

# 11

## Secret Manager

In this chapter, we will look at Google Cloud Secret Manager. A secret is any piece of data that needs to be protected, such as passwords, encryption keys, API keys, certificate private keys, and other sensitive information. Secrets can be stored in a secure and encrypted format using Google Cloud Secret Manager. Applications that run on the cloud platform often need to be able to securely store secrets and rotate them as needed. The secrets should also provide redundancy in the event that a region goes down. Traditionally, secrets were stored in configuration files embedded within the application using some form of encryption or in clear text. However, a secure way of storing secrets is by using Secret Manager. Secret Manager is a native offering on Google Cloud to store application secrets such as database passwords or API keys. It provides a single place to store application secrets and be able to manage access and audit who did what.

In this chapter, we will cover the following topics:

- Overview of Secret Manager
- Managing secrets and versions
- Accessing a secret
- Secret rotation policy
- CMEKs for Secret Manager
- Best practices for secret management

### Overview of Secret Manager

Secret Manager allows you to store and access secrets as binary blobs. It uses IAM permissions to grant access to the secrets and be able to manage them. Secret Manager is used for applications running on Google Cloud to store information such as database passwords, API keys, or certificates.

**Note**

Cryptographic keys should not be stored in Secret Manager. Cloud KMS is a better service since it allows you to encrypt the key material.

Before we start using Secret Manager, let us go over some core concepts. This will help you understand how Secret Manager works so you can make the right decision for your workloads.

## Secret Manager concepts

Let us look at some key definitions related to Secret Manager as defined in the Google Cloud documentation:

- **Secret:** A secret is an object at the project level that stores a collection of metadata and secret versions. The metadata includes replication locations, labels, and permissions.
- **Version:** A secret version stores the actual confidential information, such as API keys, passwords, and certificates. You cannot modify a secret once you create it, but you can delete it and create a new version.
- **Rotation:** Secret Manager supports rotational schedules for secrets. This, however, doesn't create a new secret version automatically but merely sends a message to a Pub/Sub topic configured for that secret based on the rotational frequency and time.
- **Replication policy:** Secrets have global names and globally replicated metadata. The replication policy can only determine where the secret payload data is stored. Upon the creation of each secret, it will have its own replication policy. You cannot modify the location in the replication policy.
- **Resource consistency:** Adding a secret version and then immediately accessing that secret version is an example of a consistent operation in Secret Manager. Consistent operations usually synchronize in minutes but may take a few hours. You should take consistency into consideration when designing your applications.

## Managing secrets and versions

Now let us look at some basic operations you can perform in Secret Manager. These operations can either be performed using `gcloud`, the Cloud console, or APIs. We have used the console to keep it simple.

## Creating a secret

Follow these steps to create a secret using the Google Cloud console:

1. Go to **Console | Security | Secret Manager**.
2. Click on **Create secret**.

[←](#) Create secret

---

### Secret details

This will create a secret with the secret value in the first version. [Learn more](#)

Name \*

! Secret name is required

### Secret value

Input your secret value or import it directly from a file.

Upload file [BROWSE](#)

Maximum size: 64 KiB

Secret value

---

### Replication policy

By default, Google automatically manages where this secret is stored. If you need to manually manage this, you can customize the locations by checking the box below. All secrets are globally accessible regardless of how they are replicated and stored. The replication policy cannot be changed after a secret is created. [Learn more](#)

Manually manage locations for this secret

---

### Encryption

This secret is encrypted with a Google-managed key by default. If you need to manage your encryption, you can use a customer-managed key instead. [Learn more](#)

Use a customer-managed encryption key (CMEK)

---

Figure 11.1 – Creating a new secret

As shown in *Figure 11.1*, a secret value can be copied directly into the console or provided via a file. Typically, binary secrets are provided via a file.

3. Choose the desired *rotation* period (**Set rotation period**). Note that this will only set a notification but not rotate the secret automatically.

## Rotation

Setting a rotation period will send rotation notifications to Pub/Sub topics. Secret Manager will not automatically rotate the secret value. [Learn more](#)

Set rotation period

---

## Notifications

Select Pub/Sub topic(s) that will receive event notifications whenever the secret or one of its versions is changed. These events can be user initiated changes or scheduled events. [Learn more](#)

[+ ADD TOPIC](#)

---

## Expiration

By default, the secret never expires. To set an expiration date for this secret, select **Set expiration date** below. If you choose an expiration date, the secret will be deleted and unavailable after that time. [Learn more](#)

Set expiration date

---

## Labels

Use labels to organize and categorize your secrets.

[+ ADD LABEL](#)

Figure 11.2 – Creating a new secret—rotation period

As shown in *Figure 11.2*, there are additional properties you can set on a secret. We recommend that you do not set **Expiration** since this can cause some unintended behavior. You can set **Labels** to organize secrets.

4. Alternatively, you can use the following `gcloud` command to create a secret:

```
gcloud secrets versions add secret-id --data-file="/file path/  
to/setcret.txt"
```

## Adding a new secret version

We will now see how to add a new secret version to an existing secret:

1. Select the secret and click on **+NEW VERSION**.

**Secret: "apiKey"**

`projects/794301636481/secrets/apiKey`

OVERVIEW	VERSIONS	PERMISSIONS	LOGS	
Versions	+ NEW VERSION	ENABLE	DISABLE	DESTROY
	<input checked="" type="checkbox"/> Version	Status	Encryption	Created on
	<input checked="" type="checkbox"/> 1	Enabled	Google-managed	5/16/22, 5:02 PM

1 version selected

Figure 11.3 – Adding a new secret version

It should show a popup to allow you to add a secret either from a file or by copying directly into the **Secret value** field.

## Add new version to "apiKey"

Input the new secret value or import it directly from a file.

Upload file

[BROWSE](#)

Maximum size: 64 KiB

**Disable all past versions**

[CANCEL](#) [ADD NEW VERSION](#)

Figure 11.4 – Adding a new secret version (continued)

As shown in *Figure 11.4*, you have an option to disable all past secret versions (see the **Disable all past versions** check box), so make sure to choose this option carefully as disabling a secret version currently used by your application could cause an outage.

## Disabling a secret

Now let us look at how to disable a secret that you no longer want to use:

1. Click on the secret you want to disable and click **DISABLE**.

## Secret: "apiKey"

projects/794301636481/secrets/apiKey

OVERVIEW		VERSIONS		PERMISSIONS		LOGS	
Versions		+ NEW VERSION		ENABLE		DISABLE	
		Version	Status	Encryption	Created on	↓	Actions
<input checked="" type="checkbox"/>	1	<span style="color: orange;">⚠</span> Disabled	Google-managed	5/16/22, 5:02 PM			<span style="font-size: 2em;">⋮</span>

1 version selected

Figure 11.5 – Disabling a secret

As seen in *Figure 11.5*, the secret will be disabled, and you will also see **Status** changed to **Disabled** on the console.

## Enabling a secret

This operation will enable a secret that is in the **Disabled** state:

1. Click on the secret you want to enable and click **ENABLE**.

## Secret: "apiKey"

projects/794301636481/secrets/apiKey

OVERVIEW		VERSIONS		PERMISSIONS		LOGS	
Versions		+ NEW VERSION		ENABLE		DISABLE	
		Version	Status	Encryption	Created on	↓	Actions
<input checked="" type="checkbox"/>	1	<span style="color: green;">✓</span> Enabled	Google-managed	5/16/22, 5:02 PM			<span style="font-size: 2em;">⋮</span>

1 version selected

Figure 11.6 – Enabling a secret

As seen in *Figure 11.6*, the secret will be enabled, and you will also see **Status** changed to **Enabled** on the console.

## Accessing a secret

Accessing a secret version returns the secret contents as well as additional metadata about the secret version. When you access a secret version, you specify its `version-id`. You can also access the latest version of a secret by specifying `latest` as the version.

Accessing a secret version requires the `secretmanager.secretAccessor` IAM role. Typically, this role is granted to the service account used by your application.

The following is a `gcloud` command that can be used to access a particular version of the secret. However, a common method is to use application libraries for access, as you will see later in the section:

```
gcloud secrets versions access version-id --secret="secret-id"
```

You can also use the `latest` keyword to get the current version, but this is *not* a recommended best practice.

## Accessing a binary secret version

You can access binary secrets directly but note that Cloud SDK formats the output as UTF-8, which can corrupt binary secrets. To get the raw bytes, Cloud SDK prints the response as base64-encoded, which you can then decode:

```
gcloud secrets versions access version-id --secret="secret-id"  
--format='get(payload.data)' | tr '_-' '/+' | base64 -d
```

## Accessing secrets from your application

You can access secrets from Google Cloud services such as Compute Engine, Application Engine, GKE, and Cloud Run.

Secret Manager has a REST and gRPC API for using and managing secrets. However, a convenient way to get access to secrets is by using client libraries. You can use these client libraries bundled with your application to access secrets. You do not need to embed service account keys within your application to access Secret Manager APIs. All you need is the right IAM role for the service account used by the Google Cloud service of your choice, such as GCE, GAE, or Cloud Functions:

- In Cloud Build, you can access secrets using environment variables in the build step of the build YAML config file by referring a field called `availableSecrets` to a specific secret version.
- For Cloud Run, you can use Secret Manager in two ways:
  - Mount the secret as a volume, which will make the secret available to the container as files. Reading the volume will read the secret from Secret Manager. You can use the `latest` keyword to fetch the secret using this method.

- Access the secret using a traditional environment variable. Environment variables are resolved by Cloud Run at the start of the container. When you use this method, do not use the `latest` keyword, use a particular version.
- For GKE, you can use Workload Identity in conjunction with Secret Manager client libraries to access secrets. You can also use the Secret Store CSI driver in conjunction with the Google Secret Manager provider for this driver. This allows you to access secrets as files mounted in Kubernetes pods. The provider is an open source project supported by Google that uses the workload identity of the pod that a secret is mounted onto when authenticating to the Google Secret Manager API. For this to work, the workload identity of the pod must be configured and appropriate IAM bindings must be applied. This plugin is built to ensure compatibility between Secret Manager and Kubernetes workloads that need to load secrets from the filesystem. The plugin also enables syncing secrets to Kubernetes-native secrets for consumption as environment variables. When evaluating this plugin, consider the following threats:
  - When a secret is accessible on the filesystem, application vulnerabilities such as **directory traversal** (`https://packt.link/wYWmj`) attacks can become higher severity as the attacker may gain the ability to read the secret material.
  - When a secret is consumed through environment variables, misconfigurations such as enabling a debug endpoint or including dependencies that log process environment details may leak secrets.
  - When syncing secret material to another data store (such as Kubernetes Secrets), consider whether the access controls on that data store are sufficiently narrow in scope.
- For multi-cloud application deployments, consider using workload identity federation on other platforms, such as Amazon Web Services or Microsoft Azure, that use current identity protocols to authenticate to Google Cloud APIs. Typically, you would want to use the native offering of the cloud providers for the secrets where your application is deployed.

Let us look at how you can achieve multi-region redundancy of secrets by using a secret rotation policy. This strategy supports the application redundancy you would need for high availability and failover.

## Secret replication policy

Secrets are a global resource entity; however, secret payloads (the underlying secret material) are stored locally within a region. Some regulated customers such as financial and healthcare institutions may have strict regionalization requirements, while other customers may want to store the secret near the data. A replication policy allows control over where secret payloads are stored.

There are two replication policy types: automatic and user-managed.

## Automatic

With the automatic policy type, the replication of the secret is managed by Google. This policy provides the highest level of availability:

- When a secret has an automatic replication policy, its payload data is copied as many times as needed. This is the easiest way to set things up, and most users should choose it. This is the policy that is used by default when a secret is created using the Google Cloud CLI or the web UI.
- A secret that is automatically replicated is stored in a single place for billing purposes.
- For a resource location organization policy evaluation, you can only make a secret with an automatic replication policy if you can make resources global.

## User-managed (user-selected)

With the user-managed replication policy type, the secret is replicated to only locations selected by users:

**Note**

The secret version addition will be impacted if one of the regions you selected is unavailable.

- The payload data of a secret with a user-managed replication policy is duplicated to a user-configured list of locations. The secret can be duplicated to as many supported locations as you choose. If there are regulations on where the secret payload data can be stored, this could be useful.
- Each location in the user-managed replication policy is treated as a separate location for billing reasons.
- A secret with a user-managed replication policy can only be created if the organization policy allows the creation of resources in all selected locations.
- As a rule, choose between two and five locations for replication. Having a large number of replication locations results in longer replication delays as well as increased latency in accessing the new secret payload.

Let us look at some of the encryption options, namely Google default encryption and customer-managed encryption, for Secret Manager.

## CMEKs for Secret Manager

By default, secrets are encrypted with Google default encryption. However, some highly regulated customers require control of keys, so Secret Manager supports **customer-managed encryption keys (CMEKs)** (within Cloud KMS) for encrypting:

**Note**

However, if you disable or permanently destroy the CMEK, the secret encrypted with that key cannot be decrypted.

- Secret payloads are encrypted by Google-managed keys before being written to persistent storage with no additional configuration required.
- Secret Manager encrypts data with a unique **data encryption key (DEK)** before writing it to persistent storage in a specific location. The Secret Manager service owns a replica-specific key called a **key encryption key (KEK)**, which is used to encrypt the DEK. This is commonly referred to as envelope encryption.
- The CMEK is a symmetric key that you control within Cloud KMS when using CMEKs with Secret Manager. The CMEK must be stored in the same GCP region as the secret. This makes sure that the Secret Manager has access to the key in the event of a zone or region failure.
- Customers who desire complete control over encryption keys can use the **Cloud External Key Manager (EKM)** key in the CMEK policy for encryption and decryption. This completely removes Google's management of the encryption key for your secrets as the encryption key is controlled outside of Google Cloud.
- CMEKs are also supported when using user-managed replication. When you create a CMEK in Cloud KMS, make sure the location of the key matches the location of the secret if you are using a user-managed replication policy.

So far, we have seen how to set up secrets in Google Cloud. Let us now go over some of the best practices for managing secrets in production and development environments.

## Best practices for secret management

Here are some general best practices that Google recommends when it comes to managing secrets:

- As with pretty much all services in Google Cloud, access to the Secret Manager API is protected by IAM. Follow the principle of least privilege when granting permissions to secrets to your applications.

- Divide applications and environments (staging/production) into independent projects. This can assist in segregating environments with IAM binding at the project level and guarantee that quotas are implemented independently.
- If necessary, establish a custom role or choose an existing role with the bare minimum of access. Think about who manages the secret (creates the secret, disables/enables it, or creates a new version) and who uses it, such as developers. You should have a separation of duties between these two roles, especially in production.
- Use secret-level IAM bindings or IAM conditions to limit access to the necessary subset of secrets when numerous services' secrets are in a single project.
- Use `gcloud auth application-default login` when developing locally. This produces a file with credentials that client libraries will immediately recognize.

Let us look at some of the best practices that you should consider while specifically developing your application.

## Best practices for development

Here are some best practices for working with Secret Manager during the development of an application that will ultimately be deployed in Google Cloud:

- To use secrets within your development IDEs such as Visual Studio Code or IntelliJ, use the Secret Manager integration plugin. Please see the Google Cloud documentation on how to enable Secret Manager for your choice of IDE (<https://packt.link/XRAGF>).
- For deployments in a single project, consider creating secrets in the project where the workload will be deployed. This makes the management of secrets much easier.
- Do not use slashes, /, when naming a secret as this violates Google Cloud's resource naming convention.
- Secret data stored in secret versions must fit within the quotas and limits specified in the Secret Manager specification. Apart from this, Secret Manager has no restrictions on the format of the secret material.
- Secrets such as API keys, SSH keys, and cryptographic keys are generated in external systems with formatting requirements.
- Use randomly generated, large, complicated values and cryptographically strong random number generators for secrets such as passwords when the format and length are not constrained.

## Best practices for deployment

Here are some best practices for Secrets Manager while deploying applications on Google Cloud:

- Use a separate project for secrets that are used by workloads in many projects, separate from the workload projects.
- Instead of granting Cloud IAM roles at the organization, folder, or project level, grant them directly on secrets. This will prevent over-grants and misuse.
- Managing secret-level IAM role grants can be easier using an **infrastructure-as-code (IaC)** approach such as Terraform. You can typically embed this in the given application CI/CD pipeline.
- The expiration date feature shouldn't be used for production workloads. This feature is best used to clean up transitory situations automatically. Consider using time-based IAM conditional access to secrets instead of secrets that expire.
- On projects that contain secrets or enclosing folders or organizations, avoid utilizing basic project-wide roles such as owner, editor, and viewer.
- Create distinct key administration and access roles according to the following guidelines:
  - Most applications require secret versions for specific secrets. For those specific secrets, grant these accounts `roles/secretmanager.secretAccessor`.
  - Grant `roles/secretmanager.secretVersionAdder` to processes that add new secret versions, for example, a CI/CD pipeline or administrator that owns the secret.
  - Grant `roles/secretmanager.secretVersionManager` to processes that handle timed disabling and destruction of secret versions.
- Rotate your secrets periodically to minimize the impact of leaked secrets.
- Exercise the rotation flow on a regular basis to lessen the chances of an outage.
- Unless your workload has specified location requirements, use the automatic replication policy. Most workloads' availability and performance requirements are met by the automated policy.
- To obtain and analyze `AccessSecretVersion` requests, enable data access logs at the folder or organization level.
- Automatically rotate secrets and have emergency rotation processes ready in case of a compromise.
- Instead of utilizing the most recent alias, refer to secrets by their version number. Create a hidden version of your program that is read at launch. Use your existing release methods to deploy updates to version numbers. Although using the most recent alias can be easy, if the latest version of the secret has an issue, your workload may be unable to use the secret version. The configuration can be evaluated and rolled back using your regular release processes if you pin it to a version number.

- Before deleting or destroying secret versions, disable them. This helps prevent outages by putting the secret in a reversible state that looks like destruct. That is, before permanently deleting data, you can disable it and wait a week to ensure there are no lingering dependencies.
- Secret Manager should not be used to store Google Cloud service account keys.
- In addition to IAM controls, you can set up a VPC Service Controls perimeter for your organization to limit access to the Secret Manager API from outside of Google Cloud using network-based controls such as IP address range and service accounts.
- The organization policy `PolicyMemberDomains` can be used to limit the identities that can be added to IAM policies for secrets.
- Estimate your peak secret usage (taking into account a *thundering herd* of queries owing to concurrent application deployments or service autoscaling) and make sure your project has enough quota to accommodate it. Request an increase in quota if more is required.
- Using Cloud Asset Inventory, keep track of secrets across your business to do the following:
  - Assist in the discovery of secrets throughout your company.
  - Identify non-conformance with organizational standards such as rotation, encryption setup, and location.

Having discussed the best practices for production, we now turn our attention to troubleshooting and auditing Secret Manager operations.

## Secret Manager logs

You can consult Cloud Logging to audit or troubleshoot the operation of Secret Manager.

Let us assume `project_id = acme-project-id`, `folder_id = acme-folder`, `billing_account_id = 123456`, and `organization_id = 987654321`. The logs of interest would have the names listed as follows:

```
projects/acme-project-id/logs/cloudaudit.googleapis.com%2Factivity
projects/acme-project-id/logs/cloudaudit.googleapis.com%2Fdata_access
projects/acme-project-id/logs /cloudaudit.googleapis.com%2Fsystem_event
projects/acme-project-id/logs/cloudaudit.googleapis.com%2Fpolicy

folders/acme-folder/logs/cloudaudit.googleapis.com%2Factivity
folders/acme-folder/logs/cloudaudit.googleapis.com%2Fdata_access
folders/acme-folder/logs/cloudaudit.googleapis.com%2Fsystem_event
folders/acme-folder/logs/cloudaudit.googleapis.com%2Fpolicy

billingAccounts/987654321/logs/cloudaudit.googleapis.com%2Factivity
```

```
billingAccounts/987654321/logs/cloudaudit.googleapis.com%2Fdata_
access
billingAccounts/987654321/logs/cloudaudit.googleapis.com%2Fsystem_
event
billingAccounts/987654321/logs/cloudaudit.googleapis.com%2Fpolicy

organizations/987654321/logs/cloudaudit.googleapis.com%2Factivity
organizations/987654321/logs/cloudaudit.googleapis.com%2Fdata_
access
organizations/987654321/logs/cloudaudit.googleapis.com%2Fsystem_
event
organizations/987654321/logs/cloudaudit.googleapis.com%2Fpolicy
```

This concludes the logging part of the Secret Manager application.

## Summary

In this chapter, we reviewed Secret Manager, its operations, and some critical aspects of setting secrets for your requirements. We also discussed best practices for the development and deployment of secrets. It is important to follow these best practices so that your application is designed optimally and to reduce the risk of outages. As a member of the application development team or information security team, you should be able to design your applications to leverage Secret Manager.

In the next chapter, we will look at the logging features of Google Cloud. Cloud Logging is one of the critical aspects of cloud deployment. We will also go over the best practices of Cloud Logging.

## Further reading

- Secret Manager IAM roles: <https://packt.link/wgzCw>
- Secret Manager rotation: <https://packt.link/slcoa>
- Creating and managing expiring secrets: <https://packt.link/js5eW>
- Analyzing secrets with Cloud Asset Inventory: <https://packt.link/v7w8z>
- Secret Manager data integrity assurance: <https://packt.link/8AmOG>



# 12

## Cloud Logging

Logging provides visibility into your environment and aids in troubleshooting and incident response. In this chapter, we will discuss what Cloud Logging is, how it works, the different types of logs and their applications, and how to collect, store, analyze, and export logs. We will also look at who log producers and consumers are and how to export logs to a centralized logging solution or a **Security Information and Event Management (SIEM)** system running either on-premises or in the cloud. Finally, we will discuss how to securely store and keep logs to meet regulatory requirements.

In this chapter, we will cover the following topics:

- Overview of Google Cloud Logging
- Understanding log categories
- Log management
- Logging and auditing best practices

## Introduction to Google Cloud logging

Cloud Logging is a managed service on Google Cloud. It gives you the ability to collect, store, search, analyze, monitor, and set alerts based on logs that are collected. You can use Cloud Logging to collect log data from over 150 applications, Google Cloud components, third-party cloud providers, and any combination of cloud platforms (hybrid environments). Cloud Logging, formerly known as Stackdriver, is part of Google Cloud's operations suite. It includes a console interface called the Logs Explorer, query logs, and an API to manage logs programmatically.

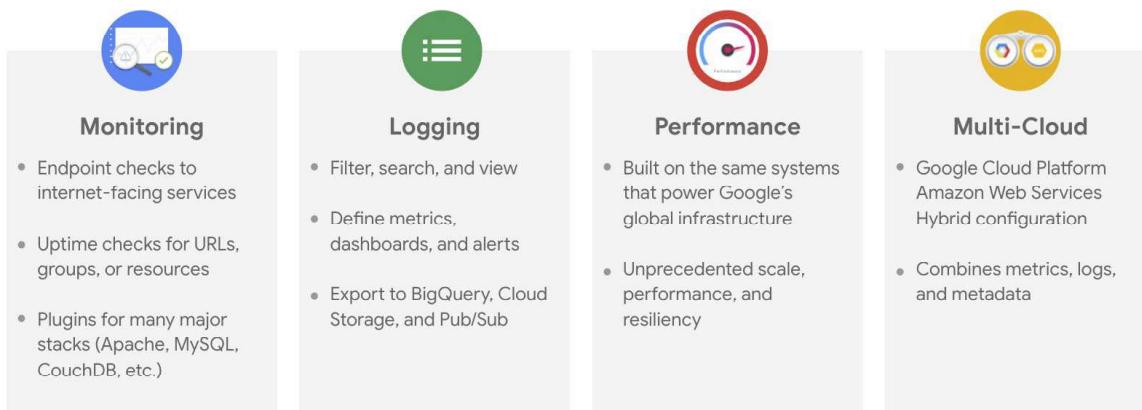


Figure 12.1 – Google Cloud's operations suite

Cloud Logging is one of the four components of Google Cloud's operations suite, as shown in *Figure 12.1*. Let's explore each of these components in more depth. Cloud Logging captures all logs and provides searching and filtering capabilities. Cloud Logging derives metrics from logs, creates dashboards, autoscales instances, and provides the ability to export logs for retention and analytics. We will cover these topics in more depth later in this chapter.

**Application Performance Management (APM)** combines the monitoring and troubleshooting capabilities of Cloud Logging and Cloud Monitoring with Cloud Trace, Cloud Debugger, and Cloud Profiler, to help you reduce latency and cost so you can run more efficient applications. This is out of the scope of the Google Cloud Professional Security Engineer exam.

Additionally, the Google Cloud operations suite offers multi-cloud capabilities in its support of both Google Cloud and Amazon Web Services and can also monitor on-premises resources and integrate with partner solutions.

By default, the operations suite monitors all system metrics, including Google **Compute Engine** (GCE) and **Google Kubernetes Engine** (GKE). The operations suite also monitors applications and managed services, such as Datastore and BigQuery. You can utilize the operations suite's Monitoring API to interact with users and collect business metrics. You can also deploy a monitoring agent on compute instances to collect lower-level and third-party tool metrics, such as disk latency/IO, for supported services. *Figure 12.2* illustrates the capabilities of the operations suite and how cloud resources generate signals and telemetry that are then consumed by different operations suite functions.

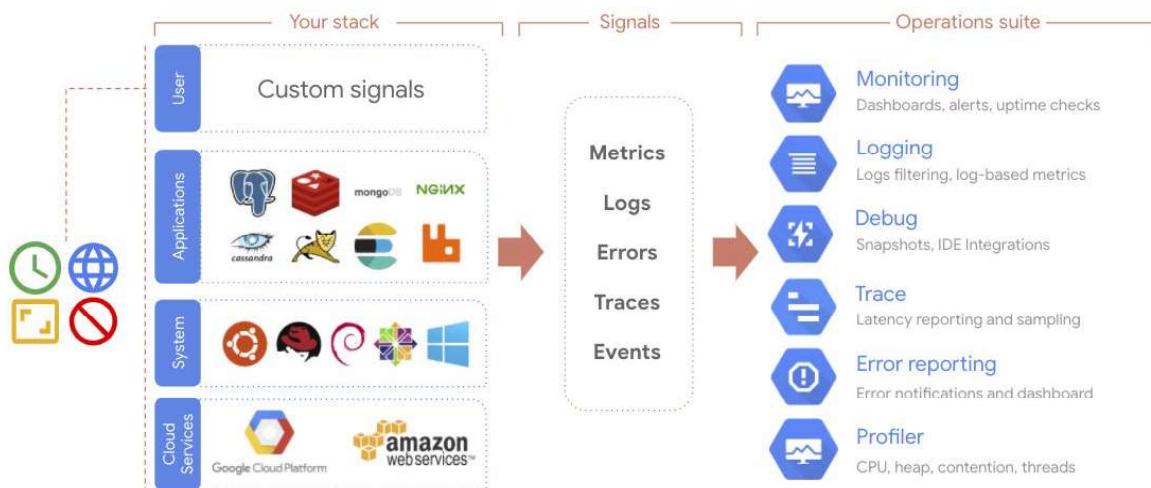


Figure 12.2 – Operations suite layers

We will now go over some fundamental Cloud Logging principles, as well as a few significant terms. It is important to understand them as they will be used throughout the chapter and assessed in the exam:

- **Log entries:** A log entry is a record of the state of a computer system or a description of specific events or transactions. Each log entry includes different types of data based on the service generating the log. Most log entries contain these attributes:
  - A timestamp to indicate when the event took place or when it was received by Cloud Logging.
  - The resource that produced the log entry.
  - A payload, also known as a message, is either provided as unstructured textual data or as structured textual data in JSON format.
  - The name of the log to which it belongs. Some logs also have metadata associated with them, such as the severity of each associated log entry.

- **Logs:** In Google Cloud resources, such as a Google Cloud project, a log is a named collection of log entries. There can be no logs unless there are log entries. A log's name is identified by the full path of the resource to which the log entries belong, followed by a simple log ID, such as `syslog`, or a structured ID that includes the log's writer, such as `compute.googleapis.com/activity`.
- **Retention period:** Log entries are stored in Cloud Logging for a specific period of time known as the retention period. Each log type has a different default retention period. You have the ability to create custom retention periods to store logs for longer periods than the defaults that are available in Cloud Logging.
- **Queries and filters:** Queries are written in the Logging query language. Queries return the matching log entries. The Logs Explorer and the Logging API both employ queries to select and display log entries.  
Filters are queries in the Logging query language that are used by log sinks to direct logs that match a given expression to a storage destination. Filters are also used to redirect matching logs to Cloud Monitoring.
- **Log Router:** In order to reach the Cloud Logging API, all collected logs must first go through the Log Router. For each log entry, the Log Router checks against pre-existing rules to identify which log entries should be ingested (stored) and routed to a destination, as well as which log entries should be excluded (discarded). We will cover the Log Router in more detail later in this chapter.
- **Sinks:** A sink consists of a filter and a storage destination. The filter logs entries to send to the sink.
- **Log-based metrics:** You can define metrics for the logs and then create alerts and triggers based on those metrics.
- **Access control:** You can control who has access to the logs using Cloud IAM permissions.

Each of the log types is categorized based on the log type, such as security, user, or platform. We will look at all these categories in more detail in the next section.

## Log categories

In this section, we broadly divide logs into three categories. We will look at each category from the perspective of security.



Figure 12.3 – Log categories

*Figure 12.3* illustrates the different types of log categories: security logs, user logs, and platform logs. Security logs consist of admin activity logs, data access logs, system event logs, and transparency logs. User logs are generated by user software, services, or applications and are written to Cloud Logging using a logging agent, the Cloud Logging API, or the Cloud Logging client libraries. Google Cloud Platform logs are service-specific logs that can help you better understand the Google Cloud services you're using. VPC flow logs, firewall logs, and other API logs are examples of platform logs.

Our focus will be only on logs that are useful from a security perspective. That doesn't necessarily mean that we will only look at the security logs category. We will also look at user logs and platform logs, specifically the subordinate log types within these categories that pertain to security.

## Security logs

Cloud Logging offers two forms of security logs: **Cloud Audit Logs** and **Access Transparency Logs**. Cloud Audit Logs generates the following types of audit logs for each cloud project, folder, and organization:

- **Admin Activity audit logs** include information on API calls and actions that potentially alter cloud resource settings or metadata. Activities such as creating a virtual machine or altering Cloud IAM permissions are examples of such changes that are captured by Admin Activity audit logs. You can't configure, exclude, or disable these types of logs. They're always recorded for security reasons. Admin Activity audit logs are created even if the Cloud Logging API is disabled. The retention period for administrator activity records is 400 days.
- **Data Access audit logs** show who accessed which data, the access time, and what actions they performed. Data Access audit logs are retained for 30 days. Not all Google Cloud services are supported. With the exception of BigQuery, Data Access audit logs are disabled by default.
- **System Event audit logs:** Events that alter resource configurations are recorded in System Event audit logs. These logs are generated automatically by Google's systems, not by users. System Event audit logs cannot be configured, excluded, or disabled; they are always written.
- **Policy Denied audit logs:** Policy Denied audit logs are created when a Google Cloud service rejects a user or service account access for a security policy violation. VPC Service Controls determines security policies and sends Cloud Logging Policy Denied audit logs. Your Google Cloud project gets charged for Policy Denied audit log storage. You can't disable Policy Denied audit logs, but you can exclude them from Cloud Logging.

The next type of security log is **Access Transparency**. Access Transparency provides logs based on the actions taken by Google staff when accessing your Google Cloud content. Access Transparency logs can help you track the compliance posture of your organization with regard to your legal and regulatory requirements. We covered Access Transparency in depth in *Chapter 3, Trust and Compliance*.

## User logs

As mentioned earlier, user logs, sometimes referred to as user-written logs, are written to Cloud Logging using an agent, the Cloud Logging API, or client libraries. Let's take a look at the user-agent-based log type:

- **Cloud Logging agent:** An agent can be installed to stream logs from instances to Cloud Logging. The Cloud Logging agent is based on Fluentd and is preconfigured by default to send syslogd logs and common third-party tools' logs (for example, MySQL, PostgreSQL, MongoDB, Cassandra, and Apache). It is possible to customize the pre-configured Fluentd agent configurations as required.

## Platform logs

Next, we will look at examples of platform logs:

- **VPC flow logs:** VPC flow logs contain granular VM flow-level network telemetry from Google Cloud. VPC flow logs provide insights into VPC traffic and can help with performance, traffic planning, a better understanding of networking costs, and forensics in the case of security-related incidents. VPC flow logs are aggregated collections of packets for a given interval and for a given connection. These logs are by default retained for 30 days and they do not lead to any performance degradation. The VPC flow log format includes a 5-tuple—for the source IP address, destination IP address, source port number, destination port number, and protocol (TCP or UDP)—and the timestamp. The metrics include packets, bytes (throughput), RTT for TCP flows, VPC annotations (region, zone, and VM name), and geographic annotations (country, region, and city). Logs are collected on each VM connection and aggregated at 5-second intervals. Google samples about one out of every 10 packets.
- **Firewall logs:** You can enable firewall rule logging individually for each firewall rule. Firewall rule logging only records TCP and UDP connections. Log entries are written from the perspective of VM instances.
- **NAT logs:** You can enable NAT logging on a NAT gateway instance. You can choose to generate logs for the creation of NAT connections and when packets are dropped because no port was available for NAT. Cloud NAT logs both TCP and UDP, including dropped egress packets. Dropped ingress packets (for example, if an inbound reply to an egress request was dropped) are not logged.

We have covered the different types of logs and their relevant examples. We will now consider the retention of logs. Defining a retention period gives you the ability to change the log period to match your compliance or regulatory requirement. Let's take a look at some of the key log types and their associated log retention periods in the following section.

## Log retention

Cloud Logging retention is critical for many organizations due to compliance requirements. The default retention period is often not sufficient; therefore, many customers configure custom retention periods for their logs. Longer retention periods come at a higher cost.

	Retention	Cost
Admin Activity audit logs	13 months (400 days)	Free
System Event audit logs	13 months (400 days)	Free
Access Transparency logs	13 months (400 days)	Free
Data Access audit logs	30 days	\$0.50 per GB over 50GB/Month
All other logs	30 days	\$0.50 per GB over 50GB/Month
Workspace admin console logs	6 months	Free
Log sink	Indefinite	Based on service option (GCS, BQ, Pub/Sub)

Figure 12.4 – Cloud Logging default retention periods and costs

Figure 12.4 shows the default retention periods and the associated costs for the different types of logs that we have covered. The costs are correct at the time of writing and may change.

Next, we'll explore the topic of log management. We will learn about log producers and consumers and understand how a log router works. We will also discuss exporting logs and creating log sinks in detail.

## Log management

A key component of logging is the ability to export logs so they can be consumed by a service running on the cloud or by a third-party solution. You will also learn how to aggregate logs, build log pipelines, perform log analysis, and deliver logs to SIEM solutions.

### Log producers

All resources that generate logs, such as products, services, and applications, whether they are in the cloud or a hybrid environment, are classified as log producers. These could be Google Cloud services or services running in your on-premises data center or a third-party cloud service provider.

## Log consumers

Log consumers are services such as centralized logging solutions and SIEM solutions. Any service that has the ability to consume logs and provide functions such as alerting, troubleshooting applications, and business intelligence is considered a log consumer. Cloud Logging on Google Cloud is an example of such a service.

## Log Router

The term Log Router has been used a few times in the chapter; we will now look at the workings of the Log Router in more detail to understand what it is and how it works. A Log Router ensures that logs are collected from producers and delivered to the consumer.

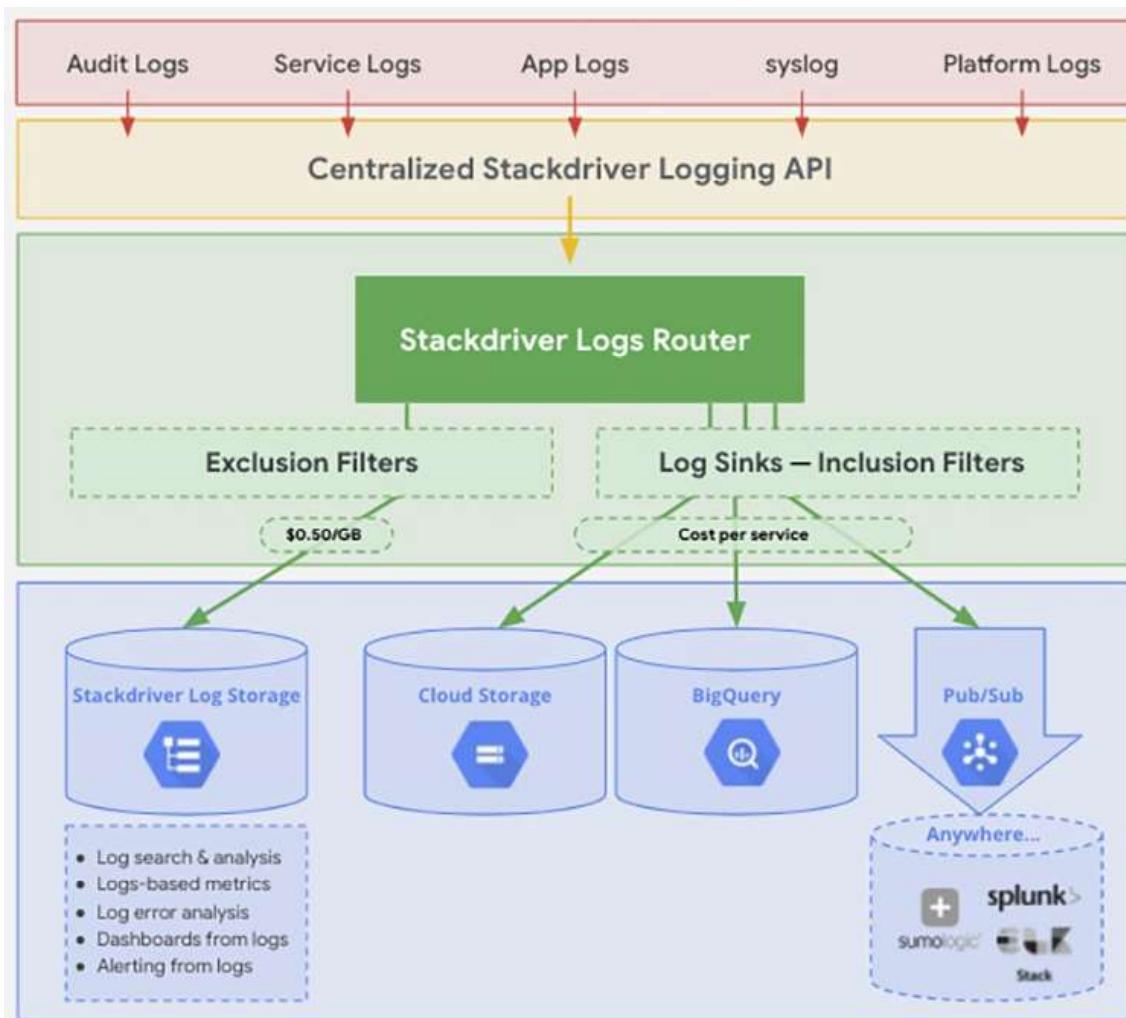


Figure 12.5 – Log Router

*Figure 12.5* shows how a Log Router works. Logs are transmitted to the Cloud Logging API, where they are processed by the Log Router. These logs include audit logs, platform logs, and user-written logs. The Log Router compares each log entry to the rules that are already in place to determine which log entries should be ingested and stored, which log entries should be included in exports, and which log entries should be discarded.

To export logs, create a filter that selects the relevant log entries to export, and then select a destination. You can choose from these options:

- JSON files can be stored in Google Cloud Storage buckets.
- Tables can be created in BigQuery datasets.
- JSON messages can also be delivered via Pub/Sub topics. This option also supports third-party integrations, such as Splunk.

Alternatively, you can create another Google Cloud project and have the log entries stored in Cloud Logging log buckets there.

Sinks are used to store the filter and the final destination. Google Cloud projects, organizations, files, and billing accounts can all be used as sinks. We covered sinks earlier in this chapter. We will look at them in more detail when we look at log exports in the following section.

## Log sinks and exports

Before we look at exporting logs, it's important to understand log sinks and inclusion/exclusion filters. Sinks tell Cloud Logging where to send logs. With sinks, you can send some or all of your logs to destinations that can handle them. Sinks are part of a specific Google Cloud resource, such as a project, billing account, folder, or organization. When a resource gets a log entry, the Log Router sends the log entry to the appropriate sinks and, if enabled, the Log Router forwards it to any ancestral sinks that belong to the resource hierarchy. The log entry is ultimately sent to the destination associated with the sink.

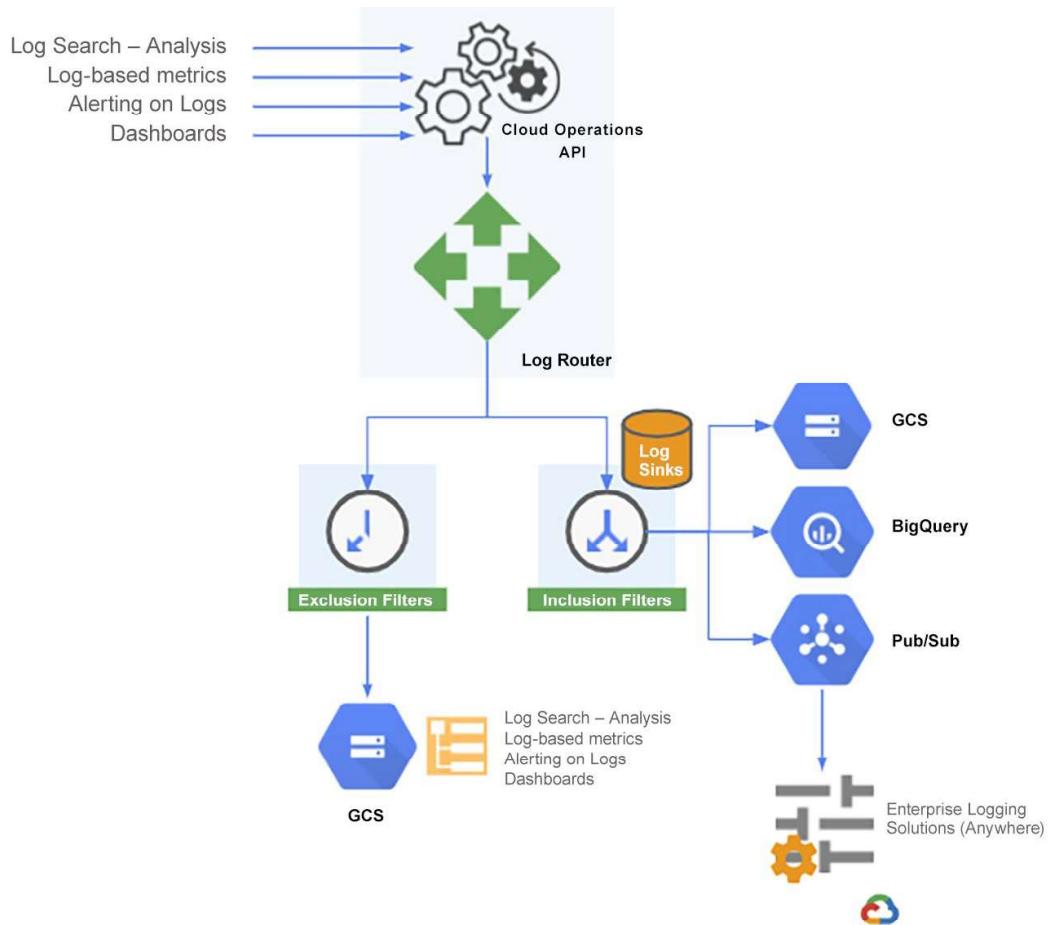


Figure 12.6 – Log exports and sinks

*Figure 12.6 shows the relationships between sinks, filters, and exports. By defining one or more sinks that have a log query and an export destination, you can easily export logs from a database. New log entries are compared to each sink as they are received by Cloud Logging. When an entry matches a sink's filter, the entry is written to the sink's destination.*

If no filters are attached to a sink, all logs will match and be forwarded to their final destination. The inclusion filter can be used to choose specific logs for the sink. Alternatively, you can use exclusion filters to keep unwanted logs out of the sink altogether. Filters are defined using the Logging query language.

Log buckets are containers used by Cloud Logging to store and organize log data in your Google Cloud projects, billing accounts, folders, and groups. Using Cloud Logging, you can study your logs in real time by having them indexed, optimized, and provided to you. Cloud Logging buckets are distinct from Cloud Storage buckets, despite the similarity of their names. Log buckets should be considered a local resource. You can pinpoint the location of the servers and storage facilities that handle your logs.

Next, we will look at the steps for how to create a log export.

## Creating a log export

In this section, you will learn how to create a log export sink. You will build a centralized logging solution by creating log buckets that are bound to specified regions with retention policies.

You can create 100 log buckets per project. Follow these steps to create user-defined log buckets:

1. From the **Logging** menu, select **Logs Storage**.
2. Click **Create Logs Bucket**.
3. Fill out **Name** and **Description** for your bucket.
4. (*Optional*) To set the region in which you want your logs to be stored, click the **Select log bucket region** drop-down menu and select a region. If you don't select a region, the global region is used, which means that the logs could be physically located in any of the regions. In our demonstration, in *Figure 12.6*, we have used a region in Australia.

### Note

You can't change the region of your log bucket once you've created it. You can, however, create an additional log bucket in a different region, reroute the associated sinks to the new log bucket, and then delete the log bucket.

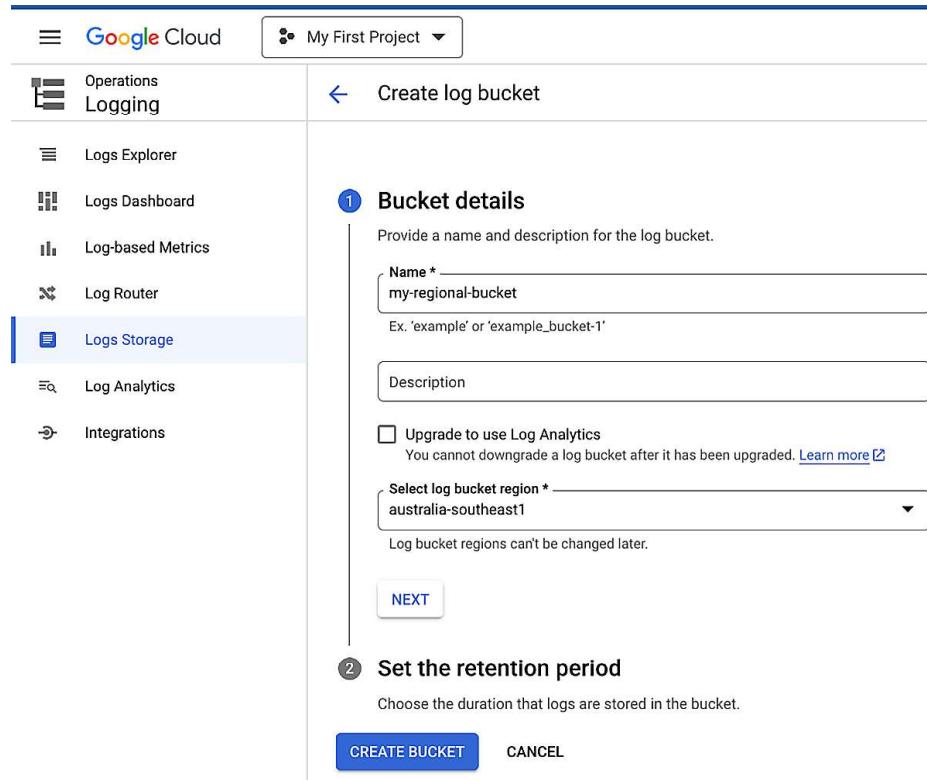


Figure 12.7 – Create log bucket

*Figure 12.7 shows how to configure a log bucket. Once you have specified the name of the bucket as per the naming convention, you can then either set the bucket region to global or select the region of your choice.*

5. *(Optional) To set a custom retention period for the logs in the bucket, click NEXT.*
6. In the **Retention period** field, enter the number of days that you want Cloud Logging to retain your logs. Google Cloud lets you define that period in seconds, days, and years, with a maximum period of retention of 100 years. If you don't customize the retention period, the default is 30 days. In *Figure 12.8*, we have kept the retention period set to its default value. You can also update your bucket to apply custom retention after you create it. Keep in mind that longer retention periods will result in higher charges.

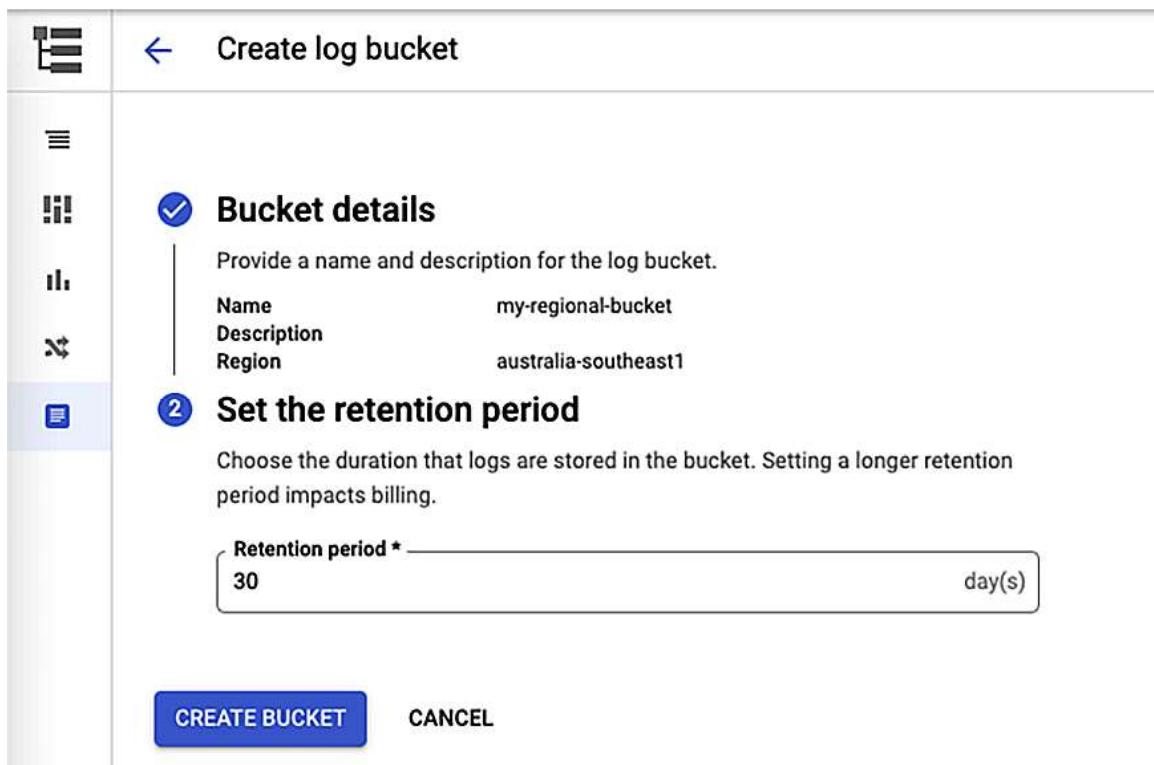


Figure 12.8 – Set the retention period for the log bucket

7. Click **CREATE BUCKET**. Your new bucket will appear in the **Log bucket** list.

You can also lock the bucket. When you stop updates to a bucket, you also stop updates to the bucket's retention policy. After you lock a retention policy, you can't delete a bucket until every log in it has been kept for as long as the policy dictates.

Next, we will cover the steps on how to create a log sink:

1. In the Cloud console, go to the Log Router, and select the Cloud project where you want to create the log sink.
2. Next, from the top left, select the **CREATE SINK** button and you will be presented with options as per *Figure 12.9*.
3. Here you can specify the details of the log sink, such as the sink name and its description; the name used in the example is `my-log-sink`.
4. Next, you can specify the destination where you want logs to be stored. There are different options available here, and based on your requirements, you can either select Google Cloud Storage (as per the example), create a new dataset in BigQuery or an existing dataset, specify Pub/Sub, or use **Splunk integration** as an option. In the example, we are using a Cloud Logging bucket, and the name of the bucket is `projects/Project_name/locations/australia-southeast1/buckets/my-regional-bucket`. You can either create a new bucket or use one that's already been created, such as the one we created in the previous step with an Australian region.

If your sink destination is a BigQuery dataset, the sink destination will be the following:  
`bigrquery.googleapis.com/projects/PROJECT_ID/datasets/DATASET_`

5. Next, you can specify the inclusion and exclusion filter; this is again based on what you want to include or exclude from your logs. For simplicity, in this example, no inclusion or exclusion filters have been used.

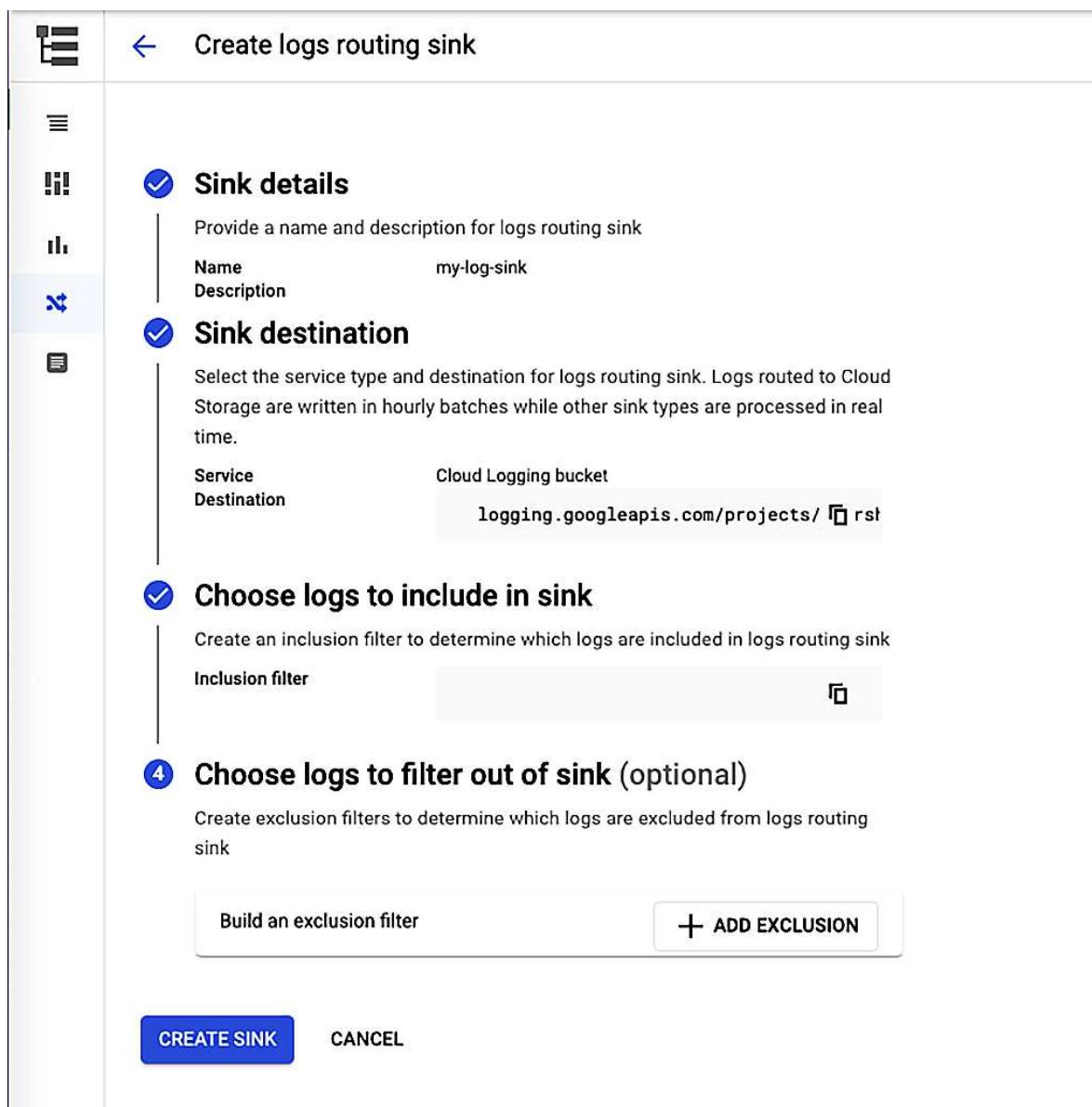


Figure 12.9 – Create a log sink

6. In *Figure 12.9*, we created a log sink named `my-log-sink` and selected the sink destination, its associated project, and geographical location, for example, `starbase155` and `australia-southeast1`, respectively. If you are routing logs between Google Cloud projects, you will need the appropriate destination permissions.
7. Finally, when all options are completed, click **CREATE SINK**.

Here we learned how to create an export and a log sink to send logs to the destination of your choice. Next, we will look at archiving and aggregating logs.

## Log archiving and aggregation

The process of archiving logs is like the process outlined in the previous section. You create a log sink, specify the inclusion and exclusion criteria, and choose the Google Cloud Storage bucket as the destination.

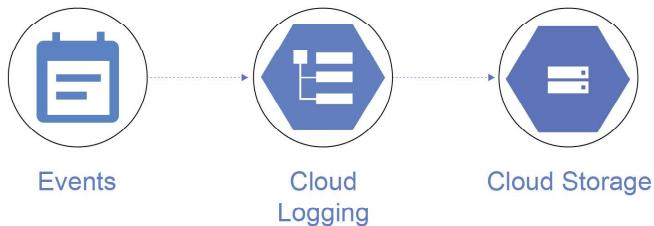


Figure 12.10 – Log archiving pipeline

*Figure 12.10* shows an illustration of what the log archiving pipeline looks like. You collect the events using Cloud Logging and then create a log sink with Cloud Storage as the destination.

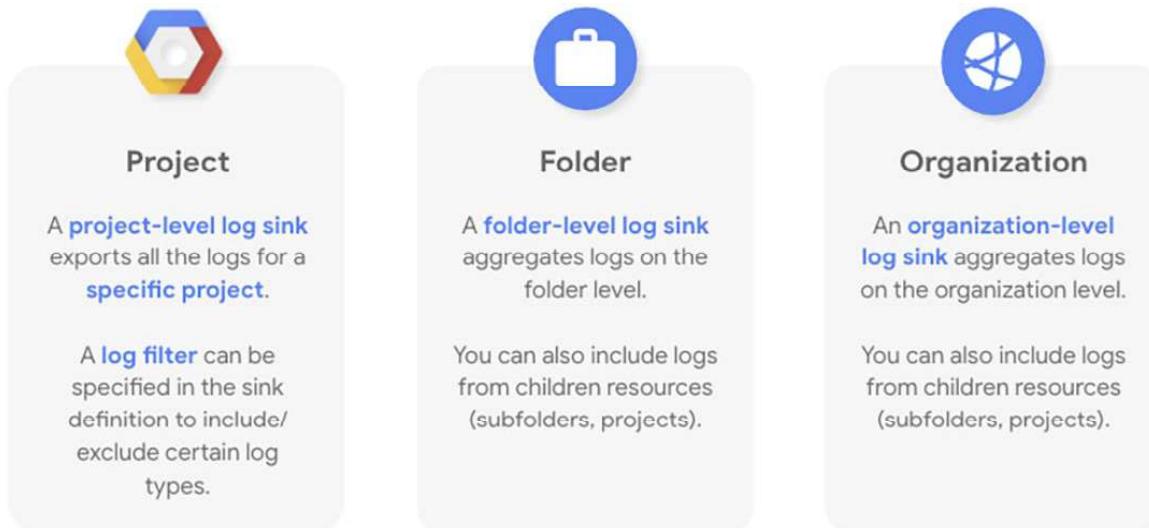


Figure 12.11 – Log aggregation

Log centralization is a typical requirement for auditing, archiving, and ensuring the non-repudiation of data and is considered a best practice. In order to achieve it, it is possible to configure aggregate exports at the organization level. This ensures that all logs on all projects (new and existing) in the organization are exported to a central location. *Figure 12.11* shows how you can configure log aggregation based on your requirements. You can aggregate logs at the project, folder, or organization level.

## Real-time log analysis and streaming

As mentioned earlier, Cloud Logging is a collection of different components. We have the Cloud Logging API, where all the logs are delivered. We also have the Log Router, where you can create your inclusion and exclusion filter and redirect the logs to a log sink. We also looked at several storage choices for logs after they've been removed from a log sink. With Google Cloud Storage, you can store the logs and apply lifecycle management rules to them based on your retention policy. If you wish to analyze the collected logs, you can do so with BigQuery.

In this section, we will look at storing logs in BigQuery for analytics. At the time of writing, there is a built-in feature that Google Cloud has called Log Analytics. As this feature is still in preview, it will not feature in the exam. In addition to Log Analytics, you can also create a pipeline that streams logs to BigQuery.

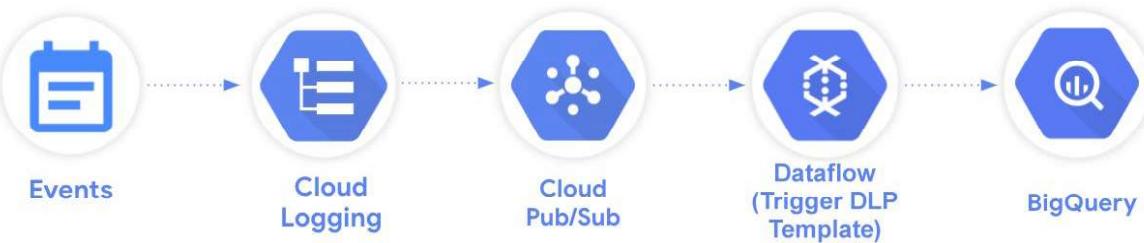


Figure 12.12 – Log streaming pipeline

Figure 12.12 illustrates a log streaming pipeline using Cloud Logging, Pub/Sub, and a DataFlow pipeline that triggers a Cloud Data Loss Prevention (DLP) template. DLP inspects sensitive data such as **personally identifiable information (PII)** to remove sensitive data before storing it for analytics in BigQuery. In our example, we're exporting through Pub/Sub, then through Dataflow, and lastly to BigQuery. Dataflow is very useful for processing logs in real time and at scale. However, we don't need to stream logs; we can simply create a log sink and deliver logs to BigQuery.

BigQuery enables you to store data in a secure and cost-effective manner. Its easy-to-use SQL interface can provide a solid foundation for creating AI data models and simplifying data operations. Data Studio and Looker are two tools that can be used to visualize BigQuery results. Visualization is out of scope for our discussion. BigQuery offers an easy-to-use and powerful capability to run queries to help you with analytics that you can use for troubleshooting and security incident response. In the next section, we will discuss an example of exporting logs to address compliance requirements. We will also cover the configuration aspects of the log pipeline discussed here.

The Logs Dashboard displays the severity and error data for the resources in your Cloud project. The Logs Dashboard allows you to spot patterns in your systems' activity, keep track of your workloads, and diagnose and address issues. All the log-based metrics are preconfigured and cannot be modified. The Logs Dashboard only shows statistics for resources that generate log data.

You can also use Logs Explorer to analyze logs. With Logs Explorer, you can run queries using the Logging query language based on your criteria, which can include logical operations such as AND and OR on the resource type.

Log Analytics can reduce the need to send logs to different systems to address separate use cases. We will briefly cover three common use cases. The first is for DevOps, where resolving issues quickly is critical to maintaining reliability. Log Analytics can help uncover patterns and correlate data across services. Faster troubleshooting can also help improve the speed of software development by reducing the time spent on debugging.

The second is for security. Investigations are a critical part of understanding and remediating security threats. Log Analytics can identify logs across large time ranges, which often means large data volumes.

The third is for IT operations, where managing a fleet of different services is a core part of IT operations. It's important to understand the events and trends across the whole fleet. Log Analytics can help identify patterns and fleet performance over time.

There are many other ways that Log Analytics helps enable business decisions, including analyzing business data reported in the logs. By unlocking valuable insights, Log Analytics brings new value to your logs.

We have covered Log Analytics and how you can use BigQuery and third-party applications for analytics. The next section will cover how to meet your compliance and regulatory requirements using log exports.

## Exporting logs for compliance

In this section, we will look at exporting logs to meet compliance requirements. Configuring log exports is an important topic in the exam. There are some advanced export configurations that you can practice, such as exporting logs for security and analytics and configuring logs to go to Elastic or Splunk when you are using a third-party SIEM tool.

In this example, we will export logs to a Cloud Storage bucket. We will then look at how to limit access to the logs. Logs can be moved to Nearline or Coldline storage classes and then deleted using Cloud Storage's object lifecycle management capability.

There are five steps to the configuration of log exports:

1. First, we will begin the setup by creating a logging export bucket in Cloud Storage.

The screenshot shows the 'Create a Bucket' wizard in the Google Cloud Platform. On the left, there's a sidebar with 'Cloud Storage' selected. The main area has a back arrow and the title 'Create a Bucket'. It lists several steps:

- Name your bucket**: Name: my-log-bucket-for-demo
- Choose where to store your data**: Location: us (multiple regions in United States), Location type: Multi-region
- Choose a storage class for your data**: Default storage class: Standard
- Choose how to control access to objects**: Public access prevention: On, Access control: Uniform
- Choose how to protect object data**: A note states that your data is always protected with Cloud Storage and provides options for additional protection.

**Protection tools** section:  
None  
Object versioning (for data recovery)  
Retention policy (for compliance) (selected)  
A note for retention policy says: "For preventing the deletion or modification of the bucket's objects for a specified minimum duration of time after being uploaded." with a link to learn more.  
Duration \*: 5 years

**DATA ENCRYPTION** section (collapsed):

**CREATE**    **CANCEL**

Figure 12.13 – Create a Cloud Storage bucket for logs

In this step, we create a storage bucket and specify the details based on our requirements. Refer to *Figure 12.13*, where we have selected a globally unique name: `my-log-bucket-for-demo`.

2. Next, you can specify the location, which can be **Regional** or **Multi-region**, based on your requirements.
3. We then set **Default storage class**, select **Uniform** access control, and finally set **Retention policy (best for compliance)**, which is the best option for compliance when storing logs. You can specify a log retention period in seconds, days, hours, months, or years. In our example, we set it to 5 years.
4. Next, we configure object lifecycle management for the Cloud Storage bucket.

The screenshot shows the Google Cloud Platform interface for managing a Cloud Storage bucket named "my-log-bucket-for-demo". The left sidebar shows "Cloud Storage" selected under "Buckets". The main pane displays the "Bucket details" for "my-log-bucket-for-demo". At the top, there are tabs for "OBJECTS", "CONFIGURATION", "PERMISSION", "PROTECTION", "LIFECYCLE" (which is currently selected), "OBSERVABILITY", and "INVENTORY REPORTS". Under the "LIFECYCLE" tab, a note states: "After you have added or edited a rule, it may take up to 24 hours to take effect." Below this, a section explains that lifecycle rules let you apply actions to a bucket's objects when certain conditions are met, such as switching objects to colder storage classes when they reach or pass a certain age. It includes a link to "Learn more". A list of rules is provided, with "Rules", "ADD A RULE", and "DELETE ALL" buttons above a table. The table has columns for "Action", "Object condition", and "Works with". A message at the bottom of the table says: "You haven't added any lifecycle rules to this bucket."

Figure 12.14 – The LIFECYCLE tab

Once you have created the bucket, you can define the object lifecycle rule. You can then move to the **LIFECYCLE** tab, as shown in *Figure 12.14*.

5. Next, click on **ADD A RULE** and specify your policy. We will add two rules: after 60 days, move logs to **Nearline storage**, and after 120 days, move logs to **Coldline storage**.

The screenshot shows the 'Bucket details' page for 'my-log-bucket-for-demo'. The left sidebar has 'Cloud Storage' selected under 'Buckets'. The main area shows the bucket's configuration: Location (us), Storage class (Standard), Public access (Not public), and Protection (Retention: 5 years). Below this are tabs for OBJECTS, CONFIGURATION, PERMISSION, PROTECTION, LIFECYCLE (selected), OBSERVABILITY, and INVENTORY REPORTS. A note says 'After you have added or edited a rule, it may take up to 24 hours to take effect.' Under 'Lifecycle rules', it says 'Lifecycle rules let you apply actions to a bucket's objects when certain conditions are met – for example, switching objects to colder storage classes when they reach or pass a certain age.' A link 'Learn more' is provided. Below this, a table lists two rules:

Action	Object condition	Works with
Set to Nearline	60+ days since object was created	
Set to Coldline	120+ days since object was created	

Figure 12.15 – Add a rule for lifecycle management

Figure 12.15 illustrates what your rules look like once you have defined the criteria.

6. Turn on audit logging for all Google Cloud services.

Using the following, you can turn on audit logging for all your services:

```
"auditConfigs": [
  {
    "service": "allServices",
    "auditLogConfigs": [
      { "logType": "ADMIN_READ" },
      { "logType": "DATA_READ" },
      { "logType": "DATA_WRITE" }
    ]
  }
]
```

7. Configure the logging export.

Next, we'll create aggregated exports and fine-tune the logging filters to export audit logs, data about virtual machines, data about storage, and data about databases. There are also Admin Activity and Data Access logs, as well as logs for certain types of resources:

```
logName:"/logs/cloudaudit.googleapis.com" OR  
resource.type:gce OR  
resource.type=gcs_bucket OR  
resource.type=cloudsql_database OR  
resource.type=bigquery_resource
```

You can create a sink either using the Google Cloud CLI or the Cloud console. We looked earlier at the Console option to configure a sink and the respective filters. Now we will look at the Google Cloud CLI version to demonstrate how you can create a sink and the filters:

```
gcloud logging sinks create gcp_logging_sink_gcs \  
storage.googleapis.com/gcp-logging-export-000100011000 \  
--log-filter='logName: "/logs/cloudaudit.googleapis.com" OR  
\\  
resource.type:\"gce\" OR \  
resource.type=\"gcs_bucket\" OR \  
resource.type=\"cloudsql_database\" OR \  
resource.type=\"bigquery_resource\"' \  
--include-children \  
--organization=324989855333
```

8. Set up an IAM policy and permissions for the bucket.

Now we will add a service account to the logging export bucket with storage object creator permissions:

- In the Cloud console, open **Buckets**.
- Select the **my-log-bucket-for-demo** bucket.
- Click **Show info panel**, and then select the **Storage Object Creator** permissions.

You have now successfully created a log export using a Google Cloud Storage bucket with object lifecycle rules based on your compliance requirements.

## Log compliance

Log compliance is an important topic, and we are going to briefly discuss it here as it is also relevant to the certification exam. It is important to make sure logs are centrally located, secure, and can only be accessed by the right resources (by applying the principle of least privilege). Log compliance is critical for customers who are regulated. Here are some ways to achieve log compliance.

**Separation of duties:**

- Use aggregated exports to a different project that is owned by a different team
- Cloud Storage is recommended for log archival

**Least privilege:**

- Grant the minimum required permissions on a project, so users cannot remove the logs
- Enforce the Bucket Policy Only feature (effectively disabling object-level permissions)

**Non-repudiation:**

- In the case of Cloud Storage, configure Bucket Lock to ensure the retention of objects within the bucket. This prevents deletion of the bucket or underlying objects by anyone until all objects have reached the end of their retention period.
- Additionally, enable object versioning on the bucket.

Google Cloud provides a comprehensive set of solutions to ensure your organization's logs are secured, monitored, and compliant with industry regulations.

## Logging and auditing best practices

In this last section, we will cover some logging and auditing best practices. From an exam perspective, you are not really tested with questions on these practices, but you may find indirect references to them. Since Data Access audit logs are not enabled by default, it's important that you review and enable Data Access audit logs for relevant services.

**Note**

You can find more details on how to enable these logs at <https://packt.link/w0ZWY>.

Let's look at some best practices for logging and auditing:

- Ensure that you specify the correct log storage destination—Google Cloud Storage for archival and retention and BigQuery for analytics and forensics
- Configure appropriate permissions for the access of your logs
- Ensure that sensitive data is stripped from the logs before they are stored
- Periodically review your audit logs that are stored in Cloud Logging, BigQuery, or your third-party SIEM solutions

As the volume of logs grows over time, managing them well and having appropriate controls is required early on. Logs play a key role in incident response and meeting regulatory requirements. Therefore, it is essential to demonstrate these best practices as you build your logging and auditing capabilities.

## Summary

In this chapter, we looked at the different categories of logs and what types of logs are available in each of those categories. We also looked at key properties of the different types of logs, such as whether they are enabled by default and the default retention period associated with each category. We also looked at the key components of Cloud Logging, such as the Cloud Logging API, the Log Router, log producers, and log consumers. Then, we looked at how to create log exports, how to specify the filters on what you want to exclude or include, and the different destinations that are available to store your logs. We covered the use of logging pipelines for analytics and storage. Finally, we looked at how to address compliance needs through logging and discussed best practices for logging and auditing.

In the next chapter, we will learn about Cloud Monitoring. That chapter will build on top of the knowledge gained in this one and introduce the topics of log-based metrics and alerting policies.

## Further reading

For more information on logging, refer to the following links:

- Design patterns for exporting Cloud Logging data: <https://packt.link/gLqqx>
- Logging, Monitoring, and Observability in Google Cloud: <https://packt.link/1EnJt>
- Exporting Cloud Logging data for security and analytics: <https://packt.link/mF2Tg>
- Exporting Cloud Logging data to Elasticsearch: <https://packt.link/dvg2g>
- Exporting logging data to Splunk: <https://packt.link/t9ipu>
- Using BigQuery and Cloud Logging to analyze BigQuery usage: <https://packt.link/tmCSD>
- Monitor and Log with Google Cloud Operations Suite: <https://packt.link/0v0dh>