

Google Cloud Platform

Cloud SQL Collector

26.05.2022

Get whatever you can imagine

Juan Carlos Cruz

Telus International

San Salvador

El Salvador, CA

Overview

Cloud SQL Collector is a Cloud Function (1er Generation) built upon Python 3.9 which operates over Cloud SQL Admin API calls and Cloud SQL direct connection, enabling your team to gather the following Cloud SQL Instances information out-of-the-box:

Cloud SQL Admin API Calls method:

- Cloud SQL Instances details (project name, instance name, region, status)
- Databases Details (name)
- Login Details (name)

Cloud SQL direct connection method:

- Databases Details (Last update date, Size)
- Databases Tables Size

Creating and Deploying Cloud SQL Collector

This manual shows you how to create and deploy it using the Google Cloud Console.

Before you begin

1. Sign in to the Google Cloud console using your Google Cloud account.
2. Enable the Cloud Functions, Cloud Build and Cloud SQL Admin APIs.
3. Prepare your development environment.

Get the Cloud SQL Collector code

1. Clone the GCPCloudSQLCollector repository to your local machine:
 - a. git clone <https://github.com/jotaccruz/GCPCloudSQLCollector.git>
2. Change to the directory that contains the Cloud SQL Collector code:
 - a. cd GCPCloudSQLCollector/
3. Take a look at the code.

Create the Cloud SQL Collector Service Account

1. To create the service account, follow the following steps:

Create service account

1

Service account details

Service account name

cloudsqlcollector

Display name for this service account

Service account ID *

cloudsqlcollector

X ↺

Email address:

cloudsqlcollector@[REDACTED].iam.gserviceaccount.com

📧

Service account description

Cloud SQL Collector Function Service Account

Describe what this service account will do

CREATE AND CONTINUE

Leave the other options default values, Service Account created.

2. Grant IAM necessary roles to operate:

| | | | | |
|--------------------------|--|--|-------------------|-------------------------|
| <input type="checkbox"/> | | cloudsqlcollector@[REDACTED] 01.iam.gserviceaccount.com | cloudsqlcollector | Cloud SQL Editor |
| | | | | Cloud SQL Instance User |
| | | | | Storage Admin |

Create the Cloud SQL Collector Config Database

Cloud SQL Collector Function stores all its configurations in a Cloud SQL MySQL database, we need to have a Cloud SQL Instance ready to use, where this database will reside.

1. Adding to the Cloud SQL Instance Logins the Service Account.

Users




All instances > cloudsqlcollectors

✓ cloudsqlcollectors

MySQL 8.0

User accounts enable users and applications to connect to your instance. [Learn more](#)

[+ ADD USER ACCOUNT](#)

| | User name  | Host name | Authentication | |
|---|---|--------------|----------------|---|
|  | root | % (any host) | Built-in |  |

Add a user account to instance cloudsqlcollectors

Choose how to authenticate

You can manage access to this instance using Cloud IAM or MySQL built-in authentication. [Learn more](#)

- ☐ Built-in authentication
- Creates a new username and password specific to this instance. User account will have cloudsqlsuperuser root access, but you can customize that later as needed. [Learn more](#)
- ☒ Cloud IAM
- Associates an existing IAM principal with this user account. Must have a role providing instance-level access assigned to connect.

Principal *

After you create a user account with Cloud IAM authentication, it will have no database privileges, so make sure permissions are granted as needed. [Learn more](#)

[ADD](#)

[CANCEL](#)

Users

All instances > cloudsqlcollectors

✓ cloudsqlcollectors

MySQL 8.0

User accounts enable users and applications to connect to your instance. [Learn more](#)

[+](#) ADD USER ACCOUNT

| | User name ↑ | Host name | Authentication | IAM Email ID | |
|--|-------------------|--------------|-----------------------|---|---|
| | cloudsqlcollector | % (any host) | IAM (service account) | cloudsqlcollector@[REDACTED]iam.gserviceaccount.com | ⋮ |
| | root | % (any host) | Built-in | | ⋮ |

2. Creating the database “cloudsqlcollector”

Create a database

Database Name *

cloudsqlcollector

Must follow the MySQL identifier rules. [Learn more](#)

Character set *

utf8

Can be changed later by executing an ALTER DATABASE query.

Collation

Default collation

Can be changed later by executing an ALTER DATABASE query.

CREATE

CANCEL

Databases

All instances > cloudsqlcollectors

✓ cloudsqlcollectors

MySQL 8.0

[+ CREATE DATABASE](#)

| Name ↑ | Collation | Character set | Type | |
|--------------------|--------------------|---------------|--------|---|
| cloudsqlcollector | utf8_general_ci | utf8 | User | ⋮ |
| information_schema | utf8_general_ci | utf8 | System | ⋮ |
| mysql | utf8_general_ci | utf8 | System | ⋮ |
| performance_schema | utf8mb4_0900_ai_ci | utf8mb4 | System | ⋮ |
| sys | utf8mb4_0900_ai_ci | utf8mb4 | System | ⋮ |

3. Granting cloudsqlcollector database, read/write access to the Service Account User executing the next statement.

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, ALTER, EXECUTE,
CREATE VIEW, SHOW VIEW ON `cloudsqlcollector`.* TO 'cloudsqlcollector'@'%'
```

4. Setting up the initial configuration.

Run all the statements in the file `init_database.sql` against the database `cloudsqlcollector`.

Create a bucket

Cloud SQL Collector will generate csv formatted files and it will be saving them in a bucket, this is because all the gathered data is already stored in GCP metadata so you don't need to store it on your side.

1. Creating the bucket

← Create a bucket

- **Name your bucket**

Pick a globally unique, permanent name. [Naming guidelines](#)

Tip: Don't include any sensitive information

✓ LABELS (OPTIONAL)

CONTINUE

- **Choose where to store your data**

This permanent choice defines the geographic placement of your data and affects cost, performance, and availability. [Learn more](#)

Location type

- ☐ Multi-region
Highest availability across largest area
- ☐ Dual-region
High availability and low latency across 2 regions
- ☒ Region
Lowest latency within a single region

CONTINUE

- **Choose a default storage class for your data**

A storage class sets costs for storage, retrieval, and operations. Pick a default storage class based on how long you plan to store your data and how often it will be accessed. [Learn more](#)

☒ **Standard** 

Best for short-term storage and frequently accessed data

☐ **Nearline**

Best for backups and data accessed less than once a month

☐ **Coldline**

Best for disaster recovery and data accessed less than once a quarter

☐ **Archive**

Best for long-term digital preservation of data accessed less than once a year

[CONTINUE](#)

- **Choose how to control access to objects**

Prevent public access

Restrict data from being publicly accessible via the internet. Will prevent this bucket from being used for web hosting. [Learn more](#)

☐ Enforce public access prevention on this bucket

Access control

☒ **Uniform**

Ensure uniform access to all objects in the bucket by using only bucket-level permissions (IAM). This option becomes permanent after 90 days. [Learn more](#)

☐ **Fine-grained**

Specify access to individual objects by using object-level permissions (ACLs) in addition to your bucket-level permissions (IAM). [Learn more](#)

[CONTINUE](#)


- **Choose how to protect object data**

Your data is always protected with Cloud Storage but you can also choose from these additional data protection options to prevent data loss. Note that object versioning and retention policies cannot be used together.

Protection tools

- ☒ None
- ☐ Object versioning (best for data recovery)
For restoring deleted or overwritten objects. To minimize the cost of storing versions, we recommend limiting the number of noncurrent versions per object and scheduling them to expire after a number of days. [Learn more](#)
- ☐ Retention policy (best for compliance)
For preventing the deletion or modification of the bucket's objects for a specified minimum duration of time after being uploaded. [Learn more](#)

2. Adding Read/Write access to the Cloud SQL Collector Service Account

 **Bucket details**

OBJECTS

CONFIGURATION


PERMISSIONS


Public access

Not public
This bucket is not publicly accessible. If you know objects never be exposed on the public internet, you should also p public access to this bucket. [Learn more](#)

PREVENT PUBLIC ACCESS

Permissions

 **ADD**

 REMOVE

Add principals to "cloudsqlcollector"

Add principals and roles for "cloudsqlcollector" resource

Enter one or more principals below. Then select a role for these principals to grant them access to your resources. Multiple roles allowed. [Learn more](#)

New principals

cloudsqlcollector@ti-dba-devenv-01.iam.gserviceaccount.com



Role *

Storage Object Creator



Access to create objects in GCS.

Condition

[Add condition](#)



Role

Storage Object Viewer



Read access to GCS objects.

Condition

[Add condition](#)





[+ ADD ANOTHER ROLE](#)

SAVE

CANCEL

Deploy Cloud SQL Collector Function

To deploy the function with the Pub/Sub method as the trigger, follow all the steps below in the Google Console:



 Cloud Functions  Create function


1 Configuration



 —

2 Code

Basics


Environment
1st gen  

Function name *
CloudSQLCollectors 


Region
us-west1  


We will use Cloud Pub/Sub event to trigger the Function so we need to create a Pub/Sub Topic, select CREATE A TOPIC option:

Create a topic

Topic ID *
CloudSQLCollectors 

Topic name: projects/ti-dba-devenv-01/topics/CloudSQLCollectors

☐ Use a schema 

☐ Set message retention duration (not free) 

Encryption

☒ Google-managed encryption key
No configuration required

☐ Customer-managed encryption key (CMEK)
Manage via Google Cloud Key Management Service

CANCEL

CREATE TOPIC

Now that the Topic is ready, select the one you have just created

Trigger

⚙️ Cloud Pub/Sub

Trigger type

Cloud Pub/Sub

Select a Cloud Pub/Sub topic *

projects/[REDACTED]/topics/CloudSQLCollectors

☐ Retry on failure ?

SAVE

CANCEL

Runtime, build, connections and security settings

< RUNTIME BUILD CONNECTIONS SECURITY AND >

Memory allocated *

256 MB

Timeout *

60

seconds ?

Runtime service account ?

Runtime service account

cloudsqlcollector

Autoscaling ?

Minimum number of instances

0

Maximum number of instances

1

Runtime environment variables

| | |
|---------------------------------------|---|
| Name 1 * CLOUD_SQL_CONNECTION_NAME | Value 1 ti-dba-devenv-01:us-west1:cloudsql |
| Name 2 * DB_USER | Value 2 cloudsqlcollector |
| Name 3 * DB_NAME | Value 3 cloudsqlcollector |
| Name 4 * BUCKET | Value 4 cloudsqlcollector |
| Name 5 * URL | Value 5 inventory/ |

Runtime, build, connections and security settings


 RUNTIME
 BUILD
 CONNECTIONS
 SECURITY AND
 

Ingress settings

- ☐ Allow all traffic
- ☒ Allow internal traffic only
 Only traffic from VPC networks in the same project or the same VPC SC perimeter is allowed.
- ☐ Allow internal traffic and traffic from Cloud Load Balancing
 Traffic from VPC networks in the same project, the same VPC SC perimeter or from Cloud Load Balancing is allowed.

Egress settings

By default, your function can send requests to the internet, but not to resources in VPC networks. To send requests to resources in your VPC network, create or select a VPC connector already created in the same region as the function.

Cloud SQL Collector Function uses a Serverless VPC Connector to reach all the CloudSQL Instances in your project out, this is because Cloud Function is a Serverless Service needing to connect to a Private Services.

← Create connector

Name *
cloudsqlcollectorvpconn

Region *
us-central1

A region is a specific geographical location where you can run your resources.

- ☒ Networks in this project
- ☐ Networks shared with me (from host project: [REDACTED])

Network *
default

Subnet *
Custom IP range

Select an unused /28 subnet or create a new one by entering an unused /28 IP range. The VPC Connector will create connector instances on this subnet.

IP range *
10.9.0.0 /28

IP range must be an unused /28 CIDR range in your VPC network, such as 10.8.0.0/28. The VPC Connector will create connector instances on IP addresses in this range. Ensure the range does not overlap with an existing subnet. [Learn more](#)

Scaling Settings

Minimum instances *

2

The minimum number of instances provisioned at any time. The connector will autoscale upward if more capacity is needed. Minimum number of instances cannot be changed later. Larger values increase your cost.

Maximum instances *

3

The maximum number of instances provisioned at any time. The connector will not autoscale above this value. Maximum number of instances cannot be changed later. This setting limits your maximum cost.



Connectors don't scale down automatically. Once the connector has reached maximum number of instances it will remain at this number.

Instance type *

f1-micro

Larger instances support higher bandwidth but raise your costs.

Serverless VPC access

[+ CREATE CONNECTOR](#)

[DELETE](#)

Serverless VPC Access allows Cloud Functions, Cloud Run (fully managed) services and App Engine standard environment apps to access resources in a VPC network using those resources' private IPs. [Learn more](#)

Filter Enter property name or value

| <input type="checkbox"/> | Name ↑ | Network | Subnet | IP address range | Region | Instance type | Min. instances | Max. instances |
|-------------------------------------|-------------------------|---------|--------|------------------|-------------|---------------|----------------|----------------|
| <input checked="" type="checkbox"/> | cloudsqlcollectorvpconn | default | | 10.9.0.0/28 | us-central1 | f1-micro | 2 | 3 |

Now we are able to select the VPC Connector which will be assigned to the function.

VPC Connector

cloudsqlcollectorvpconn

[Create a Serverless VPC Connector](#)

- ☐ Route only requests to private IPs through the VPC connector
- ☒ Route all traffic through the VPC connector

Now we are ready to add the code:

The screenshot shows the Google Cloud Functions console interface. At the top, the 'Runtime' is set to 'Python 3.9' and the 'Entry point' is 'CloudSQLCollector'. Below this, the 'Source code' section is set to 'Inline Editor'. On the left, a file explorer shows a project structure with files: 'main.py', 'requirements.txt', 'bucket.py', 'credential.py', and 'dbDriver.py'. The 'main.py' file is selected. On the right, the source code for 'main.py' is displayed, showing a copyright notice for Google LLC and a license reference to the Apache License, Version 2.0.

Runtime: Python 3.9

Entry point *: CloudSQLCollector

Source code: Inline Editor

main.py

requirements.txt

bucket.py

credential.py

dbDriver.py

```
1 # Copyright 2018 Google LLC
2 #
3 # Licensed under the Apache License, Version 2.0 (the
4 # you may not use this file except in compliance with
5 # You may obtain a copy of the License at
6 #
7 #     http://www.apache.org/licenses/LICENSE-2.0
8 #
9 # Unless required by applicable law or agreed to in wr
10 # distributed under the License is distributed on an '
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
12 # See the License for the specific language governing
13 # limitations under the License.
14
15
16 #Service Account: mssql-restore-test@ti-is-devenv-01.
17
```

Function Created.