Security BigQuery:

Access control with IAM

This page provides information on Identity and Access Management (IAM) roles and permissions for BigQuery.

IAM role types

There are three types of roles in IAM:

- **Predefined roles** provide granular access for a specific service and are managed by Google Cloud. Predefined roles are meant to support common use cases and access control patterns.
- **Custom roles** provide access according to a user-specified list of permissions.
- **Basic roles** include the Owner, Editor, and Viewer roles.

BigQuery permissions and predefined IAM roles

To grant access to a BigQuery resource, assign one or more roles to a user, group, or service account. You can grant access at the following BigQuery resource levels:

- organization or Google Cloud project level
- dataset level
- table or view level

Basic roles for projects

- Viewer
- Editor
- Owner

Controlling access to datasets

- Dataset-level permissions determine the users, groups, and service accounts allowed to access the tables, views, and table data in a specific dataset.
- For example, if you grant the bigquery.dataOwner Identity and Access Management (IAM) role to a user on a specific dataset, that user can create, update, and delete tables and views in the dataset.

Required permissions

To control access to a dataset, you need the following IAM permissions:

- bigquery.datasets.update
- bigquery.datasets.get
- bigquery.datasets.getIamPolicy (lets you control access to a dataset using Cloud Console)
- bigquery.datasets.setIamPolicy (lets you control access to a dataset using Cloud Console)

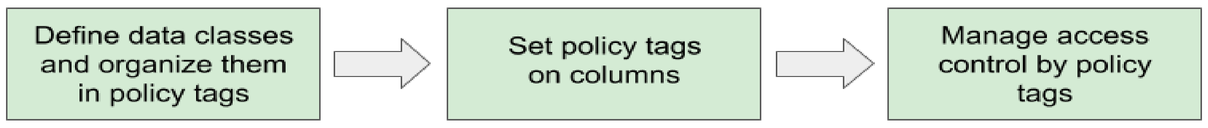
The predefined IAM role roles/bigquery.dataOwner includes the permissions that you need in order to control access to a dataset.

Introduction to column-level security

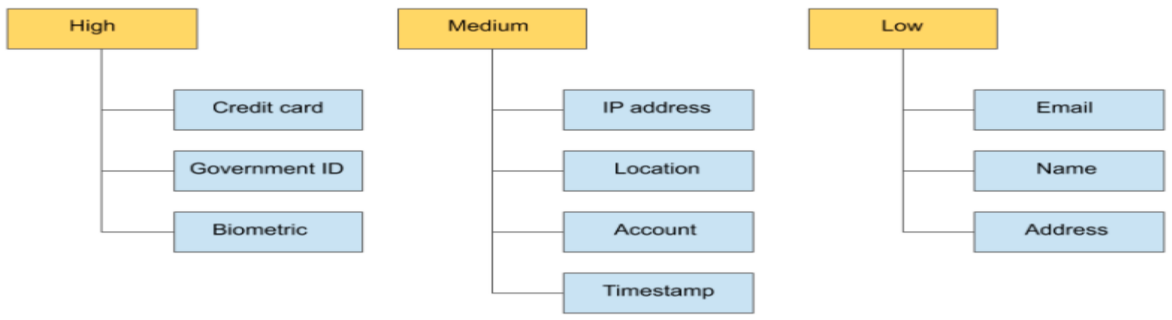
BigQuery provides fine-grained access to sensitive columns using *policy tags*, or type-based classification, of data. Using BigQuery column-level security, you can create policies that check, at query time, whether a user has proper access. For example, a policy can enforce access checks such as:

- You must be in group:high-access to see the columns containing TYPE_SSN.

Column-level security workflow


Define data classes and organize them in policy tags
Set policy tags on columns
Manage access control by policy tags

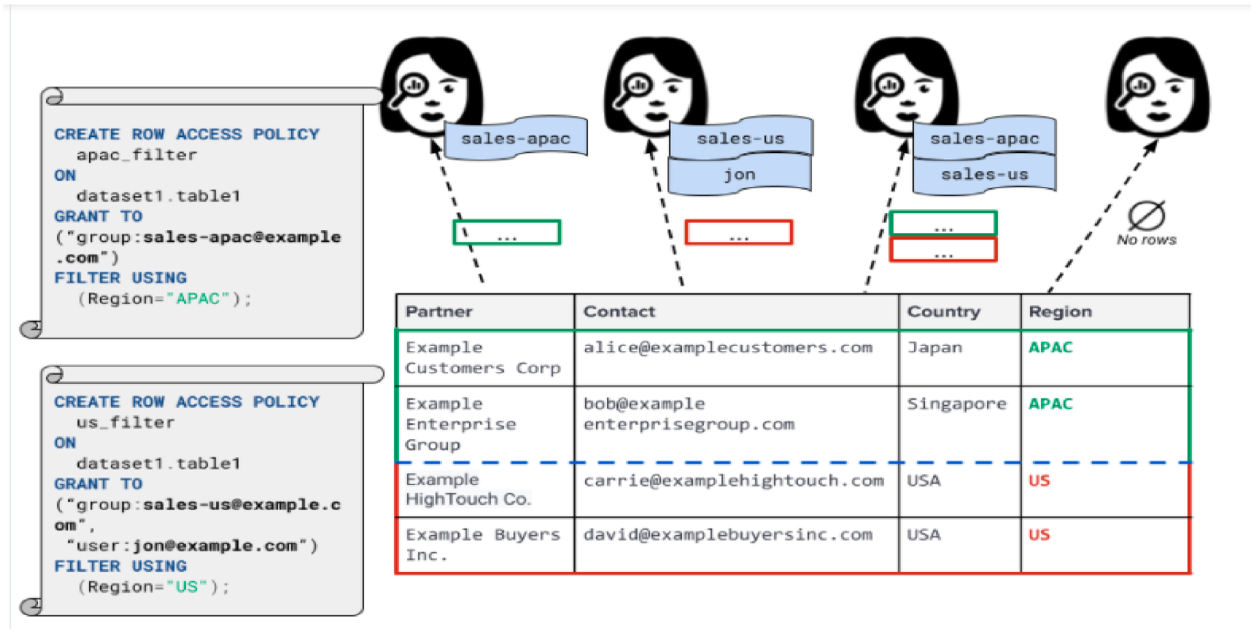
The following figure shows an example taxonomy. This hierarchy groups all data types into three top-level policy tags: **High**, **Medium**, and **Low**.



What is row-level security?

Row-level security lets you filter data and enables access to specific rows in a table based on qualifying user conditions.

BigQuery already supports access controls at the project, dataset, and table levels, as well as column-level security through policy tags. Row-level security extends the principle of least privilege by enabling fine-grained access control to a subset of data in a BigQuery table, by means of row-level access policies.



```
CREATE ROW ACCESS POLICY
  apac_filter
ON
  dataset1.table1
GRANT TO
("group:sales-apac@example
.com")
FILTER USING
  (Region="APAC");
```

```
CREATE ROW ACCESS POLICY
  us_filter
ON
  dataset1.table1
GRANT TO
("group:sales-us@example.c
om",
 "user:jon@example.com")
FILTER USING
  (Region="US");
```

Partner	Contact	Country	Region
Example Customers Corp	alice@examplecustomers.com	Japan	APAC
Example Enterprise Group	bob@example enterprisegroup.com	Singapore	APAC
Example HighTouch Co.	carrie@examplehightouch.com	USA	US
Example Buyers Inc.	david@examplebuyersinc.com	USA	US

Encryption at rest

BigQuery automatically encrypts all data before it is written to disk. The data is automatically decrypted when read by an authorized user. By default, Google manages the key encryption keys used to protect your data. You can also use customer-managed encryption keys, and encrypt individual values within a table.

Customer-managed encryption keys

If you want to manage the key encryption keys used for your data at rest, instead of having Google manage the keys, use Cloud Key Management Service to manage your keys. This scenario is known as *customer-managed encryption keys* (CMEK).

Encryption of individual values in a table

If you want to encrypt individual values within a BigQuery table, use the Authenticated Encryption with Associated Data (AEAD) encryption functions. If you want to keep data for all of your own customers in a common table, use AEAD functions to encrypt each customers' data using a different key. The AEAD encryption functions are based on AES.

Client-side encryption

Client-side encryption is separate from BigQuery encryption at rest. If you choose to use client-side encryption, you are responsible for the client-side keys and cryptographic operations. You would encrypt data before writing it to BigQuery. In this case, your data is encrypted twice, first with your keys and then with Google's keys. Similarly, data read from BigQuery is decrypted twice, first with Google's keys and then with your keys.