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 (/sql/docs/postgres/known-issues) page. If the information here does not solve your issue, see the Support Overview (/sql/docs/postgres/support) for getting further help.

To see information about recent operations, you can view the Cloud SQL instance operation logs (/sql/docs/postgres/instance-info#logs) or the PostgreSQL error logs (/sql/docs/postgres/instance-info#mysqllogs).

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 (/sql/docs/postgres/manage-connections).

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 (/sql/docs/postgres/configure-ssl-instance#add-cert).

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

- If you are having trouble connecting using an IP address, for example, you are connecting from your on-premises environment with the psql client, then make sure that the IP address you are connecting from is authorized to connect (/sql/docs/postgres/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` (/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 psql client installed. You can also run this command in Cloud Shell (/shell/docs/), which is available in the Google Cloud Console and has Cloud SDK and the psql 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 by authorizing `0.0.0.0/0`. 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.

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

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.

Second Generation is replacing First Generation; support for First Generation instances ends January 30, 2020. To de a First Generation instance to Second Generation, see Upgrading a First Generation Instance to Second Generatio docs/mysql/upgrade-2nd-gen).

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 (/sql/docs/postgres/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 (/sql/docs/postgres/quotas#fixed-limits). For more information, see Managing database connections (/sql/docs/postgres/manage-connections).

To see the processes that are running on your database, use the pg_stat_activity (https://www.postgresql.org/docs/9.6/static/monitoring-stats.html#PG-STAT-ACTIVITY-VIEW) table:

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 (/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.

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.

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 (/sql/docs/mysql/best-practices#app) for more information.

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 is usually freed, but it is not reflected in the reported **Storage Used** of the instance. You can run the `VACUUM FULL` command to recover unused space; note that write operations are blocked while the vacuum command is running. Learn more (https://www.postgresql.org/docs/9.6/static/routine-vacuuming.html).

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) (/sql/docs/postgres/configure-cmek).

- Legal issues

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

Database schemas and tables consume system resources. A very large number can affect instance performance.

Make sure that your instance is not constrained on memory or CPU. For performance-intensive workloads, your instance should have at least 60 GB of memory.

For slow database inserts, updates, or deletes, 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. Check the various `blks_hit / (blks_hit + blks_read)` ratios from the PostgreSQL Statistics Collector (https://www.postgresql.org/docs/9.6/static/monitoring-stats.html). Ideally, the ratio should be above 99%. If this is not the case, consider increasing the size of your instance's RAM.
- If your workload consists of CPU intensive queries (sorting, regexes, other complex functions), your instance might be throttled; add vCPUs.
- 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://www.postgresql.org/docs/9.6/static/sql-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.

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

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 project. GO TO THE SERVICE ACCOUNTS PAGE (https://console.cloud.goog .
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 service acco GO TO THE IAM ACCOUNTS PAGE (https://console.cloud.google.co
Cloud KMS key version is destroyed	The key version is destroyed.	If the key version is destroyed, you cannot use it to encrypt or decrypt
Cloud KMS key version is disabled	The key version is disabled.	Re-enable the Cloud KMS key version. GO TO THE CRYPTO KEYS PAGE (https://console.cloud.google.com
Insufficient permission to use the Cloud 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 Cloud KMS key version doesn't exist.	Add the `cloudkms.cryptoKeyEncrypterDecrypter` role on your u GO TO THE IAM ACCOUNTS PAGE (https://console.cloud.google.co If the role is already on your account, see Creating a key (/sql/docs/p learn how to create a new key version. See note.
Cloud KMS key is not found	The key version does not exist.	Create a new key version. See Creating a key (/sql/docs/postgres/cor
Cloud SQL instance and Cloud KMS key version are in different regions	The Cloud KMS key version and Cloud SQL instance must be in the same region. It does not work if the Cloud KMS key version is in a global region or multi-region.	Create a key version in the same region where you want to create inst (/sql/docs/postgres/configure-cmek#key). See 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 eate a new instance with a active Cloud KMS key version.