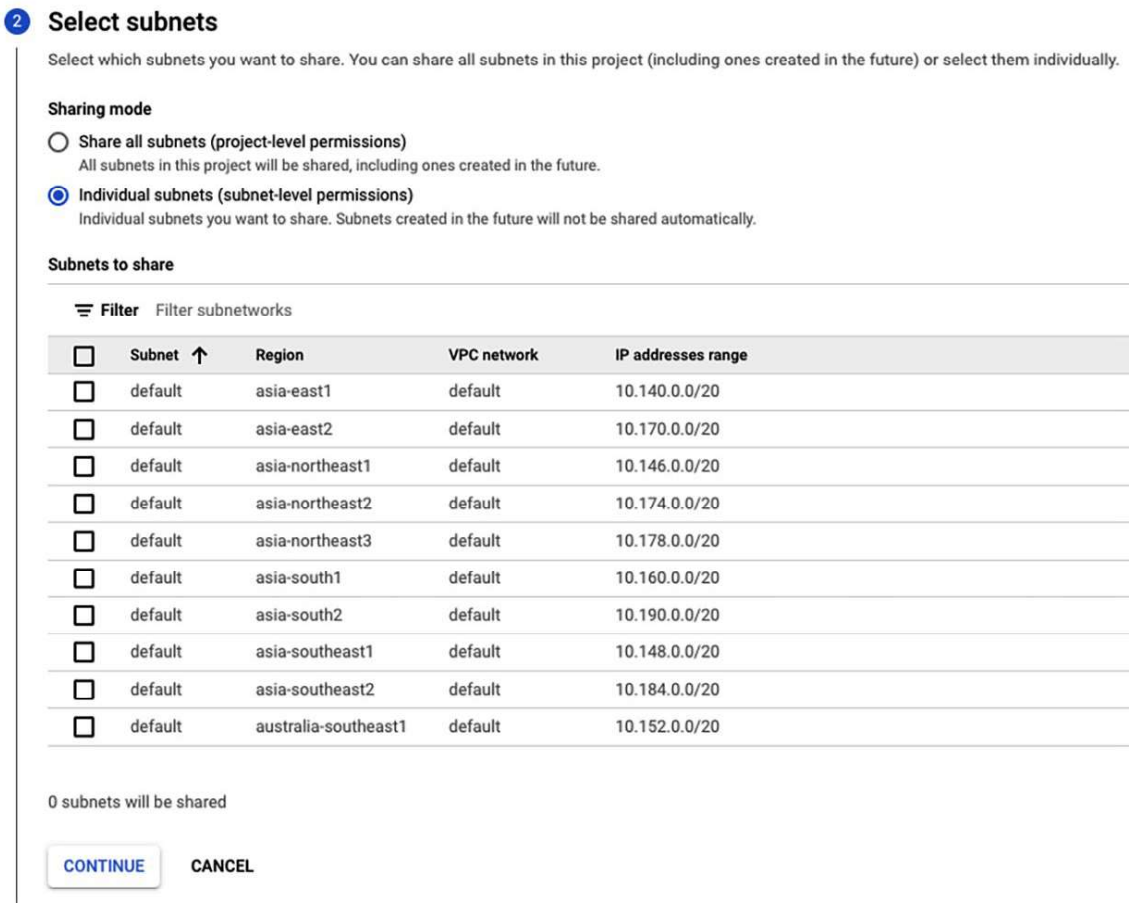

Figure 7.12 – Select subnets

7. Once done, you can then click **CONTINUE** to move to the next step to configure permissions.
8. In **Project names**, list the attached service projects. Using **Attach service projects** doesn't specify Service Project Admins; that's next.
9. Under the **Select users by role** section, you can add **Service Project Admins to Roles**. These users will have `compute.networkUser` for shared subnets. Only Service Project Admins can build Shared VPC subnet resources.
10. Once complete, you can hit the **Save** button.

Once the Shared VPC network is created, you can then attach projects using the following steps:

1. Click on the **Attached projects** tab; then, under this tab, click the **Attach projects** button.
2. When you click on **Attach projects**, you will navigate to a new page from where you can specify **Project names**. Here, check the service projects attached. Attaching service projects does not specify Service Project Admins; that is done in the next step.

3. Under **VPC network permissions**, select the role as `compute.networkUser`. Depending on the VPC network sharing mode, IAM principals are granted the Network User role for the entire host project or selected subnets. These principals are Service Project Admins.
4. In **VPC network sharing mode**, you have two options; first, there is **Share all subnets (project-level permissions)**. Selecting this option will share all current and future subnets in the VPC networks of the host project with all service projects and Service Project Admins. Alternatively, if you select **Individual subnets (subnet-level permissions)**, this will selectively share subnets from the VPC networks of the host project with service projects and Service Project Admins. Once you have made your selection, you can move on to the next step.
5. You can save and complete the process of attaching projects.

The Shared VPC Admin can also designate service accounts as Service Project Admins. Service Project Admins have two types of service accounts:

- The user-managed service account, with the following format: `USER_ID@SERVICE_PROJECT_ID.iam.gserviceaccount.com`
- The Google API service account, with the following format: `SERVICE_PROJECT_NUMBER@cloudservices.gserviceaccount.com`

For both types of service accounts, you do have the option to either give access to all subnets or only to limited subnets. In the following example, we will look at giving service account access to all subnets.

The steps to create a user-managed service account or a Google API service account as a Service Project Admin are the same. They are shown in the following steps, where you need to specify a service account principal based on whether you need a user-managed service account or Google API service account:

1. First, we will log in to the Google Cloud console as a Shared VPC Admin and then navigate to the **Settings** page. Here we will change the project to the service project that contains the service account that needs to be defined as a Service Project Admin:
2. Next, copy the project ID of the service project. For clarity, we will refer to the service project ID as `SERVICE_PROJECT_ID`.
3. Next, ensure that you change the project to the Shared VPC host project. Then, navigate to the **IAM** page in the Google Cloud console.
4. Next, we will add a new service account by clicking **Add**.

5. Specify the following details:

    I. Add `SERVICE_ACCOUNT_NAME@SERVICE_PROJECT_ID.iam.gserviceaccount.com` to the **Principals** field, replacing `SERVICE_ACCOUNT_NAME` with the name of the service account.

    II. To make the Google API service account a Service Project Admin, add `SERVICE_PROJECT_NUMBER@cloudservices.gserviceaccount.com` to the **Members** field.

    III. Select **Compute Engine | Compute Network User** from the **Roles** menu.

    IV. Click **Add**.

This concludes the Shared VPC configuration. There are a few more aspects of Shared VPC that are more network-centric and not required as part of the Google Professional Cloud Security Engineer exam. In the following section, we will cover VPC peering and how you can enable the sharing of routing information by configuring peer networks in Google Cloud.

VPC peering

VPC peering allows you to exchange the internal IP addressing of one VPC network with another VPC network irrespective of whether the VPC network is in the same project or belongs to another organization. The two main use cases for VPC peering include allowing your applications in one VPC network to communicate with services in another VPC network without needing to traverse the internet and keeping traffic within the Google Cloud network. **Software as a Service** (**SaaS**) providers can also expose their services privately to their users on Google Cloud by configuring VPC peering.

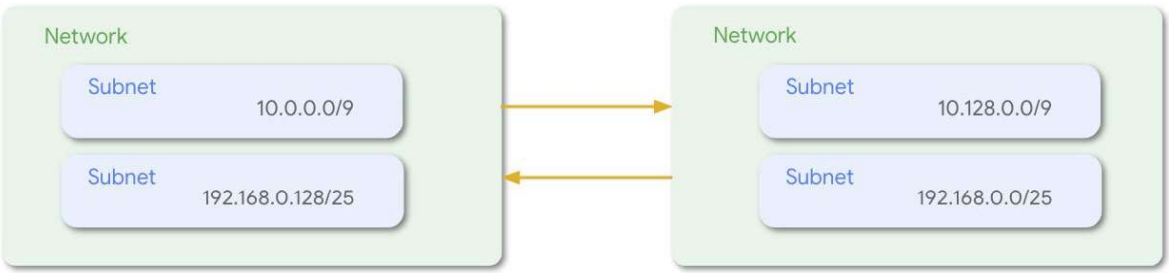

Figure 7.13 – VPC peering

Figure 7.13 shows two VPC networks that are peers and have subnet ranges in each of the networks that are exchanged once the peering is established.

Some of the key advantages of VPC peering are as follows:

- **Network latency**: As the traffic stays within the Google network and is local rather than traversing the internet, there is lower network latency.
- **Network security**: Application owners do not need to make their services publicly accessible, thereby reducing the attack surface.
- **Cost**: Egress traffic is charged, but when using VPC peering, your traffic stays within the Google network, hence no egress network charges will be applied, which helps in reducing the cost.

Next, we will look at some of the key properties of VPC peering:

- VPC peering supports Compute Engine, GKE, and GAE.
- Even though VPC networks are peered, they are administratively separate; that is, you can manage VPC attributes such as routing, firewall rules, and subnets per VPC and it has no impact on the peering.
- VPC peering requires setup on both sides. Administrators for the VPC networks have to complete the configuration on each side to complete the setup. We will look at this process in our VPC peering setup walk-through.
- Both static and dynamic routes can be exchanged.
- You can peer multiple VPCs together (limits apply).
- IAM permissions are required to have the ability to create and manage VPC peers.
- An organization policy administrator can apply the constraint to restrict VPC peering and define a set of VPC networks that can peer.

The following restrictions need to be taken into consideration before setting up VPC peering:

- IP addresses cannot overlap. Both VPC peers should have different IP address ranges.
- You cannot select which subnet routes can be shared; once peering is established, all routes are exchanged between the peering VPC networks.
- Dynamic routes can overlap with a peer network's subnet route, although overlapping destination ranges are silently dropped.

- Only direct peering is supported; you cannot have transit peers.
- Tags and service accounts cannot be used across the peering networks.
- Internal DNS names for compute engines are not accessible across the peering networks.
- Firewall rules are not exchanged across the peering networks, and you need to create rules to allow ingress traffic from compute instances. The default behavior is to block the ingress traffic to VMs by applying an implied deny ingress rule.
- Furthermore, you need to create firewall rules to restrict the access of your VMs to the services they shouldn't be able to access.
- If you are using an internal load balancer and need to restrict access, then you need to create ingress firewall rules that apply to the load balancer's backend VMs.
- Peering is permitted when establishing a peer with a Shared VPC network.

Next, we will look at how to configure a VPC peer:

1. Navigate to the **VPC Network Peering** page and click **Create connection**. Then, click **Continue**.

   Look at *Figure 7.14* for the fields that you need to complete in order to create a VPC network peer.

2. First, we will fill in **Name**; we will use the name `peer-from-starship`.
3. Next, we will select the network that is available under **Your VPC network**. Select a network you want to peer; in our example, we have selected `starshipnw`.
4. Now specify the network you want to peer with. This can be a network in your existing project, or it can be a network in a different project. If it is in another project, when you select that option, you must specify the project ID and the network name in that project. For our example and simplicity, we will select an existing project and then select the *default* network.

5. You also have the option to import or export custom routes. To do that, you can choose one or both of the following options. In our example, we will not import/export any custom routes:

   I. Use **Import custom routes** to import custom routes exported by the other network.

   II. Use **Export custom routes** to export custom routes to the other network. The other network must import the routes to see them.

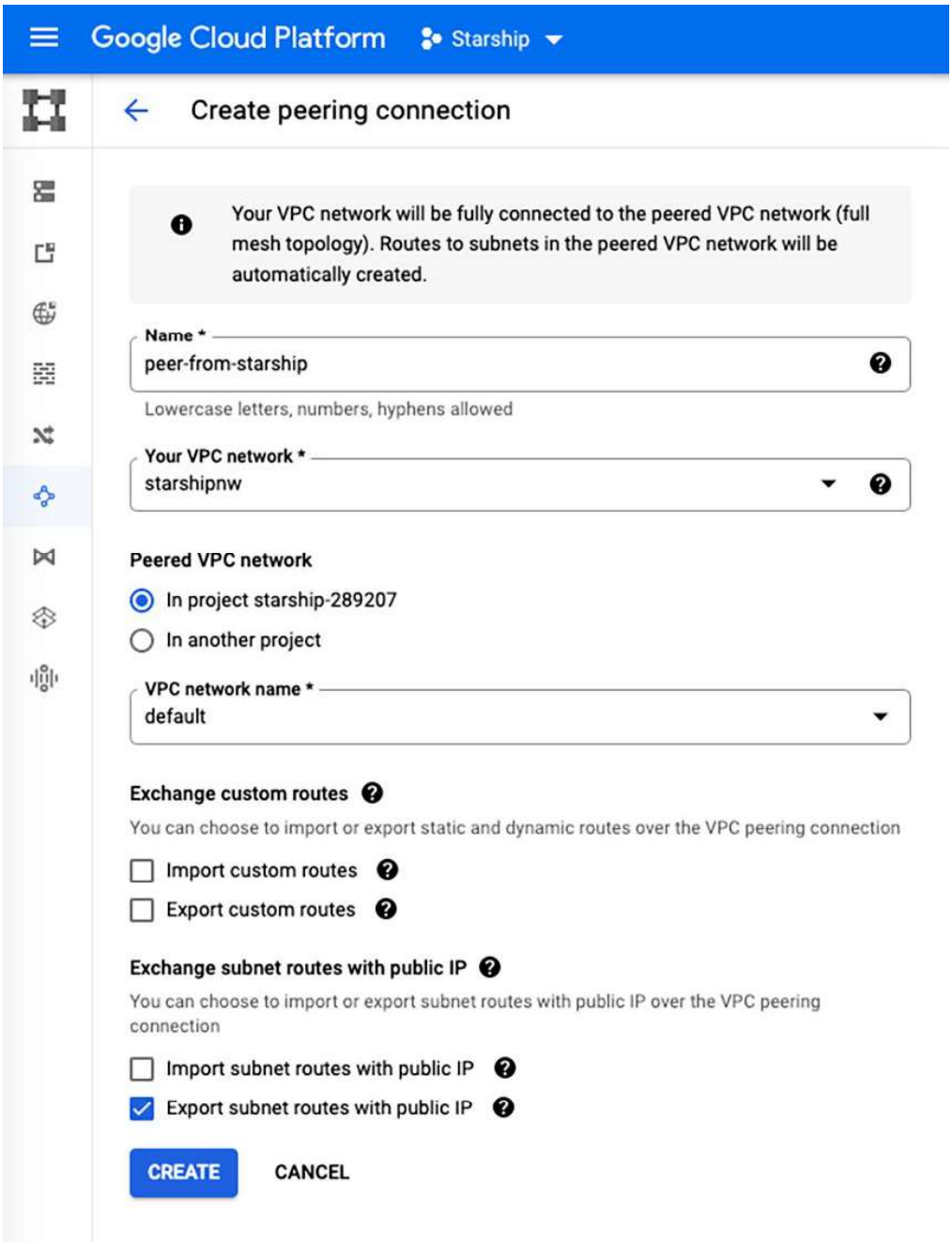

Figure 7.14 – Create peering connection

6. If your network or the partner network uses privately used public IP ranges, these routes are exported by default but not imported. Choose **Import subnet routes with public IP** to import privately used public IP subnet routes exported by the other network.
7. Click **Create**.

The peer configuration is only complete once you configure the peering on both sides. You can further modify the peers or delete them as required. The list of all VPC peers is found under the **VPC Peering** page on the Google Cloud console. Access to on-premises can be done using Cloud VPN. We will look at Cloud VPN in the *Hybrid connectivity options* section later in the chapter.

	Shared VPC	VPC Peering	Cloud VPN
Network services management (Firewalls, subnets, routes, VPN, DNS)	Central management of shared network resources	Clear network and security administrative boundaries	Clear network and security administrative boundaries
Transitivity	N/A	Non-transitive	Transitive
Scale	1000 service projects or more, depending on multiple factors	Up to 25 peered networks	Approximately 100 connected projects
Pricing	General network pricing	General network pricing	General network pricing. Excluding intra-zone traffic, which is billed as interzone.
Performance implication	None	None	Throughput limited based on number of tunnels (1.5 to 3 Gbps per tunnel)

Figure 7.15 – Cross-project communication options comparison

Figure 7.15 gives a quick snapshot of the different options available for cross-project communication across different dimensions, such as management, scale, pricing, and performance. You can use this to understand what options are available and, more importantly, which options align with your network architecture requirements.

We will now look at how micro-segmentation allows you to logically partition your network to have better control of traffic and security.

Micro-segmentation

In this section, we will look at some micro-segmentation techniques. We will cover topics such as how to create subnets, define custom routing, and use firewall rules that can help in creating segmentation in your network.

Subnets

Creating subnets for different types of workloads is a key micro-segmentation strategy. In this section, we will look at what types of subnets you can create and how to apply those subnets to your network design. Irrespective of what type of subnet you create, whether using auto mode or custom mode, on Google Cloud there are two types of **Classless Inter-Domain Routing** (**CIDR**) ranges: primary and secondary. Let us look at *Figure 7.16* to better understand the difference between the two and when to use one over the other.

	Primary	Secondary
Configuration	Mandatory – one per subnet	Optional – multiple ranges are supported
Used for	Allocation of VM primary IP reserved IPs	Allocating a different IP to multiple microservices running in a VM (e.g., containers, GKE pods).
Extendable	Range can be extended, but not shrunk	No

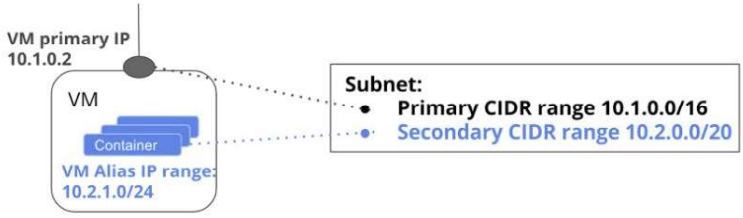

Figure 7.16 – Subnet CIDR ranges

The primary CIDR range is mandatory in a subnet; the secondary range is optional. The VMs, load balancers, and so on get IP addresses from the primary CIDR range. The secondary ranges are used for allocating IP addresses and microservices such as containers. You can extend the primary IP address range but cannot reduce the size of the IP range. Secondary IP address ranges cannot be extended and are fixed.

Custom routing

You can also use VPC routes to create custom routing to restrict and manage access to your services. Custom routing is a combination of both static and/or dynamic routing. In *Figure 7.15*, you can see different types of VPC routes that are supported, along with other attributes. Let us look at the different options available.

Route	Type	Created by	Next Hop	Restrictions	Exchanged with VPC Peering
Subnet route	System	System	VPC network	Cannot be removed	Automatic
Static route	Custom	User	Instance IP/name Cloud VPN	Must be broader than a subnet IP range	Flag controlled
Dynamic route	Custom	Cloud router (BGP session)	BGP peer	Must be broader than a subnet IP range	Flag controlled

Figure 7.17 – VPC routes

Google Cloud supports three different route types: system-generated routes, static custom routes, and dynamic custom routes. System-generated routes are created by the system and cannot be removed. These include routes added to allow communication between subnets, exchange routing information via VPC peering, and those exchanged using Cloud Router.

Static routes are added by the user manually and are also called custom routes. Using static routes, you can define a next hop, such as an internal TCP/UDP load balancer or Cloud VPN for connecting to on-premises services.

Dynamic routes are also classified as custom routes and are dynamically updated, such as creating a **Border Gateway Protocol** (**BGP**) peer via Cloud Router (we will cover Cloud Router in the *Hybrid connectivity options* section later in this chapter). The next hop is always a BGP peer, as this is configured point-to-point between two autonomous systems to dynamically exchange routing information.

You should not apply custom routes to instances as they will not persist once the instance reboots. You can, however, apply routes to all instances by using tags and service accounts.

Firewall rules

Firewall rules are one of the most important parts of network security. In this section, we will look at how firewall rules work, the evaluation criteria, the different types of firewall rules that you can create (ingress/egress), 5-tuple rules, and tag- service-account-based firewall rules.

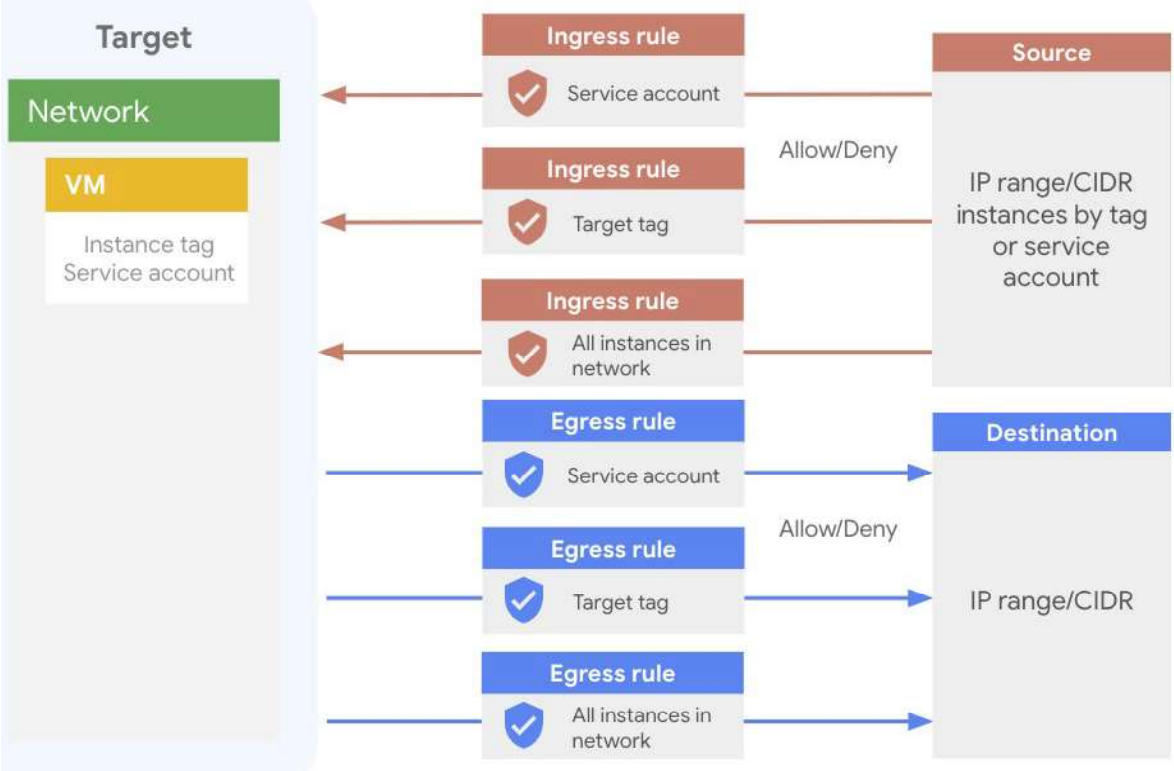

Figure 7.18 – VPC firewall rules

Let us look at some of the firewall rule features shown in *Figure 7.18*:

- Firewall rules are stateful in nature and are applied to a network inside a project. This means, for example, that if you create an ingress deny on port `22`, you don't need to configure an equivalent egress deny rule.
- Enforcement is done on the host level, which means there is a negligible performance penalty.
- The control paths that you can manage using firewall rules are VM to VM, VM to the internet, and VM to on-premises.
- For every VPC, two implied rules for ingress/egress are created, which cannot be deleted: a rule to block all incoming connections and another rule to allow all outgoing connections.

- There are some other exceptions as well; egress traffic on port `25` **Simple Mail Transfer Protocol** (**SMTP**) is always denied.
- All communication between a VM and its corresponding metadata server (`169.254.169.254`) is allowed.
- Firewall rules support both IPv4 and IPv6 addresses, but a single firewall rule can only have either IPv4 or IPv6.
- Each firewall rule has an allow or deny associated with it, and you have the option to disable firewall rules.
- ICMP response traffic is allowed through the firewall rules.
- All firewall rules are tracked irrespective of the protocol used.

This concludes our coverage of the firewall features; it's good to remember these from an exam perspective. Next, we move on to look at the different components of firewall rules.

Components of firewall rules

There are a few key components of firewall rules that we need to understand as it's important for configuration. Let us look at what these components are:

- **Direction of the rule**: Ingress rules govern connections from specific sources to Google Cloud targets, while egress rules govern connections from targets to specific destinations.
- **Priority**: The rule that is used is based on a number of factors. Rules with lower priority that conflict are ignored and only the rules with the highest priority (the lowest priority number) and that match the traffic are used.
- **Action**: The action determines whether the rule allows or denies connections.
- **Enforcement status**: Firewall rules can be either enabled or disabled without you having to delete them.
- **Target**: The target specifies the instances to which a particular rule applies.
- **Source**: The source is a filter for ingress rules or a destination filter for egress rules.
- **Protocol**: The protocol might be TCP, UDP, or ICMP, and the destination port might be `22`, `80`, or `443`, for instance.
- **Logs**: This option logs connections that match the rule to Cloud Logging.

Firewall rule evaluation

Figure 7.19 illustrates how sets of firewall rules are processed. The figure does not include implied rules or default rules for a default network, as the processing logic only cares about processing the rules and is not concerned with how the rules were inserted into the stack of rules to process.

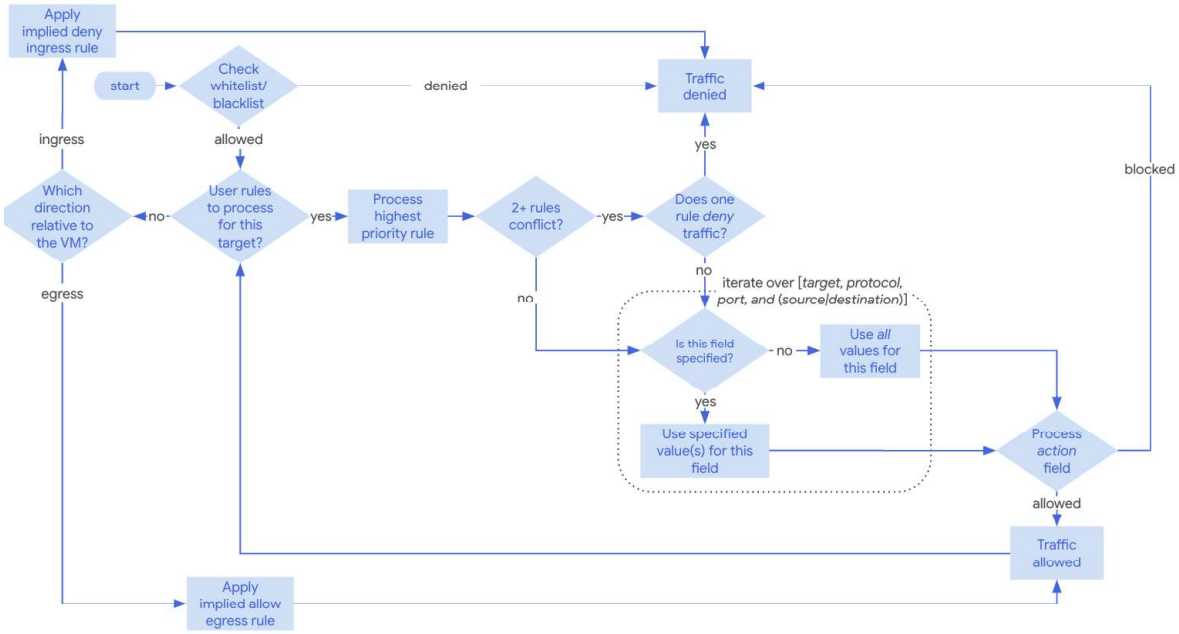

Figure 7.19 – Firewall rule evaluation logic

The evaluation logic is self-explanatory, so take some time to follow the rules and understand how the logic is applied. There are a few things to keep in mind: rule priority is rated from `0-65535`, where the lowest integer indicates the highest priority. *Target* refers to the instance(s) to which the rule applies; it is not the target of the network connection. *Port* refers to only TCP and UDP traffic.

Firewall rule options

There are three different options for how to configure your firewall rules:

- **5-tuple-based firewall rules**: You can create firewall rules based on the **5-tuple**, which specifies the source and destination IP, the port and protocol, and an associated action to allow or deny, along with the direction ingress/egress. You also have the option to either allow or deny all TCP/UDP ports or specific ports.

- **Network-tag-based firewall rule**: You can create a rule that specifies a network tag under **Targets**. A network tag is an arbitrary attribute. Any IAM principal with edit permission on an instance can associate one or more network tags with it. IAM principals assigned to a project that have the Compute Engine Instance Admin role have permission to edit an instance and change its network tags, which can change the set of applicable firewall rules for that instance.
- **Service-account-based firewall rule**: You also have the option to create firewall rules where you can specify the target based on the service account.

Let us look at how you can create firewall rules based on the three options:

1. Go to the **Firewall** page in the Google Cloud console and click on **Create firewall** rule. This will then display a screen that will let you create a new firewall rule.
2. Next, we will configure the attributes in the relevant fields, such as filling in **Name** for the firewall rule. In our example, in *Figure 7.20*, we have the name `new-rule-tags`.

> **Note**
> The firewall rule name must be unique for the project.

3. Optionally, you can also enable firewall rule logging by toggling the option in the setting to **Logs | On**. To remove metadata, you can further expand **Logs** details and then clear **Include metadata**.
4. In the next step, we will fill in **Network** for the firewall rule. For example, our network is `vpca`.
5. You can now specify **Priority** for the rule. As we discussed earlier, the lower the number, the higher the priority. Refer to the *Firewall rules evaluation* section covered earlier to understand how the logic works.
6. We will next specify **Direction of traffic**, where you can choose to select either **ingress** or **egress**.
7. For **Action on match**, choose **allow** or **deny**.
8. Next, we have the three options that you can specify as the targets of the rule:

    I. If you want the rule to apply to all instances in the network, choose **All instances in the network**.

II. If you want the rule to apply to select instances by network (target) tags, choose **Specified target tags**, then type the tags to which the rule should apply into the **Target tags** field.

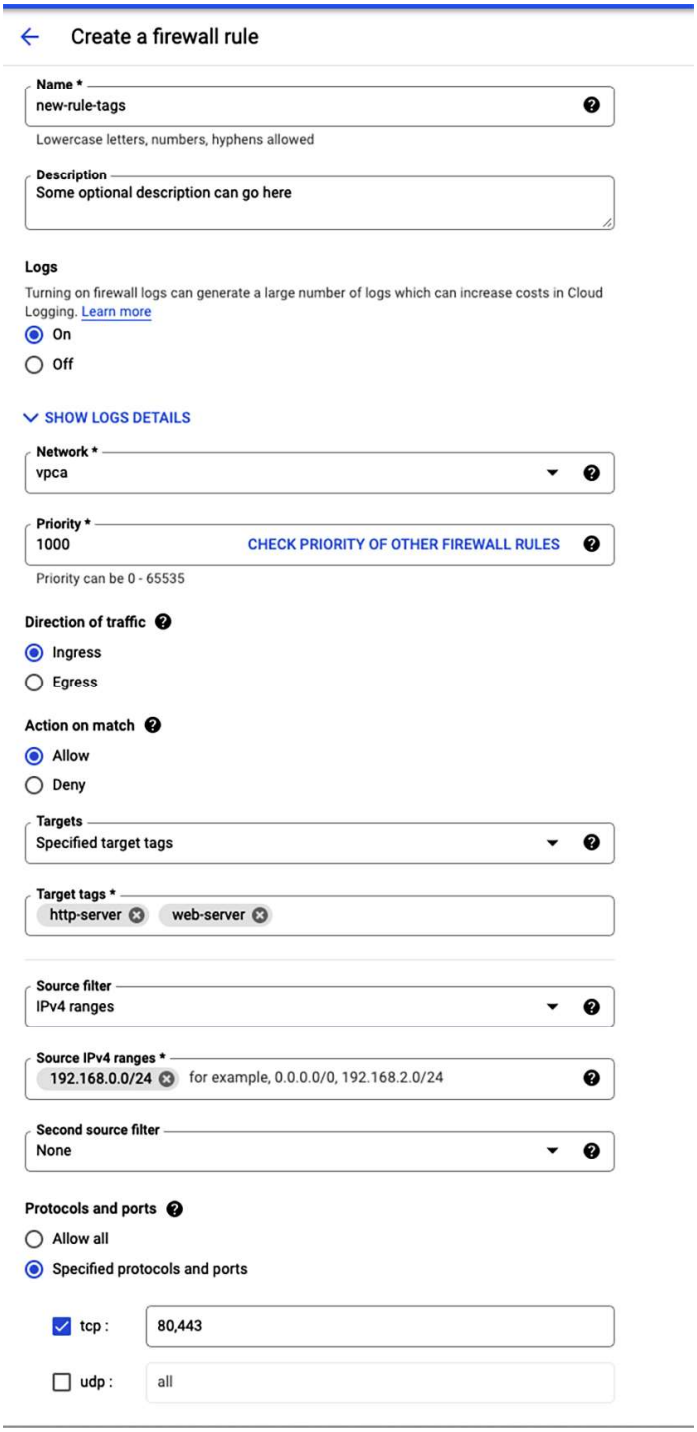

Figure 7.20 – Create a firewall rule using target tags

III. If you want the rule to apply to select instances according to the associated service account, choose **Specified service account**, indicate whether the service account is in the current project or another one under **Service account scope**, and choose or type the service account name in the **Target service account** field.

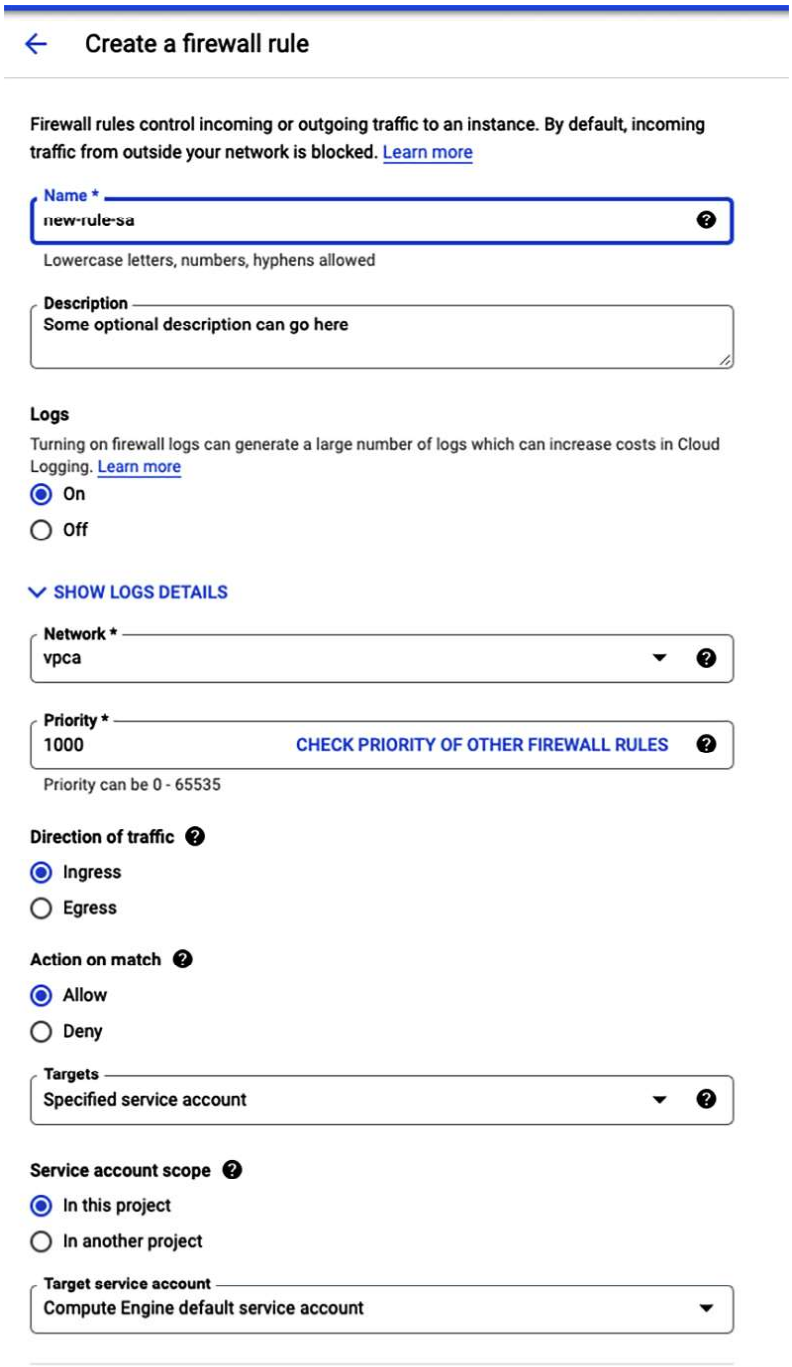

Figure 7.21 – Create a firewall rule using the service account

9. For an ingress rule, specify **Source filter**:
    I. Choose IP ranges and type CIDR blocks into the source IP ranges to establish the incoming traffic sources. You can specify any network source by using the default IP range, that is, `0.0.0.0/0`.
    II. To limit sources by network tag, from the options presented under **Source Filter**, select **Source tags** and enter the tags. Source tag filtering is only available if the target is not a service account.
    III. To limit sources by service account, choose **Service account**, indicate whether the service account is in the current project or another, then type the service account name in **Source service account**. Source service account filtering is only accessible if the target is not tagged.
    IV. Specify a second source filter. Secondary source filters cannot employ primary criteria. Source IP ranges work with source tags and source service accounts. An effective source set is the source range IP addresses plus the network tags or service accounts. The source is included in an effective source set if either the source IP range or source tags (or source service accounts) fulfill the filter requirements.

> **Note**
> Remember that **Source tags** and **Source service account** cannot be used together.

10. For an egress rule, specify **Destination filter**:
    I. Choose IP ranges and type **CIDR blocks** into **Destination IP ranges** to create outgoing traffic destinations. *Everywhere* is `0.0.0.0/0`.
11. Define the protocols and ports to which the rule applies:
    I. To apply the rule to all protocols and destination ports, select **Allow all** or **Deny all**.
    II. Define specific protocols and destination ports:
        i. Select **TCP** to include TCP ports. Enter destination ports such as `20-22`, `80`, and `8080`. Use commas as delimiters.
        ii. Select **UDP** to include UDP ports. Enter destination ports, such as `67-69` and `123`. Use commas as delimiters.
        iii. Select **Other protocols** to include protocols such as `icmp` or `sctp`.
12. To avoid enforcing the firewall rule, you can set the enforcement state to **Disabled**. Select **Disabled** from the **Disable** rule.
13. Click **Create**.

Firewall rules are referenced throughout the exam in network security questions, so do spend some hands-on time configuring firewall rules. There are additional links in the *Further reading* section that point to some labs that are available on Google Cloud Skills Boost to get more familiar with firewall rules and their implementation so you can better apply the concepts learned here.

Cloud DNS

In this section, we will look at Cloud DNS and how to configure it with some basics. For the exam, you only need basic knowledge of the Cloud DNS topics. We will look at an overview of Cloud DNS and some key components that you need to understand.

Cloud DNS is a highly scalable fully managed service offered by Google Cloud. You can create both public and private zones using Cloud DNS. Cloud DNS uses an internal metadata server that acts as the DNS resolver for both internal and external resolutions, such as resolving hostnames on the public internet. Every VM instance has a metadata server used for querying instance information – for example, name, ID, startup/shutdown scripts, custom metadata, service account, and so on – and the DNS resolver is set on VMs as part of default DHCP (Dynamic Host Configuration Protocol) leases. Overriding DHCP leases is possible by customizing the DHCP configuration; however, it is not a common pattern.

Cloud DNS also supports split-horizon, where VMs can be resolved within a private zone or public zone, depending on where the query comes from. For hybrid connectivity, you can use DNS forwarding zones, where you can forward requests to an on-premises resolver.

You can also create DNS peering, allowing you to query private zones in different VPC networks. Peering is one-way and no connectivity is required. It supports two levels of depth, meaning that transitive DNS peering across two DNS peering hops is supported. This is required in a hub-and-spoke model to allow spokes to query each other's private zones without creating a full-mesh peering topology.

With Cloud DNS, you can also create a managed reverse zone to prevent non-RFC 1918 address resolutions from being sent to the internet. **Domain Name System Security Extensions** (**DNSSEC**) is also supported, and we will cover DNSSEC in the next section. You can create resolver policies to configure access to private and restricted Google APIs from within a VPC-SC perimeter. We will cover VPC-SC in *Chapter 10*, *Cloud Data Loss Prevention*.

The internal DNS is where the records are automatically created for the VM's primary IP, such as `instance1.us-west1b.c.projectx.internal`. These are used for resolution within the same VPC and project.

Configuring Cloud DNS – create a public DNS zone for a domain name

Before you begin, please ensure that you have a valid domain name registered with the domain registrar. You will also need a Windows or Linux VM instance and the IP address to point to the A record of your zone. Next, check and enable the DNS API if it is not enabled. This can be done by navigating to **API & Services** from the menu in the Google Cloud console.

Take the following steps to configure and set up the DNS public zone:

1. In the Google Cloud console, go to the **Create a DNS zone** page.
2. For **Zone type**, select **Public**.
3. For **Zone name**, enter `my-new-zone`.
4. For **DNS name**, enter a DNS name suffix for the zone by using a domain name that you registered (for example, `example.com`).
5. For **DNSSEC**, ensure that the **Off** setting is selected.

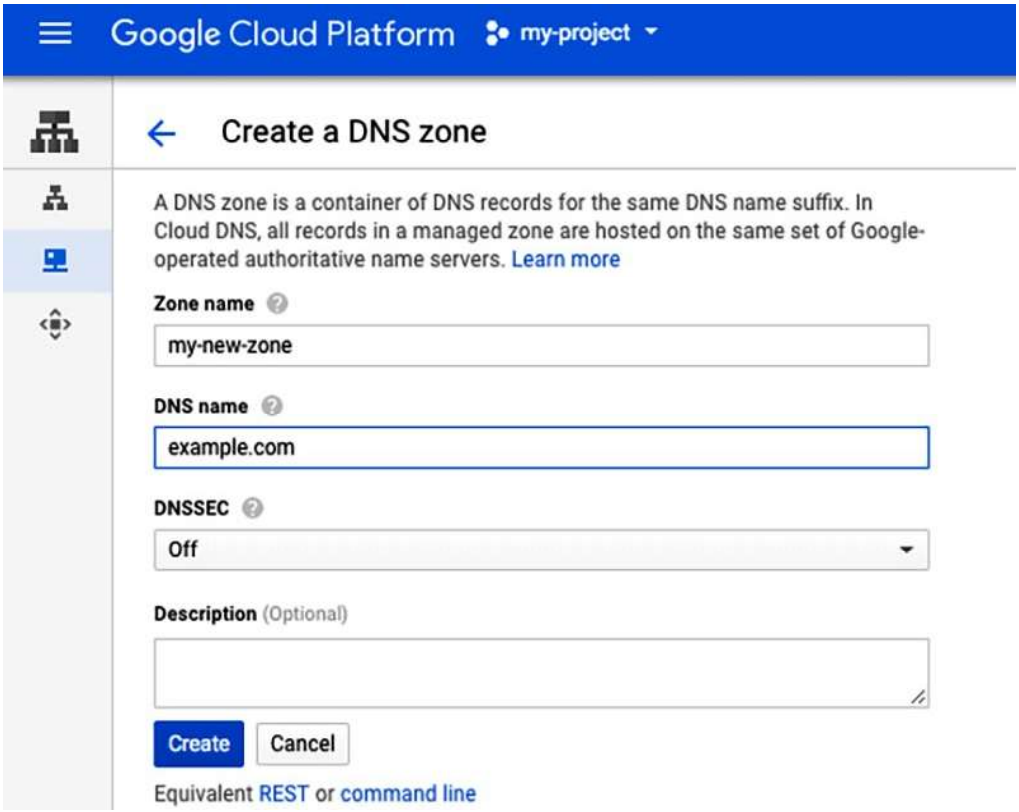

Figure 7.22 – Create a DNS zone

6. Click **Create**.

Next, we will look at the steps on how to create a record to point the domain to an external IP address:

1. In the Google Cloud console, go to the **Cloud DNS** page.
2. Click the zone where you want to add a record set.
3. Click **Add record set**.
4. For **Resource Record Type**, to create an A record, select **A**. To create an AAAA record, select **AAAA**.
5. For **IPv4 Address** or **IPv6 Address**, enter the IP address that you want to use with this domain.

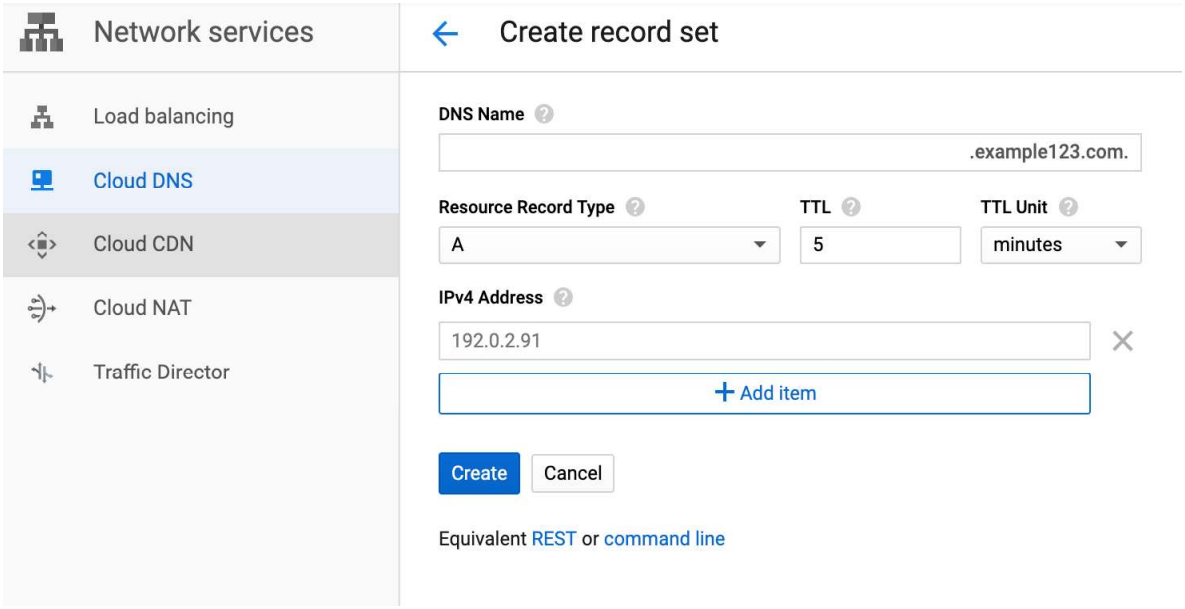

Figure 7.23 – Create a record set

6. Click **Create**.

In this last step, we will create a CNAME record for the WWW domain:

1. In the Google Cloud console, go to the **Cloud DNS** page.
2. Click the zone where you want to add a record set.
3. Click **Add record set**.
4. For **DNS Name**, enter www.
5. For **Resource Record Type**, select **CNAME**.

6. For **Canonical names**, enter the domain name, followed by a period (for example, `example.com.`).

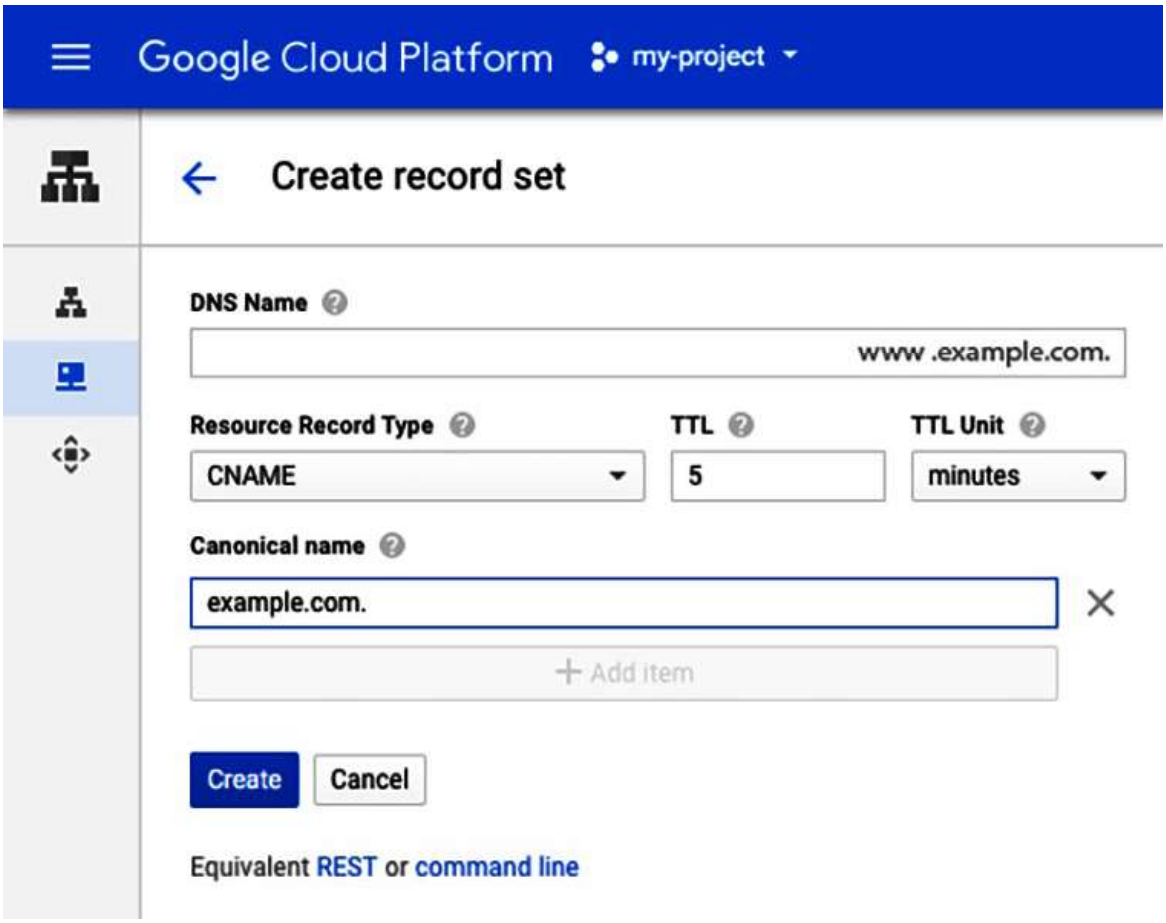

Figure 7.24 – Create a CNAME record

7. Click **Create**.

In the next section, we will take a look at DNSSEC. It is a set of protocols that are designed to increase the security of the DNS.

DNSSEC

DNSSEC is a security feature of DNS that helps prevent attackers from poisoning or manipulating DNS requests. The exam covers DNSSEC at an extremely basic level, so you should understand what DNSSEC is and how it works. From a Cloud DNS perspective, you can enable DNSSEC for your public zone by defining the parameters as shown in *Figure 7.25*.

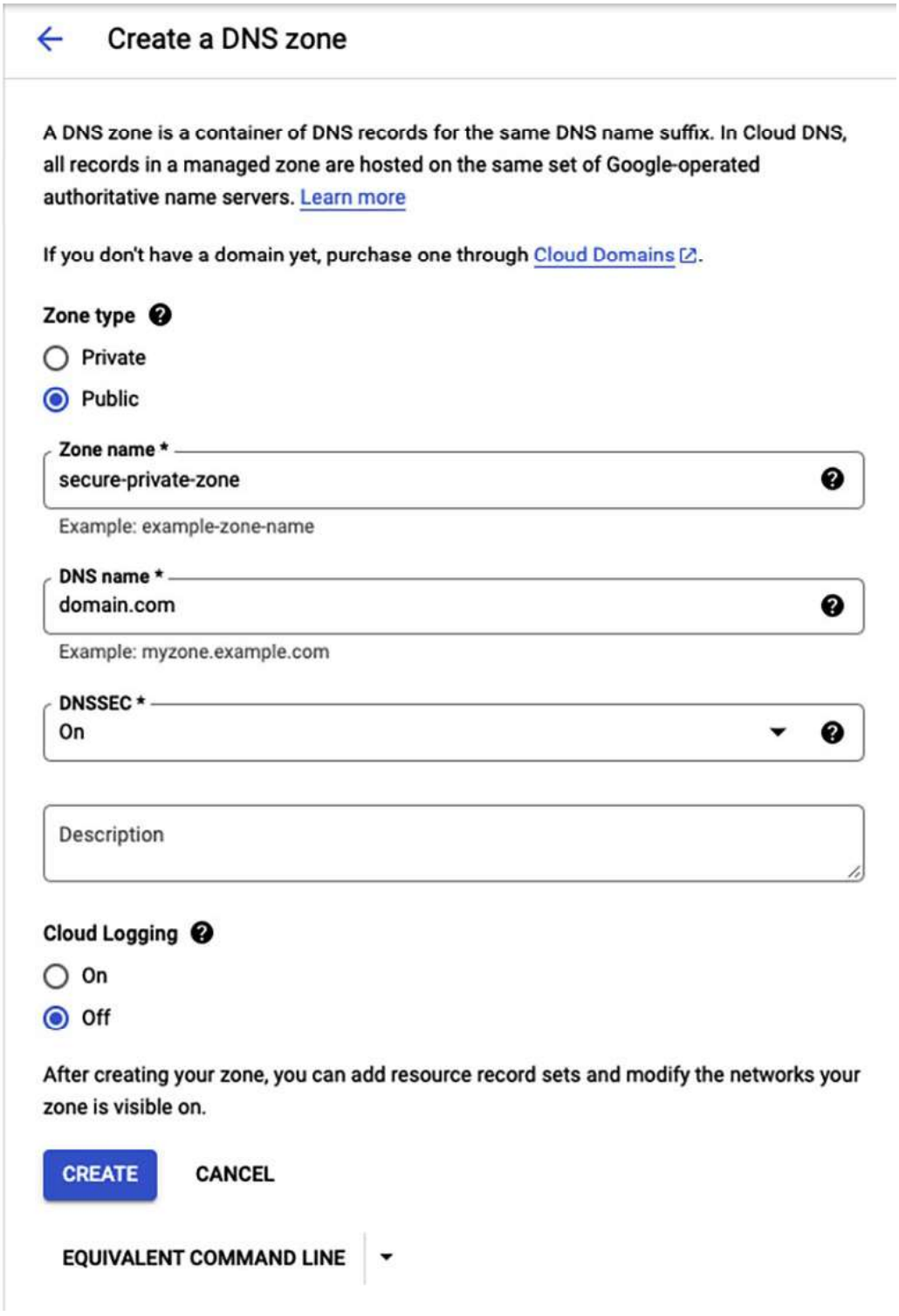

Figure 7.25 – Enable DNSSEC for a private DNS zone

In *Figure 7.25*, you first assign a name to your private zone in the **Zone name** field. Then, specify **DNS name**, which is the domain name, and then set the **DNSSEC** option to **On**. Once completed, click the **CREATE** button.

In the most basic form, DNSSEC works by including a digital signature in responses to DNS queries. So, for every DNS zone, public and private keys are created. Cloud DNS serves the public key (DSKEY), resource record signatures (RRSIG), and non-existence (NSEC) parameters to authenticate your zone's contents. All these are automatically managed by Cloud DNS once you have enabled DNSSEC.

An important thing to note is that for DNSSEC to work, both your registrar and registry should support DNSSEC for your top-level domain. This means that for DNSSEC to be effective, both the company that is responsible for registering the domain name (the registrar) and the company that is responsible for maintaining the database of domain names and their associated IP addresses (the registry) must support DNSSEC for the top-level domain (for example, `.com`, `.org`, and so on). If you are unable to add a **Delegation Signer** (**DS**) record through your domain registrar to enable DNSSEC, then enabling DNSSEC on Cloud DNS will have no effect. Another thing to be aware of is that DNSSEC is not encrypted; keys are used to authenticate the resolver, not to encrypt client-server queries and resolutions.

If you are migrating from your DNSSEC-signed zone to Cloud DNS, you can use the **Transfer** state when enabling DNSSEC for your public zone. The **Transfer** state lets you copy your relevant keys, such as the **key signing key** (**KSK**) and **zone signing key** (**ZSK**), from your old zone to Cloud DNS. You should keep the state as **Transfer** until all DNS changes have propagated to the authoritative DNS server and then change the state to **On**.

Load balancers

In this section, we will look at load balancing. There are many different load balancers that are available on Google Cloud. We will look at each of the different types of load balancers, what they are, and when to use which type of load balancer based on the traffic type. We will go over a decision tree that helps you decide what type of load balancer would meet your requirements, along with some limitations as well.

The other aspects that we will cover include the difference between external and internal load balancers and when to use which and why; regional versus global load balancers; and some considerations from a network tier perspective, as some load balancers are only supported by Premium Tier while others are also supported by Standard Tier. This can be an important thing to be aware of both from a cost and security perspective.

Let us start by looking at the different types of load balancers:

- **Global external HTTP(S) load balancer**: This load balancer handles both HTTP and HTTPS traffic and is a proxy-based Layer 7 function that caters to your services and is configured behind a single external IP address. It can service traffic to different types of Google Cloud services as backends, such as Google Compute Engine, Google GKE, and Cloud Storage. You can use an external HTTP(S) load balancer to distribute and manage traffic for your hybrid workloads in an on-premises environment as well. This load balancer is implemented on **Google Front End** (**GFE**) and is global when using Premium Tier but can be configured to be regional when using Standard Tier.
- **Regional external TCP/UDP load balancer**: This is also referred to as a regional network load balancer and is configured as a pass-through. It balances the traffic for VMs in the same region. This is offered as a managed service and can cater to any client on the internet, including Google Compute Engine, with an external IP address and clients using Cloud **Network Address Translation** (**NAT**). This load balancer is not a proxy, and it distributes traffic for the backend VMs using the packet's source and destination IP address and protocol. The traffic is terminated at the VM and responses from the backend are sent directly to the client and not through the load balancer.
- **SSL proxy load balancer (global)**: This is a reverse proxy load balancer that distributes SSL traffic from the internet to the closest available backend VMs. The SSL proxy load balancer terminates the incoming TLS requests and forwards the traffic to the backend VM as TLS traffic or TCP. As with the HTTP(S) load balancer, based on the network tier, you can configure the SSL proxy load balancer to be global by using Premium Tier or configure it to be regional if using Standard Tier. Both IPv4 and IPv6 are supported, and the traffic is intelligently routed to the backends based on capacity. Both self-managed and Google-managed certificates are supported. You can also control the SSL features on the load balancer by configuring the SSL policies.
- **TCP proxy load balancer (global)**: This is a reverse proxy load balancer, and it distributes the traffic to the VMs configured as your backends inside the Google Cloud VPC network. It uses a single external IP address for all users globally. All TCP traffic coming from the internet is terminated at the load balancer and can then be routed to the VMs as TCP or SSL. With this load balancer, you can have global load balancing if using Premium Tier or regional load balancing if using Standard Tier.
- **Internal TCP/UDP load balancer**: This load balancer distributes traffic to your internal VMs within the region in a VPC network. A typical use case is using internal load balancers in a three-tier web architecture, where they can be deployed with an external load balancer. In this architecture, you will have an external HTTP(S) load balancer in front of your web servers and then have internal load balancers behind the web servers to cater and scale traffic to your VMs, which are private and only accessible within the VPC network.

- **Internal HTTP(S) load balancer**: An internal HTTP(S) load balancer is a regional proxy-based Layer 7 load balancer. Using this load balancer, you can distribute traffic to backend services such as Compute Engine and GKE. The internal load balancer is built on the open source Envoy proxy.

	Type	Geographical scope	Network tiers	Proxy/pass-through
Internal	TCP/UDP	Regional	Premium	Pass-through
	HTTP(s)			Proxy
External	TCP/UDP	Regional	Standard/Premium	Pass-through
	HTTP(s)	Regional/Global depending on network tier	Standard/Premium	Proxy
	TCP proxy			
	SSL proxy			

Figure 7.26 – Different types of Google Cloud load balancers

Figure 7.26 gives a quick overview of the different types of load balancers, their geographical scope, and network tiers.

The topic of load balancers is very broad, with a lot of features and capabilities that are more network-centric than security-related. Therefore, for the purpose of the exam, the content here is limited to the security scope. Understanding how load balancers work and how to configure different types of load balancers is important; therefore, it is highly recommended that you spend time reading about load balancers in more detail and spend some time on hands-on exercises using Google Cloud Skills Boost. Links in the *Further reading* section will take you to Google Cloud Skills Boost network labs and documentation on understanding Google Cloud load balancing.

Now we will look at some important distinctions, such as external versus internal, and when to use what load balancer based on the use case or traffic type.

Google Cloud load balancers are of two types: external or internal. You will use an external load balancer if you have traffic coming from the internet to access services that are in your Google Cloud VPC network. An internal load balancer, on the other hand, is only used when you need to load balance traffic to VMs that are private and only accessible from within the VPC network.

Another aspect to understand is when to use global versus regional load balancers. A global load balancer will manage and distribute traffic coming in from the internet to backends that are distributed across multiple regions. Your users only interact with one application using the same unicast IP address, and a global load balancer can distribute traffic to the closest backend location. Premium Tier is a requirement for global load balancers. Regional load balancers, however, are used when you have a requirement such as regulatory requirements to keep your traffic limited to backends in a particular region. For this type of load balancer, you can use Standard Tier.

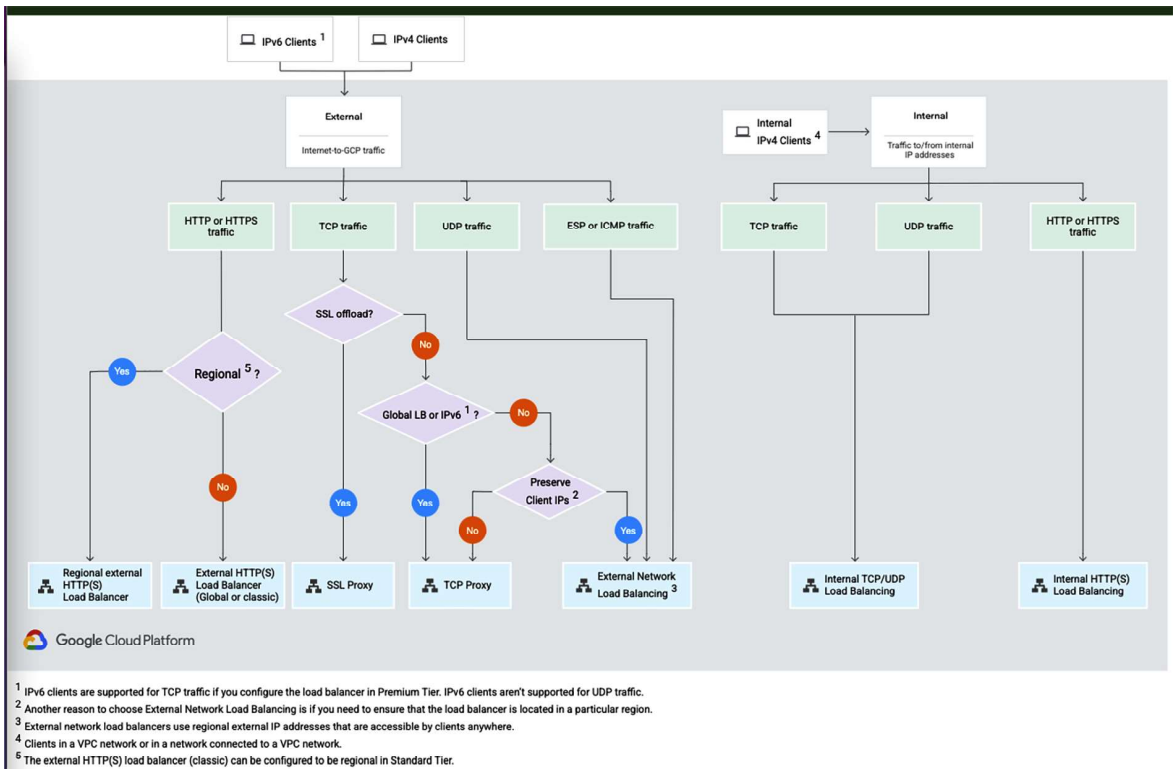

Figure 7.27 – Decision tree for choosing a load balancer

The next important thing to know is when to use which type of load balancer. You can use *Figure 7.27* as a decision tree to help you decide.

Configuring external global HTTP(S) load balancers

Let us look at the following architecture to better understand how the HTTP(S) load balancer accepts and distributes traffic. The following architecture (*Figure 7.28*) is only for illustrative purposes and is not relevant to the actual configuration.

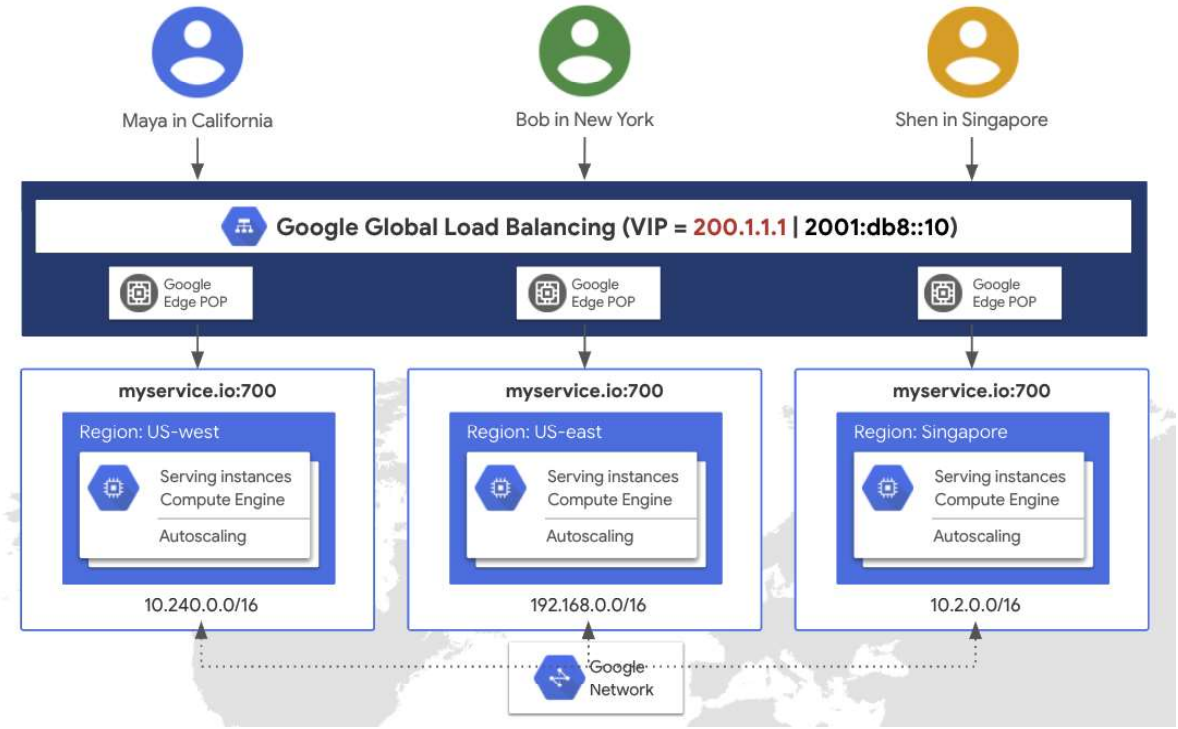

Figure 7.28 – External global HTTP(S) load balancing

Configuring an HTTP(S) load balancer is just one step. You will also need to configure your VMs, Compute Engine instances, and GKE nodes for the backend, as well as having a fixed static IP address and health check firewall rules. Refer to the links in *Further reading* to find a Google Cloud Skills Boost exercise on how to configure an HTTP(S) load balancer.

Hybrid connectivity options

Throughout this chapter, we have made references to hybrid connectivity. The term **hybrid connectivity** means that you can join your Google Cloud network to your on-premises network and a third-party cloud provider using multiple connectivity options. We will look at two options: Cloud VPN (IPSec) and Cloud Interconnect.

Before we look at the different connectivity options, it's important to understand what Cloud Router is, as it is a key component when creating a Google Cloud-based VPN. Cloud Router is a managed service and a regional resource. It is responsible for exchanging routes between your VPC and your on-premises network via BGP. Dynamic routing options include both regional and global. With regional routing, Cloud Router shares routes only for subnets in the region where Cloud Router is provisioned. With global routing, it shares routes for all subnets in the VPC network.

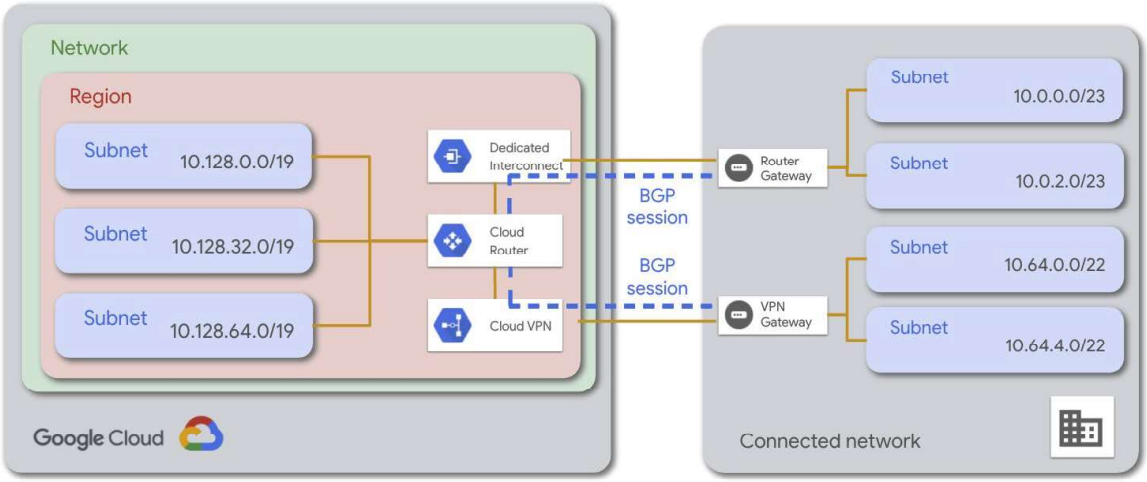

Figure 7.29 – Cloud Router

Figure 7.29 shows how Cloud Router exchanges routes using BGP with a connected network.

Let us now look at the connectivity options that are available for building hybrid connections with Google Cloud:

- **Cloud VPN** (**IPSec-based**): Creating a Cloud VPN instance, which is an IPSec-based VPN over the internet, is the fastest and easiest way to connect your cloud environment with other clouds or with your on-premises network. You can leverage your existing internet connection. This option supports a bandwidth of 1.5-3 Gbps per tunnel, depending on peer location (public or within Google Cloud/direct peer), with multi-tunnel support for higher bandwidth. A VPN device at the on-premises data center is required to support **IPSEC/IKE/Multi-tunnel** protocols.

Figure 7.30 – Cloud VPN

Figure 7.30 illustrates how a VPN using IPSec is established with Google Cloud. This deployment type supports both static- or dynamic (BGP)-based VPNs. You also have the option to configure high availability using gateways with two different interfaces.

- **Cloud Interconnect**: Using dedicated Cloud Interconnect, you can connect your on-premises network with dedicated private links that do not traverse the public internet. The traffic takes fewer hops and gets a high level of reliability and security. Your VPC network internal (`RFC 1918`) IP addresses are directly accessible from your on-premises network.

  There is no requirement for establishing a VPN or NAT. It supports high bandwidth from 8x10 Gbps, or 2x100 Gbps, depending on the need. You can further reduce your egress costs as traffic remains internal. It works with Private Google Access to allow Google API access through internal links. Although the traffic is unencrypted, you can use a self-managed IPSEC VPN if that's a requirement. Cloud VPN over Cloud Interconnect is not supported. You can create VLAN attachments and associate them with Cloud Router. You can also attach to multiple VPCs. Cloud Router is used to dynamically exchange routes.

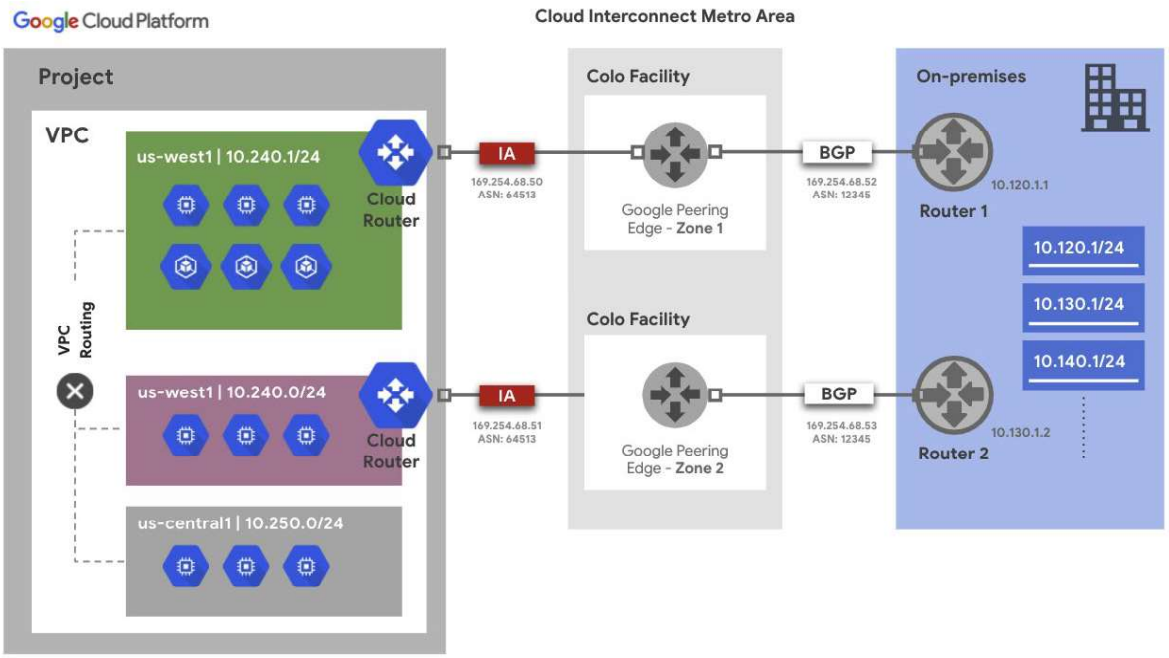

Figure 7.31 – Cloud Interconnect

In *Figure 7.31*, there are two routers in an on-premises location with links to two colo facilities in the same metro area for redundancy. There are `InterconnectAttachment` (**IA**)/VLAN attachments: one from each colo facility to a different Cloud Router instance. BGP sessions between Cloud Router and on-premises routers are used for route exchange. In this illustration, Cloud Router instances are in `us-west1`. To allow access from on-premises to resources in the `us-central1` subnet, global routing would be required.

Best practices and design considerations

In this section, we will look at some VPC best practices and design considerations that you need to factor in when designing and building your secure network on Google Cloud. From an exam perspective, it is important to understand these as you will find questions about best practices.

VPC best practices

For VPC, some of the best practices include the following:

- Prevent overlapping IPs and control subnet creation by creating VPC networks using custom subnet creation mode.
- Reduce management and topology complexity by making use of Shared VPC where possible.
- Group similar applications into fewer, more manageable, and larger subnets.

- Apply organization policies to do the following:
  - Skip the creation of default networks for new projects.
  - Restrict shared VPC host projects and subnets.
  - Restrict VPC peering usage.
- Ensure the design scales to your needs by considering the limitations of each network component.

Key decisions

Some of the key decisions you need to make include the following:

- How will your Google Cloud resources communicate with each other? Will it involve VPC design or Shared VPC?
- How will resources be segmented into networks and subnets? Will there be micro-segmentation?
- How will name resolution be solved among cloud resources and between the cloud and connected environments? Is Cloud DNS and DNSSEC to be factored into your public zones?
- What strategies will be used to connect Google Cloud with corporate networks? Will you use hybrid connectivity options?
- How will your resources communicate with the internet? Will you use external load balancing? Global or regional?

It is important not just to understand best practices but also to consider answering the preceding questions around design considerations, which will help you define requirements and build a resilient and secure network on Google Cloud and across your hybrid network.

Summary

In this chapter, we covered what VPC is and the concepts of regions and zones and how they are designed. We looked at VPC models such as Shared VPC and VPC peering. We covered micro-segmentation strategies such as custom routing, firewall rules, and subnets. We then looked at how to configure Cloud DNS and enable DNSSEC. We covered topics related to different options that are available for Google Cloud load balancing and hybrid connectivity, and finally, we looked at some VPC best practices and design considerations.

In the next chapter, we will cover Context-Aware Access and some more network security aspects, such as Identity-Aware Proxy, web application firewalls, distributed denial of service protection, and Google Private Access.

Further reading

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

- Andromeda: Performance, Isolation, and Velocity at Scale in Cloud Network Virtualization: `https://packt.link/y1soL`
- VPC Networks – Controlling Access: `https://packt.link/iQmOj`
- HTTP Load Balancer with Cloud Armor: `https://packt.link/Wi9Te`
- Cloud Load Balancing overview: `https://packt.link/Wm7kG`
- An overview of DNSSEC: `https://packt.link/D9PfM`
- Migrating DNSSEC zones to Cloud DNS: `https://packt.link/g8HmK`
- Networking in Google Cloud: Defining and Implementing Networks: `https://packt.link/YCEbn`
- Networking in Google Cloud: Hybrid Connectivity and Network Management: `https://packt.link/2BEqK`

8

Advanced Network Security

In this chapter, we will look at advanced network security. This chapter is an extension of the previous chapter and part of the overall network security domain. The chapter will focus more on how to secure your Google Cloud environment using the advanced network security features that are available on Google Cloud. In this chapter, we will be discussing context-aware security and its related topics, such as Identity-Aware Proxy and Private Google Access. We will explore their purpose and learn how to configure them for various use cases.

After that, we will look at Google Cloud **Virtual Private Cloud** (**VPC**), where you can define a context-aware approach to secure your cloud resources. To secure your web applications on Google Cloud, we will look at web application firewalls, followed by learning how you can use services such as Cloud Armor to protect your environment from distributed denial-of-service attacks. We conclude this chapter by looking at network logging and monitoring and some best practices and design considerations.

In this chapter, we will cover the following topics:

- Private Google Access
- Identity-Aware Proxy
- Cloud NAT
- Cloud Armor

Private Google Access

Private Google Access addresses the challenge where you want your **virtual machines** (**VMs**)/**Google Compute Engine** (**GCE**) instances that do not have external IP addresses but private addresses to access Google APIs. Instances without public IP addresses can't access Google Cloud's public API endpoints – but the Private Google Access service enables that capability. Let's look at some use cases on why Private Google Access is required before we learn how to configure the service.

VMs often have to communicate with managed services, for example, Google Cloud Storage, BigQuery, and GCE. Managed services have a public endpoint, for example, `storage.googleapis.com`. Assigning an external IP address to every VM that needs to communicate with a public API wouldn't be a practical or secure approach due to the shortage of valid IPv4 addresses. Private Google Access allows communication with Google API public endpoints without requiring an external IP. When interconnecting with on-premises Cloud **Virtual Private Network** (**VPN**) or Cloud Interconnect, it is possible to extend this so that access from on-premises to Google APIs remains internal. We will study more details on that when we discuss connectivity options.

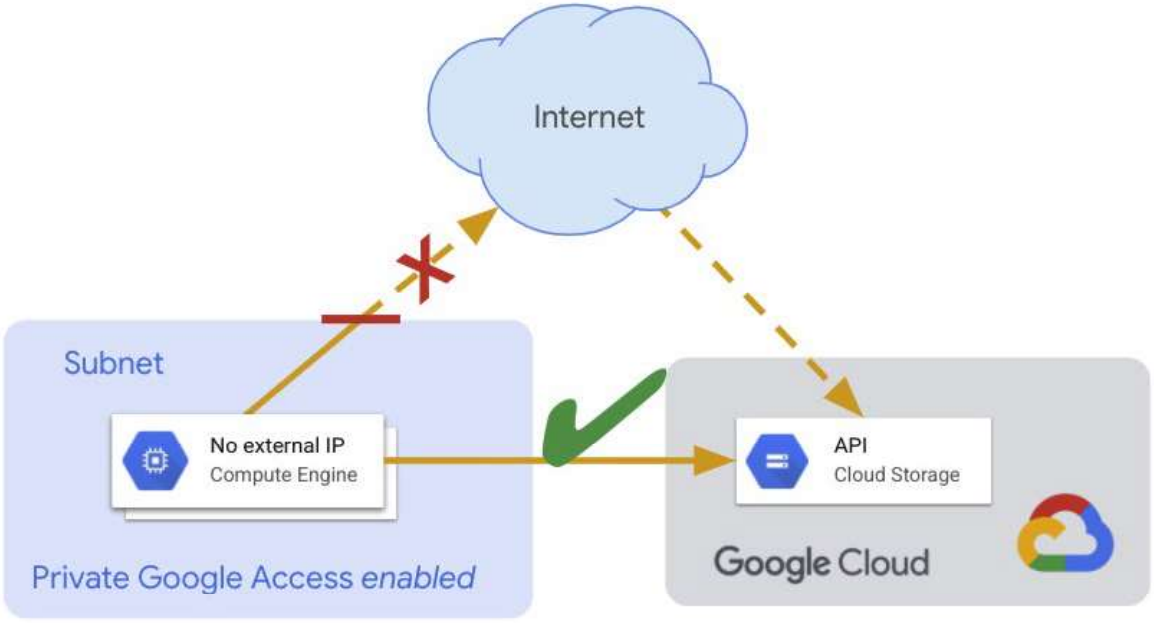

Figure 8.1 – Private Google Access

Figure 8.1 provides a high-level overview of how Private Google Access, when enabled, provides access to the public Google Cloud API securely via private connectivity.

Private Google Access is enabled on a subnet-by-subnet basis; this setting is available to you under the VPC subnet. Let's consider an example where you have two subnets (subnet-a and subnet-b) inside a VPC with four VMs. Subnet-a has Private Google Access enabled and subnet-b does not have Private Google Access. VM1 and VM2 are in subnet-a and VM3 and VM4 are in subnet-b. VM1 and VM3 have private addresses and VM2 and VM4 have external IP addresses. Based on these settings, the following is the case:

- VM1 can access Google APIs as it is part of subnet-a with Private Google Access.
- VM3, on the other hand, has a private IP address but is inside subnet-b, which does not have Private Google Access enabled; therefore, it cannot access Google APIs.

- Finally, VM2 and VM4 both have an external IP address and therefore are able to access Google APIs irrespective of whether they are in a subnet with Private Google Access enabled.

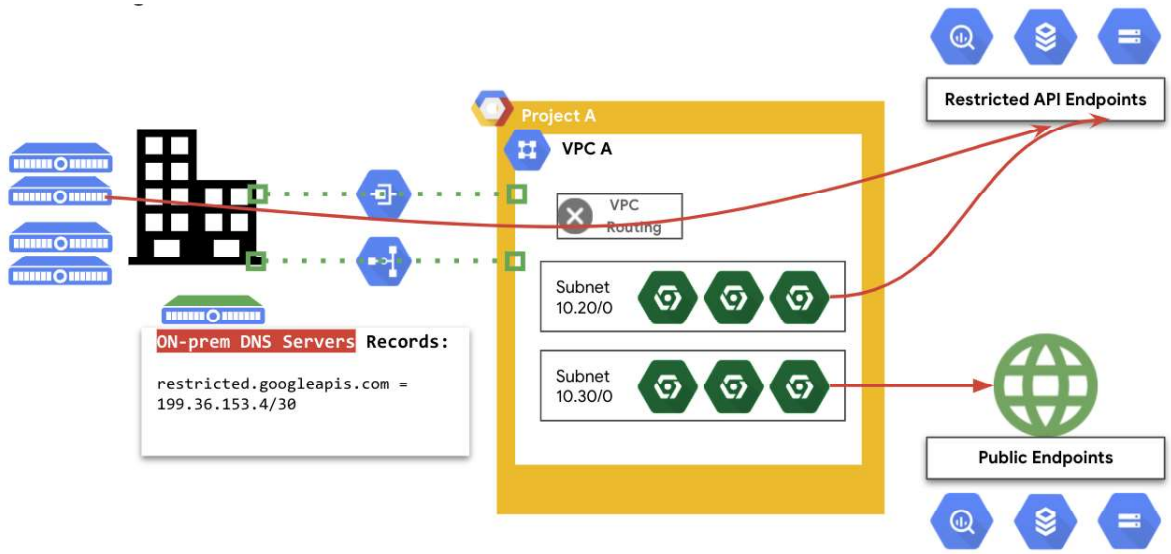

Figure 8.2 – Private Google Access for on-premises services

In *Figure 8.2*, we have an example architecture of how you can use Private Google Access to allow your on-premises services to access Google APIs over Cloud VPN. This same design principle applies if you have Cloud Interconnect instead of Cloud VPN.

Next, we will look at how you can configure Private Google Access for your subnets. But first, we will look at some prerequisites before you do so:

- The VM interface should be in a subnet that has Private Google Access enabled.
- An external IP address should not be assigned to the VM.
- The source IP address of the packet from which the traffic is sent must match the IP address of the primary interface of the VM. Alternatively, it can match an internal IP address that is from an alias IP range.
- The following network requirements must also be met:
  - Private Google Access can only be enabled on VPC networks; legacy networks are not supported as you cannot create subnets in a legacy network.
  - Enabling Private Google Access does not automatically enable it for all APIs. You have to individually select which APIs you need to enable Private Google Access for.
  - If you will be using either of the `private.googleapis.com` or `restricted.googleapis.com` domain names, you will need to create DNS records to direct the traffic to the IP address that is associated with these domains.

- You will need to configure a route to the public endpoints with the default internet gateway as the next hop.
- You will need the *egress firewall* rules to permit traffic to the IP address ranges used by Google APIs and services.
- Correct network admin **identity and access management** (**IAM**) permissions are required for the configuration.

Enabling Private Google Access is a very straightforward thing to do. But before we get to that step, it's important to look at how Google Cloud **Domain Name System** (**DNS**) service and routing options can be configured based on the options that are available as well as the firewall rules.

DNS configuration

Depending on whether you choose `restricted.googleapis.com` or `private.googleapis.com`, you need to ensure that the VMs inside your VPC can resolve DNS requests to `*.googleapis.com`. In this example, look at how you can create a private DNS zone for `*.googleapis.com` and also `*.gcr.io`:

1. Create a private DNS zone for `googleapis.com`. For this, you can use a Cloud DNS private zone.
2. When creating an A record in the `googleapis.com` zone, select one of the following domains:
   - An A record for `private.googleapis.com` pointing to the following IP addresses: `199.36.153.8`, `199.36.153.9`, `199.36.153.10`, and `199.36.153.11`
   - An A record for `restricted.googleapis.com` pointing to the following IP addresses: `199.36.153.4`, `199.36.153.5`, `199.36.153.6`, and `199.36.153.7`
3. Using Cloud DNS, add the records to the `googleapis.com` private zone.
4. In the `googleapis.com` zone, create a CNAME record for `*.googleapis.com` that points to whichever A record you created in the previous step.

Some of the Google APIs are provided using different domain names, such as `*.gcr.io`, `*.gstatic.com`, `*.pkg.dev`, and `pki.goog`. You can follow the same process of creating a DNS private zone using Cloud DNS and creating an A record and CNAME for the respective domains.

Routing options

Routing is important to understand as you do have two different ways to configure this:

- You can use the **default internet gateway**
- You can create a custom route

Based on your requirements, you can choose either of them. VPC comes configured with routing where you have the next hop already configured, and that is the default internet gateway. The VMs have the default internet gateway, where the packets are sent when the VM inside VPC sends a request to the Google APIs. Although the term default internet gateway suggests that all internet traffic is sent to the internet, this is not true. The default internet gateway routes the traffic sent from VMs to Google APIs within Google's network. Google does not allow routing traffic to Google APIs and services through other VM instances or custom next hops.

If your VPC has a default route pointing toward the default internet gateway, you can route all requests from your VM to the Google APIs and services without configuring custom routes.

You can create a custom route and define the next hop in use cases where you want to direct the traffic to Cloud VPN or an internal load balancer. In those cases, you can define a custom route with the next relevant hop.

Routing using a default route

Follow these steps to route using a default route:

1. Go to the **Routes** page in the Google Cloud console.
2. Select **Network Menu** and then **Routes** under the VPC sub-menu.

   Here you have the ability to filter the list of routes to show just the routes for the network you need to inspect (*Figure 8.3*).

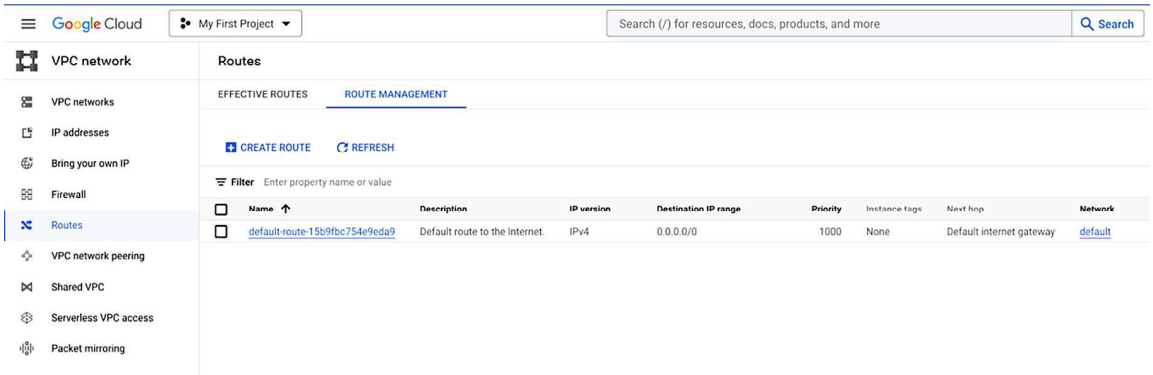

Figure 8.3 – Default internet gateway route

3. Look for a route whose destination is `0.0.0.0/0` and whose next hop is **Default Internet gateway**.

Routing using custom routes

Follow these steps to route using custom routes:

1. On the same **Routes** page in the Google Cloud console, use the **Filter table** text field to filter the list of routes using the following criteria:

- **Network**: `YOUR_NETWORK_NAME`. Here, replace `YOUR_NETWORK_NAME` with the name of your VPC network
- **Next hop**: Choose **Default Internet gateway**

2. Look at the **Destination IP range** column for each route. If you choose the default domains, check for several custom static routes, one for each IP address range used by the default domain. If you choose `private.googleapis.com` or `restricted.googleapis.com`, look for that domain's IP range.
3. You can create a custom route by clicking on **Create a route** and completing the details as indicated in *Figure 8.4*.

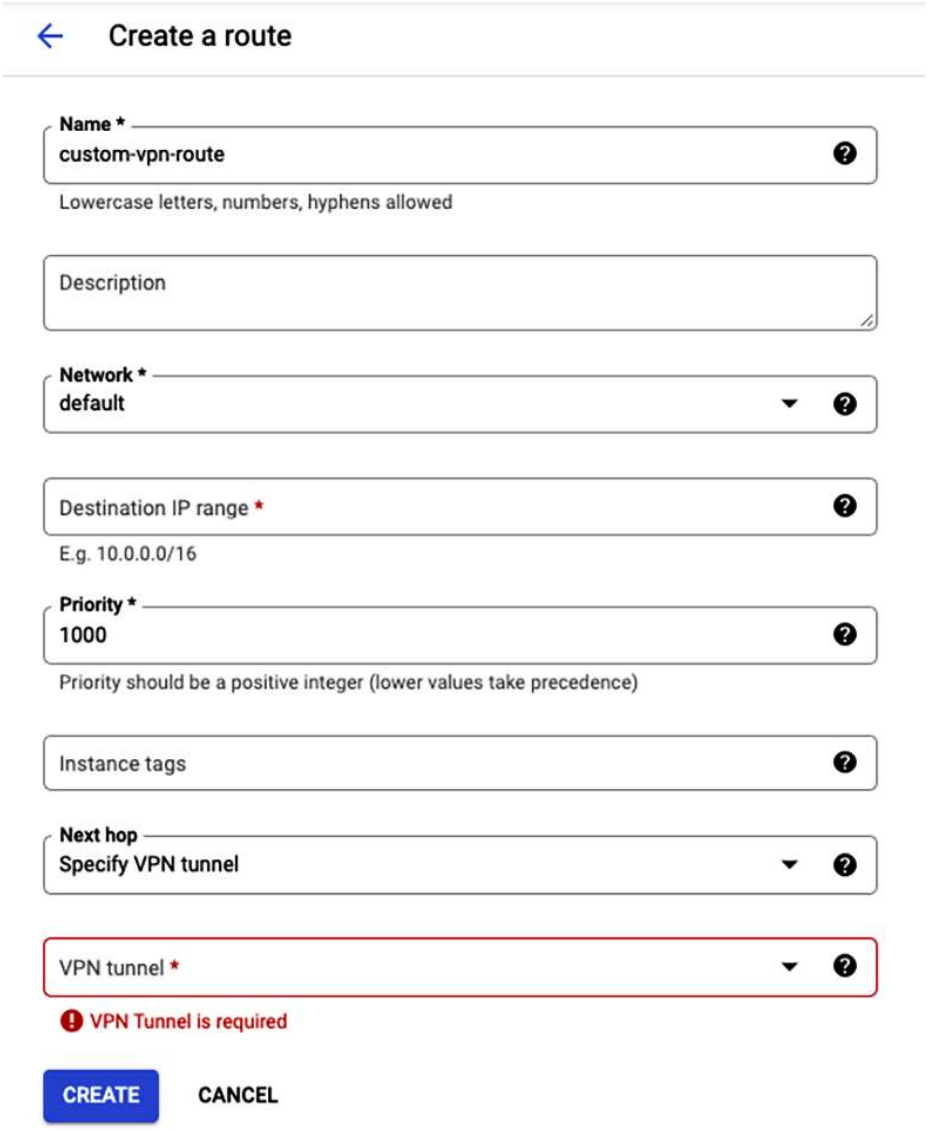

Figure 8.4 – Create a route for the VPN tunnel

4. In *Figure 8.4*, you can specify the name of the route you want to create, the **Network** name, **Destination IP range**, and **Priority**. Specify **Next hop** as the VPN tunnel and select the VPN tunnel that you have already created.

   Instance tags is optional. Also, note that the VPN tunnel name will only appear if you have already created a tunnel.

Firewall rules

Your VPC instance must have firewall rules for your VMs that allow access from VMs to the IP addresses used by Google APIs and services. Every VPC has two implied firewall rules that permit outgoing connections and block incoming connections. The implied allow egress rule satisfies this requirement.

Enabling Private Google Access for your VPC subnet

Follow these steps:

1. Go to the **VPC networks** page in the Google Cloud console.
2. From here, you can click the name of the network that contains the subnet for which you need to enable Private Google Access.
3. If you want to enable Private Google Access for an existing subnet, follow these steps:

   I. Click the name of the subnet. In *Figure 8.5*, you can see the subnet name is **holodeck**.

   II. Once you click on that, you will be taken to the **Subnet** details page.

   III. Here you can edit the details by clicking on **EDIT**.

   IV. Then go to the **Private Google Access** section, set the option to **On**, and click **SAVE**.

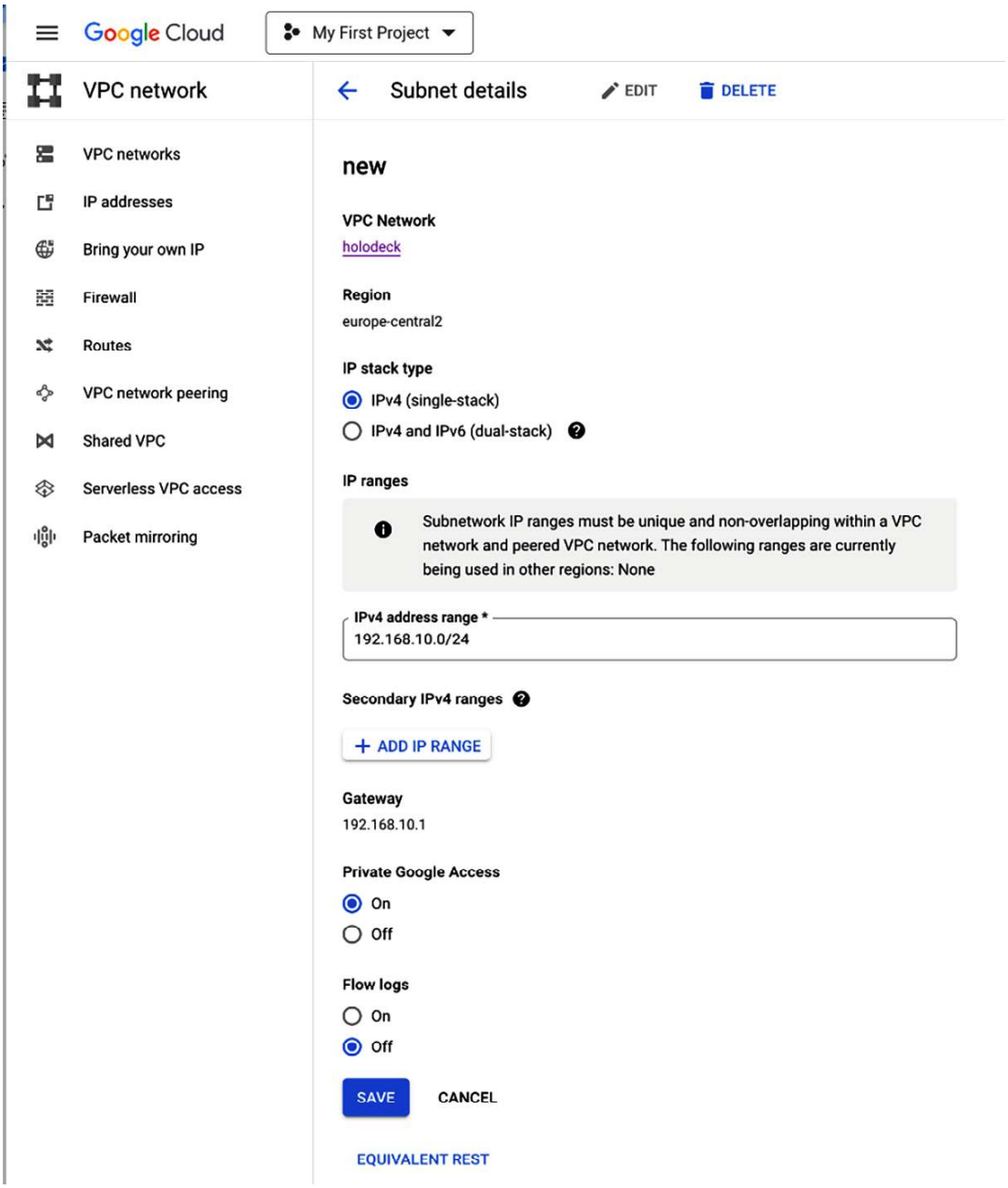

Figure 8.5 – Enable Private Google Access for the subnet

4. If you want to enable Private Google Access for a new subnet, the process is similar:

   I. First, you will need to create the subnet.

   II. While you are on the **VPC Networks** page, click **Add subnet**.

   III. Then specify **Name** and **Region** for the new subnet.

IV. Next, you will need to specify **IP address range** for the subnet. Remember that this range cannot overlap with any subnets in the current VPC network or any networks connected through VPC network peering or VPN.

V. Once you have made changes to the remaining selections on the subnet settings, such as turning on VPC flow logs, you can turn on **Private Google Access** and click **SAVE**.

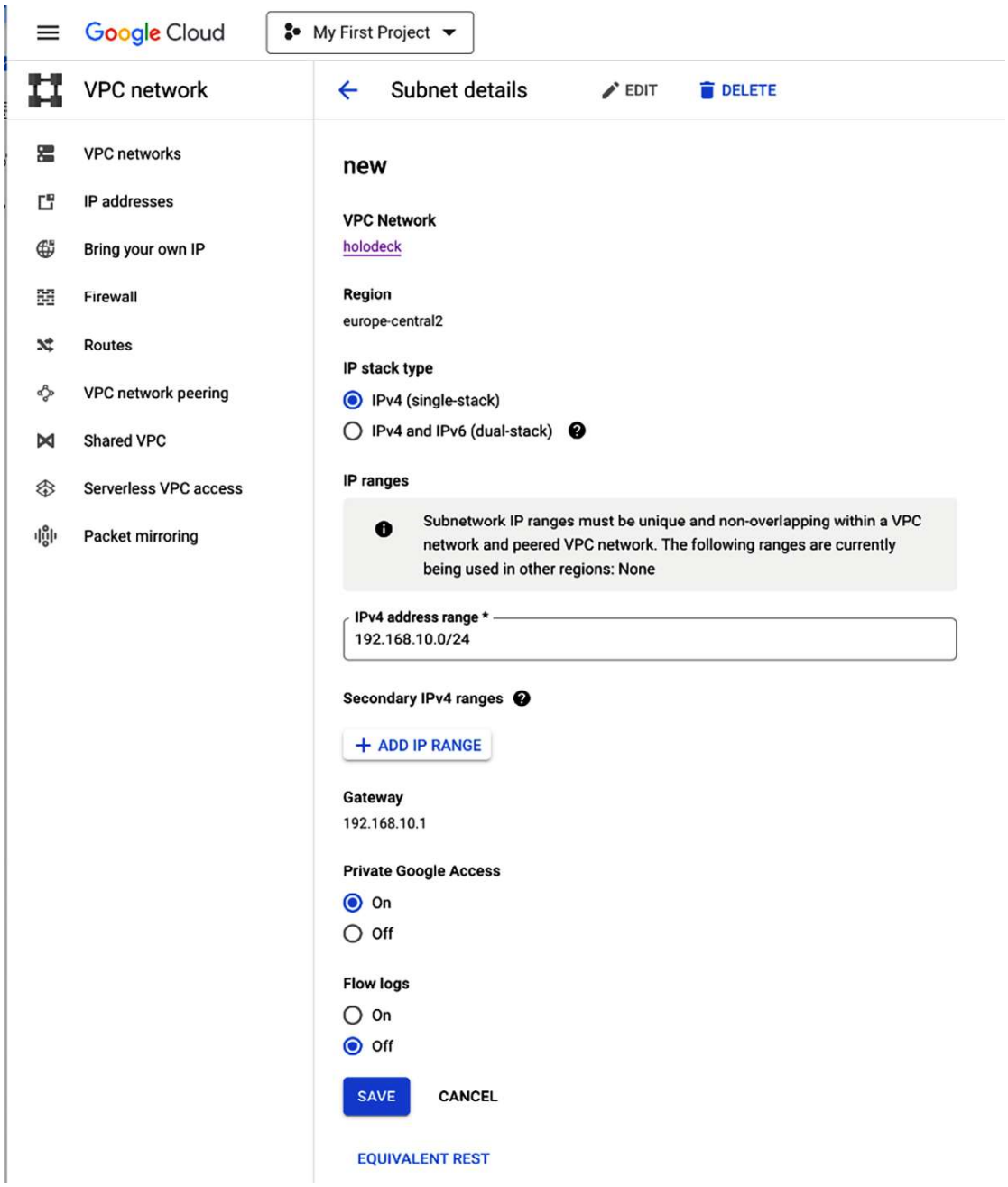

Figure 8.6 – Configuring subnet for Private Google Access

In *Figure 8.6*, the subnet has Private Google Access turned on. You can edit the subnet and turn the setting on or off.

In this section, we looked at how you can enable Private Google Access for subnets that can allow your VMs—either on-premises or on Google Cloud—to secure access to Google APIs.

Identity-Aware Proxy

IAP lets you configure centralized authorization to manage secure remote access to your VMs and applications. IAP and load balancers are in front of all your data requests. This provides a much simpler administration process, with less operational overhead, than more traditional VPN solutions. There is no VPN to implement and no VPN clients to install and maintain. It also makes the end user experience more streamlined as the user no longer has to launch the VPN client and sign in to the VPN.

In comparison to a traditional VPN, IAP takes the approach of application-based access control instead of network-based access control. Access is only possible through IAP by users who have been configured with the right IAM role. Authentication is done via Google Cloud Identity or a federated identity provider, including 2FA. To configure authorization using Cloud IAM, users need the IAP-secured Web App User role on the resource project to be configured. We will look at the steps on how to configure that in the *Enabling IAP for on-premise* section later in this chapter.

Using IAP, you can not only secure access to resources on Google Cloud, such as GCE, Google App Engine, and GKE, but you can also create secure access for your on-premises or third-party cloud provider. We will look at an example of how you can configure secure access using IAP for on-premises resources later in this section.

IAP has tight integration with some other Google Cloud products, such as Cloud IAM, Cloud Identity, and Access Context Manager. You can configure Access Context Manager to enforce additional security checks when giving access to a user. For example, you can check for geo-location, enforce IP address whitelisting, and also do endpoint device checks, such as checking for an allowed operating system, checking for X.509 certificates, checking for disk encryption, and so on. These additional controls not only give you visibility of the user context and aspects such as single and multi-factor authentication but also perform device checks before access is granted. The authentication server utilizes the request credentials, if they are legitimate, to determine the user's identity (email address and user ID). When a user is authenticated, the authentication server examines their IAM role to see whether they are allowed to access the resource in question.

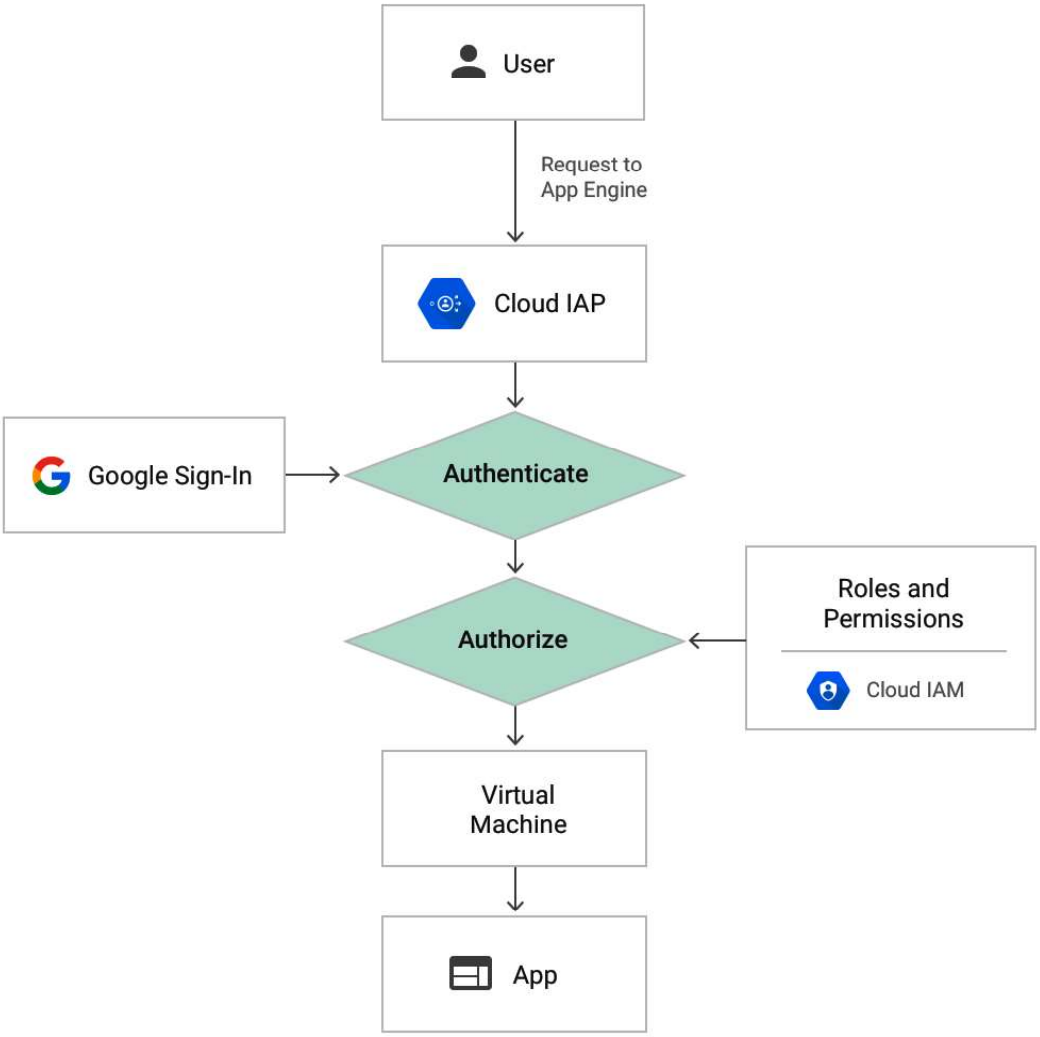

Figure 8.7 – How IAP works with App Engine

Let's look at *Figure 8.7* to understand how IAP works with App Engine. When an application or resource is protected by IAP, only users with the appropriate IAM rights (the IAP-secured Web App User role) can access it. In order to access protected resources, IAP runs both authentication and authorization checks on the requesting user.

IAP intercepts the request to access the resource and checks to see whether IAP is enabled for the service. If this option is enabled, any IAP credentials that appear in the request headers or cookies are submitted to an IAP authentication server. An OAuth 2.0 Google account sign-in flow that stores a token for future sign-ins is redirected to the user if they do not have browser credentials.

For GKE and GCE, users can bypass IAP authentication if they can access the VM's application-serving port. IAP-secured applications can't be protected from code executing on a different VM since firewall restrictions can't stop it. A similar aspect applies to Cloud Run, where, if a user has an auto-assigned URL, they can bypass the IAP authentication. In order to safeguard against traffic that does not originate from the serving infrastructure, you must implement firewall rules and a load balancer. Other alternatives, such as using ingress controls for Cloud Run and App Engine, can be used to sign headers.

Next, we will cover how IAP can work to support your on-premises applications' access. There is a new component that will be introduced in the setup, which is the IAP connector. We will understand what this connector is and also look under the hood of the IAP connector. From an exam perspective, you will not be tested on your knowledge of how an IAP connector works, but it's good to understand it anyway.

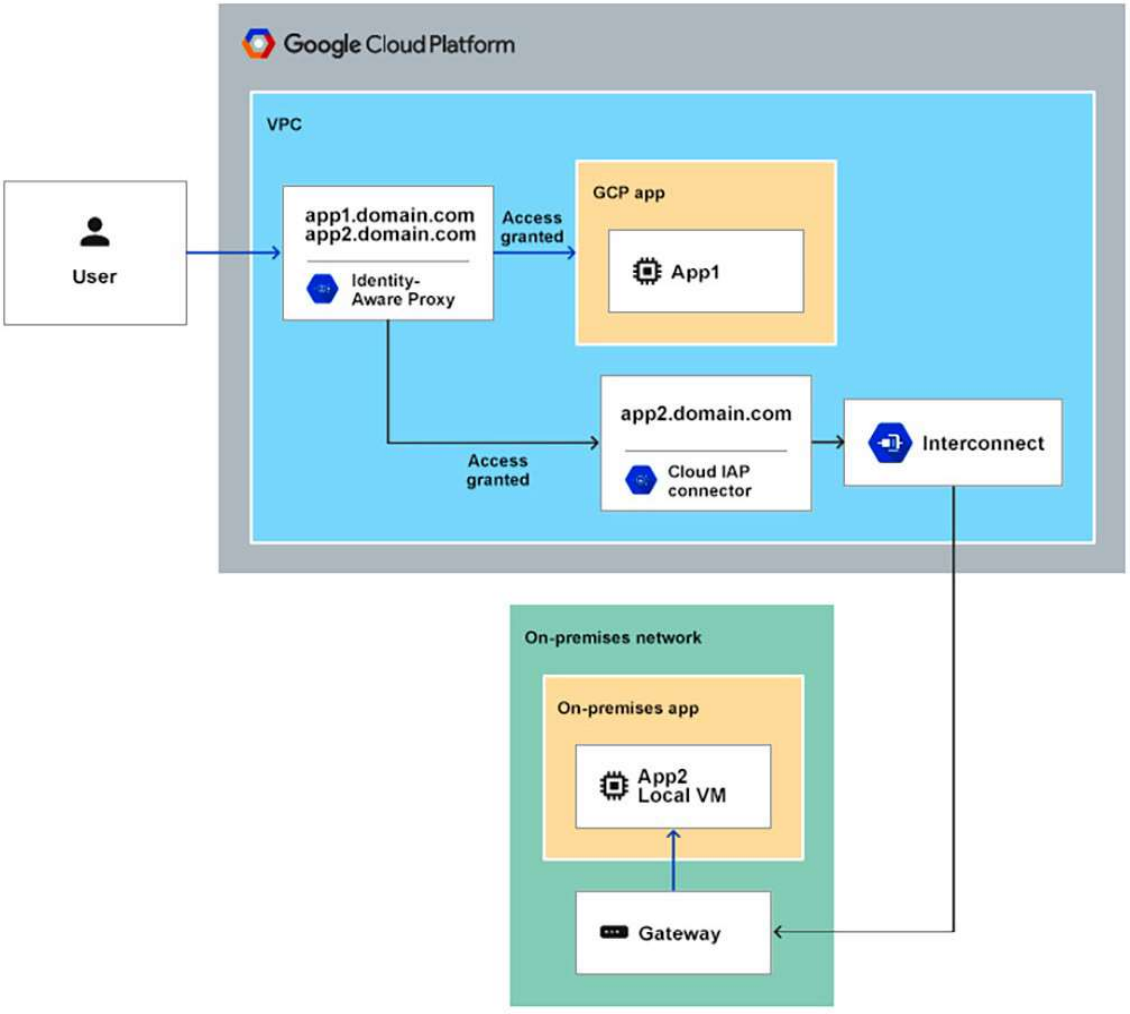

Figure 8.8 – How IAP for on-premises works

We will use the illustration given in *Figure 8.8* to understand the flow and the key components of IAP for on-premises. We will then look at how secure access is given to applications that are connected via a hybrid connector from Google Cloud to on-premises.

The initial part of the workflow is similar to how IAP works for App Engine, as explained earlier. There is no difference between the authentication and authorization checks that are performed for the user. The request is then sent to the IAP connector, which forwards the request to the on-premises network over a site-to-site connection established by Interconnect.

IAP targets on-premises applications with the use of an IAP connector. You can create an IAP connector from the IAP dashboard of the Google Cloud console. An IAP connector is deployed using a configurable Google Cloud Deployment Manager template. This template creates the resources that are needed to host and run the IAP connector. Inside your Google Cloud project, the template deploys an Ambassador proxy on a GKE cluster. This proxy is responsible for routing the traffic that is secured by Cloud IAP to your on-premises applications, by indirectly applying Google **Cloud Identity and Access Management** (**Cloud IAM**) access policies.

You have to define the routing parameters for the IAP connector. Every routing name must correspond to an ambassador-created GCE backend resource. The mapping parameter has to define the routing rules with the source and destination. This is to ensure that the requests can be routed from the source to the correct destination, which could be an application on the on-premises network. The following example shows what a mapping should look like:

```
routing:
      - name: crm
        mapping:
          - name: host
            source: www.crm-domain.com
            destination: crm-internal.domain.com
          - name: sub
            source: customer.hr-domain.com
            destination: customer.hr-internal.domain.com
      - name: billing
        mapping:
          - name: host
            source: www.billing-domain.com
            destination: billing-internal.domain.com
```

In this example, we route the incoming requests coming from `www.crm-domain.com` to `crm-internal.domain.com` and therefore create a mapping for both the host and subdomain as well. Later, we will look at the steps to enable IAP for on-premises applications.

An important point to note here is that IAP by default is integrated with Google identities and uses services such as Google Cloud Identity and Cloud IAM. However, you do have the option to use external identities instead of Google Cloud Identity Platform, which is Google's customer IAM solution. Google Cloud Identity Platform can be used with IAP. Google Cloud Identity Platform is a completely different product and is not in scope for the exam.

Next, we will look at how you can use TCP forwarding with IAP. IAP use cases are limited to supporting applications that use HTTP or HTTP(S). At the time of writing this book, there is no support for thick clients. But you can use TCP forwarding to support both **Secure Shell** (**SSH**) and **Remote Desktop Protocol** (**RDP**). It is possible for users to connect to any TCP port on GCE via TCP forwarding. IAP opens a listening port and sends all the traffic to the chosen instance. IAP uses HTTPS to encrypt all client-side traffic. The `gcloud compute ssh` IAP encapsulates SSH in HTTPS and passes it to the remote instance, eliminating the need for a listening port when using SSH with `gcloud`. TCP forwarding only works with GCE using IAP and not with on-premises apps.

Enabling IAP for on-premises

There are four different configurations that you should be familiar with: enabling IAP for on-premises apps, enabling IAP for App Engine, enabling IAP for GKE, and enabling IAP for GCE. In the following example, you will see how Cloud IAP for App Engine is enabled and how access policies for your app are added.

IAP lets you manage access to services hosted on App Engine, GCE, or an HTTPS load balancer. Perform the following steps to configure IAP for App Engine:

1. Open the menu on the left side of the Google Cloud console.
2. Click **Security**.
3. Select **Identity-Aware Proxy**. All the resources of a project are listed on the **Identity-Aware Proxy** page. If you haven't already, make sure you've configured your **OAuth Consent** screen.
4. After doing that, come back to the **Identity-Aware Proxy** page. Follow the steps to configure the consent screen:

    I. Go to the **OAuth consent** screen.

    II. Under **Support email**, select the email address that you want to display as a public contact. The email address must belong to the currently logged-in user account or a Google Group to which the currently logged-in user belongs.

III. Enter the application name you want to display.

IV. Add any optional details you'd like.

V. Click **Save**.

5. Navigate back to the **Identity-Aware Proxy** page and select an app that you want to secure. If you don't see your app here, ensure you have App Engine configured. In the case of GCE and GKE, ensure you have a load balancer configured with backends. We covered the ways to create load balancers in *Chapter 7*, *Virtual Private Cloud*.

> **Note**
>
> By enabling IAP for your selected app, only the authenticated and authorized members can access it.

6. Select an app by clicking the checkbox on the left side of a row.
7. Turn on the IAP toggle switch for the app that you have selected.
8. To confirm that you want your resource to be secured by IAP, click **Turn on** in the **Turn on IAP** window that appears.

   To control who has access to the app, members need to be given access. In the following step, you will specify the members and assign them appropriate roles to access your app.

9. Click on the **Add Member** button in the info panel. This opens the **Add Member** panel. This panel temporarily hides the example shared here.
10. In the **Add Member** panel, enter the users and/or groups in the **New members** field.
11. Select one of the IAP roles that you want to grant to each new member. These roles are as follows:

    I. **IAP Policy Admin**: Grants administrator rights over Cloud IAP permissions

    II. **IAP-secured Web App User**: Grants access to HTTPS resources that use Cloud IAP

12. Click **Save**.

Using Cloud IAP for TCP forwarding

In this section, we will look at how you can use IAP for TCP forwarding. When you create GCE instances, they appear on the **Identity-Aware Proxy** page under **SSH AND TCP RESOURCES**.

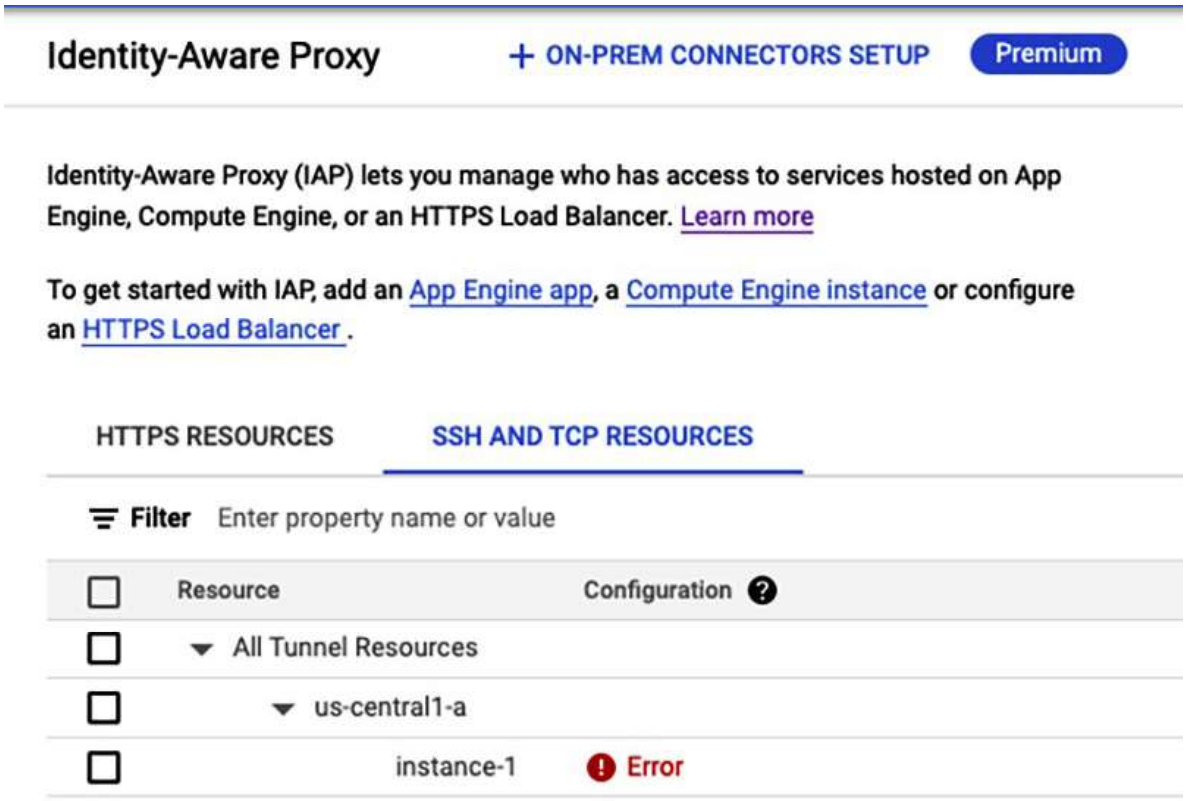

Figure 8.9 – The Identity-Aware Proxy dashboard

Figure 8.9 illustrates where you will see the GCE instances appear after they are created. You can see an `Error` message (in red) displayed in the figure. The error message appears because a corresponding firewall rule – one that will allow this instance to be accessible using IAP – does not exist.

So, let's create the firewall rule and revisit this page and see the error message disappear. Let's look at how you can create the firewall rule:

1. Click on the error message and it will take you to a page that provides details of the missing access level with the source IP address range that you need to add to your firewall rule.

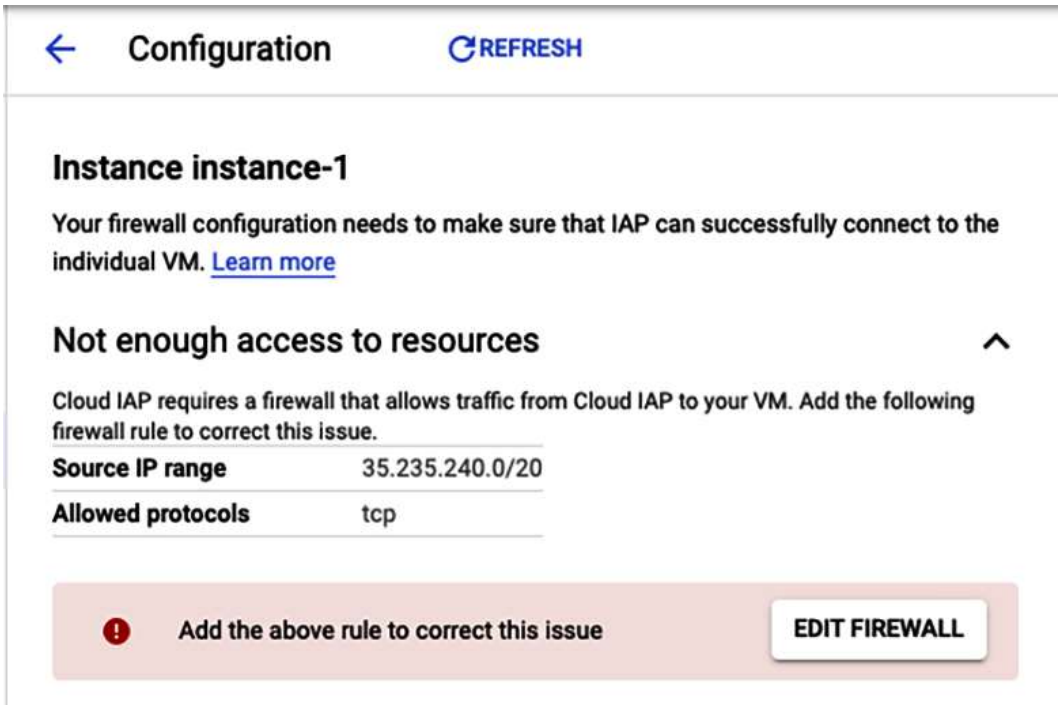

Figure 8.10 – Correcting a firewall configuration for IAP

Figure 8.10 shows the message when you click on the error.

2. From here, you can click on **EDIT FIREWALL** and it will take you directly to the firewall settings page. Here you can configure the firewall rule.

Let's look at the rule that we need to create for the VM to be accessed using IAP. To grant RDP and SSH access to all your VMs, perform the following steps:

1. Open the **Firewall Rules** page and click **Create a firewall rule**.
2. Configure the following settings:
    I. **Name**: `allow-ingress-from-iap`
    II. **Direction of traffic**: `Ingress`
    III. **Target**: `All instances in the network`
    IV. **Source filter**: `IP ranges`
    V. **Source IP ranges**: `35.235.240.0/20`

VI. **Protocols and ports**: Select `TCP` and enter `22,3389` to allow both RDP and SSH.

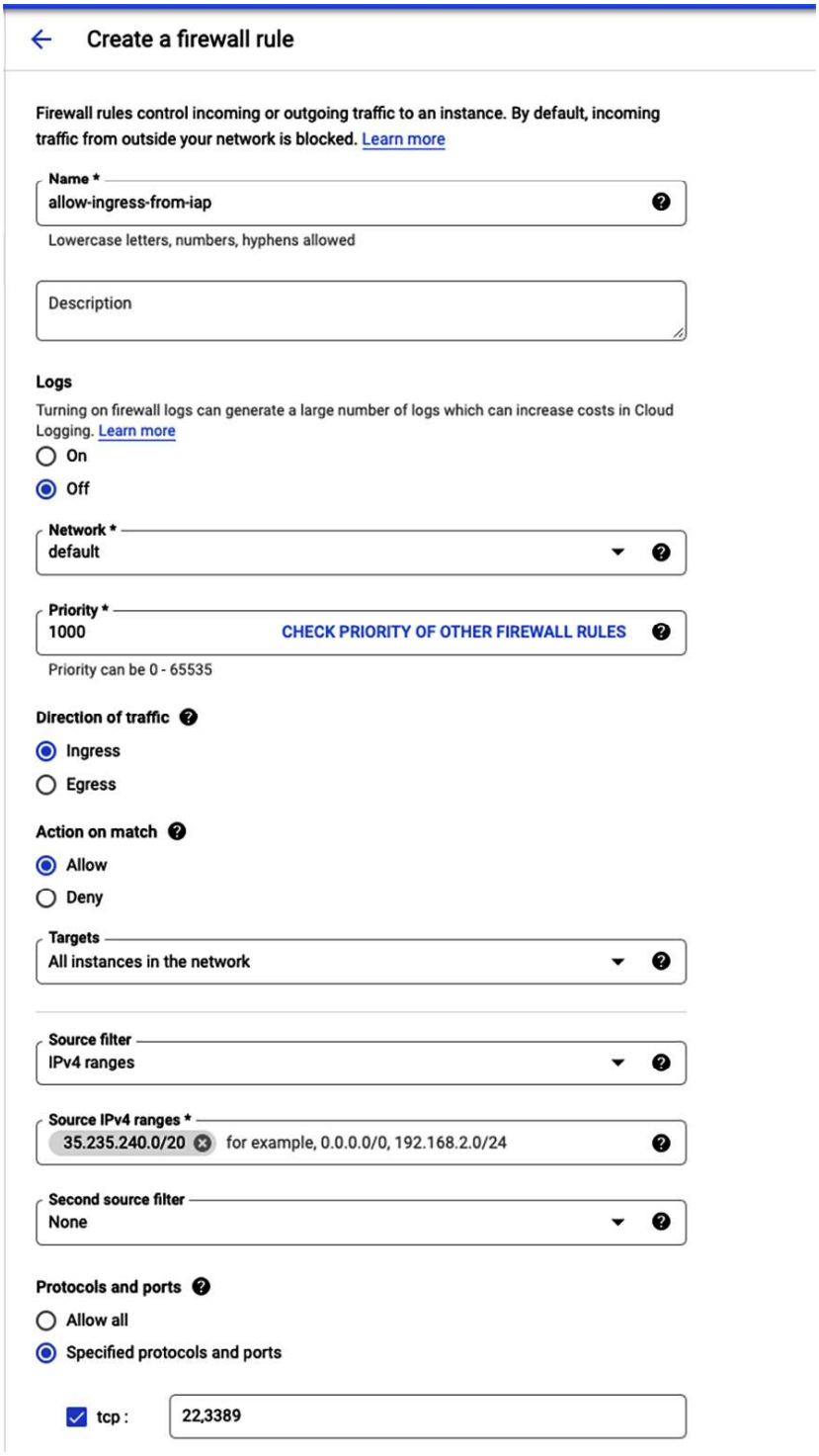

Figure 8.11 – Creating a firewall rule to enable IAP access