Chapter 6
Designing for Security and Compliance

Google Cloud Professional Data Engineer Exam objectives covered in this chapter include the following:

- **✓ 4.1 Designing for security and compliance. Considerations include:**
- Identity and access management (e.g., Cloud IAM)
- Data security (encryption, key management)
- Ensuring privacy (e.g., Data Loss Prevention API)
- Legal compliance (e.g., Health Insurance Portability and Accountability Act (HIPAA), Children's Online Privacy Protection Act (COPPA), FedRAMP, General Data Protection Regulation (GDPR))

Since data engineers work with diverse sets of data, they will likely need to use a variety of data stores that use access controls. They also should be prepared to work with sensitive data that needs additional protections. This chapter introduces several key topics of security and compliance, including

- Identity and access management
- Data security, including encryption and key management
- Data loss prevention
- Compliance

We'll begin with identity and access management, because it is fundamental to many security practices.

Identity and Access Management with Cloud IAM

Cloud IAM is Google Cloud's fine-grained identity and access management service that is used to control which users can perform operations on resources within GCP. Cloud IAM uses the concept of *roles*, which are collections of permissions that can be assigned to identities. Permissions are granted to roles, and then individuals can be assigned multiple roles to gain these permissions. Cloud IAM provides a large number of roles tuned to common use cases, such as server administrators or database operators.

Along with roles, additional attributes about resources or identities, such as IP address and date and time, can be considered when making access control decisions; this is known as *context-aware access*.

Cloud IAM maintains an audit log of changes to permissions, including authorizing, removing, and delegating permissions.

This chapter describes key aspects of Cloud IAM that you should understand when taking the Professional Data Engineer exam, including

- Predefined roles
- Custom roles
- Using roles with service accounts
- Access controls with policies

Together, these constitute the ways that you can control access to resources in GCP.

In GCP, users, groups, service accounts, and G Suite domains are authorized to access resources by granting those identities roles. As noted earlier, roles are collections of permissions. When a role is granted to an identity, that identity is granted all the permissions in that role. You do not directly assign permissions to an identity; identities get permissions only via roles.

GCP uses three types of roles:

- Primitive roles
- Predefined roles
- Custom roles

Primitive roles include the Owner, Editor, and Viewer, which existed prior to the introduction of Cloud IAM. These roles apply at the project level and so are considered course-grained access controls.

- The *Viewer role* grants read-only access resources.
- The *Editor role* includes all Viewer permissions plus the ability to modify the state of a resource.
- The *Owner role* includes all Editor role permissions and permissions to manage roles and permissions, along with setting up billing for a project.

In general, you should not use primitive roles except in cases where coarse-grained access controls are acceptable. For example, you could use primitive roles to grant access to developers in a development environment, since the developers would be responsible for administering the development environment.

PREDEFINED ROLES

Predefined roles are generally associated with a GCP service, such as App Engine or BigQuery, and a set of related activities, such as editing data in a database or deploying an application to App Engine.

The naming convention for roles is to start the role name with `roles/` followed by a string that identifies the service, such as `appengine`; followed by the type of entity to which the role applies, such as `instance` or `table`; followed by an operation, such as `get`, `list`, or `create`.

Let's look at a couple of examples. The `roles/appengine.deployer` role grants read-only access to all application and configuration settings and write access to create new versions. This role does not provide permission to modify existing applications except for deleting versions that are no longer receiving traffic. The permissions included in this role are as follows:

- `appengine.applications.get`
- `appengine.instances.get`
- `appengine.instances.list`
- `appengine.operations.*`
- `appengine.services.get`
- `appengine.services.list`
- `appengine.versions.create`
- `appengine.versions.delete`
- `appengine.versions.get`
- `appengine.versions.list`
- `resourcemanager.projects.get`
- `resourcemanager.projects.list`

As you can see, the naming convention for permissions is the name of the service followed by a resource type specific to that service and an action on resources of that type. The asterisk in this example indicates all types of actions applicable to the operation's resource, such as `get`, `list`, and `create`.

As a second example, BigQuery has a user role called `roles/bigquery.user` that grants permissions to run queries and other jobs within a project. Users can list their own jobs and datasets as well as create new datasets.

- `bigquery.config.get`
- `bigquery.datasets.create`
- `bigquery.datasets.get`
- `bigquery.datasets.getIamPolicy`
- `bigquery.jobs.create`
- `bigquery.jobs.list`
- `bigquery.models.list`
- `bigquery.readsessions.*`
- `bigquery.routines.list`
- `bigquery.savedqueries.get`
- `bigquery.savedqueries.list`
- `bigquery.tables.list`
- `bigquery.transfers.get`
- `resourcemanager.projects.get`

- `resourcemanager.projects.list`

  Many services have similar sets of roles having a similar set of permissions, often including admins, viewers, and some kind of worker roles. For example, the roles available with the following GCP services include

- **Cloud Dataproc:** `roles/dataproc.editor`, `roles/dataproc.viewer`, `roles/dataproc.admin`, and `roles/dataproc.worker`
- **Cloud Dataflow:** `roles/dataflow.admin`, `roles/dataflow.developer`, `roles/dataflow.viewer`, and `roles/dataflow.worker`
- **Cloud Bigtable:** `roles/bigtable.admin`, `roles/bigtable.user`, `roles/bigtable.viewer`, and `roles/bigtable.reader`
- **BigQuery:** `roles/biguery.admin`, `roles/biguery.connectionAdmin`, `roles/biguery.connectionUser`, `roles/biguery.dataEditor`, `roles/biguery.dataOwner`, `roles/biguery.dataViewer`, `roles/biguery.jobUser`, `roles/biguery.metadataViewer`, `roles/biguery.readSessionUser`, and `roles/biguery.user`

  Note that BigQuery uses fine-grained permissions on BigQuery resources, such as connections, metadata, and sessions.

CUSTOM ROLES

In addition to primitive and predefined roles, GCP allows for the use of custom-defined roles. With *custom roles*, you can assign one or more permissions to a role and then assign that role to a user, group, or service account.

Custom roles are especially important when implementing the *principle of least privilege*, which states that users should be granted the minimal set of permissions needed for them to perform their jobs. Because of this, you may want to grant someone a different subset/combination of permissions than what is available in the predefined roles.

Users must have the `iam.roles.create` permission to be able to create a custom role.

Figure 6.1 shows an example of the Create Role form in the IAM console. In addition to typical name, description, and identifier parameters, you can also specify a role launch stage, which can be alpha, beta, general availability, or deprecated. Custom roles usually start in the alpha stage and are then promoted to the beta or general availability stage after sufficient testing. The deprecation stage is used to indicate to users that the role should not be used.

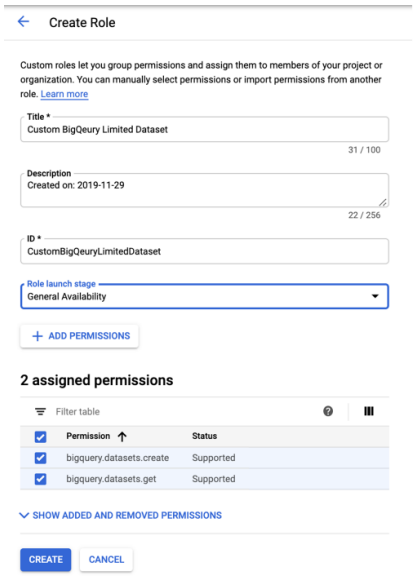

Figure 6.1 Create Role form in the cloud console

When creating a custom role, you can select from permissions assigned to predefined roles. This is helpful if you want someone to have a more limited version of a predefined role, in which case you can start with the list of permissions in a predefined role and select only the permissions that you would like to grant (see Figure 6.2).

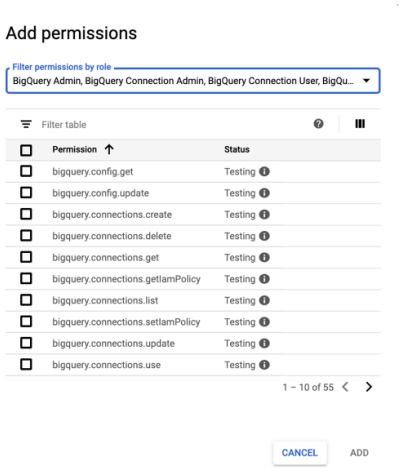

Figure 6.2 Selecting permissions from predefined roles

USING ROLES WITH SERVICE ACCOUNTS

Service accounts are a type of identity often used with VM instances and applications, which are able to make API calls authorized by roles assigned to the service account. A service account is identified by a unique email address. The email address name varies by the type of service account. For example:

- **App Engine service account:** Uses `<PROJECT-ID>` followed by `@appspot-gserviceaccount.com`, such as pde-exam-project-98765@appspot.gserviceaccount.com
- **Compute Engine service account:** Uses `<PROJECT-NUMBER>` followed by -compute@developer.gerviceaccount.com, such as 601440987865@developer.gserviceaccount.com
- **User-defined service accounts,** such as pde-exam-service-account@601440987865@iam.gserviceaccount.com

Service accounts do not have passwords and cannot be used to log in interactively via a browser. These accounts are authenticated by a pair of public/private keys.

Consider an application running in a VM that needs to write messages to a Cloud Pub/Sub topic. We could assign the role `roles/projects.topics.publish` to pde-exam-project-98765@developer.gserviceaccount.com. Note that the application or VM is the entity using this service account in order to get the publish permission. With that role assigned, the application could then call the Cloud Pub/Sub API to write messages to a topic created within the same project as the service account.

User-managed keys, also called *external keys*, are used from outside of Google Cloud. GCP stores only the public key. The private key is managed by the user. These keys are usually used as application default credentials, which are used for server-to-server authentication.

ACCESS CONTROL WITH POLICIES

You use roles to grant identities permission to perform some actions, but you can also access resources based on the rules associated with the resource. These rules are called *policies*.

A policy has three parts:

- Bindings
- Metadata
- Audit configuration

Bindings specify how access is granted to a resource. Bindings are made up of members, roles, and conditions. A *member* is an identity, which could be a user, group, service account, or domain. The *role* is simply a named collection of permissions. *Conditions* are logic expressions for describing context-based restrictions.

The *metadata* of a policy includes an attribute called `etag`, which is used for concurrency control when updating policies. This is needed because multiple applications can write to a policy at the same time. When retrieving a policy, you also retrieve the `etag`, and when writing the policy, you compare the `etag` that you retrieved with the current `etag` of the policy. If the two `etags` are different, then another application wrote to the policy after you retrieved it. In those cases, you should retry the entire update operation. Metadata also includes a version to indicate the iteration of the schema used in the policy.

Here is an example of a binding that binds the user data-enginer@example.com to the role `roles/resourcemanager.projectCreator`:

```
{
  "bindings": [
```

```
    {
      "members": [
        "user:
data-engineer@example.com"

      ],
      "role":
"roles/resourcemanager.projectCreator"

    },
  ],
  "etag": "adjfadURHlad",
  "version": 1
}
```

Audit configurations describe which permission types are logged and which identities are exempt from logging. For example, the following audit configuration enables logging on reads and writes and exempts the user data-engineer@example.com from logging read operations.

```
{
  "auditLogConfigs": [
    {
      "logType": "DATA_READ",
      "exemptedMembers": ["user:
data-engineer@example.com"]
    },
    {
      "logType": "DATA_WRITE",
    }
  ]
}
```

Policies can be defined at different levels of the resource hierarchy, including organizations, folders, projects, and individual resources. Only one policy at a time can be assigned to an organization, folder, project, or individual resource.

Policies are inherited through the resource hierarchy. Folders inherit the policies of the organization. If a folder is created within another folder, it inherits the policies of the encapsulating folder. Projects inherit the policies of the organization and any higher-level folders. Resources inherit the policies of the organization, folders, and projects above them in the hierarchy. The combination of policies directly assigned to a resource, and the policies inherited from ancestors in the resource hierarchy, is called the *effective policy*.

It is important to remember that IAM is additive only. You can't revoke, for example, permissions at a project level that were granted at the folder level.

Cloud IAM is a comprehensive service that provides for fine-grained access controls through the use of roles and policies. Predefined roles are available in Cloud IAM and are designed to group permissions needed for common use cases, such as administration of a database. Custom rules can be created when the predefined roles do not meet your specific needs, especially with respect to the principle of least privilege. Roles can also be used with service accounts to provide authorizations to VMs and applications. Also, access control policies can be used to control access to resources.

Using IAM with Storage and Processing Services

The previous section described IAM in general. Now let's take a look at some specific examples of how IAM predefined roles can be used with the following GCP services:

- Cloud Storage
- Cloud Bigtable
- BigQuery
- Cloud Dataflow

Of course, there are other relevant services, but once you understand these, you should be able generalize to other services as well.

CLOUD STORAGE AND IAM

There are several ways to control access to Cloud Storage resources, including buckets and objects in those buckets.

- Cloud IAM is the preferred way to control access to buckets and objects.
- For complex access control logic or when you need to control access to individual objects, you may need to use access control lists (ACLs).
- Signed URLs is another option for granting access. These URLs are generated by you and shared with someone to whom you want to grant access but only for a short period of time.
- If you want to control what can be uploaded to a bucket, you can use a signed policy document.

In this section, we will focus on the use of Cloud IAM with Cloud Storage.

Cloud Storage permissions are organized around resources, such as buckets, objects, and Hash-based Message Authentication Code (HMAC) keys. The bucket permissions allow users to create, delete, and list buckets. There are also permissions for getting and updating metadata as well as setting and getting IAM policies.

Object permissions also have `create`, `delete`, and `list` permissions as well as metadata and IAM policy permissions.

HMAC keys are used to authenticate access to Cloud Storage. The permissions for HMAC keys include creating, deleting, and listing keys as well as getting and updating metadata.

Five standard roles are used with Cloud Storage:

- **`roles/storage.objectCreator`**: Allows a user to create an object
- **`roles/storage.objectViewer`**: Allows a user to view an object and its metadata, but not ACLs. Users with this permission can also list the contents of a bucket
- **`roles/storage.objectAdmin`**: Gives a user full control over objects, including creating, deleting, viewing, and listing
- **`roles/storage.hmacKeyAdmin`**: Gives a user full control over HMAC keys within the project
- **`roles/storage.admin`**: Gives a user full control over buckets and objects, but when applied to a single bucket, it gives the user full control over only that bucket

If primitive roles are used in a project, they grant viewer, editor, and owner access to objects in Cloud Storage.

CLOUD BIGTABLE AND IAM

The access controls for Cloud Bigtable can be configured at the project, instance, or table level.

At the project level, you can do the following:

- Allow a user to read from any table in any instance of the project but not write to those tables
- Allow a user to read from and write to any table in any instance of the project
- Allow a user to manage any instance within the project

At the instance level, you can do the following:

- Restrict a user to be able to read from development but not production instances
- Allow a user to read and write to development instances and read from production instances
- Allow a user to manage development instances but not production instances

At the table level, you can do the following:

- Allow a user to read from a table but not write to the table
- Allow a user to write to a table but not read from the table

Cloud Bigtable has permissions that allow access to resources, such as instances, application profiles, clusters, and tables.

The predefined roles for Cloud Bigtable include Admin, User, Reader, and Viewer. Anyone with the `roles/bigtable.admin` role will be able to administer any instance in a project, including creating new instances. The `roles/bigtable.user` role allows for read and write access to tables. The `roles/bigtable.reader` role allows for read-only access to data in tables. Someone with the `roles/bigtable.viewer` role is restricted to accessing the GCP console for Bigtable.

BIGQUERY AND IAM

BigQuery provides a large number of permissions. This is understandable since BigQuery has many resources, such as tables, datasets, jobs, connections, saved queries, and more. Most of the permissions allow for creating, deleting, updating, and listing components. Some components, like tables, have permissions for other operations, such as exporting data.

The BigQuery roles are as follows:

- **`roles/BigQuery.admin`**: Gives a user permission to manage all BigQuery resources in a project.
- **`roles/BigQuery.dataEditor`**: When applied to a dataset, this gives a user permission to list tables and read metadata as well as create, update, get, and delete tables in a dataset; if this role is applied at the organization or project level, then the user can also create datasets.
- **`roles/BigQuery.dataOwner`**: When applied to a dataset, this gives a user permission to read, update, and delete the dataset as well as create, update, get, and delete the datasets tables; if this role is applied at the organization or project level, then the user can also create datasets.
- **`roles/BigQuery.dataViewer`**: When applied to a dataset, this gives a user permission to read dataset metadata, list tables in the dataset, and read table metadata; when applied at the organization or project level, the user can list all datasets in a project.
- **`roles/BigQuey.jobUser`**: Gives permission to run jobs, including queries; list jobs; and cancel the user's jobs.
- **`roles/BigQuery.metadataViewer`**: At the organization and project levels, this role allows a user to list all datasets and read metadata for all datasets in a project as well as list all tables and views and read metadata for all tables and views in project.

- `roles/BigQuery.user`: Gives permission to run jobs, enumerate and cancel the user's jobs, and enumerate datasets in a project. Users can also create new datasets, which grants `roles/Bigquery.dataOwner` to the user for the newly created table.

  Note that this list includes non-beta roles, and the list may have changed by the time you read this.

CLOUD DATAFLOW AND IAM

IAM includes permissions and roles to control access to Cloud Dataflow resources, including jobs, messages, and metrics.

The permissions for Cloud Dataflow include creating, listing, updating, and canceling jobs. There are also permissions to list messages and get metrics.

Since Cloud Dataflow is a stream and batch processing system, there are some differences in roles from the storage services. The IAM roles for Cloud Dataflow are as follows:

- `roles/dataflow.admin`: Gives permissions to create and manage jobs
- `roles/dataflow.developer`: Gives permissions to execute and modify jobs
- `roles/dataflow.viewer`: Gives permissions for read-only access to all Cloud Dataflow resources
- `roles/dataflow.worker`: Gives permissions to a Compute Engine service account to execute work units of a pipeline

The roles outlined here are predefined roles. If you find that a role has more permissions than you want to grant to a user, you can create a custom role with fewer permissions. You could also add more permissions in case a predefined role is too restrictive but the others are not restrictive enough.

Data Security

The following section originally appeared in *Official Google Cloud Certified Professional Cloud Architect Study Guide* (Wiley, 2019).

GCP provides multiple mechanisms for securing data in addition to IAM policies, which control access to data. Two essential services are encryption and key management.

ENCRYPTION

Encryption is the process of encoding data in such a way that it yields a coded version of the data that cannot be practically converted back to the original form without additional information. That additional information is a key that was used to encrypt the data. We typically distinguish between encryption at rest and encryption in transit.

Encryption at Rest

Google encrypts data at rest by default. You do not have to configure any policy to enable this feature. This applies to all Google data storage services, such as Cloud Storage, Cloud SQL, and Cloud Bigtable. *Encryption at rest* actually occurs at multiple levels:

- At the *platform level*, database and file data is protected using AES256 and AES128 encryption.
- At the *infrastructure level*, data is grouped into data chunks in the storage system, and each chunk is encrypted using AES256 encryption.
- At the *hardware level*, storage devices apply AES256 or AES128 encryption.

  At the platform level, distributed filesystems and databases encrypt data. The granularity of encryption can vary across services. For example, Cloud SQL encrypts all data in a database instance with the same key, whereas Cloud Spanner, Bigtable, and Cloud Firestore encrypt data using the infrastructure encryption mechanism.

  When Google Cloud stores data in the storage system, it stores it in subfile chunks that can be up to several gigabytes in size. Each chunk is encrypted with its own key, known as a *data encryption key (DEK)*. If a chunk is updated, a new key is used. Keys are not used for more than one chunk. Also, each chunk has a unique identifier that is referenced by access control lists (ACLs) to limit access to the chunks, which are stored in different locations to make it even more difficult for a malicious actor to piece them together.

  In addition to encrypting data that is in chunks, Google encrypts the data encryption keys using a second key. This is known as *envelope encryption*. The key used to encrypt a DEK is known as a *key encryption key (KEK)*.

  In addition to the chunk-level encryption that occurs at the infrastructure level, when blocks of data are written to persistent storage, the storage device encrypts those blocks using either AES128 or AES256. Older devices use AES128, but new storage devices use AES256.

  To summarize encryption at rest:

- Data at rest is encrypted by default in Google Cloud Platform.
- Data is encrypted at multiple levels, including the application, infrastructure, and device levels.
- Data is encrypted in chunks. Each chunk has its own encryption key, which is called a *data encryption key*.
- Data encryption keys are themselves encrypted using a *key encryption key*.

Google Cloud manages much of the encryption process, including managing keys. This is helpful for users who want Google Cloud to manage all aspects of encryption. In cases where organizations need to manage their own keys, they will have to use one of two key management methods described in the "Key Management" section.

Before delving into key management, let's look at encryption in transit.

Encryption in Transit

Encryption in transit, also called *encryption in motion*, is used to protect the confidentiality and integrity of data in the event that the data is intercepted in transit. GCP uses a combination of authenticating sources and encryption to protect data in transit.

Google distinguishes data in transit on the Google network and data in transit on the public Internet. Data within the boundaries of the Google network is authenticated but may not be encrypted. Data moving into and out of the physical boundaries of the Google network is encrypted.

Users of applications running in Google Cloud communicate with the application over the Internet. Traffic incoming from users to the Google Cloud is routed to the Google Front End (GFE), a globally distributed proxy service. The *Google Front End* terminates HTTP and HTTPS traffic and routes it over the Google network to servers running the application. The GFE provides other security services, such as protecting against distributed denial-of-service (DDoS) attacks. GFE also implements global load balancers.

All traffic to Google Cloud services is encrypted by default. Google Cloud and the client negotiate how to encrypt data using either *Transport Layer Security (TLS)* or the Google-developed protocol QUIC. (In the past, the term stood for Quick UDP Internet Connections [QUIC], but now the name of the protocol is simply QUIC.)

Within the Google Cloud infrastructure, Google uses *Application Layer Transport Security (ALTS)* for authentication and encryption. This is done at Layer 7 of the OSI network model.

GCP offers encryption at rest and encryption in transit by default. Cloud users do not have to do anything in order to ensure that encryption is applied to their data. Users of GCP services can, however, determine how encryption keys are managed.

KEY MANAGEMENT

There are many data encryption and key encryption keys in use for encryption at rest at any time in the Google Cloud.

Default Key Management

Google manages these encryption keys by default for users. Data encryption keys are stored near the data chunks that they encrypt. There is a separate data encryption key for each data chunk, but one key encryption key can be used to encrypt multiple data encryption keys. The key encryption keys are stored in a centralized key management service.

The data encryption keys are generated by the storage service that is storing the data chunk using a common cryptographic library. The data encryption keys are then sent to the centralized key management service where they are themselves encrypted using the storage systems key encryption key. When the storage system needs to retrieve data, it sends the data encryption key to the key management service, where the calling service is authenticated and the data key is decrypted and sent back to the storage system.

Customer-Managed Encryption Keys

Cloud KMS is a hosted key management service in Google Cloud. It enables customers to generate and store keys in GCP. It is used when customers want control over key management but do not need keys to reside on their own key management infrastructure. Note that customer-managed encryption keys (CMEKs) are often used to refer to KMS-based keys.

Cloud KMS supports a variety of *cryptographic keys*, including AES256, RSA 2048, RSA 3072, RSA 4096, EC P256, and EC P384. It also provides functionality for automatically rotating keys and encrypting data encryption keys with key encryption keys. Cloud KMS keys can be destroyed, but there is a 24-hour delay before the key is actually destroyed in case someone accidentally deletes a key or in the event of a malicious act.

Cloud KMS keys can be used for application-level encryption in GCP services, including Compute Engine, BigQuery, Cloud Storage, and Cloud Dataproc.

Customer-Supplied Encryption Keys

A third alternative for key management is *customer-supplied encryption keys (CSEKs)*. Customer-supplied keys are used when an organization

needs complete control over key management, including storage. CSEK is often used to refer to customer-supplied keys.

In this model, keys are generated and kept on premises and used by GCP services to encrypt the customer's data. These keys are passed with other arguments to API function calls. When the keys are sent to GCP, they are stored in memory while being used. Customer-supplied keys are not written to storage. They cannot be restored from GCP—the customer is the only one with persistent copies of the keys.

Encryption and key management are essential components of a comprehensive security regime. Data at rest and data in transit are encrypted by default. Keys are managed by default by GCP but can also be managed by cloud users. Users have two options. One is CMEK using Cloud KMS, which is a hosted managed key service that generates and stores keys in the cloud on behalf of a user. The other option is customer-supplied keys, which are managed on premises and sent to Google as part of API calls. Customer-supplied keys provide customers with the greatest amount of control, but they also require infrastructure and management procedures that are not needed when using default encryption.

Ensuring Privacy with the Data Loss Prevention API

The *Data Loss Prevention API* is a service that can detect sensitive information in text and images, redact or mask sensitive information, and perform risk analysis. This service operates as a job that applies pattern detection techniques to text or images. *Patterns* are defined as information types or InfoType detectors.

DETECTING SENSITIVE DATA

Google provides hundreds of InfoType detectors to identify known types of sensitive information, including

- Credit card numbers
- Dates of birth
- Email addresses
- Passport number
- Authentication tokens
- Passwords

There are also country-specific InfoType detectors, such as

- U.S. Social Security numbers
- Indian GST identification number (GSTIN)
- Japanese bank account numbers
- Spanish national identity numbers
- Paraguay civil identity card numbers

You specify InfoType detectors and data to inspect when creating an inspection job. The API works with text or base64-encoded images.

When an InfoType detector matches a string in a text, the API returns the InfoType detector that matched, a likelihood score, and a location specified by byte range or by record location if the text is structured. When an InfoType detector matches something in an image, the API returns the InfoType that matched, a likelihood score, and a location specified in pixel locations.

In addition to detecting patterns, the Data Loss Prevention API can redact the sensitive information. For example, if you were to scan the following text for email addresses:

- "And my email address is djohnson@example.com"

the API could return an obfuscated version such as:

- "And my email address is `[EMAIL_ADDRESS]`"

Similarly, when sensitive information is found in images, the section of the image containing it is blocked out, as shown in Figure 6.3.

Figure 6.3 An example of a redacted image generated by the Data Loss Prevention API

Image source: `https://cloud.google.com/dlp/docs/redacting-sensitive-data-images`

RUNNING DATA LOSS PREVENTION JOBS

There are two types of Data Loss Prevention jobs: inspection jobs and risk analysis jobs. *Inspection jobs* scan content for sensitive information using InfoTypes that you specify and generate reports on the location and type of sensitive information found. *Risk analysis jobs* calculate the likelihood that data could be re-identified.

Jobs are scheduled by creating *job triggers*. Job triggers start a scan on some Google Cloud storage service, including Cloud Storage and BigQuery.

After a job completes, you can automatically perform an action. There are two types of actions. The results of a scan job can be saved to BigQuery using a table that you specify. The other action is to publish the results of the scan job to a Cloud Pub/Sub topic.

INSPECTION BEST PRACTICES

Google has identified several best practices for using the Data Loss Prevention API.

First, you should inventory and prioritize the content you wish to scan. This is especially true if you have a large backlog of content that needs to be scanned. The data that is most at risk should be scanned first.

Make sure that the Cloud DLP service account has all the correct roles to access your storage services.

Start by sampling data and using simple matching criteria. This will help you identify which InfoType detectors you should use. You may find that your scans generate more false positives than are acceptable. In that case, you can create exclusion rules to reduce false positives.

Schedule scans using job triggers. Scans can be configured to inspect only data that has changed since the last scan.

Legal Compliance

Some of the material in this section originally appeared in *Official Google Cloud Certified Professional Cloud Architect Study Guide* (Wiley, 2019).

Google is regularly reviewed by independent organizations for verification of security, privacy, and compliance controls. Google Cloud's security, certifications and legal commitments can help support your compliance with a variety of regulations, but ultimately you are responsible for evaluating your own regulatory compliance. Some of the best-known regulations are listed below.

HEALTH INSURANCE PORTABILITY AND ACCOUNTABILITY ACT (HIPAA)

The *Health Insurance Portability and Accountability Act (HIPAA)* is a federal law in the United States that protects individuals' healthcare information. It was enacted in 1996 and updated in 2003 and 2005. HIPAA is a broad piece of legislation, but from a security perspective, the most important parts are the HIPAA Privacy Rule and the HIPAA Security Rule.

The *HIPAA Privacy Rule* is a set of rules established to protect patient's healthcare information. It sets limits on data that can be shared by healthcare providers, insurers, and others with access to protected information. This rule also grants patients the right to review information in their records and request information. For further details on this rule, see `www.hhs.gov/hipaa/for-professionals/privacy/index.html` (`http://www.hhs.gov/hipaa/for-professionals/privacy/index.html`).

The *HIPAA Security Rule* defines standards for protecting electronic records containing personal healthcare information. The rule requires organizations that hold electronic healthcare data to ensure the confidentiality, integrity, and availability of healthcare information; protect against expected threats; and prevent unauthorized disclosures. In practice, this requires security management practices, access control practices, incident response procedures, contingency planning, and evaluation of security measures. For more information on the HIPAA Security Rule, see `www.hhs.gov/hipaa/for-professionals/security/index.html` (`http://www.hhs.gov/hipaa/for-professionals/security/index.html`).

The *Health Information Technology for Economic and Clinical Health (HITECH) Act* was enacted in 2009, and it includes rules governing the transmission of health information. HITECH extended the application of HIPAA to business associates of healthcare providers and insurers. Business associates that provide services to healthcare and insurance providers must follow HIPAA regulations as well.

Google Cloud Platform can support your HIPAA compliance within the scope of a Business Associate Agreement (BAA). If you use Google Cloud for data and processes covered by HIPAA and enter into a BAA with Google Cloud, you should know that all of the Google Cloud infrastructure is covered under Google's Cloud's BAA and that many GCP services are as well, including Compute Engine, App Engine, Kubernetes Engine, BigQuery, Cloud SQL, and many other products. For a complete list, see `https://cloud.google.com/security/compliance/hipaa/`.

For more on HITECH, see:

`www.hhs.gov/hipaa/for-professionals/special-topics/hitech-act-enforcement-interim-final-rule/index.html` (`http://www.hhs.gov/hipaa/for-professionals/special-topics/hitech-act-enforcement-interim-final-rule/index.html`)

CHILDREN'S ONLINE PRIVACY PROTECTION ACT

The *Children's Online Privacy Protection Act (COPPA)* is a U.S. federal law passed in 1998 that requires the U.S. Federal Trade Commission to define and enforce regulations regarding children's online privacy. This legislation is primarily focused on children under the age of 13, and it applies to websites and online services that collect information about children.

The rules require online service operators to do the following:

- Post clear and comprehensive privacy policies
- Provide direct notice to parents before collecting a child's personal information
- Give parents a choice about how a child's data is used
- Give parents access to data collected about a child
- Give parents the opportunity to block collection of a child's data
- Keep a child's data only as long as needed to fulfill the purpose for which it was created
- In general, maintain the confidentiality, integrity, and availability of collected data

Personal information covered by this rule includes name, address, online contact information, telephone number, geolocation data, and photographs.

For more information on COPPA, see:

www.ftc.gov/tips-advice/business-center/guidance/complying-coppa-frequently-asked-questions (http://www.ftc.gov/tips-advice/business-center/guidance/complying-coppa-frequently-asked-questions)

FEDRAMP

The *Federal Risk and Authorization Management Program (FedRAMP)* is a U.S. federal government program that promotes a standard approach to assessment, authorization, and monitoring of cloud resources. The program is designed to ensure that cloud systems used by governments are adequately secure, reduce duplication of effort, and reduce risk management costs.

The FedRAMP framework includes four broad areas:

- **Document:** Includes categorizing systems, selecting security controls, and implementing and documenting controls
- **Assess:** The use of a third-party assessor to ensure that the controls in place are sufficient and effective
- **Authorize:** Involves analysis of risks, plans of action and milestones, and submission of a security package for authorizations
- **Monitoring:** A continuous process of monitoring systems once FedRAMP certifications are awarded

FedRAMP is required for U.S. federal agency cloud deployments. For more details on FedRAMP, see www.fedramp.gov (http://www.fedramp.gov).

GENERAL DATA PROTECTION REGULATION

The European Union's (EU) *General Data Protection Regulation (GDPR)* was passed in 2016, and enforcement of the GDPR began in 2018. The purpose of this regulation is to standardize privacy protections across the EU, grant controls to individuals over their private information, and specify security practices required for organizations holding private information of EU citizens.

GDPR distinguishes controllers and processors. A *controller* is a person or organization that determines the purpose and means of processing personal data. A *processor* is a person or organization that processes data on behalf of a controller. Controllers are responsible for gaining and managing the consent of individuals whose data is collected. Controllers direct processors on implementing the wishes of individuals who request access or changes to data. Processors are responsible for securing data and conducting audits in order to ensure that security practices are functioning as expected.

In the event of a data breach, data processors must notify the controller. Controllers in turn must notify the supervising authority, which varies by country, as well as the individuals whose data was compromised.

For more information on GDPR, see https://gdpr-info.eu/.

Exam Essentials

Understand the components of Cloud IAM. Cloud IAM provides fine-grained identity and access management for resources within GCP. Cloud IAM uses the concept of roles, which are collections of permissions that can be assigned to identities. Cloud IAM provides a large number of roles tuned to common use cases, such as server administrators or database operators. Additional attributes about resources or identities, such as IP address and date and time, can be considered when making access control decisions. Cloud IAM maintains an audit log of changes to permissions, including authorizing, removing, and delegating permissions.

Know the three types of roles. Primitive roles existed prior to Cloud IAM and include Owner, Editor, and Viewer roles. Predefined roles are

generally associated with a GCP service, such as App Engine or BigQuery, and a set of related activities, such as editing data in a database or deploying an application to App Engine. With custom roles, you can assign one or more permissions to a role and then assign that role to a user, group, or service account. Custom roles are especially important when implementing the principle of least privilege, which states that users should be granted the minimal set of permissions needed for them to perform their jobs.

Understand the purpose of service accounts. Service accounts are a type of identity that are used with VM instances and applications, which are able to make API calls authorized by roles assigned to the service account. A service account is identified by a unique email address. These accounts are authenticated by two sets of public/private keys. One set is managed by Google, and the other set is managed by users. Public keys are provided to API calls to authenticate the service account.

Understand the structure and function of policies. A policy consists of binding, metadata, and an audit configuration. Bindings specify how access is granted to a resource. Bindings are made up of members, roles, and conditions. The metadata of a policy includes an attribute called `etag` and versions. Audit configurations describe which permission types are logged and which identities are exempt from logging. Policies can be defined at different levels of the resource hierarchy, including organizations, folders, projects, and individual resources. Only one policy at a time can be assigned to an organization, folder, project, or individual resource.

Understand data-at-rest encryption. Encryption is the process of encoding data in a way that yields a coded version of data that cannot be practically converted back to the original form without additional information. Data at rest is encrypted by default on Google Cloud Platform. Data is encrypted at multiple levels, including the application, infrastructure, and device levels. Data is encrypted in chunks. Each chunk has its own encryption key, which is called a data encryption key. Data encryption keys are themselves encrypted using a key encryption key.

Understand data-in-transit encryption. All traffic to Google Cloud services is encrypted by default. Google Cloud and the client negotiate how to encrypt data using either Transport Layer Security (TLS) or the Google-developed protocol QUICC.

Understand key management. Cloud KMS is a hosted key management service in the Google Cloud. It enables customers to generate and store keys in GCP. It is used when customers want control over key management. Customer-supplied keys are used when an organization needs complete control over key management, including storage.

Know the basic requirements of major regulations. The Health Insurance Portability and Accountability Act (HIPAA) is a federal law in the United States that protects individuals' healthcare information. The Children's Online Privacy Protection Act (COPPA) is primarily focused on children under the age of 13, and it applies to websites and online services that collect information about children. The Federal Risk and Authorization Management Program (FedRAMP) is a U.S. federal government program that promotes a standard approach to assessment, authorization, and monitoring of cloud resources. The European Union's (EU) General Data Protection Regulation (GDPR) is designed to standardize privacy protections across the EU, grant controls to individuals over their private information, and specify security practices required for organizations holding private information of EU citizens.

Review Questions

You can find the answers in the appendix.

1. You have been tasked with creating a pilot project in GCP to demonstrate the feasibility of migrating workloads from an on-premises Hadoop cluster to Cloud Dataproc. Three other engineers will work with you. None of the data that you will use contains sensitive information. You want to minimize the amount of time that you spend on administering the development environment. What would you use to control access to resources in the development environment?

   1. Predefined roles
   2. Custom roles
   3. Primitive roles
   4. Access control lists

2. The auditors for your company have determined that several employees have more permissions than needed to carry out their job responsibilities. All the employees have users accounts on GCP that have been assigned predefined roles. You have concluded that the optimal way to meet the auditors' recommendations is by using custom roles. What permission is needed to create a custom role?

   1. iam.roles.create
   2. iam.custom.roles
   3. roles/iam.custom.create
   4. roles/iam.create.custom

3. You have created a managed instance group in Compute Engine to run a high-performance computing application. The application will read source data from a Cloud Storage bucket and write results to another bucket. The application will run whenever new data is uploaded to Cloud Storage via a Cloud Function that invokes the script to start the job. You will need to assign the role roles/storage.objectCreator to an identity so that the application can write the output data to Cloud Storage. To what kind of identity would you assign the roles?

1. User.
2. Group.
3. Service account.
4. You wouldn't. The role would be assigned to the bucket.

4. Your company has implemented an organizational hierarchy consisting of two layers of folders and tens of projects. The top layer of folders corresponds to a department, and the second layer of folders are working groups within a department. Each working group has one or more projects in the resource hierarchy. You have to ensure that all projects comply with regulations, so you have created several policies. Policy A applies to all departments. Policies B, C, D, and E are department specific. At what level of the resource hierarchy would you assign each policy?

1. Assign policies A, B, C, D, and E to each folder
2. Assign policy A to the organizational hierarchy and policies B, C, D, and E to each department's corresponding folder
3. Assign policy A to the organizational hierarchy and policies B, C, D, and E to each department's corresponding projects
4. Assign policy A to each department's folder and policies B, C, D, and E to each project

5. Your startup is developing a mobile app that takes an image as input and produces a list of names of objects in the image. The image file is uploaded from the mobile device to a Cloud Storage bucket. A service account is associated with the server-side application that will retrieve the image. The application will not perform any other operation on the file or the bucket. Following the principle of least privilege, what role would you assign to the service account?

1. roles/storage.objectViewer
2. roles/storage.objectAdmin
3. roles/storage.objectCreator
4. roles/storage.objectViewer and roles/storage.objectCreator

6. A data analyst asks for your help on a problem that users are having that involves BigQuery. The data analyst has been granted permissions to read the tables in a particular dataset. However, when the analyst runs a query, an error message is returned. What role would you think is missing from the users' assigned roles?

1. roles/BigQuery.admin
2. roles/BigQuery.jobUser
3. roles/BigQuery.metadataViewer
4. roles/BigQuery.queryRunner

7. Your company is subject to financial industry regulations that require all customer data to be encrypted when persistently stored. Your CTO has tasked you with assessing options for encrypting the data. What must you do to ensure that applications processing protected data encrypt it when it is stored on disk or SSD?

1. Configure a database to use database encryption.
2. Configure persistent disks to use disk encryption.
3. Configure the application to use application encryption.
4. Nothing. Data is encrypted at rest by default.

8. Data can be encrypted at multiple levels, such as at the platform, infrastructure, and device levels. At the device level, how is data encrypted in the Google Cloud Platform?

1. AES256 or AES128 encryption
2. Elliptic curve cryptography
3. Data Encryption Standard (DES)
4. Blowfish

9. In GCP, each data chunk written to a storage system is encrypted with a data encryption key. How does GCP protect the data encryption key so that an attacker who gained access to the storage system storing the key could not use it to decrypt the data chunk?

1. GCP writes the data encryption key to a hidden location on disk.
2. GCP encrypts the data encryption key with a key encryption key.

3. GCP stores the data encryption key in a secure Cloud SQL database.
4. GCP applies an elliptic curve encryption algorithm for each data encryption key.

10. The CTO has asked you to participate in a prototype project to provide better privacy controls. The CTO asks you to run a risk analysis job on a text file that has been inspected by the Data Loss Prevention API. What is the CTO interested in knowing?

1. The number of times sensitive information is redacted
2. The percentage of text that is redacted
3. The likelihood that the data can be re-identified
4. What InfoType patterns were detected

11. Your company is about to start a huge project to analyze a large number of documents to redact sensitive information. You would like to follow Google-recommended best practices. What would you do first?

1. Identify InfoTypes to use
2. Prioritize the order of scanning, starting with the most at-risk data
3. Run a risk analysis job first
4. Extract a sample of data and apply all InfoTypes to it

12. Your startup is creating an app to help students with math homework. The app will track assignments, how long the student takes to answer a question, the number of incorrect answers, and so on. The app will be used by students ages 9 to 14. You expect to market the app in the United States. With which of the following regulations must you comply?

1. HIPAA
2. GDPR
3. COPPA
4. FedRAMP

Support / Sign Out

PREV
Chapter 5 Building and Operationalizing Processing Infrastr…

NEXT
Chapter 7 Designing Databases for Reliability, Scalability, an…