8

Managing Cloud-Native Workloads with Anthos

In the previous chapters, we took a deep dive into **Google Kubernetes Engine** (**GKE**) and Cloud Run, which provide orchestration for cloud-native workloads. In this chapter, we will look at **Anthos**; a hybrid multi-cloud platform that allows you to run your cloud-native workloads in a consistent and controlled way in any environment, whether in the public cloud, on-premises, or even at the edge.

It is important to understand that Anthos is developed in a very agile way. A new version is released every 3 months. Therefore, we will not make reference in this chapter to any particular version of Anthos. Instead, we will explain to you the main concepts and make reference to the most up-to-date documentation so you can stay up to date and be ready for the exam.

> **Exam tip**
>
> As Anthos is a very hot topic, it is very likely to appear in the PCA exam. Note that Anthos is a very wide topic and deservers a book of its own, however, this chapter will give you a good introduction to help you pass the exam.
>
> For the PCA exam, be prepared with a high-level understanding of all the Anthos services described in this chapter. Even though you should not expect deep-dive questions on how Anthos works and how to configure the services involved, we encourage you to try out Anthos Qwiklabs to better understand it.

We will cover the following topics in the chapter:

- What is Anthos?
- Anthos cluster deployment options
- Anthos Config Management
- Anthos Service Mesh
- Anthos Binary Authorization
- Migrate for Anthos and GKE
- Cloud Run for Anthos

Anthos components

Anthos is not a single service but rather a set of services that are combined into a platform. It allows you to manage and observe your cloud-native workloads wherever they are deployed. As we can see in *Figure 8.1*, Anthos consists of a set of services that provide the following:

- Infrastructure management
- Cluster management
- Service management
- Policy management
- Configuration management:

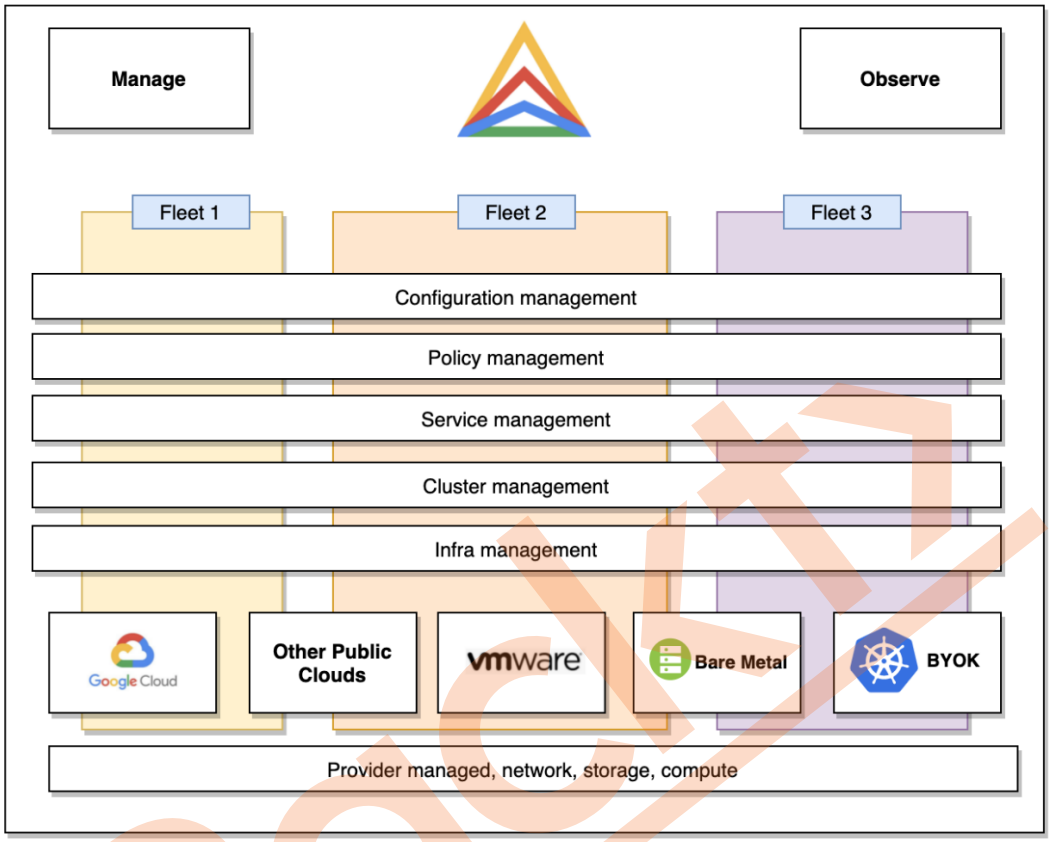

Figure 8.1 – Anthos components

Anthos allows you to group your Kubernetes resources into fleets/environs to manage them as a single entity. We will have a closer look at fleets later in this chapter.

Anthos clusters

Anthos clusters allow you to work with multiple Kubernetes clusters and provide consistent and secure environments for your cloud-native workloads. Anthos can deploy Anthos clusters in **Google Cloud Platform** (**GCP**) using GKE, on-premises, and other public clouds. You can also attach already-existing Kubernetes clusters and take advantage of some Anthos features without having to provision a new Anthos cluster and migrate your workloads.

Anthos Connect Agent

Anthos Connect Agent can connect any Kubernetes cluster to GCP. By connecting a cluster to GKE Hub, you are able to access the cluster directly from Cloud Console and interact with the hosted workloads. The agent runs as a Kubernetes Deployment called Connect Agent and establishes an encrypted connection with the GKE hub:

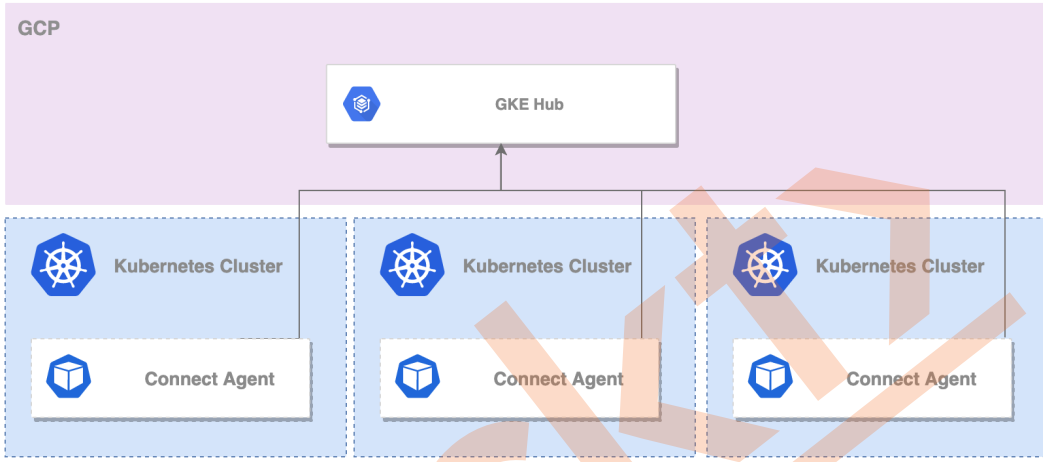

Figure 8.2 – Anthos Connect agents

For scenarios where the cluster cannot access the internet directly, the agent can be configured to use NAT or proxies for the egress to the GKE hub. Once you have the cluster registered in the GKE hub, you get a GKE-like experience of interacting with your Kubernetes cluster.

Fleets (formerly environs)

Fleets (formerly called environs) allow you to logically organize **fleet-aware resources** such as Kubernetes clusters to allow easier administration. As an example, you can apply unified configuration or policies on your resources to achieve consistency within your fleet.

Fleets have the following constraint: A single fleet can be associated with a single GCP project called the **fleet host project** and the fleet project can have only one fleet. This constraint results in isolation of resources on the project level.

Fleets can be used to work with **fleet-enabled components** such as the following:

- **Workload identity pools** – Allows you to uniformly authorize workloads in the pool to external services
- **Anthos Config Management** – Allows you to uniformly apply configuration and enforce policies on clusters and resources in the fleet
- **Anthos Service Mesh** – Allows you to create a service mesh across the resources in the fleet
- **Multi Cluster Ingress** (**MCI**) – Allows you to create a common ingress for the clusters and service endpoints in the fleet to provide load balancing

Another important element of fleets is the concept of sameness. An object in different contexts but with the same name can be treated as the same entity. This includes the following contexts:

- Namespace sameness
- Service sameness
- Identity sameness when accessing external resources
- Identity sameness within a fleet

As an example, for namespace sameness, namespaces with the same name across different clusters in the same fleet are treated as a single entity.

Anthos cluster deployment options

Anthos comes with multiple deployment options allowing you to host cloud-native workloads almost anywhere. As was already said, Anthos is a very fast-developing platform. Therefore, you should always consult the documentation to find the most recent list of supported endpoints and features, at `https://cloud.google.com/anthos/deployment-options`. The following diagram shows the deployment options available and planned at the time of writing this book:

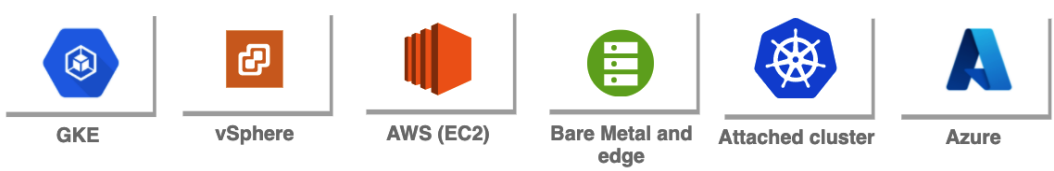

Figure 8.3 – Anthos deployment options

Also note that you can attach Kubernetes clusters that are provisioned using a managed Kubernetes service in **Azure Kubernetes Service** (**AKS**) and **Amazon Elastic Kubernetes Service** (**EKS**). If you inspect the preceding link you may notice that new Anthos features are always introduced to GKE (that is, on GCP) first. This is because GKE fits very well into the Anthos ecosystem and leverages GCP infrastructure. You can also expect that clusters that have been provisioned outside of Anthos and were later attached might have some limitations in terms of features due to technical constraints.

Now let's have a high-level look at each of the deployment options.

Note that in *Chapter 5*, *Managing Kubernetes Clusters with Google Kubernetes Engine*, we went through an extensive review of the GKE service. We encourage you to revisit the chapter if you would like to get more information on this deployment option. To enable Anthos for GKE follow the step-by-step guide you can find here: `https://cloud.google.com/anthos/docs/setup/cloud`.

Anthos clusters on VMware (GKE on-prem)

Anthos clusters on VMware (GKE on-prem) was the first deployment option that brought GKE to data centers on premises. It can run on your existing virtualization platform based on **VMware vSphere**. Anthos communicates with vSphere as an infrastructure management layer via the **vCenter** API to provision **virtual machines** (**VMs**) that host the Kubernetes nodes. This way we are able to provision both the control plane and the workload clusters for the GKE on-prem deployment.

As vSphere is one of the most mature and widely adopted virtualization platforms on the market, it has a very sophisticated set of features including software-defined networking, storage, and advanced APIs. It is also considered a state-of-the-art foundation for private clouds. As vSphere removes the management overhead by providing automation for software-defined infrastructure, it was a natural choice for Google to start with vSphere as the first endpoint for GKE on-prem. You must be aware though that this comes with some costs, as you need to purchase vSphere licenses on top of the regular licenses.

GKE on-prem components

In *Chapter 5*, *Managing Kubernetes Clusters with Google Kubernetes Engine*, we learned that the management plane for the GKE clusters resides in GCP and we as consumers do not need to worry about it. GCP takes care of the provisioning and management of both the control and data planes of GKE Kubernetes clusters.

When we move to on-premises or another cloud provider, the management plane resides outside of GCP. In most of the non-GCP Anthos clusters, the management plane is hosted on a so-called **admin cluster** as per *Figure 8.4*:

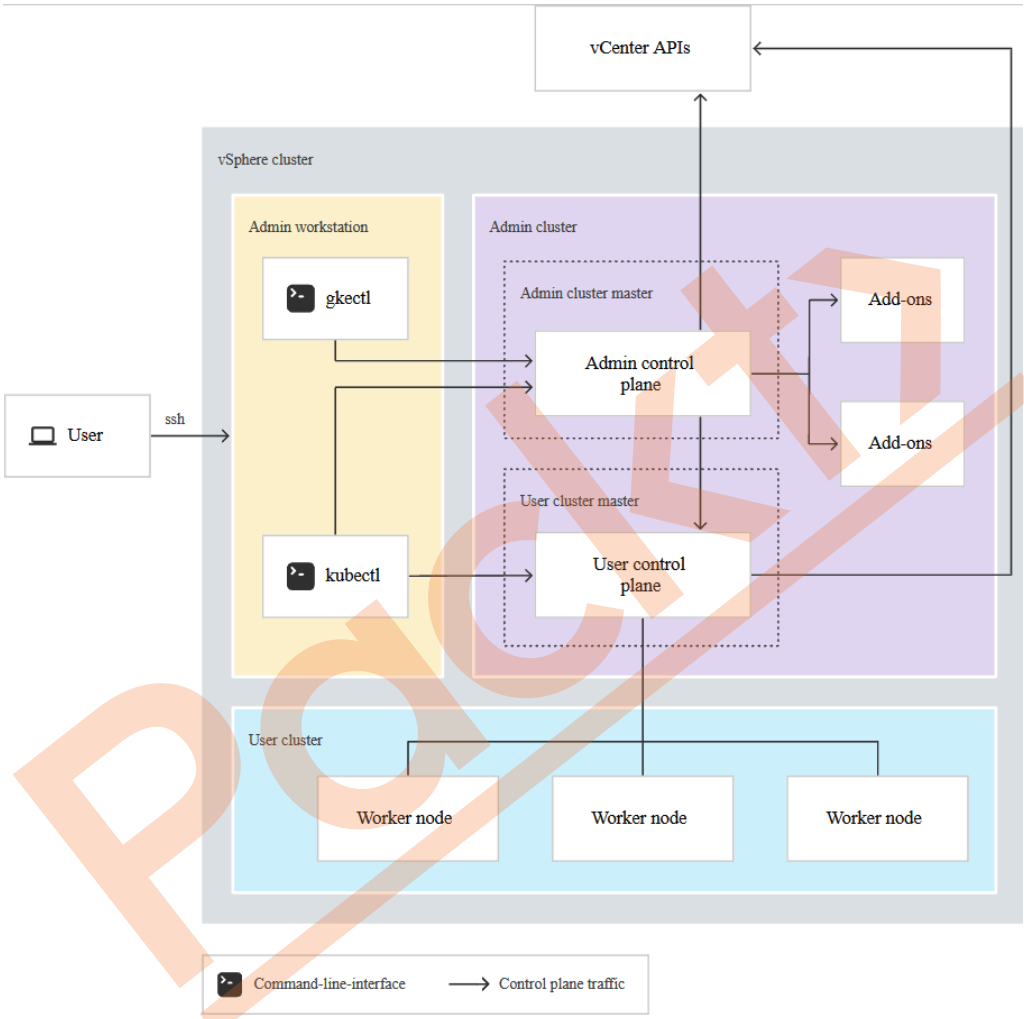

Figure 8.4 – GKE on-prem components (Source: `https://cloud.google.com/anthos/clusters/docs/on-prem/overview?hl=pl`)

Important Note

This architecture is constantly being refined by Google with some attempts to minimize or remove the admin cluster role. Therefore, be aware that this might change in the future.

An admin cluster is a Kubernetes cluster that is provisioned as the base layer of the Anthos cluster and runs the following components:

- **Admin cluster control plane**: This hosts the Kubernetes API, scheduler, and controllers for the admin cluster
- **User cluster control plane**: These are nodes of the user cluster control plane. They host the Kubernetes API, scheduler, and controllers for the user clusters
- **Add-ons**: Runs Kubernetes add-ons such as Google Cloud's operations suite

It is important to note that the master nodes of the **user clusters** (workload clusters) run in the admin cluster, not in the user cluster itself.

In *Figure 8.4* you can also see the **Admin workstation**, which is a Linux-based VM. This machine contains all the tools required to download, provision, and manage the GKE on-prem cluster.

Load balancing

Load balancing is needed for both the management plane of Anthos on-premises and also for exposing the Kubernetes services. There is a choice of three load balancing modes:

- **Integrated** – Uses F5 BIG-IP load balancers that need to be provisioned outside of the Anthos clusters. There is deep integration with F5 via the API so when services are deployed in the Anthos cluster, operators take care of creating objects in the F5 appliance without any manual interaction needed. Note that F5 licenses are not included in the cost of Anthos
- **Bundled** – The load balancer is provisioned during the Anthos deployment. It is based on the open source Google Seesaw load balancer project. To learn more about this project, visit `https://github.com/google/seesaw`
- **Manual** – Allows you to integrate with other third-party load balancers

The choice of the right load balancing mode depends on multiple factors including your network architecture, support for particular features, and cost limitations. You might need to consult your data center network architect before taking the decision on which load balancing mode to use.

Deployment of GKE on-prem clusters

As you can imagine, the deployment of an on-prem cluster is not as straightforward as a deployment of a GKE on Google Cloud. As of today, there is no way to deploy the GKE on-prem clusters from the Google Cloud Console directly. For GKE on-prem you will use command-line tools called `gkeadm` and `gkectl` that can be run directly from your admin workstation residing on premises.

The high-level steps of deploying GKE on-prem are as follows:

1. Deploy and/or prepare the vSphere infrastructure.
2. Prepare your Anthos GCP project and a service account.
3. Deploy the admin workstation and download the Anthos images.
4. Deploy the integrated, manual, or bounded load balancers for the admin cluster.
5. Deploy the admin cluster.
6. Deploy the integrated, manual, or bounded load balancers for the admin cluster.
7. Deploy the user cluster.
8. Configure additional services (such as Anthos Service Mesh, Config Management, and so on).

Note that this procedure can change, so for a step-by-step installation guide for GKE on-prem clusters, visit `https://cloud.google.com/anthos/clusters/docs/on-prem/overview?hl=pl`.

Anthos on AWS

Anthos clusters on AWS (GKE on AWS) allow you to extend your cloud-native platform to the AWS cloud. It leverages AWS-native services including the following:

- EC2 VMs
- **Elastic Block Storage** (**EBS**)
- **Elastic Load Balancer** (**ELB**)

This allows you to leverage the benefits of Anthos but also consume AWS services natively.

Anthos on AWS components

Anthos on AWS already follows the architectural decision to reduce the footprint or completely remove the admin cluster (see the *GKE on-prem components* section). Therefore, there are two components to Anthos clusters on AWS:

- **Management service**: This allows us to install, update, and manage the user clusters via the AWS API
- **User cluster**: Runs the user workloads

The following figure shows the architecture of Anthos on AWS:

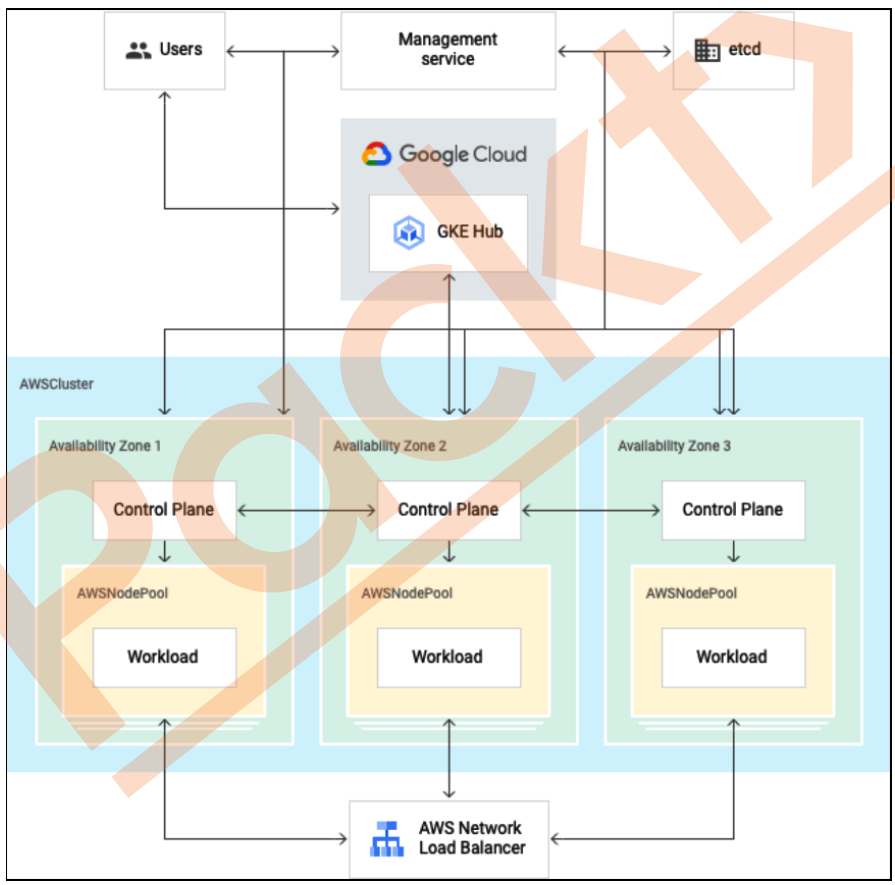

Figure 8.5 – Anthos on AWS (Source: `https://cloud.google.com/anthos/clusters/docs/aws/concepts/architecture?hl=pl`)

As we can see in the diagram, both the management service and the users cluster are located in AWS and are connected to the GKE hub located in GCP.

Load balancing

Anthos on AWS uses the AWS-native ELB so there is no need install or configure an additional load balancer. This eliminates the need for additional tools to manage third-party products and gives you the best-in-class load balancing option deeply integrated with the AWS ecosystem.

Deployment of Anthos on AWS

The deployment of Anthos on AWS consists of the following steps:

1. Prepare the AWS account.
2. Install the management service.
3. Create user clusters to run your workloads.

You can find the step-by-step installation procedure by following the link: `https://cloud.google.com/anthos/clusters/docs/aws/how-to/installation-overview?hl=pl`.

Anthos on Azure

At the time of writing this chapter, the preview of Anthos on Azure has been announced. The capabilities at this time are limited. You can deploy the Anthos cluster and workload, but some management options are not there yet. As an example, you can connect to the cluster using the Connect gateway but you cannot use Terraform to manage your clusters. However, Google has announced this will be solved soon.

Anthos on Azure components

The architecture of Anthos on Azure is very similar to what we have already seen with AWS. Therefore, there are two components to Anthos clusters on Azure:

- **Management service**: This allows us to install, update, and manage the user clusters via the Azure API
- **User cluster**: This runs the user workloads

The following figure shows the architecture of Anthos on Azure:

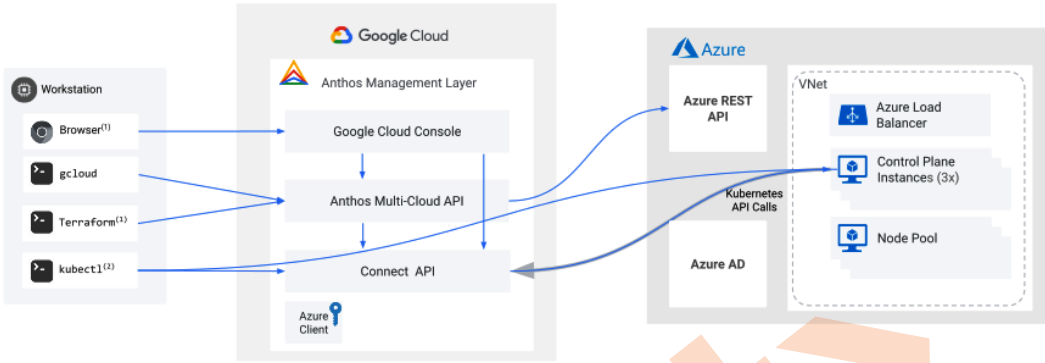

Figure 8.6 – Anthos on Azure (Source: `https://cloud.google.com/anthos/clusters/docs/azure/concepts/architecture`)

Let's have a closer look at each of the components of Anthos on Azure.

Management service

The management service of Anthos on Azure consists of the following:

- The **Anthos multi-cloud API** that accepts the `gcloud` command to provision and manage the Anthos clusters
- **AzureClient**, a secure connection to Azure using a X.509 key pair, with the private key stored in GCP and the public key in Azure
- **App Registration**, which allows the Anthos cluster to create resources in Azure using AzureClient
- The **Connect API**, which allows management of the cluster directly from Google Cloud Console

User cluster

The user cluster has two components: a control plane and at least one node pool. The control plane is responsible for storing the configuration in a highly available `etcd` database. The control plane can manage multiple node pools. The node pool consists of VMs that act as worker nodes.

Load balancing

Anthos on Azure uses the Azure-native Azure Load Balancer for both the cluster control planes and workloads. The load balancer is automatically provisioned by Anthos automation when the user deploys a Kubernetes `Service` of type `Loadbalancer`.

Deployment of Anthos on Azure

The deployment of Anthos on Azure consists of the following steps:

1. Prepare the Azure account.
2. Install the management service.
3. Create the user clusters.
4. Create node pools.

You can find the step-by-step installation procedure by following the link: `https://cloud.google.com/anthos/clusters/docs/azure/how-to/installation-overview`.

Anthos clusters on bare metal

With Anthos on bare metal, you can take advantage of your existing infrastructure and don't need vSphere as the virtualization layer. As you are running Anthos directly on the hardware, you get the best performance possible, including low network latency. The cost of using Anthos clusters on bare metal is reduced by not having to pay for vSphere licenses. Another benefit is that you can reduce latency by moving the Anthos clusters to the edge of the network.

Figure 8.7 shows the three layers of an Anthos cluster on bare metal: the hardware, operating system, and Anthos itself. You can install the cluster on the nodes running CentOS, RHEL, or Ubuntu:

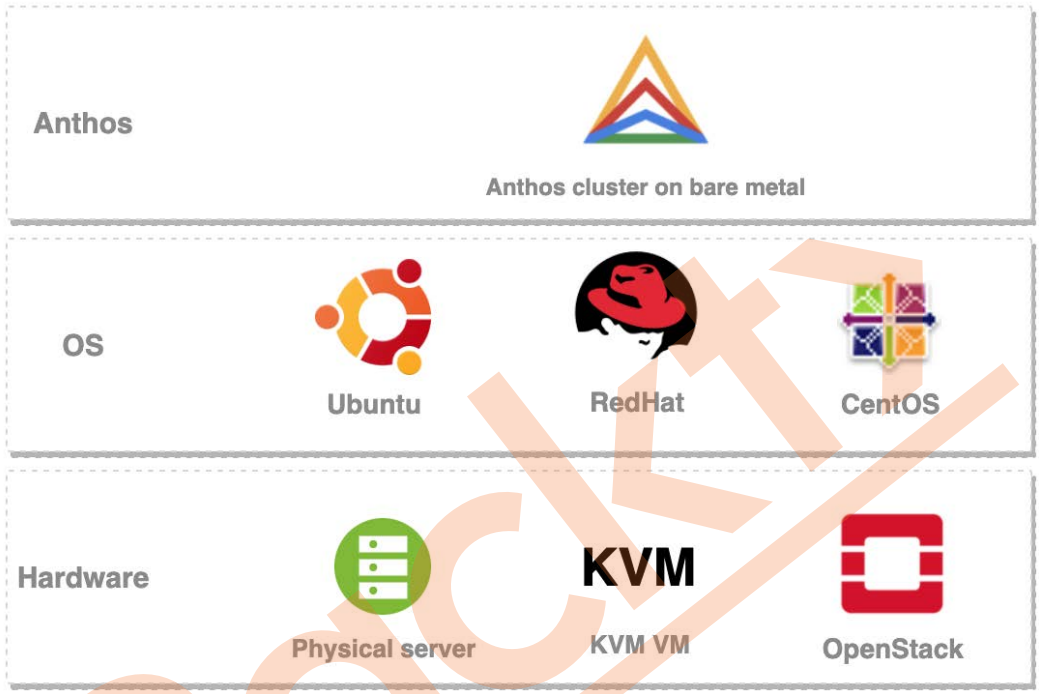

Figure 8.7 – Anthos cluster on bare metal

Consult the documentation for the exact supported versions of your chosen OS.

The Anthos cluster nodes can be also installed on VMs so you can take advantage of your existing hypervisors.

In this deployment option you are responsible for providing the hardware and managing the operating system of the cluster nodes. Google is responsible for providing you the Anthos binaries. You manage the Anthos cluster yourself.

Anthos on bare metal cluster types

Anthos clusters on bare metal consist of two types of clusters:

- **Admin clusters** to control the resources of your clusters
- **User clusters** to run workloads

You can also choose to deploy the cluster in the following modes:

- **Hybrid cluster**: Clusters that combine administration tasks and workloads, as well as controlling other user clusters. The first cluster acts as both the admin and user clusters while other clusters act as user clusters
- **Standalone cluster**: A single standalone Anthos cluster on bare metal
- **Multi-cluster**: A single management cluster with one or more user clusters

This gives you a lot of flexibility in terms of architecting your cluster. You can either choose to have a standalone cluster or manage multiple clusters with an admin cluster.

Load balancing modes

As in other deployment options, you need to make a choice on a load balancer for both the control plane and workloads. With bare metal you can choose from the following:

- **Bundled load balancer**: Where the load balancer is deployed on a dedicated pool of worker nodes on your control plane cluster nodes
- **Manual load balancer mode**: Where you deploy the load balancer outside of the cluster. You can use either **MetalLB** or **F5 BIG-IP**

Note that for the bundled load balancer there is a requirement that all nodes are in the same L2 subnet as it uses ARP broadcasts.

Summary of Anthos clusters on bare metal

Anthos clusters on bare metal cover all use cases where there is a need for high performance and direct access to underlying hardware. One such use case is in telco scenarios. Another interesting feature is the capability of hosting VMs using an open source project called Kubevirt (`https://kubevirt.io`). To learn more about Anthos clusters on bare metal have a look at the following link: `https://cloud.google.com/anthos/clusters/docs/bare-metal`.

Anthos Config Management

Anthos Config Management comes with a suite of components that provide you with constant configuration and policies across multiple Anthos clusters. The components are interoperable but can also be used independently of each other. *Figure 8.8* shows three components included in the service:

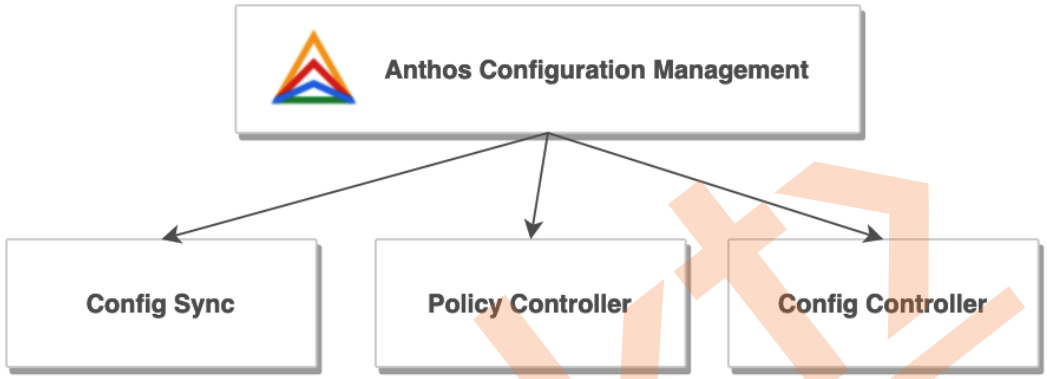

Figure 8.8 – Anthos Configuration Management

Let's look at the components in more detail:

- **Config Sync** continuously reconciles the state of your clusters with a central set of configurations stored in one or more Git repositories. This feature lets you manage common configurations with an auditable, transactional, and version-controlled deployment process that can span hybrid or multi-cloud environments:

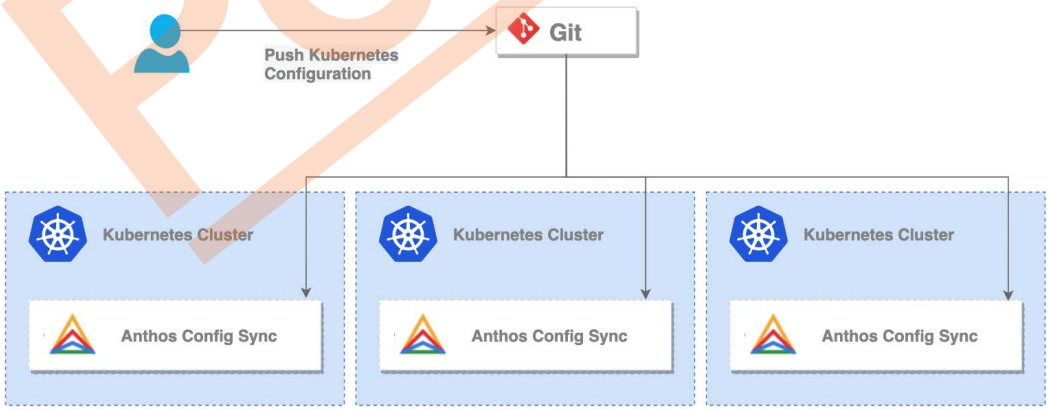

Figure 8.9 – Anthos Configuration Manager

- **Policy Controller** enables the enforcement of fully programmable policies. You can use these policies to actively block non-compliant API requests, or simply to audit the configuration of your clusters and report violations. Policy Controller is based on the open source Open Policy Agent Gatekeeper project and comes with a library of pre-built policies for common security and compliance checks:

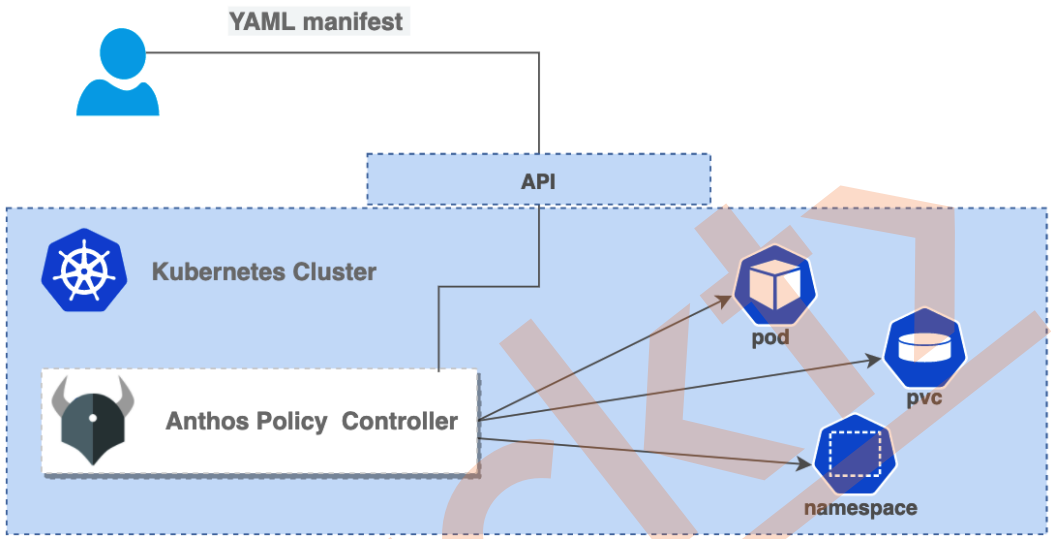

Figure 8.10 – Anthos Policy Controller

The policies are enforced by using objects called constraints. For example, you can use a constraint to require each namespace to have at least one label, or restrict the repositories a given container image can be pulled from. You can also create your own polices with the use of constraint templates. You can find the Google-developed constraint library at `https://cloud.google.com/anthos-config-management/docs/reference/constraint-template-library`.

As policies are defined as Kubernetes objects, they can be pushed directly to the cluster with the use of a Git repository and the Config Sync service.

As we can see, Config Management is a very powerful service that can help you keep control of your Anthos clusters.

Config Connector allows you to provision and manage GCP resources using Kubernetes manifests. This way, you gain a single API to provide both GCP and Kubernetes resources:

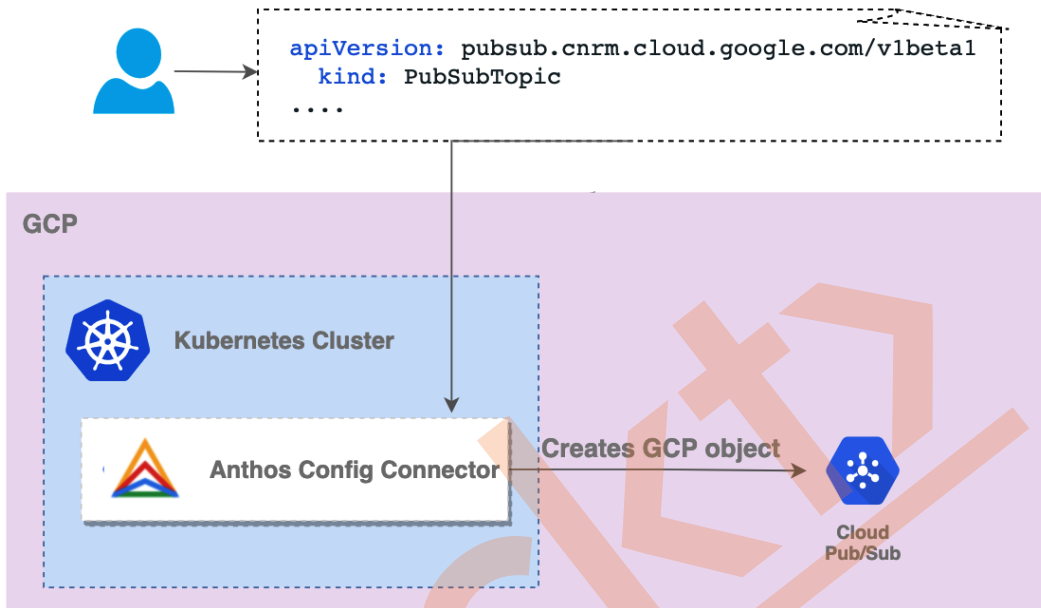

Figure 8.11 – Anthos Config Connector

Config Connector comes with **Custom Resource Definitions** (**CRDs**) and associated controllers for almost all GPC services. If you want to learn more about the resources supported, visit `https://cloud.google.com/config-connector/docs/reference/overview`.

Config Controller service, at the time of writing this book, is in Preview state. It is a Google-hosted service that combines Config Connector and Config Sync services to allow GitOps operations for Google Cloud and Anthos resources. To achieve this, an administrator defines the desired state of the resources using Config Connector CRDs and stores them in a Git repository. Config Sync makes sure the desired state is enforced.

Note: It is possible that rebranding of this service might happen without any notice, so please refer to the link: `https://cloud.google.com/anthos-config-management/docs/concepts/config-controller-overview` to learn about current status of the service.

Service management with Anthos Service Mesh

Anthos Service Mesh is a fully managed service mesh for microservices architectures. It is based on an open source project, started by Google, called Istio. Google is gradually integrating Istio's functionalities with other Google Cloud services such as the Cloud operations suite to provide an enterprise-grade service mesh offering. Let's have a look at Istio first to understand how it works so we can better understand Anthos Service Mesh.

Istio

Istio is an open source service mesh project. It runs in Kubernetes but may also integrate with traditional workloads hosted on VMs. It provides a programable and application-aware network. With the use of open source Envoy proxies, the traffic between microservices is intercepted and can be controlled, secured, and observed. This allows us to gain full control and visibility of the traffic between complex microservice-based applications. As we see in *Figure 8.12*, Istio injects Envoy sidecar proxies to the Kubernetes Pods:

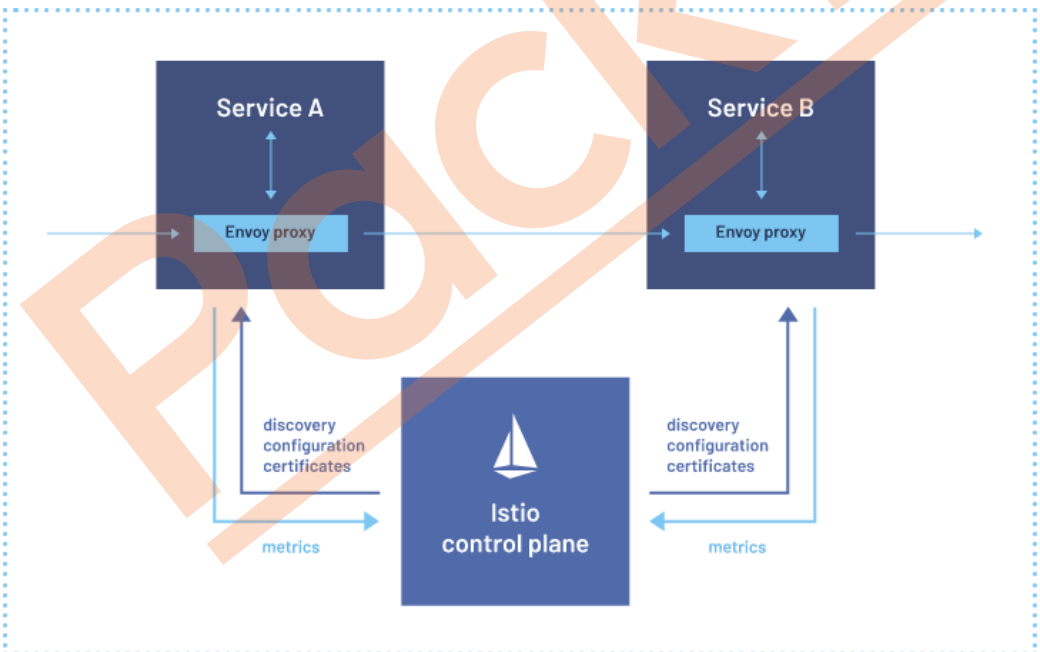

Figure 8.12 – Istio (Source: `https://istio.io/latest/`)

This way, all the ingress and egress traffic needs to pass through the proxies. The Istio control plane is a Kubernetes application that runs on your cluster. It gathers the metrics and controls network traffic based on the characteristics defined in the Kubernetes manifest.

With Istio you can perform the following tasks:

- **Request routing**: Route requests to multiple versions of a microservice
- **Fault injection**: Inject faults to test the resiliency of the application
- **Traffic shifting**: Gradually migrate traffic between different versions of a microservice to another
- **Query metrics**: Query for Istio metrics using **Prometheus**
- **Visualize metrics**: Use the Istio Dashboard to monitor mesh traffic using **Grafana**
- **Access external services**: Configure the Envoy proxy to pass through requests for unknown services
- **Visualize the service mesh**: Use the Kiali web-based graphical user interface to view service graphs of the mesh and your Istio configuration objects
- **Encrypt the traffic**: Enforce encryption of the traffic between microservices using mTLS

> **Exam tip**
>
> Make sure you understand well all the capabilities of Istio and Anthos Service Mesh: request routing; fault injection; traffic shifting; querying metrics; visualizing metrics; accessing external services; visualizing your mesh.

Configuring Istio

You can install Istio on you any Kubernetes cluster by following the official guide at `https://istio.io/latest/docs/setup/getting-started`. Once you have Istio installed you can define for which namespace you want the sidecar proxies to be injected by labeling the namespace:

```
kubectl label namespace default istio-injection=enabled
```

In the guide mentioned above, you download the `Istio` repository that contains a sample application called `Bookinfo`. Deploy the application to the default namespace by running the following command:

```
kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml
```

You can list the Pods by running the following command:

```
kubectl get pods
```

When you run the preceding commands, you will see the sidecar proxies have been automatically added to the Pods. As we see here, 2/2 containers (the workload and the proxy) are reported as **READY**:

```
NAME                              READY   STATUS    RESTARTS   AGE
details-v1-558b8b4b76-2llld       2/2     Running   0          2m41s
productpage-v1-6987489c74-lpkgl   2/2     Running   0          2m40s
ratings-v1-7dc98c7588-vzftc       2/2     Running   0          2m41s
reviews-v1-7f99cc4496-gdxfn       2/2     Running   0          2m41s
reviews-v2-7d79d5bd5d-8zzqd       2/2     Running   0          2m41s
reviews-v3-7dbcdcbc56-m8dph       2/2     Running   0          2m41s
```

Figure 8.13 – Bookinfo application pods

Using Istio

As we have already seen, Istio has a lot of capabilities. As an example, you can perform traffic shifting. Let's say we want to split the traffic equally between the `reviews:v1` and `reviews:v3` services.

You can define this in a Kubernetes manifest with a `VirtualService` object and apply it using the `kubetctl apply` command:

```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
    - reviews
  http:
  - route:
    - destination:
        host: reviews
        subset: v1
      weight: 50
    - destination:
        host: reviews
        subset: v3
      weight: 50
```

Istio will make sure that the incoming traffic is split evenly between those two services.

> **Exam tip**
>
> It is not required for the PCA exam to have in-depth knowledge of Istio, but if you would like to learn more on how to use it, you can find a step-by-step guide at `https://istio.io/latest/docs/setup/getting-started/`.

Using Anthos Service Mesh

Now that we know what Istio is, let's have a look at **Anthos Service Mesh**, which is a managed version of Istio. Note that as usual, the features for Anthos Service Mesh will be first introduced to GKE on GCP and next to the remaining flavors of Anthos clusters. As the list is pretty long, have a look at the up-the-date list at `https://cloud.google.com/service-mesh/docs/supported-features`.

As Anthos Service Mesh is simply an Anthos-verified and optimized distribution of Istio, you can use the Istio CRDs as you would with the open source version.

The additional advantages that come from Anthos Service Mesh are due to the integration with the GCP ecosystem. This means you get a single-pane-of-glass view in the Google Cloud Console for your mesh. You can visualize all your services and traffic between them and create alerts in Cloud Monitoring. You can access it from the **Service Mesh** menu in the Anthos console:

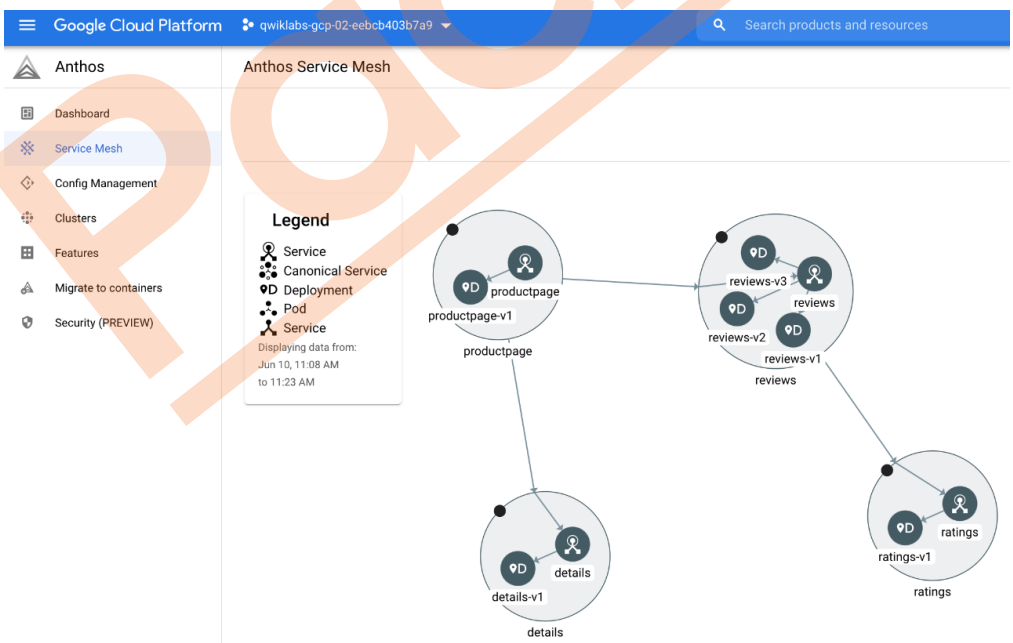

Figure 8.14 – Service Mesh visualization

You can also get details of the traffic per service using the **Traffic** option from the left-hand menu in Anthos Service Mesh:

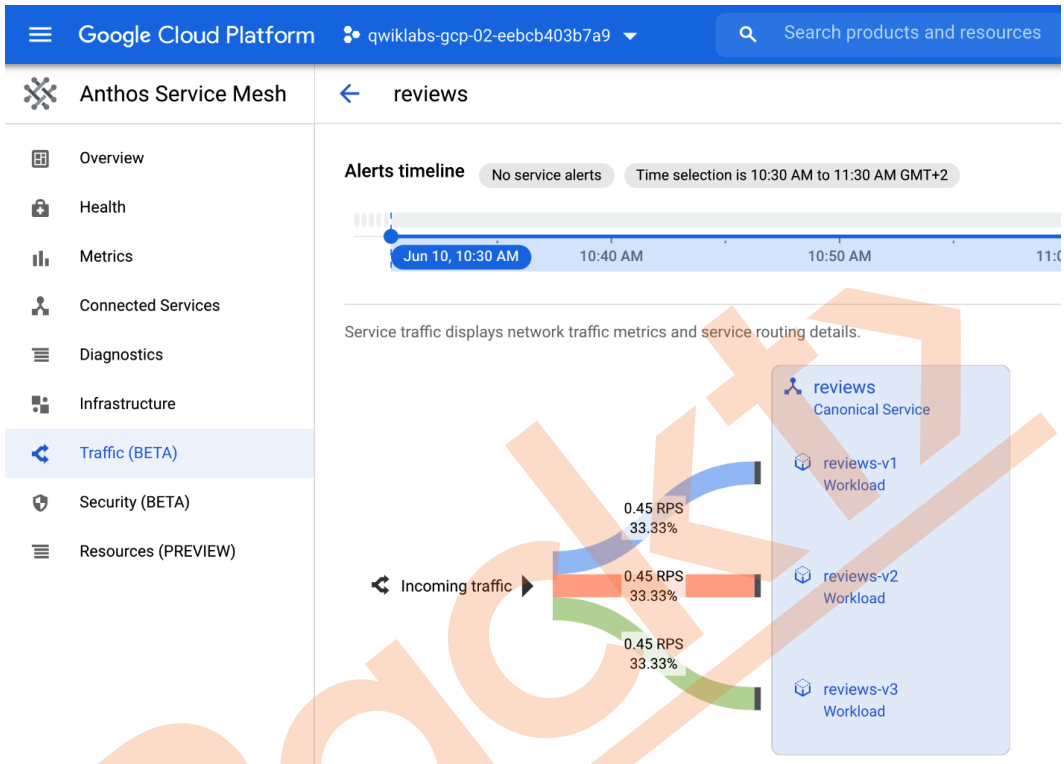

Figure 8.15 – Service traffic visualization

This will show you how much traffic is routed to each version.

Service - Level Objectives

Another very useful feature is that you can create so-called **Service-Level Objectives** (**SLOs**) based on **Service-Level Indicators** (**SLIs**). SLIs measure how well you service is performing. The metrics can be based either on availability or latency:

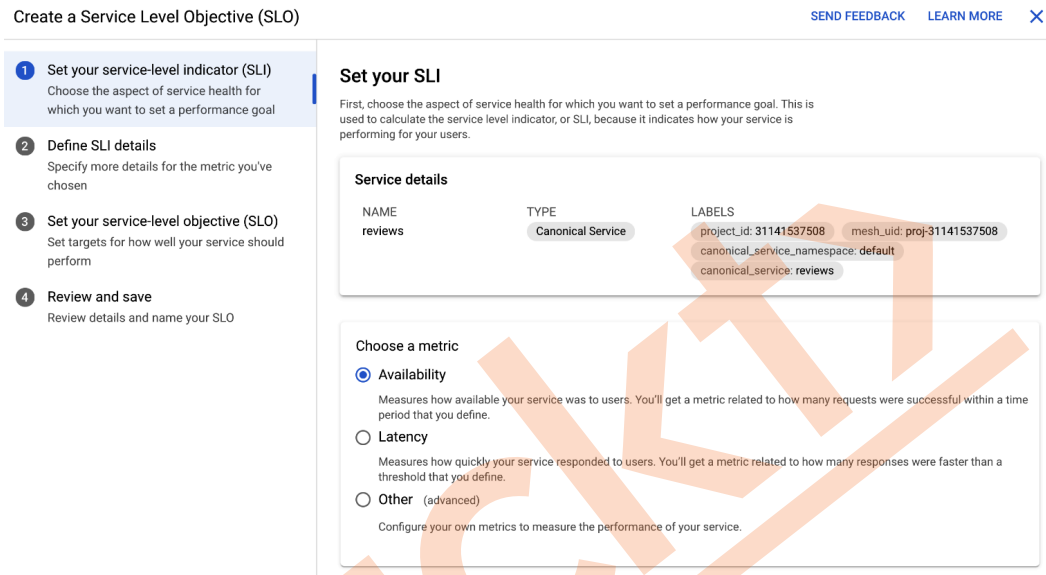

Figure 8.16 – Creating an SLO

You can also choose to use an advanced option and define your own metric.

Summary of Anthos Service Mesh

As we have learned here, Anthos Service Mesh is a very powerful tool that allows you to get control of the traffic within your Anthos clusters. It also gives brilliant insights into the health of your application by being able to see metrics relevant to this.

If you want to get hands-on experience with Anthos Service Mesh, please check the *GSP656 -Traffic Management with Anthos Service Mesh* lab at `https://www.qwiklabs.com/`.

Anthos Multi Cluster Ingress (MCI)

Anthos MCI is a service available under your Anthos subscription. It is built on the architecture of Google's external HTTP(S) Load Balancing service with proxies deployed at over 100 Google **points of presence** (**PoPs**) around the world. As we see in *Figure 8.17*, MCI provides a single IP for the app independent of where the app is deployed.

> **Exam tip**
>
> MCI should be used when there is a need for load balancing for multi-cluster and multi-regional applications. It's a controller for the external HTTP(S) Load Balancer that provides ingress for traffic coming from the internet to one or more Anthos clusters.

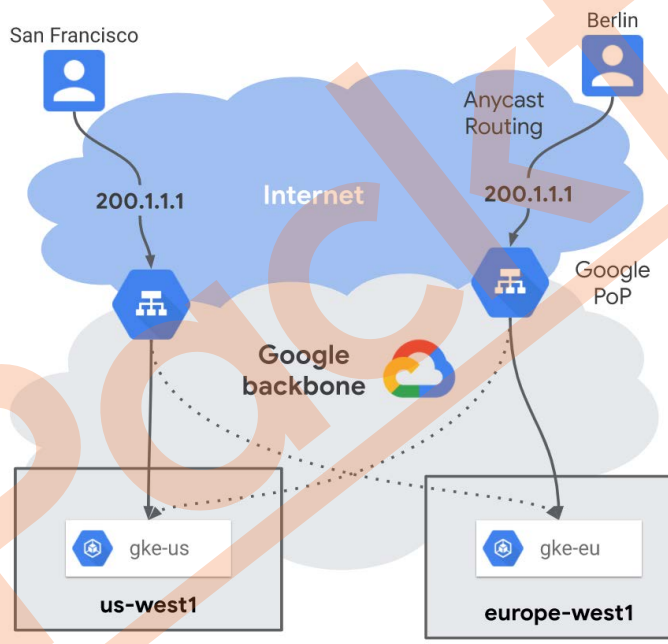

Figure 8.17 – Anthos MCI (Source: `https://cloud.google.com/kubernetes-engine/docs/concepts/multi-cluster-ingress`)

Anthos MCI comes with two very powerful features supporting the multi cluster setup.

MCI's proximity-based routing directs the users to the cluster nearest to their location. MCI's health checks allow us to perform seamless cluster upgrades and traffic failovers.

You can learn more about working with MCI at `https://cloud.google.com/kubernetes-engine/docs/concepts/multi-cluster-ingress`.

Anthos Binary Authorization

Binary Authorization is a Google Cloud service aimed at providing security for your containerized software supply chain. It reduces the risk of deploying defective, vulnerable, or unauthorized software.

It allows you to create policies that kick in when there is an attempt to deploy a container on one of the supported platforms. It is done by means of so-called **attestations** that certify the images for the deployments.

At the time of writing this book, Binary Authorization supports the following platforms:

- GKE
- Cloud Run
- Anthos clusters on VMware

Binary Authorization compliments the suite of Google software supply chain services that includes Cloud Source Repositories, Cloud Build, Artifact Registry, and Container Analysis:

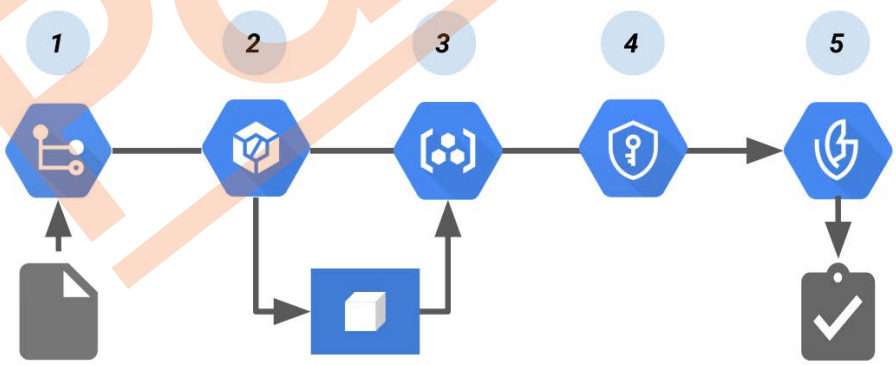

Figure 8.18 – The Cloud Build pipeline that creates a Binary Authorization attestation (Source: `https://cloud.google.com/binary-authorization/docs/cloud-build?hl=zh-tw`)

As we see in the preceding diagram, the pipeline works as follows:

1. The developer pushes the code to build the container image into **Cloud Source Repositories**.
2. **Cloud Build** builds the container image.
3. **Cloud Build** pushes the container image to **Artifact Registry.**
4. **Cloud Key Management Service** creates a cryptographic key pair to sign the container image. The signature is then stored in an attestation.
5. During deployment, the attestor in **Binary Authorization** verifies the attestation using the public key from the key pair.

Use of Binary Authorization should be considered a best practice for the deployment of production workloads.

Migrate for Anthos and GKE

Migrate for Anthos and GKE allows us to modernize traditional VM-hosted applications by extracting and migrating them into containers that can run on top of GKE or Anthos clusters. This comes with the following benefits:

- **Better density** by using containers to optimize usage of cluster resources
- A **security-optimized node kernel** with no need to update the operating system
- **Integrate traditional apps with modern services** including Anthos Service Mesh, Cloud Logging, Cloud Monitoring, and other GCP-native services to offload infrastructure-related functionalities to GCP
- **Unified policy and configuration management** with use of declarative definitions
- **Modern image-based management and orchestration** through integration with CI/CD tools

As of the time of writing this book, the following target platforms are supported:

- GKE
- Anthos clusters on GCP
- Anthos clusters on VMware
- Anthos clusters on AWS

The service is available for both Windows and Linux VMs as sources for the migration.

For Linux VMs, the migration process includes the following steps:

1. Swapping the VM's OS kernel to the kernel used by the GKE node
2. Setting up the network interface, DNS, logging, and health status
3. Running the application and services found in the user space of the VM
4. Removal of the VM or hardware-related services

For Windows VMs, a Dockerfile with the container image definition and a ZIP file with the application content are generated. An official Microsoft Windows Server image is used.

The migration process is divided into multiple phases:

1. Discovery phase
2. Migration planning phase
3. Landing zone phase
4. Migration and deployment phase
5. Operate and optimize phase

If you want to learn more about the architecture of Migrate for Anthos and GKE, and the detailed procedure for migration, visit `https://cloud.google.com/migrate/anthos/docs/concepts`.

Cloud Run for Anthos

In *Chapter 7*, *Deploying Cloud-Native Workloads with Cloud Run*, we learned about Google Cloud Run, which allows us to run serverless-like cloud-native workloads on GCP. With Cloud Run for Anthos this capability is available to be executed on Anthos clusters either in GCP, public clouds, or on-premises. As Cloud Run for Anthos is powered by the open source Knative (`https://knative.dev`) project, the workloads are portable to any Kubernetes cluster.

For Anthos-enabled projects, when you create a Cloud Run service, you should choose **Cloud Run for Anthos** rather than **Cloud Run (fully managed)** for the deployment platform:

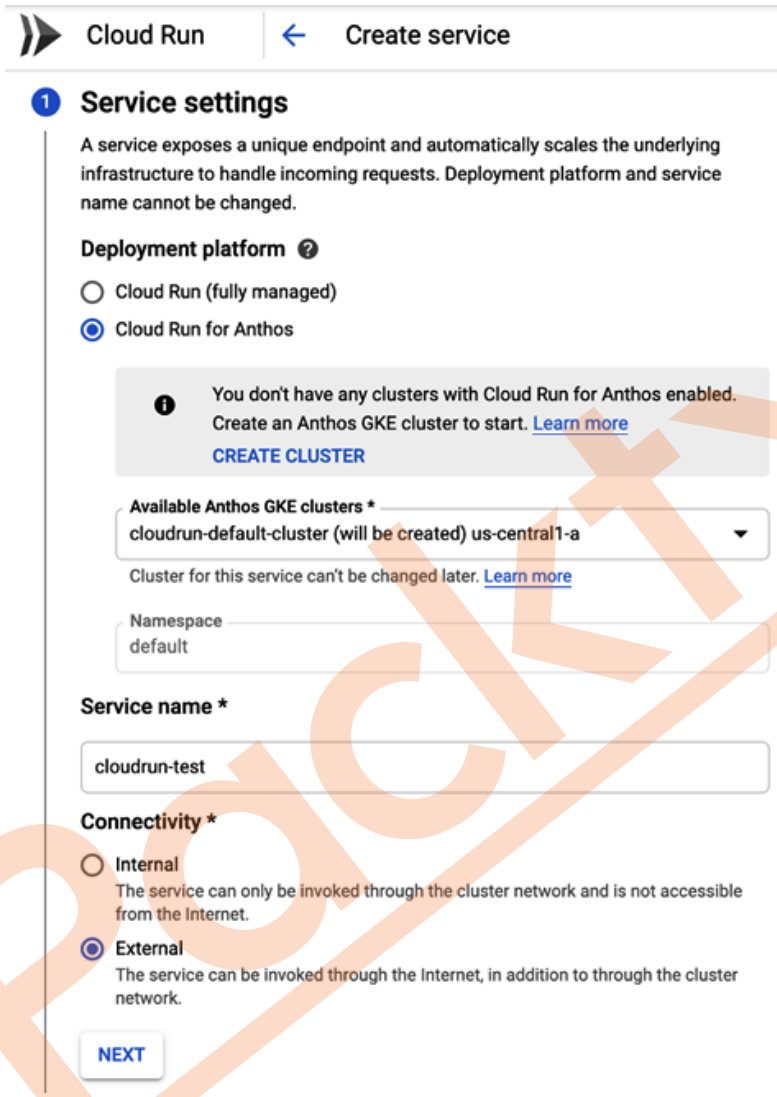

Figure 8.19 – Cloud Run for Anthos

If you don't have any Anthos clusters with Cloud Run enabled, the cluster will be created for you. If you want to create a cluster with Cloud Run enabled, add the following flags to the `gcloud` command:

```
gcloud container clusters create CLUSTER_NAME \
  --addons=HttpLoadBalancing,CloudRun \
  --machine-type=n1-standard-2 \
  --enable-stackdriver-kubernetes
```

Note that you need to have the default zone and region specified for this command to run correctly. Once executed successfully, you can deploy the Cloud Run service on your cluster.

Cloud Run (fully managed) versus Cloud Run for Anthos

In the following table we have a simple summary of the differences between Cloud Run (fully managed) and Cloud Run for Anthos:

Feature	Cloud Run	Cloud Run for Anthos
Price	Pay-per-use	Anthos cluster cost
Compute	CPU and memory limits	Anthos cluster nodes' capabilities (includes GPUs)
Isolation	Uses gVisor sandbox	Anthos cluster isolation
Scaling	1,000 containers with extensible quotas	Anthos cluster-based
Domains	Available	
VPC access	Serverless VPC access	Direct access to VPC
Service mesh	Not connected to Istio Service Mesh	Services connected to Istio Service Mesh
Execution environment	Google-native infrastructure	Anthos cluster

As we can see, the differences are not only in the pricing model and execution environment but also in the functionalities provided. The choice of which option to run your workloads on needs to be based on your requirements. Each of the services has its pros and cons.

Quotas and limits

Anthos comes with a set of quotas and limitations. These are in place to make sure you get the best performance, and you are protected from provisioning too many resources. As the quotas and limits might change, it is best to refer to the documentation to find the most up-to-date quotas, at `https://cloud.google.com/anthos/clusters/docs/on-prem/concepts/scalability`.

As an example, you can see what quotas are applicable to Anthos clusters on VMware (GKE on-prem) at the time of writing this book:

- You can deploy a maximum of 20 user clusters per admin cluster.
- For each user cluster:
  - You must create a minimum of 3 nodes.
  - You can create a maximum of 250 nodes.
  - You can create a maximum of 7,500 pods:

- For each node, you can create a maximum of 110 pods.
- For each Google Cloud project, you can register a maximum of 15 user clusters in the GKE hub, but you can submit a request to increase your quota.

Now let's review the Anthos pricing.

Pricing

Anthos clusters are charged per vCPU of the worker nodes in the user clusters. This excludes both the admin cluster and the master node. There are two charging models you can choose from:

- **Pay-as-you-go** pricing, where you are billed at the current applicable rates for Anthos-managed clusters as you use them. You can start using pay-as-you-go Anthos without any long-term commitment
- **Subscription** pricing, where you pay a monthly charge for your Anthos deployments. You can benefit from a discounted price for long-term subscription commitments

As the actual pricing might change, please refer to the documentation for the price list, at `https://cloud.google.com/anthos/pricing`.

Summary

In this chapter, we learned about Anthos and its most important features. We went over the Anthos deployment options, including **GKE** (**Google Cloud**), **on-premises** (**VMware**), **on-premises** (**bare metal**), **AWS**, **attached clusters**, and **Azure**. We looked at how to control Anthos clusters' configuration in sync with the defined configuration and enforce policies using **Anthos Config Management**. We learned how to control and observe traffic in microservice architectures using **Anthos Service Mesh**. We also understood how we can distribute inbound traffic using **Anthos MCI**. Finally, we learned how to run serverless-like cloud-native workloads on Anthos clusters using **Cloud Run on Anthos**. We compared this to running the workload on **Cloud Run** (**fully managed**). In the next chapter, we will have a look at running serverless functions with Google Cloud Functions.

Further reading

You can refer to the following links for more information:

- **Anthos main page**: `https://cloud.google.com/anthos/docs/concepts/overview`
- **Anthos on VMware**: `https://cloud.google.com/anthos/clusters/docs/on-prem/`
- **Anthos clusters on bare metal**: `https://cloud.google.com/anthos/clusters/docs/bare-metal/1.6/concepts/about-bare-metal`
- **Anthos on AWS**: `https://cloud.google.com/anthos/clusters/docs/aws/concepts/architecture`
- **Anthos on Azure**: `https://cloud.google.com/anthos/clusters/docs/azure/`
- **Athos Configuration Management**: `https://cloud.google.com/anthos/config-management`
- **Anthos Service Mesh**: `https://cloud.google.com/anthos/service-mesh`
- **Istio**: `https://istio.io/latest/docs`
- **Anthos for Anthos and GKE**: `https://cloud.google.com/migrate/anthos/docs/concepts`
- **Cloud Run on Anthos**: `https://cloud.google.com/anthos/run/`
- **Multi Cluster Ingress**: `https://cloud.google.com/kubernetes-engine/docs/concepts/multi-cluster-ingress`
- **Config Connector**: `https://cloud.google.com/config-connector`