Reference architecture:

GKE Enterprise hybrid environment (part 1) - design prerequisites and considerations

Part 2 - Implementation details

December 2024

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.

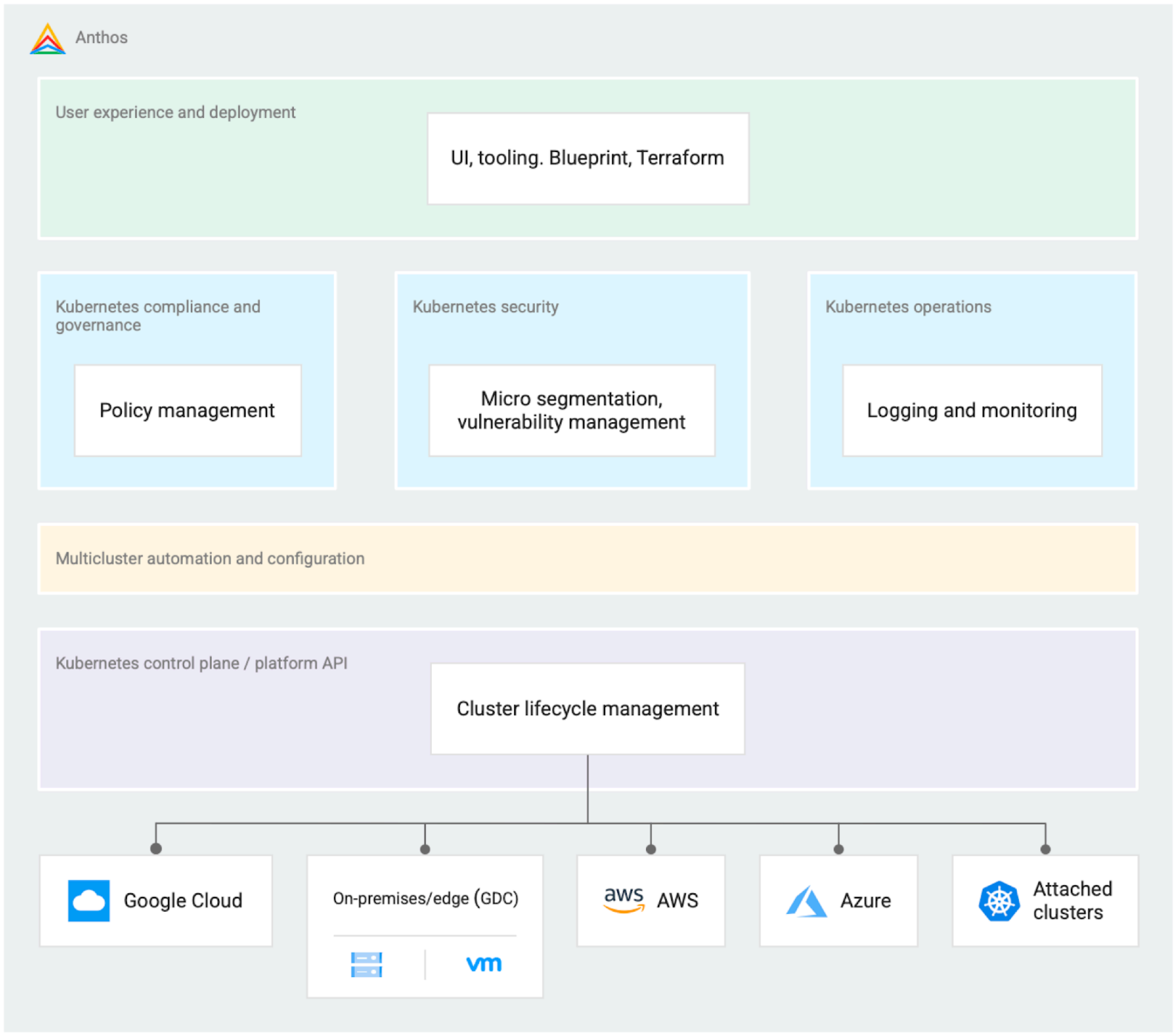
GKE Enterprise is Google's cloud-centric container platform for running modern apps anywhere consistently at scale. GKE Enterprise can help organizations by providing a consistent platform that lets them:

- 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.

GKE Enterprise helps you increase operational consistency in governance and security and developer velocity while reducing cost, deployment risk, and operational complexity. Specifically, GKE Enterprise 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 (***GKE Enterprise hybrid environment***).
- Google Cloud customers who want to better manage their containerized applications (***GKE Enterprise on Google Cloud***).
- Customers who want to solve multicloud complexity with a consistent governance, operations, and security posture (***GKE Multi-Cloud***).

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


Anthos
User experience and deployment
UI, tooling. Blueprint, Terraform
Kubernetes compliance and governance
Policy management
Kubernetes security
Micro segmentation, vulnerability management
Kubernetes operations
Logging and monitoring
Multicluster automation and configuration
Kubernetes control plane / platform API
Cluster lifecycle management
Google Cloud
On-premises/edge (GDC)
vm
aws AWS
Azure
Attached clusters

This reference architecture provides opinionated guidance to deploy GKE Enterprise 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 GKE unless stated otherwise. For example, some sections discuss *VMware vSphere clusters* composed of ESXi servers that pool compute resources.

This reference architecture is separated into two parts. Make sure you read both parts carefully to learn how to plan, design, and implement your own GKE Enterprise hybrid environment:

- Part 1 (this document) - Architecture, GKE Enterprise components, reference deployments, design prerequisites, and design considerations.
- Part 2 - Implementation details.

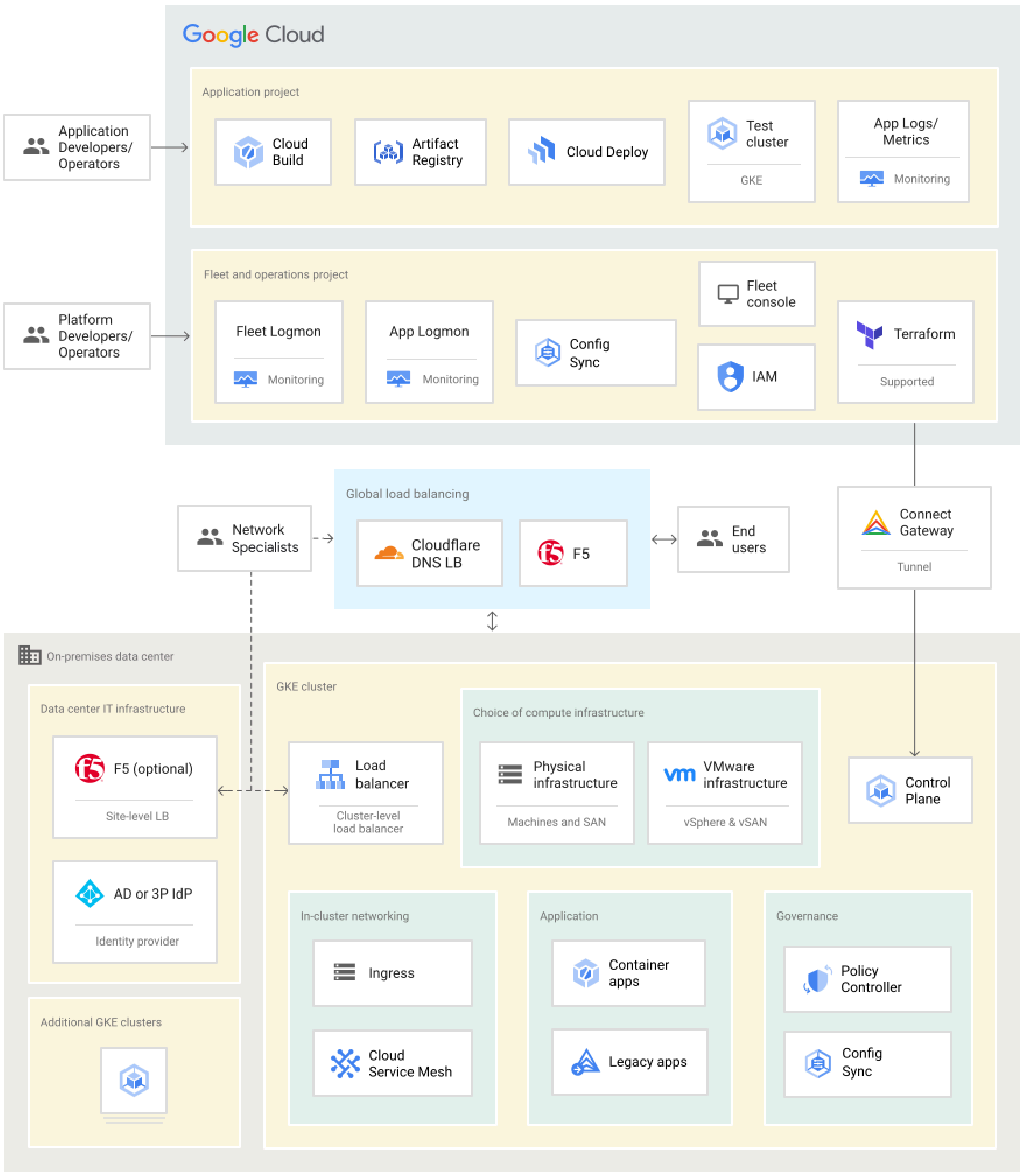
Architecture

The following architecture diagram provides an overview of a complete GKE Enterprise 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 GKE 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
Config Sync
Fleet console
IAM
Terraform
Supported
Global load balancing
Network Specialists
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 GKE clusters
GKE cluster
Load balancer
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
Config Sync

GKE Enterprise components

GKE Enterprise 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 GKE 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 GKE on VMware and GKE on bare metal using the Google Cloud console.
- **Fleets** let you group together multiple Kubernetes clusters from on-premises GKE clusters, GKE on Google Cloud, and GKE Multi-Cloud to enable and configure multi-cluster functionalities. Clusters are automatically added to a fleet when they're created.
- **Policy Controller, Config Sync, and Config Controller** are Google Cloud-hosted and in-cluster components that let you automatically deploy shared environment configurations and enforce approved security policies across fleets.
- **Service Mesh** is a suite of tools that helps you monitor and manage a reliable service mesh on-premises or on Google Cloud. Service Mesh is powered by Istio, a highly configurable and powerful open source service mesh platform.

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

GKE on VMware and **GKE 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.

- GKE on VMware integrates with VMware vCenter to provision Kubernetes clusters and provide lifecycle management of the virtual machines and operating systems.
- GKE on Bare Metal uses 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 a GKE Enterprise hybrid environment. These reference deployments show you how to plan, design, and implement the components that make up a complete GKE Enterprise hybrid environment. The Design prerequisites and Design considerations sections in the rest of this reference architecture provide additional context to plan and design your GKE Enterprise 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.

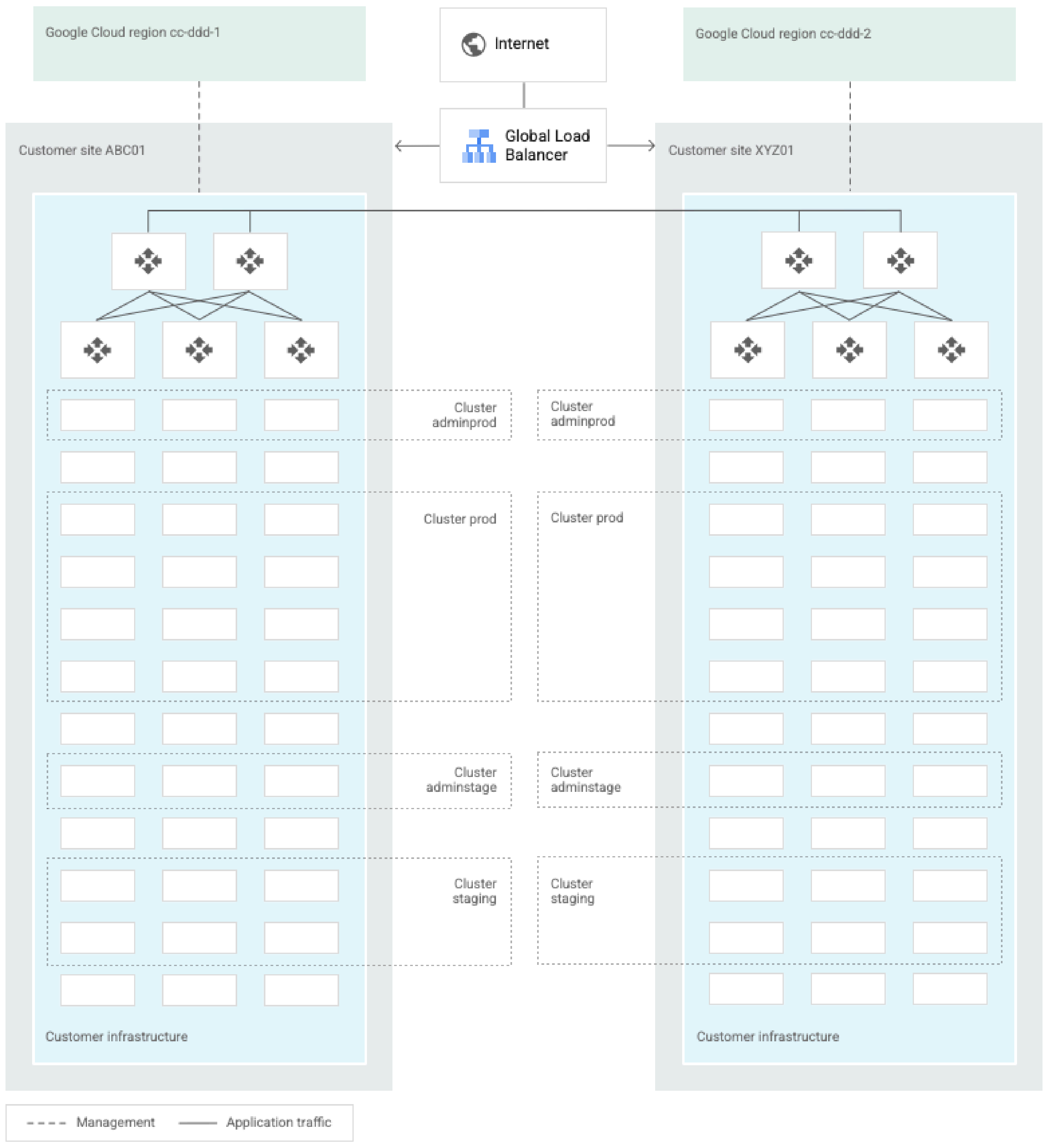
On-premises GKE clusters have two types of clusters - *admin* clusters and *user* clusters. Admin clusters manage the lifecycle of user clusters - they are administrative. User clusters are where you deploy your applications. Separate admin clusters for production and staging environments isolates changes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

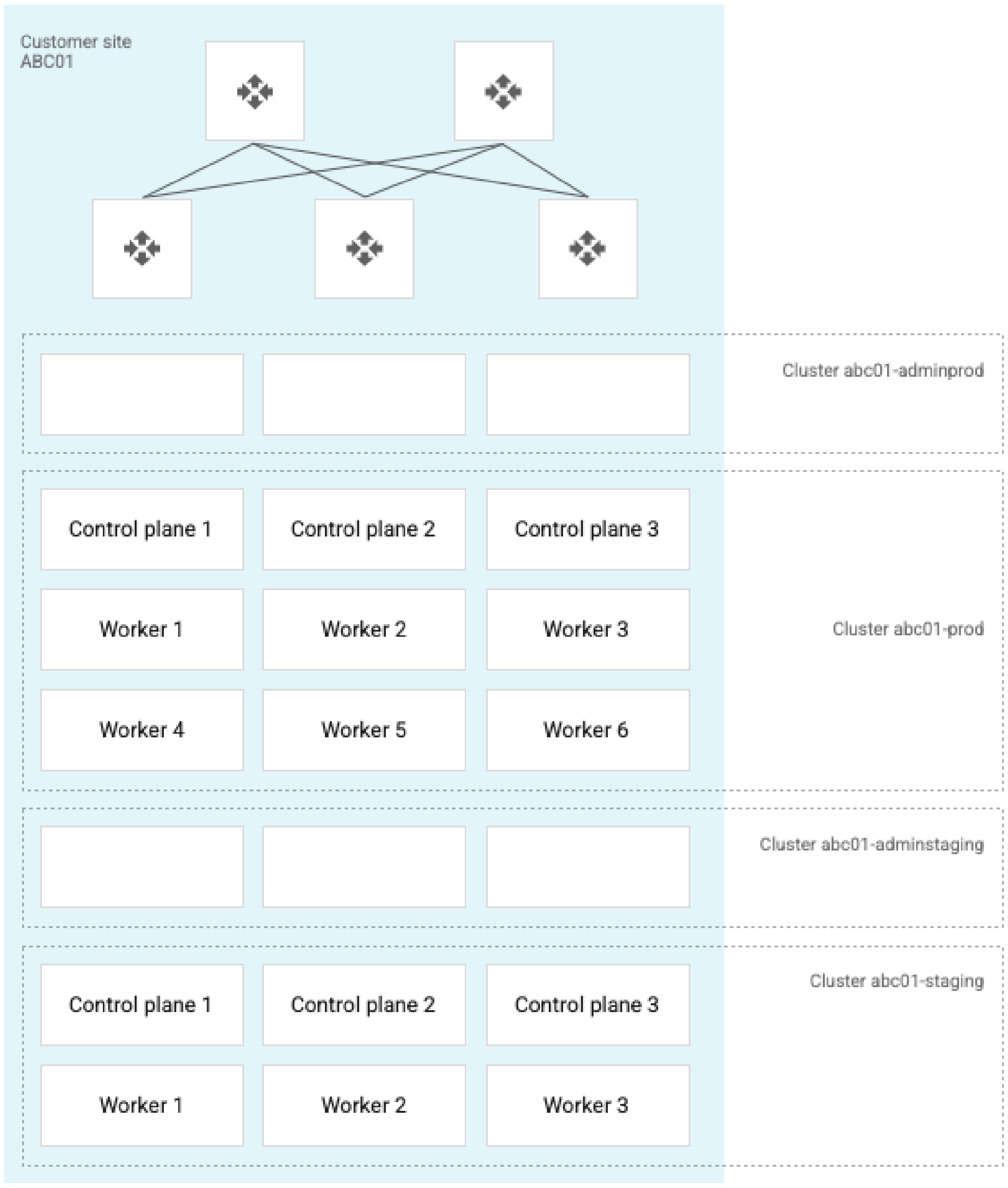
In both of the following example deployments, the clusters are configured in the following way:

- 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.

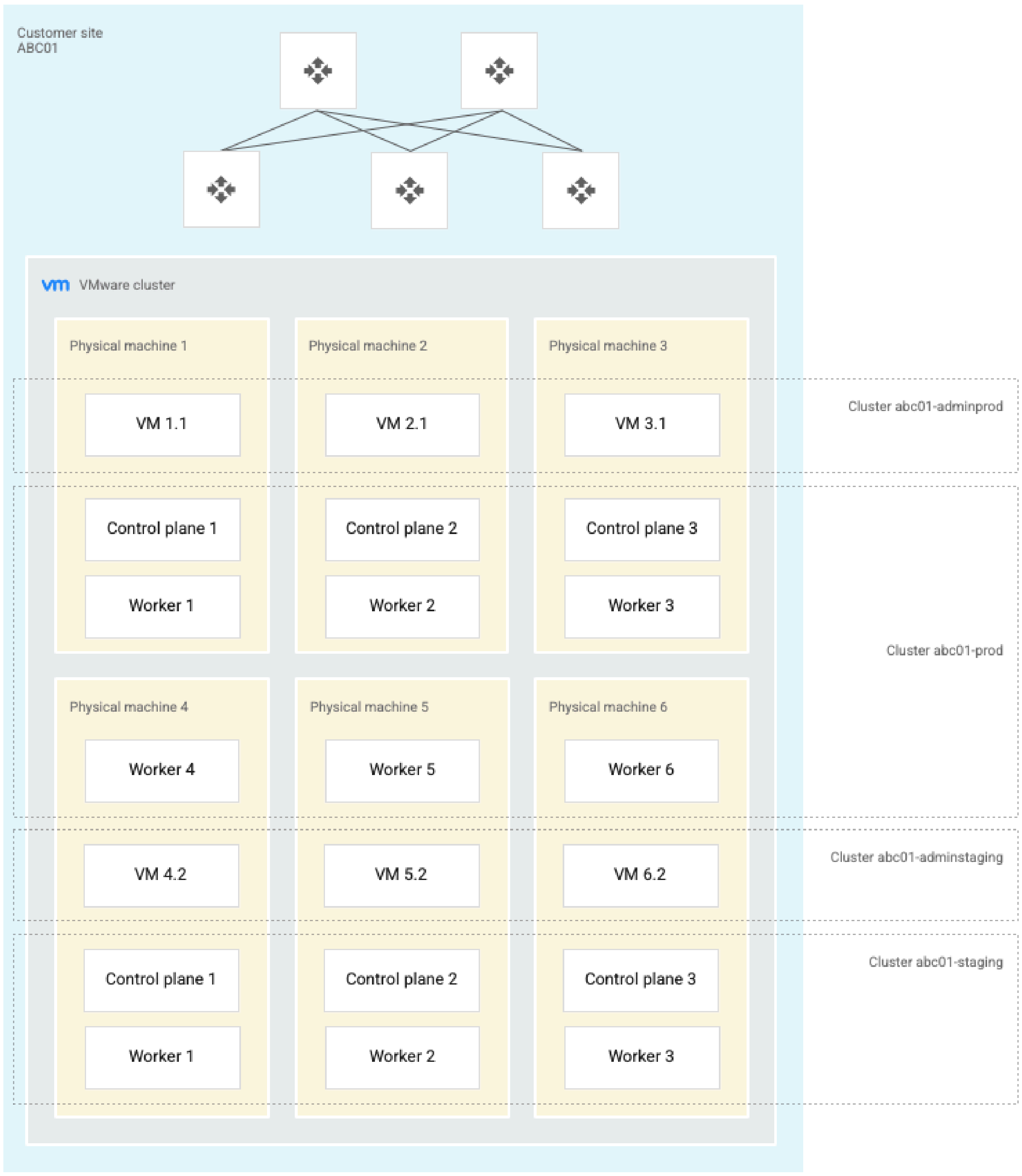
For simplicity of presentation, only one production cluster is shown per site. This design is a good starting point to bring a few large applications to production. As you bring more applications and lines of business to GKE, it's useful to have several production clusters on a given site. Tradeoffs 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 GKE 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 GKE on VMware, vSphere controls the VM placement. This management control might result in a less organized arrangement than illustrated. If you want a stricter deployment pattern, you can use VM host group affinity to force vSphere VM placement to follow the rack layout. With this approach, an ESXi host group can be created that reflects a rack. These groups can also be thought of as "partitions" within a vSphere cluster. When you create node pools, you can assign the host groups to a specific node pool. This behavior forces vCenter to follow the suggestions and place all VMs from a node pool onto the referenced ESXi host group.

Clusters and environments

GKE 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 GKE-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 Config Sync and Policy Controller.

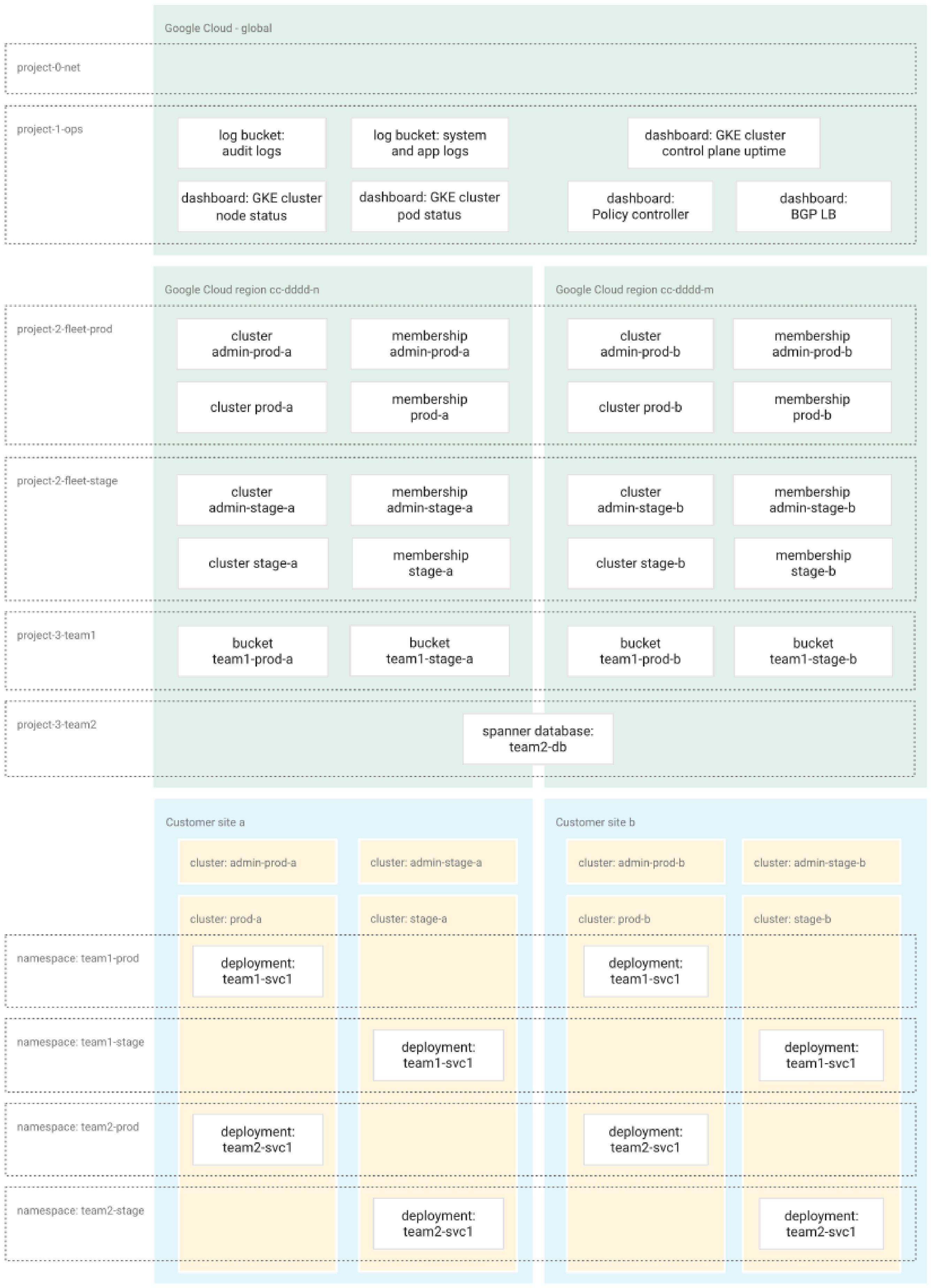
The following diagram gives a global view of logical cloud and on-premises resources that exist in this reference architecture. Resources are spread across several projects with appropriate identity and access management (IAM) controls applied. Also shown are examples of applications deployed across 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.

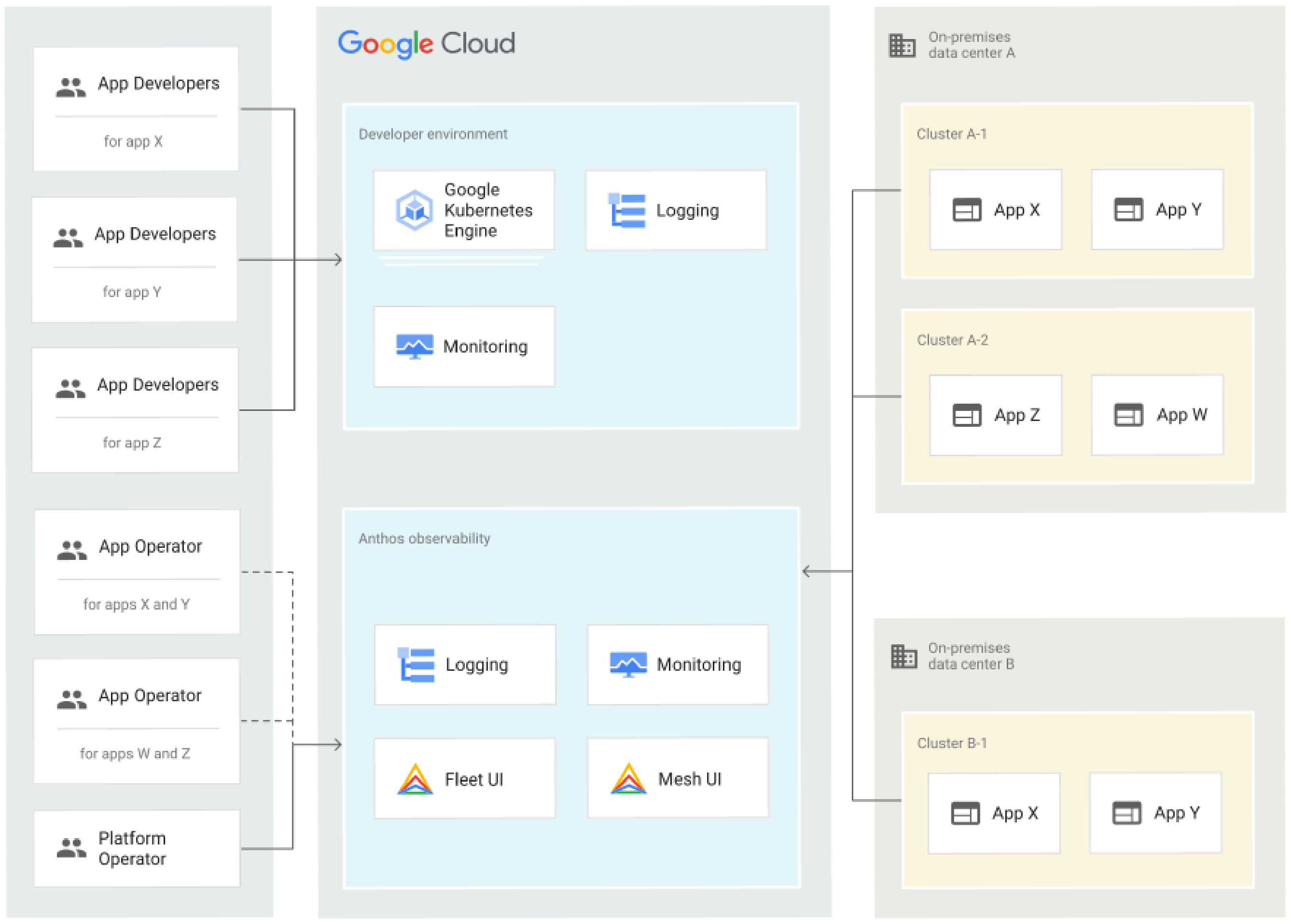
The diagram also shows resources deployed across namespaces in two customer sites as follows:

- 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: GKE cluster control plane uptime
dashboard: GKE cluster node status
dashboard: GKE 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 or platform admins and 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:



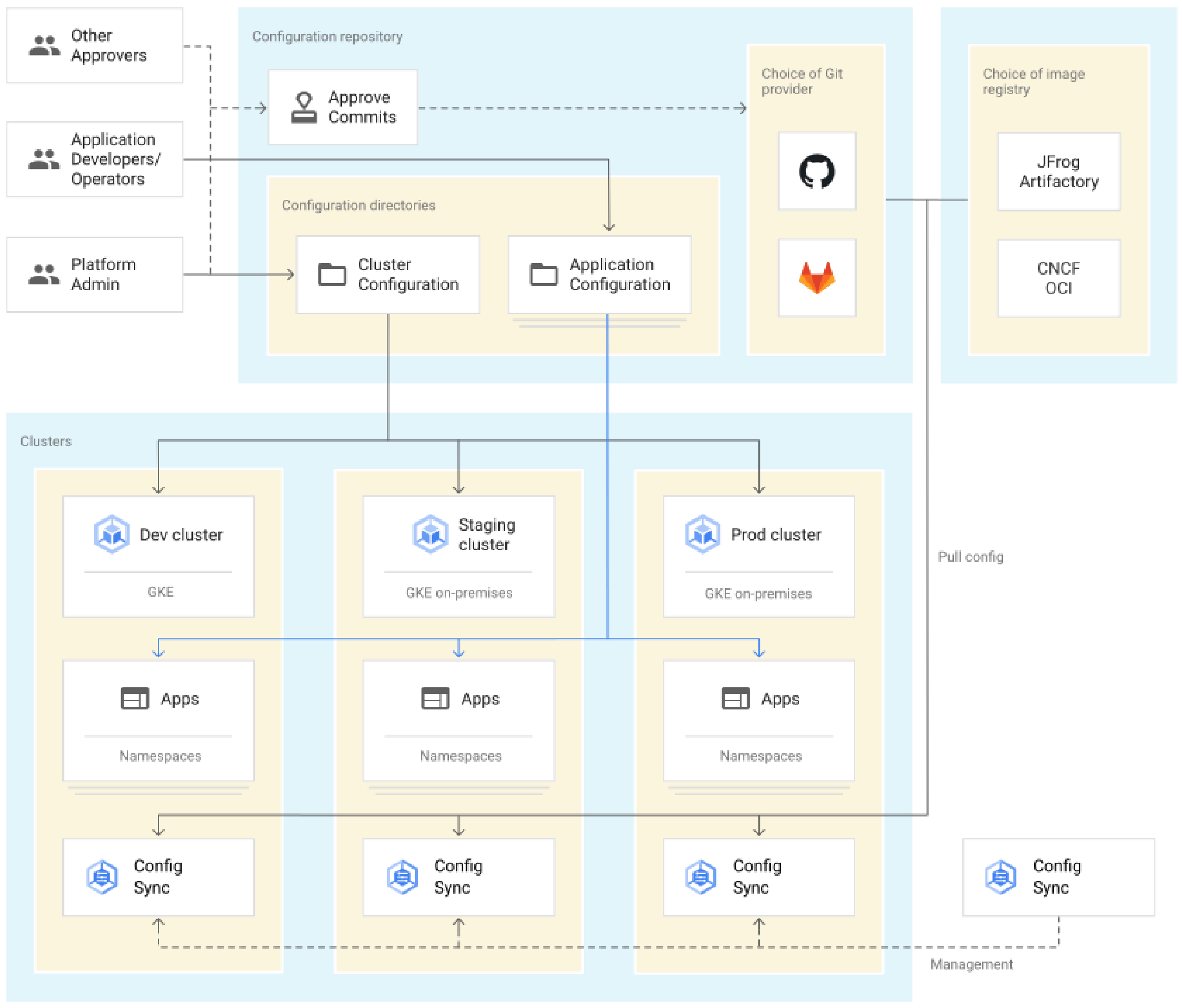
In this approach, different personas are granted access to only the environments they need. Applications that run in on-premises clusters direct logging and monitoring data back into Google Cloud for analysis and review.

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

Configuration management

Config Sync can be used to manage Kubernetes objects in all the clusters. Config Sync 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 Config Sync deployment uses one folder containing configuration for all clusters. Separate additional folders each hold configuration data, one for application:



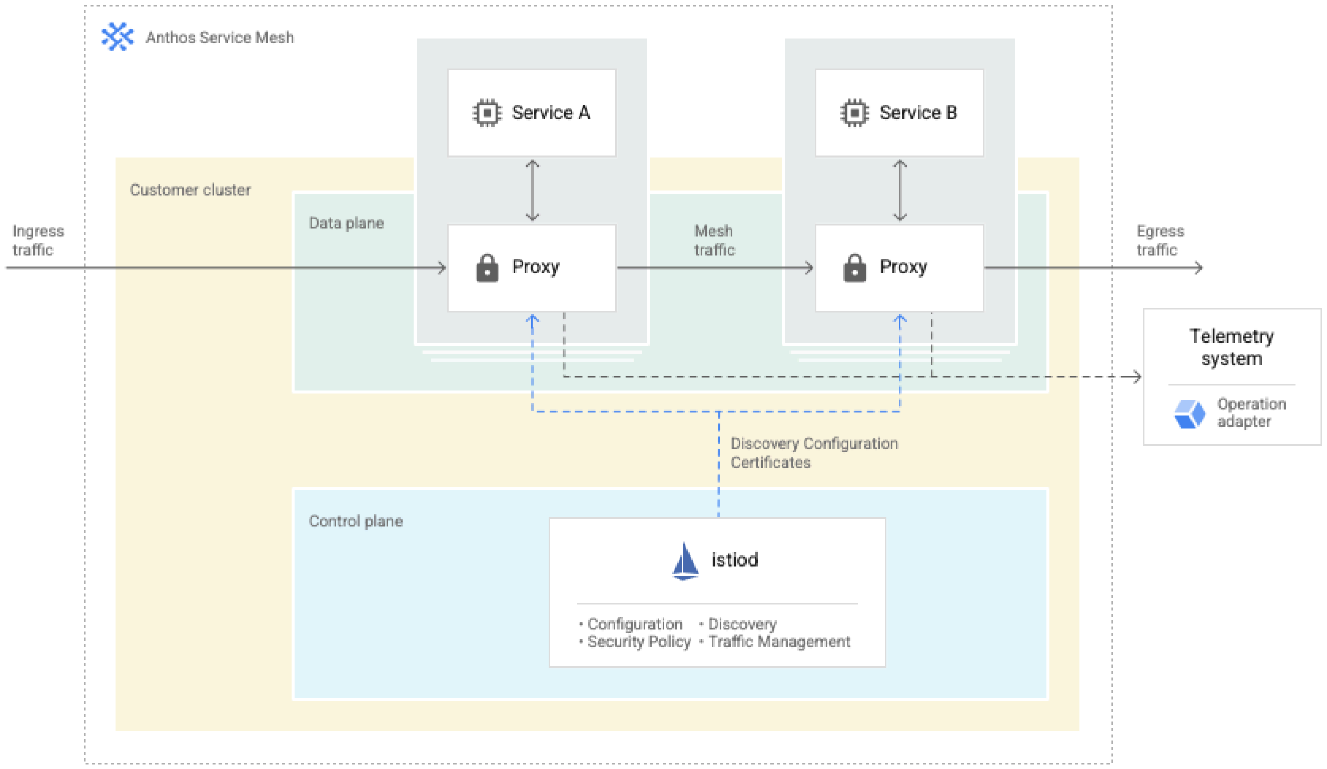
A following section in this reference architecture on implementation details provides more context and considerations to deploy this configuration management piece.

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.
- Service Mesh encrypts and controls communication between services in the same cluster.

The following diagram shows how 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 GKE Enterprise 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 GKE Enterprise 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

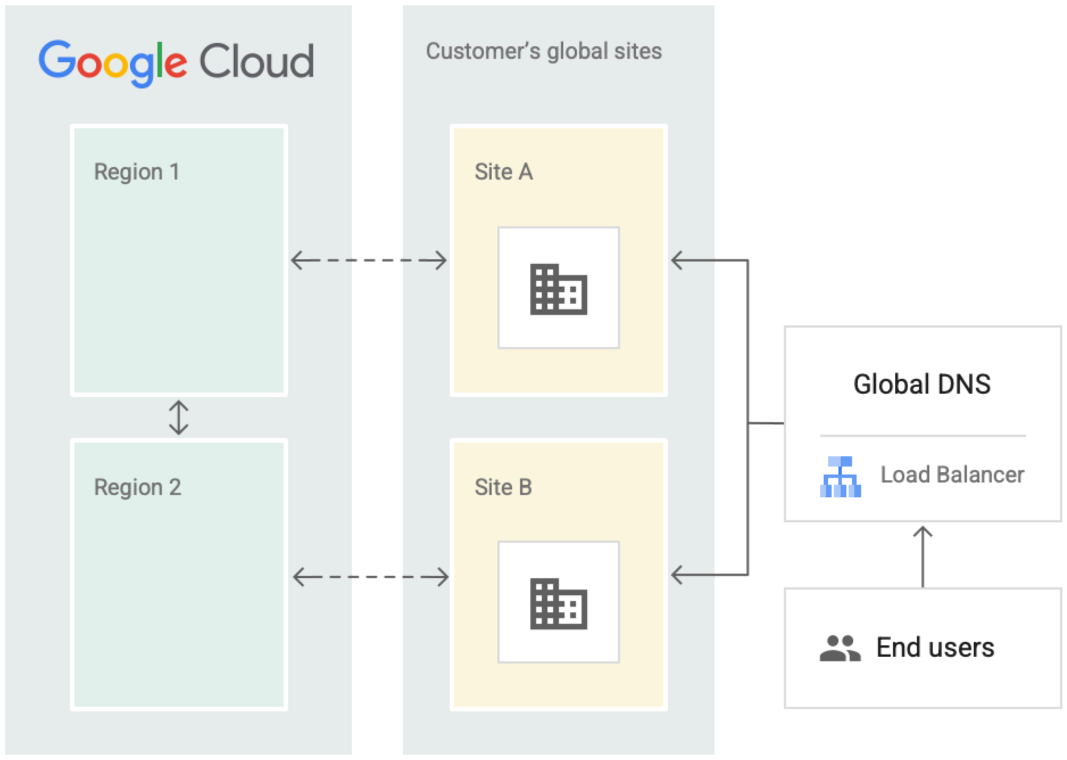
Identify several on-premises data centers, or *sites*, to run GKE clusters. Sites may be data centers that you own, colocation facilities, or major offices or facilities with secure machine rooms. Each site typically has a dozen to hundreds or more physical machines.

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. GKE Enterprise 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 GKE clusters. These sites run the associated Google Cloud services for the GKE Enterprise hybrid environment in this reference architecture.

The following diagram shows how you can connect using Cloud DNS and Cloud Load Balancing to GKE clusters that run in two different on-premises sites. Two proximate Google Cloud regions provide additional GKE Enterprise 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 Config Sync. The use and configuration of Config Sync is discussed later in this reference architecture.

Plan the Google Cloud organization and structure. Before you deploy GKE Enterprise, set up a Google Cloud organization and any folders that you want to use. GKE Enterprise 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 networking specialists 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`. A developers team and operators team for that application are granted access to this project. A central operators 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 a GKE Enterprise 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. Config Sync 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. GKE 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 vSphere 8.0 or later
License	vSphere Enterprise Plus

Compute requirements

GKE offers two ways to run clusters in an on-premises environment - GKE on bare metal and GKE 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 GKE 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 GKE on bare metal and GKE 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

GKE on VMware also requires 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 GKE clusters from one of the following:

- For GKE on bare metal:
  - Public Ubuntu LTS (20.04 LTS or 22.04 LTS)
  - RHEL (8.8, 8.10, 9.2, 9.4)

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

Storage

For GKE on VMware, we recommend using the storage policy in the cluster configuration. With VMware Storage Policy Based Management (SPBM), GKE on VMware places the VMs on datastores that belong to the same storage policy. This behavior ensures that VMs are allocated storage resources based on the administrator's specific requirements. Additionally, GKE on VMware automatically adjusts its metadata at the cluster level when storage vMotion happens, which prevents the clusters from breaking. Daily operations on the storage, such as adding or removing a datastore, or migrating GKE on VMware from one datastore to another, are simpler and easier under SPBM. GKE on VMware also supports VMware-supported storage such as VMFS or vSAN.

For GKE 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 GKEs 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.
    - On-premises GKE clusters use 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 GKE management services connect to GKE 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 GKE 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 GKE 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 GKE 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 GKE 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 GKE on bare metal, ensure L2 or L3 connectivity between machines, and from the admin cluster to the machines.
    - For GKE 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 `NetworkPolicy` and Service Mesh to control traffic between applications within a cluster.
- All the VMs that are part of your GKE 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 structures, all those basic requirements might require cross-team collaboration.

In GKE 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 GKE 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, GKE on VMware can enable direct routing mode for pod-to-pod connectivity, while GKE on bare metal must use tunnel mode.

The following table summarizes the basic network requirements:

	GKE on bare metal	**GKEs 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. GKE 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 GKE deployment.
- If you don't have an existing Local Traffic Manager, use the MetalLB bundled load balancing. MetalLB is available for both GKE on bare metal and GKE on VMware.

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

Goal	**Cluster type**	**Mode to use**
Use GKE-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 GKE Enterprise hybrid environment for your needs, and understand areas such as security and compliance, operations, and observability.

Hybrid deployment

GKE Enterprise offers two ways to run clusters in an on-premises environment - GKE on bare metal and GKE 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	**GKE on bare metal**	**GKE 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 GKE on VMware and GKE 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 GKEs 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, GKE 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. GKE on VMware is 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, GKE on VMware is a better option. This approach provides better bin packing, autoscaling of nodes, and hitless planned downtime.
- **Workload characteristics:** GKE on VMware is 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 GKE on bare metal:

- **vSphere license costs:** If you want to optimize on cost, use GKE 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.

GKE 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 GKE on VMware, vSphere handles physical machine failure. GKE on VMware uses VM anti-affinity rules to ensure VMs, like the control plane or three-node node pool, are spread across three different ESXi hosts. To give you control of VM distribution across hosts, you can also define VM host group affinity rules. Make sure that your vSphere instance can tolerate the loss of one node.
4. For GKE 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 GKE on bare metal, a cluster can use machines spanning multiple racks.
- For GKE 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 GKE on VMware, don't use VMware Stretched Clusters.

GKE on VMware can help to increase resiliency and support smarter availability designs by using VM host group affinity rules. These rules let vSphere be more granular on where to place specific VMs. This functionality allows the creation of so-called ESXi host groups within an existing vSphere cluster. ESXi host groups can match the racks where the selected hosts are placed, or could match hosts within a specific datacenter room or closeby location (same campus DC). This ability gives GKE awareness of the physical infrastructure. By mapping the ESXi host groups to nodepools, it's possible to create availability zones within the same vSphere cluster. GKE and vSphere make sure that VMs are placed in the corresponding host group, which means applications can be deployed across node pools to ensure a higher resiliency and availability.

On-premises GKE clusters are 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 GKE system container images[13].
    a. For GKE on VMware, use *thick images* that contain system container images. GKE on VMware loads these images into your local registry.
    b. To replace a failed machine in GKE on bare metal, GKE 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 GKE 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:

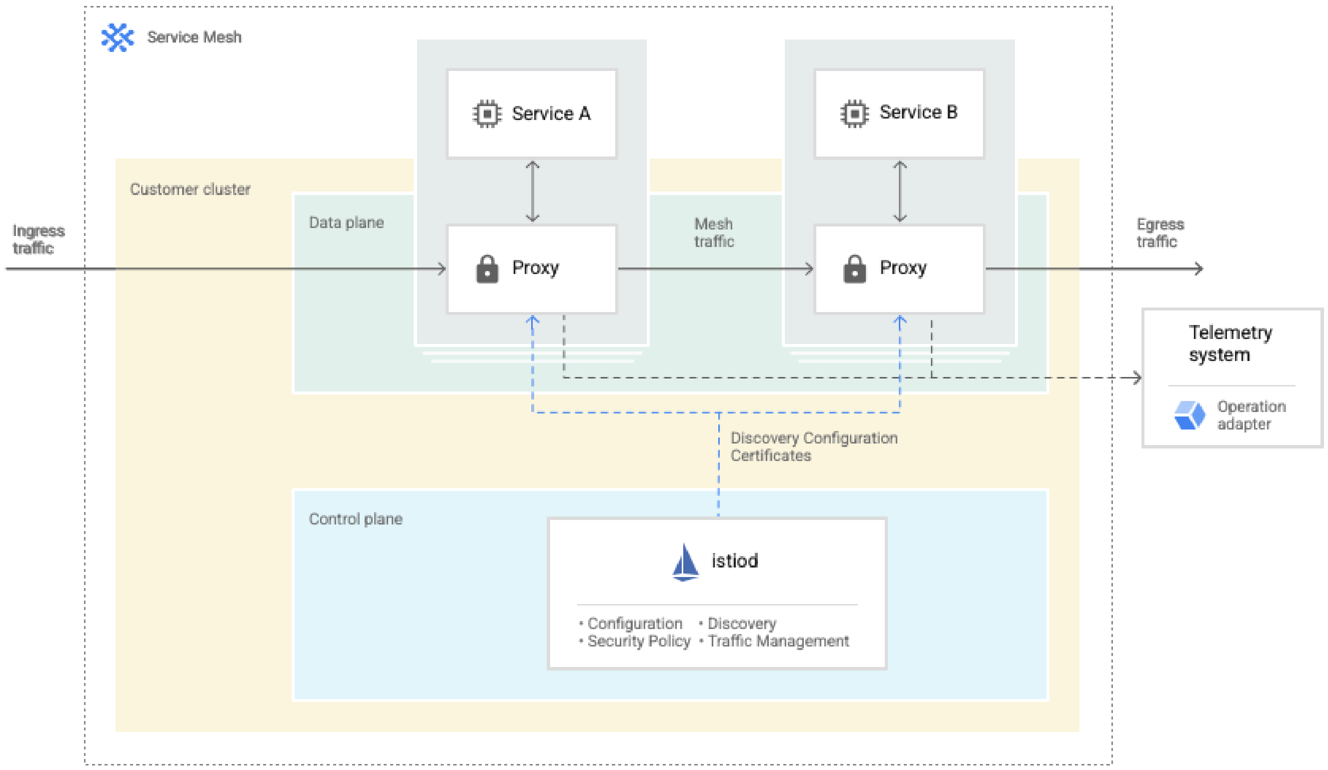
- Place the admin clusters on an administrative subnet and VLAN.
    - Although this separation isn't required by GKE, separating management traffic from non-management traffic is a commonly accepted practice.
    - Place all nodes of an admin cluster on the same VLAN.
- Place user clusters from different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

To help mitigate the risk of data exfiltration with GKE on bare metal, you can use VPC Service Controls. VPC Service Controls work with either Cloud Interconnect or Cloud VPN to provide additional security for your clusters. Using VPC Service Controls, you can add projects to service perimeters that protect resources and services from requests that originate outside the perimeter.

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

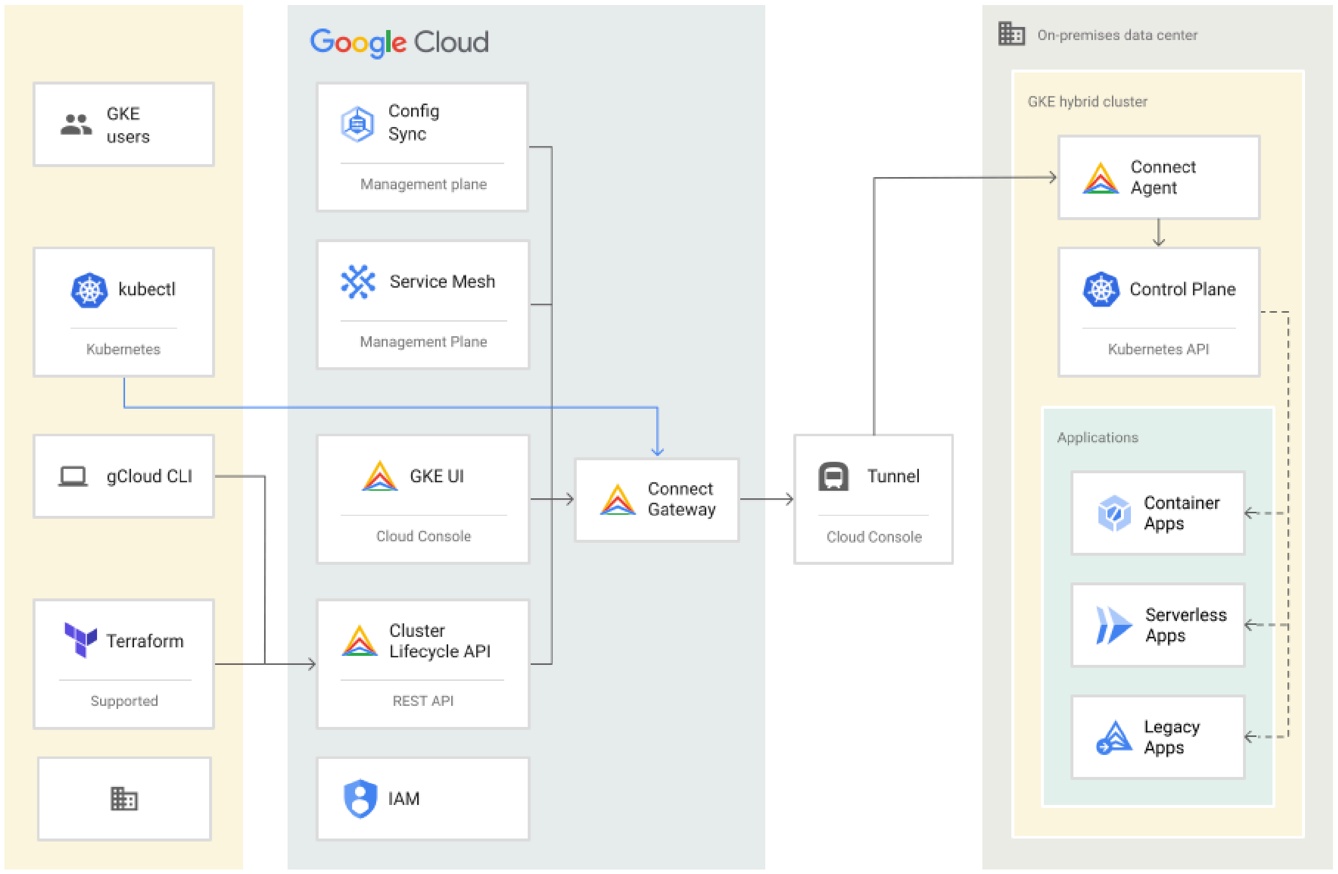
- Containers in a cluster can communicate to each other by default. Communication within a cluster can be restricted using Kubernetes Network Policies, or with Service Mesh.
- Several user clusters can be placed on the same VLAN.
  - In this case, isolation can be achieved with separate namespaces and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

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



When using 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 GKE on bare metal and GKE 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

GKE 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.

Config Sync 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 Config Sync	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.

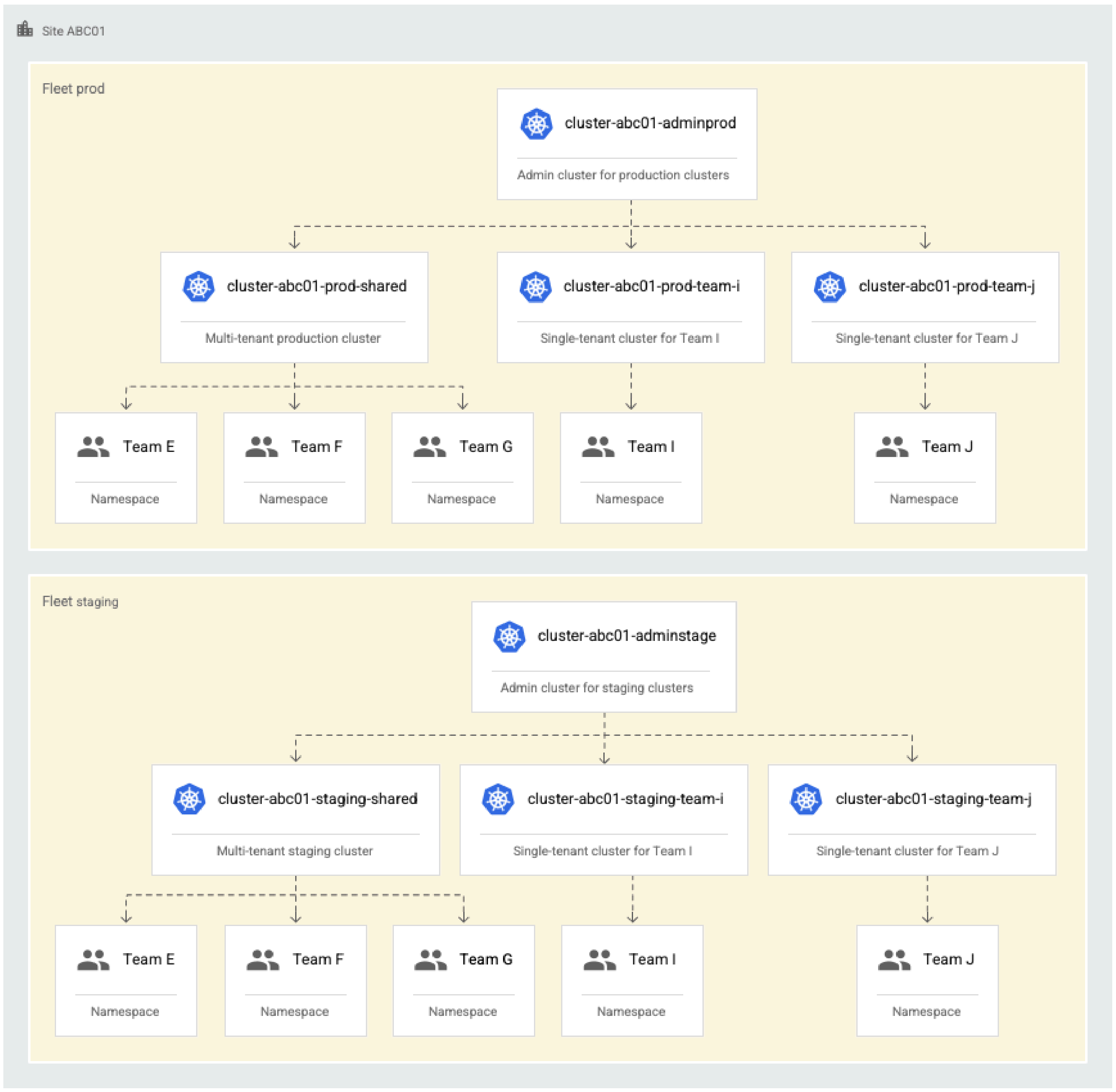
The following are examples of common deployment patterns:

- **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:


Site ABC01
Fleet prod
cluster-abc01-adminprod
Admin cluster for production clusters
cluster-abc01-prod-shared
Multi-tenant production cluster
cluster-abc01-prod-team-i
Single-tenant cluster for Team I
cluster-abc01-prod-team-j
Single-tenant cluster for Team J
Team E
Namespace
Team F
Namespace
Team G
Namespace
Team I
Namespace
Team J
Namespace
Fleet staging
cluster-abc01-adminstage
Admin cluster for staging clusters
cluster-abc01-staging-shared
Multi-tenant staging cluster
cluster-abc01-staging-team-i
Single-tenant cluster for Team I
cluster-abc01-staging-team-j
Single-tenant cluster for Team J
Team E
Namespace
Team F
Namespace
Team G
Namespace
Team I
Namespace
Team J
Namespace

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

Observability

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

Cloud Monitoring automatically creates a multi-project metrics scope inside the fleet host project. Fleet deployments that contain cluster members from different projects can see a unified view of cluster resource metrics coming from any of their projects.

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 GKE 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

GKE Enterprise 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 GKE Enterprise 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 Config Sync 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 Config Sync. 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 Config Sync .yaml files in one or more Git repositories. If stored in a single repository, .tf files cluster creation and .yaml files for Config Sync 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

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/kubernetes-engine/distributed-cloud/vmware/docs/how-to/plan-ip-addresses
9. https://cloud.google.com/kubernetes-engine/distributed-cloud/vmware/docs/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/kubernetes-engine/enterprise/docs/resources/partner-storage
12. https://cloud.google.com/kubernetes-engine/enterprise/docs/concepts/anthos-connectivity
13. https://cloud.google.com/kubernetes-engine/distributed-cloud/bare-metal/docs/installing/registry-mirror#create_clusters_from_the_registry_mirror
14. https://cloud.google.com/kubernetes-engine/distributed-cloud/bare-metal/docs/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/kubernetes-engine/distributed-cloud/vmware/docs/how-to/logging-and-monitoring
19. https://cloud.google.com/service-mesh/docs/observability-overview
20. https://cloud.google.com/stackdriver/pricing
21. https://cloud.google.com/kubernetes-engine/fleet-management/docs/quotas
22. https://cloud.google.com/kubernetes-engine/distributed-cloud/vmware/docs/concepts/scalability#gke_hub
23. https://registry.terraform.io/providers/hashicorp/google/latest/docs