5

Understanding Google Cloud Identity

In this chapter, we will look at Google Cloud Identity, which is Google's **Identity as a Service** (**IDaaS**) and **Enterprise Mobility Management** (**EMM**) product. We will cover aspects such as directory management, how to create and manage user accounts and groups, and how to sync directory services such as Active Directory using **Google Cloud Directory Sync** (**GCDS**). There are other features and services that will be covered, including **Single Sign-On** (**SSO**) and device and application management.

Furthermore, we will look at how you can use Google Cloud Identity to enforce **2-step verification** (**2SV**), password management, session management, and reporting and admin log activity. As the topics within Cloud Identity are very broad and cover some aspects that are related to Google Workspace (formerly known as G Suite), we will limit our discussion in this chapter to the topics that are relevant to the Google Professional Cloud Security Engineer exam.

In this chapter, we will cover the following topics:

- Overview of Cloud Identity
- Account security, such as configuring users, two-factor authentication, session management, and SAML
- How to configure Google Cloud Directory Sync

Overview of Cloud Identity

Google Cloud Identity is different from some of the other cloud security products that we will cover in this book. What makes it different is that it covers two different platforms: **Google Workspace** and **Google Cloud**. Google Workspace is out of scope as it's not covered in the Google Professional Cloud Security Engineer exam; the features and aspects that we will cover will only pertain to the use of Cloud Identity with regard to Google Cloud.

First, let's understand a few aspects of Cloud Identity. Cloud Identity is accessed via a separate console (`admin.google.com`). Cloud Identity is also the first product that you will interact with when you configure your Google Cloud environment, as the super administrator account exists in Cloud Identity. There'll be more on the super administrator account later in this chapter. Cloud Identity only provides an authentication service and not authorization. The authorization aspect is covered by Google Cloud **Identity and Access Management** (**IAM**), which is accessed via the Google Cloud console (`console.cloud.google.com`).

Before we get into the configuration and setup of Cloud Identity, let's look at some of the key features to better understand Cloud Identity. As mentioned earlier, Cloud Identity is the same IDaaS that powers Google Workspace, and it's available both as a free version and a premium version. The premium version has no cap on the number of users, although you can ask Google in the free version to increase the number of users from the default 50-user license. The other differences include the premium version's support for Secure LDAP, access to the Google security center, mobile device management, auto-exporting audit logs to BigQuery, and session length management, which lets you control the duration of the authentication tokens. Finally, the free version comes with no **service-level agreements** (**SLAs**), whereas the premium version offers a 99.9% SLA.

Now that we know about the key differences between free and premium Cloud Identity, let's take a quick look at some of the features of Cloud Identity that we will cover in this chapter.

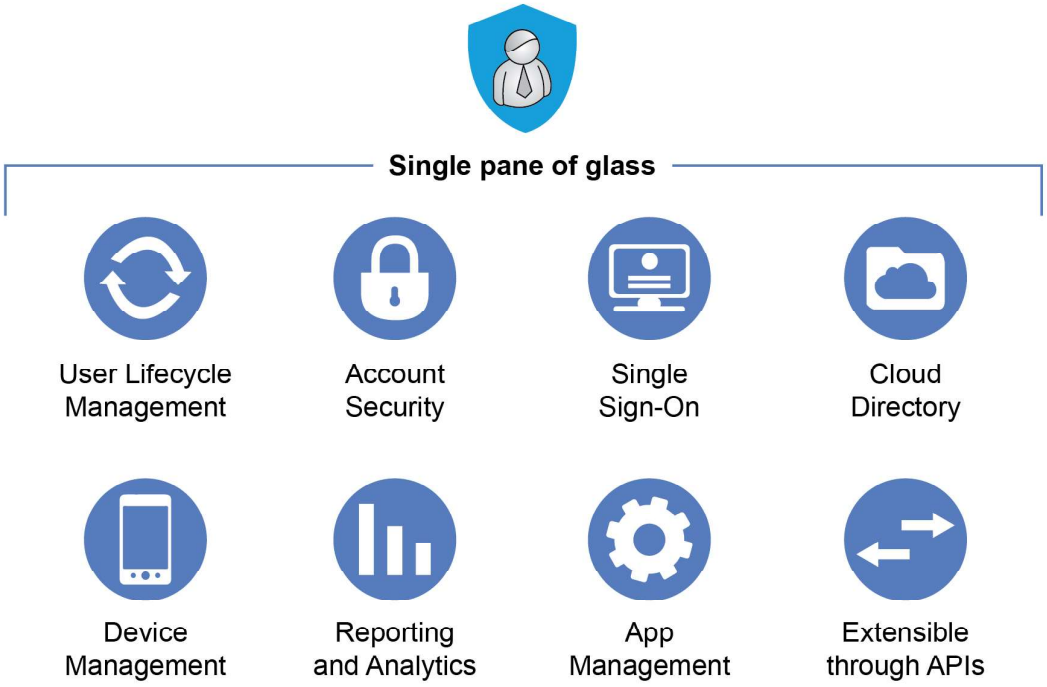

Figure 5.1 – Google Cloud Identity features

The preceding figure captures the key features of Google Cloud Identity. Let's look at some of the features that are more relevant to Google Cloud. In terms of user lifecycle management, you can create or import user accounts into a cloud-based directory. If you use Active Directory in your on-premises environment, you can use the GCDS tool to configure Active Directory and import users, groups, and memberships to Google Cloud Identity.

You can control the process of provisioning and de-provisioning as users join the organization, change roles, or leave. Account security lets you enforce the use of 2SV, apply password policies, and turn on/off access to the Google Cloud console. You can also use Google Authentication or SSO with an external **Identity Provider** (**IdP**), such as Okta or Ping Identity. We will look at SSO in more detail later in this chapter.

In the next section, we will cover how to get started with Cloud Identity, beginning with setting up your domain verification. Domain verification is *not* covered in the exam and is included here only for context.

Cloud Identity domain setup

Before we begin, remember that Cloud Identity is accessed via `admin.google.com`; for the purpose of segregation of the admin role, it is recommended to set up a different domain for Google Cloud if you already have a Workspace domain. Separating the Google Cloud account from the Google Workspace account is a good idea due to governance and compliance requirements.

Configuring a new domain includes the following steps:

1. Sign up for Cloud Identity. Provide your personal details, such as your name, email address, business, geographical region, and domain name.
2. The next step is to perform a domain verification, to ensure that you are the owner of the domain that you are trying to set up.

   If you know who your domain registrar is, you can select the name and then follow the guidance from the Cloud Identity web interface to verify the domain. There is registrar-specific guidance provided, which can be accessed from this link: `https://packt.link/QiqiK`.

   If you do not have your registrar listed and need generic guidance on how to verify your domain using the method of adding a TXT record with a verification code generated from Google Workspace, you can follow the instructions provided here: `https://packt.link/H2aNx`.

As mentioned earlier, these steps are listed for completeness so that you are aware of the Cloud Identity domain setup. Next, we will look at the super admin user role, how to secure access, and some best practices before we move on to understanding how users and groups can be created and managed.

Super administrator best practices

Cloud Identity has a super admin role, which is not the same as the Google Cloud IAM Organization Administrator, but the super admin is by default an Organization Administrator and has administrator privileges for both platforms. The super admin is managed via `admin.google.com` and has the access rights to manage users, groups, and security settings for Cloud Identity. It is also used to delegate access to the Cloud IAM Organization Administrator role. For all purposes, your super admin should create a Cloud IAM Organization Administrator role, but it should never be used to access `console.cloud.google.com` to provision or manage Google Cloud resources.

Given the importance of a super administrator, there are some best practices that should be followed. These best practices include the following:

- Always enable 2SV/multi-factor authentication for the super admin account.
- Keep a backup security key in a secure location.
- Disable device trust and limit the web session length.
- To ensure resilience and avoid over-reliance on a single individual, it is recommended to limit the usage of super admin accounts and keep the number of such accounts to a maximum of 4. This precautionary measure mitigates the risk associated with a single point of failure and ensures an adequate number of backup super admins in case of emergencies.
- Set recovery phone and email options, and ensure the recovery email is secure and protected with 2SV.

> **Note**
>
> Domain DNS access to verify domain ownership is needed in the event that you lose access to super admin accounts if a recovery email/phone was not set up.

- Create a dedicated super admin account and lock it away. This is a break-glass account to be used only in emergencies.
- Assign Cloud Identity admin roles for the delegated administration of users, groups, and so on.
- Delegate the setup and management of Google Cloud organization resources to other administrators and assign fundamental IAM roles to them to ensure the separation of duties.
- The Cloud Identity super admin will always bypass any SSO; they will always authenticate with Google Cloud Identity directly.

The topic of Google Cloud Identity is very broad and some areas overlap with Google Cloud Workspace features and functions. In this section, we only covered what is relevant from a Google Cloud Professional Security Engineer exam perspective. You can refer to the documentation for Cloud Identity if you wish to learn further. Next, we will cover some account-security-related features of Cloud Identity.

Securing your account

Google Cloud Identity provides a number of different options that can help you secure your account and enforce strong security controls. In this section, we will look at how to enforce 2SV using security keys, enforce a password policy and password recovery options, and configure user security settings such as session length, as well as doing a walk-through of the Google security center.

2-step verification

With 2SV, users log in to their accounts using their username and password (also referred to as *something the users know*) as well as a second factor (*something they have*), which could be a physical security token or a mobile phone that can generate a key. Google Cloud Identity supports a number of methods that can be used as a second factor for authentication. These methods include the following:

- **Security keys**: A physical security key, such as Google's Titan Security Key or a YubiKey.
- **Google prompt**: Users can set up their mobile phone (iOS or Android) to send a sign-in prompt. When a user attempts to sign in and successfully completes the first step of entering a username and password, the user then gets a sign-in prompt on their authorized device to confirm whether it's the user who has initiated a sign-in.
- **Code generators**: Another option is to use code generators, such as Google Authenticator, Authy, and so on. Users can use code generators to generate a one-time code to be used as a second factor.
- **Backup code**: An offline method can be used where the user does not have access to a mobile phone to generate a code. They can print backup codes and use them instead. Backup codes are often used for recovery.
- **Text or call**: You can also opt to have your one-time password sent to you via **Short Message Service** (**SMS**) on your mobile phone, or you can receive a phone call on your authorized device with the one-time code.

There are a variety of choices for what type of 2SV can be configured. You can use any combination of 2SV options. Next, we will take a look at how you enforce 2SV in the web interface.

Before you enforce 2SV, you have to ensure that users are aware that they need to enroll for 2SV and choose their verification methods.

Next, let's look at how you can enforce 2SV from the console:

1. Log in with your credentials to `admin.google.com`.
2. Navigate in the left-hand side menu to **Security | Authentication | 2-step verification** (see *Figure 5.2*). Once in the settings, select the organizational group from the left where you want to enforce the 2SV policy.

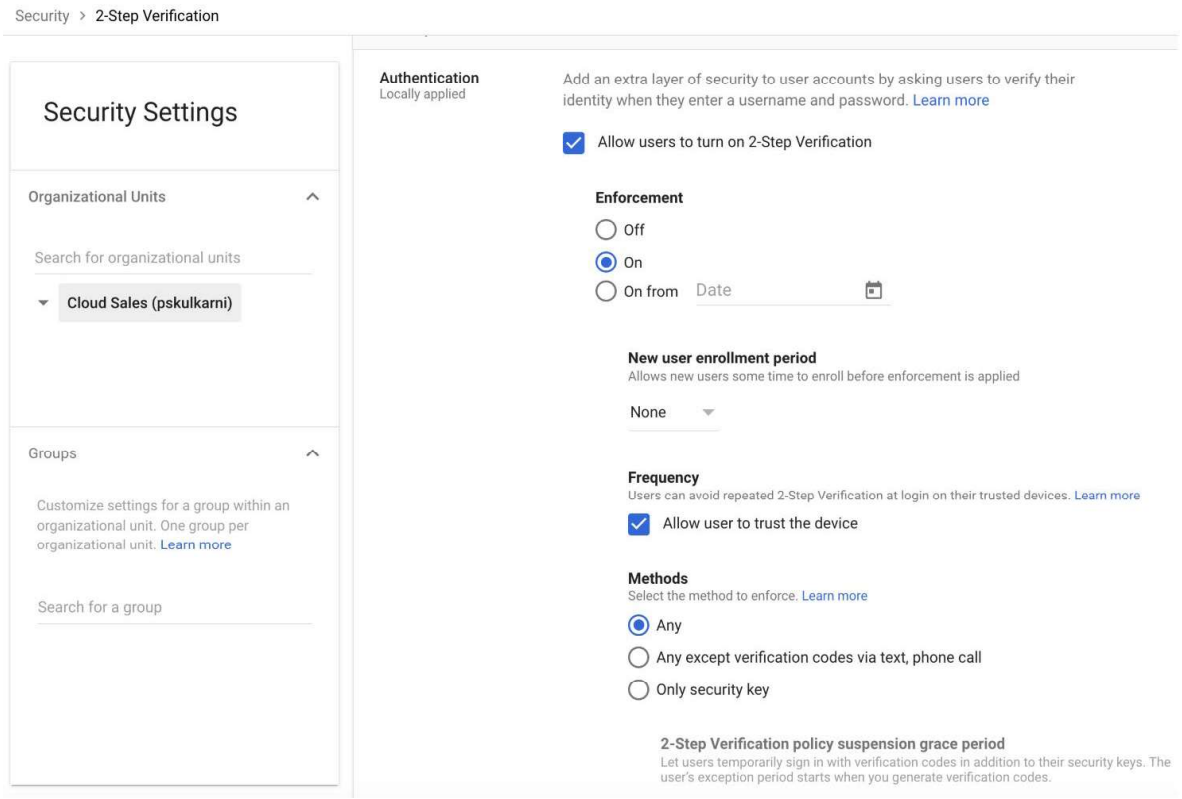

Figure 5.2 – Account security (2SV)

3. From the configuration options available here, you can set **Enforcement** to **On** or **Off** or schedule it to turn on for specific data. Selecting **On** will enforce it immediately.

4. The other important configuration is to let users specify whether the device they are logging in from is a trusted device. With this setting in place, when users first log in, they will be prompted with a username and password and will be asked whether they want to remember the device. If the user selects this option, then the next time they log in, they will not be prompted for 2SV. This will remain in place unless the user deletes the cookies or user access is revoked.
5. As part of your 2SV policy, you can also create an exception group for users to whom the policy may not apply. This can be done for a variety of users; although it is not recommended, you do have the option to configure this.
6. Next, you can give your users time to enroll in 2SV, from zero to six months. This helps you in rolling out 2SV if it's being done for the first time. Users will get time to configure their 2SV, and after that, they will be able to log in. If they do not do that by the enrollment deadline, they will be unable to log in.
7. Finally, you have the option to select the verification methods. As a best practice, selecting **Only security key** is recommended as it is the most secure option. But you are not limited to that and have the option to configure other options, such as **Any except verification codes via text, phone call**, which will enable all supported verification methods. Based on your organizational requirements, you can select these options.
8. Other configuration options include the ability to configure a suspension policy and the ability to use backup codes.

> **Note**
>
> You can read more about these settings using the following link: `https://packt.link/BVasl`. For the purpose of the exam, you should be able to configure a 2SV policy as explained earlier.

User security settings

As an administrator, you have the ability to view a user's settings and perform actions such as resetting passwords, adding or removing security keys, resetting user sign-in cookies, turning temporary challenges off, or viewing and revoking application-specific passwords or access to third-party applications. Let's take a quick look at where you can access these settings from:

1. Log in via `admin.google.com`.
2. Navigate to **Directory | Users** and select the username you want to view the security settings for, which can be done by clicking on the name.
3. Once you are on the user's details page, you can see multiple options, such as user details, groups, permissions, and security settings. For this exercise, we will be navigating to the security settings.

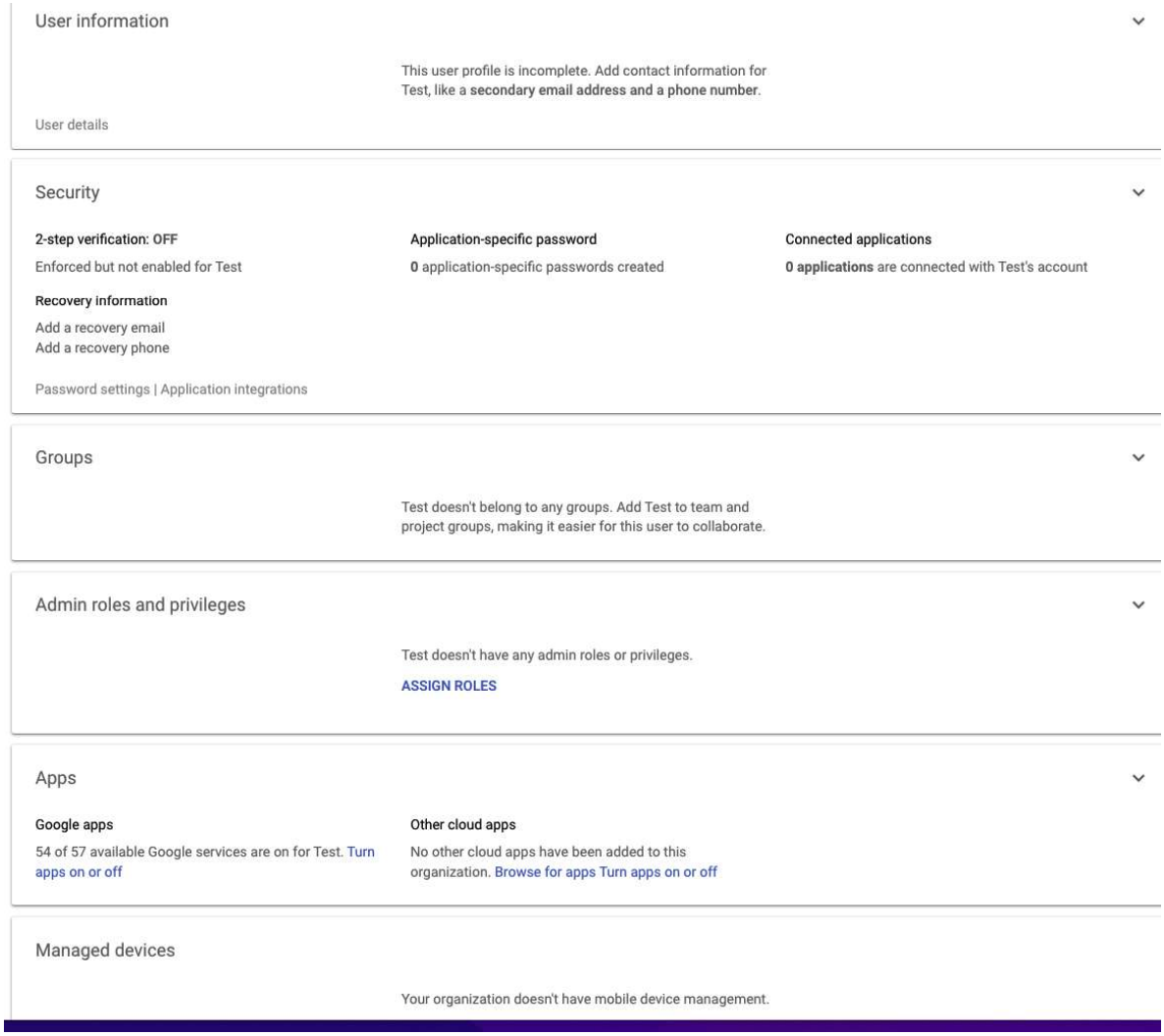

Figure 5.3 – Cloud Identity – user details

4. On the **User information** page, scroll to the **Security** section and click on it to be taken to the security settings for the selected user.

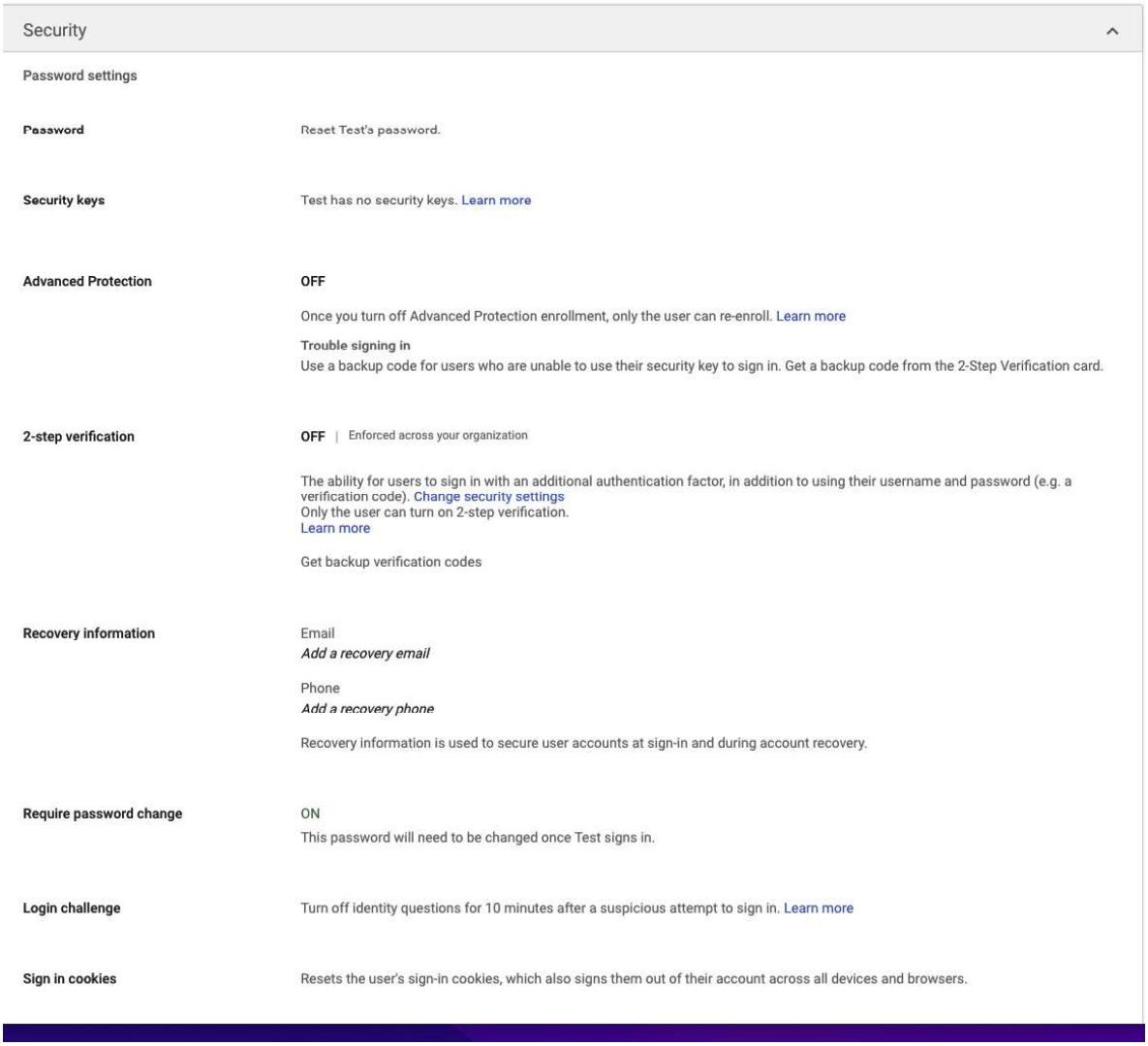

Figure 5.4 – Cloud Identity – user security settings

5. From here, you can view and configure multiple security settings. We will briefly look at what each one of them does:

    - **Reset password**: This lets you reset the user's password. You can create a password automatically and view the password. When you email the password to the user, you can set **Change password on next sign-in**. You can also copy and paste the password if needed, although that is not recommended; you should email the password from the console.
    - **Security keys**: You can add or revoke security keys for the user.
    - **2-step verification**: Only a user can turn this setting on. As an administrator, you can only check the 2SV settings or, if required, get a backup code for a user that has been locked out.
    - **Require password change**: You can enforce a password change for a user.

Other settings include viewing or revoking application-specific passwords, and you can also revoke third-party application access.

> **Note**
>
> To read more about the user security settings, you can use this link: `https://packt.link/HUyYn`.

Session length control for Google Cloud

As an administrator, you can configure the session length of Google Cloud services. You do have the option to configure it for Google services as well, such as Gmail, but this is not in our scope, and we will only focus on configuring the session length for Google Cloud services. There are a variety of use cases for which you would want to enforce session length. Examples include access by privileged users or billing administrators who are subject to session length restrictions – you might want them to sign in more frequently by timing out their sessions.

You do have the option to create groups to which this policy will not apply. When enforced, this policy applies to Google Cloud console access, the `gcloud` command-line tool (the Google Cloud SDK), and any application, whether owned by you or a third party, that requires user authorization for Google Cloud scopes (`https://packt.link/Q1sR2`). Note that Google Cloud session length control does not apply to the Google Cloud mobile app. We will next see how you can enforce session length from the admin console:

1. Log in via `admin.google.com`.
2. Navigate to **Security | Access and Data Control | Google Cloud session control**.

   Refer to the following screenshot to see the multiple options that are available to us to configure.

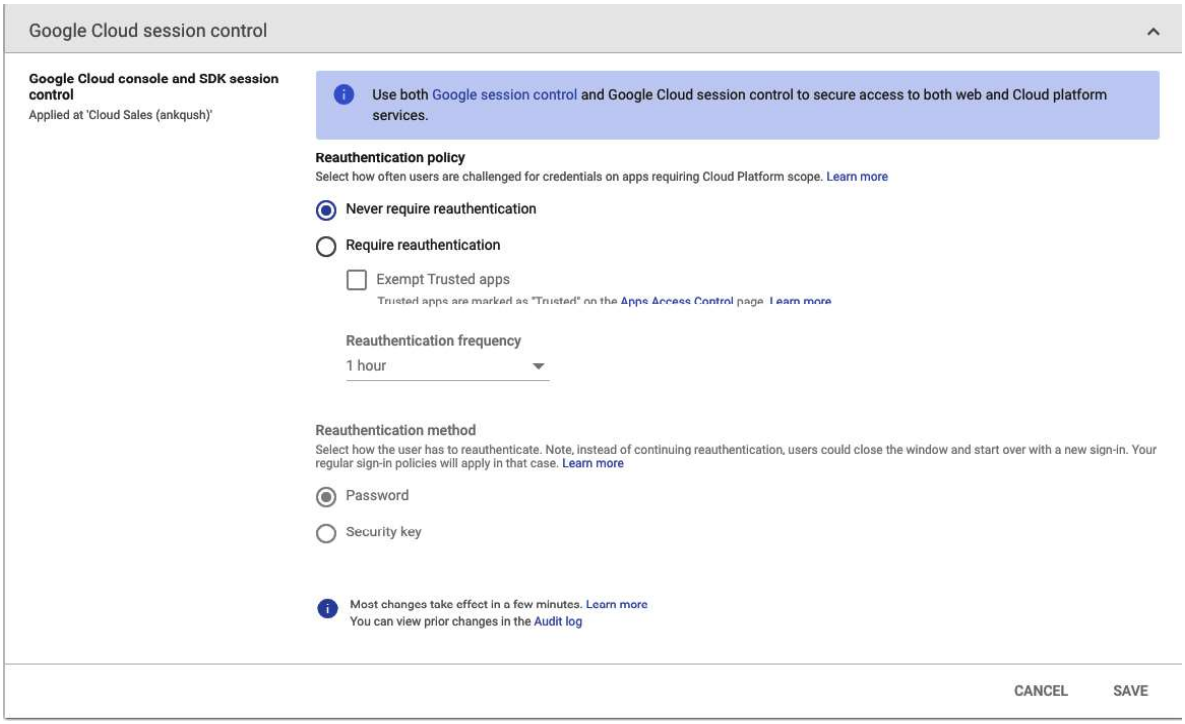

Figure 5.5 – Google Cloud session control

3. From the left-hand side menu, select the organizational unit and the user group to which you want this policy to apply. As mentioned earlier, you can apply this policy selectively to users in a certain group and not subject all users to session control, having different session control settings for standard users versus privileged users.

4. Let's take a look at other options available to us that we can configure:

    - **Reauthentication policy**: Here, you select **Require reauthentication** and then specify **Reauthentication frequency**. The minimum frequency is one hour and the maximum is 24 hours. There is an option to exempt trusted apps from the reauthentication policy. You can configure which applications are trusted by navigating to **Security | Access and Data Control | API controls**. From there, you can manage trust for third-party applications.

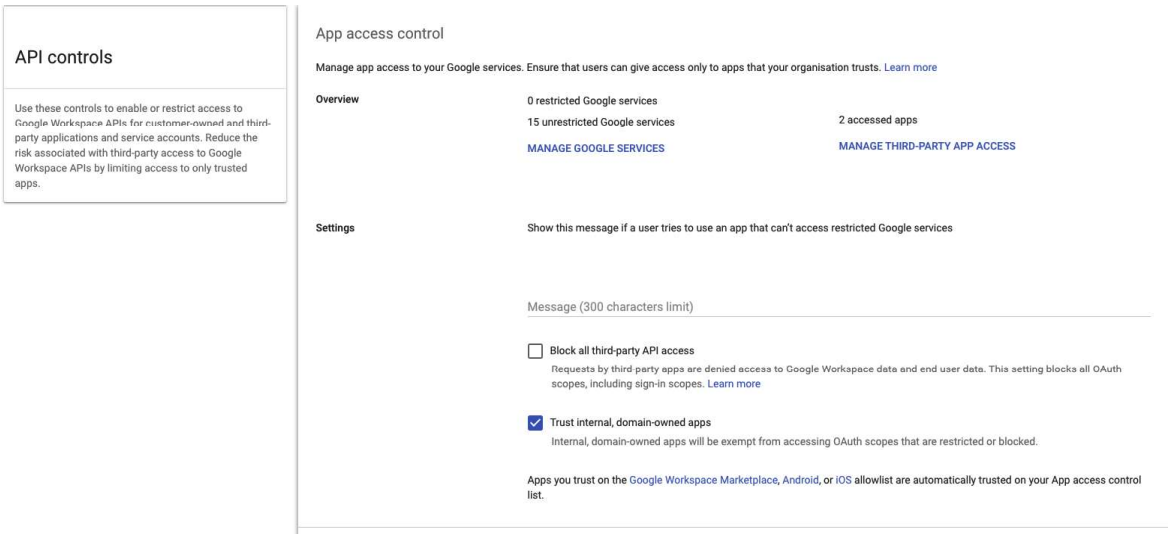

Figure 5.6 – Managing trusted third-party applications

    - Next, you can configure **Reauthentication method**. Here you can specify either password or security keys. This depends on your use case. It's highly recommended that you enforce security keys for privileged users.

> **Note**
>
> You can read more about Google Cloud session length here: `https://packt.link/B2wvT`.

SAML-based SSO

With **SSO**, you can configure your enterprise cloud applications to let users sign in using their Google-managed credentials; in other words, Google Workspace will act as an identity provider. There are two different options for you to configure SSO: pre-integrated or custom. With the pre-integrated option, Google supports over 300 pre-configured applications that you can deploy from the Cloud Identity console. The custom **Security Assertion Markup Language** (**SAML**) option can be used if you have custom applications or applications that are not pre-configured.

It is assumed that you understand how SAML 2.0 works. In this section, we will only look at how to configure SSO using SAML and not cover the foundational knowledge of how Google implements SAML. If you wish to understand more about how SSO using SAML works on Google Cloud, please refer to the *Further reading* section at the end of this chapter.

For the purpose of the exam, we will walk through the steps required to configure SSO using SAML for a custom application.

Before we begin, ensure that you have the **Assertion Consumer Service** (**ACS**) URL and the entity ID, as they will be required when you configure your external IdP and optionally a start URL. Also, determine at this stage whether your external IdP supports **multi-factor authentication** (**MFA**); if it does, you can configure the option to use MFA by external IdP, but if for some reason your custom application does not support MFA, you can configure Google Cloud Identity to enforce MFA.

Now let's look at the steps required to complete the SSO setup using SAML for a custom application:

1. Begin by logging in to the admin console at `admin.google.com`.
2. Navigate from the main console to **Apps** | **Web and mobile apps**.
3. From there, select **Add app** | **Add custom SAML app**.

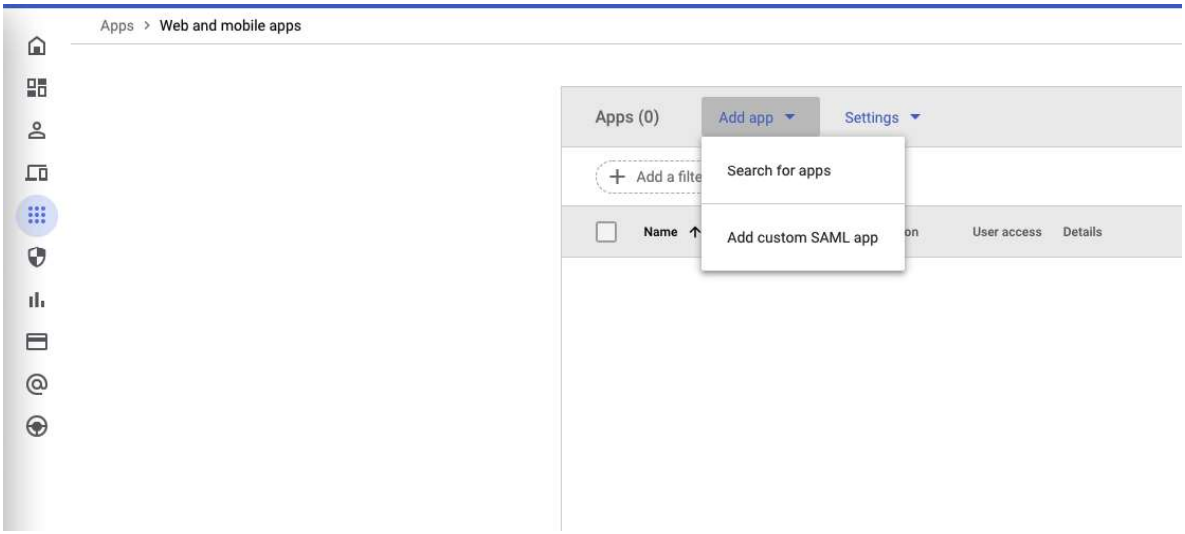

Figure 5.7 – Cloud Identity – Add custom SAML app

4. This will take you to a four-step wizard to configure your custom SAML app.

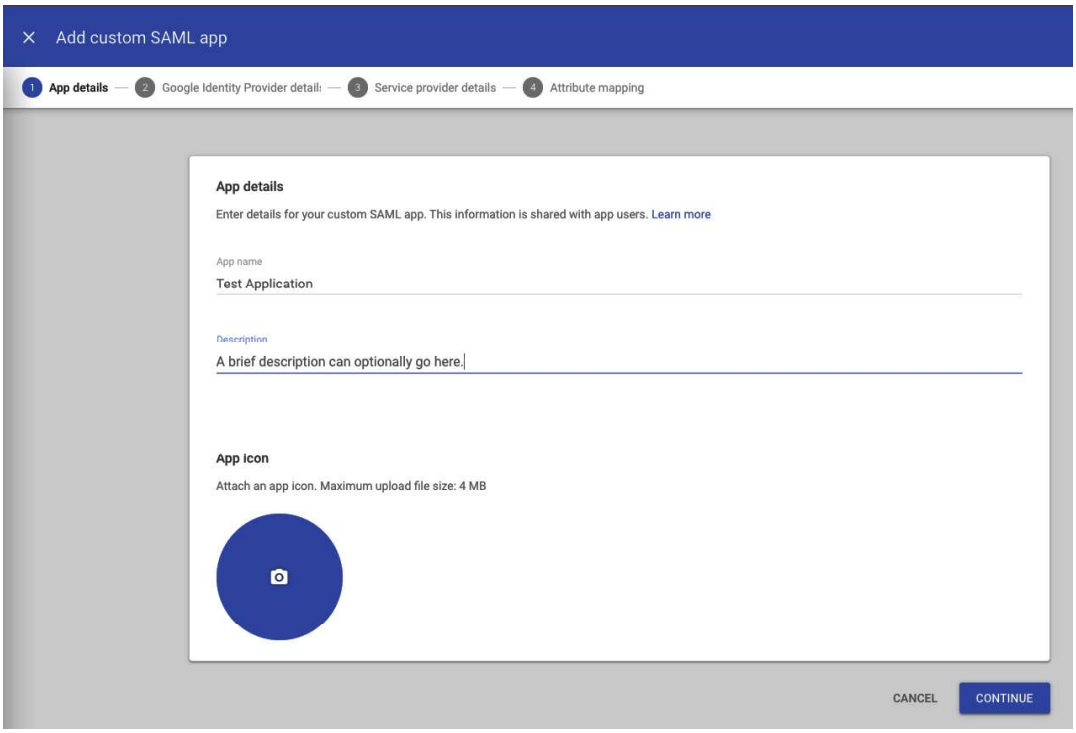

Figure 5.8 – Configure your custom SAML app

5. Follow the steps to give a name to your custom application and optionally provide a brief description and upload an icon. Once done, click **Continue**.
6. The next step will provide you with two options: either download the IdP metadata or copy and paste the SSO URL, entity, and certificate with an SHA-256 fingerprint. These are Google Identity Provider details that will be required by your service provider. Once you have decided either to download or copy and paste the information, you move on to the next step by clicking the **CONTINUE** button.
7. Optionally, you can now log in to your service provider console and copy and paste the information that was captured in *Step 6*.
8. In this step, you can fill in **ACS URL**, **Entity ID**, and optionally a start URL. The ACS URL is simply an endpoint at your service provider's end; this is where the IdP will redirect the authentication response. The **Entity ID** value is provided by your service provider. At this stage, you can also set the option to enable **Signed response**. This option enables you to sign the entire SAML authentication response instead of the default option, which only signs the assertion within the response. The default **Name ID** value is the primary email; however, you have the option to create the custom attributes that need to be created using the admin console before you start the **Add SAML custom app** process.

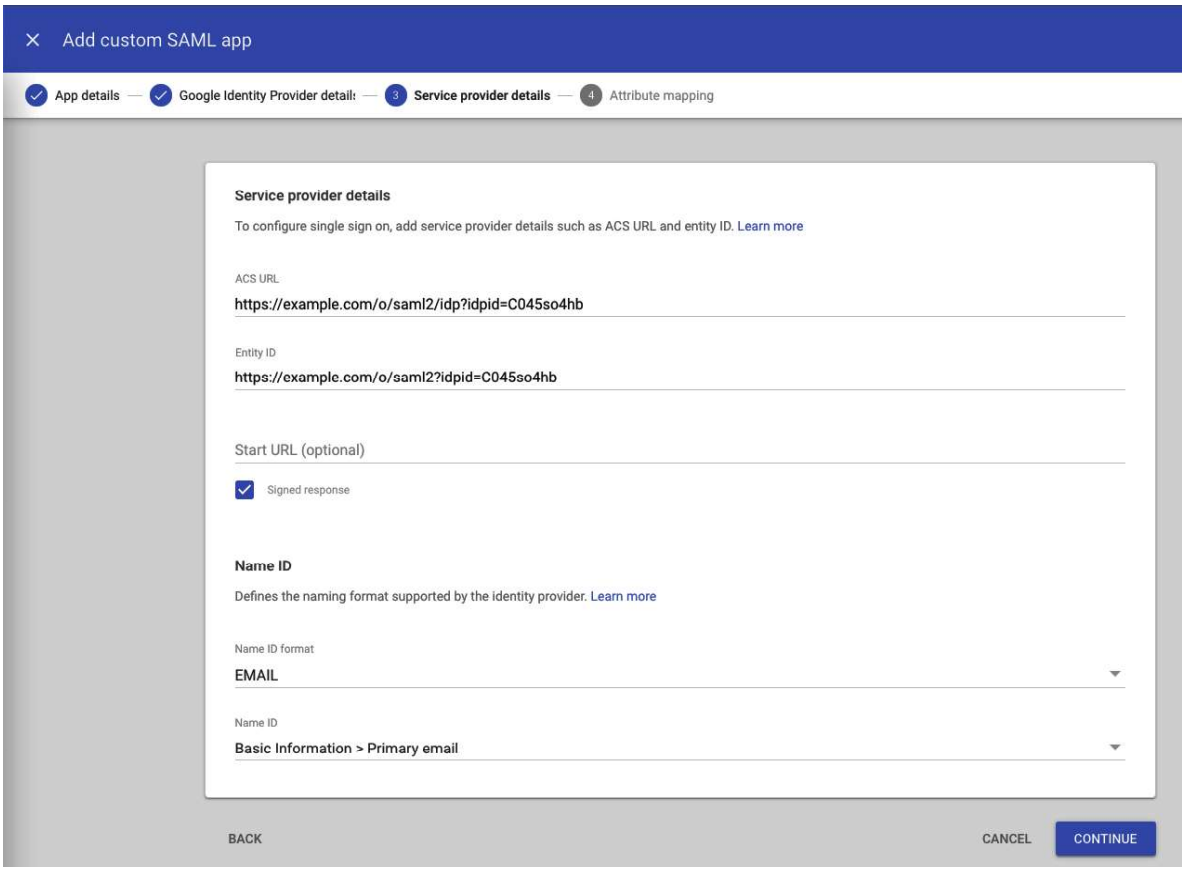

Figure 5.9 – SAML service provider details

9. The next step is optional, for you to create attribute mappings. If you don't have mappings, you can click **FINISH** to complete the setup process.

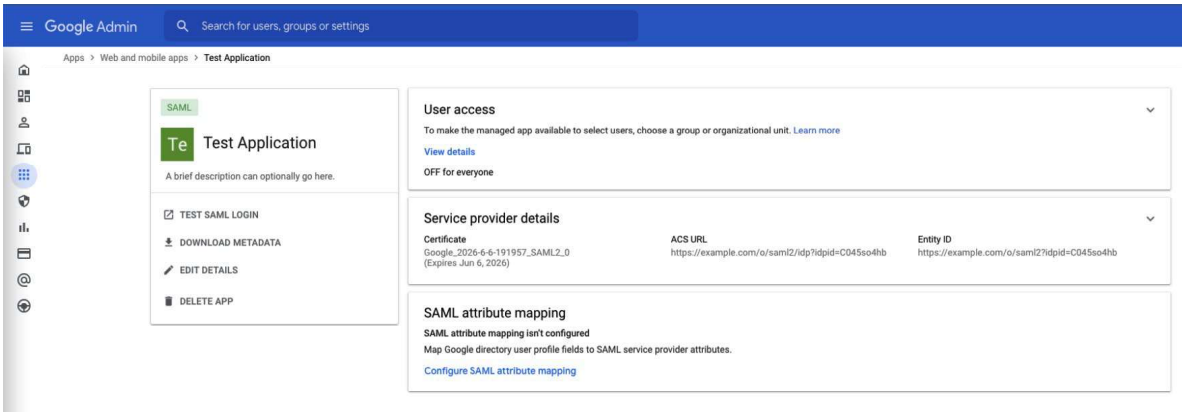

Figure 5.10 – App details page

You will be able to view the configured app on the **Apps | Web and mobile apps** page, and you can view the details by clicking on the name of the app, as seen in *Figure 5.8*.

Additional security features

There are some additional security features that are available in Cloud Identity that are not part of the exam's scope. Therefore, we will briefly cover them here for the purpose of completeness. There will be links in this section and in the *Further reading* section on some of these additional topics, should you need to refer to them. These features include the following:

- **Google Cloud security center**: The security center provides capabilities such as advanced security analytics and visibility into security-related issues, all from a central dashboard. From the security dashboard, you can view reports, such as reports on user login attempts, file exposure, DLP events, spam-related information, OAuth grant activity, user reports, and so on.

> **Note**
> You can find a full list of reports that you can view from the dashboard here: `https://packt.link/yGWLH`.

Another key capability of the security center is the security investigation tool, which lets you identify, triage, and take action on security-related events.

> **Note**
> You can learn more about the security investigation tool here: `https://packt.link/ovam0`.

Finally, the dashboard also provides you with an overview of your security health, which gives you visibility of the admin console and its related activities.

- **Monitor and enforce password requirements**: You can enforce strong passwords based on your organization's policy. This includes the length, complexity, and expiration of your passwords. You can also stop password reuse so that users cannot reuse previously used passwords. The monitoring function lets you monitor the password strength.

> **Note**
> You can read more about this topic at this link: `https://packt.link/Rol3H`.

- **Self-service password recovery**: You have two options for password recovery: either you can let users recover the password themselves or you can get users to contact an administrator to reset the password.

> **Note**
> You can read more details about this topic here: `https://packt.link/ccOZe`.

In this section, we covered some optional security features that are available for Google Cloud Identity. Next, we will cover directory management, which includes aspects of identity federation.

Directory management

This is one of the most important sections of the entire chapter. We will learn how to configure identity provisioning, in particular, how to integrate Microsoft **Active Directory** (**AD**) with Google Cloud Identity using the GCDS tool. We will look at some other directory management tasks, such as how to create users and groups and assign admin permissions and how we can provision and de-provision user access using Google Cloud Identity and third-party IdPs. Finally, we will have a look at how to automate user lifecycle management.

Google Cloud Directory Sync

This section will be a deep dive into GCDS. We will start by understanding what GCDS is, the benefits of using it, how it works, and how to configure it using Configuration Manager.

GCDS helps you to synchronize your Microsoft AD or LDAP objects, such as security users and groups, to your Google Cloud Identity account.

> **Note**
> To look at the entire list of content that is synced, you can check this link: `https://packt.link/rMBYS`. This content is synced as a one-way sync from your LDAP or AD. Your AD or LDAP is not modified in any way.

GCDS features and capabilities

Let's look at some of the capabilities that GCDS has to offer. It is important from an exam perspective to understand what GCDS can and cannot do:

- GCDS is available as a tool to download on your server. It comes packaged with all necessary components, and once installed, you can begin the setup process. There are also built-in security features in GCDS to help ensure your content is secure during the synchronization. Before any synchronization can begin, an OAuth token is created using Configuration Manager, which is used to connect to your Google account. You will further need to authorize GCDS with your Google account for authentication.
- GCDS lets you sync information such as organizational units, mailing lists, users and their related attributes (you have the option to specify this), extended user information (again, you can specify what to synchronize), and calendar information, such as rooms, contacts, and passwords.
- Using configuration management, you will be able to specify which attributes you want to include, exclude, or create exceptions for.
- To make the entire setup of using GCDS easy to use, you do have the option, if you are using Microsoft AD or Open LDAP, to use the default settings in Configuration Manager.
- With GCDS, you have the ability to create exclusion rules where you can specify what you want to include in your sync. All of this can be easily managed and configured from Configuration Manager, which we will see a bit later.

Now, let us understand how GCDS works.

How does GCDS work?

By using GCDS, your Google Cloud Identity domain's data can be automatically provisioned from your LDAP directory server by adding, editing, and deleting users, groups, and non-employee contacts. Your LDAP directory server's data is safe and secure.

Configuration Manager is a GUI-based wizard that walks you through the steps to configure synchronization. You set up your LDAP directory server settings and connection, configure the Google Cloud Identity domain settings, and select what LDAP data to synchronize and what to exclude.

The following steps explain how GCDS data flows:

1. Relevant data is exported in a list from your LDAP server. You can set up rules to specify how this list is generated.
2. GCDS connects to the Google Workspace domain and generates a list of users, groups, and shared contacts that you specify.
3. GCDS compares these lists and generates a list of changes.

4. GCDS updates your Google Workspace directory to match the AD data.
5. When the synchronization is complete, GCDS emails a report to any addresses that you specify.

Figure 5.11 illustrates how GCDS works and some of the best practices associated with it – for example, provisioning it from a single source of truth, such as your on-premises Active Directory, and running it on dedicated hardware.

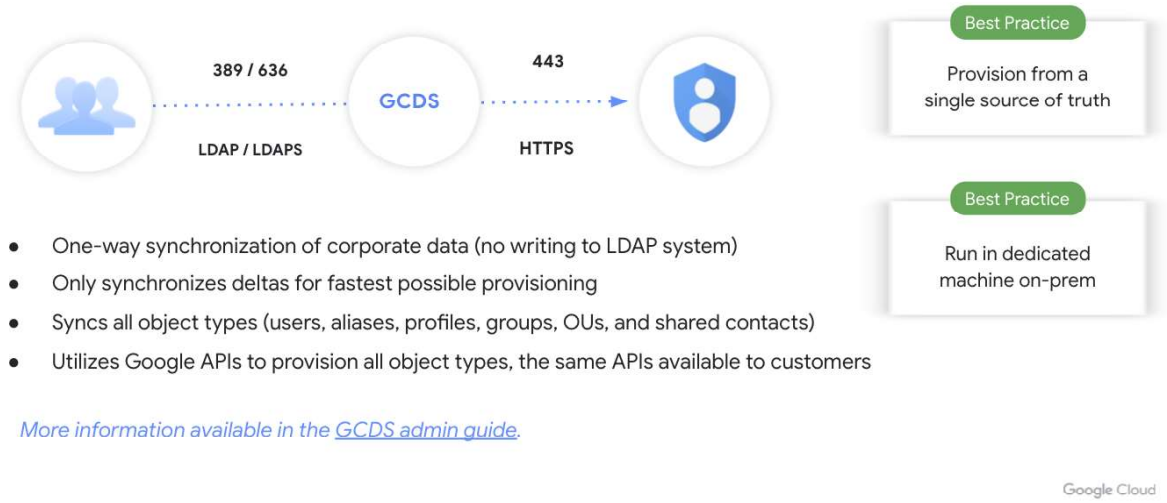

Figure 5.11 – The workings of GCDS

All connections from GCDS to LDAP are secured using TLS, and you have the option to specify that in Configuration Manager. For standard LDAP, no encryption is used. All communications between Google and GCDS are done over HTTPS.

In the next section, we will walk through Configuration Manager. Configuration Manager is used to configure GCDS to work with your user directory and Cloud Identity.

Using GCDS Configuration Manager

In this section, we will look at how to configure a sync using Configuration Manager. As we look at each step, we will explain what you can and cannot do:

1. To download and install GCDS on your server, make sure you select the correct version, such as 32-bit or 64-bit for Windows or Linux. GCDS can be downloaded from the following link: `https://packt.link/ZruGO`.

2. Once installed, you need to gather information related to your LDAP server, as that can help you configure the sync more quickly. To gather information on your LDAP structure, you can install a third-party browser such as JXplorer or Softerra LDAP Administrator. Besides information on the structure, you should also use the browser to capture information such as the LDAP base **Distinguished Name** (**DN**) and security groups. GCDS requires a base DN to search users and groups. You do have the ability to use multiple base DNs in your configuration. Similar to security groups, if you would like to synchronize them, you need to identify which ones you would like to synchronize.
3. Next, you will need to connect the following information to your LDAP server:

    I. Hostname and IP address of the LDAP server.

    II. Network-related information such as access and proxies.

    III. Whether you will be installing Secure LDAP using SSL or Standard LDAP with no encryption.

    IV. User credentials of the LDAP server with read and execute permissions.

> **Note**
> You can only get data from a single LDAP directory; if you are using multiple Microsoft AD directories, then consider consolidating or syncing from a Global Catalog or using a virtual directory product.

4. You should next clean up your LDAP directory to identify users and mail-enabled groups and review the naming guidelines for names and passwords to ensure there are no unsupported characters.
5. Next, mark all Google users in your LDAP directory. This can be done using organizational units, groups, descriptive names, and custom attributes.
6. As mentioned earlier, before you can begin configuring, you need to authorize GCDS with your Google account. The following steps will authorize the account and generate an OAuth token that will be used by Configuration Manager to connect and synchronize:

    I. Open **Configuration Manager** and then select **Google Domain Configuration**.

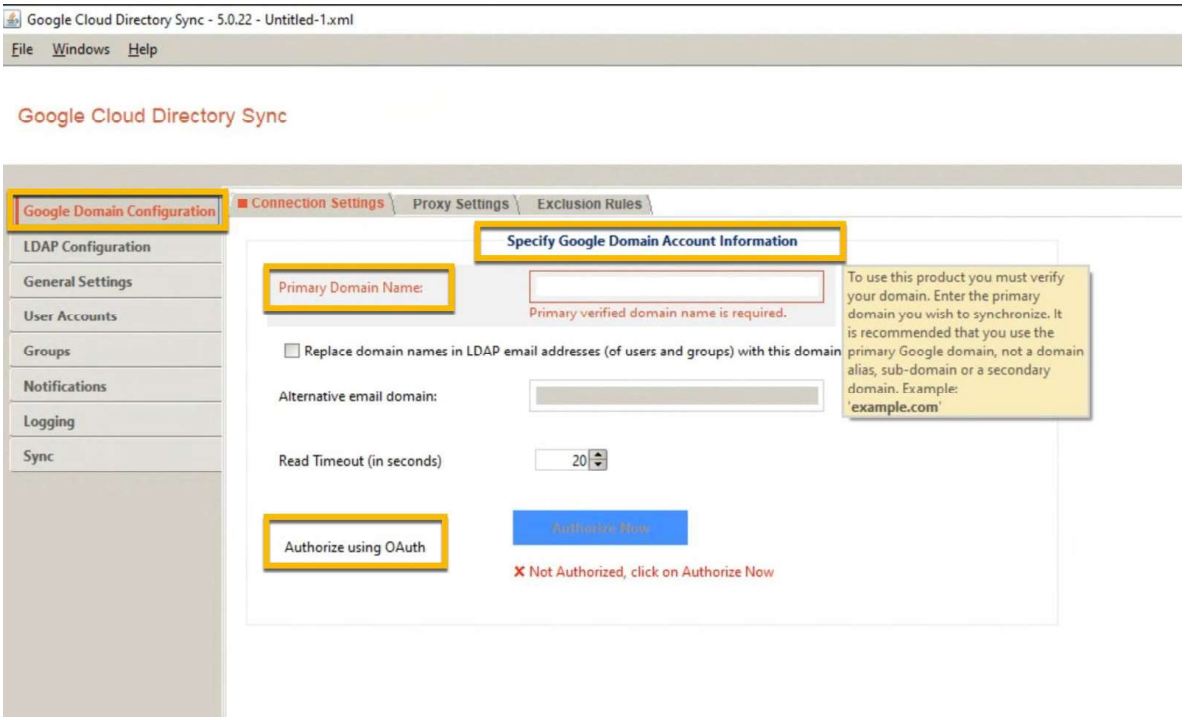

Figure 5.12 – Google Domain configuration

II. Click **Authorize Now**, which will prompt you to sign into your Google Account using your super admin credentials.

III. An OAuth token will be generated, which you will need to copy and paste into **Configuration Manager**; after that, click **Validate**.

7. In the next steps, we will configure the sync using Configuration Manager. For the purpose of this walk-through, we will be using the default settings. Configuration Manager uses a step-by-step process to set things up. The first step in preparing and planning includes identifying the information you want to synchronize, such as user accounts, groups, user profiles, organizational units, calendar resources, licenses, and so on.

8. Next, click the **Connection Settings** tab under **Google Domain Configuration** and specify **Primary Domain name** (if the domain is not verified, you will be directed to a domain verification process). Specify the alternate email address and, if applicable, authorize it by signing into your Google account using super admin credentials and the OAuth token to validate.

9. Under the **Proxy Settings** tab, specify whether you have a proxy in your network.
10. On the **LDAP Configuration** page, specify the LDAP server information. For the purpose of this exercise, we will assume that we are using Microsoft AD or Open LDAP. See the description link for examples of the accepted ways to connect to your LDAP server.

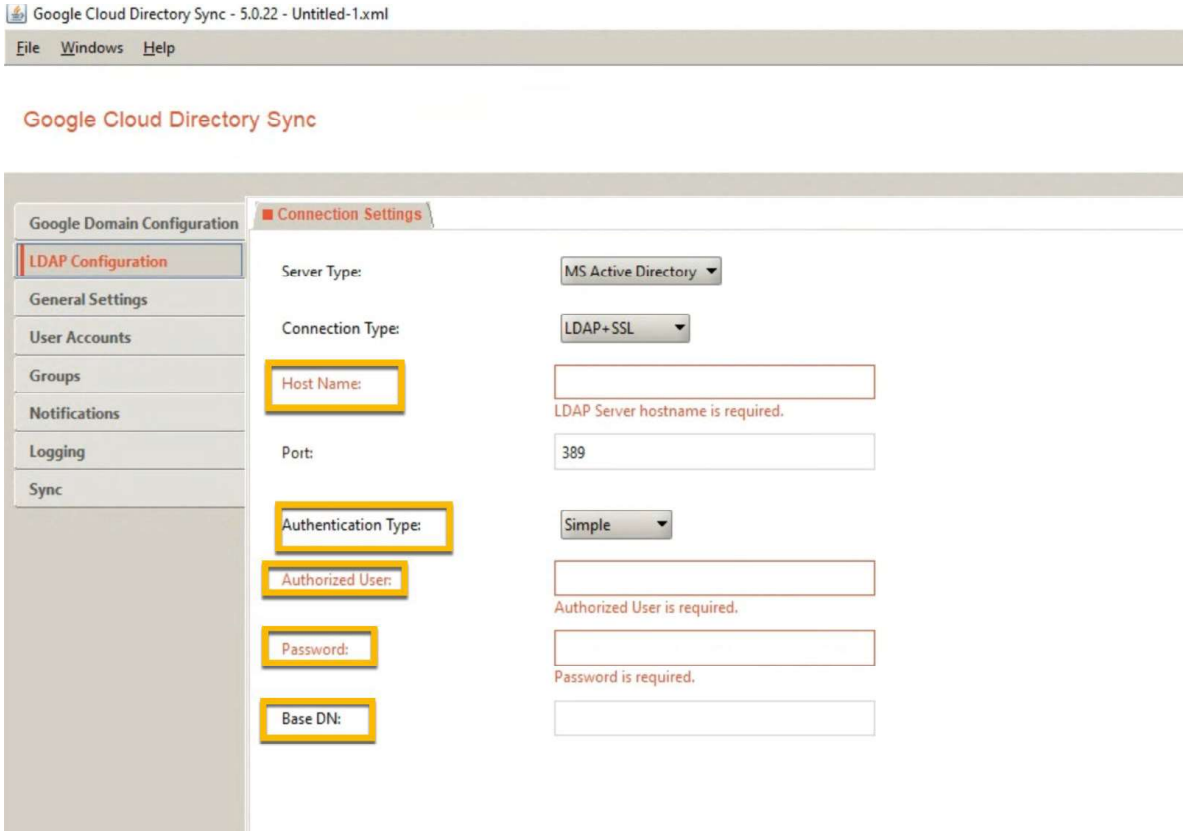

Figure 5.13 – LDAP configuration

The next few steps are for you to specify what you want to synchronize. Specify inclusions, exclusions, and any exceptions in the settings.

11. On the **Licenses** page, you can specify the license information, add the Google product, and specify the SKU and product.

> **Note**
> For further information on licenses, refer to this page: `https://packt.link/KbYfK`.

12. Next, you can configure your notification settings. Every time data is synchronized, GCDS will send an email.
13. Configure logging based on the detail you need.
14. Finally, verify your sync; at this stage, Configuration Manager will simulate and test the sync, and if there are any failures, it will ask you to address them. Once the simulation is successful, Configuration Manager will generate a report and you can then select **Sync and apply**.

This concludes the steps required to configure the GCDS tool. This lets you connect, configure, and synchronize your LDAP directory to Google Cloud Directory. We will now move on to the next section to see how user provisioning works.

User provisioning in Cloud Identity

Google Cloud requires users and groups to be pre-provisioned in Cloud Identity. There are multiple options available to help you do so. The following screenshot shows the various options, and we will briefly cover which options are recommended and why:

Method	Effort	Staff involved	Notes
Manual Provisioning	High	Google Admin	Easiest method, but not scalable
CSV Upload via Google Admin	Medium	Google Admin	More flexibility, but not scalable
Google Cloud Directory Sync (GCDS)	Medium	LDAP Admin	Integrates with LDAP, scalable, requires no programming
Third-Party Tools (Okta, Ping, etc.)	Medium	LDAP Admin	Scalable, may incur additional cost
Admin SDK Directory API	High	LDAP Admin Development Staff	Scalable, flexible, requires in-depth programming

Figure 5.14 – User provisioning options in Google Cloud Identity

Based on the methods listed in *Figure 5.14*, you can choose to manually add users to Cloud Identity as it's an easy option and requires little effort if you only have 5-10 users to add. The effort required increases if you have more users to add.

The next option is to use a CSV file where you can upload users in bulk; this is more convenient as you can add more users at once. You will, however, need to prepare your CSV file by listing the users' details, such as first name, last name, email address, password, and org path. Listing passwords in CSV files is not recommended, so many organizations choose not to select this option.

The third option is configuring Active Directory Federation using GCDS, which we discussed in detail in the previous section. The effort required is not low as it does require following a few steps, but once established, it's a very scalable and secure way to provision/de-provision users on Google Cloud Identity.

Another option, which many organizations choose, is to select a third-party provisioning solution that can be federated with Google Cloud Identity. This is very similar to using GCDS but offers a commercial third-party product. It's a scalable method but it does have costs involved.

The following architecture is a very common deployment scenario, where Google Cloud Identity is used to federate with either GCDS or a third-party IdP such as Okta or Ping Identity.

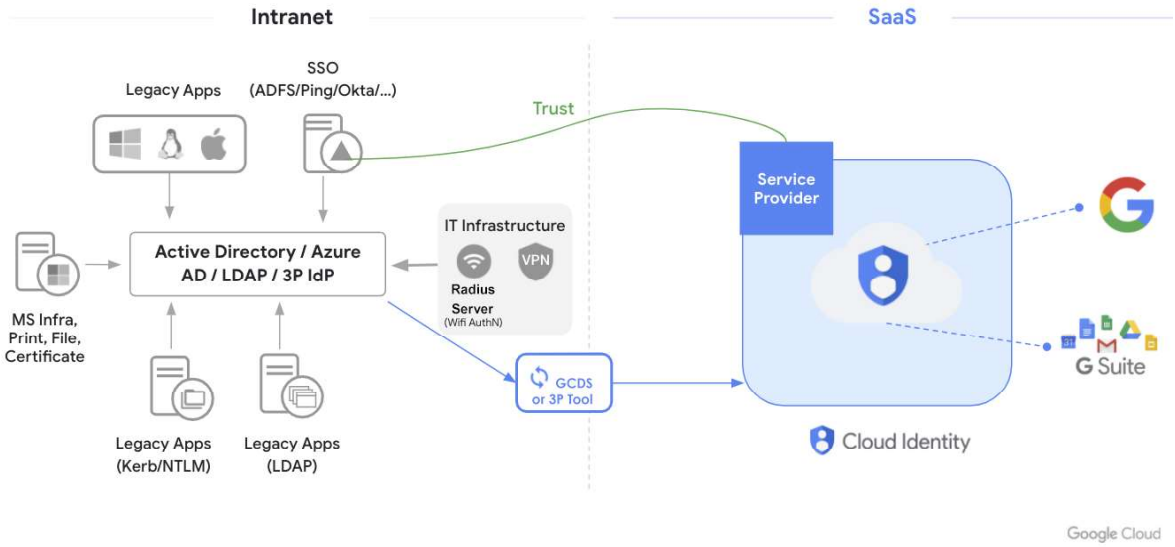

Figure 5.15 – Third party as an IdP

In the next section, we will look at how you can use Cloud Identity and federation to manage the user lifecycle.

Automating user lifecycle management with Cloud Identity as the IdP

User provisioning and de-provisioning in larger enterprises can become very complex. Users leaving the organization or moving within the organization can lead to changes in the scope of what applications a user has access to. In order to address this, Google Cloud Identity provides a catalog of automated provisioning connectors between third-party apps and Cloud Identity. In the *SAML-based SSO* section, we looked at how, by using Cloud Identity and SAML-based SSO, you can use more than 300 pre-configured applications and, if required, create custom applications as well. This option is typically used when Cloud Identity is your IdP and you want to integrate third-party apps with Cloud Identity. Let us look at this briefly.

Once your SSO using SAML has been configured, you can enable fully automated user provisioning, such as creating, deleting, or modifying a user's identity across cloud apps. Cloud Identity administrators can be configured to synchronize the subset of users and automate the entire user lifecycle management process, which further leads to a great user experience.

Let's look at an example of automated user provisioning flow. In *Figure 5.16*, we see an example of a user, Bob, who joins an organization. Even before the user's first day, the admin adds Bob to Cloud Identity as a user and adds him to the relevant organization. This gives Bob access to all the apps that particular organization's members have access to.

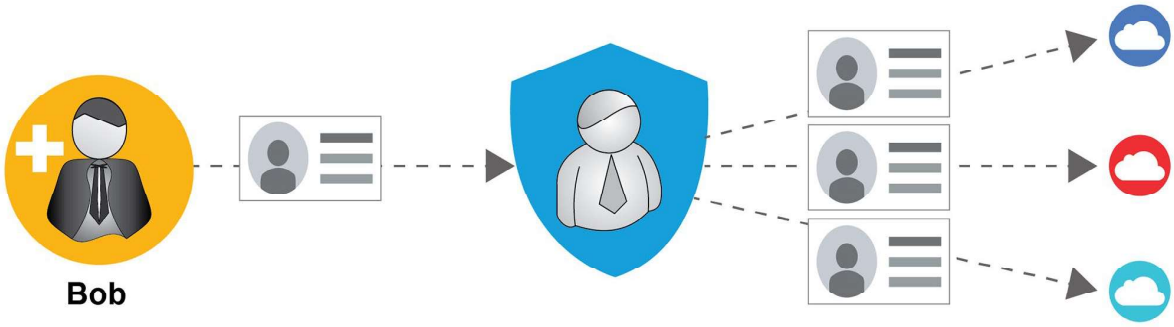

Figure 5.16 – Cloud Identity replicates Bob's identity to all allowed cloud apps

In *Figure 5.17* we see how Bob is able to sign in to all cloud apps using SSO.

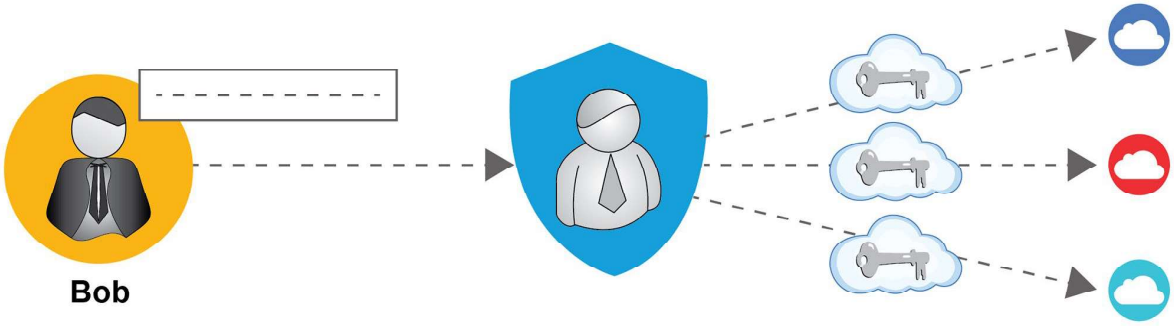

Figure 5.17 – On Bob's first day, he can sign in to all cloud apps using SSO

We have now covered the process of automating user lifecycle management. This helps large organizations to automatically provision and de-provision users and give them the right level of access to business applications. Let us now look at some of the APIs that are used by third-party products to manage users and groups in Cloud Identity.

Administering user accounts and groups programmatically

User and group accounts in Cloud Identity can be managed via the Directory API. This REST API is part of the Admin SDK. The following are a couple of the provisioning use cases that you can implement using this API. In fact, the third-party apps that we discussed in the previous section use this API:

- Creating and managing users and adding administrators
- Creating and managing groups and group memberships

The API can be accessed from this page: `https://packt.link/w0lbi`.

Some of the operations that the API allows you to do are as follows:

- User lifecycle operations:
  - Creating a user
  - Updating a user
  - Deleting a user
  - Granting an administrator role
  - Searching users
  - Adding a relationship between two user accounts, for example, setting manager-employee relations
- Group lifecycle operations:
  - Creating a group
  - Updating a group
  - Adding a group alias
  - Searching groups
  - Searching all groups for a member
  - Deleting a group
- Group membership operations:
  - Adding a member to a group
  - Updating a group membership
  - Retrieving a group's members
  - Deleting membership

This concludes our section covering the Directory API.

Summary

In this chapter, we covered Google Cloud Identity. We looked at what services and features are available and how to design and build your authentication strategy on Google Cloud using Cloud Identity. The topics covered included domain setup, super administrator best practices, account security, how to enforce 2SV, how to configure user security settings, session management, how to configure SSO using SAML, how to use GCDS to federate AD with Cloud Identity, user and group provisioning, automated user lifecycle management, identity federation, and SSO.

In the next chapter, we will cover Google Cloud Identity and Access Management, looking at the authorization aspect of Google Cloud.

Further reading

For more information on Google Cloud Identity, refer to the following links:

- Difference between Cloud Identity Free and Premium: `https://packt.link/oKxkl`
- The Google security center: `https://packt.link/f5yss`
- Active Directory user account provisioning: `https://packt.link/pWIUp`
- Automated user provisioning and deprovisioning: `https://packt.link/Rsnr2`
- Active Directory single sign-on: `https://packt.link/4e85k`
- Best practices for planning accounts and organizations: `https://packt.link/j8eod`
- Best practices for federating Google Cloud with an external identity provider: `https://packt.link/24owJ`
- GCDS FAQ: `https://packt.link/EcGwY`
- Google Cloud Identity training: `https://packt.link/TABqf`
- Google Cloud Identity license information: `https://packt.link/j03kr`

6

Google Cloud Identity and Access Management

In this chapter, we will explore Google Cloud **Identity and Access Management** (**IAM**), an essential service to comprehend for the exam. With IAM, you can authorize cloud services, and assign appropriate access to users and applications. Acquiring a good understanding of IAM is crucial to ensure that your cloud implementation follows the principle of least privilege, restricting access to only what is necessary.

In this chapter, we will cover the following topics:

- Overview of IAM
- IAM roles and permissions
- Service accounts
- IAM policy bindings
- IAM conditions
- Cloud Storage, IAM, and ACLs
- Logging and IAM APIs

Overview of IAM

In the previous chapter, we discussed Cloud Identity and its role in authentication, user management, and device management in Google Cloud and Google Workspace. Now, let us explore IAM, which focuses on authorization in Google Cloud. Authorization is a key principle of cloud computing, addressing various requirements and ensuring secure access to resources. Some of the problems it solves are the following:

- How do I grant access to people and workloads?
- How do I provide time-bound access?

- How do I create service accounts with the least privilege?
- How do I enable services in a particular project but not others?
- How do I grant just the right access to users?
- How do I operate in multi-cloud environments?
- How do I find over-provisioned access?
- How do I troubleshoot access issues?

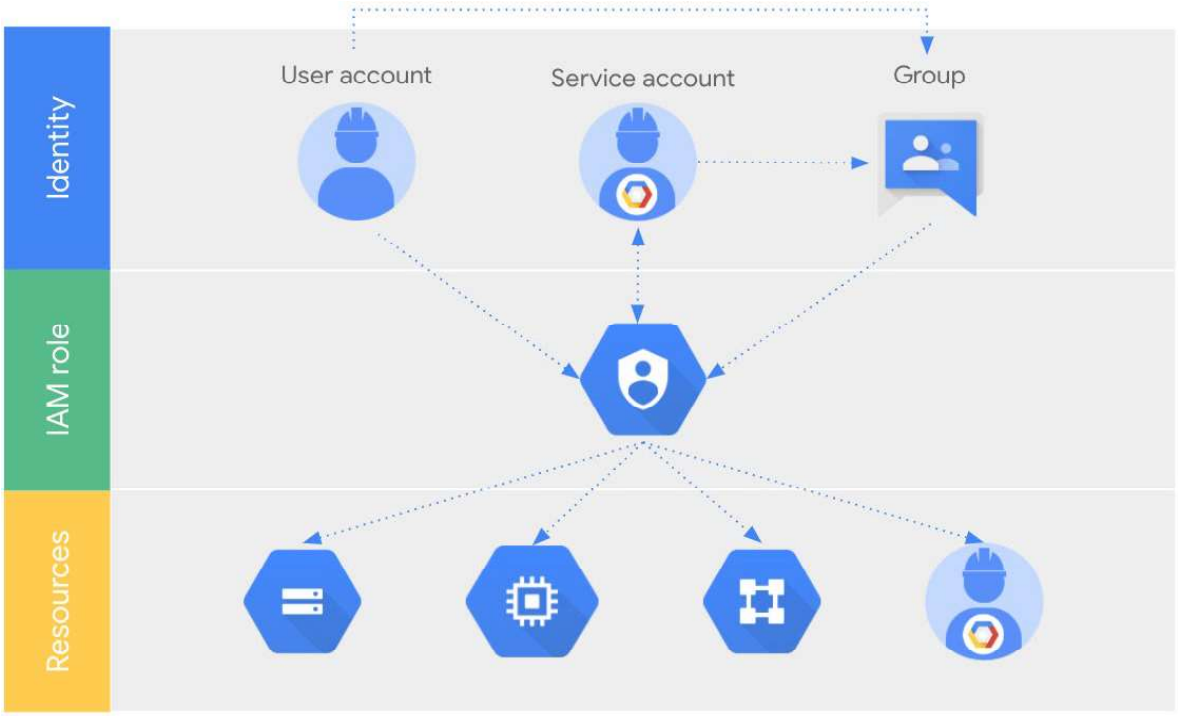

Figure 6.1 – IAM overview

In *Figure 6.1*, you can see how IAM works on the principle of *who* (identity) has *what* access (role) to *which* resource. The principal entity is usually a user, group, or service account. However, there are other principals, as explained later. The role is a set of fine-grained permissions. Permissions are predefined actions you can take on a particular resource (for example, BigQuery). Permissions for a given resource are fixed and cannot be created. We will talk about service accounts later in this chapter. At this point, the thing to remember is a service account represents a service or application identity in Google Cloud. It is one of the most important distinctions in how Google Cloud handles permissions for workloads.

In Cloud IAM, the following principals are supported:

- **Google account (a Gmail account is a form of Google account)**: Individuals who use a personal Google cloud have a Google account. This could be a developer, an administrator, or anyone else who works with the service. A person can be identified by any email address that is linked to a Google account. This can be `gmail.com` or any other domain.
- **Service account**: A service account is an identity for an application or VM instance instead of a person. Service accounts are principals as well as resources. We will see more on this later.
- **Google group**: A Google group is a group of Google accounts and service accounts. There is a unique email address for each Google group. Google groups provide a convenient way to apply IAM policies to a group of individuals. Instead of individually managing access controls for each person, you can apply or modify access controls for an entire group simultaneously. Additionally, you can easily add or remove members from a Google group, eliminating the need to modify IAM policies for user additions or removals.
- **Google Workspace account**: A Google Workspace account is a Google account linked to your organization-specific Google Workspace. It has your business email address. Google Workspace contains Google apps such as Docs, Slides, and Sheets in addition to a host of other apps that you may have licensed.
- **Cloud Identity domain**: Cloud Identity domains are like a Google Workspace account but without any of the Google apps. If your organization does not use Google Workspace, then you typically get a free edition of Cloud Identity with Google Cloud. Note that there are features that only a premium edition of Cloud Identity can offer. Refer to Google support documentation for a comparison between free and premium editions.
- **All authenticated users**: `allAuthenticatedUsers` is a unique identifier for all service accounts and all people on the internet who have signed in with their Google account, that is, *any authenticated Google Identity*. People who do not have a Google Workspace account or Cloud Identity domain can still use this identifier. For example, they can use their personal Gmail accounts. Users who have not been authenticated, such as anonymous visitors, are not on the list. Some types of resources do not work with this type of principal.
- **All users**: The default value is `allUsers`, which is a unique identifier that represents everyone who is on the internet, whether they are authenticated or not. Some types of resources do not work with this type of principal.

Now that you understand the types of principals, let us look at how they get access to cloud resources.

IAM roles and permissions

The *role* is a grouping of permissions in Google Cloud. Permissions cannot be granted to principals directly.

There are three types of roles possible within IAM:

- Basic roles or legacy roles
- Predefined roles
- Custom roles

Basic roles

There are three basic roles that exist in IAM. These are considered legacy now and were introduced prior to IAM:

- **Viewer**: Permissions for read-only actions that do not affect the state, such as viewing (but not modifying) existing resources or data
- **Editor**: All Viewer permissions, plus permissions for actions that modify state, such as changing existing resources
- **Owner**: All Editor permissions and the ability to manage roles and permissions for a project and all resources within the project

It is important to note that the basic roles are meant to make you productive in no time but are not recommended for production usage. In particular, the Editor and Owner roles grant a lot of permissions under the cover and will create over-permissive access in your Google Cloud environment. Enterprises usually should figure out a custom owner role that has just the right permissions to do the job and is not overly restrictive.

Predefined roles

IAM provides several service-specific predefined roles that are created by Google Cloud for easy assignment. These roles are usually product- or service-level roles based on a predefined set of permissions, for example, BigQuery Data Viewer. These roles are maintained by Google. In other words, any changes in permissions (addition/deprecation) are taken care of by Google, hence it is not recommended to create custom roles for services.

At any level of the resource hierarchy, you can grant the same user various roles. For instance, the same user can have the roles of Compute Network Admin and Logs Viewer on a project and simultaneously also have the role of BigQuery Editor on a particular BigQuery dataset within that project.

Now that you understand what predefined roles are, let us look at how to choose a predefined role for your workload. We will take the example of BigQuery roles.

Choosing predefined roles

There are a lot of predefined roles in IAM. Usually, these roles are tailored for a given cloud service. Let us take the example of BigQuery. Here are the predefined roles supplied by Google for BigQuery:

- BigQuery Admin
- BigQuery Connection Admin
- BigQuery Connection User
- BigQuery Data Editor
- BigQuery Data Owner
- BigQuery Data Viewer
- BigQuery Filtered Data Viewer
- BigQuery Job User
- BigQuery Metadata Viewer
- BigQuery Read Session User
- BigQuery Resource Admin
- BigQuery Resource Editor
- BigQuery Resource Viewer
- BigQuery User

How do you decide which predefined role to use? Consider the following steps:

1. Identify the necessary permissions.
2. Find the role/s that contain the permissions.
3. Choose the most appropriate roles *matching* the permissions. This could be one role or multiple roles making sure *least privileges* are granted.
4. Decide at which *level* of the resource hierarchy to grant the role.
5. Grant the role to a principal, and use conditions if needed.

It is also important to understand at what *level* (resource/project/folder/organization) the role should be granted to the principal. We make this point often since it is important to know the difference, as it impacts the ability of what the user can do.

Let us take an example of the *BigQuery User* role. Review the *permissions* it contains:

- `bigquery.bireservations.get`
- `bigquery.capacityComitments.get`

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

When you assign the *BigQuery User* role to a project or dataset, the following happens:

- This role allows the creation of a new dataset in a given project
- This role allows the principal to read the dataset's metadata and list the tables in that dataset
- This role allows the principal to run BigQuery tasks and SQL queries within the project
- This role also gives the principal the ability to list their own jobs and cancel the jobs they created
- This role also grants the BigQuery Data Owner role (`roles/bigquery.dataOwner`) to those who create new datasets within the project

Now that you understand how predefined roles work, let us look at some best practices for choosing a predefined role.

Best practices for predefined roles

Not every task requires a comprehensive list of permissions. Instead, prioritize the relevant permissions needed for the specific task. If a predefined role includes the required permissions, it is likely to include other permissions as well. Take this opportunity to identify the most impactful permissions needed. In general, fewer predefined roles provide more potent permissions.

The following types of IAM permissions are *considered* powerful:

- Resource creation and deletion permissions
- Permissions to gain access to confidential information, such as encryption keys or **Personal Identifiable Information** (**PII**)
- Permissions to configure a resource's IAM policy
- The ability to make changes to organizational structures, folders, and projects
- Permission to assign IAM policies

Compare these to the following permissions, which are less powerful:

- Resource listing permissions
- Access rights to non-sensitive data
- The ability to change settings that provide negligible risk, such as Compute Engine virtual machine instances' minimum CPU platform

Once you have discovered the required permissions, compile a list of roles that might be a good fit.

To assign permissions to a single Google Cloud service, you must identify as many specified roles as possible. For example, it is unlikely that a single specified role will contain the relevant permissions for both services if a principal needs to access Cloud Storage and administer Cloud SQL databases. If such a role exists, it may have unrelated permissions. Choosing a specific role that satisfies the requirements for Cloud Storage access and another one with Cloud SQL permissions can help reduce risk.

Let us look at some of the best practices for role assignments.

Assigning roles to resources

Select where you need to grant the predefined roles in the resource hierarchy. Here are a few points to consider:

- You can grant roles on lower-level resources (for example, a Compute Engine VM or a Cloud Storage bucket) if you just need access to them.
- To allow access to a project, folder, or organization's resources, grant the appropriate roles at that level. You can use IAM conditions to provide a role solely on a particular resource in the project, folder, or organization.

- You can use tags to define IAM policies for cloud services that support tags. Do not confuse labels and tags. Tags are an IAM construct and are defined at an organization level.

Let us take an example to demonstrate this.

If the principal needs access to a single Cloud Spanner database but several VM instances, you want to choose to grant the *Cloud Spanner admin* role on the database and the *Compute Admin* role on the project (in addition to the *Service Account User* role since some compute instances are preconfigured to use a service account).

You may verify which resources the principal can access and troubleshoot access difficulties using the IAM Policy Analyzer and Policy Troubleshooter (part of the Policy Intelligence tools, which are beyond the scope of this book).

You may have granted fewer permissions than the principal requires if the principal is unable to complete their tasks despite having the intended privileges on the relevant resources. The next step is to add more rights to the list, then search for predefined roles that have those permissions and pick the best ones. In some cases, you will have no choice but to create a custom role.

So far, we have looked at what predefined roles are and the best practices surrounding their assignment. Now let us look at custom roles.

Custom roles

In some instances, the predefined roles do not solve the purpose, or you would want to scale down the permissions the principal might get to a more appropriate level based on your requirements. In that case, you can create custom roles. The IAM Role Recommender (covered as part of IAM Policy Intelligence) also provides recommendations to create custom roles based on usage patterns. Please note that some IAM permissions (labeled `TESTING` and `NOT_SUPPORTED`) are not supported in custom roles. In fact, always use permissions with the `SUPPORTED` label.

Consider the following points while creating custom roles:

- Permissions that are only available in folders or organizations cannot be included in a custom role created at the project level. Because a project cannot contain other projects, for example, the `resourcemanager.organizations.get` permission cannot be used in a custom role at the project level; as a result, the permission is only applicable at the folder or organization level.
- If you create a custom role for a certain project or organization, you can only use that role there. Individuals in other projects and organizations, and resources within them, cannot be granted that role.

You will be able to manage all your organization's custom roles when you have the *Role Administrator* role.

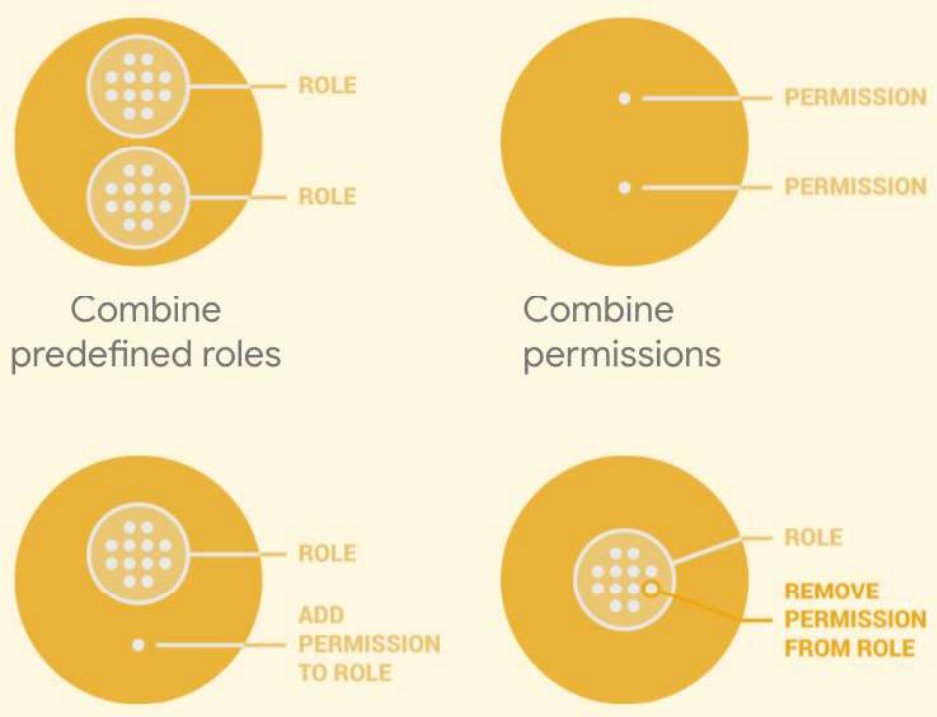

Figure 6.2 – IAM—custom roles

Figure 6.2 shows a few ways you can create custom roles:

- By combining predefined roles
- By combining permissions of predefined roles
- By combining a predefined role and adding additional permissions
- Starting with a predefined role but removing some permissions from that role

Lifecycle of a custom role

Consider the following permissions dependencies while creating custom roles:

- A predefined role or many predefined roles may exist for the service before you decide to establish your own custom role.
- `TESTING` and `NOT_SUPPORTED` permissions are ineligible for inclusion within a custom role.
- Even if a service's API is enabled, you may not be able to see or use certain permissions in a custom role. Consider checking out the Google Cloud documentation for custom role permissions.

- You can only use some permissions if you combine them with another related one. It is necessary to employ the *read-modify-write* permission while changing an IAM policy, for example. You will also need the `getIAMPolicy` permission for that service in addition to the `setIAMPolicy` permission to make changes to the IAM policy.

This concludes our discussion of roles in IAM. We will go over best practices for managing privileged roles and then move on to *service accounts* in the next section.

Best practices for managing privileged roles and separation of duties

Managing privileged roles and ensuring the separation of duties is crucial for maintaining a secure and compliant environment in Google Cloud. Here are some best practices for managing privileged roles and implementing separation of duties using IAM roles and permissions:

- **Principle of least privilege**: Adhere to the principle of least privilege when assigning roles and permissions. Only grant users and service accounts the minimum privileges necessary to perform their designated tasks. Avoid assigning excessive permissions that go beyond their job requirements.
- **Role-Based Access Control** (**RBAC**): Utilize Google Cloud IAM's RBAC model to define and assign roles based on job functions and responsibilities. Identify privileged roles that have elevated permissions and ensure they are only assigned to authorized individuals who require those privileges for their specific tasks.
- **Segregation of duties**: Implement segregation of duties to prevent conflicts of interest and reduce the risk of abuse or unauthorized actions. Separate critical functions and assign them to different individuals or teams. This ensures that no single person has complete control over sensitive operations.
- **Predefined roles**: Leverage predefined roles provided by Google Cloud IAM. These roles are designed with specific responsibilities and permissions, making it easier to assign appropriate privileges based on common use cases. Regularly review and update roles to align with changes in job functions and responsibilities.
- **Custom roles**: Create custom roles when predefined roles do not meet your specific requirements. This allows for fine-grained control over permissions and ensures that privileged roles are tailored to your organization's needs. Exercise caution when defining custom roles and regularly review and audit them.
- **Role hierarchy**: Establish a role hierarchy to manage privileged roles effectively. Consider creating a tiered structure where higher-level roles inherit permissions from lower-level roles. This ensures a logical and controlled flow of permissions and allows for easier management of access control.

- **Dual control and approval processes**: Implement dual control mechanisms and approval processes for critical actions or changes that require privileged access. This involves requiring multiple authorized individuals to review and approve such actions, reducing the risk of unauthorized or accidental actions.
- **Regular reviews and audits**: Conduct regular reviews and audits of privileged roles and their assigned permissions. Validate that the permissions assigned are still necessary and appropriate. Remove any unnecessary or outdated permissions and ensure that roles align with current job functions and responsibilities.
- **Monitoring and logging**: Implement robust monitoring and logging mechanisms to track privileged activities and detect any unusual or unauthorized actions. Enable the logging of IAM role changes, permission modifications, and administrative activities to establish an audit trail and facilitate incident response if needed.
- **Continuous Education and Awareness**: Provide ongoing education and training to privileged users, administrators, and stakeholders regarding the importance of role management, separation of duties, and best practices in access control. Foster a culture of security awareness and accountability within your organization.

By implementing these best practices, organizations can effectively manage privileged roles, enforce separation of duties, and maintain a secure and compliant environment in Google Cloud. Regular reviews, defined processes, and continuous monitoring help ensure that access privileges remain aligned with business requirements and minimize the risk of unauthorized access or malicious actions.

Policy binding

IAM policy binding is how you assign roles to principals for them to be able to act on resources. The binding can exist at the resource level (not all resources may support this), project level, folder level, or organization node level. It is important to understand at what level the binding should be created to achieve the least privileged access while fulfilling business goals.

The higher in the Google Cloud resource hierarchy the binding is created, the larger the effect it has on downstream resources such as folders, projects, and actual workloads. For this reason, be careful where you define your IAM policy binding. We will see more on IAM policy binding later, in the *IAM policy bindings* section.

Service accounts

Google Cloud service accounts are a critical part of the platform. A cloud resource or workload uses a specific type of account. To make API calls, an application deployed (for example, Compute Engine, App Engine, or GKE) or a workload (Dataproc, Dataflow, and so on) utilizes service accounts. Using a service account as an identity, the application can access cloud resources (either in the same or a different project) based on the role it has been assigned.

There are some main differences between a service account and a user account:

- Service accounts have no ability to log in to the Cloud console like a normal user.
- Public and private RSA key pairs are used to authenticate the service account and sign the API request.
- A service account can be impersonated by a human or another service account.
- Service accounts are not visible in Cloud Identity because they belong to the Google Cloud managed domain. While service accounts can be added to a Google group, it is not recommended since groups are meant for specific purposes that service accounts don't fit into.
- A service account is not created in your Google Workspace domain. They are not under the control of your Google Workspace and Cloud Identity administrators. They live in Google Cloud and should be managed there.

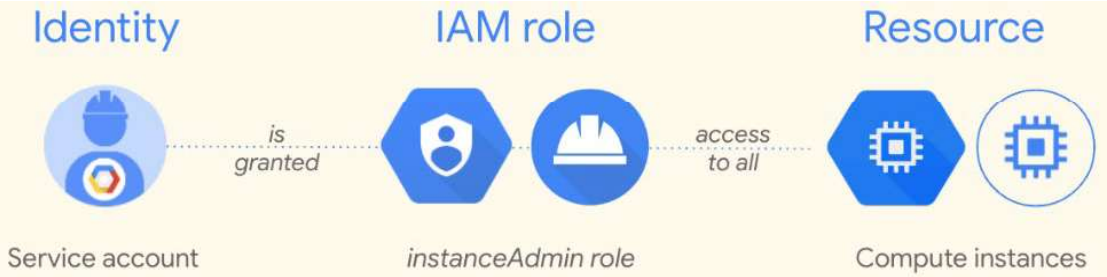

Figure 6.3 – Service account and resource

As depicted in *Figure 6.3*, service accounts get access to perform actions on a Google Cloud resource via an IAM role, just like users. Now we will see how to manage the lifecycle of service accounts.

Creating a service account

Service accounts can be created under the **IAM | Service accounts** menu in the Cloud console. Service accounts can also be created using gcloud, REST APIs, or client libraries.

To create a service account, the user should have the *Service Account Admin* role (`roles/iam.serviceAccountAdmin`).

It is recommended to wait for a minimum of 60 seconds before you use the service account post creation.

Disabling a service account

You can disable a service account either by using the Cloud console, *gcloud*, or APIs. However, note that if a resource is using a service account, after disabling it the resource can no longer access Google Cloud resources.

Deleting a service account

If a service account is deleted, its role bindings are not instantly eliminated; they are automatically purged from the system within 60 days. Do not include deleted service accounts in your quota. It is best practice to disable a service account before deleting it to make sure your production workloads are not negatively impacted.

Undeleting a service account

You can use the `undelete gcloud` command to undelete a deleted service account in some cases if it meets the following criteria:

- The service account was deleted less than 30 days ago
- There is no existing service account with the same name as the deleted service account

You will need the numeric ID of the service account to undelete it.

Now that we have looked at what service accounts are, let us look at how to use them. To be able to use a service account, you need a service account key or workload identity federation.

Service account keys

When you create a service account key, you are creating a private-public key pair. The public key is stored within Google Cloud, and you get a private part of that key pair. This is how you establish the identity of the service account. The key is used to authenticate that service account to Google Cloud from outside of Google Cloud. Within Google Cloud, you do not need keys (except for GKE). These keys need to be stored securely as leakage of the key in the wrong hands could cause unauthorized access to your Google Cloud environment. Crypto mining is one of the biggest misuses of such leaked keys.

Google Cloud, however, has introduced a more secure access method called **Workload Identity Federation** (**WIF**). WIF lets your workloads access resources directly, using a short-lived access token, and eliminates the maintenance and security burden associated with service account keys. WIF may be suitable in some scenarios such as multi-cloud authentication (for example, calling Google Cloud APIs from AWS or Azure) or, in some cases, hybrid cloud workloads. You will see more details on this in the *Configuring Workload Identity Federation using Okta* section.

Here is a decision tree that you can use to determine when you need to generate service account keys.

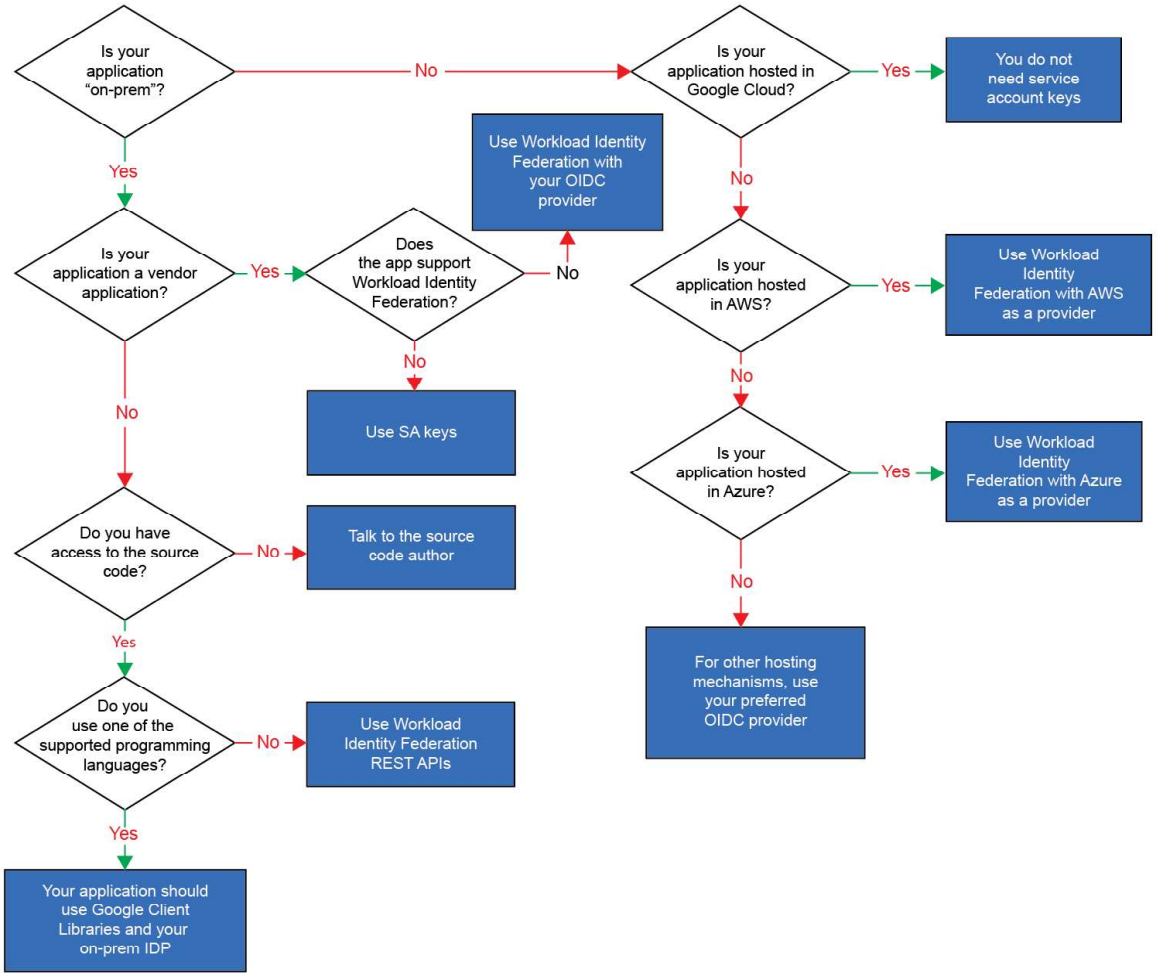

Figure 6.4 – Decision tree for service account keys

As you can see in *Figure 6.4*, you can select the right way to connect to Google Cloud APIs from hybrid workloads in a secure way. We recommend incorporating this decision tree in your organization process for development teams to refer to. Now, let us look at how to manage the lifecycle of service account keys.

Creating a service account key

Service account keys can be generated through the Cloud console, gcloud, using the APIs, or one of the client libraries. The key is either in the format of JSON or `PKCS#12` (P12 for backward compatibility).

> **Note**
>
> There is a hard limit of 10 keys per service account. If you generate more than 10, the oldest key becomes invalid.

Keys for service accounts do not automatically expire. Using **Resource Settings**, you may set the validity period for a service account key. However, the certification exam may not reflect this new feature.

The service account JSON key looks like this:

```
{
  "type": "service_account",
  "project_id": "project-id",
  "private_key_id": "key-id",
  "private_key": "-----BEGIN PRIVATE KEY-----\private-key\n-----END
PRIVATE KEY-----\n",
  "client_email": " prod-service-account@project-id.iam.
gserviceaccount.com ",
  "client_id": "client-id",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://accounts.google.com/o/oauth2/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/
v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/
metadata/x509/service-account-email"
}
```

In addition to creating a service account key, you can also upload the public part of your service account key to use with your workloads (thus bringing your own service account key to the cloud). Let us look at how to do this.

Bringing your own key to Google Cloud

Google Cloud allows you to bring your own private key. After creating a public-private key pair outside of Google Cloud, you will upload the public key portion of the key pair to the given service account.

For example, the following section generates a 2048-bit RSA key pair and wraps the public key in a self-signed certificate that is valid for 365 days. The following steps will also generate the service account JSON file for you:

1. Generate a private key:

```
PROJECT_ID=$(gcloud config get-value project)
SV_ACCT_EMAIL=sa-account-name@${PROJECT_ID}.iam.gserviceaccount.
com
SV_ACCT_ID=$(gcloud iam service-accounts \
        describe $SV_ACCT_EMAIL \
```

```
    --format 'value(uniqueId)')
PRIVATE_KEY=$(openssl genrsa 2048)
```

2. Create a self-signed certificate:

```
openssl req -x509 -new  \
  -key - \
  -subj /CN=customer@acme.com \
  -out csr.pem <<< $PRIVATE_KEY

openssl x509 \
   -in csr.pem \
   -signkey - \
   -days 365 \
   -out certificate.pem <<< $PRIVATE_KEY
```

3. Upload the public part of the key to Google Cloud:

```
gcloud iam service-accounts keys \
   upload certificate.pem \
   --iam-account $SV_ACCT_EMAIL \
   --format json > uploaded.json
PRIVATE_KEY_ID=$(jq -r .name uploaded.json | \
   awk -F/ '{print $NF}')
```

4. Generate the service account key file using the private part of the key. Look at the following code snippet:

```
touch look-no-keys.json
chmod 0600 look-no-keys.json
jq -n \
   --arg PRIVATE_KEY "$PRIVATE_KEY" \
   --arg PROJECT_ID $PROJECT_ID \
   --arg SV_ACCT_EMAIL $SV_ACCT_EMAIL \
   --arg SV_ACCT_ID $SV_ACCT_ID \
   --arg PRIVATE_KEY_ID $PRIVATE_KEY_ID \
  '{
  "type": "service_account",
  "project_id": $PROJECT_ID,
  "private_key_id": $PRIVATE_KEY_ID,
  "private_key": $PRIVATE_KEY,
  "client_email": $SV_ACCT_EMAIL,
  "client_id": $SV_ACCT_ID,
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
```

```
  "auth_provider_x509_cert_url": "https://www.googleapis.com/
oauth2/v1/certs",
  "client_x509_cert_url": @uri "https://www.googleapis.com/
robot/v1/metadata/x509/\($SV_ACCT_EMAIL)"
}' > sa-key.json
```

This should produce the `sa-key.json` key file, which you can use in your code.

Now that we have seen how to generate a service account key, let us look at some of the best practices associated with service account keys.

Best practices when managing service account keys

As a Google Cloud security practitioner, it is critical that you securely manage the key that grants access to your service account. Here are a few things to keep in mind:

- Determine whether you truly need to use service account keys or whether your application could use a more secure method such as WIF.
- The service accounts created for development purposes shouldn't have access to production resources.
- Have a procedure to rotate your service account keys. Rotate them often—daily, if possible—in development environments.
- Developers should download a new key every day, and enable this by developing a daily key rotation mechanism.
- You can develop an audit process by monitoring the usage of the service account keys. The `serviceAccount.keys.list()` API method will allow you to find all the keys associated with a given service account. You can match key IDs with audit logs. Any suspicious activity can be alerted for further investigation.
- Local development should always be done using the client libraries and Google Application Credentials.
- Make it impossible for developers to upload their private keys to external code repositories. There are several ways to achieve this, as we will see in the latter part of this chapter.
- Finally, perform periodic key scans on external repositories and correct any issues that are discovered. There are commercial products available on the market that will alert you to key leakage. Security Command Center Premium, a native Google Cloud offering, also has this capability. More on this in *Chapter 14*, *Security Command Center*.

Key rotation is one of the best practices we have mentioned. Now let us see how to do this.

Key rotation

Once you have conducted proper research and determined the necessity of using service account keys, it is crucial to prioritize regular rotation of the keys for development purposes. Implementing a key rotation process is essential to minimize the risk of any potential leaks or unauthorized access. The following outlines a recommended approach for effectively rotating service account keys:

1. Keyrotator is a simple CLI tool written in Python that you can use as is or as the basis for a service account rotation process. Run it as a cron job on an admin instance, say, at midnight, and write the new key to Cloud Storage for developers to download in the morning.
2. Create a dedicated project setup for shared resources.
3. Create a bucket in the dedicated project; do *NOT* make it publicly accessible.
4. Create a group for the developers who need to download the new daily key.
5. For additional security, you can have each developer use their own key. This will require you to provision a storage bucket for each developer in the project. This will also reduce the risk of non-repudiation.
6. Grant read access to the bucket using IAM by granting the `storage.objectViewer` role to your developer group for the project with the storage bucket.

A common issue in developing a process for key rotation is accidentally committing the key to the source repository. We will take an example of GitHub and show how to detect a key that has been committed to GitHub.

Preventing committing keys to external source code repositories

Your CI/CD pipeline should have the ability to detect a key being pushed to Git. If a key is found, the push should fail. A tool called `git secrets` can be used for this purpose. It will be invoked automatically when you run the `git commit` command.

You will need to configure `git secrets` to check for patterns that match the service account key file. For example, you can use the following command to detect private keys:

```
git secrets --add 'private_key'
git secrets --add 'private_key_id'
```

Now, when you try to run `git commit` and it detects these two keywords, you will receive an error message and be unable to do the commit unless you remove the key file.

Scanning external repositories for keys

There are various commercial products available that monitor service account key leakage in public source repositories. We will look at an open source tool called Trufflehog that you can use to scan your *private* Git repository.

Trufflehog is an open source tool you can use with `git secrets`. You can find it at `https://packt.link/uoyAx`. Trufflehog uses Shannon entropy to look through a repo's history for secrets. It does this by looking at the amount of entropy in the data.

Here is how you can use Trufflehog:

```
trufflehog git https://github.com/repot/trufflehog-testing
```

The output looks like this when it finds a service account key in the GitHub repo:

```
🐷🔑🐷  TruffleHog. Unearth your secrets. 🐷🔑🐷
Found verified result 🐷🔑
Detector Type: GCP
Decoder Type: PLAIN
Raw result: svc-acct-key-test@test-project-1231232.iam.
gserviceaccount.com
Repository: https://github.com/repo
Timestamp: 2022-11-04 09:41:04 -0700 PDT
Line: 6
Commit: 147d9d9e85555965f28f80cd45d1567c9947b98e
File: sa-key.json
Email: jdoe <jdoe@acme.com>

Found unverified result 🐷🔑❓

Detector Type: PrivateKey

Decoder Type: PLAIN
```

As you can see, Trufflehog has found a private key in the Git repository. It also shows the email of the committer that committed the key.

So far, we have seen ways of managing security around service account keys. We will move on to understand a few more features of service accounts, such as how to create short-lived credentials.

Service account impersonation

One of the key features of IAM is that you can impersonate a service account. Impersonation is the ability to let other users and resources *act like* the service account. We will look at that later in the section:

- **Service Account User**: This role has the `iam.serviceAccounts.actAs` permission, which allows principals to access all the resources that the service account can access through their own account. Suppose the principal is a user and has the *Service Account User* role. The principal can then impersonate the service account to set up Cloud SQL on that service account's project.

  This role also lets people link a service account to a resource. However, this will not allow the principal to create a short-term OAuth token for the service accounts or use the Google Cloud CLI to impersonate service accounts. If you would like that ability, then you need to have the *Service Account Token Creator* role.

- **Service Account Token Creator**: This allows principals to impersonate service accounts to create OAuth 2.0 access tokens, which you can use to authenticate with Google APIs. It also allows principals to sign **JSON Web Tokens** (**JWTs**) and sign binary blobs so that they can be used for authentication.

  To impersonate service accounts from outside of Google Cloud or from GKE workloads, use the Workload Identity User role.

  There are a couple of considerations when granting impersonation capabilities to principals:

  - Be extra careful when granting the role on a project, folder, or organization. This will allow a principal to act on any service accounts created within that project, folder, or organization. This is *not* a customary practice.
  - Granting the role on a specific service account will allow a principal to act in the place of another user on that account. This is a customary practice.

Now let us see how to enable service account access across projects.

Cross-project service account access

When managing many service accounts in an enterprise environment, it can be managed in a single project. That way, you can provide greater control over service account governance and the lifecycle process, and monitor the activity for any threat vector. This design pattern can be quite useful when you have a project implementing automation or monitoring tasks that span multiple projects and need to have access to them. This requires the ability of a service account from one project to be able to call resources from other projects.