Cloud SQL (https://cloud.google.com/sql/)
Documentation (https://cloud.google.com/sql/docs/)
MySQL (https://cloud.google.com/sql/docs/mysql/) Guides

Diagnosing issues with Cloud SQL instances

MySQL | PostgreSQL (https://cloud.google.com/sql/docs/postgres/diagnose-issues) | SQL Server

This page contains a list of the most frequent issues you might run into when working with Cloud SQL instances and steps you can take to address them. You should also review the Known Issues (https://cloud.google.com/sql/docs/mysql/known-issues) page. If the information here does not solve your issue, see the Support Overview (https://cloud.google.com/sql/docs/mysql/support) for getting further help.

Viewing logs

To see information about recent operations, you can view the Cloud SQL instance operation logs (https://cloud.google.com/sql/docs/mysql/instance-info#logs) or the MySQL error logs (https://cloud.google.com/sql/docs/mysql/instance-info#mysqllogs).

Instance unresponsive

If your instance stops responding to connections or performance is degraded, make sure it conforms to the Operational Guidelines (https://cloud.google.com/sql/docs/mysql/operational-guidelines). If it does not conform to these guidelines, it is not covered by the Cloud SQL SLA (https://cloud.google.com/sql/sla).

Connection issues

Verify that your application is closing connections properly

If you see errors containing "`Aborted connection nnnn to db:`", it usually indicates that your application is not terminating connections properly. It could also be caused by network issues. This error does not mean that there are problems with your Cloud SQL instance.

For examples of best practices for connection management, see Managing connections (https://cloud.google.com/sql/docs/mysql/manage-connections).

Verify that your certificates have not expired

If your instance is configured to use SSL, go to the Cloud SQL Instances page (https://console.cloud.google.com/sql/instances) in the Cloud Console and open the instance. Open its **Connections** page and make sure that your server certificate is valid. If it has expired, you must add a new certificate and rotate to it. Learn more (https://cloud.google.com/sql/docs/mysql/configure-ssl-instance#add-cert).

For First Generation instances, you should also check the expiration date of your client certificates, shown on the **Connections** page. If they have expired, you must create new client certificates (https://cloud.google.com/sql/docs/mysql/configure-ssl-instance#client-certs) before you can connect to your instance using SSL.

Verify that you are authorized to connect

If your connections are failing, check that you are authorized to connect:

- If you are connecting from App Engine standard environment to a First Generation instance, ensure that the App Engine application is authorized to connect (https://cloud.google.com/sql/docs/mysql/connect-app-engine) to the Cloud SQL instance.

★ **Note:** Second Generation is replacing First Generation; support for First Generation instances ends January 30, 2020. To upgrade a First Generation instance to Second Generation, see Upgrading a First Generation Instance to Second Generation (https://cloud.google.com/sql/docs/mysql/upgrade-2nd-gen).

- If you are having trouble connecting using an IP address, for example, you are connecting from your on-premises environment with the mysql client, then make sure that the IP address you are connecting from is authorized to connect (https://cloud.google.com/sql/docs/mysql/configure-ip) to the Cloud SQL instance. Here's your current IP address (http://www.google.com#q=whats+my+ip).
- Try the `gcloud sql connect` (https://cloud.google.com/sdk/gcloud/reference/sql/connect) to connect to your instance. This command authorizes your IP address for a short period of time. You can run this in an environment with Cloud SDK and mysql client installed. You

can also run this command in Cloud Shell (https://cloud.google.com/shell/docs/), which is available in the Google Cloud Console and has Cloud SDK and the mysql client pre-installed. Cloud Shell provides a Compute Engine instance that you can use to connect to Cloud SQL.

- Temporarily allow all IP addresses to connect to an instance. For IPv4 authorize `0.0.0.0/0` (for IPv6, authorize `::/0`. This confirms that your client can connect.

Note: Authorizing all IP addresses opens your database to any client that tries to connect. If you have sensitive or proprietary data in your database, you should not use this method to investigate connectivity issues.

Verify how you connect

If you get an error message like:

```
ERROR 1045 (28000): Access denied for user 'root'@'1.2.3.4' (using password: NO)
```

when you connect, verify that you are providing a password.

If you get an error message like:

```
ERROR 1045 (28000): Access denied for user 'root'@'1.2.3.4' (using password: YES)
```

when you connect, verify that you are using the correct password and that you are connecting over SSL if the instance requires it.

Determining how connections are being initiated

You can see information about your current connections by running the following command:

```
SHOW PROCESSLIST;
```

Connections that show an IP address, such as `1.2.3.4`, are connecting using IP. Connections with `cloudsqlproxy~1.2.3.4` are using the Cloud SQL Proxy, or else they originated from App Engine. Connections from `localhost` are usually to a First Generation instance from App Engine, although that path is also used by some internal Cloud SQL processes.

Note: Second Generation is replacing First Generation; support for First Generation instances ends January 30, 2020. To upgrade a First Generation instance to Second Generation, see Upgrading a First Generation Instance to Second Generation (https://cloud.google.com/sql/docs/mysql/upgrade-2nd-gen).

Understand connection limits

There are no QPS limits for Cloud SQL instances. However, there are connection, size, and App Engine specific limits in place. See Quotas and Limits (https://cloud.google.com/sql/docs/mysql/quotas).

Database connections consume resources on the server and the connecting application. Always use good connection management practices to minimize your application's footprint and reduce the likelihood of exceeding Cloud SQL connection limits (https://cloud.google.com/sql/docs/mysql/quotas#fixed-limits). For more information, see Managing database connections (https://cloud.google.com/sql/docs/mysql/manage-connections).

Show connections and threads

If you get the “too many connections” error message or want to find out what is happening on an instance, you can show the number of connections and threads with `SHOW PROCESSLIST`.

From a MySQL client (https://cloud.google.com/sql/docs/mysql/connect-admin-ip), run:

```
mysql> SHOW PROCESSLIST;
+----+-----------+--------------+-----------+---------+------+-------+--------------
| Id | User      | Host         | db        | Command | Time | State | Info
+----+-----------+--------------+-----------+---------+------+-------+--------------
|  3 | user-name | client-IP    | NULL      | Query   |    0 | NULL  | SHOW processl
|  5 | user-name | client-IP    | guestbook | Sleep   |    1 |       | SELECT * from
| 17 | user-name | client-IP    | employees | Query   |    0 | NULL  | SHOW processl
+----+-----------+--------------+-----------+---------+------+-------+--------------
3 rows in set (0.09 sec)
```

For information about how to interpret the columns returned from `PROCESSLIST`, see the MySQL reference (https://dev.mysql.com/doc/refman/5.7/en/show-processlist.html).

To get a quick thread count, you can use:

```
mysql> SHOW STATUS WHERE Variable_name = 'Threads_connected';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Threads_connected | 7     |
+-------------------+-------+
1 row in set (0.08 sec)
```

Connections from Compute Engine

If you expect that connections between your Compute Engine instance and your Cloud SQL instance will include long-lived unused connections, then you should be aware that connections with a Compute Engine instance time out after 10 minutes of inactivity. For more information, see Networking and Firewalls (https://cloud.google.com/compute/docs/networks-and-firewalls) in the Compute Engine documentation.

To keep long-lived unused connections alive, you can set the TCP keepalive (http://tldp.org/HOWTO/TCP-Keepalive-HOWTO/usingkeepalive.html). The following commands set the TCP keepalive value to one minute and make the configuration permanent across instance reboots.

```
# Display the current tcp_keepalive_time value.
cat /proc/sys/net/ipv4/tcp_keepalive_time

# Set tcp_keepalive_time to 60 seconds and make it permanent across reboots.
echo 'net.ipv4.tcp_keepalive_time = 60' | sudo tee -a /etc/sysctl.conf

# Apply the change.
sudo /sbin/sysctl --load=/etc/sysctl.conf

# Display the tcp_keepalive_time value to verify the change was applied.
cat /proc/sys/net/ipv4/tcp_keepalive_time
```

Connecting with IPv6

If you get either of the error messages

```
Can't connect to MySQL server on '2001:1234::4321' (10051)
Can't connect to MySQL server on '2001:1234::4321' (101)
```

when you connect it is likely that you are attempting to connect to the IPv6 address of your instance but do not have IPv6 available on your workstation. You can verify whether IPv6 is functional on your workstation by going to ipv6.google.com (https://ipv6.google.com/), if it does not load then you do not have IPv6 available. To fix this you should connect to the IPv4 address or your Cloud SQL instance. You may need to add an IPv4 address to your instance (https://cloud.google.com/sql/docs/mysql/configure-ip) first.

Connection times out

If a client cannot connect to the Cloud SQL instance using private IP, check to see if the client is using any IP in the range 172.17.0.0/16. Connections from any IP within the 172.17.0.0/16 range to Cloud SQL instances using private IP will fail. Similarly, Cloud SQL instances created with an IP in that range will be unreachable. This range is reserved for the docker bridge network.

Occasional connection failures

When Cloud SQL restarts an instance due to maintenance events, connections might be routed to the failover replica. When connecting to the failover replica:

- Read requests from clients using unencrypted connections will succeed as normal. However, write requests will fail and return an error message, such as 'Error 1290: The MySQL server is running with the –read-only option so it cannot execute this statement.'
- Read and write requests from clients using encrypted connections will fail and return an error message, such as 'x509: certificate is valid for master-instance, not failover-instance.'

After the event is over, Cloud SQL should reset the connection. You should retry the connection. We recommend that you design your applications to handle occasional connection failures by implementing an error handling strategy like exponential backoff. See Application implementation (https://cloud.google.com/sql/docs/mysql/best-practices#app) for more information.

Instance issues

Backup

Backups (https://cloud.google.com/sql/docs/mysql/backup-recovery/backups) for First Generation instances can fail if you have more than 10,000 database tables. For best performance, keep your number of tables to a reasonable number.

Import and export

Imports and Exports (https://cloud.google.com/sql/docs/mysql/import-export) in Cloud SQL are the same as using the mysqldump utility except that with the Cloud SQL import/export feature, you transfer data using a Cloud Storage bucket. You can import and export one database, all instance databases, or selected data in CSV format.

Imports and exports into Cloud SQL using the import functionality (via a Cloud Storage bucket) can take a long time to complete, depending on the size of the database. This can have the following impacts:

- On First Generation instances, operations are limited to 24 hours.

★ **Note:** Second Generation is replacing First Generation; support for First Generation instances ends January 30, 2020. To upgrade a First Generation instance to Second Generation, see Upgrading a First Generation Instance to Second Generation (https://cloud.google.com/sql/docs/mysql/upgrade-2nd-gen).

- You cannot stop a long-running operation.

In addition, you can perform only one import or export operation at a time for each instance.

You can decrease the amount of time a single operation requires by using one of the following approaches:

- Use the Cloud SQL import or export functionality, but do it with smaller batches of data that will take less than 24 hours to complete.
- Don't use the Cloud SQL import or export functionality, but instead replay a dump file directly to Cloud SQL. For example, you can use `cloudsql-import` (https://github.com/GoogleCloudPlatform/cloudsql-import), which imports by replaying a mysqldump file over a MySQL connection. `cloudsql-import` is resilient to connection failures and instance restarts.

Other points to keep in mind when importing:

- If your import is crashing, it could be due to an out-of-memory (OOM) error. If this is the case, you can try adding the `--extended-insert=FALSE --complete- insert` parameters. This reduces the speed of your import, but it also reduces the amount of memory it requires.

Disk space

If your instance reaches the maximum storage amount allowed, writes to the database fail. If you delete data, for example, by dropping a table, the space freed is not reflected in the reported **Storage Used** of the instance. See the FAQ How can I reclaim the space from a dropped table? (https://cloud.google.com/sql/faq#reclaimingspace) for an explanation of this behavior.

Reaching the maximum storage limit can also cause the instance to get stuck in restart.

Avoid data corruption

Avoid generated columns

Due to an issue in MySQL, using generated columns might result in data corruption. For more information, see MySQL bug #82736 (https://bugs.mysql.com/bug.php?id=82736).

Clean shutdowns

When Cloud SQL shuts down an instance (e.g, for maintenance), no new connections are sent to the instance and existing connections are killed. The amount of time mysqld has to shutdown is capped to 1 minute. If the shutdown does not complete in that time, the mysqld process is forcefully terminated. This can result in disk writes being aborted mid-way through.

Database engines

InnoDB is the only supported storage engine for Second Generation instances because it is more resistant to table corruption than other MySQL storage engines, such as MyISAM (https://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html).

By default, Cloud SQL database tables are created using the InnoDB storage engine. If your `CREATE TABLE` syntax includes an `ENGINE` option specifying a storage engine other than InnoDB, for example `ENGINE = MyISAM`, the table is not created and you see error messages like the following example:

```
ERROR 3161 (HY000): Storage engine MyISAM is disabled (Table creation is disallowed)
```

You can avoid this error by removing the `ENGINE = MyISAM` option from the `CREATE TABLE` command. Doing so creates the table with the InnoDB storage engine.

InnoDB is strongly recommended for First Generation instances, because of its stronger data consistency guarantees.

Changes to system tables

MySQL system tables use the MyISAM storage engine, including all tables in the `mysql` database, for example `mysql.user` and `mysql.db`. These tables are vulnerable to unclean shutdowns; you should issue the `FLUSH CHANGES` command after making any changes to these tables. If MyISAM corruption does occur, `CHECK TABLE` and `REPAIR TABLE` can get you back to good state (but not save data).

Global Transaction Identifiers (GTID)

All Second Generation instances have GTID enabled automatically. Having GTID enabled protects against data loss during replica creation and failover, and makes replication more robust. However, GTID comes with some limitations imposed by MySQL, as documented in the MySQL manual (https://dev.mysql.com/doc/refman/5.7/en/replication-gtids-restrictions.html). The following transactionally-unsafe operations cannot be used with a GTID-enabled MySQL server:

- `CREATE TABLE ... SELECT` statements;
- `CREATE TEMPORARY TABLE` statements inside transactions;
- Transactions or statements that affect both transactional and non-transactional tables.

If you use a transactionally-unsafe transaction, you will see an error message like the following example:

```
Exception: SQLSTATE[HY000]: General error: 1786
CREATE TABLE ... SELECT is forbidden when @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1.
```

Working with triggers and stored functions

If your instance has binary logging enabled, and you need to work with triggers or stored functions, make sure your instance has the `log_bin_trust_function_creators` (https://dev.mysql.com/doc/refman/5.7/en/replication-options-binary-log.html#sysvar_log_bin_trust_function_creators) flag set to on (https://cloud.google.com/sql/docs/mysql/flags).

Suspended state

There are a number of reasons why Cloud SQL may suspend an instance, including:

- Billing issues

  For example, if the credit card for the project's billing account has expired, the instance may be suspended. You can check the billing information for a project by going to the Google Cloud Console billing page (https://console.cloud.google.com/billing), selecting the project, and viewing the billing account information used for the project. After you resolve the billing issue, the instance should return to runnable status within a few hours.

- KMS key issues

  For example, if the KMS key version used to encrypt the user data in the Cloud SQL instance is not present, or if it has been disabled or destroyed. See Using customer managed encryption keys (CMEK) (https://cloud.google.com/sql/docs/mysql/configure-cmek).

- Legal issues

  For example, a violation of the Google Cloud Acceptable Use Policy (https://cloud.google.com/terms/aup) may cause the instance to be suspended. For more information, see "Suspensions and Removals" in the Google Cloud Terms of Service (https://cloud.google.com/terms/).

- Operational issues

  For example, if an instance is stuck in a crash loop, i.e., it crashes while starting or just after starting, Cloud SQL may suspend it.

While an instance is suspended, you can continue to view information about it or you can delete it, if the suspension was triggered by billing issues.

Cloud SQL users with Platinum, Gold, or Silver support packages (https://cloud.google.com/support/) can contact our support team directly about suspended instances. All users can use the guidance above along with the google-cloud-sql (http://stackoverflow.com/questions/tagged/google-cloud-sql) forum.

Performance

Enable query logs

In order to tune the performance of your queries, you can configure Cloud SQL to log slow queries by adding the database flags (https://cloud.google.com/sql/docs/mysql/flags) `--log_output='FILE'` and `--slow_query_log=on` to your instance. This makes the log output available using the Logs Viewer in the Google Cloud Console (https://cloud.google.com/logging/docs/view/logs_viewer). Note that Stackdriver logging charges (https://cloud.google.com/stackdriver/pricing) apply.

Do not set log_output to `TABLE`. Doing so can cause connection issues as described in Tips for working with flags (https://cloud.google.com/sql/docs/mysql/flags#tips-general-log).

Enable lock monitoring

InnoDB monitors provide information about the InnoDB storage engine's internal state, which you can use in performance tuning.

Access the instance using MySQL Client and obtain on-demand monitor output:

```
SHOW ENGINE INNODB STATUS\G
```

For explanations of the sections in the monitor output, see InnoDB Standard Monitor and Lock Monitor Output (https://dev.mysql.com/doc/refman/5.7/en/innodb-standard-monitor.html).

You can enable InnoDB monitors so that output is generated periodically to a file or a table, with performance degradation. For more information, see Enabling InnoDB Monitors (https://dev.mysql.com/doc/refman/5.7/en/innodb-enabling-monitors.html).

Keep a reasonable number of database tables

Database tables consume system resources. A very large number can affect instance performance and availability, and cause the instance to lose its SLA coverage. Learn more (https://cloud.google.com/sql/docs/mysql/operational-guidelines).

General performance tips

- First Generation instances have a 10-GB limit on temporary space used by SQL operations. Some operations (for example, large aggregations with GROUP BY) could be impacted by this limit. In some cases, you can avoid hitting the temporary-space limit by writing the query differently or using different SQL syntax. Second Generation instances do not impose this limit.
- Make sure that your machine type is sufficiently large for the workload.

For slow database inserts, updates, or deletes, consider the following actions:

- For First Generation instances, file system replication asynchronous mode (an instance configuration setting) is much faster than synchronous mode. If you are using synchronous mode and your application can withstand some data risk then you should switch to asynchronous mode.
- Check the locations of the writer and database; sending data a long distance introduces latency.

For slow database selects, consider the following:

- Caching is extremely important for read performance. Compare the size of your data set to the size of RAM of your instance. Ideally, the entire data set should fit in 70% of the instance's RAM, in which case queries will not be constrained to IO performance. If this is not the case, consider increasing the size of your instance's tier.
- If your workload consists of CPU intensive queries (sorting, regexes, other complex functions), your instance might be throttled; increase the tier.
- Check the location of the reader and database - latency will affect read performance even more than write performance.
- Investigate non-Cloud SQL specific performance improvements, such as adding appropriate indexing, reducing data scanned, and avoiding extra round trips.

If you observe poor performance executing queries, use `EXPLAIN` (https://dev.mysql.com/doc/refman/5.7/en/explain.html) to identify where to:

- Add indexes to tables to improve query performance. For example, make sure every field that you use as a JOIN key has an index on both tables.
- Improve `ORDER BY` operations. If `EXPLAIN` shows "Using temporary; Using filesort" in the **Extra** column of the output, then intermediate results will be stored in a file that is then

sorted, which usually results in poor performance. In this case, take one of the following steps:

- If possible, use indexes rather than sorting. See ORDER BY Optimization (https://dev.mysql.com/doc/refman/5.7/en/order-by-optimization.html) for more information.
- Increase the size of the `sort_buffer_size` (https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_sort_buffer_size) variable for the query session.
- Use less RAM per row by declaring columns only as large as required.

Customer-managed encryption keys (CMEK)

Cloud SQL administrator operations, such as create, clone, or update, might fail due to KMS errors, and missing roles or permissions. Common reasons for failure include a missing KMS key version, a disabled or destroyed KMS key version, insufficient IAM permissions to access the KMS key version, or the KMS key version is in a different region than the Cloud SQL instance. Use the following troubleshooting table to diagnose and resolve common problems.

Customer-managed encryption keys troubleshooting table

For this error...	The issue might be...	Try this...
Per-product, per-project service account not found	The service account name is incorrect.	Make sure you created a service account for the correct user pr **GO TO THE SERVICE ACCOUNTS PAGE** (HTTPS://CONSOLE.CL .
Cannot grant access to the service account	The user account does not have permission to grant access to this key version.	Add the **Organization Administrator** role on your user or servic **GO TO THE IAM ACCOUNTS PAGE** (HTTPS://CONSOLE.CLOUD
KMS key	The key version is destroyed.	If the key version is destroyed, you cannot use it to encrypt or d

version is destroyed		
KMS key version is disabled	The key version is disabled.	Re-enable the KMS key version. GO TO THE CRYPTO KEYS PAGE (HTTPS://CONSOLE.CLOUD.(
Insufficient permission to use the KMS key	The `cloudkms.cryptoKeyEncrypterDecrypter` role is missing on the user or service account you are using to run operations on Cloud SQL instances, or the KMS key version doesn't exist.	Add the `cloudkms.cryptoKeyEncrypterDecrypter` role on GO TO THE IAM ACCOUNTS PAGE (HTTPS://CONSOLE.CLOUD If the role is already on your account, see Creating a key (https://cloud.google.com/sql/docs/mysql/configure-cmek#k
KMS key is not found	The key version does not exist.	Create a new key version. See Creating a key (https://cloud.goo note.
Cloud SQL instance and KMS key version are in different regions	The KMS key version and Cloud SQL instance must be in the same region. It does not work if the KMS key version is in a global region or multi-region.	Create a key version in the same region where you want to crea (https://cloud.google.com/sql/docs/mysql/configure-cmek#k

Note: If the instance is in a failed state during the `create` operation, you must delete it, add the role to the account you are using, and create a new instance with a active KMS key version.

Last updated November 19, 2019.