluster environment and go through a dry run. Take note of any significant steps that might affect subsequent vertical scaling changes in your production environment.

Consider developing scripts that execute the steps shown in this tutorial. This way, in your production environment, you can scale automatically instead of scaling manually.

For further reading, see these [MySQL tutorials](#) (/docs/tutorials#mysql).

Try out other Google Cloud features for yourself. Have a look at our [tutorials](#) (/docs/tutorials).

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](#)

(<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](#) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2020-06-16.