05 Service Account --KirkYagami🧑‍💻🕵️

Service accounts overview | IAM Documentation | Google Cloud

A service account is a special kind of account typically used by an application or compute workload, such as a Compute Engine instance, rather than a person. A service account is identified by its email address, which is unique to the account.

Applications use service accounts to make authorized API calls by authenticating as either the service account itself, or as Google Workspace or Cloud Identity users through domain-wide delegation. When an application authenticates as a service account, it has access to all resources that the service account has permission to access.

The most common way to let an application authenticate as a service account is to attach a service account to the resource running the application. For example, you can attach a service account to a Compute Engine instance so that applications running on that instance can authenticate as the service account. Then, you can grant the service account IAM roles to let the service account—and, by extension, applications on the instance—access Google Cloud resources.

Accessing GCP Resources Using Users

- Associate the required IAM Role to the User:
  - User: yagamikirk@gmail.com
  - Roles: Compute Viewer, Storage Admin

How GCP Services Access Other GCP Services?

- **Compute Engine wants to access Cloud Storage Buckets.**
- Solution: Use IAM Service Accounts.

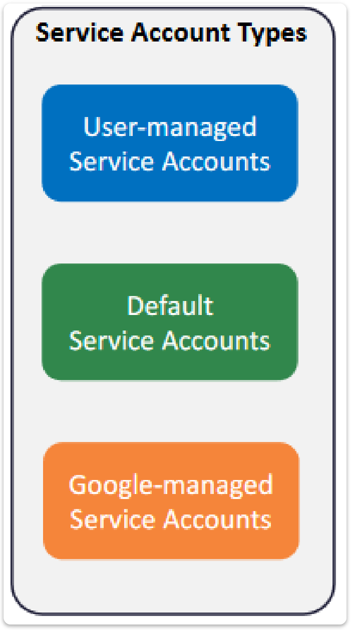
Using Service Accounts

- GCP Services can access other GCP Services.
- On-premises services (Application workloads) can access GCP Services.

Types of Service Accounts

- **User-managed service accounts (RECOMMENDED):**

    - You create and manage these service accounts.
    - Example:
        - Email: mysvc103@gcplearn9.iam.gserviceaccount.com
- Default service account (NOT RECOMMENDED):
    - Created automatically when you enable certain Google Cloud services.
    - Example (Compute Engine):
        - project-number-compute@developer.gserviceaccount.com
        - 899156651629-compute@developer.gserviceaccount.com
    - These are Google-created but user-managed service accounts.
    - Automatically grants Editor Role (Basic role) which contains extensive permissions (NOT RECOMMENDED TO USE IT, least-privilege approach is recommended).


Service Account Types
User-managed Service Accounts
Default Service Accounts
Google-managed Service Accounts

Types of Service Accounts

- Google-managed service accounts:
    - Google-created and Google-managed service accounts.
    - Google Cloud services need access to your resources so that they can act on your behalf.
    - We don't have access to edit or modify these service accounts.
    - We can see them in our project allow-policy.

Example Commands

```
gcloud projects get-iam-policy PROJECT-ID
gcloud projects get-iam-policy bigdata3844
```

DEMO

1. Click Activate Cloud Shell at the top of the Google Cloud console.
   When you are connected, you are already authenticated, and the project is set to your Project_ID, . The output contains a line that declares the Project_ID for this session:
2. (Optional) You can list the active account name with this command:

   ```
   gcloud auth list
   ```

~~~
3. Click **Authorize**.
   Output:
   ```shell
   ACTIVE: *
ACCOUNT: "ACCOUNT"

To set the active account, run:
    $ gcloud config set account `ACCOUNT`
~~~

4. (Optional) You can list the project ID with this command:

   ```
   gcloud config list project
   ```

```


## What are service accounts?


A service account is a special Google account that belongs to your application or a
[virtual machine](https://cloud.google.com/compute/docs/instances/) (VM) instead of
an individual end user. Your application uses the service account to [call the
Google API of a service]
(https://developers.google.com/identity/protocols/OAuth2ServiceAccount#authorizingr
equests), so that the users aren't directly involved.
```

~~~
For example, a Compute Engine VM may run as a service account, and that account can
be given permissions to access the resources it needs. This way the service account
is the identity of the service, and the service account's permissions control which
resources the service can access.

A service account is identified by its email address, which is unique to the
account.
### Types of service accounts

#### User-managed service accounts

When you create a new Cloud project using Google Cloud console and if Compute
Engine API is enabled for your project, a Compute Engine Service account is created
for you by default. It is identifiable using the email:

```shell
PROJECT_NUMBER-compute@developer.gserviceaccount.com
~~~

If your project contains an App Engine application, the default App Engine service account is created in your project by default. It is identifiable using the email:

```
PROJECT_ID@appspot.gserviceaccount.com
```

Google-managed service accounts

In addition to the user-managed service accounts, you might see some additional service accounts in your project's IAM policy or in the console. These service accounts are created and owned by Google. These accounts represent different Google services and each account is automatically granted IAM roles to access your Google Cloud project.

Google APIs service account

An example of a Google-managed service account is a Google API service account identifiable using the email:

```
PROJECT_NUMBER@cloudservices.gserviceaccount.com
```

This service account is designed specifically to run internal Google processes on your behalf and is not listed in the Service Accounts section of the console. By default, the account is automatically granted the project editor role on the project and is listed in the IAM section of the console. This service account is deleted only when the project is deleted.

Understanding IAM roles

When an identity calls a Google Cloud API, Google Cloud Identity and Access Management requires that the identity has the appropriate permissions to use the resource. You can grant permissions by granting roles to a user, a group, or a service account.

Types of roles

There are three types of roles in Cloud IAM:

- Primitive roles, which include the Owner, Editor, and Viewer roles that existed prior to the introduction of Cloud IAM.
- Predefined roles, which provide granular access for a specific service and are managed by Google Cloud.
- Custom roles, which provide granular access according to a user-specified list of permissions.

Task 1. Create and manage service accounts

When you create a new Cloud project, Google Cloud automatically creates one Compute Engine service account and one App Engine service account under that project. You can create up to 98 additional service accounts to your project to control access to your resources.

Creating a service account

Creating a service account is similar to adding a member to your project, but the service account belongs to your applications rather than an individual end user.

- To create a service account, run the following command in Cloud Shell:

```
gcloud iam service-accounts create my-sa-123 --display-name "my service account"
```

The output of this command is the service account, which looks similar to the following:

```
Created service account [my-sa-123]
```

Granting roles to service accounts

When granting IAM roles, you can treat a service account either as a resource or as an identity.

Your application uses a service account as an identity to authenticate to Google Cloud services. For example, if you have a Compute Engine Virtual Machine (VM) running as a service account, you can grant the editor role to the service account (the identity) for a project (the resource).

At the same time, you might also want to control who can start the VM. You can do this by granting a user (the identity) the serviceAccountUser role for the service account (the resource).

Granting roles to a service account for specific resources

You grant roles to a service account so that the service account has permission to complete specific actions on the resources in your Cloud Platform project. For example, you might grant the `storage.admin` role to a service account so that it has control over objects and buckets in Cloud Storage.

- Run the following in Cloud Shell to grant roles to the service account you just made:

```
gcloud projects add-iam-policy-binding $DEVSHELL_PROJECT_ID \
    --member serviceAccount:my-sa-123@$DEVSHELL_PROJECT_ID.iam.gserviceaccount.com
--role roles/editor
```

The output displays a list of roles the service account now has:

```
bindings:
- members:
  - user:email1@gmail.com
    role: roles/owner
- members:
  - serviceAccount:our-project-123@appspot.gserviceaccount.com
  - serviceAccount:123456789012-compute@developer.gserviceaccount.com
  - serviceAccount:my-sa-123@my-project-123.iam.gserviceaccount.com
  - user:email3@gmail.com
    role: roles/editor
- members:
  - user:email2@gmail.com
role: roles/viewer
```

```
etag: BwUm38GGAQk=
version: 1
```

Task 2. Use the client libraries to access BigQuery using a service account

In this section, you query the BigQuery public datasets from an instance with the help of a service account that has the necessary roles.

Create a service account

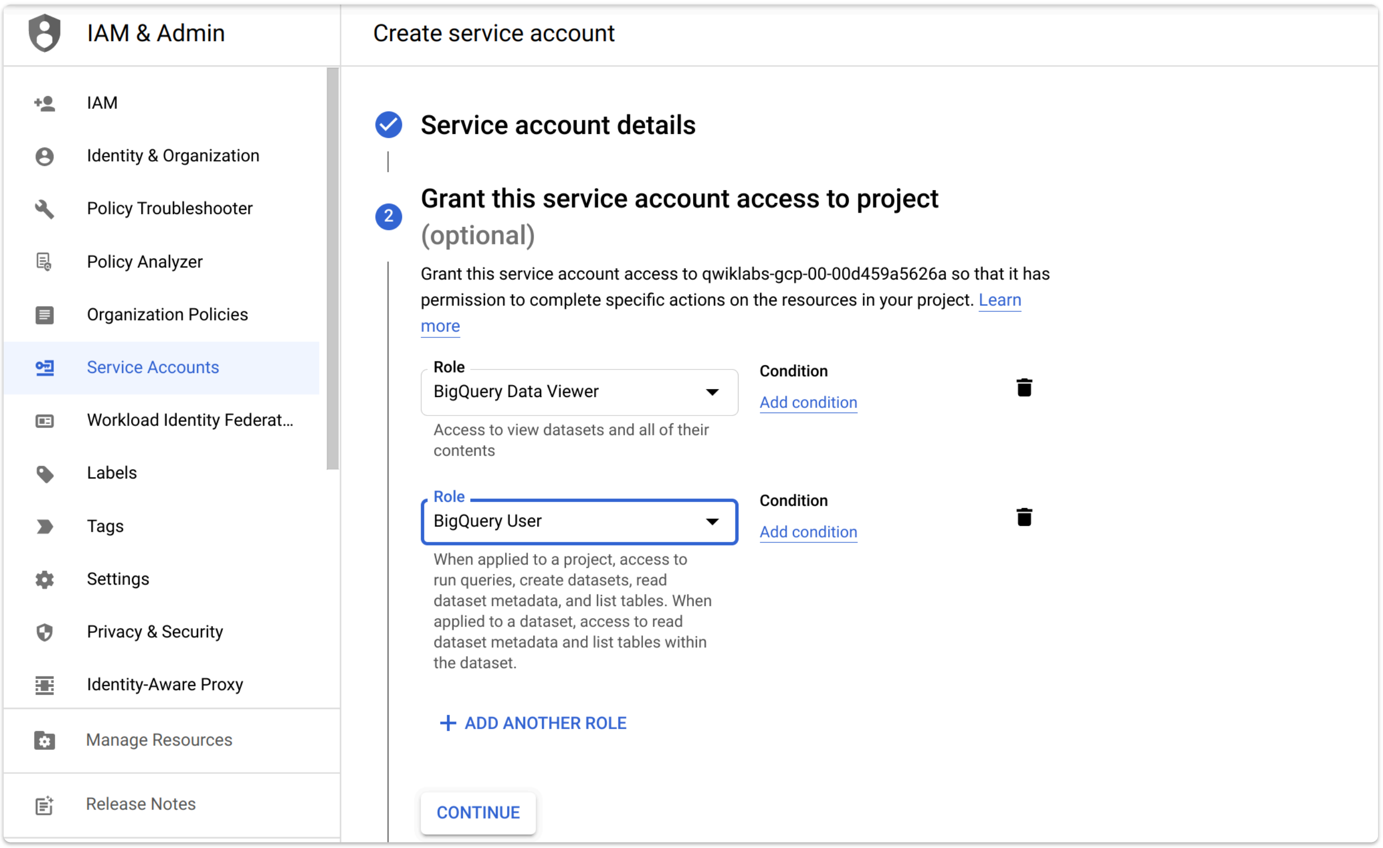
First create a new service account from the console.

1. Go to **Navigation menu** > **IAM & Admin**, select **Service accounts** and click on **+ Create Service Account**.
2. Fill necessary details with:

- **Service account name:** bigquery-sa

3. Now click **Create and Continue** and then add the following roles:
   - **Bigquery** > **BigQuery Data Viewer**
   - **BigQuery** > **BigQuery User**

Your console should resemble the following:



Service account details

Grant this service account access to project (optional)

Grant this service account access to qwiklabs-gcp-00-00d459a5626a so that it has permission to complete specific actions on the resources in your project. Learn more

Role	Condition
BigQuery Data Viewer	Add condition
BigQuery User	Add condition

BigQuery Data Viewer: Access to view datasets and all of their contents

BigQuery User: When applied to a project, access to run queries, create datasets, read dataset metadata, and list tables. When applied to a dataset, access to read dataset metadata and list tables within the dataset.

4. Click Continue and then click Done.

Create a VM instance

1. In the console, go to Compute Engine > VM Instances, and click Create Instance.
2. Create your VM with the following information:

Configuration	Value
Name	bigquery-instance
Region	---
Zone	---
Series	E2
Machine Type	e2-medium
Boot Disk	Debian GNU/Linux 11 (bullseye) x86/64
Service account	bigquery-sa
Access scopes	Set access for each API
BigQuery	Enabled

Note: If the `bigquery-sa` service account doesn't appear in the drop-down list, try typing the name into the "Filter" section.

3. Click **Create**.

Put the example code on a Compute Engine instance

1. In the console, go to **Compute Engine** > **VM Instances**.
2. SSH into `bigquery-instance` by clicking on the **SSH** button.
   In the SSH window, install the necessary dependencies by running the following commands:

```
sudo apt-get update
```

```
sudo apt-get install -y git python3-pip
```

```
pip3 install --upgrade pip
```

```
pip3 install google-cloud-bigquery
```

```
pip3 install pyarrow
```

```
pip3 install pandas
```

```
pip3 install db-dtypes
```

Now create the example Python file:

```
echo "
from google.auth import compute_engine
from google.cloud import bigquery

credentials = compute_engine.Credentials(
    service_account_email='YOUR_SERVICE_ACCOUNT')

query = '''
SELECT
  year,
  COUNT(1) as num_babies
```

```
FROM
  publicdata.samples.natality
WHERE
  year > 2000
GROUP BY
  year
'''

client = bigquery.Client(
    project='Your Project ID',
    credentials=credentials)
print(client.query(query).to_dataframe())
" > query.py
```

Add the Project ID to `query.py` with:

```
sed -i -e "s/Your Project ID/$(gcloud config get-value project)/g" query.py
```

Run the following to make sure that the `sed` command has successfully changed the Project ID in the file:

```
cat query.py
```

Example output (yours may differ):

```
from google.auth import compute_engine
from google.cloud import bigquery

credentials = compute_engine.Credentials(
    service_account_email='YOUR_SERVICE_ACCOUNT')

query = '''
SELECT
  year,
  COUNT(1) as num_babies
FROM
  publicdata.samples.natality
WHERE
  year > 2000
GROUP BY
  year
'''
```

```
client = bigquery.Client(
    project=,
    credentials=credentials)
print(client.query(query).to_dataframe())
```

Add the service account email to `query.py` with:

```
sed -i -e "s/YOUR_SERVICE_ACCOUNT/bigquery-sa@$(gcloud config get-value
project).iam.gserviceaccount.com/g" query.py
```

Run the following to make sure that the sed command has successfully changed the service account email in the file:

```
cat query.py
```

Example output (yours may differ):

```
from google.auth import compute_engine
from google.cloud import bigquery
credentials = compute_engine.Credentials(
    service_account_email='bigquery-sa@.iam.gserviceaccount.com')

query = '''
SELECT
  year,
  COUNT(1) as num_babies
FROM
  publicdata.samples.natality
WHERE
  year > 2000
GROUP BY
  year
'''

client = bigquery.Client(
    project=,
    credentials=credentials)
print(client.query(query).to_dataframe())
```

The application now uses the permissions that are associated with this service account. Run the query with the following Python command:

```
python3 query.py
```

The query should return the following output (your numbers may vary):

```
Row year  num_babies
0   2008  4255156
1   2006  4273225
2   2003  4096092
3   2004  4118907
4   2002  4027376
5   2005  4145619
6   2001  4031531
7   2007  4324008
```

Awesome work! You made a request to a BigQuery public dataset with a `bigquery-sa service account.