- 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.

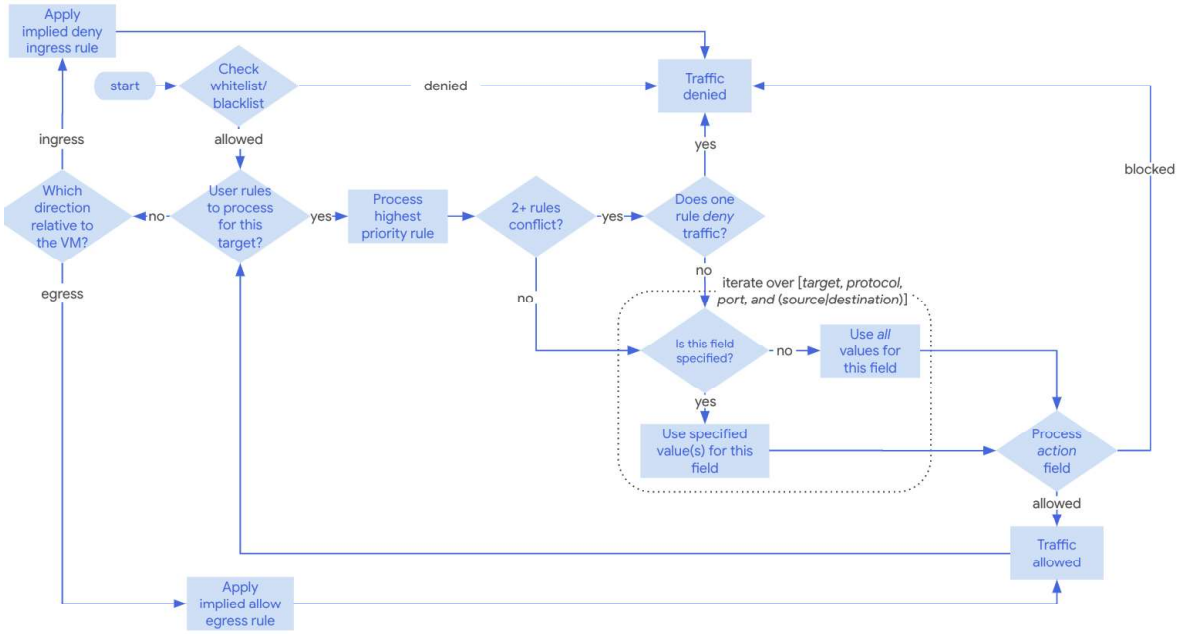

Figure 7.19 – Firewall rule evaluation logic

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.

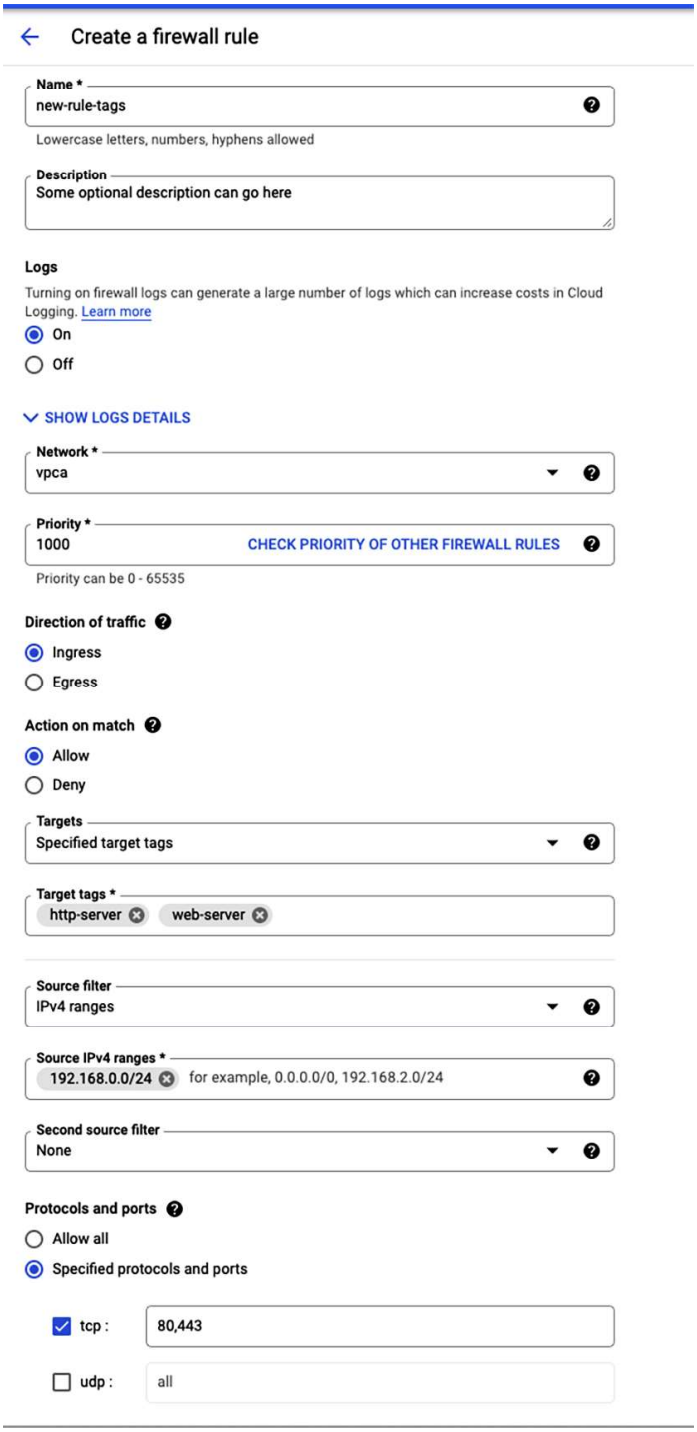

Figure 7.20 – Create a firewall rule using target tags

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.

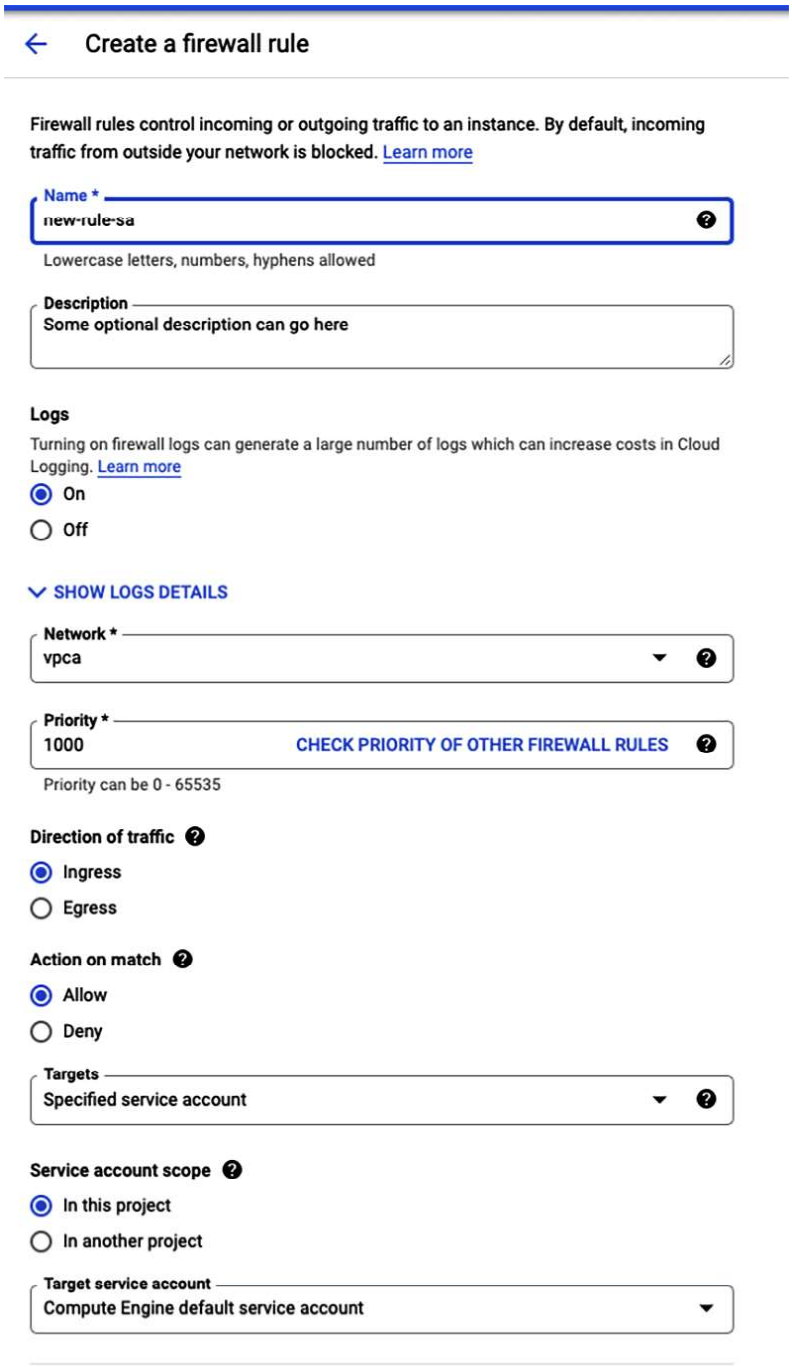

Figure 7.21 – Create a firewall rule using the service account

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.

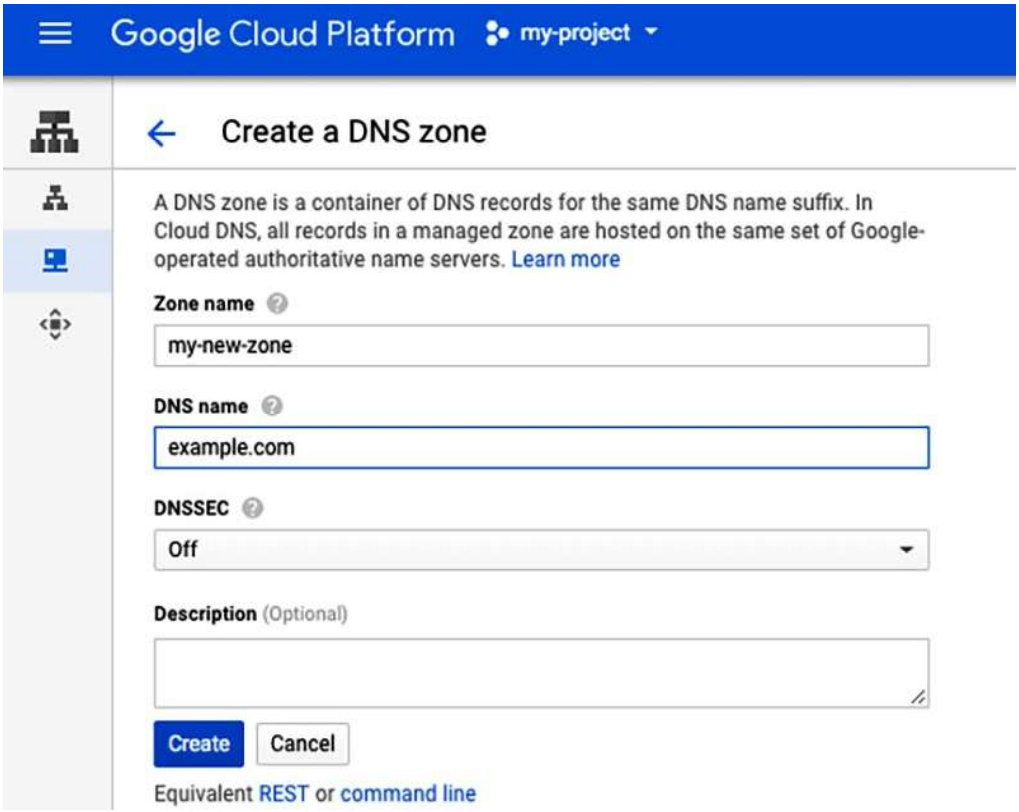

Figure 7.22 – Create a DNS zone

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.

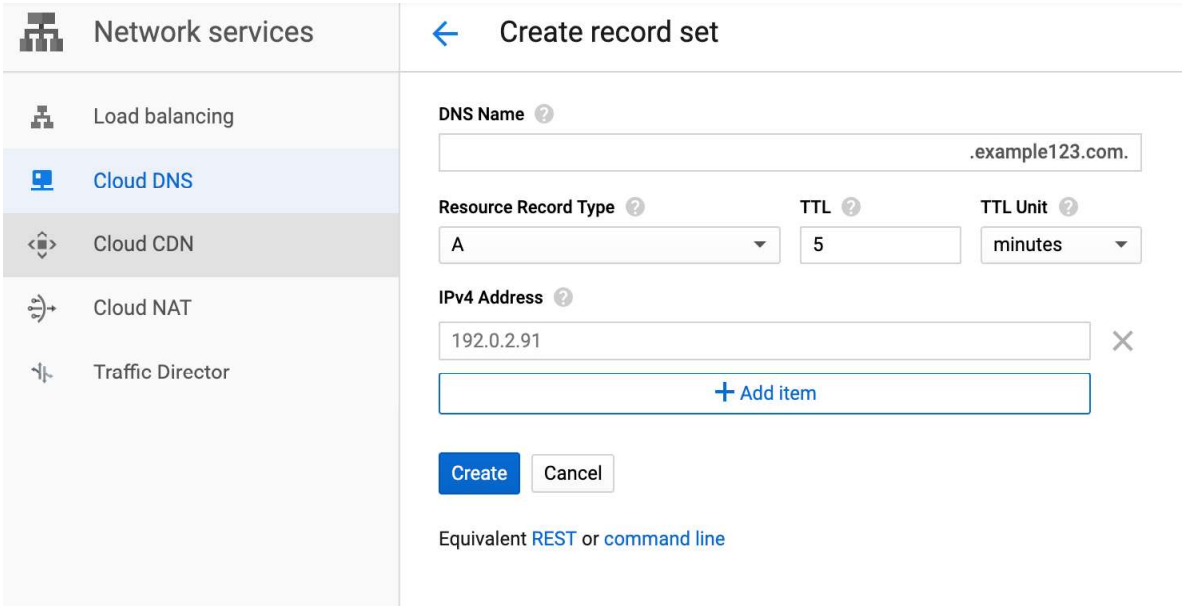

Figure 7.23 – Create a record set

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.`).

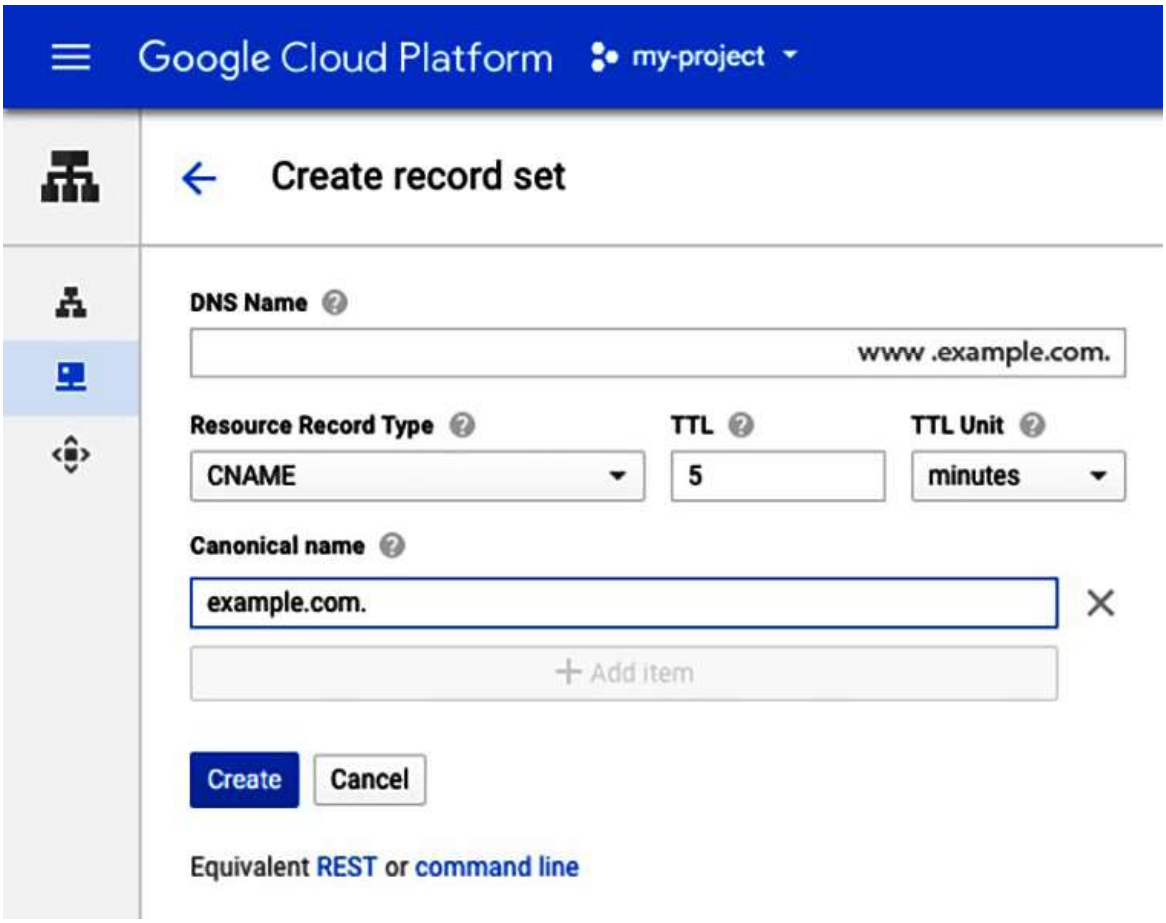

Figure 7.24 – Create a CNAME record

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*.

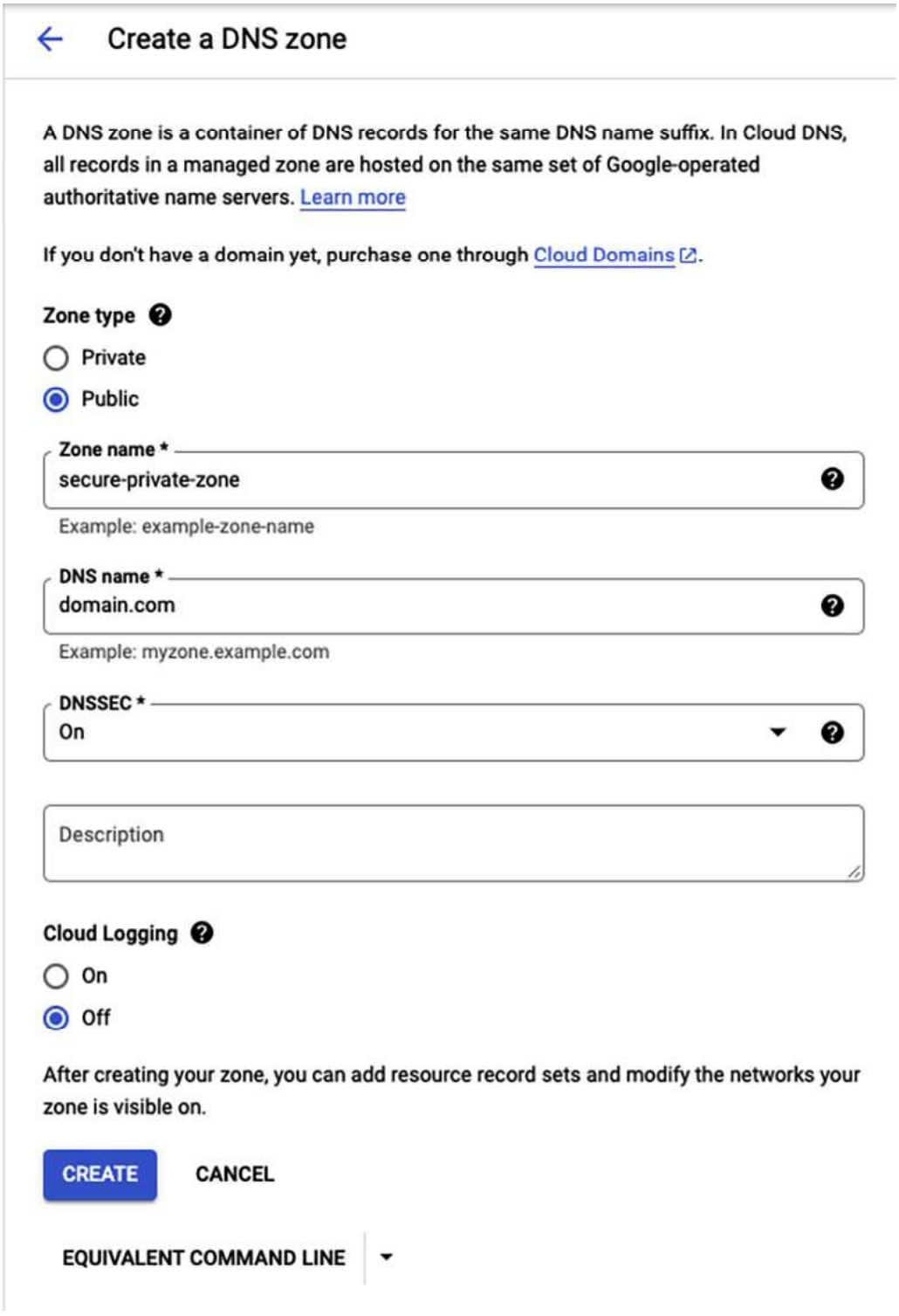

Figure 7.25 – Enable DNSSEC for a private DNS zone

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.

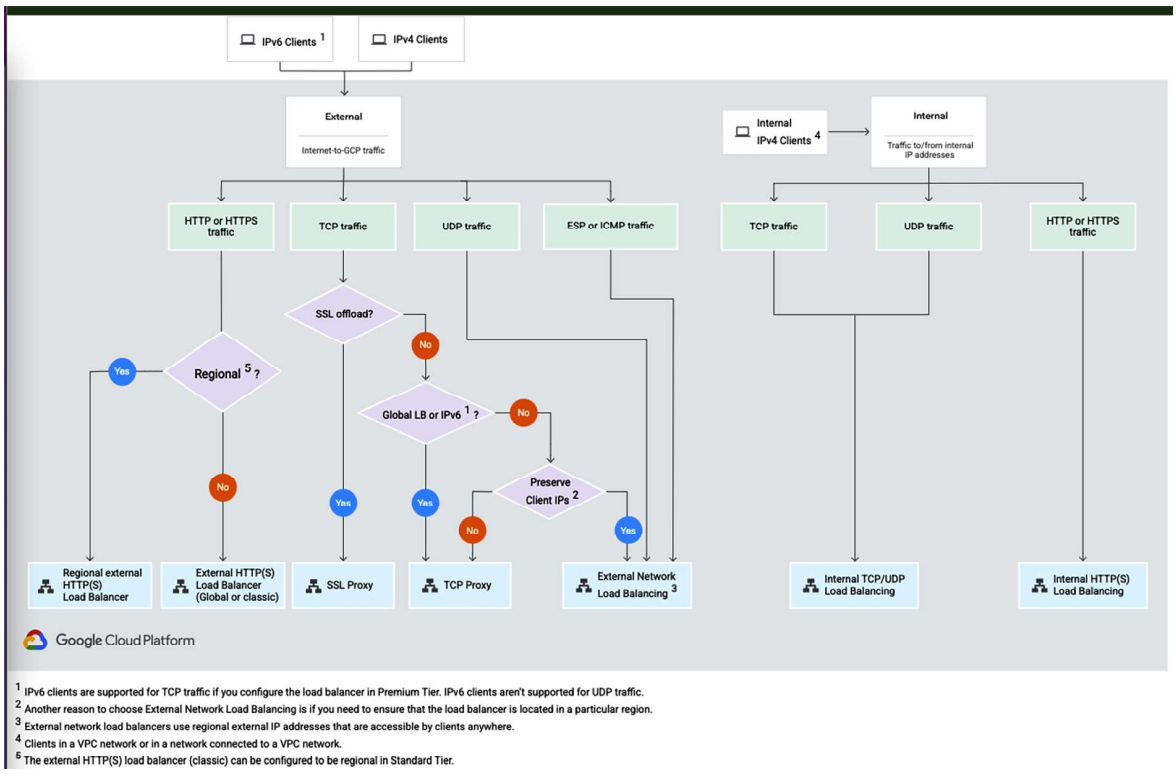

Figure 7.27 – Decision tree for choosing a load balancer

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.

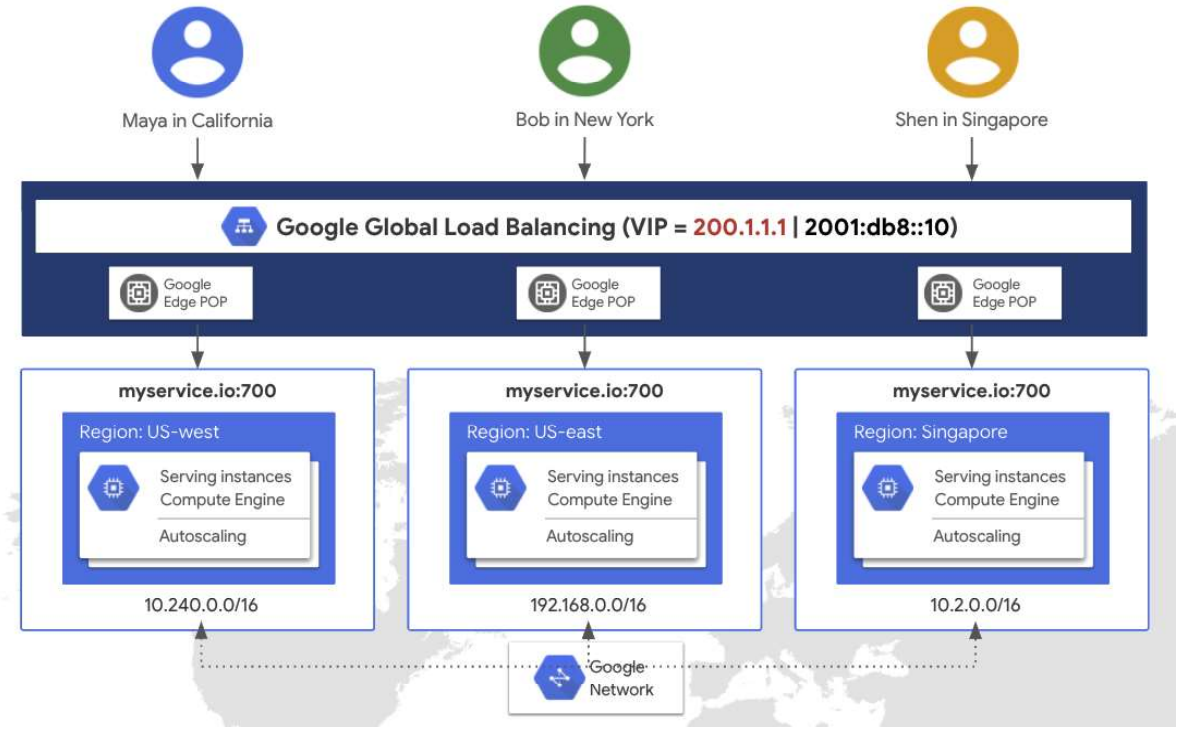

Figure 7.28 – External global HTTP(S) load balancing

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.

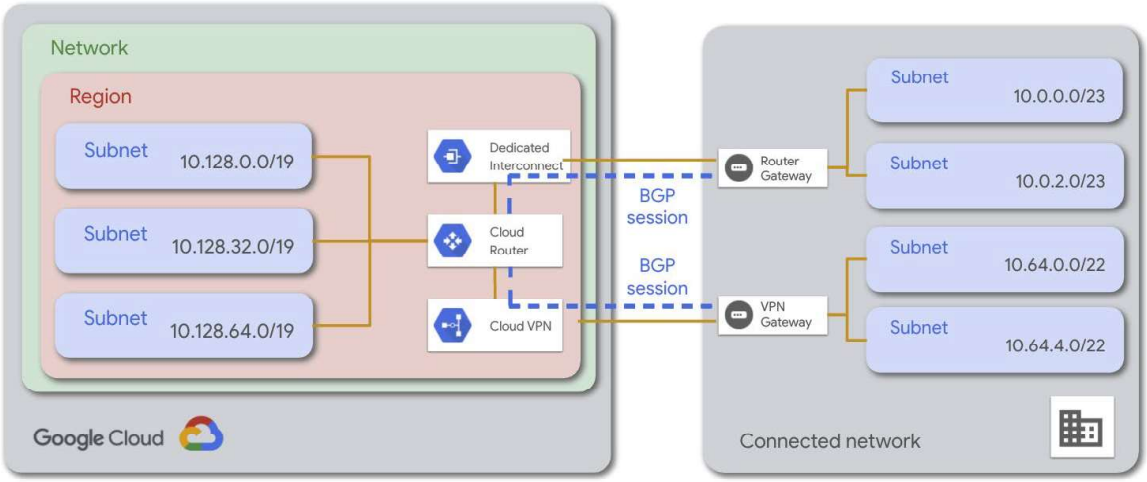

Figure 7.29 – Cloud Router

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.

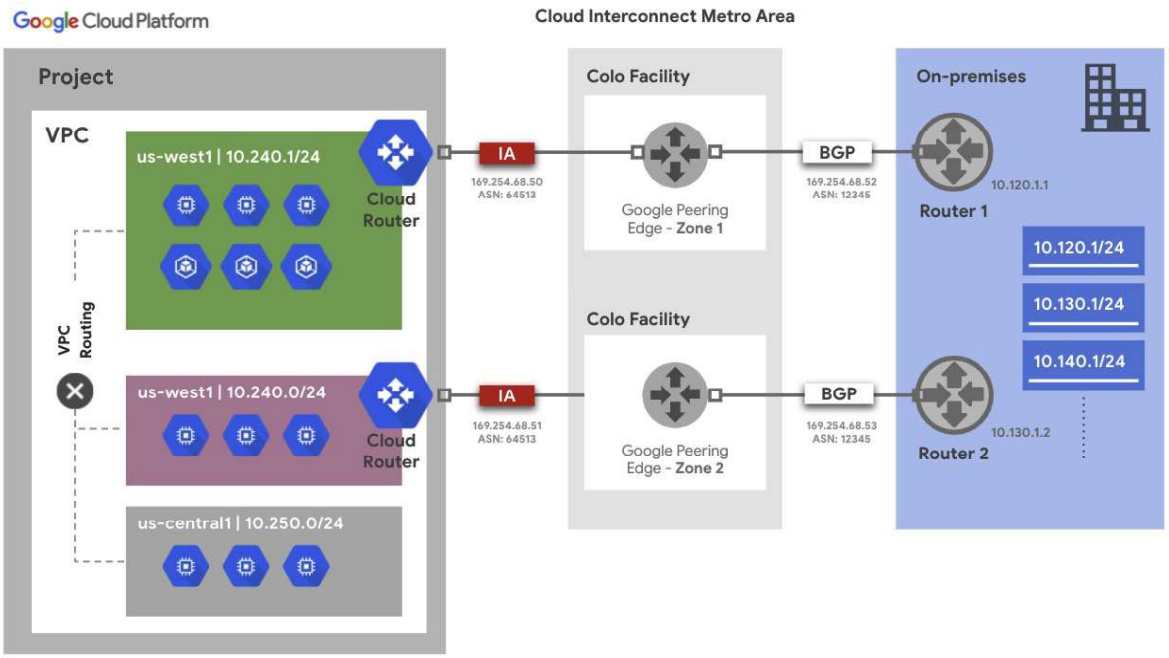

Figure 7.31 – Cloud Interconnect

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.

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.

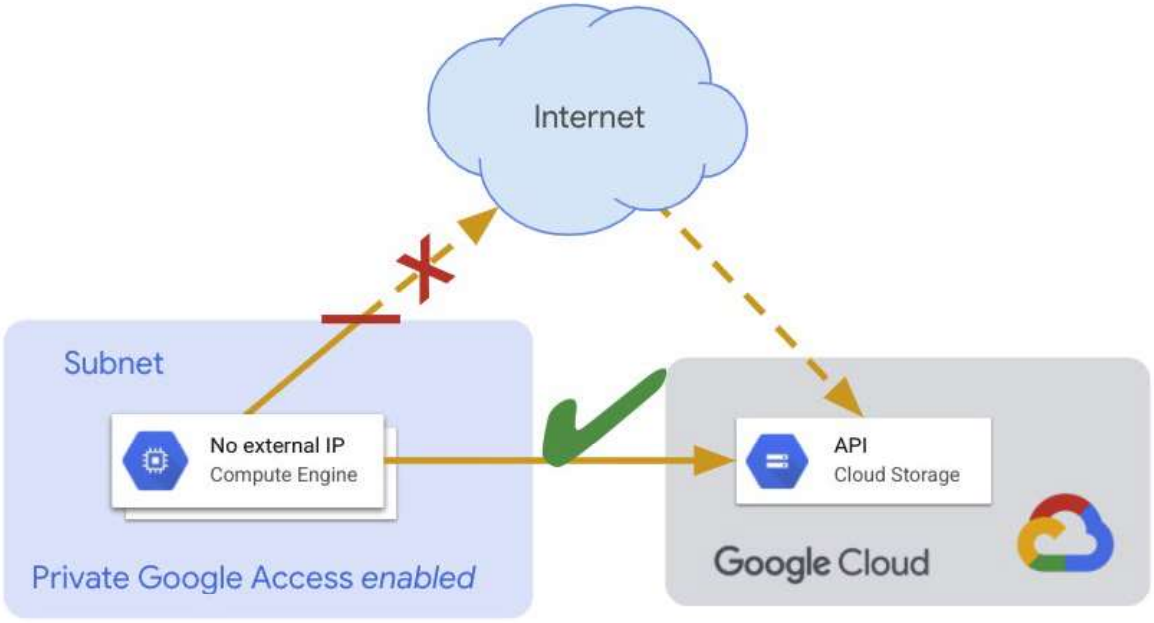

Figure 8.1 – Private Google Access

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.

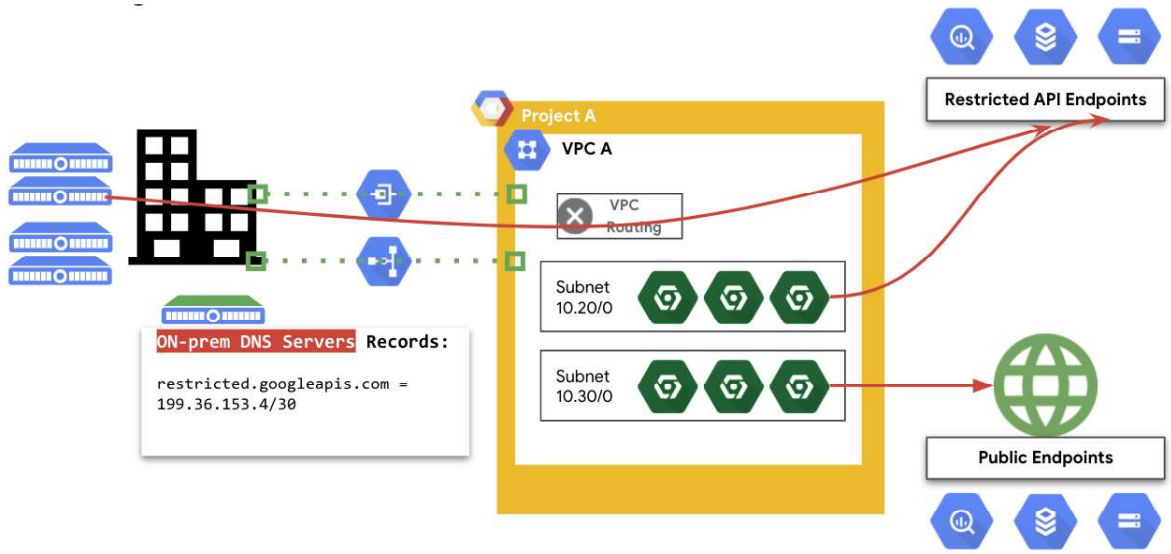

Figure 8.2 – Private Google Access for on-premises services

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*).

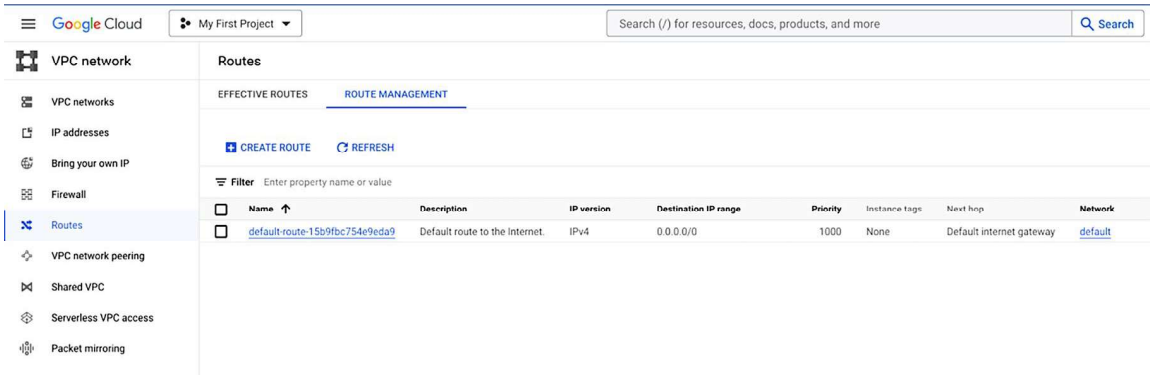

Figure 8.3 – Default internet gateway route

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*.

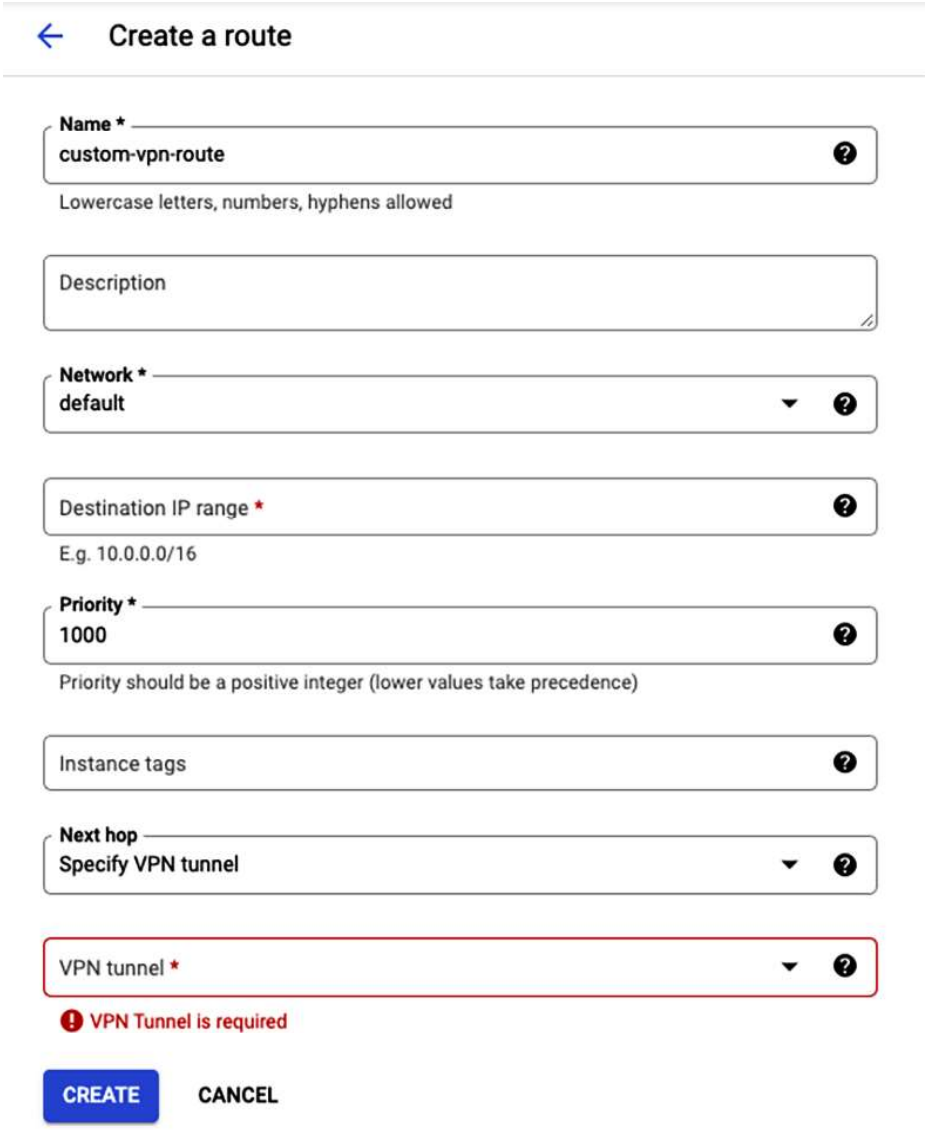

Figure 8.4 – Create a route for the VPN tunnel

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**.

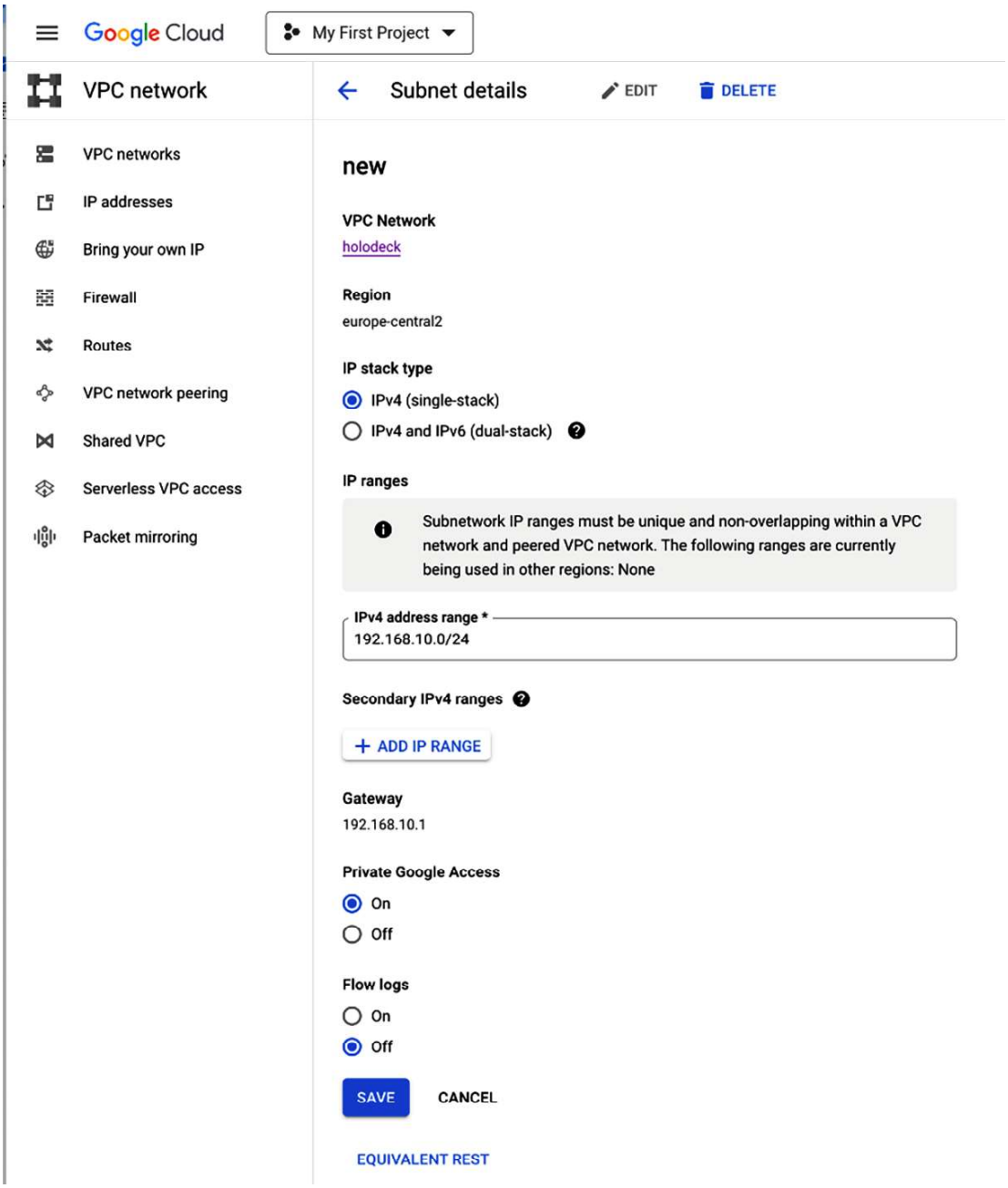

Figure 8.5 – Enable Private Google Access for the subnet

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**.

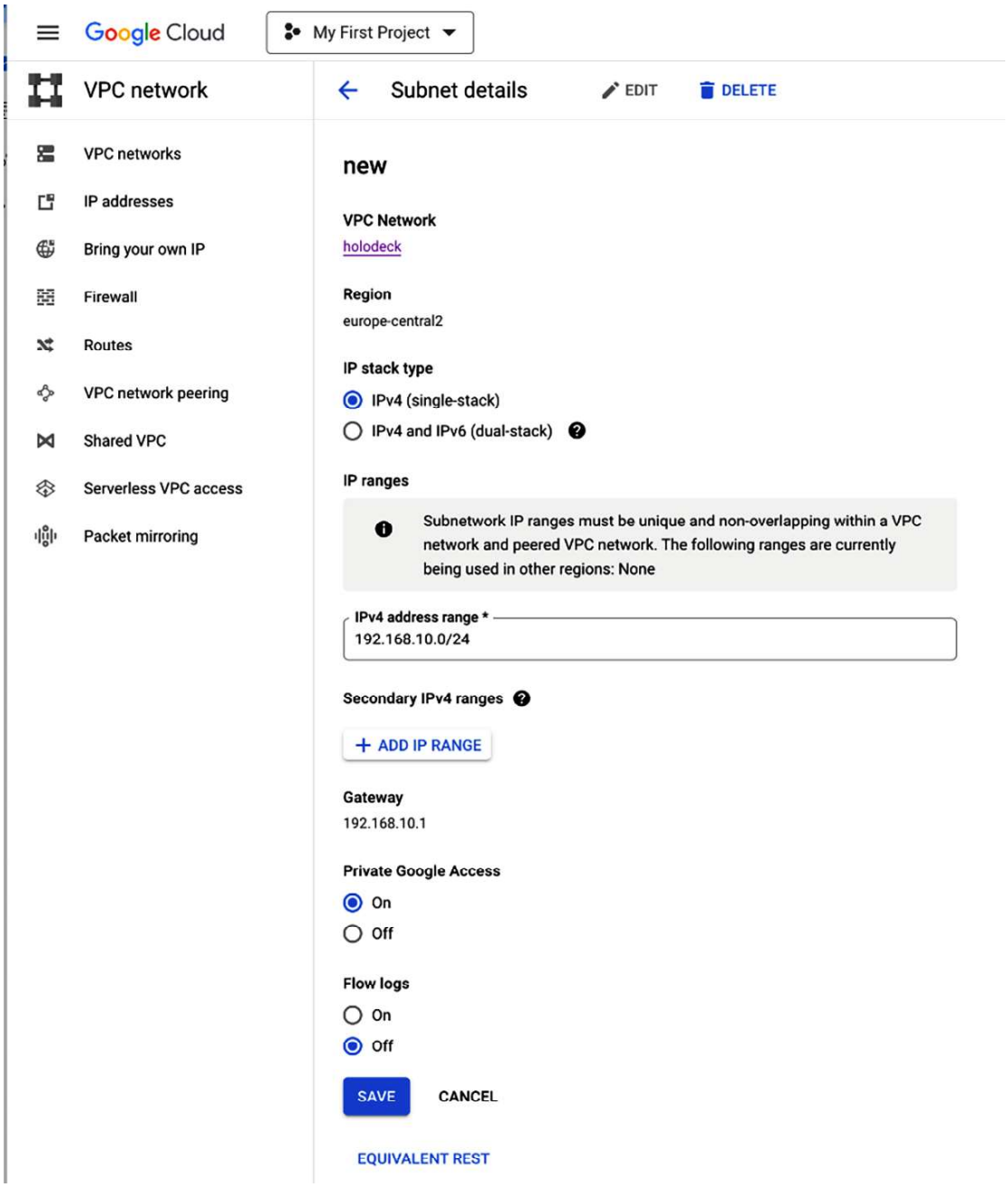

Figure 8.6 – Configuring subnet for Private Google Access

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.

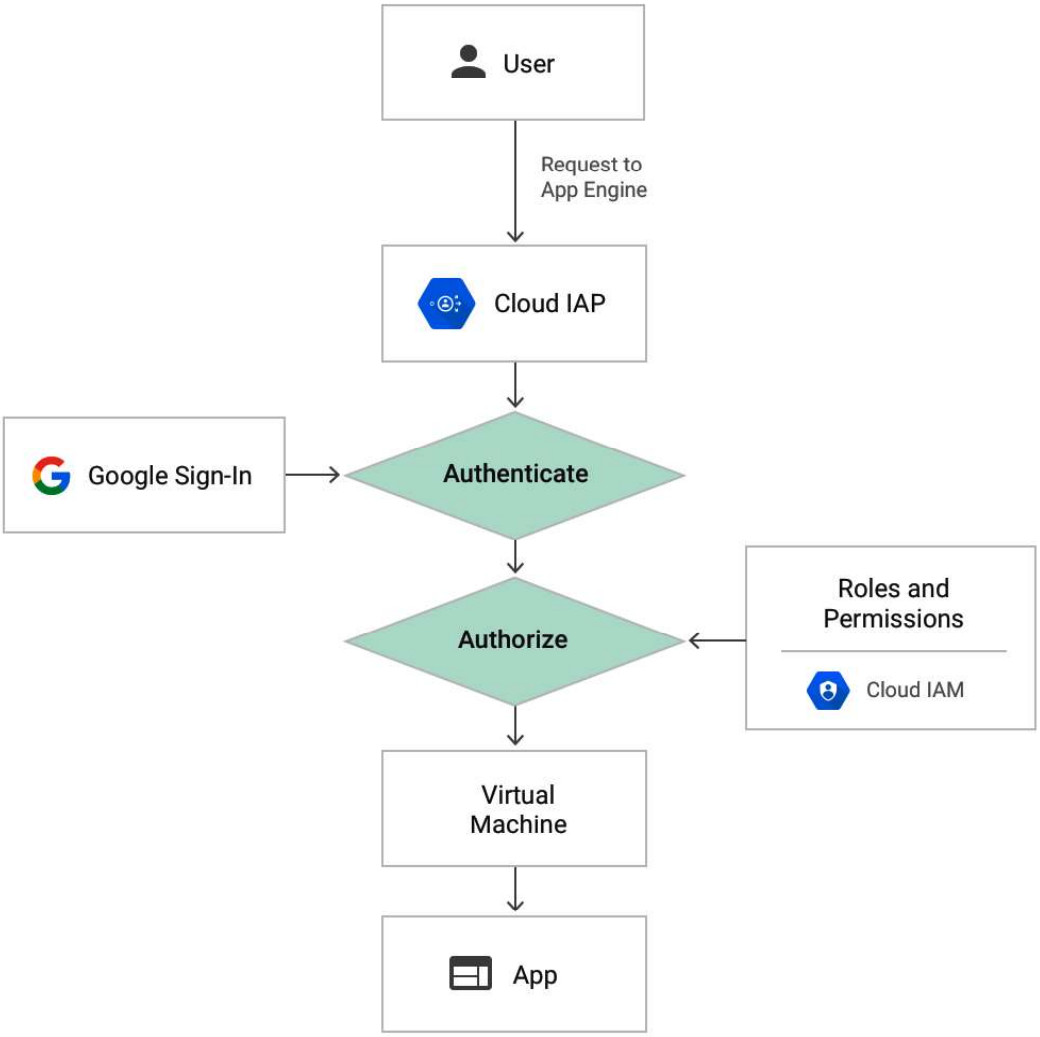

Figure 8.7 – How IAP works with App Engine

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.

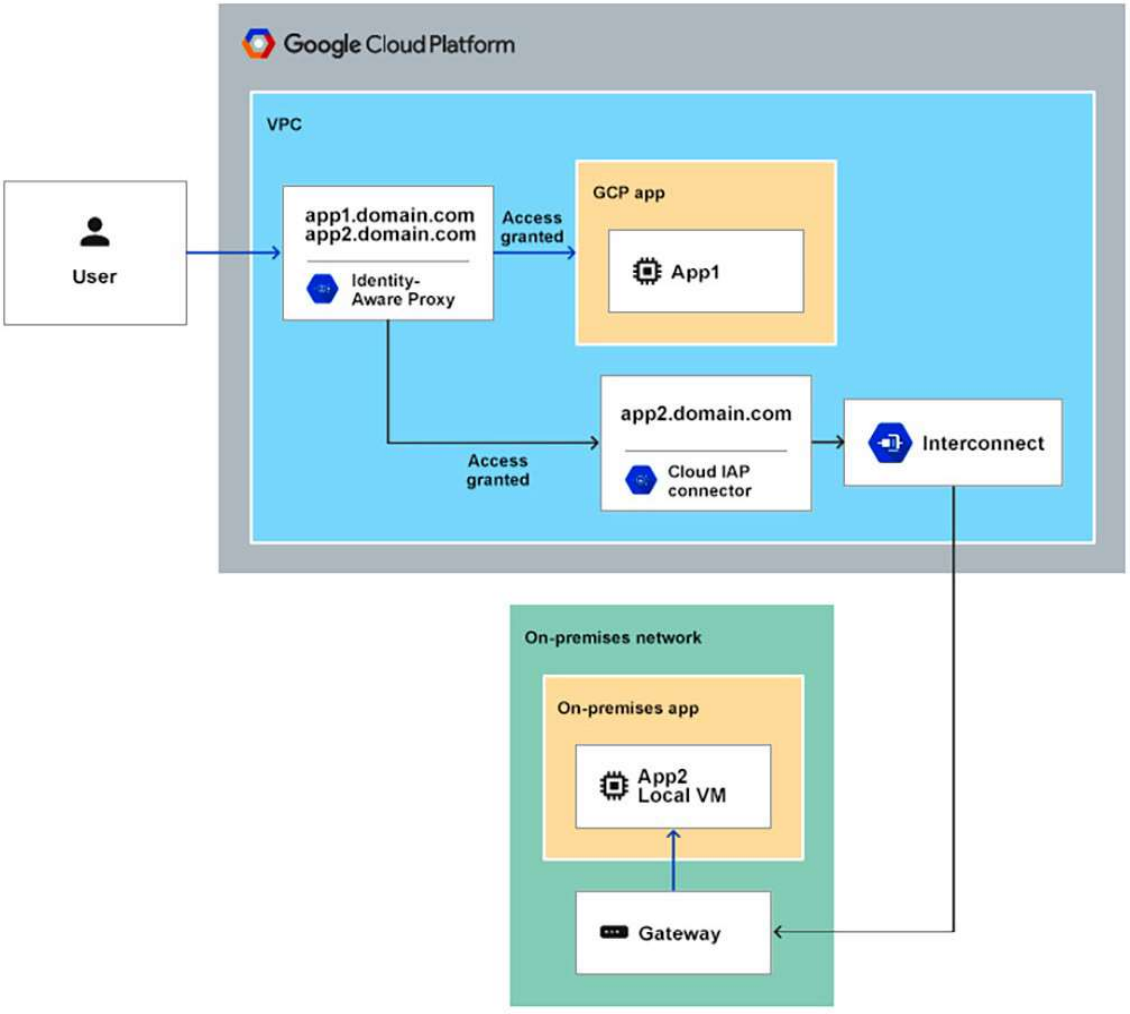

Figure 8.8 – How IAP for on-premises works

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**.

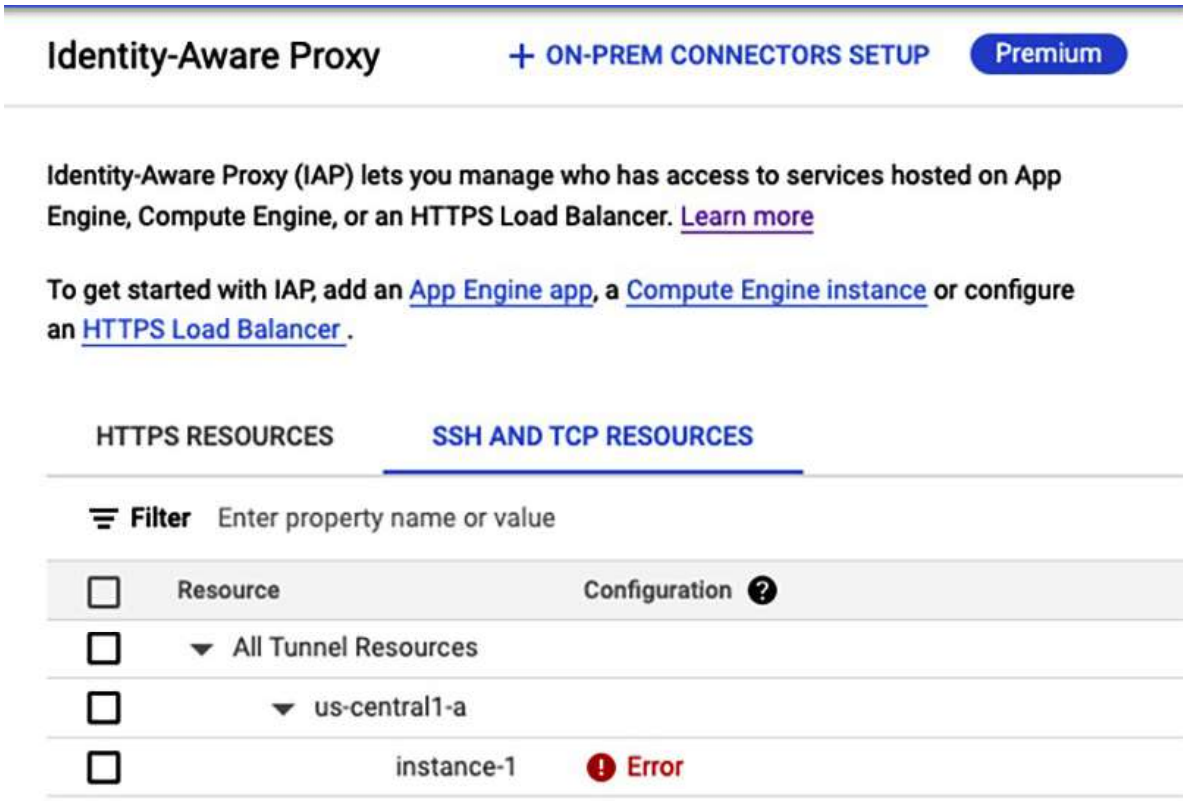

Figure 8.9 – The Identity-Aware Proxy dashboard

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.

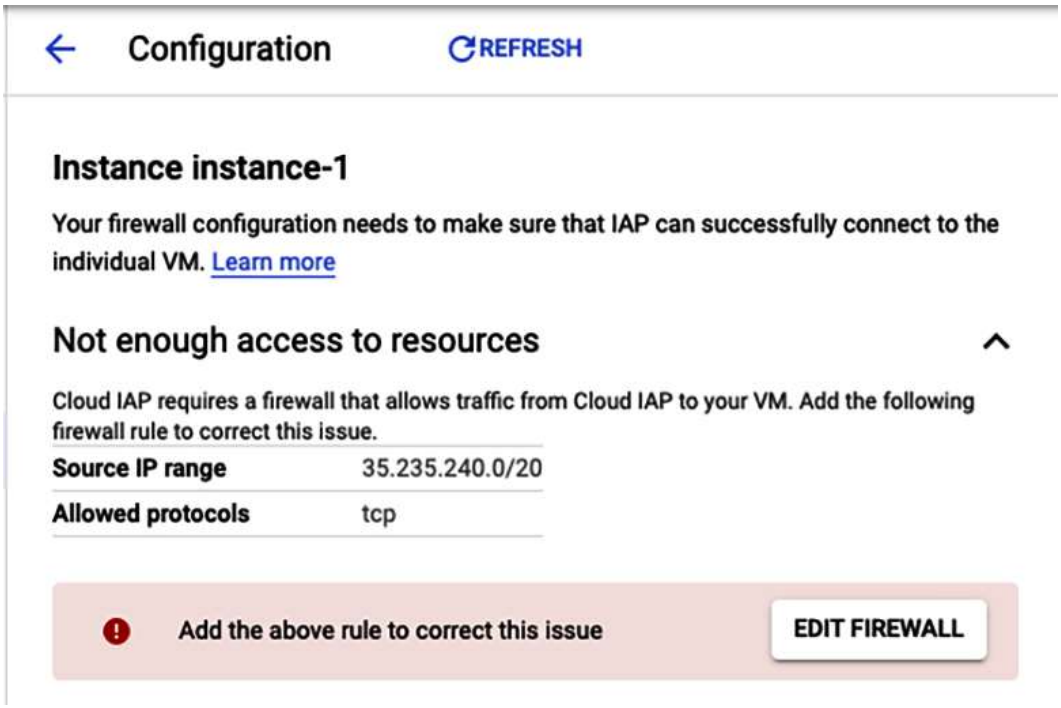

Figure 8.10 – Correcting a firewall configuration for IAP

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.

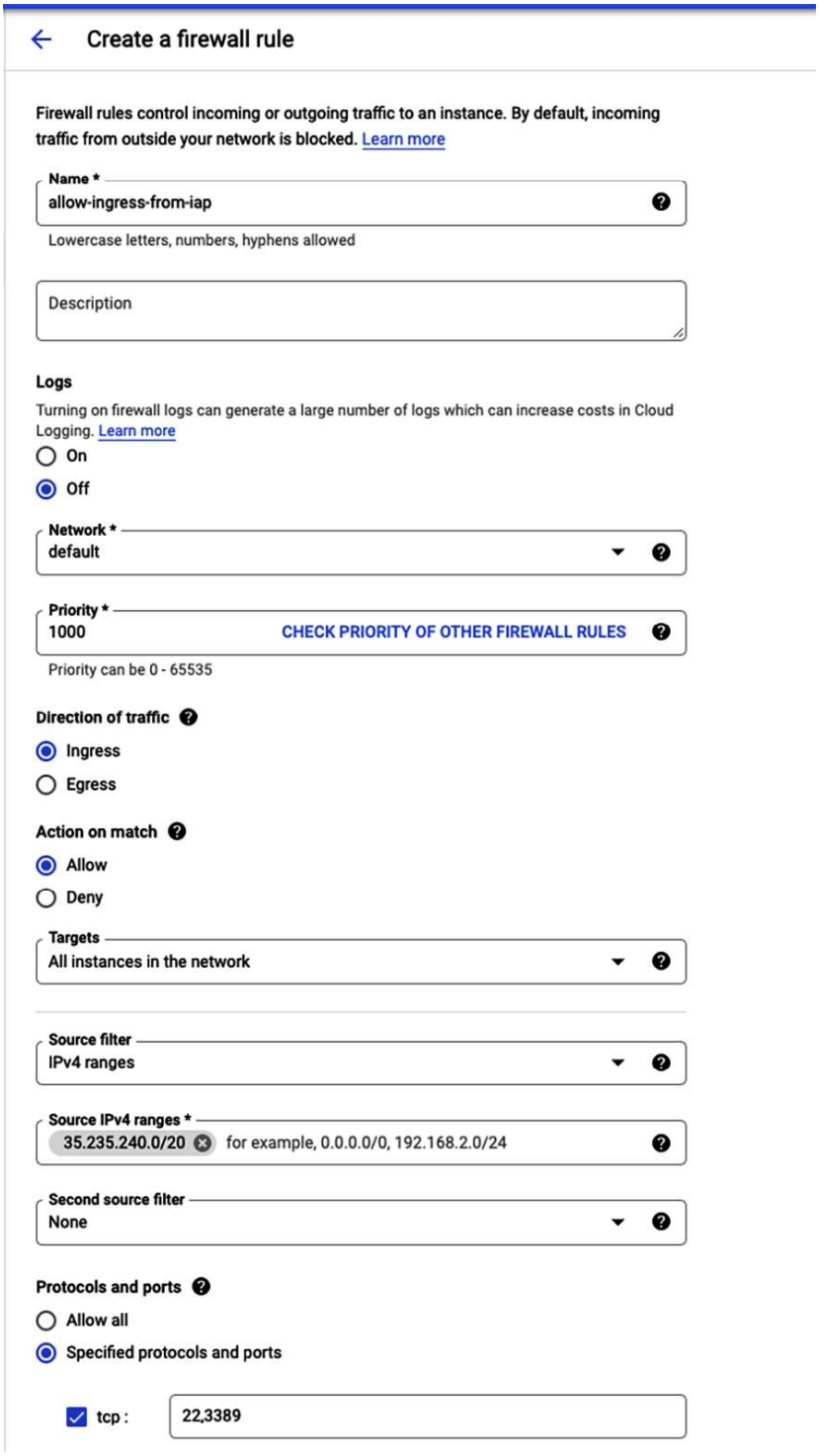

Figure 8.11 – Creating a firewall rule to enable IAP access

3. Click **Create**.

   Figure 8.12 shows how your configuration should look once you have populated the firewall rule settings as per the preceding example. You only specify `22` or `3380` if required under **Protocols and ports**. The example illustrates how you can open both SSH and RDP ports.

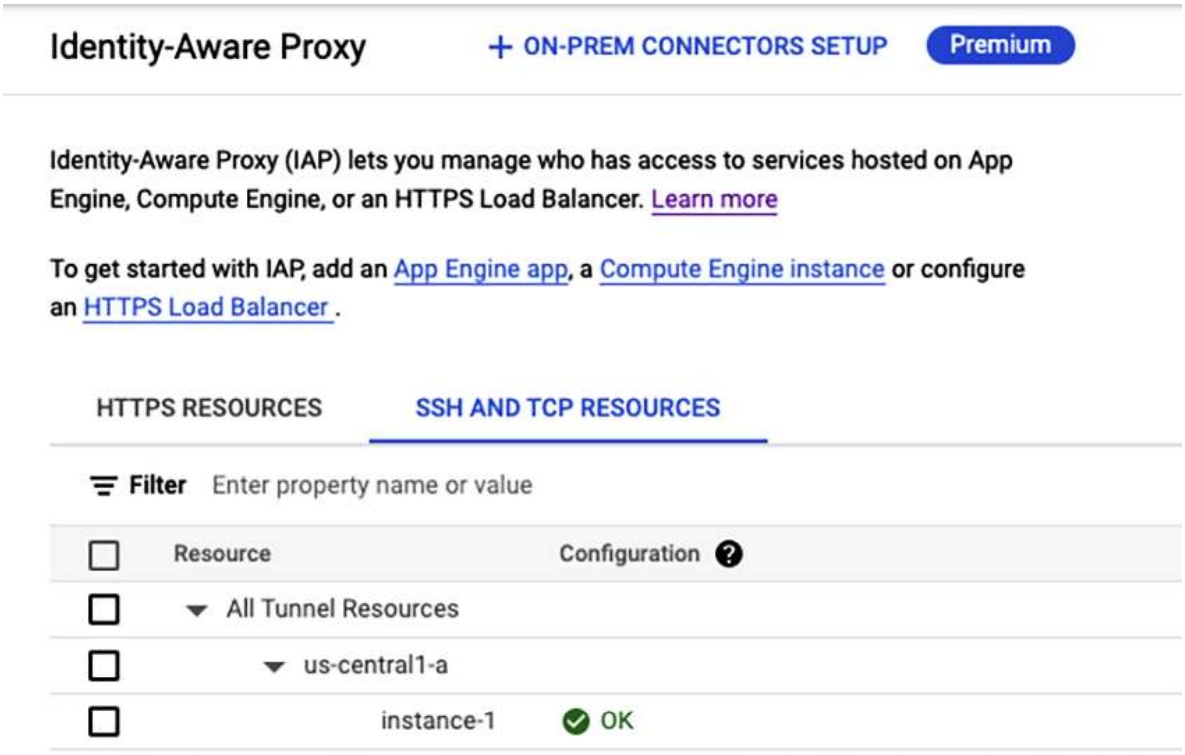

Figure 8.12 – SSH and TCP resources after creating the firewall rule successfully

In *Figure 8.12*, observe that after applying the firewall rule, when we navigate back to **SSH AND TPC RESOURCES** under the IAP settings, an `OK` message is presented. The `Error` message is no longer displayed.

Next, we need to give access permissions to the users. You can offer access to all VMs in a project – to a user or a group – by establishing IAM permissions at the project level. Let's take a look at what's involved:

4. Navigate to the IAP admin page from the left menu under the **Security** sub-menu.
5. From there, you can select the **SSH and TCP Resources** tab.
6. Then, you need to select the VM instances that you want to configure. You can click on **Show info panel** if the info panel is not visible. This panel comes up on the right-hand side of the console.

7. We will next look at how you can add a member and configure them. Refer to *Figure 8.13* on how to add a new member and apply the relevant role.
8. We add a new member that can either be an email address or a group email address that you have the option to configure in Google Cloud Identity.
9. Once you have added the user or group, you can set **Role** to `IAP-secured Tunnel User`.

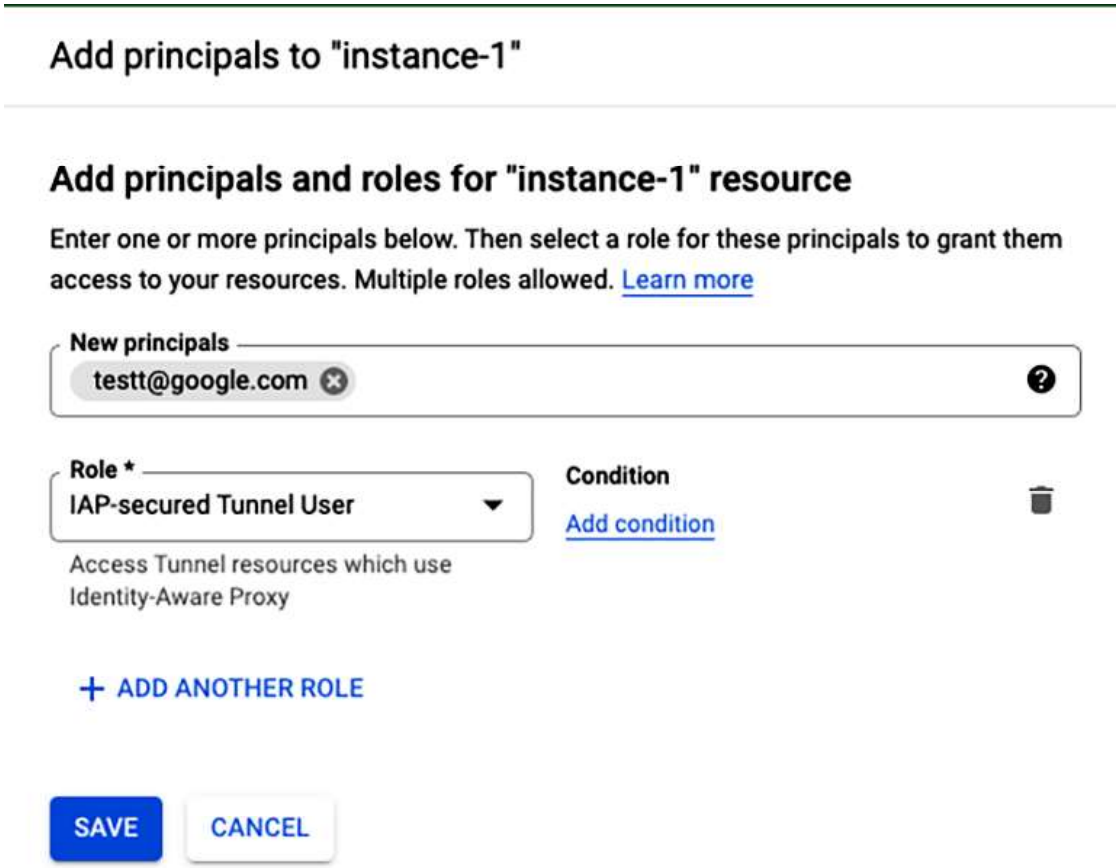

Figure 8.13 – Adding a principal and applying the IAP-secured Tunnel User role

You can also optionally create more fine-grained permissions by using the **Add condition** feature and configure a member restriction.

> **Note**
>
> You can refer to *Chapter 6*, *Google Cloud Identity and Access Management*, to learn more about how to add conditions to an IAM principal.

10. Once finished, you can click **SAVE**.

> **Note**
>
> In the preceding step, *Step 9*, we are referring to the access levels that can be created using Access Context Manager. Within Access Context Manager (which is a different product), you have the option of configuring policies based on access levels. For example, you can create an access policy for all privileged users called `PrivAccess`. Here you can specify certain parameters – you might want to restrict access to users only coming from a certain IP range or geo-location, for example. With the advanced features that you can get as part of Access Context Manager using the Beyond Corp Enterprise solution, you can also configure a policy that can do endpoint checks, such as checking for X.509 certifications, ensuring the operating system is approved and patched, hard disk encryption is enforced, and so on. This is part of Access Context Manager and is outside the scope of the Google Cloud Professional Security Engineer exam.

TCP forwarding requires different permissions based on how the user plans to utilize it. On the other hand, the user would need the following rights in order to connect to an unauthenticated VM using `gcloud compute ssh`:

- `iap.tunnelInstances.accessViaIAP`
- `compute.instances.get`
- `compute.instances.list`
- `compute.projects.get`
- `compute.instances.setMetadata`
- `compute.projects.setCommonInstanceMetadata`
- `compute.globalOperations.get`
- `iam.serviceAccounts.actAs`

If you don't have an external IP address, you can connect to Linux instances through IAP. The `gcloud compute ssh` command can be used to establish a secure connection to your instance. For TCP tunneling to work, your instance's access settings (specified by IAM permissions) must permit it:

```
gcloud compute ssh instance-1
```

IAP TCP tunneling is used automatically if an external IP address is not provided by the instance. If the instance has an external IP address, IAP TCP tunneling is bypassed in favor of the external IP address. You can use the `--tunnel-through-iap` flag so that `gcloud compute ssh` always uses IAP TCP tunneling.

RDP traffic can be tunneled using IAP to connect to Windows instances that do not have an external IP address.

In order to connect to a VM's remote desktop, you can use IAP TCP forwarding with IAP Desktop:

1. In the application, select **File | Add Google Cloud project**.
2. Enter the ID or name of your project and click **OK**.
3. In the **Project Explorer** window, right-click the VM instance you want to connect to and select **Connect**.

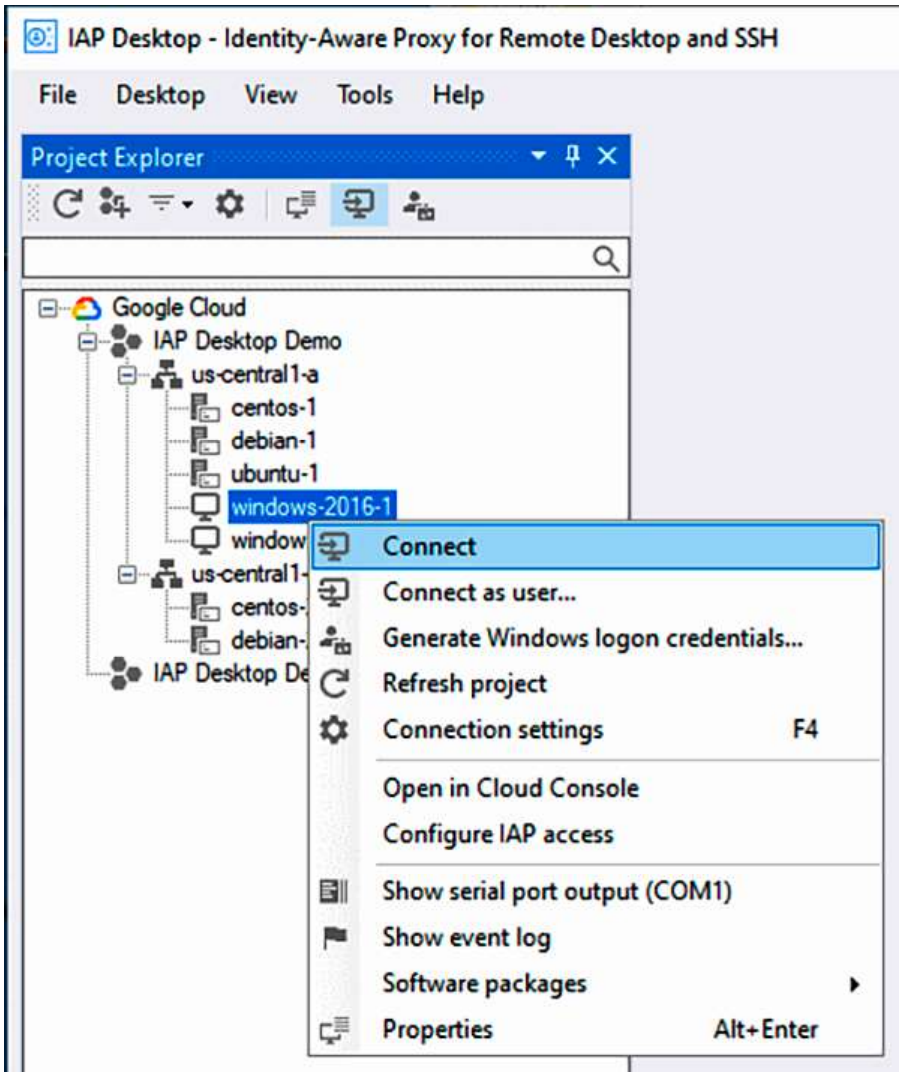

Figure 8.14 – Configuring IAP Desktop for IAP

Figure 8.14 is a screenshot of the IAP Desktop tool, which can be used to establish RDP connections for your Windows instances.

> **Note**
>
> You can find more information about IAP Desktop from its GitHub page at `https://packt.link/lZh5q`.

We will next move on to the topic of Cloud NAT. Here we will look at how NAT is used with some examples and gain a better understanding of the topic.

Cloud NAT

Cloud NAT is a topic that does not appear a lot in the exam. However, it is important to know how it works and the use cases for why you would need NAT. We will also look at the Google Cloud implementation of NAT architecture, which is different from the traditional NAT architecture.

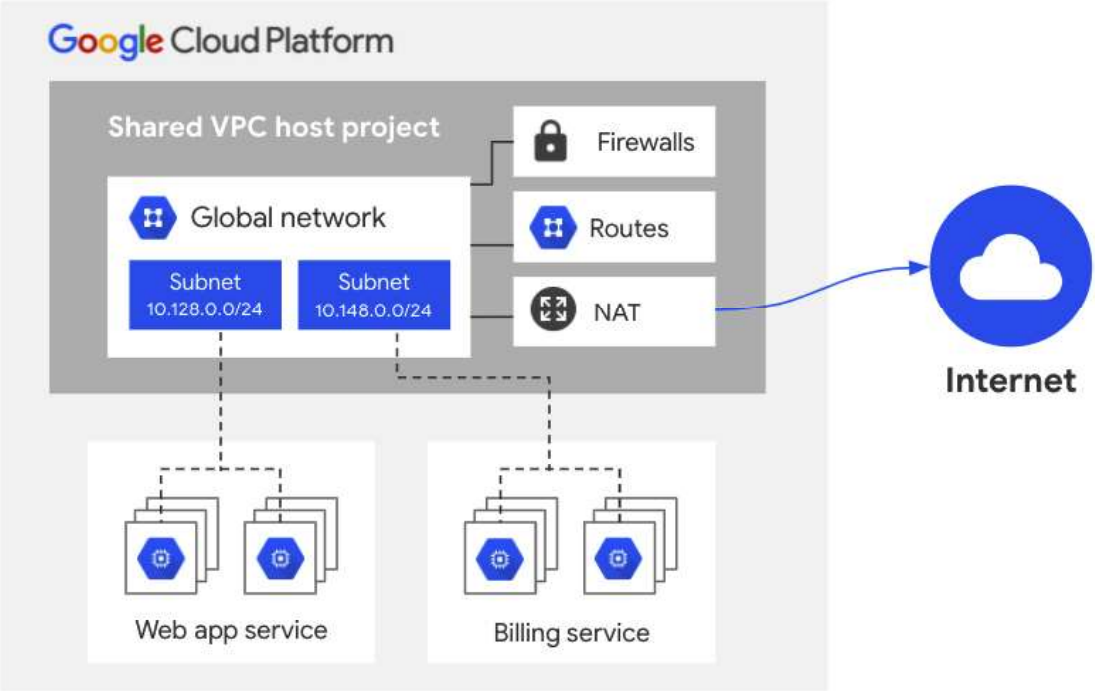

Figure 8.15 – Cloud NAT allowing outbound connections only to the internet

Figure 8.15 shows how Google Cloud NAT works. Cloud NAT is offered as a managed service that provides high availability and seamless scalability. It allows outbound connections only to the internet, whereas inbound traffic is allowed only if it is in response to a connection initiated by an instance. Cloud NAT is a regional resource, fully distributed and software-defined. There are no intermediate NAT proxies in the data path. NAT configuration is stored in the control plane and is pushed to the hosts; this means NAT keeps working regardless of the control plane state, and there are no choke points or performance penalties. Cloud NAT also supports alias IPs on VMs and relies on the default internet route in VPC.

It is important to note that there are cases when NAT is not performed even when configured. They are as follows:

- The VM has an external IP.
- The default internet route has changed.
- There's communication between backend VMs and the load balancer proxy.
- There's communication with Google APIs (Private Google Access is used instead).

Next, we will look at how you can create a Cloud NAT instance for a GCE instance that does not have an external IP address and wants to communicate with the internet. There are eight key steps involved in the configuration:

1. First, create a VPC network and a subnet as covered in *Chapter 7*, *Virtual Private Cloud*.
2. Then, create a VM instance without an external IP address. It is important to select the **No External Interface** option under the **Networking** tab of the GCE instance creation wizard.
3. Create an ingress firewall rule to allow SSH connections. This is similar to how we created a firewall rule to allow IAP in the IAP section. Ensure that the source IP range for IAP is specified, that is, the range `35.235.240.0/20`.
4. Navigate to the **Identity-Aware Proxy** settings and under **SSH & TPC RESOURCES**, select **Add Member** and give the `IAP-secured Tunnel User` permission. Again, this step is similar to what we configured in the *Using Cloud IAP for TCP forwarding* section.
5. In this step, we will navigate to the GCE page and log in via SSH from the console to the GCE instance that we created in *Step 2*. Once you have successfully logged in to the GCE instance via SSH, from the **command-line interface** (**CLI**), issue the `curl example.com` command. This should not result in any output.

6. Next, we will create a NAT configuration using Cloud Router. The NAT configuration is very straightforward. Once you have logged in to your Google Cloud console and the project is selected, you can go to the **Cloud NAT** page (**Networking | Network Services | Cloud NAT**). Once there, click on **Get started** or **Create NAT gateway**. You may see a welcome message if you are accessing this page for the first time. If you have already visited the page in the past, you will not see the welcome message; do not be confused if you don't see the following message. You should see a screen similar to the one shown in *Figure 8.16*.

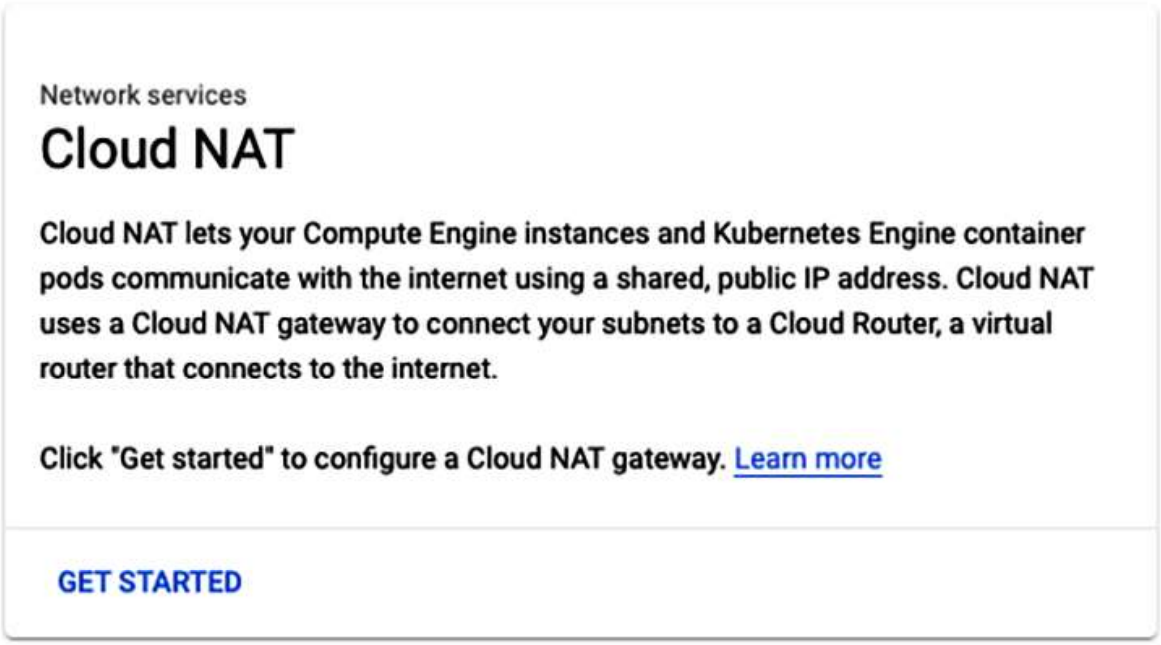

Figure 8.16 – Getting started with Cloud NAT

7. Refer to *Figure 8.17*, where you see the various properties you can configure when you create a Cloud NAT gateway. Here enter the name for your gateway; for this example, we are using the name `nat-config`. Then, specify your network; for this example, we have set **VPC network** to `startshipnw`. In your configuration, it will be the network that you have created, or you can even use the default network. Next, set **Region** to `us-east1 (your region)`. For **Cloud Router**, select `Create new router`. Then, enter the name of your preference; in our example, we have used the name `my-router`. Once finished, click **CREATE**.

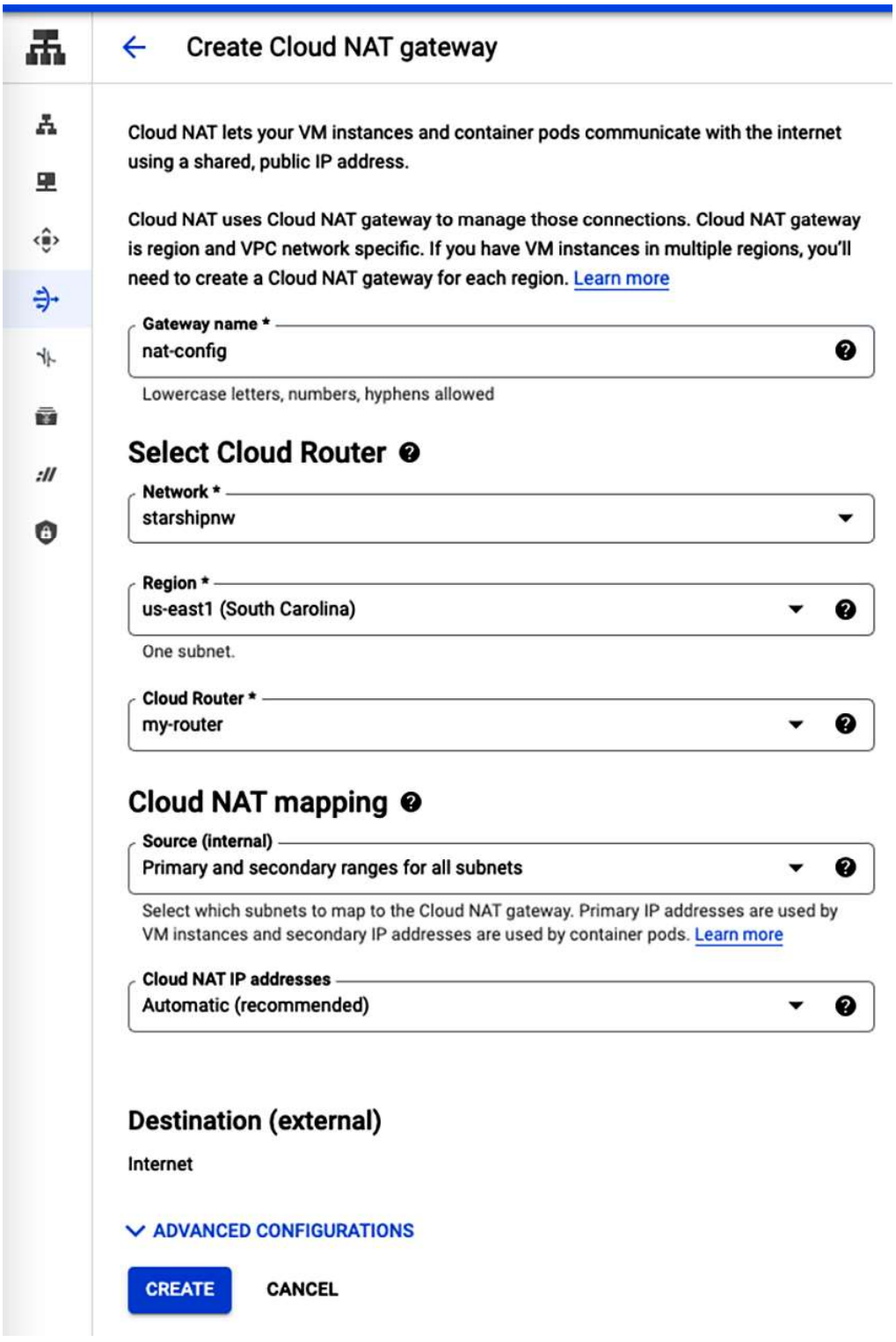

Figure 8.17 – Configuration to create a Cloud NAT gateway

8. Once the steps are completed, we can now go back to our instance under GCE, repeat the test, and issue the `curl example.com` command. This time, we will see an output similar to the following:

```
<html>


<head>

<title>Example Domain</title>

…

…

…

</head>

<body>

<div>

<h1>Example Domain</h1>
<p>This domain is established to be used for illustrative
examples in documents. You can use this domain in examples
without prior coordination or asking for permission.</p>

<p><a href="http://www.iana.org/domains/example">More
information...</a></p></div>

</body>

</html>
```

This concludes the process to create a Cloud NAT instance, which will now allow your VM that was configured with no internet access to access the internet. We will now move on to learn about Cloud Armor, which provides DDoS mitigation and a WAF function.

Google Cloud Armor

This section covers DDoS protection and the use of WAFs to provide safety for your web-based infrastructure. You can protect your Google Cloud workloads from a wide range of threats, including DDoS attacks and application attacks, such as XSS and SQL injection, with Cloud Armor (SQLi). Some capabilities are built in to provide automated protection, while others require manual configuration. We will look at those capabilities of WAFs in more detail in this section:

Figure 8.18 – How Google Cloud Armor secures your infrastructure

Cloud Armor leverages Google's global and distributed infrastructure to detect and absorb attacks and filter traffic through configurable security policies at the edge. It should be kept in mind that several aspects of Google Cloud Armor are only available for applications running behind an external HTTP(S) load balancer. *Figure 8.18* illustrates the placement of Cloud Armor, which is in line with the external load balancer, providing the ability to create IP allow and deny lists, enforce geo-location-based rules, and configure the WAF and custom rules from Layer 3 to Layer 7.

There are some Layer 3 and Layer 4 protections built into Cloud Armor that do not require any user configuration. These pre-built rules provide protection against volumetric attacks such as DNS amplification, SYN floods, Slowloris, and so on.

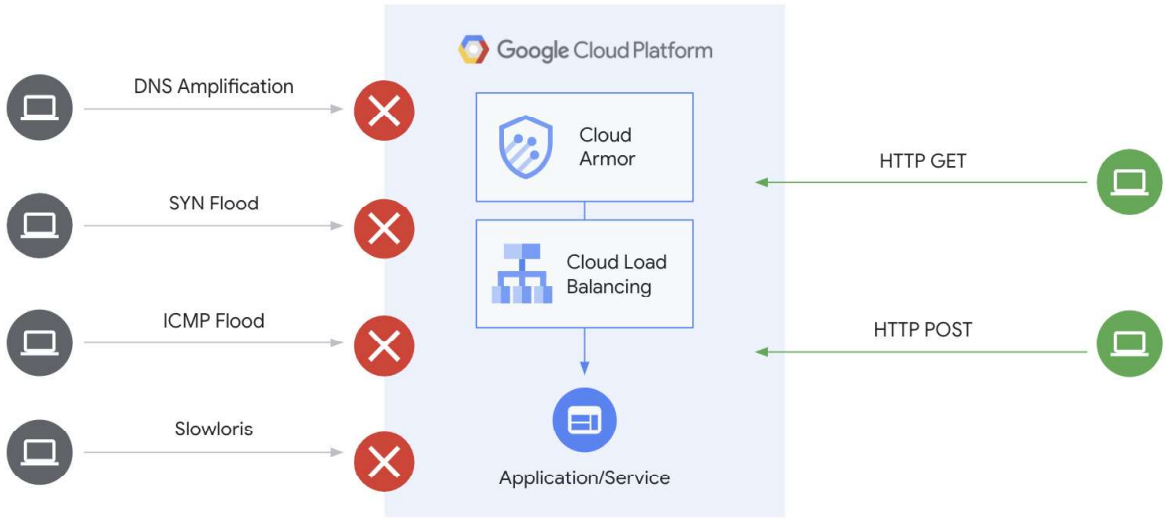

Figure 8.19 – DDoS protection against volumetric attacks

Figure 8.19 illustrates the built-in security policies for Cloud Armor that provide protection against these common types of volumetric attacks.

Next, we will look at how Cloud Armor works and some deployment models that will help you understand the use cases. As mentioned earlier, Google Cloud Armor is always in line and scales to match Google's global network. It protects applications and services behind external HTTP(S) load balancers, SSL proxy load balancers, or TCP proxy load balancers from volumetric DDoS attacks on the network or protocol. Because it can detect and counteract network threats, Cloud Armor's load balancing proxies will only allow properly formatted requests. Custom Layer 7 filtering policies, including preconfigured WAF rules to limit OWASP Top 10 web application vulnerability risks, are also available to backend services behind an external HTTP(S) load balancer. Google Cloud Edge can be used to define rules that enable or restrict access to your external HTTP(S) load balancer, as near as is feasible to the source of incoming traffic, through the use of security policies. A firewall can be set up to block unwanted traffic from accessing or consuming resources in your private cloud.

Cloud Armor is available in two tiers, called **Managed Protection Standard** and **Managed Protection Plus**. The key difference between them is the DDoS response team and cost protection that Plus provides compared to the Standard edition. There is no difference in the capabilities of the product.

Let's take a look at the Cloud Armor deployment models before we deep dive into some technical aspects such as security policies and WAF rules.

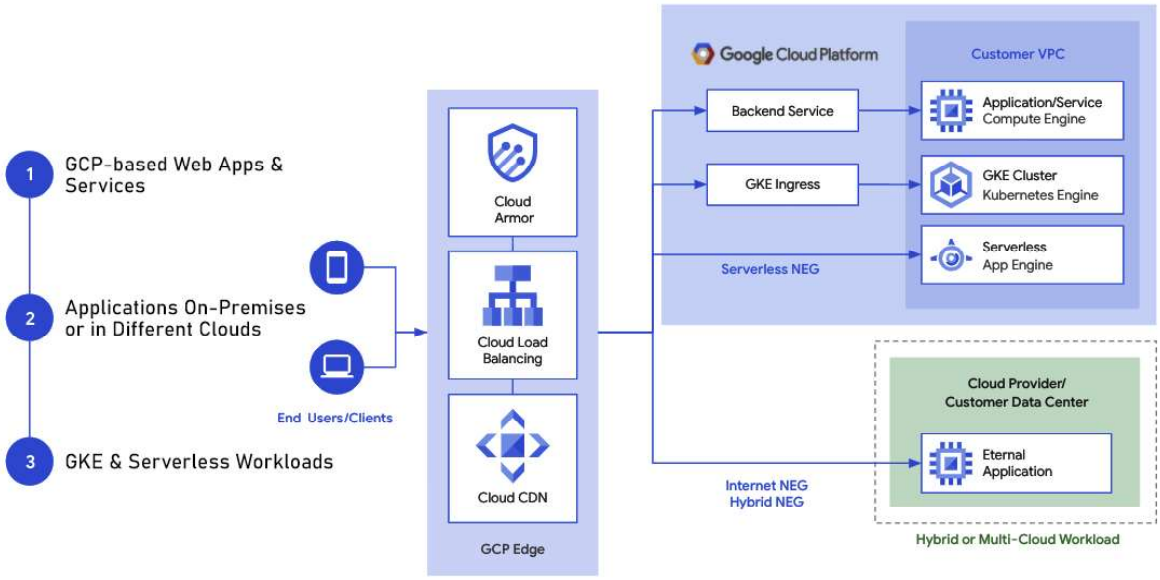

Figure 8.20 – Cloud Armor deployment models

We will use *Figure 8.20* to look at the deployment models:

- Cloud Armor supports the protection of Cloud **Content Delivery Network** (**CDN**) origin servers by enforcing Cloud Armor security policies on dynamic requests as well as cache misses destined for the CDN origin server. Most applications are often complex, serving both cacheable static content as well as dynamic requests from the same services. Cloud Armor is a security service that can help protect websites and applications from malicious attacks. It is able to inspect and filter requests that are sent to the origin servers of CDN. This helps to prevent so-called *cache-busting* attacks, which are attempts to bypass the cache and force the origin servers to send updated content. Additionally, Cloud Armor can help protect dynamic portions of websites and applications from the 10 most common security vulnerabilities as defined by OWASP.
- There is often a need to enforce a consistent set of security controls to applications no matter where they are deployed, whether they are migrating to Google Cloud or running in a hybrid configuration. With internet **Network Endpoint Groups** (**NEGs**), you can leverage all of Google's edge infrastructure, including global load balancers, Cloud CDN, and Cloud Armor, to protect your website or applications no matter where they are hosted. Cloud Armor can help protect your applications, whether from DdoS attacks or other common web attacks, without you having to deploy your applications on Google Cloud.

> **Note**
>
> NEGs do not fall within the scope of the exam. You can refer to the *Further reading* section to read more about NEGs and how to configure them.

- With GKE Ingress support for Cloud Armor, you can help protect your containerized workloads by placing them behind Google's global load balancers and configuring Cloud Armor security policies for Layer 7 filtering and WAF use cases.

We will now look at the key components of Cloud Armor in more detail, with examples of how to configure them.

Security policies

In Cloud Armor, you can build security policies that can help you defend applications operating behind a load balancer against DDoS and other web-based assaults, regardless of whether the apps are deployed on Google Cloud or in a hybrid or multi-cloud architecture. It is possible to stop traffic before it reaches your load-balanced backend services or backend buckets using security policies that filter Layer 3 and Layer 7 requests for common web exploits. An IP address, an IP range, a region code, and request headers are only a few examples of the conditions used to filter traffic by security policies.

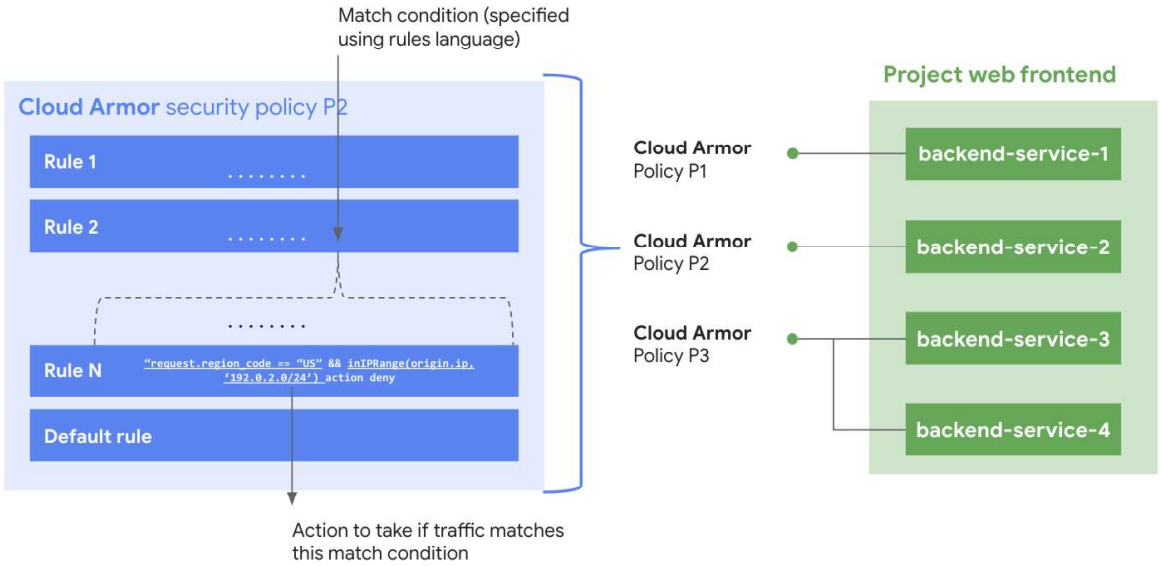

Figure 8.21 – Cloud Armor security policies

This section will take a closer look at security policies, looking at factors such as the requirements before you develop a policy and some of security policies' most important features (as depicted in *Figure 8.21*). Backend services behind an external load balancer are only able to access the security policies available to those services. Instance groups, serverless NEGs, internet NEGs for external services, and Google Cloud Storage buckets are all supported backend services for Google App Engine, Cloud Run, and Cloud Functions. You must employ internet NEGs for all of your hybrid or multi-cloud protection needs.

Security policies cannot be applied until the load balancer is an external HTTP(S) load balancer and the backend service's external load balancing scheme and the backend service's protocol are met – that is, the protocol must be either HTTP, HTTPS, or HTTP/2.

There are two categories of security policies that you can configure **– backend security policies** and **edge security policies**. Instance groups or NEGs, including internet, zonal, and serverless NEGs, can be used to filter and protect backend services. The actions associated with backend security policies are allow, deny, redirect (to Google reCAPTCHA), throttle, and rate-based ban for each source IP range or geography. You can also create WAF rules and named IP lists, including Layer 7 filtering and adaptive protection (adaptive protection is a new feature and currently out of scope for the exam).

For edge security policies, you can create and allow or deny based on source IP and source geography. These policies allow users to establish access control and filtering policies for cached information. When compared to backend security rules, edge security policies only allow for filtering based on a handful of criteria. A backend policy cannot be used to set an edge security policy.

At the edge of Google's network – upstream of the Cloud CDN cache – edge security measures are installed and implemented. Two levels of security can be provided by coexisting backend security policies and edge security policies. Regardless of the resources that a backend service refers to, they can all be applied simultaneously (for example, instance groups or network endpoint groups). In order to secure backend buckets, you must use edge security policies. You should keep in mind that edge security policies are examined and executed prior to IAP. An edge security policy blocks a request before the identity of the requestor is evaluated by IAP.

Rule evaluation order for a security policy is determined by rule priority. When determining which rules are examined first, look at the rule with the lowest numeric value. This rule has the highest logical priority and is evaluated first.

If none of the higher priority rules is met, or if no other rules are present, a default rule is applied to all security policies. A priority of `2147483647` is automatically assigned to the default rule, which is always included in the policy.

Although the default rule cannot be deleted, you can change the default action associated with the rule. It is possible to change the default rule's action from allow to deny.

Use the Google Cloud Armor custom rules language to define expressions in the match condition of one or more rules. When Cloud Armor receives a request, it analyzes it against the following expressions. Depending on whether the rules are met, inbound traffic is either denied or allowed. The following are some instances of Google Cloud Armor expressions written in the **Common Expression Language** (**CEL**):

- To define expressions in a rule, use the `gcloud --expression` flag or the Google Cloud console.
- In the following example, requests from `2001:db8::/32` in the `AU` region match the following expression:

```
origin.region_code == "AU" && inIpRange(origin.ip,
'2001:db8::/32')
```

- The following example matches with requests from `192.0.2.0/24` and with a user agent that contains the string `WordPress`:

```
inIpRange(origin.ip, '192.0.2.0/24') && has(request.
headers['user-agent']) && request.headers['user-agent'].
contains('WordPress')
```

WAF rules

In this section, we will look at the different types of rules that you can create. There are some WAF rules that come pre-configured, but you do have the option of creating custom rules to meet your requirements. Let's take a look at the options available.

You can create IP-address-based rules both at the edge and for your backend services. These rules are the IP address `allowlist-` and `denylist`-based rules. Both Ipv4 and Ipv6 are supported. HTTP `403` (Unauthorized), `404` (Access Denied), or `502` (Bad Gateway) responses are all possible outcomes of deny rules. An HTTP `429` error is an error code that is returned when an application has exceeded a set number of requests. This is usually caused by an action rule that limits the number of requests that can be made in a given period of time.

There are pre-configured rules based on the OWASP ModSecurity core rule set version 3.0. These rules can provide you with protection from the following types of attack:

- SQL injection
- **Cross-site scripting** (**XSS**) attacks
- **Local file inclusion** (**LFI**) attacks
- **Remote file inclusion** (**RFI**) attacks
- **Remote code execution** (**RCE**) attacks

Use rate-limiting rules to restrict requests per client based on the threshold you define, or temporarily ban clients who exceed the request threshold for the duration you set.

There are bot management rules as well (in preview only), which are out of the scope of the exam. Pre-configured rules also exist for named IP lists; we will cover them in more detail in the *Named IP lists* section.

> **Note**
>
> You can fine-tune the pre-configured WAF rules or write custom rules. For the purpose of the exam, you need to know what types of rules exist and how they work. You are not tested on writing custom WAF rules.

Named IP lists

Using the named IP list feature in Cloud Armor, you can reference third-party maintained IP ranges and IP addresses within a security policy. You don't have to specify individual IP addresses. Instead, you can reference an entire list containing multiple IP addresses that you would like to allow or deny. It is important to note that named IP lists are not security policies, but they can be specified in a policy.

It's possible to build security rules that allow all IP addresses that are in the provider list and reject all IP addresses that are not in the provider list: `ip1`, `ip2`, `ip3`....`ipN`:

```
gcloud beta compute security-policies rules create 1000 \
    --security-policy POLICY_NAME \
    --expression "evaluatePreconfiguredExpr('provider-a')" \
    --action "allow"
```

Custom named IP address lists are not possible to construct. Only third-party providers who work with Google and maintain named IP address lists can use this capability. CloudFlare and Imperva are a couple of the providers who partner with Google Cloud to supply named IP lists for Google's named IP list service.

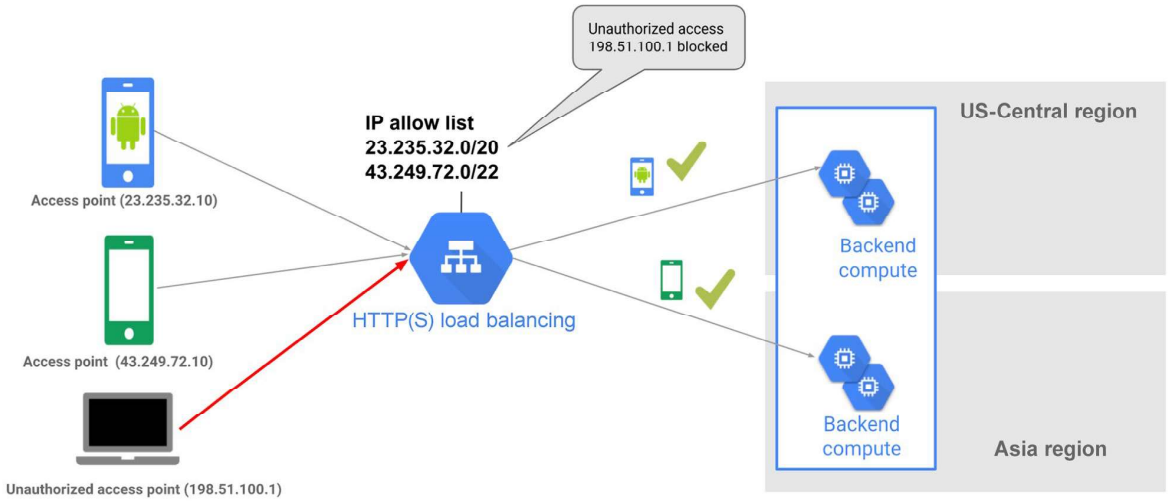

Figure 8.22 – Cloud Armor named IP list

In *Figure 8.22*, you can see how named IP lists work. CDN provides a list of IP addresses, such as `23.235.32.0/20`, `43.249.72.0/22`, and so on. Only communication coming from these IP addresses is permitted by this security rule. This means that the traffic of two CDN providers can be accessed (the access points are `23.235 32 10` and `43.249 72 10`), while unapproved traffic from `198.51.100.1` is banned.

Summary

In this chapter, we looked at some advanced network security concepts. We covered the usage and configuration of Private Google Access, IAP and its use cases, and Cloud NAT. Finally, we looked at Cloud Armor and how it provides protection against DDoS and web-application-based attacks. We also covered additional details related to Cloud Armor, such as security policies, WAF rules, and named IP lists.

In the next chapter, we will cover data security, which is an important topic for the exam. We will look at the Google Cloud Key Management system and then cover data loss prevention and Secret Manager in the subsequent chapters.