Reference architecture:

Anthos hybrid environment

February 2023

Table of contents

Overview

Organizations that embrace cloud-first technologies like containers, container orchestration, and service meshes, often reach a point where they need more than a single Kubernetes cluster. Many organizations that use Google Cloud also want to run workloads in their own data centers, factory floors, retail stores, or even in other public clouds.

However, operating multiple Kubernetes clusters has its own difficulty and overhead in terms of consistent configuration, security, and management. For example, manually configuring one Kubernetes cluster at a time creates risks, and it can be challenging to see exactly where errors are happening.

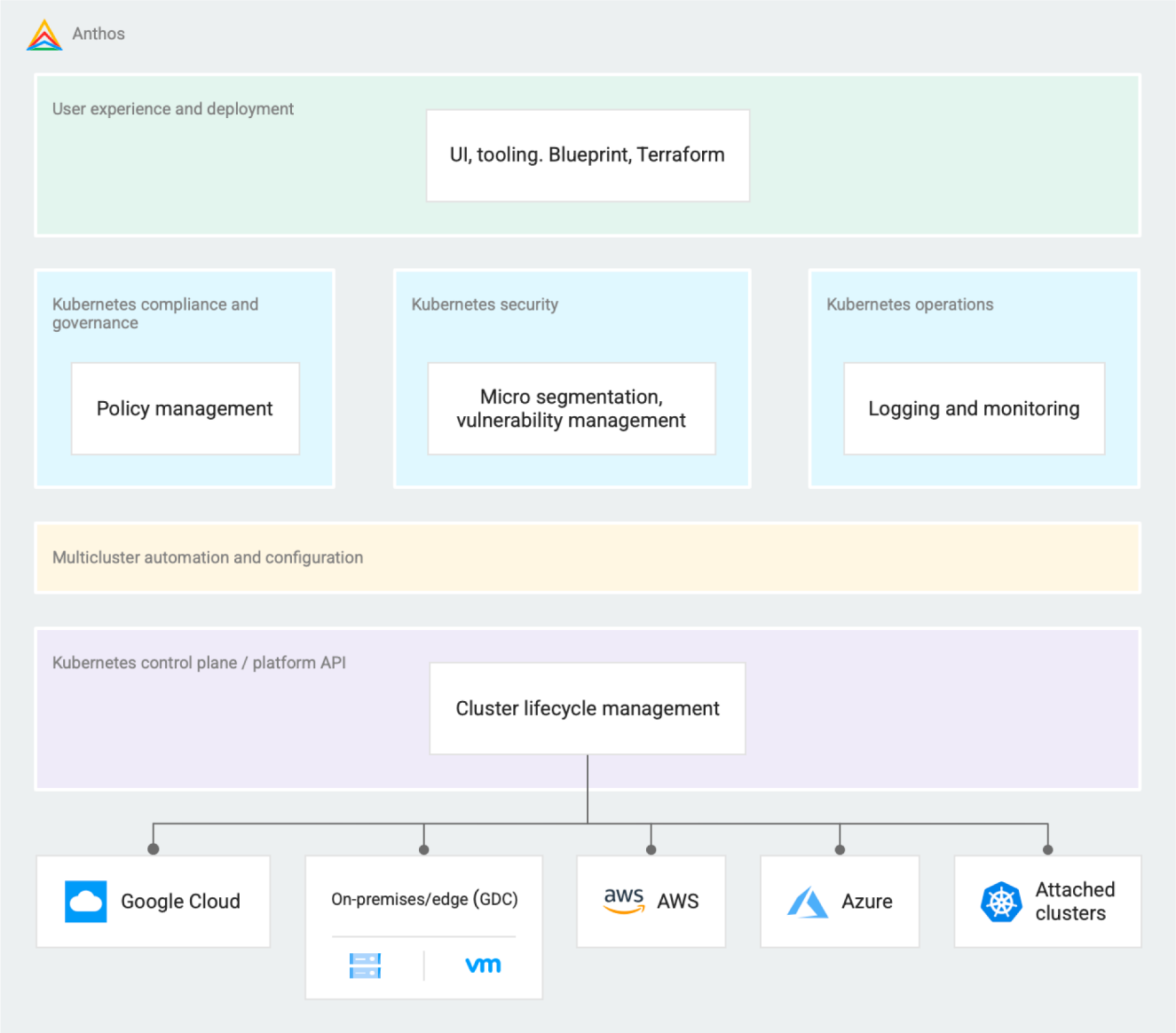
Anthos is Google's cloud-centric container platform for running modern apps anywhere consistently at scale. Anthos can help organizations by providing a consistent platform that lets them:

- Modernize applications and infrastructure in-place.
- Create a unified cloud operating model (single pane of glass) to create, update, and optimize container clusters wherever they are.
- Scale large multi-cluster applications as *fleets* - logical groupings of similar environments - with consistent security, configuration, and service management.
- Enforce consistent governance and security from a unified control plane.

Anthos helps you increase operational consistency in governance and security and developer velocity while reducing cost, deployment risk, and operational complexity. Specifically, Anthos helps with the following areas:

- Customers who want cloud-like experience on-premises or are looking for a unified solution while migrating their applications to cloud (***Anthos hybrid environment***).
- Google Cloud customers who want to better manage their containerized applications (***Anthos on Google Cloud***).
- Customers who want to solve multicloud complexity with a consistent governance, operations, and security posture (***Anthos Multi-Cloud***).

The following diagram shows a high-level overview of Anthos in a hybrid environment. Anthos can help with Kubernetes compliance and governance, security, and operations, along with multi-cluster automation and configuration. Kubernetes clusters managed by Anthos can then run on-premises, in Google Cloud, or in another cloud provider:



This reference architecture provides opinionated guidance to deploy Anthos in a hybrid environment to address some common challenges that might face.

In this reference architecture, the term *cluster* means a Kubernetes cluster managed by Anthos unless stated otherwise. For example, some sections discuss *VMware vSphere clusters* composed of ESXi servers that pool compute resources.

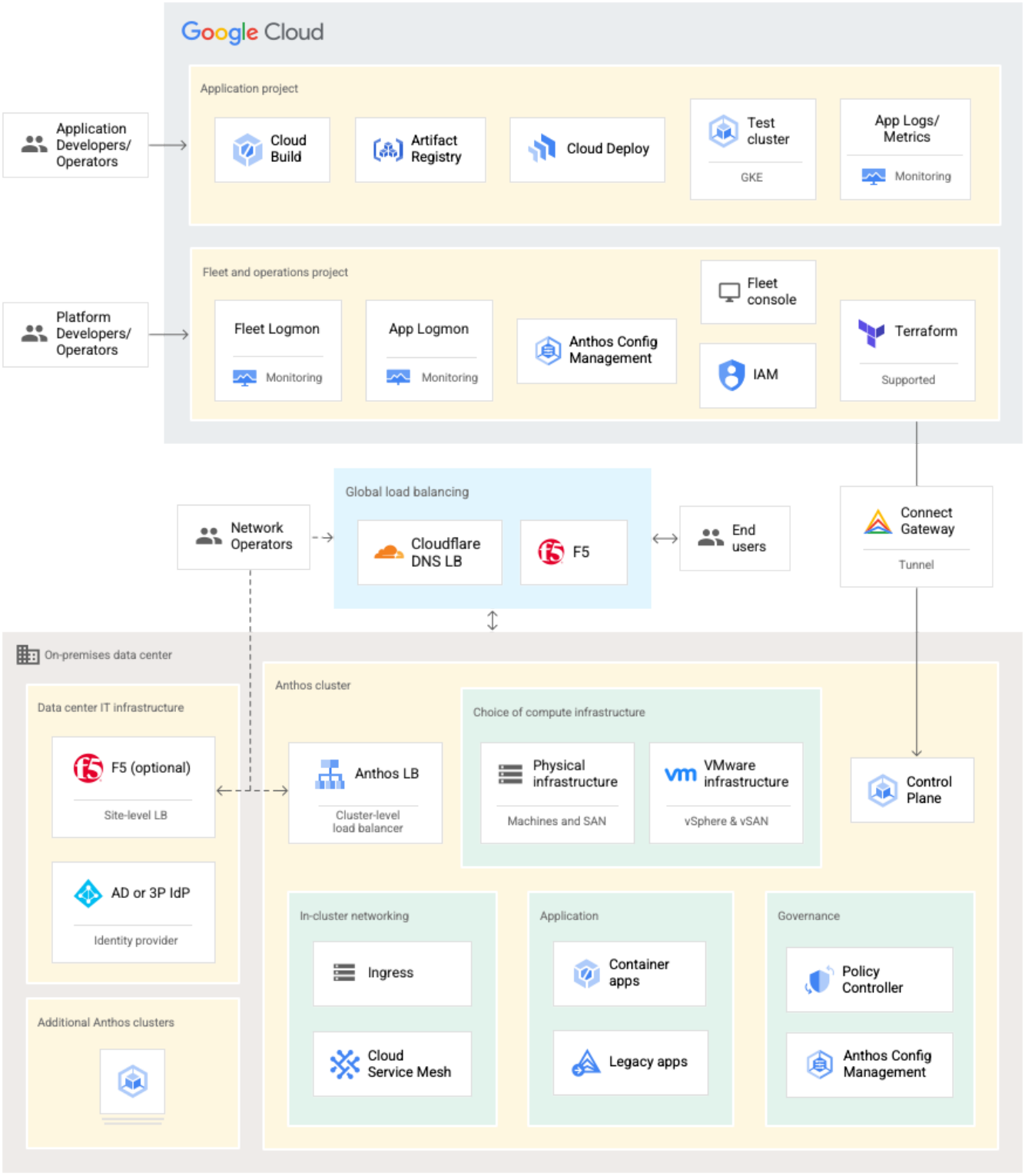
Architecture

The following architecture diagram provides an overview of a complete Anthos deployment in a hybrid environment. This reference architecture shows you how to appropriately plan, deploy, and configure these components:

- Google Cloud-based services help you manage logging and monitoring data, store container images, and provide configuration management.
- Global load balancing solutions manage traffic flows between the cloud-based services and your on-premises environment.
- The Connect gateway provides secure communication for Anthos management between environments.
- On-premises components that run in your own data center like physical servers and clusters, identity solutions, and load balancers complete the hybrid approach.

Although you might not use all of these services in your own deployments, this reference architecture explains how all of these components can work together. As your business needs change, you can add additional features and services into your environment to better run and manage your workloads.

The diagram also shows some different user personas who interact with the services, such as application developers, application operators, platform developers and operators, and network operators. Each of the personas has access to the resources that they need.


Google Cloud
Application project
Application Developers/ Operators
Cloud Build
Artifact Registry
Cloud Deploy
Test cluster
GKE
App Logs/ Metrics
Monitoring
Fleet and operations project
Platform Developers/ Operators
Fleet Logmon
Monitoring
App Logmon
Monitoring
Anthos Config Management
Fleet console
IAM
Terraform
Supported
Global load balancing
Network Operators
Cloudflare DNS LB
F5
End users
Connect Gateway
Tunnel
On-premises data center
Data center IT infrastructure
F5 (optional)
Site-level LB
AD or 3P IdP
Identity provider
Additional Anthos clusters
Anthos cluster
Anthos LB
Cluster-level load balancer
Choice of compute infrastructure
Physical infrastructure
Machines and SAN
VMware infrastructure
vSphere & vSAN
Control Plane
In-cluster networking
Ingress
Cloud Service Mesh
Application
Container apps
Legacy apps
Governance
Policy Controller
Anthos Config Management

Anthos components

Anthos consists of services that run in Google Cloud, and various functional components that run on-premises. This reference architecture uses the following Google Cloud-based services:

- **The Google Cloud console** provides a consolidated view of Anthos clusters that span across sites, and lets platform administrators manage and observe their setup across fleets.
- **The Connect gateway** lets people and automation tools connect to clusters in the cloud using Identity and Access Management (IAM).
- **Hosted cluster lifecycle management services** let you create and manage Anthos clusters on VMware and Anthos clusters on bare metal using the Google Cloud console.
- **Fleets** let you group together multiple Kubernetes clusters from on-premises Anthos clusters, Anthos on Google Cloud, and Anthos Multi-Cloud to enable and configure multi-cluster functionalities. Clusters are automatically added to a fleet when they're created.
- **Anthos Config Management** is a set of Google Cloud-hosted and in-cluster components that let you automatically deploy shared environment configurations and enforce approved security policies across fleets.
- **Anthos Service Mesh** is a suite of tools that helps you monitor and manage a reliable service mesh on-premises or on Google Cloud. Anthos Service Mesh is powered by Istio, a highly configurable and powerful open source service mesh platform.

Anthos provides conformant Kubernetes releases with opinionated provisioning and enhancements to more deploy and manage cluster lifecycle. In a hybrid deployment, Anthos lets you deploy Kubernetes clusters with GKE version consistency on bare metal or VMware vSphere environments with support from Google.

Anthos clusters on VMware (GKE on-prem) and **Anthos clusters on bare metal** provide upstream Kubernetes releases with proprietary enhancements and support. All Kubernetes cluster components, like the control plane and worker nodes, are hosted in your on-premises data center. Clusters can be managed locally or through the cloud.

- Anthos clusters on VMware integrate with VMware vCenter to provision Kubernetes clusters and provide lifecycle management of the virtual machines and operating systems.
- Anthos clusters on bare metal use your own physical or virtual machines, and you manage the base operating system.

Reference deployments

This section of the reference architecture provides high-level examples for some important components in an Anthos hybrid environment. These reference deployments show you how to plan, design, and implement the components that make up a complete Anthos hybrid environment. The Design prerequisites and Design considerations sections in the rest of this reference architecture provide additional context to plan and design your Anthos hybrid environment.

Global view

This reference architecture consists of two or more on-premises computing customer sites and two or more Google Cloud regions, as shown in the following diagram.

There are two sites in this global view diagram, with a total of eight clusters. Multiple clusters are used for several reasons, such as:

- Two sites, `ABC01` and `XYZ01`, are used for disaster recovery.
- Two environments, `production` and `staging`, to test infrastructure changes.
- Two types of clusters are used in each environment: admin clusters and user clusters. This approach separates administrative resources, which is a security best-practice.

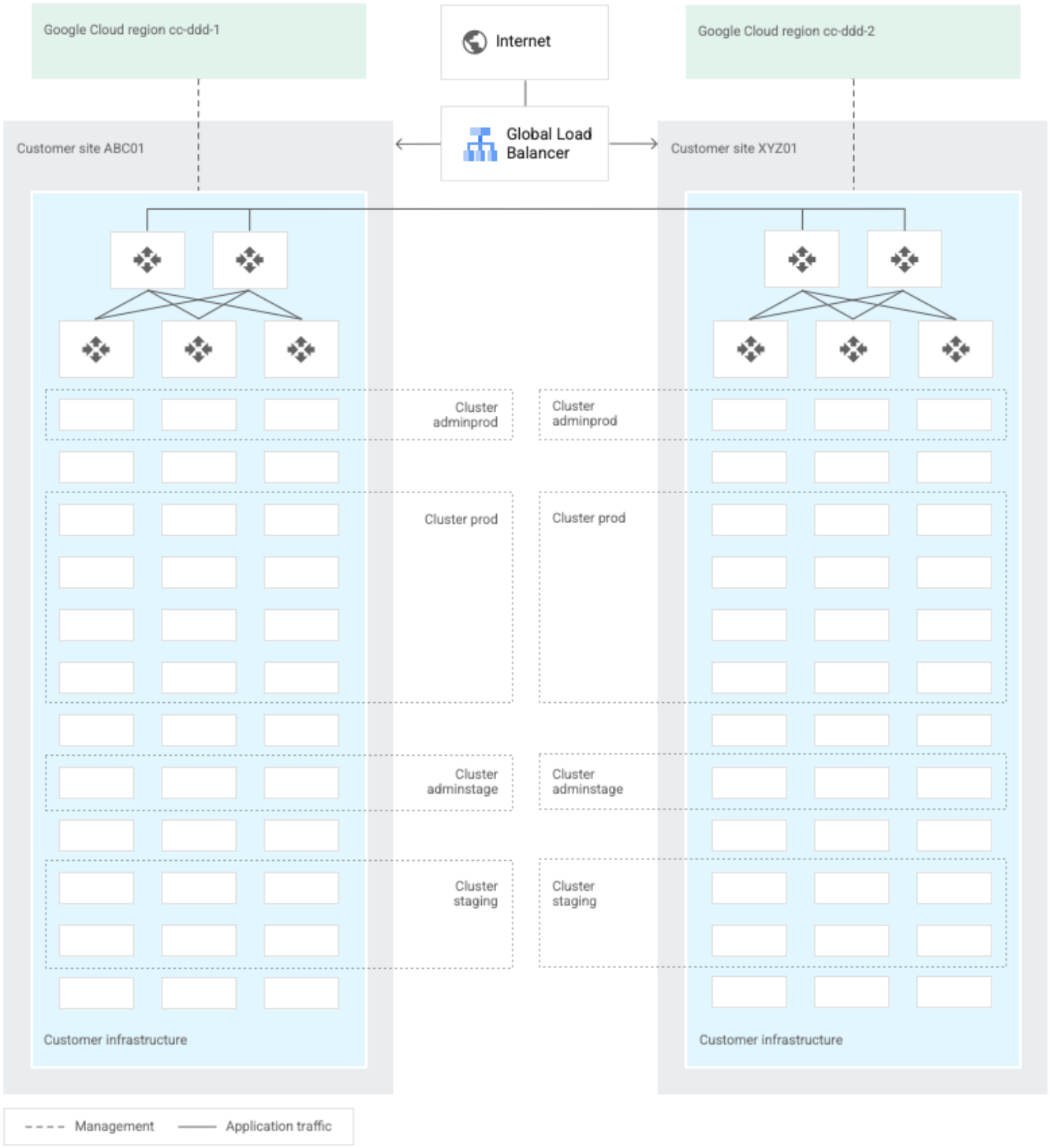
Anthos clusters have two types of clusters - *admin* clusters and *user* clusters. Admin clusters manage the lifecycle of user clusters - they are administrative. User clusters are where you deploy your applications. Separate admin clusters for production and staging environments isolates changes and allows progressive updates.

Using two sites provides for disaster recovery in the following ways:

- Allows critical applications to run in active-active configuration across both sites, with DNS load balancing to distribute traffic.
- Databases and backups can be replicated across sites.

Each customer site in the following diagram is also paired with a different Google Cloud region. Multiple Google Cloud regions serve several purposes, such as:

- If there's a configuration error at the cloud level, the scope of problems is limited. Changes made in one region at a time shouldn't affect applications that are architected for multi-region / multi-site. This behavior is an extra line of defense against misconfiguration, along with having multiple environments.
- In the unlikely event of a Google Cloud service outage, the outage is usually isolated to a single cloud region. A following section on availability provides details about the effect of regional outages.
- Network paths to cloud services can be optimized on a per-site basis.



This reference architecture uses separate production and staging environments. Separate staging environments let you safely test the following types of changes:

- New versions of software that you want to deploy to production.
- Validate new Anthos software versions before you upgrade production clusters.
- Validate changes to cloud-based settings which affect multiple clusters at once.

Staging environments allow infrastructure changes with potentially broad effects to be tested before deployment to production. A staging environment contains realistic versions of your applications, and might also be used to test application or infrastructure changes. Not every site needs a staging environment, particularly if you have more than two sites. However, having at least two staging clusters allows changes to multi-cluster management features of Anthos to be better tested than with only one.

Additional clusters to support other environments[1] can be created on-premises. For example, an on-premises sandbox cluster can be used to test new Anthos features and keep the staging environment as close as possible to production. You can also create test and development environments with cloud-based clusters using Google Kubernetes Engine (GKE).

Regional view

In the previous global view, inbound traffic passes through a global load balancer which selects a site to receive the traffic. The site traffic then either passes through a local traffic manager inside a perimeter network and is routed to an appropriate cluster to handle the request, or arrives at VIPs to be balanced using Anthos bundled load balancing.

To tolerate limited failures within a site, and to aggregate sufficient compute resources, clusters can be spread across racks. However, clusters don't stretch across sites. Each site is the boundary of a cluster.

The following diagrams illustrate a single site with the four clusters spread across three racks, in the case where virtualization isn't used.

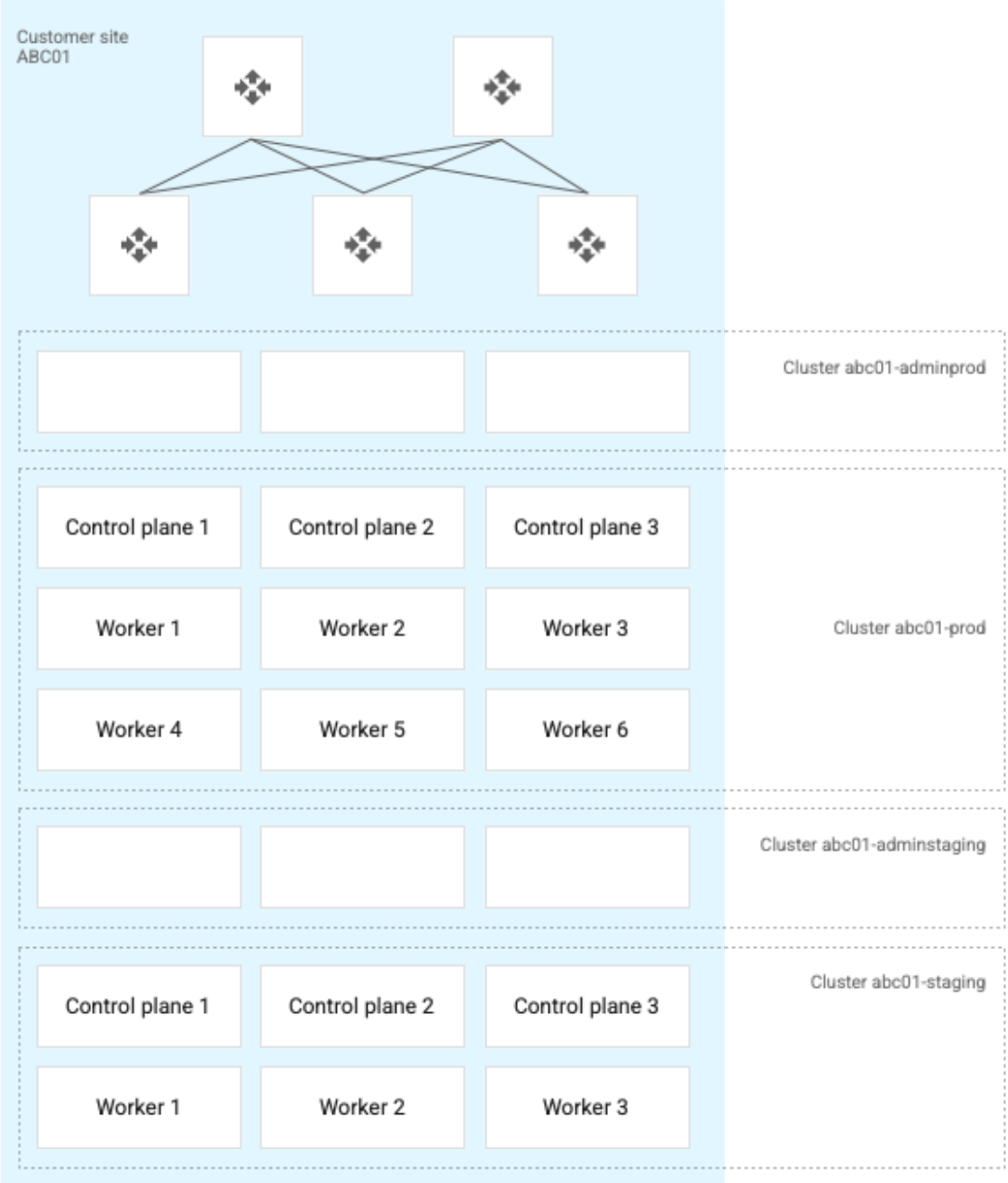
In both of the following example deployments, the clusters are configured in the following way:

- Each cluster has a highly available (HA) control plane with three members. This configuration allows for continued control plane availability concurrently with operating system upgrades, control plane software updates, or single-machine hardware or kernel failures.
- User clusters have separate control planes and worker nodes.
    - The separate control plane nodes reduce the likelihood of an application security breakout being escalated to the control plane.
    - Your applications run on the user cluster worker nodes.
- The admin clusters run local management functions for the user clusters.
    - There are two admin clusters so that admin cluster configuration changes and updates can be tested in the staging environment first.

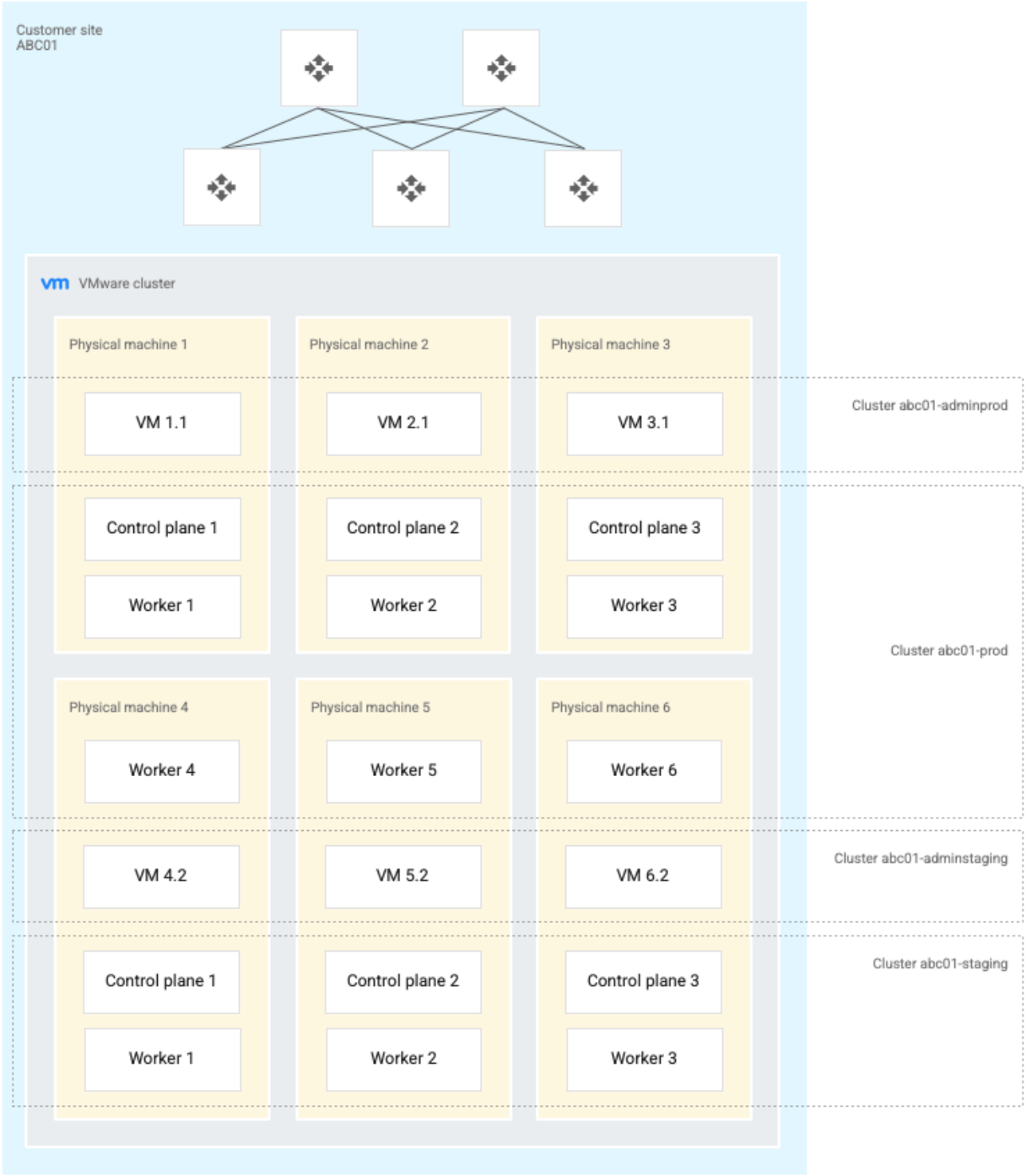
For simplicity of presentation, only one production cluster is shown per site. This design is a good starting point to bring a few large applications to production. As you bring more applications and lines of business to Anthos, it's useful to have several production clusters on a given site. Tradeoffs in the number of clusters per site are described in more detail in the section on Multi-tenancy.

Each cluster typically resides on a separate subnet and VLAN from other clusters. We suggest that you put all of a cluster's nodes on a single subnet and VLAN, although other configurations are possible.

The following diagram shows an example of an Anthos clusters on bare metal deployment:



When VMware or other virtualization is used, the ESXi hosts or equivalent can be spread across several racks. This approach enables the VMs to be spread across hosts, as shown in the following diagram:



For Anthos clusters on VMware, vSphere controls the VM placement. This management control might result in a less organized arrangement than illustrated. This behavior is normal.

Clusters and environments

Anthos works together with Google Cloud's resource hierarchy[2] to help apply policies, organize resources, and control access. Organization level policies and tags can then be applied to resources

and users of the Anthos-related projects. This approach reinforces consistent behavior with how cloud-only resources are controlled.

On-premises clusters are represented by cloud resources. Kubernetes-specific resources, such as Pods or Deployments, are typically not directly governed by Google Cloud resource hierarchy, but by Anthos Config Management and Policy Controller.

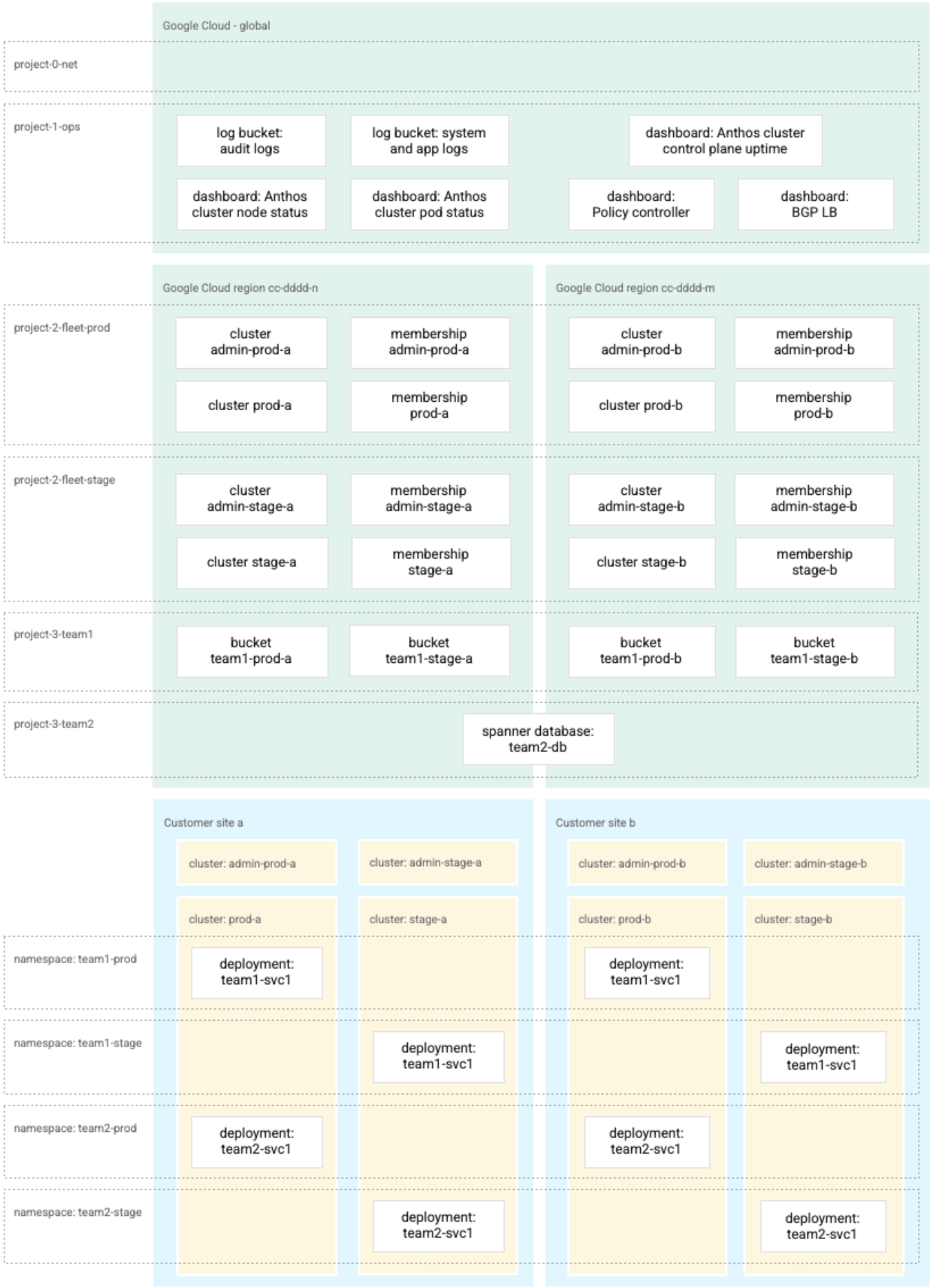
The following diagram gives a global view of logical cloud and on-premises resources that exist in this reference architecture. Resources are spread across several projects with appropriate identity and access management (IAM) controls applied. Also shown are examples of applications deployed across sites, namespaces, and clusters. By separating resources, you can provide more granular access to management capabilities. Resources spread across regions and locations help provide redundancy and provide resources close to where your users are.

In this diagram, resources in Google Cloud are organized into projects and locations as follows:

- `project-1-ops` holds operational data like audit logs, application logs, and dashboards. This project is a global project in Google Cloud that spans all regions.
- `project-2-fleet-prod` holds production clusters and identity information. This project spans two regions for redundancy or geo proximity to where you want applications to run.
- `project-2-fleet-stage` holds staging clusters and identity information. This project spans the same two regions as the production clusters and data.
- `project-3-team1` and `project-3-team2` are where individual teams can store data and run applications. Some teams might split between production and staging buckets and across regions, or some teams might use services with built-in replication across regions like Cloud Spanner.

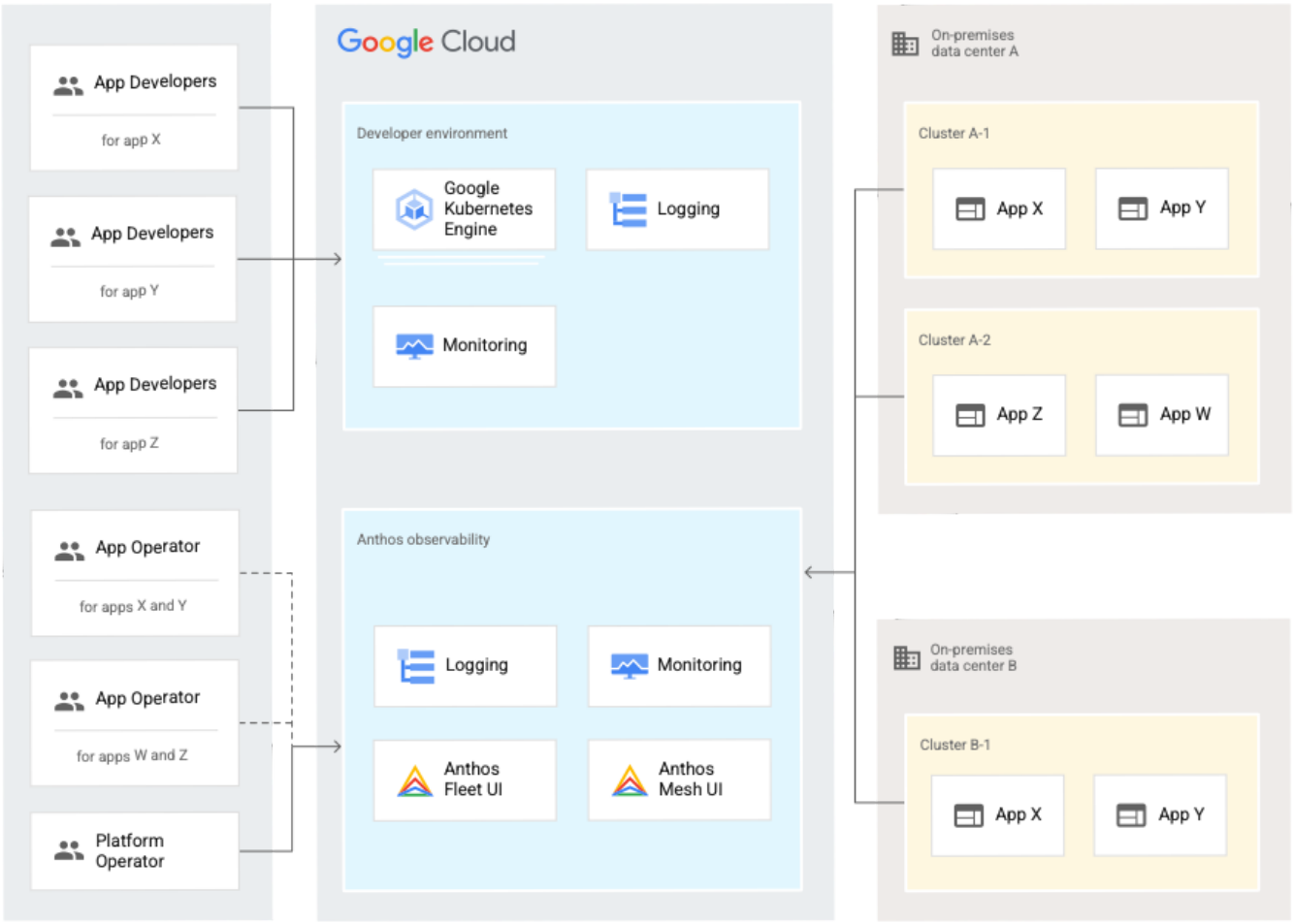
The diagram also shows resources deployed across namespaces in two customer sites as follows:

- Each site has admin production and staging clusters, like `abc01-adminprod` and `xyz01-adminstage`.
- Each site also has two user production and staging clusters, like `abc01-prod` and `xyz01-staging`. These clusters are where your application workloads run.
- Multiple namespaces like `team1-prod` and `team2-stage` exist across user clusters and sites for individual teams to deploy and run applications in both staging and production environments.


Google Cloud - global
project-0-net
project-1-ops
log bucket: audit logs
log bucket: system and app logs
dashboard: Anthos cluster control plane uptime
dashboard: Anthos cluster node status
dashboard: Anthos cluster pod status
dashboard: Policy controller
dashboard: BGP LB
Google Cloud region cc-dddd-n
Google Cloud region cc-dddd-m
project-2-fleet-prod
cluster admin-prod-a
membership admin-prod-a
cluster admin-prod-b
membership admin-prod-b
cluster prod-a
membership prod-a
cluster prod-b
membership prod-b
project-2-fleet-stage
cluster admin-stage-a
membership admin-stage-a
cluster admin-stage-b
membership admin-stage-b
cluster stage-a
membership stage-a
cluster stage-b
membership stage-b
project-3-team1
bucket team1-prod-a
bucket team1-stage-a
bucket team1-prod-b
bucket team1-stage-b
project-3-team2
spanner database: team2-db
Customer site a
Customer site b
cluster: admin-prod-a
cluster: admin-stage-a
cluster: admin-prod-b
cluster: admin-stage-b
cluster: prod-a
cluster: stage-a
cluster: prod-b
cluster: stage-b
namespace: team1-prod
deployment: team1-svc1
deployment: team1-svc1
namespace: team1-stage
deployment: team1-svc1
deployment: team1-svc1
namespace: team2-prod
deployment: team2-svc1
deployment: team2-svc1
namespace: team2-stage
deployment: team2-svc1
deployment: team2-svc1

Observability

The following diagram shows how application developers and application or platform operators can view logging and monitoring data in Google Cloud. The on-premises clusters and applications send this logging and monitoring data back to Google Cloud for analysis and review:



In this approach, different personas are granted access to only the environments they need. Applications that run in on-premises clusters direct logging and monitoring data back into Google Cloud for analysis and review.

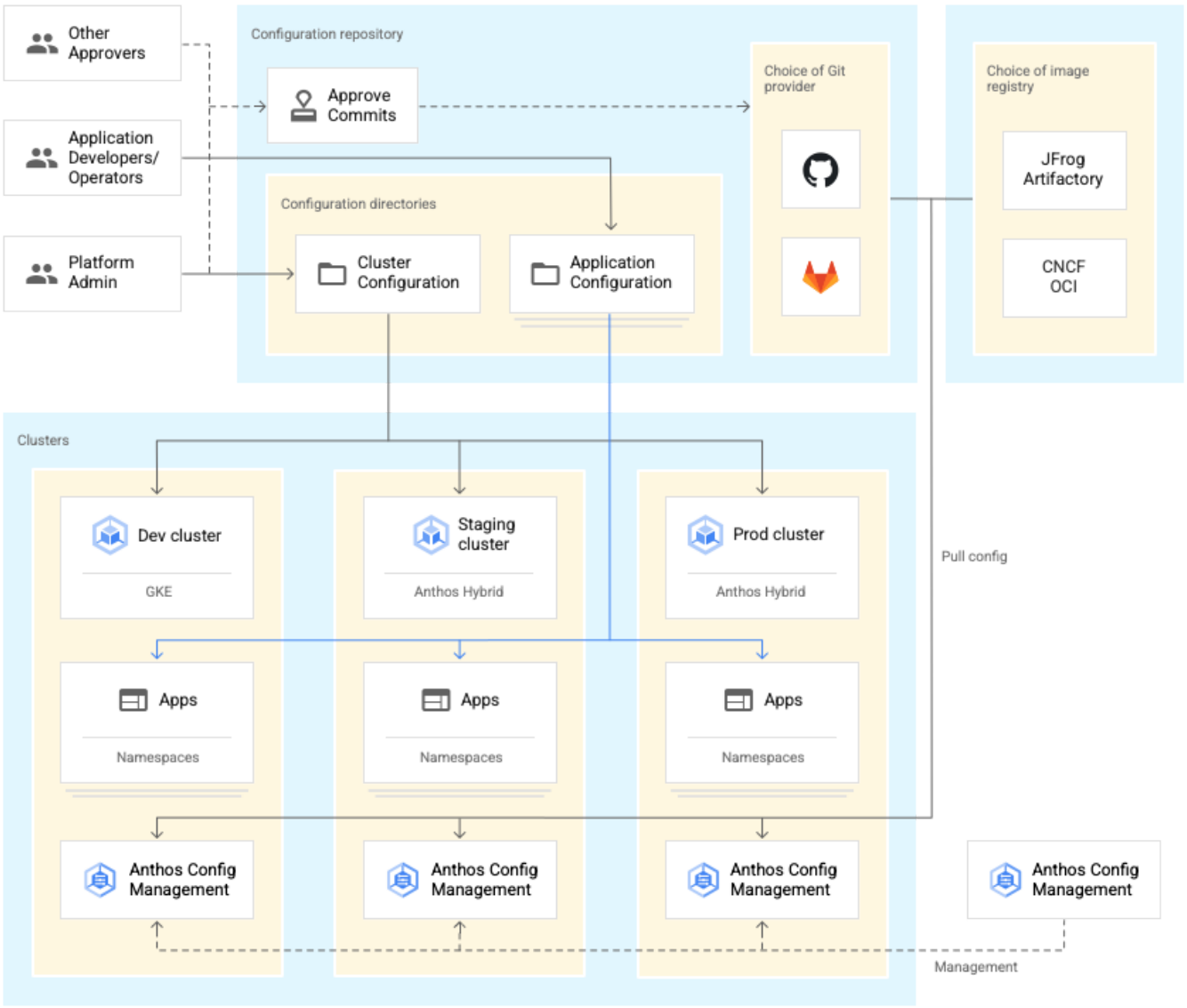
A following section in this reference architecture on implementation details provides more context and considerations to deploy this observability piece.

Configuration management

Anthos Config Management can be used to manage Kubernetes objects in all the clusters. Anthos Config Management is a GitOps-style tool[3] that uses a Git repository or Open Container Initiative (OCI)

image as its storage mechanism and source of truth. Git provider workflows allow multiple stakeholders to participate in review of changes.

As shown in the following diagram, a common Anthos Config Management deployment uses one folder containing configuration for all clusters. Separate additional folders each hold configuration data, one for application:



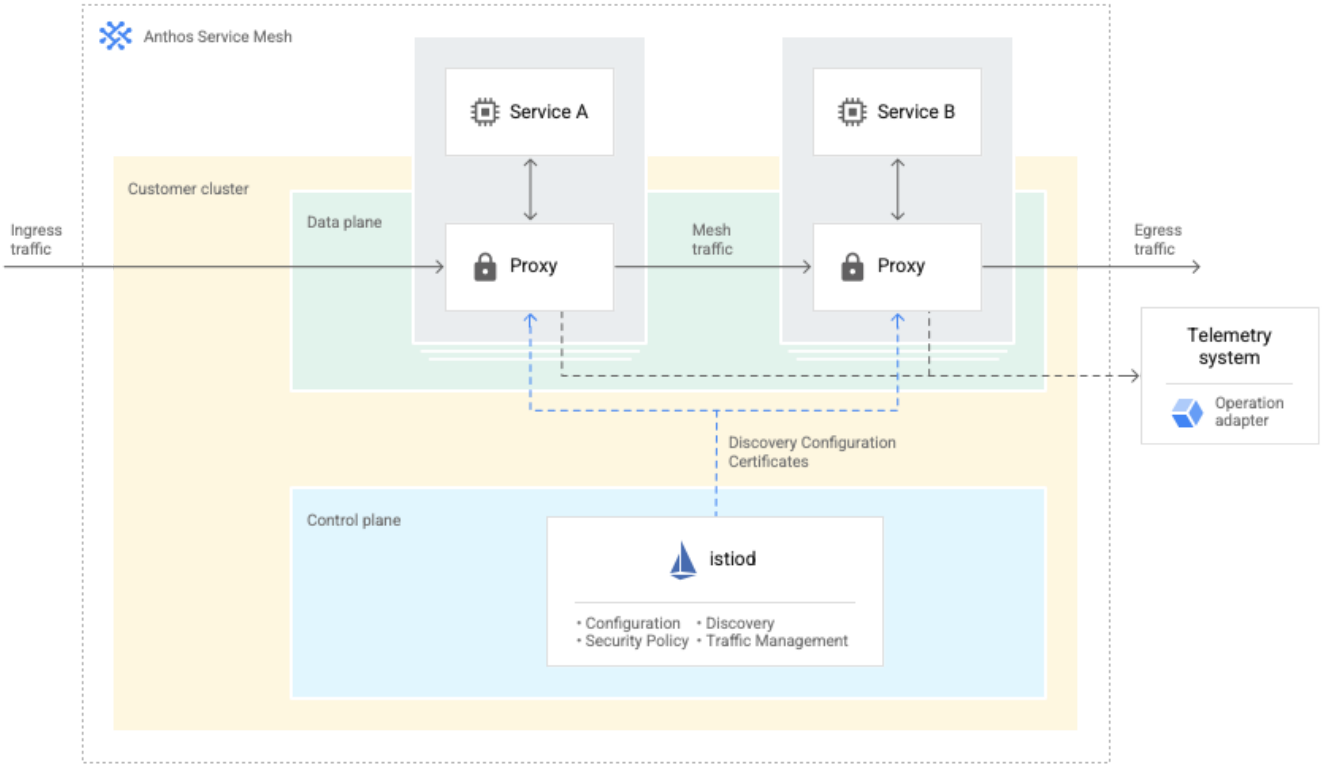
A following section in this reference architecture on implementation details provides more context and considerations to deploy this configuration management piece.

Security

The following services can help secure traffic as part of a deployment of this reference architecture. These services help with authentication, connectivity, and communication in a cluster:

- Anthos Identity Service connects clusters to on-site identity providers to authenticate local access.
- Connect gateway and workforce identity federation can provide secure cloud-mediated access to mobile workforce clusters without using a VPN.
- Workload Identity provides on-premises workloads with managed short-lifetime credentials for access to cloud resources.
- Anthos Service Mesh encrypts and controls communication between services in the same cluster.

The following diagram shows how Anthos Service Mesh can control the flow of traffic between services within a cluster:



A following section in this reference architecture on implementation details provides more context and considerations to deploy this service mesh piece.

Design prerequisites

To deploy Anthos in a hybrid environment that follows the guidance in this reference architecture, there are several design prerequisites that must be met. Prerequisites include deciding on Google Cloud regions and local sites to use, planning for the required on-premises hardware, and understanding the network and interconnectivity requirements.

This section of the reference architecture discusses the design prerequisites to deploy Anthos in a hybrid environment. Identify the key areas in this section that align with your own needs and goals, then verify that your environment meets the defined requirements.

Regions and sites

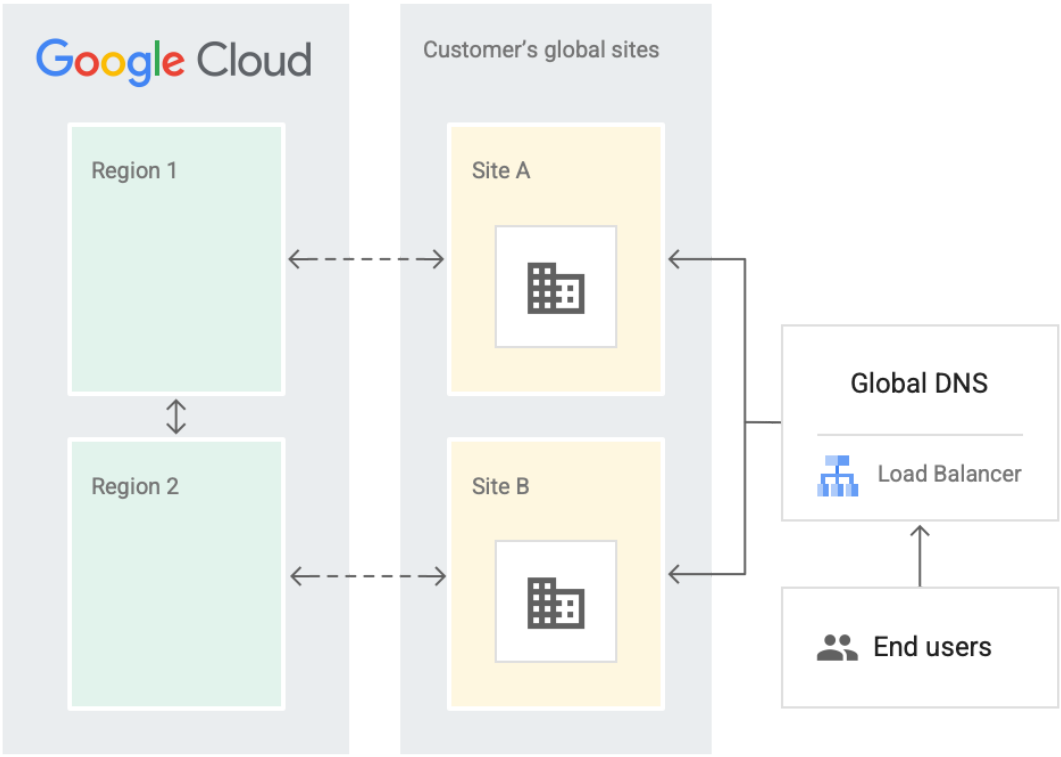
Identify several on-premises data centers, or *sites*, to run Anthos clusters. Sites may be data centers that you own, colocation facilities, or major offices or facilities with secure machine rooms. Each site typically has a dozen to hundreds or more physical machines.

To build and manage highly available applications, choose at least two separate geographic sites. You can choose more than two sites to support your required workloads and availability demands. Anthos hybrid environments typically have fewer than 50 sites.

Each site should have a short name. This reference architecture recommends a naming convention for sites of a city code plus two digits, like `nyc01`.

Choose at least two different Google Cloud regions which are proximate to the sites of your Anthos clusters. These sites run the associated Google Cloud services for the Anthos hybrid environment in this reference architecture.

The following diagram shows how you can connect using Cloud DNS and Cloud Load Balancing to Anthos clusters that run in two different on-premises sites. Two proximate Google Cloud regions provide additional Anthos services:



Google Cloud setup

Install the latest version of the Google Cloud CLI[4] and related tools, such as the latest available version of `kubectl` and `anthos-auth`. If you're behind a corporate proxy or firewall, you can reconfigure `gcloud` CLI[5] so it can successfully access the internet. You manage resources using different tools:

- Google Cloud resources are managed using the Google Cloud console, `gcloud` CLI, or Terraform.
- In-cluster resources are managed with Anthos Config Management. The use and configuration of Anthos Config Management is discussed later in this reference architecture.

Plan the Google Cloud organization and structure. Before you deploy Anthos, set up a Google Cloud organization and any folders that you want to use. Anthos projects should be placed within this folder hierarchy.

Plan the projects, naming conventions, and permissions to hold your Google Cloud resources. This reference architecture deploys resources across several projects. This approach lets different teams access and own the resources appropriate to them:

1. A host network project, such as `project-0-net`, that contains Google Cloud networking resources. A network operations team typically owns this project.
2. Two fleet projects, such as `project-2-fleet-prod` and `project-3-fleet-staging`, that contain Google Cloud resources which represent your on-premises clusters. These projects also include logging, monitoring metrics, dashboards, and alerts for your clusters and applications.

Depending on their permissions, users with access to this project can view, create, and update your on-premises clusters, and view logging and monitoring data.

3. Several application projects, one for each application or small set of related applications, such as `project-n-app-xyz`. The development team and the operations team for that application are granted access to this project. A central platform team can own and allocate the projects. This project also contains images and cloud CI/CD pipelines for the applications. As new applications are onboarded, you need a process to create these projects on-demand.

Plan your cluster names. Every cluster must have a unique name within the project in which it's registered. A consistent format for cluster names is recommended. This reference architecture uses the site code and an environment code in the cluster name.

On-premises setup

The following guidance helps you understand the prerequisites to design the on-premises parts of an Anthos hybrid environment.

Site services

Run a local container image registry in each site. This local registry ensures that images can still be pulled during network outages, and reduces image network traffic when applications scale up rapidly. Application configuration is also stored in an image registry. Use an OCI-compliant registry. Many popular image registries support the Open Container Initiative (OCI) Distribution Specification including Artifact Registry and JFrog Artifactory[6].

Provide Git access at each site. Some configuration is stored in Git. Anthos Config Management needs read-only access to your Git repository to read cluster configuration. Google Cloud doesn't provide or support a Git repository. A cloud-based Git provider may be used as the primary storage and for reviews. Use either a GitHub or GitLab for Enterprise account to ensure sufficient fine-grained access controls and API request rates. A Git mirror at each site can be provided if it's necessary to continue operations when there's an internet or Git provider outage.

Provide an identity provider. Federate your corporate ID provider to Cloud Identity, including the ability to create new groups in your identity provider as needed. Local and cloud-based identity providers are supported. Management operations may be limited if there's an internet outage or provider outage affecting a cloud-based identity provider.

Optionally run VMware. Anthos supports creating clusters based on the following types of infrastructure:

1. Physical machines, managed by you.
2. Virtual machines, on VMware and managed by you.
3. Virtual machines, by using integration with VMware.

When using the third option with this reference architecture, the VMware requirements are as follows:

Version	vSphere 7.0 Update 3 or later
License	vSphere Enterprise Plus

Compute requirements

Anthos offers two ways to run clusters in an on-premises environment - Anthos clusters on bare metal and Anthos clusters on VMware. Review the following compute requirements for each option. For advice on selecting one of these options, see the Design Considerations section.

The following sizes are suggested for deploying the eight clusters of the reference architecture described in a previous section. Other Anthos documentation might mention other node sizes. The sizes in the following table are for this reference architecture.

The following sizes and counts apply to both Anthos clusters on bare metal and Anthos clusters on VMware.

Recommended node sizes for this reference architecture:

Node config	vCPUs per node	RAM per node (GB)	Storage space per node (GiB)	Storage throughout per node (sequential IOPS)
Admin cluster node	4	16	128	50
Staging control plane node	8	32	256	50
Production control plane node	32	128	256	500
Staging and production worker node	4	16	128	Varies

Anthos clusters on VMware also require 380 GiB of storage for vCenter for VM templates, log and metrics buffering, and `etcd` object data.

Recommended node quantities for this reference architecture:

Node config	Clusters used in	Number needed	Total

Admin cluster nodes	`abc01-adminprod`
	`abc01-adminstage`
	`xyz01-adminprod`
	`xyz01-adminstage`
		3
		3
		3
		3
			12

Staging control plane nodes	`abc01-staging` `xyz01-staging`	3 3	6
Production control plane nodes	`abc01-production` `xyz01-production`	3 3	6

Staging and production worker nodes	`abc01-staging`
	`xyz01-staging`
	`abc01-production`
	`xyz01-production`
		3 -10
		3 -10
		10 - 500
		10 - 500
			26 - 1020

All			50 - 1044

The number of worker nodes varies with application needs. Production clusters usually have ten or more nodes. Up to 500 nodes per production cluster are supported in this configuration. Staging cluster size depends on how application workloads are scaled down in the staging environment and how traffic is generated or directed to the staging environment. The requirements might be one tenth of the production requirements, but should be at least three nodes.

Operating system

Select a primary operating system for your Anthos clusters from one of the following:

- For Anthos clusters on bare metal:
  - Public Ubuntu (20.04)
  - RHEL (8.2, 8.3, 8.4, 8.5, 8.6)

- For Anthos clusters on VMware:
  - Container-Optimized OS[7] from Google
  - Ubuntu from Google

Anthos clusters on VMware also support worker nodes with Windows Server 2019 10.0 and later.

Storage

For Anthos clusters on VMware, use VMware-supported storage such as VMFS or vSAN.

For Anthos clusters on bare metal, use NAS/SAN storage from one of the following supported storage providers:

- Dell EMC
- Hitachi
- NetApp

Networking

The following network requirements should be considered as part of your own deployments that follow this reference architecture.

IP addresses

- IPv4 network addresses are used for machines and for service and pod IP ranges.
- For Anthos clusters on VMware, make sure you plan your IP address needs[8].

Internet connectivity

- An internet connection is needed for operational purposes.
    - Firewalls must allow outbound connections to certain Google Cloud services[9]. Google Cloud won't make direct inbound connections.
    - Anthos uses the following two patterns to communicate with Google Cloud:
        - TLS connections initiated to a Google Cloud service, such as logs that are sent to a logging service.
        - Some Anthos management services connect to Anthos on-premises clusters by using a tunnel (the Connect Agent). This tunnel also appears to customer firewalls as an outbound TLS connection.
    - Your employees and automation solutions may connect to clusters over this tunnel.
    - The Anthos install and update processes pull images from a Google Cloud service.
- Use of a perimeter network for ingress traffic might not be necessary.
    - Traffic only reaches applications when explicitly configured to do so. This traffic includes containerized workloads (Pods) and VMs workloads (using Anthos for VMs on bare metal clusters).
- Use of an outbound HTTP proxy is common.
    - You can run an HTTP CONNECT-based proxy, like Squid, between your cluster and the rest of the internet to limit connections to known domains.
    - Size the proxy to handle peak outbound traffic. This traffic includes container image pull during applications scale up, if there is no registry behind the proxy.

Intercluster connectivity

- All clusters need outbound connectivity to Google Cloud such as for the connect agent or logging and monitoring.
- For Anthos clusters on bare metal:
    - Machines in a cluster need to be routable to each other (L3 connected).
    - Machines can span VLANs to span multiple power and networking failure domains.
- For Anthos clusters on VMware, VMs must be on the same L2 domain.
- We recommend running in island-mode for IPv4.
    - When using dual-stack networking, flat-mode is used for IPv6.

Intracluster connectivity

- All the machines in a single cluster need to have L2 or L3 connectivity with each other.
    - For Anthos clusters on bare metal, ensure L2 or L3 connectivity between machines, and from the admin cluster to the machines.
    - For Anthos clusters on VMware, the VMware Distributed vSwitch or standard vSwitch must be configured to provide L2 connectivity between dynamically provisioned VMs.
        - Set your precreated port groups to the correct VLAN ID[10].
        - Make sure that the L2 subnet (VLAN) is correctly configured and accessible by all ESXi hosts within the chosen vSphere cluster. Also check that the core routers are interconnected.
- It's not recommended to create network security boundaries between node pools in a single cluster.
    - You might have regulations that require network isolation between applications, such as in-scope for PCI vs not in scope for PCI. If so, consider using multiple clusters if the applications don't require direct connectivity.
    - Alternatively, consider using Anthos `NetworkPolicy` and Anthos Service Mesh to control traffic between applications within a cluster.
- All the VMs that are part of your Anthos clusters on VMware infrastructure must use the same Network Time Protocol (NTP) server.

Basic network

You can choose software-defined networking (SDN) solutions so long as the core networking configurations are correct. For example, consider the following basic network functionality:

- The IP address, netmask, and gateway all work correctly.
- ARP or NDP messages are honored.
- DHCP, DNS, and NTP servers work correctly.
- Internet gateway or proxies also work correctly.

Depending on different organization structure, all those basic requirements might require cross-team collaboration.

In Anthos clusters on bare metal, you manage the machines. These machines can be in different VLANs (L2 network) so long as they're L3 connected. There can't be any NAT or proxy between L2 networks.

In Anthos clusters on VMware, the software stack manages the VM lifecycle. All the nodes in the same cluster connect to the same virtual portgroup, so they're L2 connected. As a result, Anthos clusters on VMware can enable direct routing mode for pod-to-pod connectivity, while Anthos clusters on bare metal must use tunnel mode.

The following table summarizes the basic network requirements:

	Anthos clusters on bare metal	**Anthos clusters on VMware**
Customer-managed network	L2 and L3	L3
L2 / switch	Your choice	VMware Distributed Switch
Node placement per cluster	L3 connected	L2 connected
Pod connectivity mode	Geneve tunnel	Direct routing (tunnel disabled)

Load balancer

Global load balancing is also called a DNS load balancer or Global Traffic Manager. If you have an existing global load balancing solution, continue using it. Anthos doesn't integrate with the global load balancer.

Site-level load balancing requirements:

- If you have an existing F5 Local Traffic Manager or other load balancers such as HAProxy or Citrix AD, you can continue using it. Use the manual load balancing mode during setup of your Anthos deployment.
- If you don't have an existing Local Traffic Manager, use the MetalLB bundled load balancing. MetalLB is available for both Anthos clusters on bare metal and Anthos clusters on VMware.

The following table provides an overview for load balancing modes for ingress traffic to your Anthos clusters. The HA control plane load balancing is automatically selected at installation:

Goal	**Cluster type**	**Mode to use**
Use Anthos-provided load balancer	Either	MetalLB
Use an existing load balancer	Either	Manual

On-premises to Google Cloud connectivity

All on-premises clusters connect to a Google Cloud region through the connect gateway, a special-purpose network tunnel service. The session is initiated from within the on-premises network. Connect gateway uses a gRPC / TLS tunnel, and only supports HTTP access to the Kubernetes cluster API. The tunnel doesn't provide general access to the on-premises network.

Design considerations

This section helps as you design your own implementation of this reference architecture. These design considerations help you implement the most appropriate Anthos hybrid environment for your needs, and understand areas such as security and compliance, operations, and observability.

Hybrid deployment

Anthos offers two ways to run clusters in an on-premises environment - Anthos clusters on bare metal and Anthos clusters on VMware. Which one to choose is based on your product requirements. The following table details support of some functionality across the two types of clusters:

Functionality	Anthos clusters on bare metal	Anthos clusters on VMware
Performance in general	Yes	Yes
Hypervisor overhead	No	Yes
Customize a supported node OS	Yes	No
Automated node lifecycle	No	Yes
Automated OS lifecycle	No	Yes
Cluster autoscaling	No	Yes
Cluster auto repair	No	Yes
High availability	Yes	Yes
Legacy VM workloads	Yes (A4VM)	No
Automated volume allocation for stateful container workloads	No[11]	Yes
Large scale deployment	Yes, with removing additional hypervisor cost	Yes, with on-demand scale up and down
Highly performance-sensitive workloads	Yes	No
Storage options	External storage	External storage or VMware vSAN or VMFS

The choice between Anthos clusters on VMware on Anthos clusters on bare metal is up to you. Existing deployments, investments, and requirements often help you determine which one is the most appropriate choice.

When you might use Anthos clusters on VMware:

- **vSphere investments**: If you have considerable vSphere investments and want to continue with your investment on vSphere. This guidance includes when you have deep operational knowledge of vSphere and plan to continue that investment.
- **vSphere operations:** You might have operationalized vSphere to provide virtualized compute, networking, and storage technologies to your customers, either as self-service or IT standard. If you want to leverage and continue to use this operational model, Anthos clusters on VMware would be ideal.
- **Node management:** You might want to automate all aspects of node management and to provide those capabilities as self-service to their customers. Anthos clusters on VMware are an ideal choice as it provides an automated lifecycle for nodes and OS, autoscaling and repair, and high availability.
- **Scalability and density:** If you have densely packed Kubernetes nodes and autoscaling of clusters, Anthos clusters on VMware are a better option. This approach provides better bin packing, autoscaling of nodes, and hitless planned downtime.
- **Workload characteristics:** Anthos clusters on VMware are a better fit for workloads that are highly stateful, need higher availability against planned and unplanned downtimes, and can run on hardware platform-independent virtualization layers. They're also ideal for workloads that use software defined storage (SDS) as their block store.

When you might use Anthos clusters on bare metal:

- **Cost optimization:** If you want to optimize on cost, use Anthos clusters on bare metal as you don’t have to incur the cost of vSphere licenses.
- **Operating system support:** If you want to bring your own OS due to company or regulatory requirements. This approach provides more options for OS supportability.
- **Workload characteristics:** If you run high performance and latency sensitive applications that can't tolerate hypervisor overhead. This solution is also ideal for workloads which need hardware dependent features like SR-IOV or CPU pinning.
- **VM workload support:** If you want to run workloads running in virtual machines together with containers. This approach supports both containers and VMs managed through a single Kubernetes stack.

Availability

We recommend that you have multiple environments such as development, staging, and production. These environments let you adequately develop, test, and deploy applications in a more controlled way than a single deployment to production. Production clusters should use highly available (HA) clusters.

We recommended that staging clusters are HA to let you test application availability under simulated infrastructure failures. Development environments and similar can be non-HA.

Anthos workloads automatically move within a cluster in response to hardware faults. For this behavior to be effective, consider the following:

1. If one machine fails, is there enough spare capacity in the cluster to allow all workloads to fit on the remaining machines?
2. If there isn't enough spare capacity, how long will it take to get a replacement machine?
3. For Anthos clusters on VMware, vSphere handles physical machine failure. Anthos clusters on VMware use VM anti-affinity rules to ensure VMs, like the control plane or three-node node pool, are spread across three different ESXi hosts. Make sure that your vSphere instance can tolerate the loss of one node.
4. For Anthos clusters on bare metal, you need a monitoring solution to detect and react to a failed physical machine.

To mitigate single-points of failure at a rack level, machines from several racks can be used to form a cluster:

- For Anthos clusters on bare metal, a cluster can use machines spanning multiple racks.
- For Anthos clusters on VMware, a cluster can use ESXi hosts placed across racks.
    - VMware will typically spread VMs across hosts.
- Neither cluster type should span sites.
    - For Anthos clusters on VMware, don't use VMware Stretched Clusters.

Anthos is designed to ensure that applications continue to operate despite a temporary disconnection from Google Cloud[12], such as a cloud service or network connectivity outage. Applications continue to run on the same nodes, and cluster networking and load balancing operate normally. Although unlikely, it's possible a physical machine might fail during a temporary disconnection event.

Consider the behavior of replacement Pods that start on other machines when a physical machine fails, concurrently with a temporary disconnection from Google Cloud:

1. Application container images might be stored in another site or in a cloud service such as Artifact Registry. Machines would be unable to pull the image to start the replacement process. For this reason, run a local image repository in each site that mirrors your primary image registry.
2. An HA control plane can tolerate the loss of one node when the physical machine is part of the control plane. The loss of additional nodes on several independent physical machines is unlikely to occur within the timeframe to replace the first failure. This potential problem can be further mitigated by running a local registry with a mirror of the Anthos system container images[13].
    a. For Anthos clusters on VMware, use *thick images* that contain system container images. Anthos clusters on VMware load these images into your local registry.

   b. To replace a failed machine in Anthos clusters on bare metal, Anthos needs to install certain additional OS packages. Consider running a local OS package repository[14]. This approach can also be useful to remove a machine for troubleshooting.
3. If it's necessary to add worker nodes to the cluster, they may not be able to start without pulling certain container images. Running a local registry mirror ensures they can start.

Consider the availability of authentication services. Anthos Identity Service bridges to third-party authentication providers for login over local networks. Anthos Identity Service doesn't have a cloud dependency. However, Anthos Identity Service can be configured to use a cloud-based identity provider, such as Azure AD. If there's a network outage between your site and the internet, you might not be able to authenticate to your clusters.

Network security and management

Consider when to use customary network security practices like VLANs and firewalls, and when to use Anthos networking features.

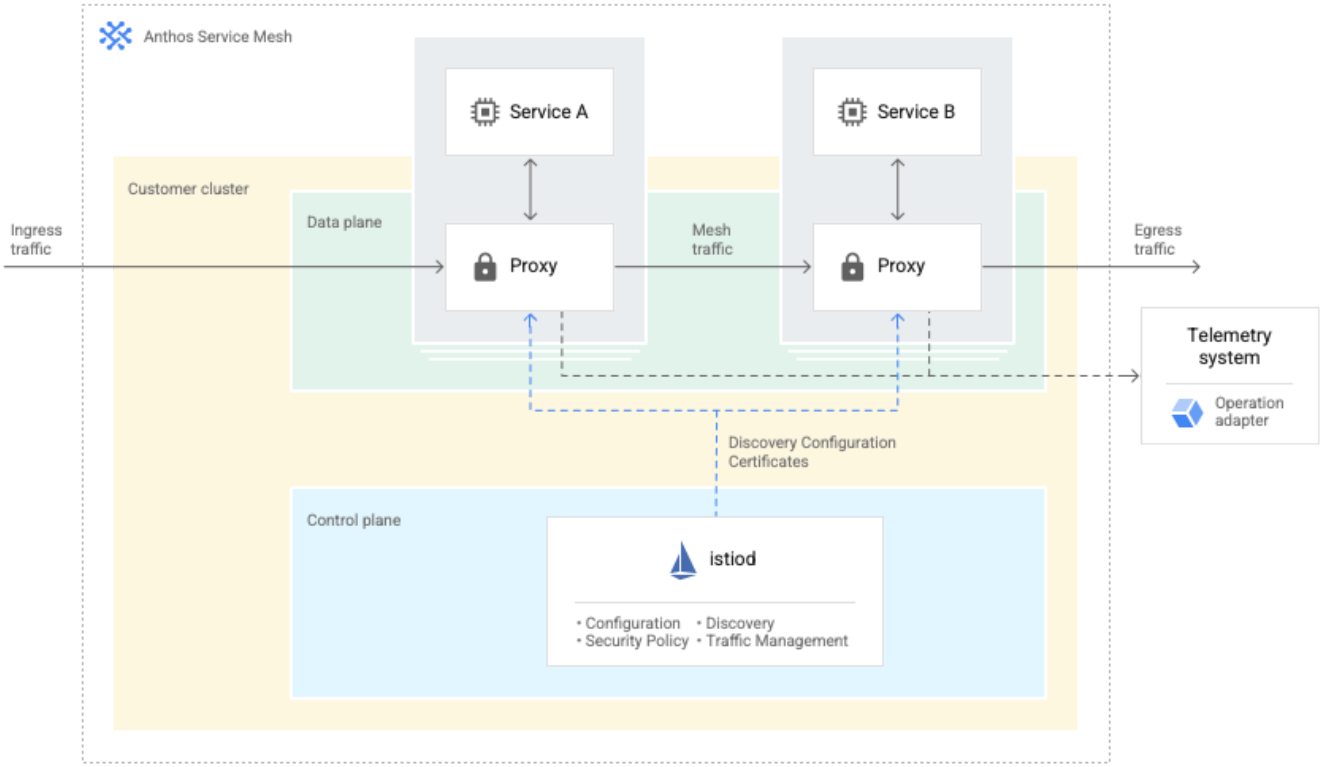
Customary network security practices are recommended for separating clusters from other clusters or from other parts of the network, and include the following suggestions:

- Place the admin clusters on an administrative subnet and VLAN.
  - Although this separation isn't required by Anthos, separating management traffic from non-management traffic is a commonly accepted practice.
  - Place all nodes of an admin cluster on the same VLAN.
- Place user clusters from different environments or with different security requirements on different subnets and VLANs from each other.
  - Although this separation isn't required by Anthos, separating production and non-production environments is a common practice.

In a user cluster, you can use Anthos networking features to replace the removed boundaries when previously separated applications are consolidated onto a single cluster. Some factors to consider including the following:

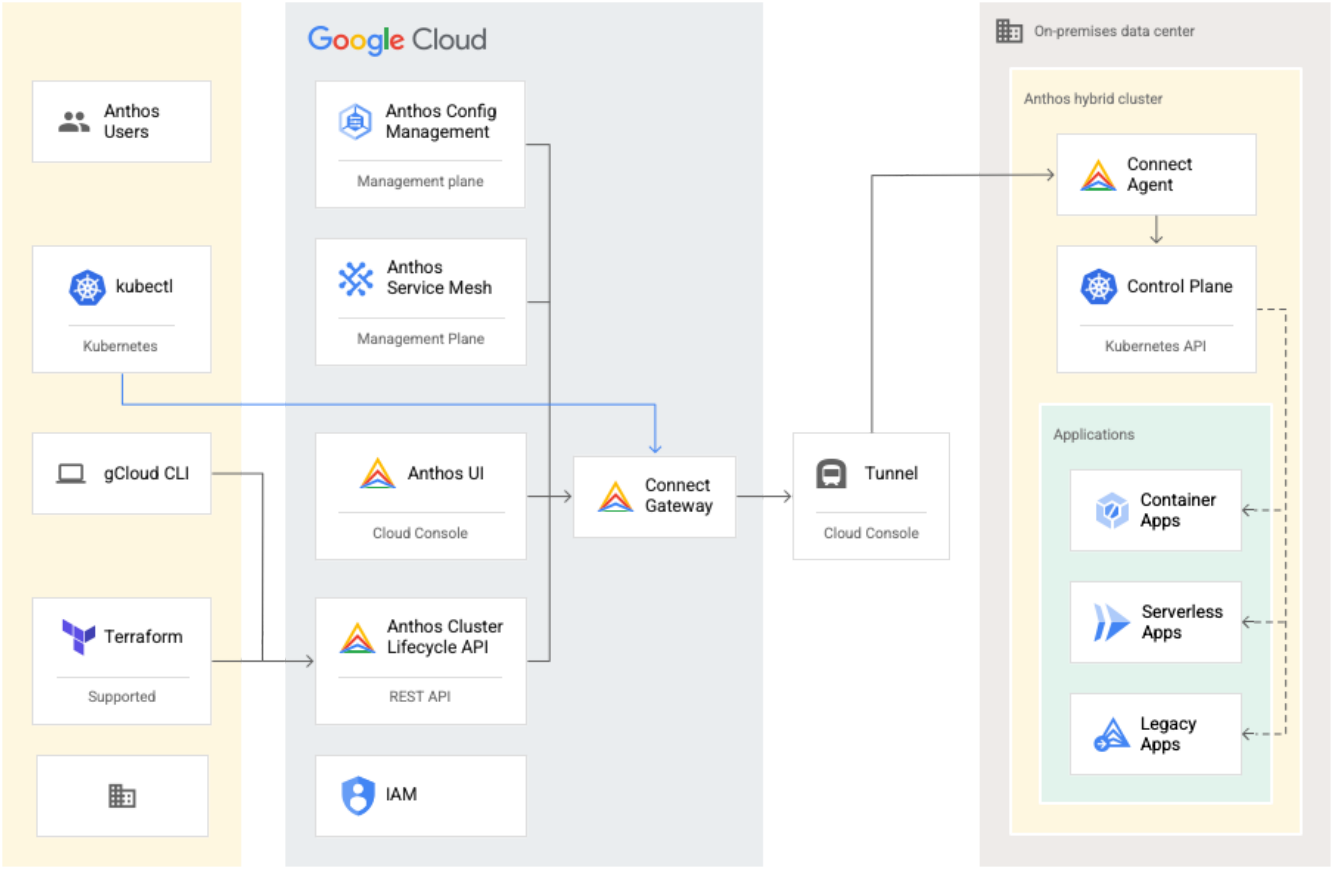
- Containers in a cluster can communicate to each other by default. Communication within a cluster can be restricted using Kubernetes Network Policies, or with Anthos Service Mesh.
- Several user clusters can be placed on the same VLAN.
  - In this case, isolation can be achieved with separate namespaces and the use of Kubernetes Network Policies.
  - This approach can make reallocation of machines between uses easier for Anthos clusters on bare metal.

The following diagram shows how Anthos Service Mesh can control the flow of traffic between services in the same cluster:



When using Anthos Service Mesh to secure service traffic within a cluster, you should also secure external traffic. Don't allow direct connections to clusters for management operations by users or automated processes. Instead, connect to clusters by using the private on-premises network or through the connect gateway.

The following diagram shows how all operations can use the connect gateway to connect to clusters using a secure tunnel:



Scale and limits

The following scale and limits apply to deployments of this reference architecture. Take these values into consideration for your own design.

Cluster and node limits

The following scalability limits apply when all the recommendations in this reference architecture are followed. These limits apply to Anthos clusters on bare metal and Anthos clusters on VMware. Other documentation may mention lower limits for more general use cases, or higher limits for more specific use cases.

Clusters	Total per fleet	250
	Admin clusters per site	1-2 *recommended*
	User clusters per admin cluster	100
Pods	Per node	110
	Per cluster	15,000
	Changed per second	20 *created / modified / deleted*

Nodes	Per cluster	500
	Per node pool	200
Services	Per cluster	10,000
	Per cluster, type `nodePort OR LoadBalancer`	250 *total of both types, average of ≤ 60 endpoints per service*
	In a single namespace	5,000

Load balancing limits

Anthos clusters configured in accordance with this reference architecture and that use MetalLB load balancing support clusters with the following maximums:

Maximum flows per Service	Up to 30,000
Maximum VIPs per cluster	1,000
Failover time (min-max)	2 seconds - 15 seconds

Manual load balancer performance varies, but typically meets or exceeds the preceding limits.

Anthos Config Management limits

Use multiple repositories, at least one root repository and one repository for each namespace. This approach isolates faults to a smaller surface, provides better use of repository access controls, supports more objects, and reduces sync latency. The following additional scalability limitations apply:

Total resources in any one root or namespace repository *Mean resource size = 3 kB*	4,000
Number of clusters managed by Anthos Config Management	250
Number of namespaces repositories	2,500

Observability limits

The following observability limits apply to deployments of this reference architecture:

Logging throughput per node	100 KB per second per node
Number of teams with restricted logging access *Limit applies to the number of logViews in the operations project, which provides access to a group for a subset of logs.* *No limit on number of teams with no or full logs access.*	25

Multi-tenancy

When you deploy a new application, consider whether to place the application in a new namespace in an existing cluster, or to create a new cluster. The former mode is called *multi-tenant,* and the latter are *single-tenant clusters*. *Tenant* means an application developer team. This term is not the same as a customer of a SaaS application provider.

Advantages of multi-tenant clusters include the following:

- Reduced resource fragmentation.
- No need to wait for cluster creation for new tenants.
- Easier to configure communication between services in the same cluster.
- Cost-effective to provide high-availability for small workloads.
- Lower administrative effort with fewer clusters.

Advantages of single-tenant clusters include the following:

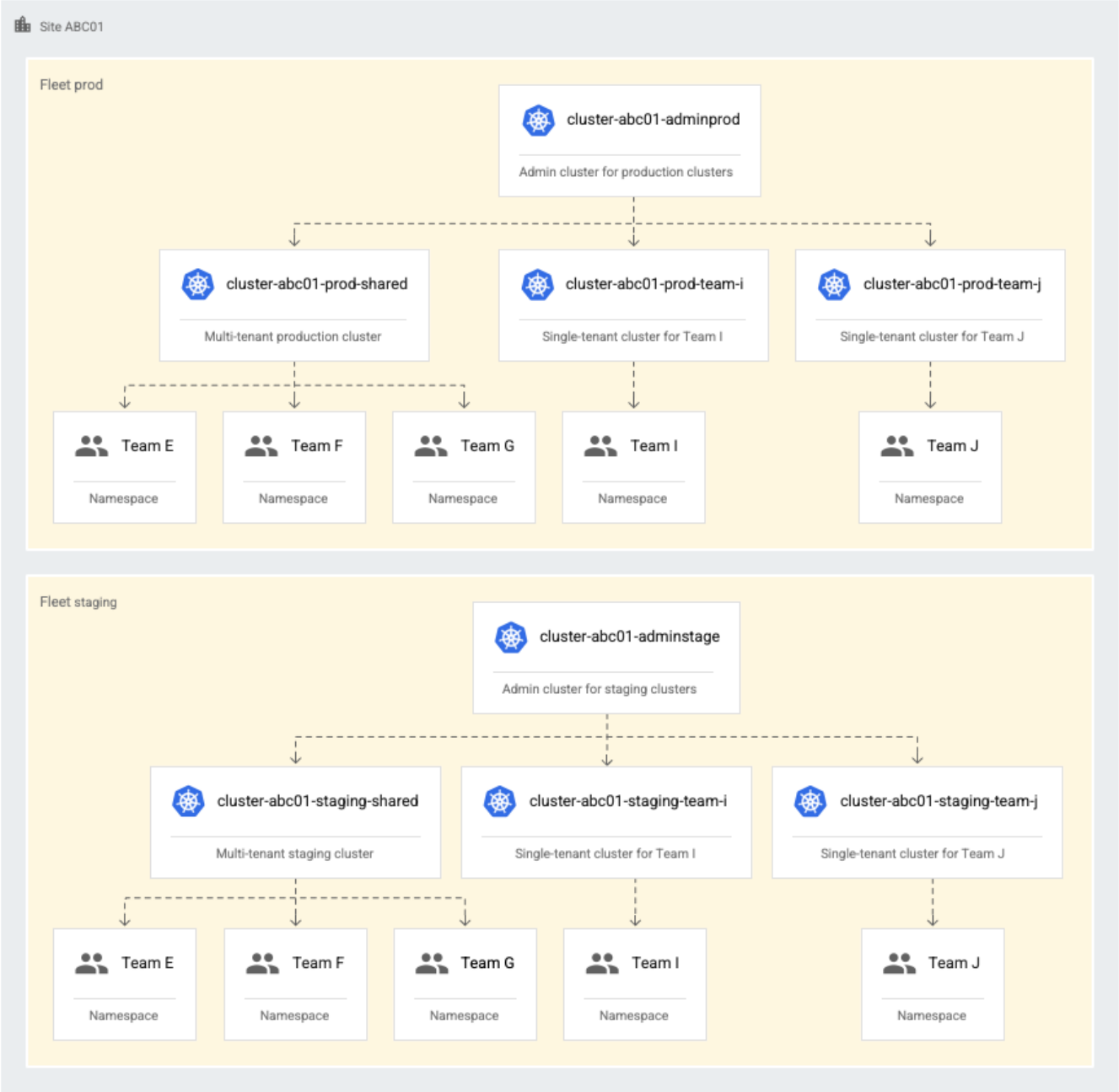
- Chargeback can be based more directly on hardware costs.
- Delegates management of cluster capacity and application priorities to an application team, such as a single-tenant cluster that runs many batch jobs throughout the day.
- Allows application teams to manage their own API extensions, or *operators*, without affecting other tenants.
- Allows application teams to manage namespaces when dynamically creating applications, such as in a SaaS application.
- Separates noisy-neighbor applications from other applications.
- Provides an extra security boundary for applications that run risky code.
- Placing an entire cluster within a network boundary to meet compliance requirements.

The following are examples of common deployment patterns:

- **One multi-tenant cluster for small workloads and a few single-tenant clusters for critical workloads.** It's common to have a few large applications, and many small applications. The operational overhead of a single-tenant cluster is justified for large workloads, but might not be for small workloads. For example, a large, highly replicated search application with hundreds of Pods might run in a single-tenant cluster per site, such as `cluster-abc01-prod-search`. Tens or hundreds of miscellaneous small services might then run in a multi-tenant cluster per site, such as `cluster-abc01-prod-shared`.
- **Separation of workloads to meet compliance requirements**. For example, site `abc01` might have two production clusters: `cluster-abc01-prod-pci` for PCI-DSS-compliant workloads, and `cluster-abc01-prod-nonpci` for other workloads. These clusters use different network configurations for the two sets of machines.
- **Separation of workloads by business unit**. Business units might own specific machine configurations, or want to be charged back at the granularity of machines. In this example, each business unit can have its own production clusters.

In the preceding example deployment patterns, a single admin cluster is sufficient for all production workloads. It's still recommended to have a corresponding set of staging clusters before you deploy workloads to the production clusters. The same trade offs apply to teams using Google Cloud clusters[15].

The following diagram illustrates how deployments look with the addition of some single-tenant clusters:



As more user clusters are added on a site, the number of admin clusters remains fixed at two.

Observability

Anthos distinguishes between system and application observability data. System observability data includes logging and monitoring of Anthos system components. System observability data is sent to the site's corresponding Google Cloud region.

This reference architecture recommends, and assumes, that application observability data is sent to the nearest Google Cloud region. Consider the following when deciding whether to send application observability to Google Cloud:

- Sending logging and monitoring to a 3rd party logging solution might be a smaller change to current practices.
- Sending logging data to a local logging store might offer a more familiar path to meeting compliance requirements. However, Cloud Logging supports several features for meeting compliance requirements[16]. These features include encrypted logs storage with Customer Managed Encryption Keys (CMEK), External Key Managers[17], Access Transparency, and Access Approval.
- Preconfigured logging dashboards[18,19] are available for Anthos on Google Cloud.
- Prometheus compatibility for Cloud Monitoring.
- There's a cost for observability data that's ingested and stored in Google Cloud[20].

Multi-cluster management

Anthos can help increase developer velocity, reduce cost, reduce deployment risk, and increase reliability. To fully achieve these benefits, you typically have multiple clusters at a single site.

Clusters can be resized as needs change, or the hardware can be used to create short-lived clusters for specific tasks. In addition to having a large production cluster, multiple small clusters are useful for staging and development environments, and separating regulated workloads.

Staging environments, which are similar to the production environment, reduce deployment risk through representative testing and QA. Access to developer environments increases deployment velocity through earlier identification of system integration issues.

You might need to request a quota increase[21] if you plan to have more than 50 clusters in one fleet[22].

Gitops-based configuration management

When you first explore Anthos operations for a proof of concept, you can create and update clusters using the Google Cloud console. This approach provides guidance and validation of parameters at each step.

As you gain experience, switch to using declarative configuration to create and update clusters. The Google Cloud Terraform provider[23] supports creating on-premises clusters, and clusters created by the Google Cloud console can be imported into Terraform. Terraform communicates with a Google Cloud service to validate the commands and securely communicate with your admin clusters.

You can also use Anthos Config Management to declaratively configure the resources inside a cluster. These resources include Namespaces, CRDs needed for operators to be installed in the cluster, and Kubernetes RBAC roles and bindings.

When a declarative configuration is adopted, the steps to create a new cluster are as follows:

- Define, plan, and apply a new cluster in Terraform.
- Define in-cluster files for Anthos Config Management. You can define these by copying and customizing from a base configuration, or generation from a template with substitution.

Store both the Terraform .tf files and Anthos Config Management .yaml files in one or more Git repositories. If stored in a single repository, .tf files cluster creation and .yaml files for Anthos Config Management can be reviewed as a single pull request.

Application availability

Unlike some virtual machines, containers don't use live migration if there's an infrastructure failure. Therefore, applications should be built to have redundant containers within the same cluster. This approach allows the application to tolerate failure of or maintenance operations on a single machine. Applications that need the highest level of availability should be built spanning multiple sites as well.

The following table outlines the minimum recommended application replication:

Availability requirement	Application type	Redundancy model	In-cluster redundancy	Cross-site redundancy
Low	Stateless	None	Single pod	Single site
	Stateful	None	Single pod	Single site
Medium	Stateless	Active-active	Three pods	Single site
	Stateful	Active-passive	Two pods	Single site
High	Stateless	Active-active	Three pods minimum, typically more	Multi-site
	Stateful	Active-active	Three pods minimum, typically more	Multi-site

Examples of applications with different availability requirements:

Availability requirement	Examples
Low	A developer testing environment
Medium	Asynchronous data processing, or internal services
High	Websites and services used by customers

Implementation details

To help you successfully deploy an Anthos hybrid environment that follows this reference architecture, this section contains some important implementation details. These details include the suggested Anthos Config Management repository structure, recommended project and cluster permission and role assignments, and application namespace examples.

Per-site preparation

Services

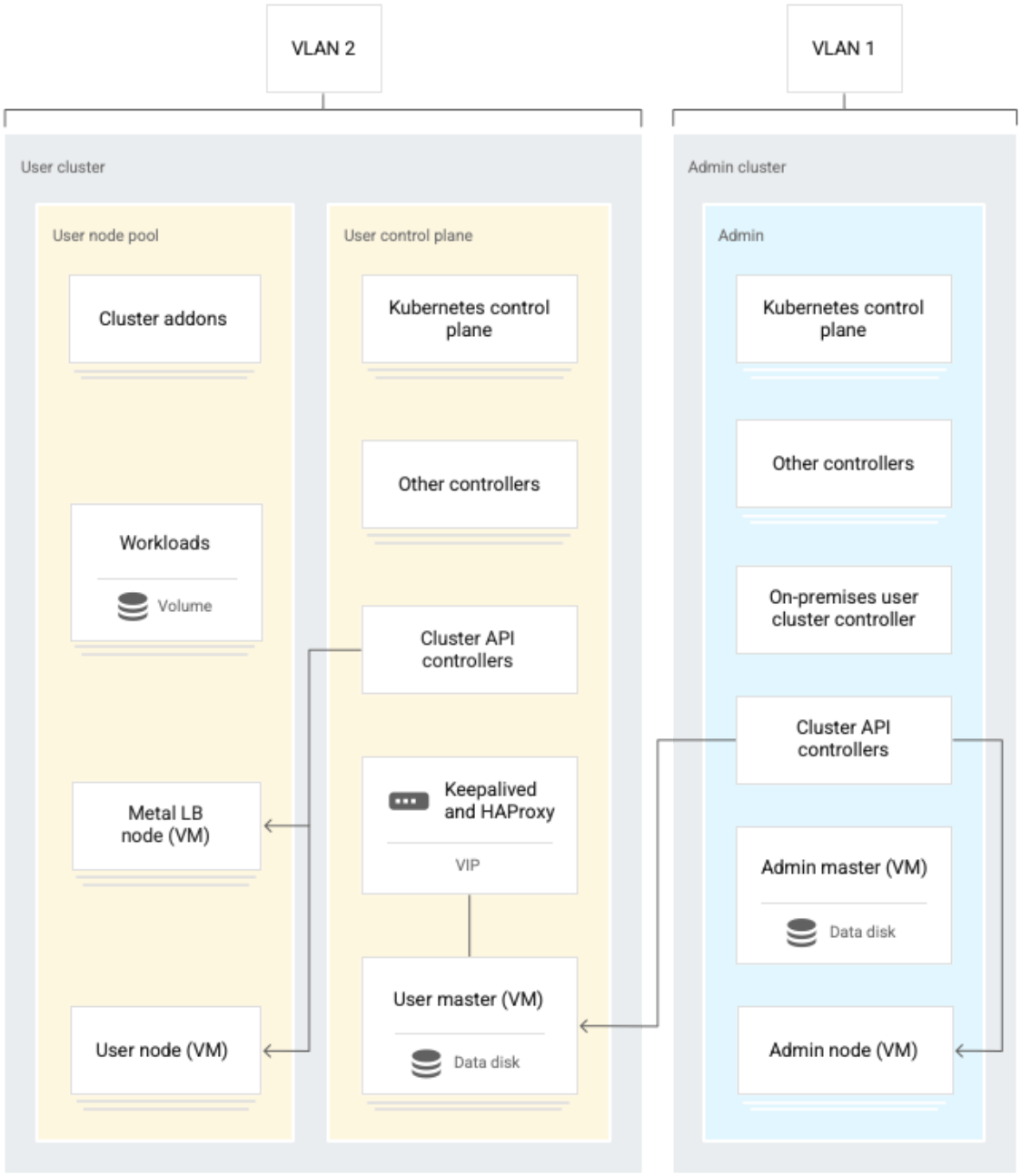
Create an image repository in each site, such as using JFrog Artifactory or Harbor. This site-level repository isn't part of the Anthos deployment or supported by Google. If you use Artifact Registry, you could set up the site to replicate from the cloud. Or, you could replicate between local repositories across sites. This approach stores artifacts close to where they're needed for deployments, and lets you replicate a single source of truth across all sites with similar local repositories.

Networking

Place each admin cluster on its own VLAN. Place each user cluster on its own VLAN. The following guidance also applies:

- For Anthos clusters on VMware, all nodes of a user cluster should be on the same broadcast domain (one subnet and VLAN).
- For Anthos clusters on bare metal, it’s simpler if all the nodes of a user cluster are on the same broadcast domain. Multiple domains may be used if necessary, with the following additional guidance:
    - All the load balancer nodes must be on the same broadcast domain.
    - Routing traffic between the L2 domains is your responsibility. Anthos doesn't configure this routing.
- Avoid using VLAN-per-node-pool as a security mechanism for separating individual applications in a cluster. This configuration can be complex to manage, and can limit the cost-efficiency benefits otherwise expected from moving to containers. Instead, use Kubernetes Network Policies and Anthos Service Mesh L7 Authorization to meet isolation requirements.
- All virtual IP addresses (VIPs) must be in the load balancer machine subnet and routable to the gateway of the subnet.

The following diagram shows an example of how the admin cluster and user clusters should be in their own VLANs. The user control plane and user node pool components share the same VLAN. The user cluster components communicate to the admin cluster and VLAN through the admin cluster API controllers:



Dataplane v2

Use Dataplane V2 in your deployments. GKE Dataplane V2 is a data plane that's optimized for Kubernetes networking, and is based on eBPF on Linux and Open vSwitch on Windows nodes. Dataplane V2 lets you flexibly process network packets in-kernel using Kubernetes-specific metadata.

Handling high traffic

If a site-level load balancer like F5 is available and a cluster is expected to handle a lot of traffic, balance traffic across several VIPs of the same cluster. Use MetalLB to balance within the cluster.

Avoid using only a single ingress to handle large numbers of backend services, such as 2,000 services. Instead, create several ingresses.

Configure vSphere

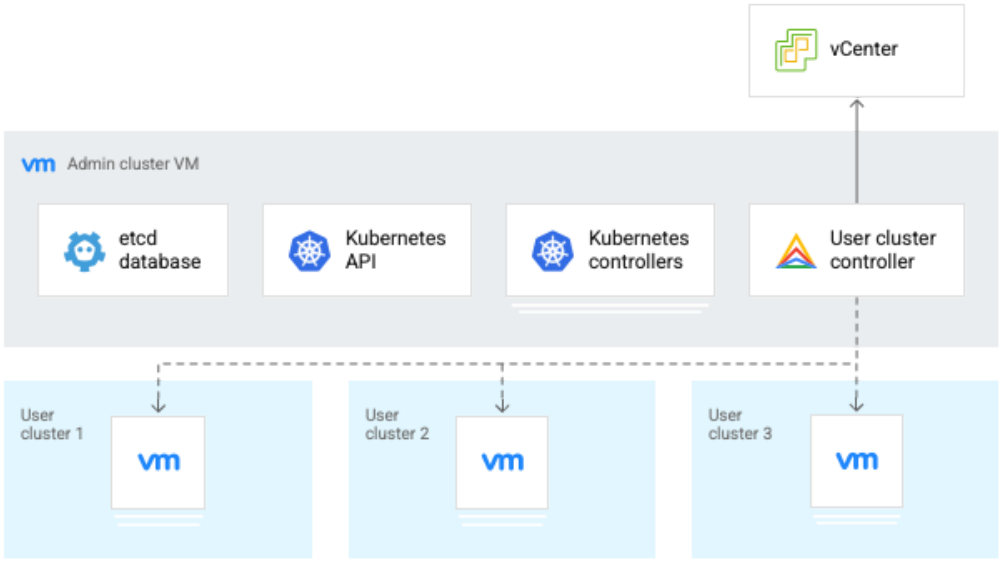
When using vSphere with Anthos clusters on VMware, configure as follows:

vCenter user account privileges	vCenter Server Administrator role is *not* required for Anthos deployment after the vSphere environment is set up. For Anthos deployments, we recommend creating several roles with varying degrees of privilege to limit access to your vCenter environment[7].
vCenter settings	• Enable vCenter High Availability (HA) • Enable vMotion • Enable vSphere HA Host Monitoring with `Host Failure Response` set to `Restart VMs` • Disable vSphere Storage DRS

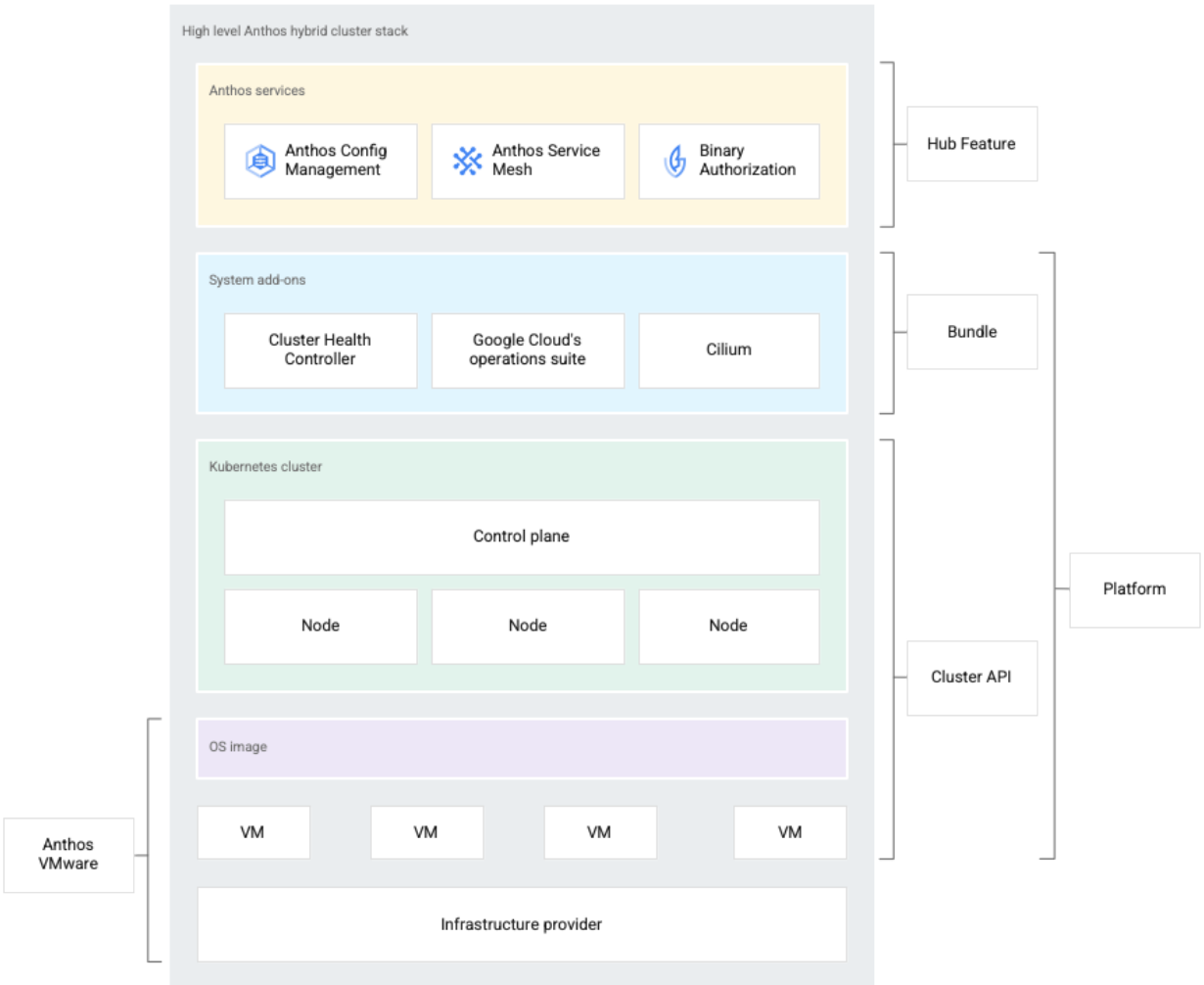
Use the following Anthos options when creating Anthos clusters on VMware:

- `enableControlplaneV2: true`
- `enableDataplaneV2: true`
- `antiAffinityGroups.enabled: true`

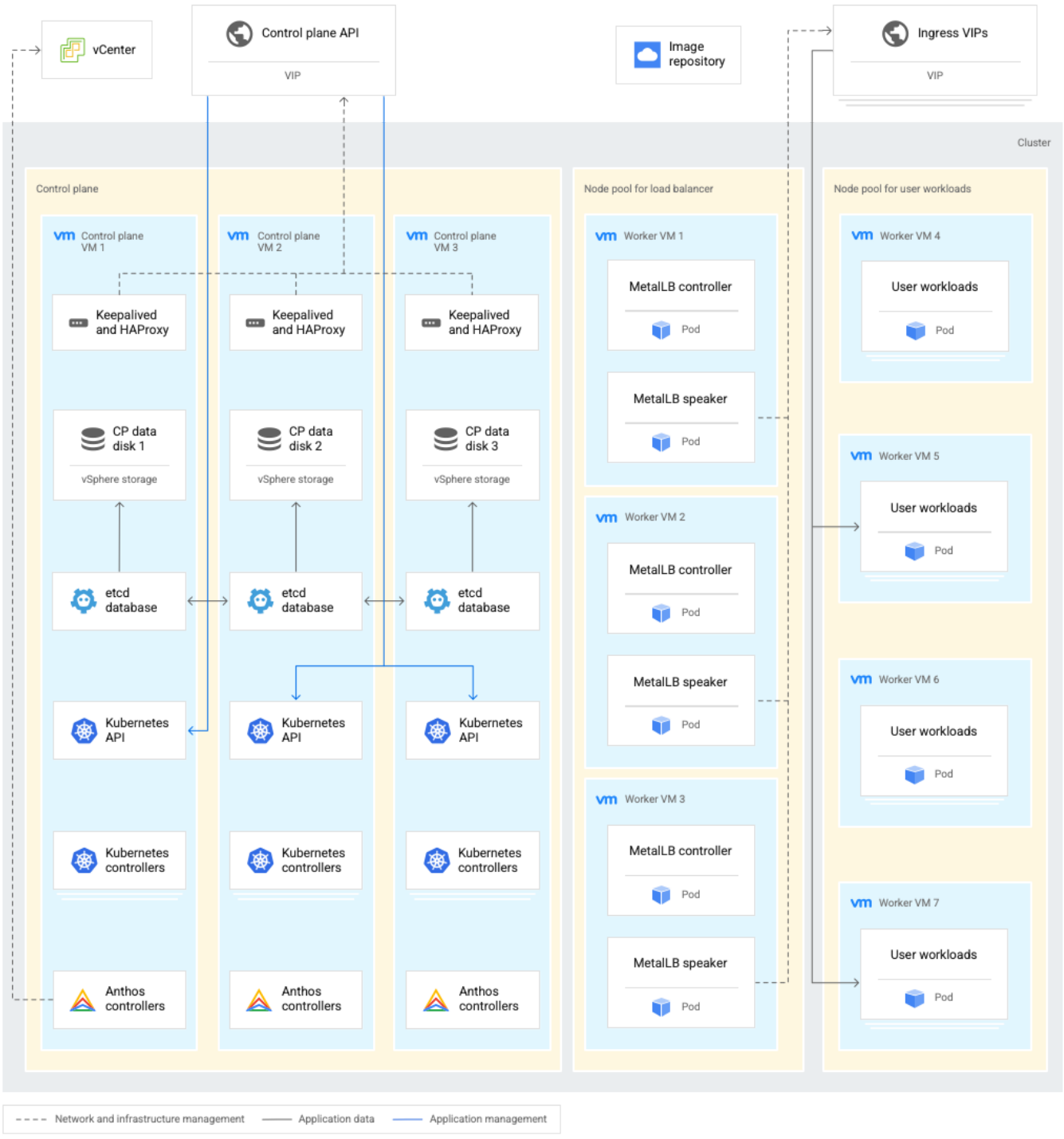
The following diagram provides an overview of how Anthos clusters on VMware look in a deployed state. The user cluster controller in the admin cluster communicates with vCenter. This connection lets the controller create the user cluster VMs:



The following diagram shows a more complete example of Anthos clusters on VMware deployed in a hybrid environment. Additional services like Anthos Config Management, Binary Authorization, and Operations Suite supplement the on-premises Anthos services that run in VMware:

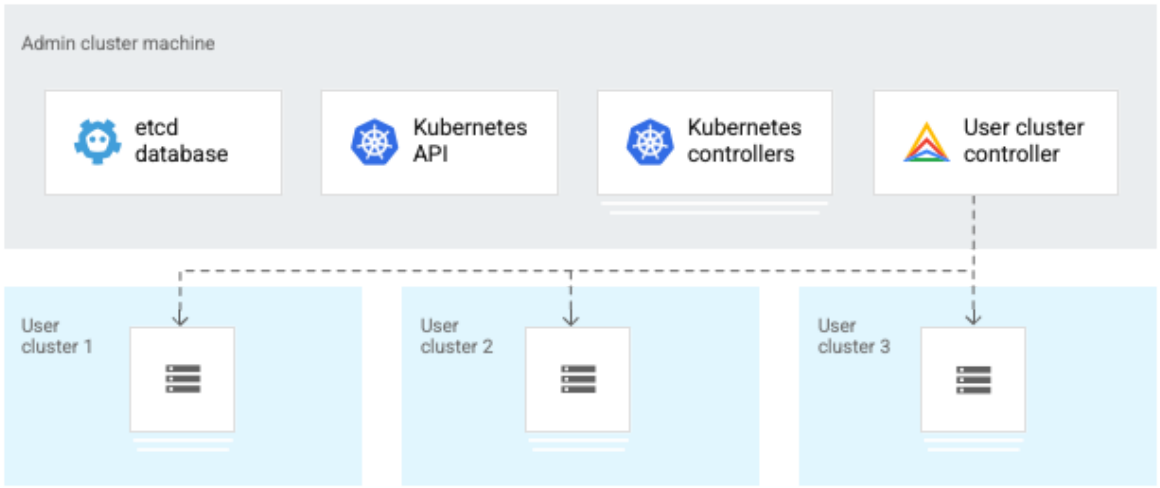


The following diagram shows that the control plane VIP sends traffic to the Kubernetes API. Keepalived / HAproxy keep the control plane VIP pointed at a working control plane VM. Node pools are created by a controller in the user cluster that talks to vCenter. The control plane VMs are created by the admin cluster. There's a dedicated node pool for MetalLB pods, but using the same VLAN as the other node pool. The MetalLB pods communicate using ingress VIPs, which also provide inbound traffic for the user workloads:

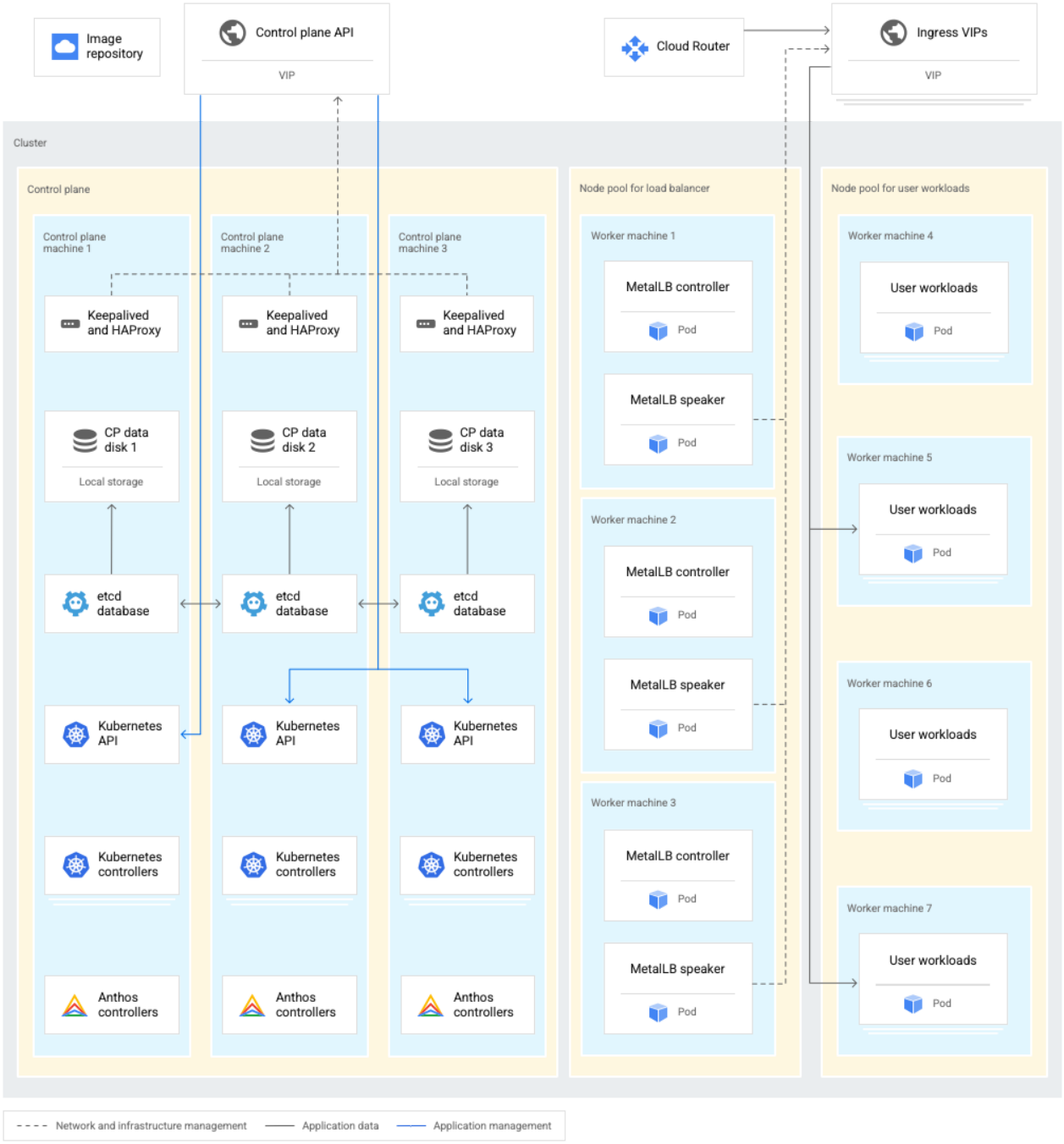


Bare metal deployments

The following diagram provides an overview of how Anthos clusters on bare metal look in a deployed state. The user cluster controller in the admin cluster communicates with the physical machines in your environment. This connection lets the controller create and manage the physical user clusters:



The following diagram shows that the control plane VIP sends traffic to the Kubernetes API. Keepalived / HAproxy keep the control plane VIP pointed at a working control plane machine. Node pools are created by a controller in the user cluster. The control plane machines are created by the admin cluster. There's a dedicated node pool for MetalLB pods, but using the same VLAN as the other node pool. The MetalLB pods communicate using ingress VIPs, which also provide inbound traffic for the user workloads:



Fleet management

- Place the prod clusters and their admin clusters in the production fleet, `project-2-fleet-prod`.
- Place the staging cluster and its admin cluster in the staging fleet, `project-3-fleet-staging`.

Hardware

Anthos can run on a wide range of customer-provided hardware. When you run on virtual machines, they can be easily sized to their roles.

If you deploy Anthos on physical machines, it can be more efficient to include at least some medium machines instead of all large-sized machines. Large machines are those that include more RAM and number of vCPUs than medium machines. Medium-sized physical machines have better utilization when they run as user cluster control plane nodes and admin cluster nodes.

Operating system

For Anthos clusters on bare metal, the base operating system on the nodes is customer-managed. Only install those OS packages that are prerequisites for Anthos, are needed to monitor the hardware and operating system, or to debug issues.

Because applications are containerized, they largely don't depend on libraries or services that run on the base operating system. Instead, application dependencies such as libraries are managed at the container image level.

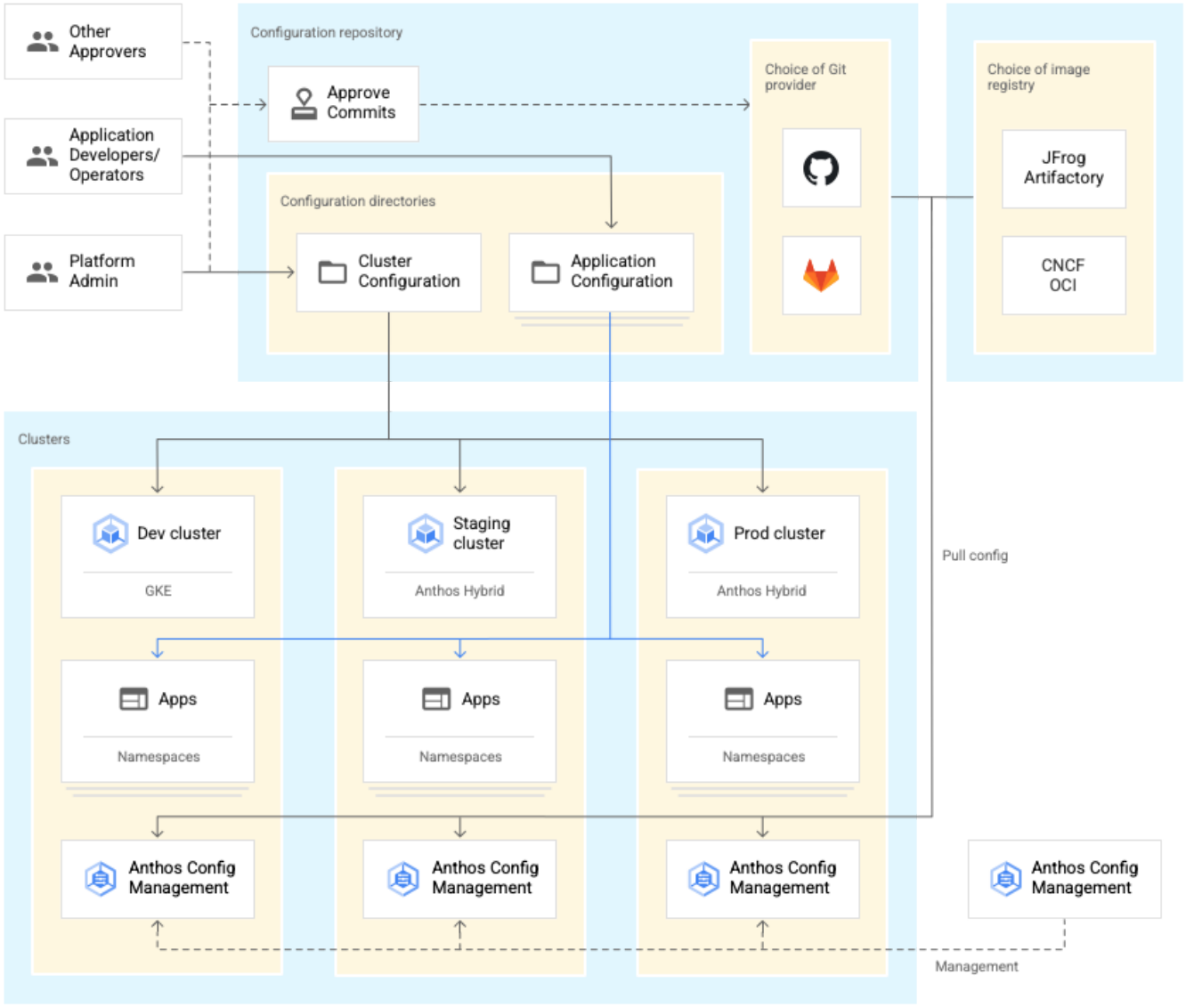
Anthos Config Management

Anthos Config Management is used to manage Kubernetes objects in all the clusters. Anthos Config Management is a GitOps-style tool. GitOps is a process where a Git source control repository is the source of truth for configuration. Git provider workflows allow multiple stakeholders to participate in review of changes.

Anthos Config Management runs an agent in every cluster that pulls config directly from Git. It can also pull bundled files from an image registry.

Anthos Config Management can pull configuration directly from a Git repository. It can also pull from an OCI image repository[24] or a Helm repository[25].

As shown in the following diagram, the recommended Anthos Config Management deployment uses one folder containing configuration for all clusters. Other individual folders each hold configuration data for one application:



An Anthos Config Management component per cluster pulls configuration directly from a source control repository or an image registry. A cloud management plane configures data sources and Anthos Config Management version.

Anthos Config Management manages in-cluster resources. Resources that affect the whole cluster (cluster-wide or cluster-scoped) are managed differently from resources that only affect one namespace (namespace-scoped). The latter are used to define applications.

Different categories of resources each have a set of tooling:

- Google Cloud resources are managed with the Google Cloud console, `gcloud` CLI, or Terraform.
- In-cluster resources are managed with Anthos Config Management.
    - Anthos Config Management doesn't manage admin clusters and their resources.

This reference architecture uses a specific subset of Anthos Config Management features and mode. Additional features exist and can be used where appropriate. In particular, this reference architecture uses the following choices in the use of Anthos Config Management:

- Uses unstructured mode.
- Doesn't use cluster selectors or namespace selectors. Therefore, cluster labels also aren't required.
- Doesn't use Config Controller to manage cloud resources.
- Configuration can be specified and delivered to each cluster as a template with parameters substituted with per-cluster values. Or, use a fully rendered, or *hydrated*, configuration to each cluster.
    - A fully hydrated configuration allows for users to understand what the desired state is, without having to mentally run a transformation process.

Cluster configuration

Platform owners define what policies are deployed to all clusters. Policies include the following:

- **PodSecurity admission controller:** Includes preventing Pods from using the root Linux user.
- **NetworkPolicies:** Control the network traffic inside your clusters.
- **ClusterRoles and ClusterRoleBindings:** Control permissions within a cluster.
- **Anthos Service Mesh:** Includes policies such as for authorization, transport security, or security policy constraints.
- **Anthos Config Management:** Install the Policy Controller security bundle for Anthos Service Mesh[26].
- User permissions (RBAC) granted cluster-wide.

The configuration for each cluster is stored in a Root Repository, also known as a RootSync. The Root Repository configuration includes both literal configuration resources, and pointers to other repositories holding application resources.

Application configuration

Platform owners can control the specifics of each application, or delegate control to application owners.

Applications repositories, also referred to as *Namespace Repos* or *RepoSyncs*, configure resources within a single namespace, including the following:

- Workloads such as Deployments and StatefulSets.
- VolumeClaims
- RBAC permissions scoped to namespace, such as Role and RoleBinding.
- Services
- Anthos Service Mesh configuration for services in this namespace.

This reference architecture offers a choice of two patterns for deploying application configuration. Each approach has advantages and disadvantages:

- Helm templates:
  - Simple to set up.
  - Uses familiar templating patterns.
  - Templates can become complex to reason about over time.
  - Parameters substituted by Anthos Config Management in each cluster.

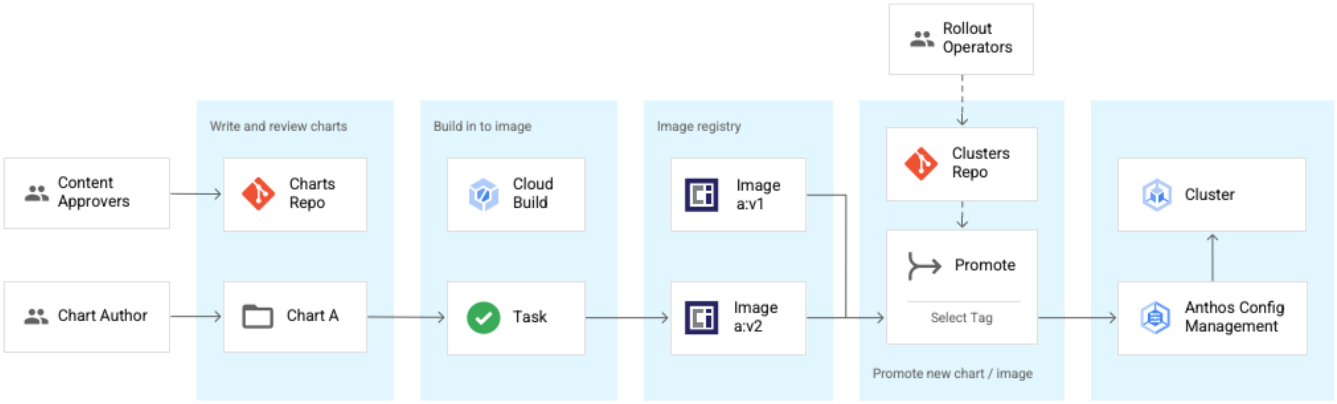
- Fully hydrated configuration:
  - More steps to set up.
  - Can use templating, or other config customization tools like `kustomize` or `kpt`.
  - Validation tools can run directly on the configuration during the review process.
  - Requires setting up a rendering pipeline that reads and writes to Git.
  - Both *dry* and *hydrated*, or wet, configurations are stored in Git.
    - Dry means configuration that uses templates and doesn't have per-cluster or per-app customizations applied.
    - Hydrated, or wet, mean configuration with all template parameters substituted and per-cluster and per-app customizations applied.
  - Storing both types of config in Git has the following advantages:
    - The Git repository is a source of truth, and you don’t need to mentally run a transformation process to be sure of the desired state.
    - More flexibility in what processes can be used.
    - Greater scope to validate configuration during the review process.

Helm templates

The Helm templates approach is appropriate for teams that already use the popular open source Helm tool to define applications. In the Helm approach, the following benefits and management approaches can be used:

- Application owners specify their applications as Helm Charts, with a limited number of parameters (`Values`).
- Platform owners control which application instances are deployed in which clusters and namespaces using the Root Repo.
- Use Helm templating language to parameterize charts per cluster.
- Platform owners control parameter values by cluster.
- Platform owners control promotion of new chart versions (rollout across clusters).
- Anthos Config Management pulls hydrated config from Git and charts from an image registry. Helm configuration and container images can be stored in the same registry.
- Platform policy is validated before deployment, and again when applied to the cluster.

The following diagram shows how you can write and review Helm charts that can build images to be stored in a registry. You can then deploy the Helm carts to the cluster using Anthos Config Management:



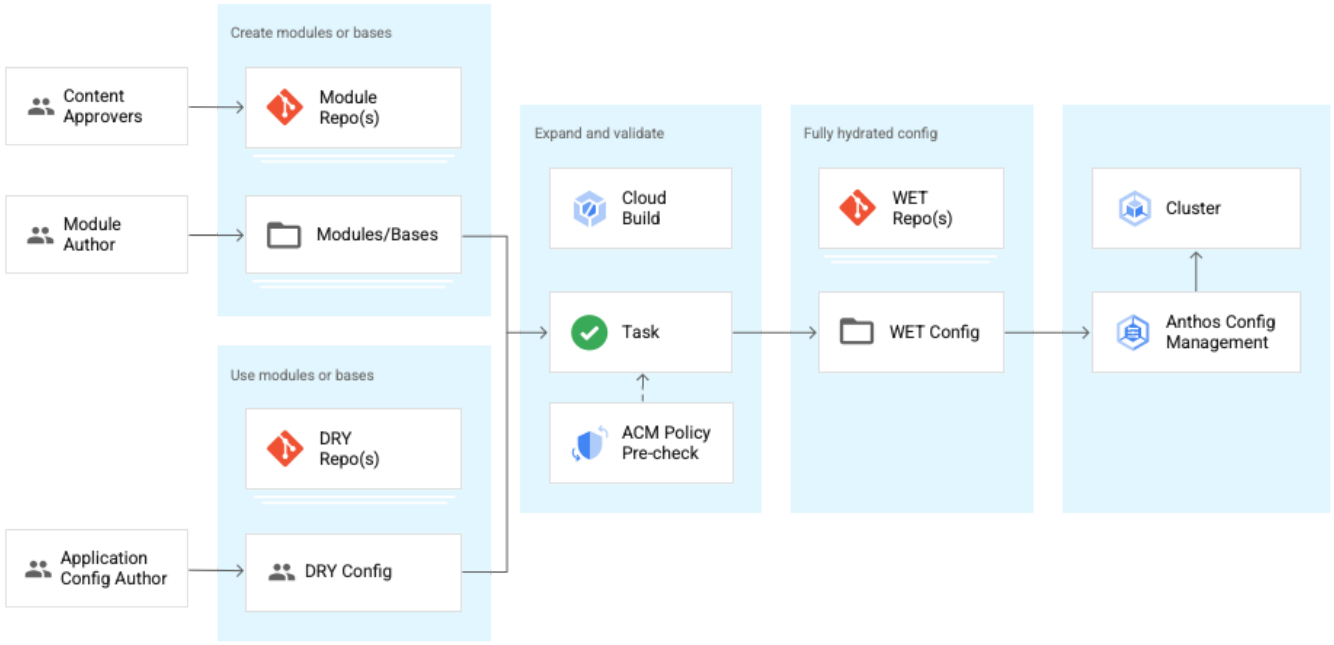
Fully hydrated configuration

The fully hydrated pattern works well when a central platforms team wants to create modules and allow other teams to modify them, subject to code review.

In the fully hydrated approach, the following benefits and management approaches can be used:

- Platform owners define a base application definition.
- Applications define instantiations with changes to the base definition using the platform owner's choice of packaging, templating, or customization tool, such as `kustomize` or `kpt`.
- Platform owners review modifications to the base application definition, but don't necessarily explicitly have to manage parameters.
- A pipeline renders the packages, templates, and customizations to a fully hydrated version of the configuration.
- Anthos Config Management takes the configuration verbatim from the repository. This approach is a key advantage of the fully hydrated approach as there's no interpretation of your desired configuration.
- Platform policy is validated both before deployment in the expansion pipeline, and again when applied to the cluster.

In the following diagram, the modules define tasks to be performed that generates a final configuration that is applied to clusters using Anthos Config Management:



Install considerations

The following steps are performed once to provide a foundation for the deployment:

- Create a Git repository for the root config, such as `$ORGNAME/anthos-acm-root`
- Create sub directories for each fleet, such as `fleet-prod` and `fleet-nonprod`
- Enable Anthos Config Management on all clusters of the fleet.
    - This step can be done with the Google Cloud console or with Terraform.

The following steps are performed as each cluster is created:

- Create a directory for that cluster within the root config Git repository, such as `fleet-prod/cluster-abc01-prod`.
    - Each cluster name must be unique across the organization.
- Set up configuration sync for that cluster.
    - You can use the Google Cloud console or Terraform.
    - When using Terraform, create a `member_feature`. The membership has already been created when the cluster was, and doesn't require import. Instead, it's an output parameter of the `gkeonprem` user cluster.
- The path within the repository is different for each cluster:
    - For example, `fleet-prod/cluster-abc01-prod`. This path is referred to as adding a RootSync to the cluster.
- Other settings can be the same across sites, such as the following:
    - All clusters use the same URL, like `http://$GITPROVIDER/$ORGNAME/anthos-acm-root`
        - If Git is mirrored or separated across sites, `$GITPROVIDER` can be different per site.

    - All clusters use the same branch, such as HEAD.
    - Set Unstructured mode.

When these steps are done and several clusters have been created, the directory structure for `anthos-acm-root` looks like the following example:

```
fleet-prod/
  cluster-abc01-prod/
    asm/
    policy/
    storage/
  cluster-xyz01-prod/
    asm/
    policy/
    storage/
fleet-nonprod/

cluster-abc01-staging/
    asm/
    policy/
    storage/
```

Related policy objects can be grouped into directories by concern. In this example, categories `asm/`, `policy/`, and `storage/` are used.

Application deployment

The following steps are performed for each application that you deploy:

- Create a Git repository to hold that application's configuration.
    - For example, if the application's name is $X, make a repository `$ORGANIZATION/app-config-$X`.
    - Use multiple Git repositories to reflect administrative boundaries. Each team should work in only one repository and ideally different teams have their own repositories.
    - To make sure that changes can be reviewed by all stakeholders, use the `CODEOWNERS` feature of GitHub or GitLab.
- For each environment and site where the app should run, complete the following steps:
    - Create a namespace in the format like `$appname-$env-$site` by adding it to the Root Repo
    - Create a `reposync.yaml` file in the namespace directory, such as `cluster-xxx/namespaces/app-site-env/reposync.yaml`
    - For the Helm approach, point to the artifact registry path and image name (method). Set cluster-specific values.

- For the fully hydrated approach, set the Git repository and directory for the namespace. A single Git repository can be used for Root and Application repositories.

After deploying two applications, the `anthos-acm-root` directory looks like the following example:

```
fleet-prod/
  cluster-abc01-prod/
    asm/
    namespaces/
      team1/
        app1/
          reposync.yaml
          rbac.yaml
        app2/
          reposync.yaml
          rbac.yaml
    policy/
    storage/
  cluster-xyz01-prod/
    asm/
    namespaces/
      team1/
        app1/
          reposync.yaml
          rbac.yaml
        app2/
          reposync.yaml
          rbac.yaml
     policy/
    storage/
```

```
fleet-nonprod/
  cluster-abc01-staging/
    asm/
    namespaces/
      team1/
        app1/
          reposync.yaml
          rbac.yaml
        app2/
          reposync.yaml
          rbac.yaml
    policy/
    storage/
```

Each `reposync.yaml` file follows one of the following two patterns:

1. A Git repository and path holding fully hydrated configuration, like the following example:

```
# File: fleet-prod/cluster-abc01-prod/namespaces/app1/reposync.yaml
apiVersion: configsync.gke.io/v1beta1
kind: RepoSync
metadata:
  name: app1
  namespace: config-management-system
spec:
  sourceFormat: unstructured
  sourceType: git
  git:
```

```
   repo: https://gitmirror.abc01.company.com/team1/app1-config/
   branch: main
   auth: token
   secretRef:
      name: git-repo-secret
```

2. An OCI image that holds a Helm chart and has a tag, and Helm parameters like the following example:

```
# File: fleet-prod/cluster-abc01-prod/namespaces/app1/reposync.yaml
apiVersion: configsync.gke.io/v1beta1
kind: RepoSync
metadata:
  name: app1
  namespace: config-management-system
spec:
  sourceFormat: unstructured
  sourceType: helm
  helm:
    repo:  oci:/imgreg.abc01.company.com/team1/manifests/app1:v1.1
    chart: app1
    version: v2
    releaseName: instance1
    namespace: app1
    auth: token
    secretRef:
      name: helm-repo-secret
    values:
      param1: 1234
      param2: asdf
```

Roll out of cluster configuration

Determine an order of updates for clusters. Update staging before production, and one site at a time. Apply edits to each cluster subdirectory at a time, such as in the following order:

1. `fleet-nonprod/cluster-abc01-staging`
2. `fleet-prod/cluster-abc01-prod`
3. `fleet-prod/cluster-xyz01-pro`

Roll out of an application configuration

Use a development cluster to test changes. There can be one cluster for each application team. These clusters should use the same base configuration or Helm chart, but be parameterized to use test data and dependencies. Refactoring applications so that they can work in different environments with similar configuration is an important part of achieving frequent successful deployments.

Complete the following steps to deploy an application at a new image tag:

- Helm approach:
  - The app is built into an image with a new tag.
  - Update the Root repository to use the tag, progressively in a series of clusters, like in `cluster-acb01-staging` and then in production clusters like `cluster-abc01-prod` and `cluster-xyz01-prod`.

- Fully hydrated approach:
  - Modify the parameters that control the image tag in that cluster, such as a `Kustomize.yaml` or Kptfile.
  - Review and commit.
  - Pipeline expands configuration and commits to Namespace repository.
  - Anthos Config Management pulls the fully expanded config from Namespace repo.
  - Repeat this process progressively in a series of clusters.

Recommended policies

The following policy bundles are recommended:

- `K8sPSPHostFilesystem`[27]
  - This policy ensures that applications don't take unexpected dependencies on the host operating system version, which allows independent application and OS updates. This policy also helps to increase security.
- Consider requiring mesh authorization for each namespace[28].
  - Start with Namespace level, rather than at the workload selector or mesh level.
  - Use EnvoyFilters when finer control of authorization is needed.
- You can also port existing policies from OpenShift[29].

Anthos Service Mesh

Anthos Service Mesh provides security, observability, and traffic management features both within and across clusters[30]. In this reference architecture, only the within-cluster capabilities of Anthos Service Mesh are used. Clusters don't have to connect with each other. However, metrics for equivalent services, those with the same service name and namespace name, can be readily viewed in aggregate.

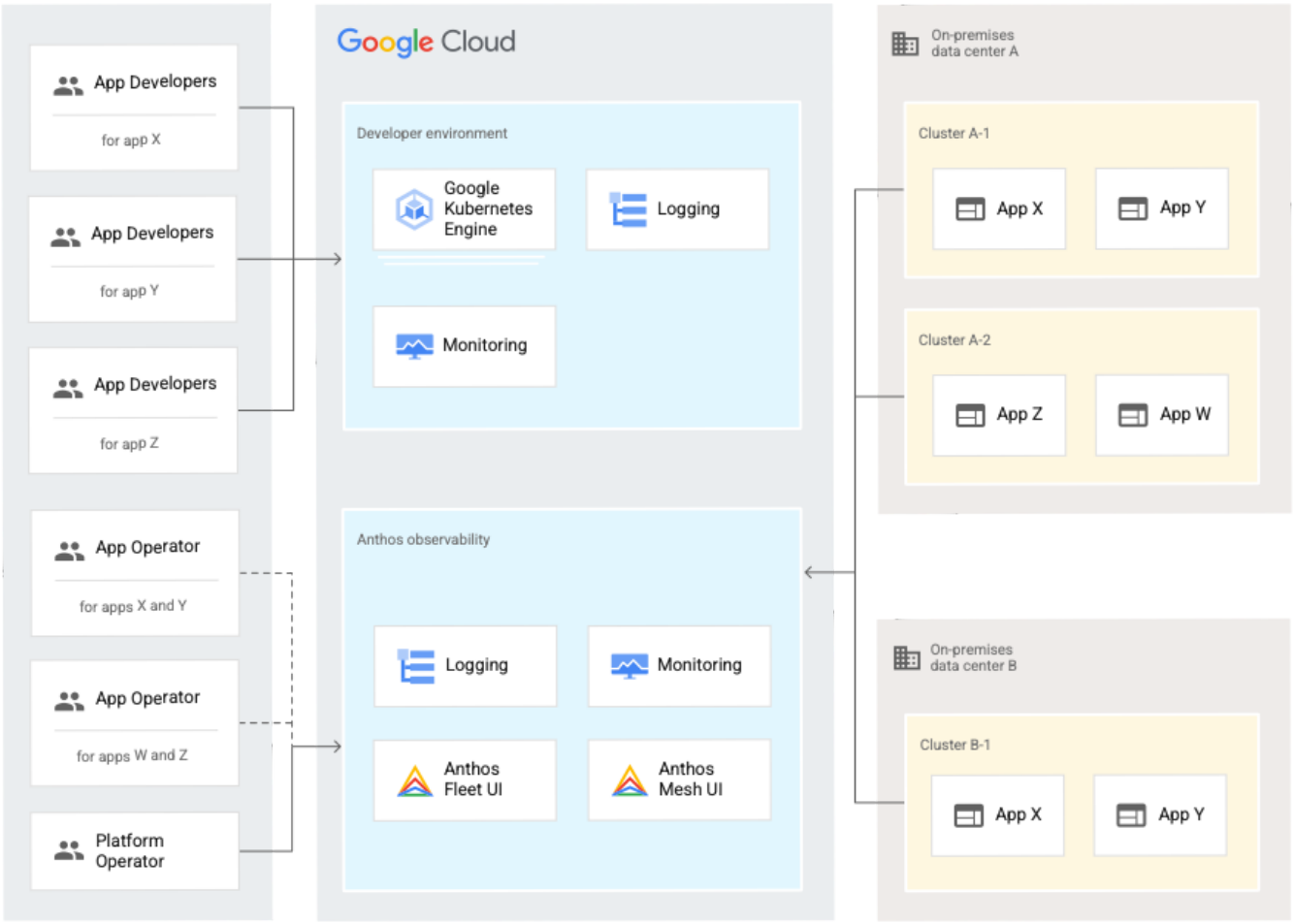
Configure Anthos Service Mesh as follows:

- Anthos Service Mesh for Anthos clusters is unmanaged. Follow the instructions to set up a multi-cluster mesh. However, don't set up cross-cluster trust[31].
- Enable strict mTLS.
  - As a security best practice, enable strict mTLS at the mesh level[32].
  - To enforce this setting, use the policy bundle with strict mode[33].

- mTLS certificates for in-mesh communication:
    - Protect the CA signing key by selecting either Mesh CA or Certificate Authority Service (CA Service).
        - Use CA service if you have special CA requirements[34]. Otherwise, use Mesh CA.
    - Install Mesh CA or CA Service.
- Certificates for serving ingress traffic:
    - Use external certificate management on ingress[35] and egress[36].
    - Applications can continue to use existing application certificates, such as Let's Encrypt or Google-managed certificates.
- Enable Cloud Monitoring (HTTP in-proxy metrics), which is part of the platform and included as system metrics.
- Implementation of the following examples items is optional and can be enabled when convenient after initial deployment:
    - User authentication with Identity-Aware Proxy.
    - User authentication with your existing Identity Provider.
    - Anthos Service Mesh user authentication.

Observability

The following diagram shows how application developers and application or platform operators can view logging and monitoring data in Google Cloud. The on-premises clusters and applications send this logging and monitoring data back to Google Cloud for analysis and review:



The following observability considerations apply to this reference architecture:

- All observability data for production clusters, such as system logs, system metrics, application logs, and application metrics, is directed into `project-2-fleet-prod`.
- All observability data for staging clusters is directed into `project-3-fleet-staging`.
- The platform team may create a dashboard for each application, with application operators given access to that application's dashboard.
    - The dashboard may contain container-level metrics, service metrics (Anthos Service Mesh), application-specific metrics if applicable, log-based metrics, and logs panels (if appropriate).
    - Grant members of the team access to their appropriate dashboard. This approach gives access to their metrics without exposing other team's metrics.
    - Use Identity and Access Management (IAM) conditions based on the resource name attached to the project resource, such as `project-x-ops`, if the teams need log access.
- If compatible with company policies, give application operators and application developers `view` access to `project-x-ops`.
    - Application developers and application operators can benefit from debugging other services that they depend on or are dependencies of.

    - Placing all metrics and logs in a single project allows most effective use of Cloud Operations.
    - Placing all metrics and logs in a single project ensures proper tagging of logs and metrics with cluster identifiers.
- Platform teams should create dashboards and grant permissions with IAM conditions on them for one or more appropriate teams to see the dashboard.

Application monitoring

- All application metrics go to the operations project.
- Application metrics are automatically labeled and the labels can be used for aggregation of metrics. Labels include the cluster name, namespace name, project, pod name, deployment name, and user-provided Kubernetes labels. The following Applications section provides more details on recommended labels.
- Applications which provide Anthos Service Mesh services also produce service logs. These service logs also go to the operations project.
- You have two options for giving application developers access to monitoring data:
    - Give application developer teams `view` permissions on the namespace and on the cloud resources in this team / function project.
    - Delegate dashboard access with a third-party tool.

System monitoring

We recommended that you set service level indicators (SLIs) to track the health of Anthos clusters, and define playbooks for responding to these conditions. The following suggested SLIs cover the ability and performance of the cluster to schedule workloads. These SLIs use metrics that are available in Cloud Monitoring:

- **Control plane responsiveness:**
    - Suggested SLI: 90th percentile control plane request latency
    - PromQL expression:

```
histogram_quantile(0.9,sum(
rate(kubernetes_io:anthos_apiserver_request_duration_seconds
_bucket[5m])) by (cluster_name, le))
```

    - Alert: Establish a baseline value and then alert if 2x above baseline.
    - Response to alert: Control plane latency may increase if an excessive number of objects are created, or if automated processes are sending an excessive number of requests.

- **Node availability:**
    - Suggested SLI**:** Number of nodes not ready for more than 10 minutes.
    - PromQL expression:

```
sum(kubernetes_io:anthos_kube_node_status_condition{conditio
n="Ready",status="true"})/sum(kubernetes_io:anthos_kube_node
_status_condition{condition="Ready"})
```

- Alert: Start by alerting if any nodes aren't ready for more than 10 minutes. Adjust with experience with your workloads.
- Response to alert:
  - Check last known values of secondary node status conditions, such as with the following PromQL expressions:
    - Memory pressure:

```
sum
(kubernetes_io:anthos_kube_node_status_condition{
condition="MemoryPressure",status="true"})
```

    - Running out of disk space:

```
sum
(kubernetes_io:anthos_kube_node_status_condition{
condition="DiskPressure",status="true"})
```

  - Confirm network reachability between unready node and control plane.

- **Scheduler latency:**
  - Suggested SLI: Median end-to-end scheduling latency (the length of time that it takes from Pod creation to when the node name is set on the Pod object).
  - PromQL expression:

```
histogram_quantile(0.5,sum(kubernetes_io:anthos_scheduler_po
d_scheduling_duration_seconds_bucket) by (le))
```

  - Alert: Start by alerting if the median scheduling time exceeds one second. Adjust with experience with your workloads.
  - Response to alert:
    - Investigate number of pending pods in scheduler queue (`kubernetes_io:anthos_scheduler_pod_scheduling_duration_seconds`)
    - Investigate pending pod resource status.

- Monitor your cluster networking (DataplaneV2) metrics to avoid exceeding kernel resource limits[37].

Logging

Choose one of the following options for providing logs access to application operator teams:

- Grant application operator teams `view` access to the logs project. This approach works well if a single application operator team already manages most applications and has application logs access.
- Don't grant `view` access to application operator teams. This approach works well if existing policies don't allow application operator teams to access logs.
- Create logging views[38] for each application operator team. Create a filter to control which logs each application operator team can use. This approach works when there are several, but less than 25, different application operator teams that each manage different sets of applications.

Roles and permissions

In this section, permissions are suggested for two different types of organizations.

In the first type of organization, a platform team hides the complexity of operations and infrastructure for developers. Application developers don't directly participate in operating applications in staging and production environments. They might be unaware of facts like their containerized applications are deployed onto Kubernetes, what clusters exist, or how individual applications are replicated. For this type of organization, the *minimum permissions* are recommended.

The second type of organization promotes the development of a *DevOps culture* - a shared sense of responsibility between development and operations teams for the health of services. This approach requires additional permissions for application teams. The *expanded permissions* are recommended for the second type of organization.

Anthos supports both types of organization, and organizations at various points in between.

The following roles are assumed:

- Application developer team:
    - In the first type of organization, there may be no need for this role to have permissions on any of the resources covered in this document.
    - In the second type, application developer teams are assumed to take a role in observing whether their applications meet service level objectives. To make these observations, teams need some level of access to monitoring, and to see the configuration and status of deployed applications. Teams don't necessarily need to see the logging data itself, which might contain more sensitive information.
    - In larger organizations of the second type, there might be multiple distinct application developer roles, like `app-dev-team-1`, `app-dev-team-2`, or `app-dev-team-3`. Each team might specialize in developing and operating different sets of applications.

- Application operator team:
  - The application operator team typically sets SLOs or alerts for applications, and responds to application-level issues.

- Network operator team:
  - The network operations team manages global and local traffic in aggregate, and controls the lower layers of the network stack.

- Platform team:
  - The platform team develops and manages a platform which application developers use to deploy applications, while meeting numerous security, governance, reliability, cost, and other constraints.
  - In the first type of organization, the platform team may also take on the application operator role.
  - In the second type of organization, the application operators are typically a distinct role.

Project permissions

The following table outlines the roles that each persona should be assigned to the various projects used in this reference architecture:

Role	Project	Minimum permissions	Expanded permissions
Application developer team *xyz*	`project-0-net`	⌀	⌀

	`project-2-fleet-prod`
	
	`project-3-fleet-staging`
		⌀	`roles/monitoring.viewer`
			
			`roles/gkeonprem.admin`
			`roles/gkehub.viewer`
			`roles/gkehub.gatewayEditor`
			`roles/gkeonprem.viewer`

	`project-n-app-xyz`	⌀	`roles/viewer`
Application operations team for application of team *xyz*	`project-0-net`	⌀	`roles/viewer`

	`Project-2-fleet-prod`
	
	`project-3-fleet-staging`
		`roles/monitoring.viewer`
		
		`roles/gkehub.viewer`
		`roles/gkehub.gatewayEditor`
		`roles/gkeonprem.viewer`
			`roles/monitoring.viewer`
			
			`roles/logging.viewer`
			or
			`roles/logging.viewAccessor`
			
			`roles/gkehub.viewer`
			`roles/gkehub.gatewayEditor`
			`roles/gkeonprem.viewer`
			`roles/logging.viewer`

	`project-n-app-xyz`	`roles/viewer`	`roles/editor`
Network operations team	`project-0-net`	`roles/viewer`	`roles/owner`
	`project-2-fleet-prod` `project-3-fleet-staging`	⌀	`roles/monitoring.viewer`
	`project-n-app-n`	⌀	`roles/viewer`

Platform team	`project-0-net`	`roles/monitoring.viewer`	`roles/monitoring.viewer`
	`project-2-fleet-prod` `project-3-fleet-staging`	`roles/owner`	`roles/owner`
	`project-n-app-xyz`	`roles/owner`	`roles/owner`

Some application development teams might develop multiple applications, and some application operator teams might operate apps from multiple development teams. In the previous table, `xyz` refers to one or more applications or groups of closely related applications. Some individuals might also work on multiple teams.

Cluster permissions

The following table outlines the permissions that each persona should be assigned to the clusters used in this reference architecture:

Role	Cluster type	Minimum permissions	Expanded permissions
Application developer and operator teams *xyz*	Admin cluster	⌀	⌀
	User cluster (production)	⌀	Cluster read-only[1]
	User cluster (non-production)	Namespace read-only[1]	Cluster read-only Namespace read-write[1]
Network operations team	Admin cluster	⌀	⌀
	User cluster (production)	⌀	⌀
	User cluster (non-production)	⌀	⌀
Platform team	Admin cluster	Cluster admin [1]	
	User cluster (production and non-production)	Cluster admin [1]	
Anthos Support (Google employees)	Admin cluster	Read-only [2]	
	User cluster	⌀	Read-only [2]

1: Generate with `gcloud container hub memberships generate-gateway-rbac` (ref)
2: Generate with `gcloud container hub memberships generate-gateway-rbac –anthos-support` (ref)

Applications

Namespaces and app projects

Use the following guidance for naming namespaces and projects:

- When you create a new application, give it a descriptive name prefix, such as `shoppingcart` for a shopping cart service.
- Create a project to hold Google Cloud resources related to the application, like images, such as `project-5-shoppingcart-app`.
- Compose a namespace name which is the same as the application name. In this example, the namespace would be called `shoppingcart`.
- Typically, you use the global DNS load balancer to balance incoming traffic across sites for the shopping cart service.

Some services might run at multiple sites and are fungible, meaning that any one of them can serve the same request. In this scenario, use the following pattern:

- Use the same namespace name in all clusters that receive a share of the same traffic from the global traffic manager.
- Use the same service name at each site.

As an example, you might run the `shoppingcart` service in cluster `cluster-abc01-prod` and `cluster-xyz01-prod`. Both clusters serve requests for the shopping cart portion of the same website, so both clusters use the same namespace name.

For a set of services that have similar configuration, but handle different types of requests or data, use different namespace names for each instance. For example, if three separate database instances need to be deployed to hold user, product, and sales data, put them in namespaces `user-db`, `product-db`, and `sales-db`.

During version rollout, Anthos Service Mesh can be used to do blue-green update, or Kubernetes deployment may be used for rolling update.

For a set of services that have a similar service configuration, like from the same Helm chart, but aren't fungible or won't ever see the same request stream, use different namespace names. For example, two services on different sites or the same site that serve different purposes like logistics and sales, use namespaces like `logistics-db` and `sales-db`.

Application workspaces

- Allocate application names from a global centrally managed list. Avoid reuse within your organization.
    - Namespace names are constructed from application name plus suffixes which indicate environment and location.
    - Don't use the same application or namespace name for two unrelated purposes, even across clusters. If these clusters are later combined into a single fleet, the namespace names will then collide.
- Create a new namespace when deploying an application that is new to Anthos.

    - Several closely interacting applications can share a namespace, if they're developed and managed by the same team.
- Make a team or function project for each namespace.
    - Several namespaces that are developed and managed by the same people can share a project.
- For each application that needs to interact with Google Cloud resources:
    - Create a service account in the application's project, such as `project-`*`n`*`-app-`*`xyz`*.
    - Grant the workload identities in that namespace permission to access cloud resources that they need to operate, such as Cloud Storage buckets or Cloud databases.

Design and deploy applications

- When possible, refactor monolithic applications into smaller services[39].
- When building images for newly written applications, prefer distro-less images.
- Where possible, use active-active replication for applications, and use the following considerations:
    - Run at least three Pods for low traffic production applications.
    - Place these Pods behind a service.
    - When porting larger single instance applications, reduce the memory and CPU requirements proportionally.
    - Running a single Pod is acceptable for development environments where you don't need redundancy.
- Run distributed applications across clusters, with multi-cluster ingress. This approach helps distribute application load, and helps minimize the impact of a misconfiguration or failed application update in one cluster.
- Stateful applications can be deployed in containers.
    - Use pod disruption budgets (PDBs) to define your tolerance of disruptions[40].
- When deploying a new application, create a `PodMonitoring` resource[41] for each workload (statefulset, deployment) in the application's namespace. This approach sends application metrics to the operations project. Mesh service-level metrics should also go to that project.

References

1. https://cloud.google.com/kubernetes-engine/docs/best-practices/upgrading-clusters#multiple-enviros
2. https://cloud.google.com/resource-manager/docs/cloud-platform-resource-hierarchy
3. https://www.weave.works/blog/gitops-operations-by-pull-request
4. https://cloud.google.com/sdk/docs/install
5. https://cloud.google.com/sdk/docs/proxy-settings
6. https://www.jfrog.com/confluence/display/JFROG/Docker+Registry
7. https://cloud.google.com/container-optimized-os/docs/concepts/features-and-benefits
8. https://cloud.google.com/anthos/clusters/docs/on-prem/latest/how-to/plan-ip-addresses
9. https://cloud.google.com/anthos/clusters/docs/on-prem/latest/how-to/firewall-rules
10. https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.networking.doc/GUID-004E2D69-1EE8-453E-A287-E9597A80C7DD.html
11. https://cloud.google.com/anthos/docs/resources/partner-storage
12. https://cloud.google.com/anthos/docs/concepts/anthos-connectivity
13. https://cloud.google.com/anthos/clusters/docs/bare-metal/latest/installing/registry-mirror#create_clusters_from_the_registry_mirror
14. https://cloud.google.com/anthos/clusters/docs/bare-metal/latest/installing/package-server
15. https://cloud.google.com/kubernetes-engine/docs/concepts/multitenancy-overview
16. https://cloud.google.com/blog/products/identity-security/5-must-know-security-and-compliance-features-in-cloud-logging
17. https://cloud.google.com/logging/docs/routing/managed-encryption#external_key_considerations
18. https://cloud.google.com/anthos/clusters/docs/on-prem/latest/how-to/logging-and-monitoring
19. https://cloud.google.com/service-mesh/docs/observability/explore-dashboard
20. https://cloud.google.com/stackdriver/pricing
21. https://cloud.google.com/anthos/clusters/docs/on-prem/latest/concepts/scalability#gke_hub
22. https://cloud.google.com/anthos/fleet-management/docs/quotas
23. https://registry.terraform.io/providers/hashicorp/google/latest/docs
24. https://opencontainers.org/
25. https://helm.sh/docs/topics/chart_repository/
26. https://cloud.google.com/anthos-config-management/docs/how-to/using-asm-security-policy
27. https://cloud.google.com/anthos-config-management/docs/latest/reference/constraint-template-library#k8spsphostfilesystem
28. https://cloud.google.com/service-mesh/docs/security/anthos-service-mesh-security-best-practices#enable-access-controls
29. https://cloud.google.com/architecture/migrating-containers-openshift-anthos-scc
30. https://cloud.google.com/architecture/service-meshes-in-microservices-architecture
31. https://cloud.google.com/service-mesh/docs/unified-install/off-gcp-multi-cluster-setup
32. https://cloud.google.com/service-mesh/docs/security/anthos-service-mesh-security-best-practices

33. https://cloud.google.com/anthos-config-management/docs/how-to/using-asm-security-policy#high_strictness_level
34. https://cloud.google.com/service-mesh/docs/unified-install/install-anthos-service-mesh#install_ca_service
35. https://istio.io/v1.14/docs/tasks/traffic-management/ingress/secure-ingress/
36. https://istio.io/latest/docs/tasks/traffic-management/egress/egress-gateway-tls-origination/
37. https://cloud.google.com/anthos/clusters/docs/bare-metal/latest/limits#dataplane_v2_ebpf_limit
38. https://cloud.google.com/logging/docs/logs-views
39. https://cloud.google.com/architecture/microservices-architecture-refactoring-monoliths
40. https://cloud.google.com/kubernetes-engine/docs/best-practices/upgrading-clusters#reduce-disruption
41. https://cloud.google.com/anthos/clusters/docs/on-prem/latest/how-to/application-logging-monitoring