Google Cloud Whitepaper
December 2021

Google Cloud security foundations guide

Google Cloud

Table of contents

Disclaimer

The content contained in this document is correct as of December 2021. This whitepaper represents the status quo as of the time it was written. Google Cloud's products, security policies, and systems might change going forward as we continually improve protection for our users.

I. Security Foundations Blueprint

This guide presents an opinionated view of Google Cloud security best practices, organized to allow users to adopt or adapt them and then automatically deploy them for their estates on Google Cloud. This document can be useful to you if you are a CISO, security practitioner, risk or compliance officer.

Cloud adoption continues to accelerate across enterprises, with more businesses moving from investigating the use of public cloud infrastructure to actually delivering production services to their customers through public clouds. Conventionally, security in public clouds differs intrinsically from customer-owned infrastructure because there is a delineation of shared responsibility for security between the customer and the cloud provider. Figure 1.1.1 shows a matrix of the conventional shared security responsibility for workloads in the cloud.

Figure 1.1.1 Shared security responsibilities

Google Cloud product and service offerings range from classic platform as a service (PaaS), to infrastructure as a service (IaaS), to software as a service (SaaS). As shown in Figure 1.1.1, the conventional boundaries of responsibility between you and the cloud provider change based on the services you've selected. At a minimum, as a part of their shared responsibility for security, public cloud providers should enable you to start with a solid, secured foundation. Providers should then empower and make it easy for you to understand and execute your part of the shared responsibility model.

The catalog of Google Cloud offerings continues to grow rapidly. Each Google Cloud service exposes a wide range of configurations and controls so that you can customize it to match your business and security needs. In creating and setting-up your core infrastructure, our goal is to get you started faster and more securely by encoding key Google Cloud security best practices by default in this opinionated security foundations blueprint. You can then build on top of a **reliable, secured foundation** and either optimize the controls or take advantage of additional service-specific security guidance from our posture blueprints.

At Google, with our recent announcement for the Risk Protection Program, we seek to do even more and move from a traditional model of shared responsibility between customers and their cloud providers to **shared fate**. In this approach, we are active partners with you to deploy securely on our platform and not just delineators of where our responsibility ends.

1. Starting off with security in mind

Cloud Security is different from on-premises security because of the combination of the following:

- Differences in security primitives, visibility, and control points within the infrastructure, products and services.
- New cloud-native development methodologies like containerization and DevSecOps.
- The continued velocity and variety of new cloud products and services and how they can be consumed.
- Cultural shifts in how organizations deploy, manage, and operate systems.
- Understanding and distribution of risk between customers and cloud providers

You have many decisions to make when setting up your Cloud deployment. You might need help deploying workloads with secure defaults simply, quickly, and effectively in order to accelerate your ability to deliver business impact and value to your customers. But you might not have the time to build the new skills necessary to cope with the differences and new challenges of a cloud transition. Therefore, you can often benefit from curated and opinionated guidance for both a secured foundational starting point and for customization to match your specific needs.

Here's good news: you can build on Google Cloud more quickly, effectively, and securely by using these **security blueprints**, which add an important new set of layers in the hierarchy of security on Google Cloud. All in all, there are now five main parts, starting with core Google Cloud infrastructure as the base. The four subsequent layers are Google Cloud products and services; security foundations blueprints;

blueprints for security posture, workloads, and applications; and lastly, solutions that bundle the products, services, and blueprints with use case examples, with customer references, and with a commercial offer to simplify your understanding, adoption, and consumption. Each layer builds on top of the previous layers as discussed below.

1.1) Core Google Cloud infrastructure

Google has a global-scale technical infrastructure that's designed to provide security through the entire information processing lifecycle at Google. This infrastructure provides secure deployment of services, secure storage of data with end-user privacy safeguards, secure communications between services, secure and private communication with customers over the internet, and safe operation by administrators.

1.2) Google products and services

Google Cloud products and services are built on top of the core infrastructure and are designed and operated consistently in accordance with the cloud-native security principles that are shared in our BeyondProd whitepaper. They include built-in security features to enable access control, segmentation, and data protection. In addition, Google Cloud builds specific security products to help you meet your policy requirements and help protect your critical assets with our security products and capabilities.

1.3) Security foundations blueprints

The goal of **this security foundations blueprint** is to provide you with curated, opinionated guidance and accompanying automation that helps you optimize the native controls and services to build a secured starting point for your Google Cloud deployment. This security foundations blueprint covers the following:

- Google Cloud organization structure and policy
- Authentication and authorization
- Resource hierarchy and deployment
- Networking (segmentation and security)
- Key and secret management
- Logging
- Detective controls
- Billing setup
- Creating and deploying secured applications
- General security guidance

1.4) Security posture, workload, and applications blueprints

On top of strong security foundations, we provide additional blueprints for security posture, workload, and applications to give you curated, opinionated guidance to help design, build, and operate workloads and applications. Examples include our PCI on GKE blueprint and Healthcare Data Protection Toolkit.

1.5) Solutions

A combination of products and blueprints that incorporate Google Cloud security best practices provides you with the capabilities, architecture, and guidance for configuring, deploying, and operating specific sets of services. Solutions package these products together with vertical and use-case specific customization, additional examples, customer references, and a bundled commercial offer. Our goal is to simplify and accelerate your ability to adopt and apply the solutions for your organization's business needs. You can find our current solution portfolio on the Google Cloud solutions page.

Figure 1.1.2 shows the layers of security that you can build on when you use Google Cloud and follow the guidance and templates provided by the security blueprints.

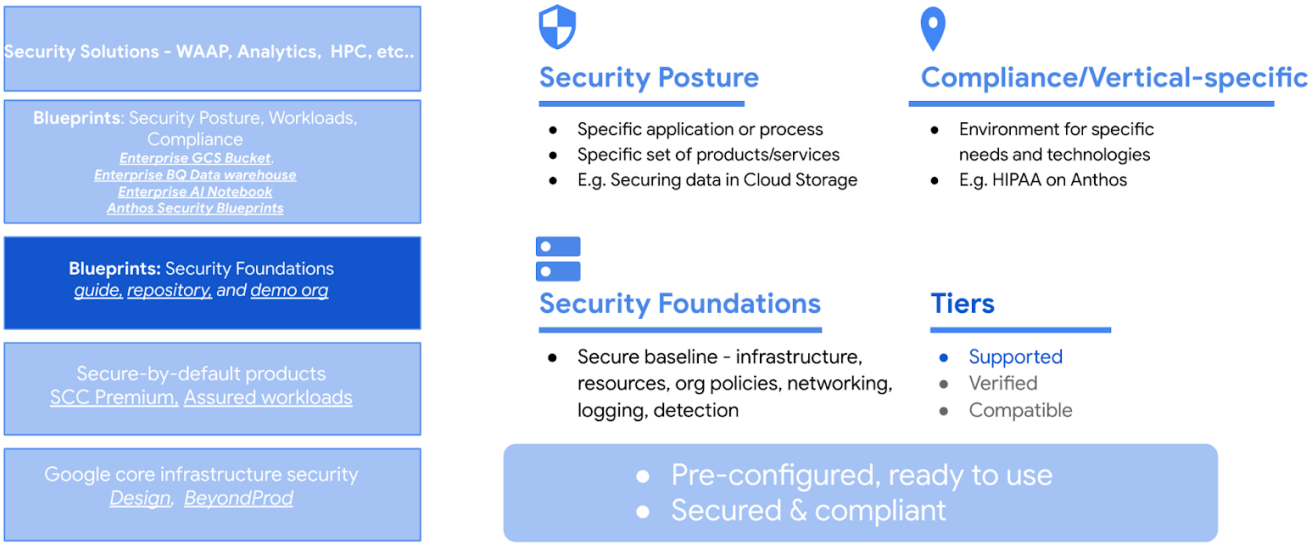

Figure 1.1.2 Google Cloud hierarchy of security

2. Beginning with a security foundations blueprint

This guide provides our opinionated security foundations blueprint and captures a step-by-step view of how to configure and deploy your Google Cloud estate. This document can provide a good reference and starting point because we highlight key topics to consider. In each topic, we provide background and discussion of why we made each of our choices. In addition to the step-by-step guide, this security foundations blueprint has an accompanying Terraform automation repository and an example demonstration Google organization so you can learn from and experiment with an environment configured according to the blueprint.

2.1) How you can use the security foundations blueprint

This guide can be useful to you as you take on one or more of the following roles in your organization:

- **Security leader** who needs to understand Google's key principles for Cloud Security and how they can be applied and implemented to secure your own organization's deployment.
- **Security practitioner** who needs detailed instructions on how to apply security best practices to set up, configure, deploy, and operate a security-centric infrastructure landing zone that's ready for you to deploy your workloads and applications to.
- **Security engineer** who needs to understand how to configure and operate multiple different security controls so that they correctly interact with one another.
- **Business leader** who needs your teams to accelerate their learning and understanding of the advanced skills they need on Google Cloud to meet your advanced needs. In this role, you also need to be able to share Google security reference documentation with your risk and compliance teams.
- **Risk and Compliance officer** who needs to understand the controls available on Google Cloud to meet their business requirements and how those controls can be automatically deployed. You also need to have visibility into controls drift and areas that need additional attention to meet the regulatory needs of your business.

In each of these cases, you can use this document as a reference guide. You can also use the provided Terraform scripts to automate, experiment, test, and accelerate your own live deployments. You can modify the provided scripts to customize them for your needs.

2.1.1) Create a better starting point for compliance

For the compliance and regulatory frameworks that are required for your business, you need a clear understanding of and evidence for the following:

- Whether the Google cloud services you choose meet the requirements.
- Whether your configuration and use of these Google cloud services continue to meet requirements.

For the first case, Google Cloud provides the Compliance resource center. This site enables you to search by framework, region, or industry to find which Google Cloud services are approved and help support your compliance.

For the second case, after you've deployed the security foundations blueprint, Security Command Center Premium provides you a dashboard overview and downloadable compliance reports of your starting posture for the CIS 1.0, PCI-DSS 3.2.1, NIST-800-53 and ISO27001 frameworks at the organization, folder, or project level.

Figure 1.2.1 shows the default compliance report for a sample project deployed in the security foundations blueprint compared against the CIS 1.0 and PCI DSS 3.2.1 frameworks. As the figure shows, a majority of the assessed compliance controls are configured out of the box from the blueprint. The missing controls are logging configuration controls that require user input before they can be set up correctly.

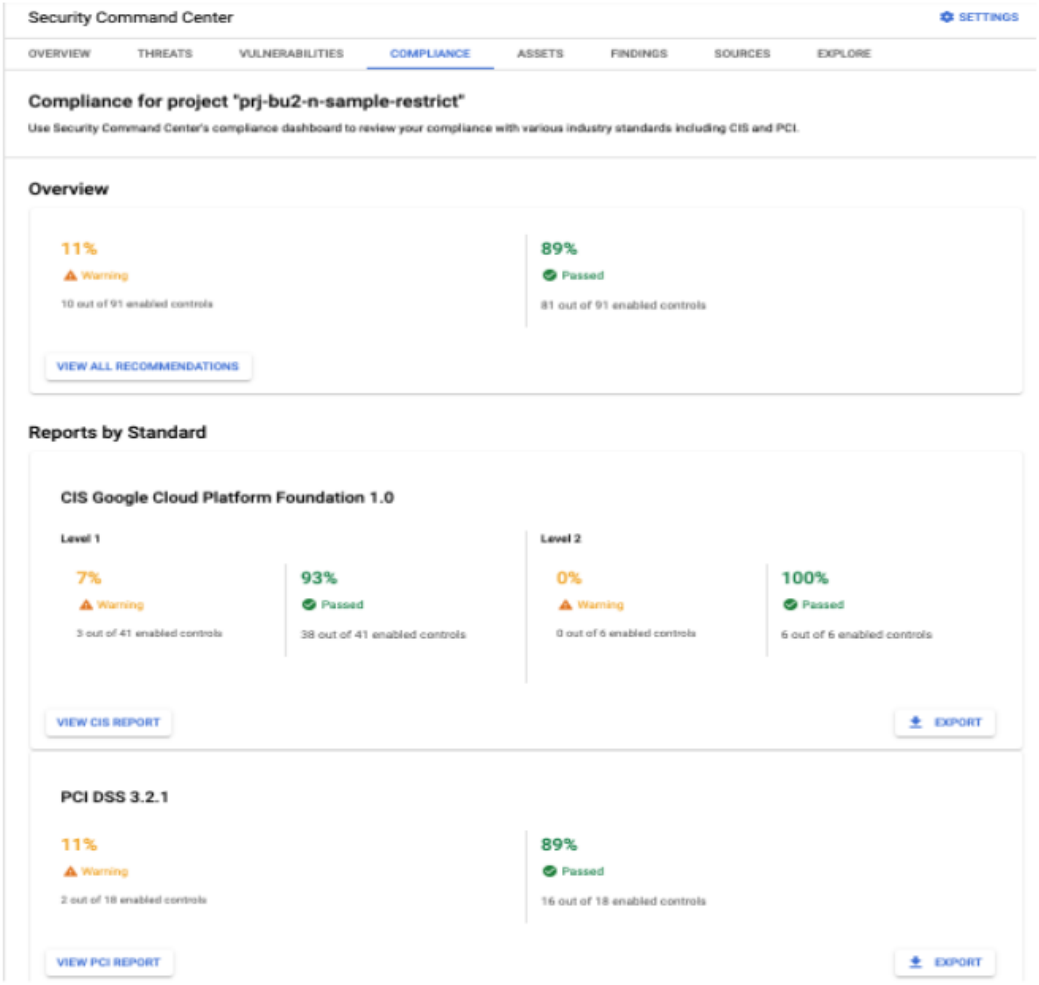

Figure 1.2.1 Security Command Center Premium compliance summary for a restricted network sample project

2.1.2) Implement key security principles

Beyond implementing compliance and regulatory requirements, you need to protect your infrastructure and applications.

The security foundation blueprint and the associated automation scripts help you adopt three Google Cloud security principles that are core to Google's own security. These are:

- Executing **defense in depth**, **at scale**, **by default**.
- Adopting the **BeyondProd** approach to infrastructure and application security.
- De-risking cloud adoption by moving toward a **shared fate** relationship.

2.1.2.1) Defense in depth, at scale, by default

A core principle for how Google secures its own infrastructure is that there should never be just one barrier between an attacker and a target of interest (defense in depth). Adding to this core principle, security should be scalable and it should be enabled by default.

The security foundations blueprint embodies these principles in multiple ways. Data is protected by default through multiple layered defenses using policy and controls that are configured across networking, encryption, IAM, detection, logging, and monitoring services.

As an example, the data in a production project by default has three levels of network protections: VPC segmentation, VPC service perimeters, and firewall rules. It's further protected by multiple levels of access protection using IAM, access context levels, and multi-factor validation of user identities.

The blueprint provides an example of a secured CI/CD pipeline to build applications that have access to the data; the security features of the pipeline include both build-time vulnerability assessments and deploy-time policy checks. In addition, the data itself is protected through customer-managed encryption keys (CMEKs). All admin access and data access can be logged using default audit logging. Security Command Center Premium provides ongoing security monitoring and detection. In addition, you can supplement this with custom detection through BigQuery.

2.1.2.2) BeyondProd

In 2019, we published BeyondProd, Google's new approach to native cloud security. This was motivated by the same insights that drove our BeyondCorp effort in 2014, because it had become clear to us that a perimeter-based security model wasn't secure enough. BeyondProd does for workloads and service identities what BeyondCorp did for workstations and users. In the conventional network-centric model, once an attacker breaches the perimeter, they have free movement within the system. Instead, the BeyondProd approach uses a zero-trust model by default. It decomposes historically large monolithic applications into microservices, thus increasing segmentation and isolation and limiting the impacted area, while also creating operational efficiencies and scalability.

A core concept in BeyondProd is that developers should be able to write and deploy secured applications without needing to have deep expertise in security, and without having to implement security features themselves. Therefore, the key tenets of BeyondProd are:

- Security is holistic and built in, not bolted on.
- Protection of the network at the edge, while still required, is not the primary and not the only defense point.
- No inherent mutual trust exists between services.
- Trusted machines run code with known provenance.
- Logical choke points are used for consistent policy enforcement across services; for example, to ensure authorized data access.
- Change rollout is simple, automated, and standardized.
- Isolation is enforced and monitored between workloads.

The security foundations blueprint jumpstarts your ability to adopt BeyondProd. Security controls are designed into and integrated throughout each step of the blueprint architecture and deployment. The different layers of security controls are designed to work together and are not an afterthought. No inherent mutual trust exists between services in the system. Predefined IAM roles create default separation of duties. The resource hierarchy design creates clear IAM and network boundaries between projects by default. VPC service perimeters enable you to enforce segmentation and isolation by service and by workload. Logical control choke points like organization policies provide you with consistent, default preventive policy enforcement at create time and deploy time. Centralized and unified visibility through Security Command Center Premium provides unified monitoring and detection across all the resources and projects in your organization.

You can learn more about how Google Cloud capabilities map to BeyondProd principles in Table 2.12.5.

2.1.2.3) Shared fate

To move you from shared responsibility to shared fate, we believe that it's our responsibility to be active partners with you in deploying and running securely on our platform. This means providing holistic integrated capabilities throughout your Day 0 to Day N journey, as in the following:

- Design and build time: Supported security foundations and posture blueprints that encode best practices by default for your infrastructure and applications.
- Deploy time: "Guard rails" through services like organization policies and assured workloads that enforce your declarative security constraints.
- Run time: Visibility, monitoring, alerting, and corrective-action features through services like Security Command Center Premium.

Together, these integrated services reduce your risk by starting and keeping you in a more trusted posture with better quantified and understood risks. As shown in Figure 1.2.2, this improved risk posture can then allow you to take advantage of risk protection services, thus de-risking and ultimately accelerating your ability to migrate and transform in the cloud.

Figure 1.2.2 The value of shared fate

2.2) Updates from v2

The updated v2.5 guide and the accompanying repository of Terraform scripts represents an incremental update from v2. In this version, we have focused on adding some best practices that were requested by customers in regulated industries, including the following:

- Clarifications on how to control and restrict the location of key services when you're deploying the security foundations blueprint landing zone. (Section 4.4.1)
- Information about how to deploy Assured Workload projects for selected services and applications to help meet regulatory requirements. (Section 4.4.2)

In addition, v2.5 includes fixes and corrections based on customer feedback. We also include an errata section in our Terraform documentation.

The balance of this document is organized into sections that cover the following:

- This introduction
- The foundation security model
- Foundation design
- The example.com sample that expresses the opinionated organization structure
- Resource deployment
- Authentication and authorization
- Networking
- Key and secret management
- Logging
- Detective controls
- Billing
- Creating and deploying secured applications
- General security guidance

II. Step-by-step guide

1. Introduction

This section takes you step by step through building a Google Cloud deployment with a secured foundation that you can use to run workloads securely in the cloud. Figure 2.1.1 shows the high-level architecture of example.com, the reference organization used in this guide. The diagram shows a hybrid organization. Some components run on premises, while others are deployed in Google Cloud, with dedicated high-speed connections between the components. The Google Cloud resources are deployed using infrastructure as code (IaC) cloud foundation processes that help make the deployment faster and more repeatable than manual processes.

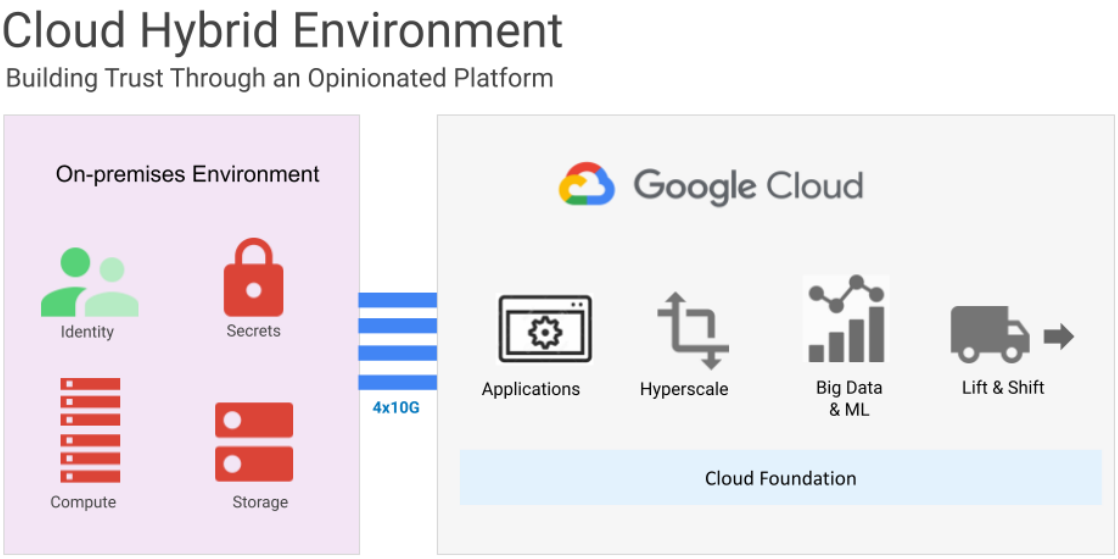

Figure 2.1.1 The example.com high-level architecture

This guide achieves security for example.com by using Google's opinionated principles in areas including the following:

- Security
- Governance
- Compliance
- Auditing
- Provisioning
- Networking
- High availability
- Scaling
- Monitoring
- Alerting
- Chargeback support

The security of example.com takes a unified approach to governance, security objectives, compliance, identity, connectivity, and workloads. This unified approach is used throughout this guide.

You can find scripts, code, and other deployment artifacts for the example.com organization in the `terraform-example-foundation` GitHub repository.

2. Google Cloud foundation security model

Foundation security in Google Cloud is enabled through a combination of preventative controls and detective controls. Preventative controls are realized through policy and architecture. Policy is defined as a series of programmatic constraints that protect the organization from threats. Architecture is defined as how infrastructure constructs can be used to protect the organization.

Detective controls are defined as monitoring capabilities that look for anomalous or malicious behavior within the organization. Figure 2.2.1 depicts how these three pillars of the security model come together to form the example.com reference architecture.

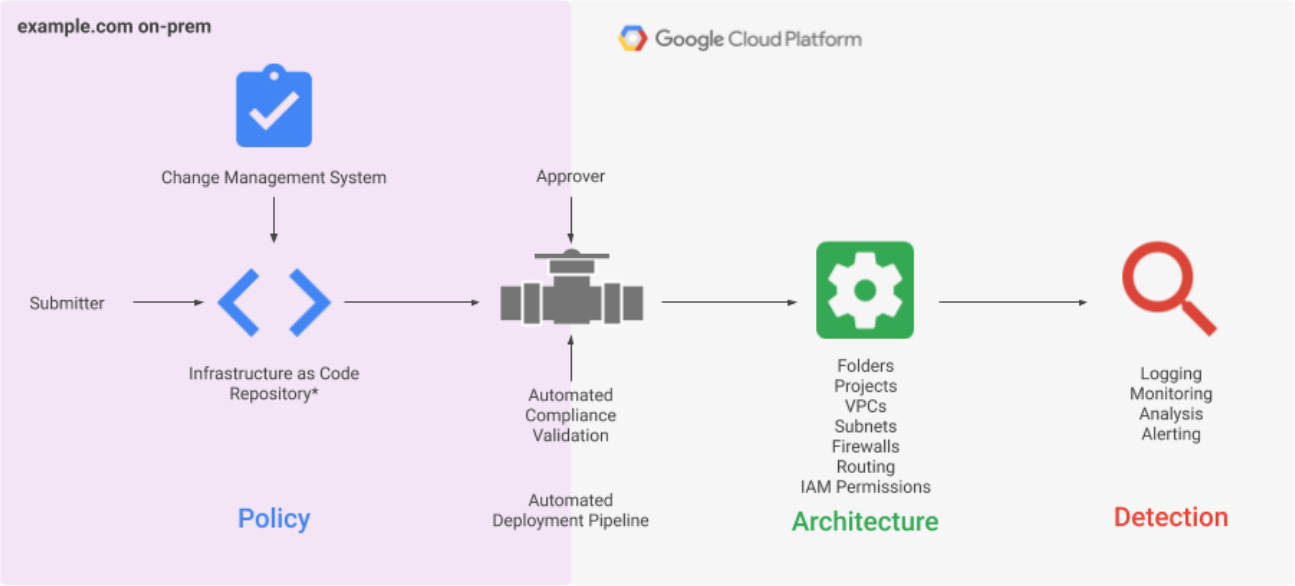

Figure 2.2.1 The example.com security model

Deployment best practices, as detailed in Section 5, are implemented through codifying the Google Cloud infrastructure into Terraform modules, putting the modules under source code control, and deploying those modules through an automated pipeline that has policy validation and approval stages. The architectural constructs, as detailed in Section 6 through Section 9, encompass the folder and project structure of the Google Cloud organization, connectivity, networking structure, firewalls, and identity management. Detective controls are detailed in Section 10 and explain Google Cloud's platform capabilities to detect vulnerabilities, misconfiguration, and malicious behavior in order to enable security posture management. Section 10 also covers how to leverage Google Cloud's native capabilities in security analytics, as well as how to integrate Google Cloud capabilities with third-party SIEM tools. Section 11 describes how to interpret billing data from a security perspective. Section 12 describes an example application to show how to design and deploy applications securely, taking advantage of Google Cloud components and controls.

3. Google Cloud foundation design

3.1) Key architectural decisions

The example.com reference architecture that's described in this document is based on certain architectural decisions; if you need to make changes to those assumptions, the example.com architecture also needs to be modified. Table 2.3.1 lists architectural decisions.

Decision area	Decision
Organization structure	A single organization is used for all environments and services for example.com to manage all resources and policies in one place.
	The folder hierarchy has a single layer, consisting of bootstrap, common, production, non-production, and development folders to allow for segregation of policies, privileges, and access.
	Google Cloud organization policies are used to augment the organization's security posture by allowing you to define resource configuration constraints that apply consistently across all projects.
Resource deployment	Foundation resources are deployed through a deployment pipeline to enable automated and standardized review, approval, and rollback.
	Terraform is used as the infrastructure as code (IaC) tool.
	An on-premises Git repository is used as a source repository for the Terraform modules.
	The deployment pipeline's actions are initiated from the on-premises environment.
	An approval stage in the pipeline is needed to deploy resources.
	Workloads are deployed through separate pipelines and access patterns.
Authentication and authorization	Google Cloud federates with the example.com on-premises Active Directory identity management system.
	Single sign-on (SSO) is used to allow users to log into Google Cloud.
	A firecall process is used to provide elevated access to the Google Cloud environment.
	Groups are used for assigning Cloud IAM permissions.
Networking	Dedicated Interconnect connections are used to connect the on-premises environment with Google Cloud in a configuration that provides an SLA of 99.99%.
	Shared VPC is used as the primary networking topology to enable centralized management of firewall rules and connectivity.

	VPC Service Controls provide perimeter protection for services that store highly sensitive data to enable service-level segmentation.
	Access to Google Cloud APIs from the cloud and from on-premises is through private IP addresses.
	Address allocation for Google Cloud is a contiguous RFC 1918 /16 IP address space.
	Cloud DNS is used in Google Cloud, and DNS forwarding is used to communicate with on-premises DNS servers.
	Tag-based firewall rules are used to control network traffic flows.
Key and secret management	Cloud KMS is used for cryptographic keys
	Secret Manager is used to store secrets.
	Separate Secret Manager instances are used to store organization-level and folder-level secrets.
Logging	Organization-level log sinks are used to aggregate logs.
	Logs are sent to Cloud Storage, to BigQuery, and to a SIEM through Cloud Pub/Sub.
Detective controls	An enterprise SIEM product integrates with Google Cloud Logging.
	Security Command Center Premium is enabled to detect infrastructure misconfigurations, vulnerabilities, and active threat behavior, and to share those findings with the enterprise SIEM tools.
	Logs in BigQuery are used to augment detection of anomalous behavior by Security Command Center Premium.
Billing	Billing alerts are used on a per-project basis with thresholds set at 50%, 75%, 90%, and 95%.

Table 2.3.1 Key architectural decisions

3.2) Pre-work

The reference architecture that's described in this document is based on the assumption that you've completed the following pre-work:

- ❏ You have set up Cloud Identity.
- ❏ You have validated your Google Cloud domain.
- ❏ You have established a billing account.
- ❏ You have reconciled conflict accounts.
- ❏ You have set up your Dedicated Interconnect connections.

For more information, see the Google Cloud onboarding documentation.

3.3) Naming conventions

You should have a standardized naming convention for your Google Cloud resources. Table 2.3.2 describes recommended conventions for creating Google Cloud element names for the example.com reference architecture. Table 2.3.3 describes the field values that are used in the names.

Note: Fields listed in braces ({ }) are optional.

Resource	Naming convention
Folder	`fldr-`*`environment`* Example: `fldr-prod`
Project	`prj-`*`business-code`*`-`*`environment-code`*`{-`*`project-label`*`}-`*`unique-number`* Example: `prj-acde-p-shared-base-43212`
VPC	`vpc-`*`environment-code`*`-`*`vpc-type`*`{-`*`vpc-label`*`}` Example: `vpc-p-shared-base`
Subnet	`sb-`*`vpc-name`*`-`*`region`*`{-`*`subnet-label`*`}` Example: `sb-p-shared-base-us-east1-net1`
Firewall	`fw-`*`vpc-name`*`-`*`priority`*`-`*`direction`*`-`*`action`*`-`*`src-label`*`-`*`dest-label`*`-`*`protocol`*`-`*`port`*`{-`*`firewall-label`*`}` Example: `fw-p-shared-base-1000-i-a-all-all-tcp-80`
Cloud router	`cr-`*`vpc-name`*`-`*`region`*`{-`*`cloud-router-label`*`}` Example: `cr-p-shared-base-us-east1-cr1`
Route	`rt-`*`vpc-name`*`-`*`priority`*`-`*`instance-tag`*`-`*`next-hop`*`-{`*`route-label`*`}` Example: `rt-p-shared-base-1000-all-default-windows-activation`
Cloud Interconnect connection	`ic-`*`onprem-dc`*`-`*`colo`* Example: `ic-dal-lga-zone1-1422`
Cloud Interconnect VLAN attachment	`vl-`*`ic-name`*`-`*`cr-name`* Example: `vl-dal-da1-zone1-p-shared-base-us-east1-cr1`
Group	`grp-gcp-`*`group-label`* Example: `grp-gcp-billing-admin`

Role	`rl-`*`function`*`{-`*`role-label`*`}` Example: `rl-compute-admin`
Service account	`sa-{-`*`service-account-label`*`}` Example: `sa-p-acde-shared-base-data-bkt-reader`
Storage bucket	`bkt-`*`project-name`*`{-`*`bucket-label`*`}` Example: `bkt-p-acde-shared-base-data`

Table 2.3.2 Naming conventions for example.com

Field	Description	Values
`environment`	A description of the folder-level resources within the Google Cloud organization.	`bootstrap`, `common`, `prod`, `nonprod`, `dev`
`environment-code`	A short form of the environment field.	`b`, `c`, `p`, `n`, `d`
`business-code`	A 4-character code that's used to associate a project with a business unit or group.	A uniquely identifiable 4-character code. For common projects that are not related to a business unit, `zzzz` is used.
`unique-number`	A globally unique identifier.	A 5-digit integer
`vpc-type`	The type of VPC network that's being established.	`shared`, `service`, `float`, `nic`, `peer`
`region`	The region that the resource is located in.	Any valid Google Cloud region. Short forms are used for some names and directions: Australia (`au`), North American (`na`), South America (`sa`), Europe (`eu`), southeast (`se`), and northeast (`ne`).
`priority`	A numerical value that specifies the priority of the Google Cloud route or firewall rule.	For details, see the documentation for firewall priorities and routing order.
`direction`	The direction of traffic relative to Google Cloud that the firewall applies.	`i` for ingress, `e` for egress
`action`	The action to take if a firewall rule matches.	`a` for allow, `d` for deny
`src-label`	The instance source label to which a firewall is applied.	`all` (indicating `0.0.0.0/0`), the source IP address range, or source tags (list)

`dest-label`	The instance destinations to which a firewall is applied.	`all`, the destination tags (list), or the service accounts (list)
`protocol`	The protocols to which a firewall is applied.	`all`, a single protocol, or a combination of protocols (`tcp`, `udp`, `tcpudp`, and so on)
`port`	The port or port range on which a firewall is applied.	A port number or port number range
`instance-tag`	The instances to which a route is applied.	instance tags
`next-hop`	The next-hop destination where the route will direct traffic, if applicable.	`default`, an instance ID, an IP address, a VPN tunnel name, oran internal load-balancer address
`onprem-dc`	The name of your data center to which an interconnect is connected.	Customer dependent
`colo`	The colocation facility name that the interconnect from the on-premises datacenter is peered with.	A valid Google Cloud colocation facility code
`function`	The name of the resource type that a custom role is associated with.	Resource dependent
`*name`	The name of a resource without its prefix.	Resource dependent
`*label`	A descriptive field to enhance the description of the resource.	Resource dependent

Table 2.3.3 Naming convention field values

4. The example.com Google Cloud organization structure

The foundation of creating deployments within Google Cloud is the organization, which may be created through either our Cloud Identity service or Google Workspace. The Google Cloud organization provides a resource hierarchy that provides an ownership structure for resources and attachment points for policy and access control.

The resource hierarchy consists of folders, projects, and resources, and it defines the shape and use of Google Cloud services within an organization. Policies are inherited, and these policies can't be altered by resource owners who are lower in the hierarchy. Folders and projects inherently provide you with secure-by-default isolation because all external network and access permissions are denied by default and therefore must either be explicitly enabled or enabled by inheritance from a higher level in the hierarchy.

Folder structure is dependent on the nature of the organization and the type of workload being deployed. Figure 2.4.1 shows the example.com reference architecture.

The architecture consists of five folders as described in Section 4.1 and of a number of projects that are detailed in Section 4.2. The organizational structure for example.com is codified in Terraform modules and deployed through the deployment pipeline as detailed in Section 5.

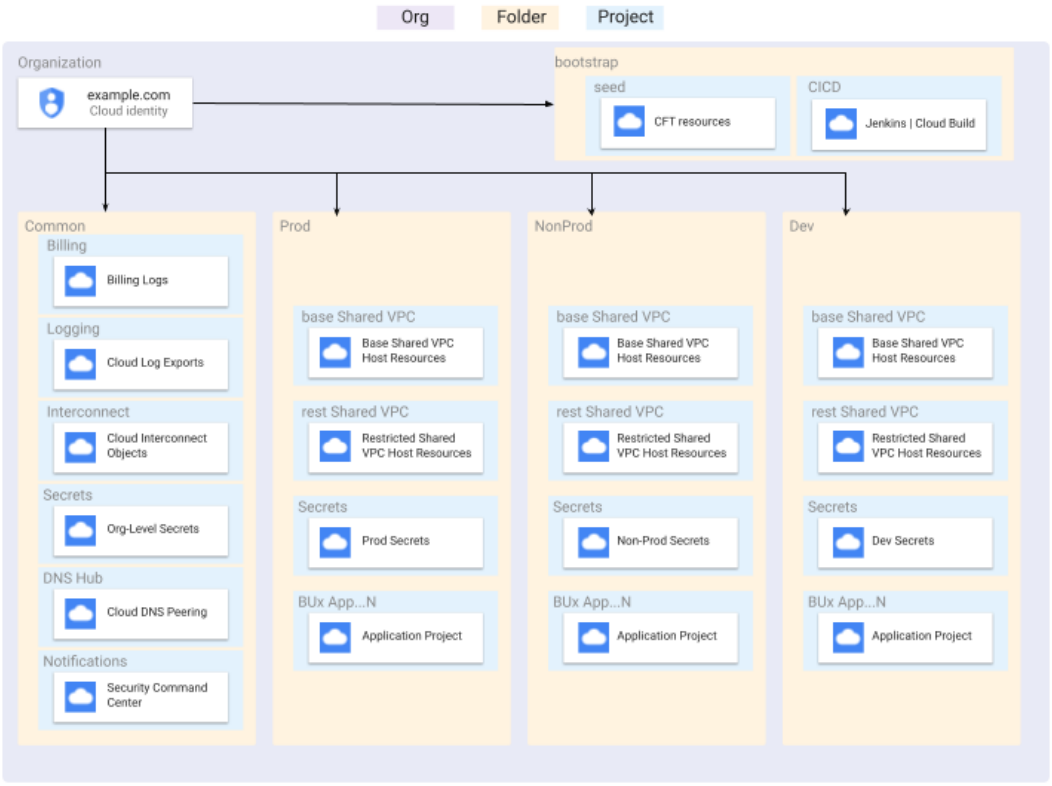

Figure 2.4.1 The example.com organization structure

Note: The example.com folder structure provides a basis for a typical application-deployment architecture. However, for production scenarios, you should tailor your deployment strategy to your specific needs around resource isolation, geographic operating needs, business unit autonomy, data isolation, and access control. In addition, some of your workloads such as analytics might need production data in a non-production environment. This might lead you to create different environmental classifications, such as an analytics folder. You might also need to create multiple Google Cloud organizations to increase separation and isolation.

4.1) Folders

You use folders as a method of grouping projects into related groups. You can apply security policies and access permissions at the folder level; these policies and permissions are then inherited by resources within the folder structure.

The key concept of folders is that they separate projects into logical groupings; you can then apply consistent policy to the child projects at the folder level to ease administrative overhead. As shown in Table 2.4.1, the reference architecture used by example.com uses five folders. Three of them (`production`, `non-production`, and `development`) separate the organizations into different environments. These folders are deployed through the deployment pipeline detailed in Section 5.

Folder	Description
`bootstrap`	Contains the seed and CICD projects that are used to deploy foundation components.
`common`	Contains projects with cloud resources used by the organization.
`production`	Contains projects with cloud resources that have been promoted into production.
`non-production`	Contains a replica of the production environment to let you test workloads before you put them into production.
`development`	Used as a development and sandbox environment.

Table 2.4.1 Folders used in the example.com reference architectures

4.2) Projects

Projects in Google Cloud are constructs that contain cloud resources. Each project inherits the policy of the parent folder or folders and of the Google Cloud organization. You can enable or disable APIs within each project to support the services that you need for that project. Each project is associated with a billing account, as explained in Section 5.3.4.

4.2.1) Common folder and bootstrap folder projects

In the example.com architecture, a series of projects reside under the `common` folder and `bootstrap` folder that contain resources that are used across the example.com organization. These projects,

detailed in Table 2.4.2, provide various enterprise functions and are created through the infrastructure deployment pipeline.

Project	Description	More information
CICD	Houses the deployment pipeline that's used to build out the foundation components of the organization. This project is highly restricted.	Section 5
seed	Contains the Terraform state of your infrastructure and Terraform service account to update the state. This project is highly restricted.	Section 5
interconnect	Houses the Cloud Interconnect connections that provide connectivity between an enterprise's on-premises environment and Google Cloud.	Section 7.3
DNS hub	Provides a central point of communication between an enterprise's on-premises DNS system and Cloud DNS.	Section 7.5
org-secrets	Contains organization-level secrets.	Section 8
logging	Provides a destination for log sink and detective controls.	Section 9
Security Command Center	Provides a standalone project for managing Security Command Center alerting.	Section 10.1
billing	Contains a BigQuery dataset with the organization's billing exports.	Section 11

Table 2.4.2 Projects in the `common` folder

4.2.2) Projects present in all environment folders

Under each environment folder (`production`, `non-production`, and `development`) in the example.com reference architecture, a series of common projects are created, as detailed in Table 2.4.3. These projects provide resource connectivity to the on-premises environment and to a deployment environment for workload-specific resources.

Project type	Description
Base Shared VPC host project	Provides the project for the Shared VPC network that hosts the organization's workloads that don't require VPC Service Controls.
Restricted Shared VPC host project	Provides the project for the Shared VPC network that will host the organization's workloads that run within a perimeter controlled by VPC Service Controls.
secrets	Contains folder-level secrets.
application projects	Involve projects where application resources are deployed. Project deployment patterns are described in Section 7.1.1 and consist of Shared VPC service project patterns, floating project patterns, peered project patterns, and dual-NIC project patterns.

Table 2.4.3 Environment folder project types

4.3) Organization policy setup

You can use the Organization Policy Service to apply policies to a Google Cloud organization. The organizational policies are inherited by all child resources of the resource node to which the policy is set, unless an explicit override is set on a child resource. Organization policy constraints in the example.com reference architecture are defined in Terraform modules and deployed through the deployment pipeline. Table 2.4.4 provides a list and description of the policies that are set as part of the example.com foundation.

Please note that if you are setting up a regulated workload that must adhere to a compliance regime, our Assured Workloads service (Section 4.4.2) should be used, and will automatically apply technical controls on your behalf to confidently secure your regulated workloads.

Policy constraint	Description	Recommended value
`compute.disableNestedVirtualization`	(boolean) When true, disables hardware-accelerated nested virtualization for all Compute Engine VMs.	`true`
`compute.disableSerialPortAccess`	(boolean) When `true`, disables serial port access to Compute Engine VMs.	`true`
`compute.disableGuestAttributesAccess`	(boolean) When `true`, disables Compute Engine API access to the guest attributes of Compute Engine VMs.	`true`
`compute.vmExternalIpAccess`	(list) Defines the set of Compute Engine VM instances that are allowed to use external IP addresses.	`deny all=true`
`compute.skipDefaultNetworkCreation`	(boolean) When `true`, causes Google Cloud to skip the creation of the default network and related resources during Google Cloud project resource creation.	`true`
`compute.restrictXpnProjectLienRemoval`	(boolean) When `true`, restricts the set of users that can remove a Shared VPC project lien.	`true`
`sql.restrictPublicIp`	(boolean) When `true`, restricts public IP addresses on Cloud SQL instances.	`true`
`iam.allowedPolicyMemberDomains`	(list) Defines the set of members that can be added to Cloud IAM policies. The allowed/denied list must specify one or more Cloud Identity or Google Workspace customer IDs. Note that domain restricted sharing can interfere with some Google Cloud services, and you might need to provide exceptions for some Google Cloud services.	*`your-cloud-identity-id`*

iam.disableServiceAccountKeyCreation	(boolean) When true, disables the creation of service account external keys.	true
storage.uniformBucketLevelAccess	(boolean) When true, requires buckets to use uniform IAM-based bucket-level access.	true
iam.automaticIamGrantsForDefaultServiceAccounts	(boolean) When true prevents the default App Engine and Compute Engine service accounts that are created in your projects from being automatically granted any IAM role on the project when the accounts are created	true

Table 2.4.4 Organizational policies in example.com

Note: You might need to modify organization policy constraints to accommodate certain application types or workflows—for example, configuring new log exports, purposefully creating public-facing buckets, or giving a VM an external IP address for internet access.

4.4) Additional policy controls

In addition to the organizational policy constraints detailed in the previous section, Google Cloud has additional policies that you can apply to an organization to tighten the organization's security posture. Table 2.4.5 lists the additional policy controls that are used in the example.com reference architecture.

Control	Description	Management
Limit session and gcloud timeouts	You can change the session timeout for Google Cloud sessions (including the gcloud tool) to durations as low as 1 hour.	Google Workspace or Cloud Identity Admin Console
Disable Cloud Shell	Google Cloud provides a native Cloud Shell interface for working with the cloud environment. You can disable this shell.	Admin Console
Use phishing-resistant security keys	You can enable phishing-resistant security keys as a second-factor authentication (2FA) method that helps provide protection against automated bots, bulk phishing, and targeted attacks. The keys use public key cryptography to verify a user's identity, and they use the URL of the login page to help make sure that an attacker can't access a user account even if the user is tricked into providing a username and password.	Google Workspace or Cloud Identity Admin Console. If you're federating identities from an external system, you might need to configure this in your identity provider, or add an additional layer of protection on top of your existing SSO settings directly within Cloud Identity.
Enable access transparency	Access transparency provides you with logs that capture the actions that Google personnel take when accessing your content. You must be on Gold, Platinum, Enterprise, or Premium support plans.	Contact Google Cloud Support
Enable access approval	Access approval enables you to explicitly approve access to your data or configurations on Google Cloud	Google Cloud Console

	before Google Cloud support or engineering can access the data.	

Table 2.4.5 Additional example.com policy controls

4.4.1) Restricting resource locations

You can limit the physical location of a new resource in Google Cloud by using resource location constraints. You use the same mechanism that's used to create organizational policy controls as described in Section 4.3 to restrict resource locations in the example.com organization. When you set up resource restriction policies, you can specify the location type (multi-region, region, or zone) that's associated with the policy. Be aware that if you use resource location constraints, some services might not work correctly, as described in Resource Locations Supported Services.

4.4.2) Assured Workloads

For your regulated workloads that are required to be compliant with specific regulatory compliance regimes (for example, FedRAMP Moderate, FedRAMP High, IL4, CJIS, or ITAR), Google Cloud provides Assured Workloads. You can apply the technical controls and resource location restrictions provided by Assured Workloads to a folder to support your regulatory compliance requirements.

Assured Workloads provides the following security controls:

- Data residency. Customer data is stored in a multi-region or a region that you select. If your developer attempts to store data at rest in a region outside of the selection, that action will be restricted.
- Personnel data access controls based on attributes. This feature ensures that only Google personnel who satisfy compliance requirements such as location and background checks are able to perform compliant data access requests, and only while performing support operations for Google's regulated customers.
- Personnel support case ownership controls based on attributes. Only Google support personnel who satisfy compliance requirements such as location and background checks are able to provide support for Assured Workloads customers.
- Encryption. Google-managed and customer-managed encryption keys are FIPS-140-2 compliant, and they support regulatory compliance for encryption.

You should use Assured Workloads only if your Google Cloud use case is actively subject to regulatory compliance, or if you want to have additional security controls applied to your projects. You might want these controls so that your projects conform to a compliance regime's security posture, even if the projects are not subject to a direct requirement from a regulatory body. If you use Assured Workloads, we recommend that you select only the compliance regime for your use case. If you are not bound by a compliance regime but you want to apply additional security controls, you can use the *US Regions and Support* or *EU Regions and Support* compliance regime selector in the Assured Workloads creation workflow. To see locations where Assured Workloads is available, see Assured Workloads locations.

Assured Workloads are enabled at the folder level, and they require that the folder be registered with Google Cloud before you deploy workloads. To register a folder, you use an onboarding form. Registration can take up to 10 business days. After the folder has been registered, additional folders and projects can be created in that folder (in this blueprint, the Assured folder) for workloads that are subject to various compliance regimes. The example.com code repository provides you with the deployment artifacts that you can use to create compliant workload projects after the folder has been registered.

4.4.3) Google Cloud Console

Google Cloud lets you restrict access to the Google Cloud Console. You can use context-aware access to provide granular access controls to the console based on a user's identity and the context of their access request. To use context-aware access, you create an access level in Access Context Manager that allows access only from a specified range of IP addresses. For example, you can allow access from the public IP address range of your corporate network. After you create the access context, you create a Google Group in Cloud Identity and add users to that group who have context-aware access restrictions.You then create an access binding between the access level and the Google group.

Note: There is an important distinction between allowing any 2FA scheme, most of which don't resist phishing, and requiring a phishing-resistant 2FA mechanism such as Titan Security Keys or other keys based on the phishing-resistant FIDO U2F or CTAP1 standards. Enterprises can enforce 2-step verification, including security key verification, on accounts gradually, which allows migration. Enforcement can then be enabled for more privileged users first and expanded as practical.

5. Resource deployment

Resources in the example.com reference architecture can be grouped into one of two categories: foundation or workload components. Foundation resources need to be tightly secured, governed, and audited to help avoid exposing the enterprise to any security or compliance risks. Foundation resources include resources in the hierarchy such as organizational policies, folders, projects, APIs, identities (for example, service accounts), role bindings, custom role definitions, Shared VPC networks, subnets, routes (dynamic and static), firewall rules, and Dedicated Interconnect connections. Workload components include resources such as Cloud SQL databases and Google Kubernetes Engine clusters.

Resources within Google Cloud can be deployed from the Google Cloud Console web interface, using the `gcloud` command-line tool, using API calls directly, or using an infrastructure as code (IaC) tool. For creating the foundation elements of your enterprise's environment, you should minimize the amount of manual configuration so that you limit the possibility of human error. In the example.com reference architecture, Terraform is used for IaC, deployed through a pipeline that's implemented using Jenkins.

The combination of Terraform and Jenkins allows the foundation elements of example.com to be deployed in a consistent and controllable manner. The consistency and controllability of this approach helps enable governance and policy controls across the example.com Google Cloud environment. The example.com reference pipeline architecture is designed to be consistent with most enterprise's security controls.

5.1) CICD and seed projects

The example.com organization uses the Cloud Foundation Toolkit to create the basic resources necessary to stand up an IaC environment within Google Cloud. This process creates a `bootstrap` folder at the root of the organization that contains a `CICD` project and a `seed` Terraform project.

The `CICD` project is a tightly controlled project within the organization hierarchy that's used to host the Jenkins deployment pipeline. It also hosts a service account that's used to run the Jenkins agents' Compute Engine instances. The Jenkins service account can impersonate the Terraform service account, which is located in the `seed` project, and which has permissions to deploy the foundation structures within the organization. The `CICD` project is created through a scripted process. It has direct connectivity to the on-premises environment, separate from the connectivity described in Section 7.2.

The `seed` project contains the Terraform state of the foundation infrastructure, a highly privileged service account that's able to create new infrastructure, and the encryption configuration to protect that state. When the CI/CD pipeline runs, it impersonates this service account. The reason for having independent `CICD` and Terraform `seed` projects is due to separation of concerns. While Terraform is used as the IaC tool and has its own requirements, the deployment of that IaC is the responsibility of the CI/CD pipeline.

Note: The example.com reference architecture describes using Jenkins for the deployment pipeline. If you want to use a Google Cloud native service, the code repository also incorporates a Cloud Build-based pipeline.

5.2) Deployment pipeline architecture

Figure 2.5.1 shows the foundation deployment pipeline for example.com. Terraform code that defines the example.com infrastructure is stored in an on-premises Git repository. Code changes to the main branch of the repository trigger a webhook that in turn triggers a deployment of the Terraform code into the example.com organization.

The Jenkins pipeline is implemented using a horizontal distributed-build architecture. In this model, a Jenkins manager (master) handles HTTP requests and manages the build environment, while the execution of builds is delegated to the Jenkins agents in the `CICD` project.

The Jenkins manager is hosted on-premises, co-located with the source-control management system, while the Jenkins agents are hosted in the Google Cloud environment. The Jenkins agent is a simple SSH server with Java and Docker installed on it—it needs to have access only to the manager's SSH public key. The Jenkins manager in turn acts as an SSH client. The manager uses the SSH private key to connect to the agent and deploys a Java executable JAR file that allows it to instruct the agent to execute the pipeline. The pipeline creates temporary containerized Terraform workers that deploy and modify the example.com infrastructure.

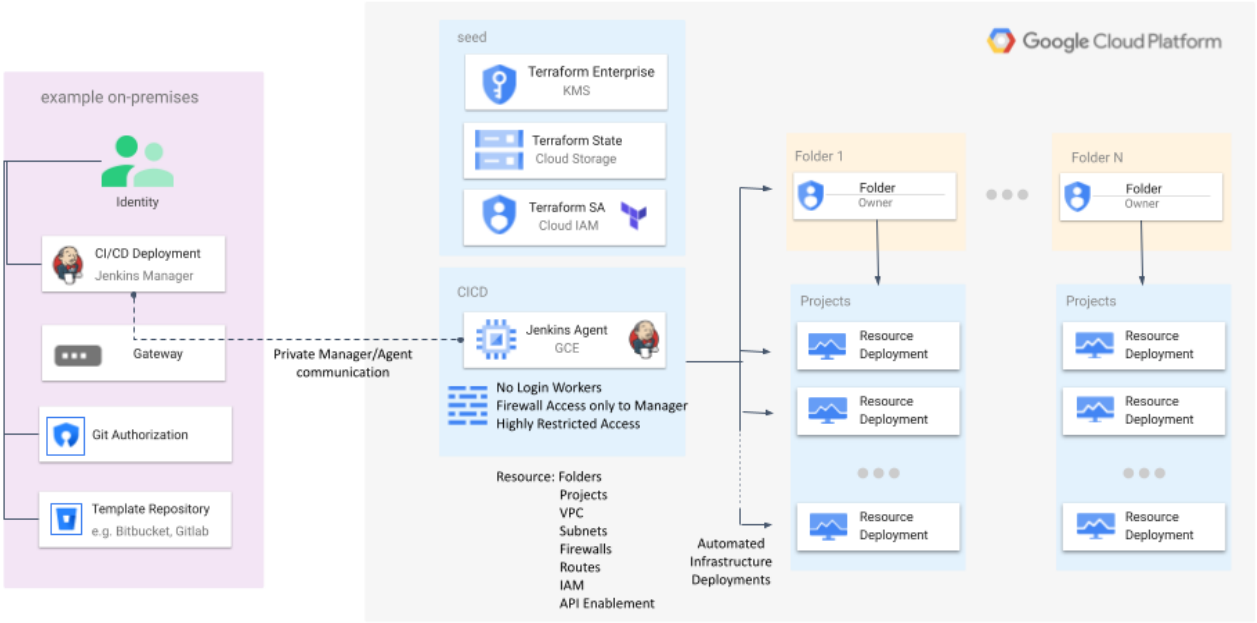

Figure 2.5.1 The example.com foundation deployment pipeline

Table 2.5.1 details the security and governance controls that are integrated with the pipeline.

Control	Description
Pull request (PR)	Code that's merged with the main branch needs to have an approved PR.
Policy checks	Policy checks are enforced by the pipeline against the Terraform code using Terraform Validator.
Deployment approval	An optional manual approval stage is included in the pipeline for deploying code.

Table 2.5.1 The example.com foundation pipeline security controls

Using this architecture in Google Cloud allows a clean authentication and authorization process for Google Cloud APIs. The Jenkins agent uses a custom service account that doesn't have permission to create new infrastructure and that instead impersonates the Terraform service account to deploy infrastructure. As you can see in Table 2.5.2, the Terraform service account in turn has the administrative roles that are needed in order to create organization policies, folders, projects, and other foundation components. Because these service accounts include very sensitive permissions, access to the `CICD` and `seed` projects where these agents run is highly restricted.

Role title	Role
Access Context Manager Admin	`roles/accesscontextmanager.policyAdmin`
Billing Account User	`roles/billing.user`
Compute Network Admin	`roles/compute.networkAdmin`
Compute Shared VPC Admin	`roles/compute.xpnAdmin`
Folder Admin	`roles/resourcemanager.folderAdmin`
Logs Configuration Writer	`roles/logging.configurationWriter`
Organization Administrator	`roles/resourcesmanager.organizationAdmin`
Organization Policy Administrator	`roles/orgpolicy.policyAdmin`
Organization Viewer	`roles/resourcemanager.organizationViewer`
Security Center Notification Configurations Editor	`roles/securitycenter.notificationConfigEditor`
Service Account Admin	`roles/iam.serviceAccountAdmin`
Security Admin	`roles/iam.securityAdmin`

Table 2.5.2 Service account permissions for the foundation deployment pipeline

Running the agents in Compute Engine forgoes the requirement to download a service account's JSON key, which would be required if these agents ran on-premises or in any other environment outside of Google Cloud. Within the example.com organization, only the Jenkins agents and select firecall accounts have permissions to deploy foundation components.

Note: Policy should not be hard-coded directly into Terraform modules or hard-coded directly into deployment pipelines. Policy should be handled by services like the Terraform Validator or Open Policy Agent.

5.3) Project deployment

Projects in the example.com organization are deployed through the deployment pipeline. This section describes project attributes that are assigned when the project is created.

5.3.1) Project labels

Project labels are key-value pairs that are included with the billing export into Cloud Monitoring, enabling enhanced analysis on billing and resource usage. Table 2.5.3 details the project metadata that's added to each project in the example.com deployment.

Label	Description
`business-code`	A 4-character code that describes which business unit owns the project. The code `abcd` is used for projects that are not explicitly tied to a business unit.
`billing-code`	A code that's used to provide chargeback information.
`primary-contact`	The primary email contact for the project.
`secondary-contact`	The secondary email contact for the project.
`environment`	A value that identifies the type of environment, such as `nonprod` or `prod`.

Table 2.5.3 Project labels for example.com

5.3.2) IAM permissions

IAM permissions are defined and created on a per-project basis as part of project deployment.

5.3.3) Google Cloud APIs

Google Cloud APIs are enabled on a per-project basis, and the pipeline has policy checks in place to ensure that only approved APIs can be enabled using an allow/deny list.

5.3.4) Billing account

New projects are linked to the primary billing account. Chargeback to the appropriate business unit is enabled through the use of project labels, as described in Section 11.2.

5.3.5) Networking

Project networking structures such as VPC networks, subnets, firewalls, and routes are enabled through the deployment pipeline.

5.3.6) Project editor

There are two project editors associated with a project. One is the custom service account that's used by the deployment pipeline in the seed project. The other project editor is a firecall account that can be used if automation breaks down or in an emergency, as described in Section 6.3.

5.4) Repository structure

The example.com code repository is distributed as a combined single repository to make it easy for you to fork, copy, and use. However, the code has been constructed in such a way that each step in the code is executed through a separate Jenkins job and repository. The top-level folders and the contents of each folder are shown in Table 2.5.4. You can find the code for example.com in the `terraform-example-foundation` GitHub repository.

Folder	Description	example.com components

`0-bootstrap`	This is where initial projects and IAM permissions are deployed for subsequent IaC stages (1-4).	`bootstrap` folder
		• `seed` project
		• `CICD` project
		  ○ Jenkins pipeline
		  ○ Terraform Validator

`1-org`	This is for organization-wide concerns such as policy, log exports, IAM, and so on.	organization policy organization-wide IAM settings `common` folder • base_network_hub • billing export project • log export project • interconnect project • org-wide secrets project • DNS project • restricted_network_hub • SCC notifications project
`2-environments`	This is for modular creation of new top-level environments, including required projects and	`dev` folder • base Shared VPC host projects • restricted Shared VPC host projects

	the top-level folder.	• environment secrets projects • environment monitoring project nonprod folder • (same projects as dev folder) prod folder • (same projects as dev folder)
3-networks	This is for modular creation and management of VPC networks.	VPC networks firewall rules Cloud Routers routes
4-projects	This is for creation of projects for different teams or business units, with an application workload focus.	application projects

Table 2.5.4 The example.com repository structure

5.5) Foundation creation and branching strategy

To build out the example.com foundation, you start by creating a fork of the example.com repository and then create separate repositories for each of the folders in the example.com repository. Once you've created separate repositories, you manually deploy the code that's in the `0-bootstrap` repository. The code in this repository creates the initial projects and the foundation deployment pipeline. After the code from the `0-bootstrap` folder has run, the code in the `1-org` folder runs by using the foundation deployment pipeline to create organizational-wide settings and to create the common folder.

After the code has been deployed in the `0-bootstrap` and `1-org` folders, the remainder of the foundation is built by deploying the code from the `2-environments`, `3-networks`, and `4-projects` folders. The code uses a persistent branch strategy to deploy code through the foundation deployment pipeline to the appropriate environment.

As shown in Figure 2.5.2, the example.com organization uses three branches (`development`, `non-production`, and `production`) that reflect the corresponding environments. You should protect each branch through a PR process. Development happens on feature branches that branch off `development`.

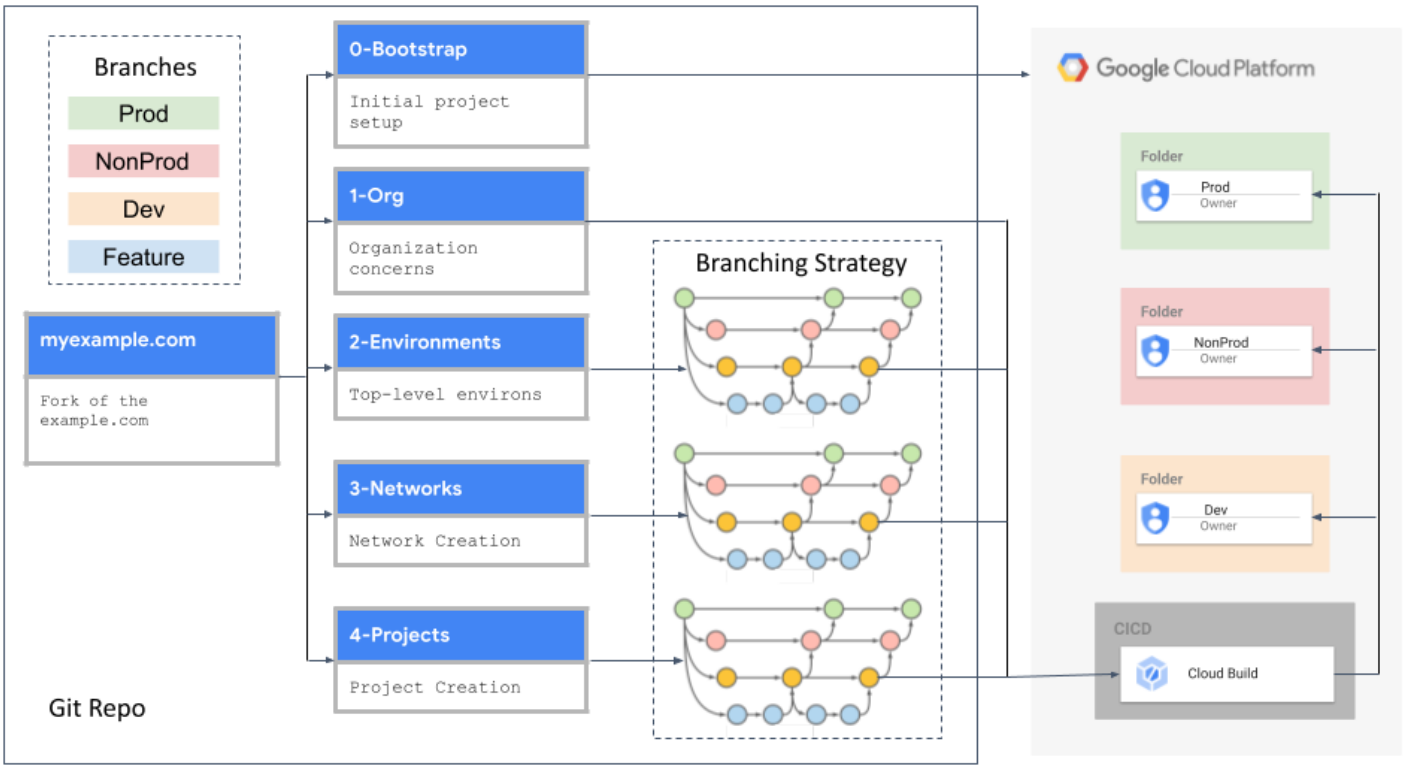

Figure 2.5.2 The example.com deployment branching strategy

When a feature or bug fix is complete, you can open a PR that targets the `development` branch. Submitting the PR triggers the foundation pipeline to perform a plan and to validate the changes against all environments. After you've validated the changes to the code, you can merge the feature or bug fix into the `development` branch. The merge process triggers the foundation pipeline to apply the latest changes in the `development` branch on the `development` environment. After the changes have been validated in the `development` environment, changes can be promoted to `non-production` by opening a PR that targets the `non-production` branch and merging those changes. Similarly, changes can be promoted from `non-production` to `production`.

5.6) The foundation pipeline and workloads

The pipeline architecture that's laid out in Section 5.1 through Section 5.4 deploys the foundation layer of the Google Cloud organization. You should not use the pipeline for deploying higher-level services or applications. Furthermore, as shown in Figure 2.5.3, the access pattern to Google Cloud through deployment pipelines is only one potential access pattern. You might need to evaluate the access pattern and the controls on the workload for each workload individually.

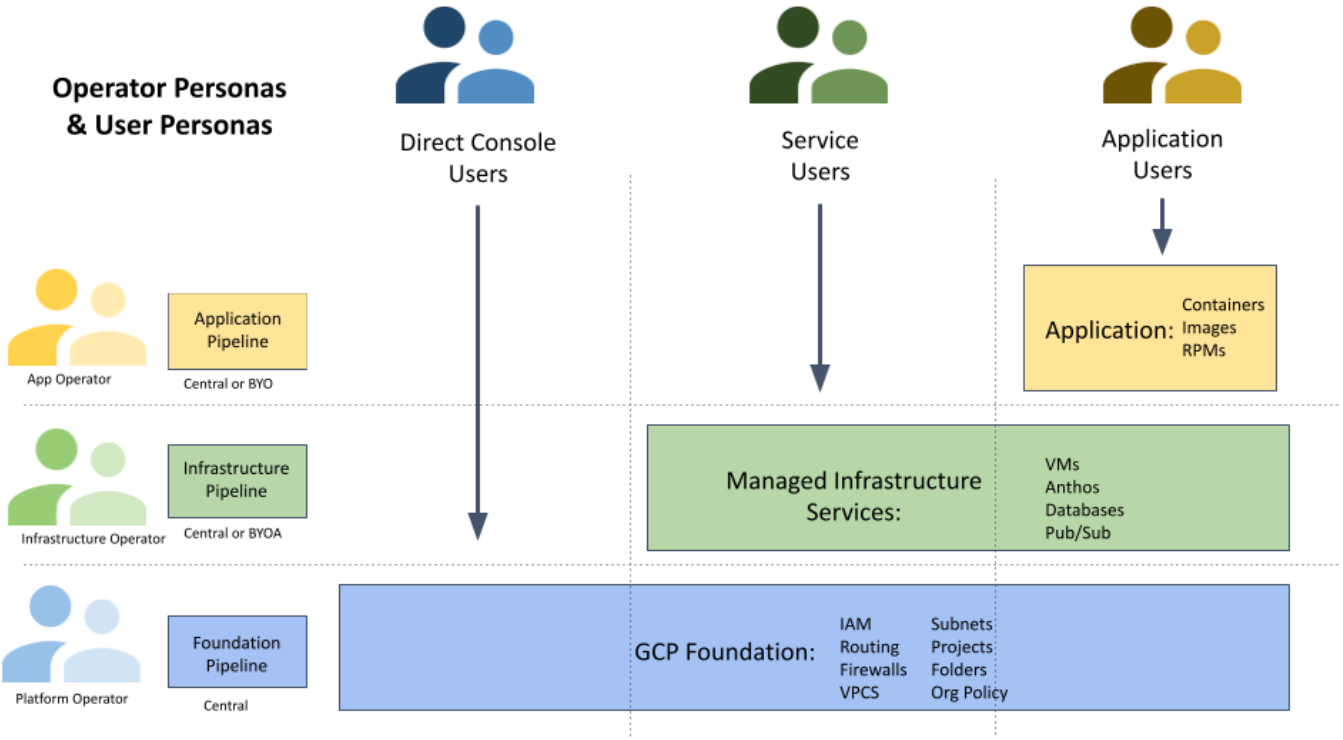

Figure 2.5.3 Access patterns for example.com

The example.com foundation provides you with an infrastructure pipeline, detailed in Section 5.7, that you can use to deploy infrastructure components such as a Cloud SQL instance or a Google Kubernetes Engine (GKE) cluster. The secured foundation also includes an example application pipeline, described in Section 5.8, that you can use to deploy containers to GKE clusters. The application pipeline is maintained in a separate repository from the secured foundation.

5.7) The infrastructure pipeline

The infrastructure pipeline that comes with the example.com foundation builds off the Cloud Build-based code of the foundation pipeline. The infrastructure pipeline manages the lifecycle of the infrastructure components independently of the foundation components. The service account that's associated with the infrastructure pipeline has a more limited set of permissions compared to the service account that's associated with the foundation pipeline.

When the foundation pipeline creates a project, it creates a service account that has a controlled set of permissions. The service account that's associated with the infrastructure pipeline is allowed to impersonate the project service account and to perform only those actions that are permitted to the project service account. This strategy allows you to create a clear separation of duties between the people who deploy foundation components and those who deploy infrastructure components.

The infrastructure pipeline is created by the foundation pipeline and deployed in the `prj-bu1-c-infra-pipeline` and `prj-bu2-c-infra-pipeline` projects in the `common` folder, rather

than in the `CICD` project in the `seed` folder where the foundation pipeline is deployed. To deploy infrastructure components, you create a separate repository to define those components, such as `5-infrastructure`. You can then deploy components to your various environments using the same branching strategy that the foundation pipeline uses.

If your organization has multiple business units and you want each business unit to be able to deploy infrastructure components independently, as seen in Figure 2.5.4, you can create multiple infrastructure pipelines and repositories. Each of the infrastructure pipelines can have a service account with permission to impersonate only the service accounts of the projects that are associated with that business unit.

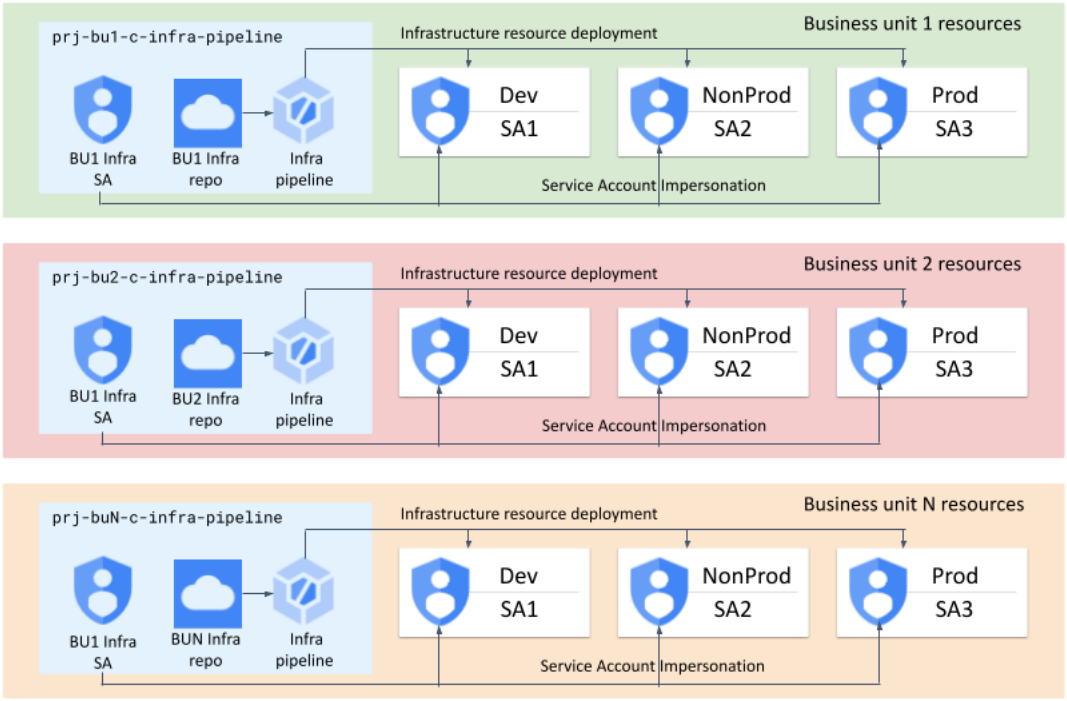

Figure 2.5.4 Multiple infrastructure pipelines

5.8) The application pipeline

The application pipeline that's used with the secured foundation is a Cloud Build-based implementation of the secured shift-left blueprint pipeline that enables you to deploy applications to a Kubernetes cluster. The application pipeline consists of a continuous integration (CI) pipeline and a continuous delivery (CD) pipeline, as shown in Figure 2.5.5.

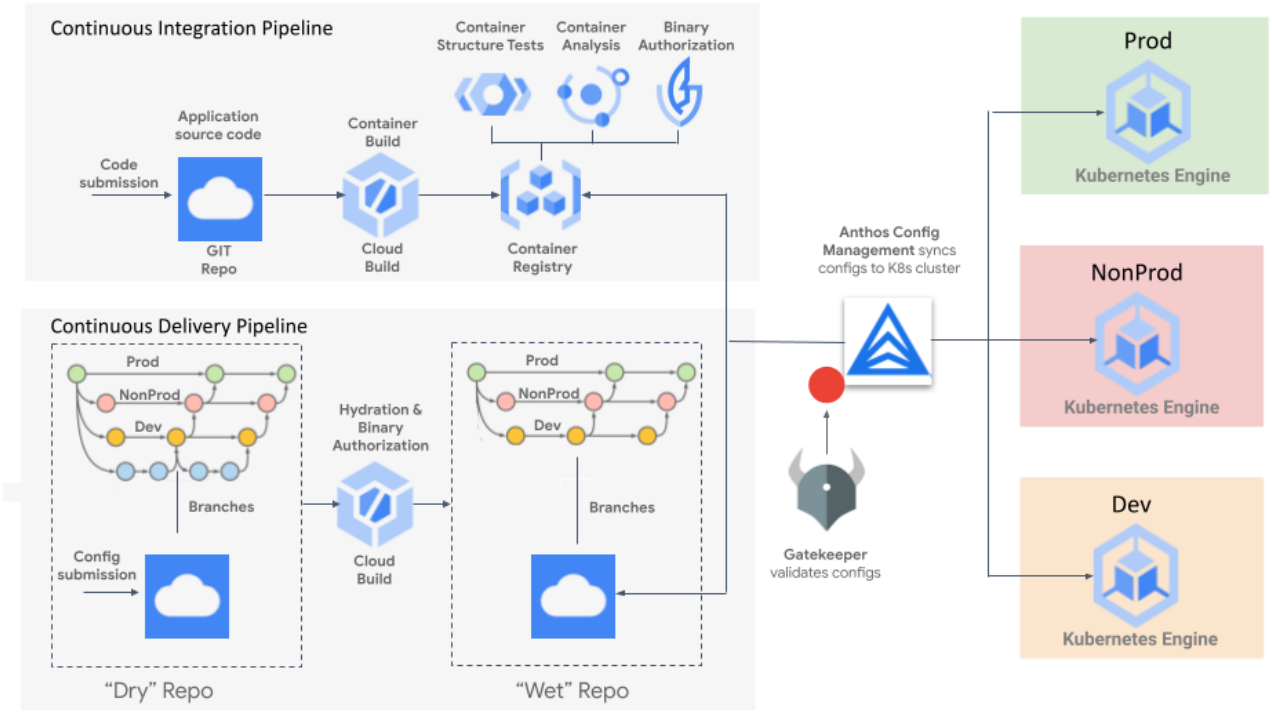

Figure 2.5.5 Application deployment pipeline

The application pipeline uses immutable container images across your environments. This means that the same image is deployed across all environments and will not be modified while it's running. If you must update the application code or apply a patch, you build a new image and redeploy it. The use of immutable container images requires you to externalize your container configuration so that configuration information is read during runtime.

5.8.1) Continuous integration

The application CI pipeline starts when you commit your application code to the `release` branch; this operation triggers the Cloud Build pipeline. Cloud Build creates a container image, and then Cloud Build pushes the container image to Container Registry, creating an image digest. When you build your application, you should follow best practices for building containers.

After the container has been pushed to the Container Registry, the container is analyzed using the Container Structure Test framework. This framework performs command tests, file existence tests, file content tests, and metadata tests. The container image then goes through vulnerability scanning to identify vulnerabilities against a vulnerability database that's maintained by Google Cloud. To help avoid compromised artifacts being introduced, IAM policies are used so that only the Cloud Build service account can contribute to the repository.

After the container has successfully gone through vulnerability scanning, the application pipeline uses Binary Authorization to sign an image. Binary Authorization is a service on Google Cloud that provides software supply-chain security for container-based applications by using policies, rules, notes,

attestations, attestors, and signers. At deployment time, the Binary Authorization policy enforcer ensures the provenance of the container before allowing the container to deploy.

5.8.2) Continuous delivery

The application CD pipeline uses Cloud Build and Anthos Config Management to enable you to deploy container images to your `development`, `non-production`, and `production` environments. Deployments to your environments are controlled through a persistent branching strategy. To start your deployment, you submit a Kubernetes manifest into a "dry" repository on the `development` branch. The manifest contains the image digest of the container or containers that you want to deploy. The initial submission into the dry repository triggers Cloud Build to place the manifest into the "wet" repository.

The CD pipeline uses Anthos Config Management to monitor the wet repository and to deploy the container or containers that are specified in the manifest file or files to the GKE cluster that corresponds to the Git branch. Anthos Config Management synchronizes the states of your clusters with your Git repository using Config Sync. If Config Sync Operator fails to apply changes to a resource, the resource is left in the last known good state.

After you validate the container image in the development environment, you can promote changes to the `non-production` environment by opening a PR that targets the `non-production` branch and then merging the changes. Changes can be promoted from the `non-production` branch to the `production` branch in a similar way. In each promotion, the CD pipeline uses the Binary Authorization framework to create a new signed attestation to ensure that the container image has the appropriate providence. If you need to roll back, you can revert to the last known good commit.

6. Authentication and authorization

Google Cloud includes a number of services and features to support authentication and authorization. These allow for federation with existing identity providers, fine-grained access control, and security controls to manage privileged identities.

6.1) Cloud Identity, directory provisioning, and single sign-on

Google Cloud uses Google Cloud Identity for authentication and access management. Manually maintaining Google identities for each employee can add unnecessary management overhead when all employees already have an account in an existing identity store such as Active Directory. By federating user identities between Cloud Identity and Active Directory, you can automate the maintenance of Google identities and tie their lifecycle to existing identity management processes within your organization.

In the reference architecture, as shown in Figure 2.6.1, relevant users and groups are synchronized periodically from Active Directory to Cloud Identity using the Google Cloud Directory Sync tool. This process ensures that when you create a new user in Active Directory, that user can be referenced in Google Cloud. This process also ensures that if a user account is deleted or suspended, those changes are propagated. Provisioning works one way, which means changes in Active Directory are replicated to Cloud Identity but not the other way around. The sync process doesn't include passwords by default; in a federated setup, Active Directory remains the only system that manages these credentials.

> **Note**: During initial setup, the Directory Sync process must be authenticated interactively using a 3-legged OAuth workflow. We strongly recommend that you create a dedicated account for this sync, as opposed to using an employee's admin account, because the sync will fail if the account performing the sync no longer exists or doesn't have proper privileges. The sync account requires the groups admin and user management Admin administrator roles within the Cloud Identity tenant, and you should protect access to the account as you would to any other highly privileged account. For additional best practices for using Directory Sync, see the documentation.

Single sign-on (SSO) is used in the example.com reference architecture for user authentication. During sign on, Google Cloud delegates authentication to Active Directory Federation Services by using the Security Assertion Markup Language (SAML) protocol. This delegation ensures that only Active Directory manages user credentials and that any applicable policies or multi-factor authentication (MFA) mechanisms are being enforced. For a sign-on to succeed, the user must be provisioned in both Active Directory and Cloud Identity.

In the example.com reference architecture, the on-premises Active Directory system is used as the source of truth for users, groups, and SSO. Google Cloud Directory Sync is installed on a domain-joined host on-premises (either Windows or Linux). A scheduled job (using Task Scheduler or `cron`) is used to synchronize users and groups into Cloud Identity on an hourly basis. To control which users are synced

to Cloud Identity, an attribute is added to the users in Active Directory, and Directory Sync filters on that attribute. In the example.com domain, Google Cloud-specific groups are prefixed with `grp-gcp-` (for example, `grp-gcp-billing-viewer` or `grp-gcp-security-reviewer`) and a filter is applied to sync only those groups. The example.com Directory Sync filters are shown in Table 2.6.1.

Filter type	Filter rule
Group filter	`(&(objectCategory=group)(cn=grp-gcp-*))`
User filter	`(&(objectCategory=user)(msDS-cloudExtensionAttribute1=googlecloud))`

Table 2.6.1 Google Cloud Directory Sync user and group filters

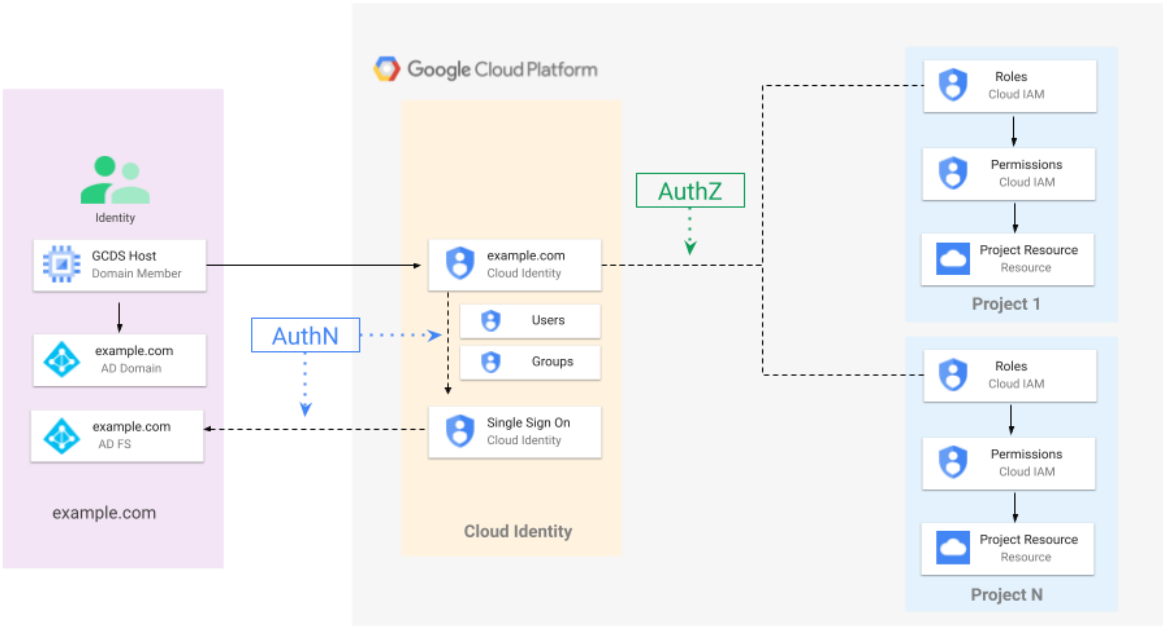

Figure 2.6.1 The example.com identity structure

Note: Groups that are synced using Directory Sync need a valid `mail` attribute that has a suffix that matches the Cloud Identity organization name (for example, `group@example.com`). The `mail` attribute doesn't have to be a valid distribution group for receiving email. But the full email address is what Cloud Identity uses as the unique identifier for the group, and it's how you reference the group when you assign IAM roles.

6.2) Users and groups

Groups for example.com follow the naming convention defined earlier in Table 2.3.2. In the example.com organization, users are not assigned roles directly; instead, groups are the primary method of assigning roles and permissions in Cloud IAM. IAM roles and permissions are assigned during project creation through the deployment pipeline and are then updated as needed.

Users are assigned group membership through the on-premises Active Directory system; as such, group creation and membership is strictly controlled. Table 2.6.2 shows the groups setup in the example.com foundation. You can set up additional groups as needed on a workload-by-workload basis.

Group	Description	Roles	Scope
`grp-gcp-billing-viewer@example.com`	Members are authorized to view the spend on projects. Typical members are part of the finance team.	Billing Account Viewer BigQuery Data Viewer BigQuery User	organization for Billing Account Viewer billing project for BigQuery Data Viewer and BigQuery User
`grp-gcp-platform-viewer@example.com`	Members have the ability to view resource information across the Google Cloud organization.	Viewer	organization
`grp-gcp-security-reviewer@example.com`	Members are part of the security team responsible for reviewing cloud security.	Security Reviewer	organization
`grp-gcp-network-viewer@example.com`	Members are part of the networking team and review network configurations.	Compute Network Viewer	organization
`grp-gcp-audit-viewer@example.com`	Members are part of an audit team and view audit logs in the logging project.	Logs Viewer Private Logs Viewer Bigquery Data Viewer	logging project
`grp-gcp-scc-admin@example.com`	Members can administer Security Command Center.	Security Center Admin Editor	SCC project
`grp-gcp-secrets-admin@example.com`	Members are responsible for putting secrets into Secrets Manager.	Secret Manager Admin	global secrets project production secrets project `non-production` project `development` project
`grp-gcp-`*`businesscode-environment-code`*`-developer@example.com`	Groups are created on a per-business code or per-environment basis to manage resources within a particular project.	service- and project-specific permissions	specified project
`grp-gcp-platform-operator@example.com`	Members are responsible for platform operations, and they are added into other groups and inherit those permissions.		

Table 2.6.2 Groups in example.com

There are three types of roles in Cloud IAM:

- Basic roles include the Owner, Editor, and Viewer roles.
- Predefined roles provide granular access for specific services and are managed by Google Cloud.
- Custom roles provide granular access according to a user-specified list of permissions.

Google recommends using predefined roles as much as possible. The IAM recommender should be used to ensure that IAM permissions that are granted to users and groups are not overly permissive. You should eliminate or at least severely limit the use of basic roles in your Google Cloud organization because the wide scope of permissions inherent in these roles goes against the principles of least privilege. Whenever possible, you should instead use predefined roles, which are designed with inherent separation of duties, and then add custom roles as needed.

Note: Groups and users should not have any permissions to alter the foundation components laid out by the deployment pipeline described in Section 5 unless they are one of the privileged identities.

6.3) Privileged identities

Privileged identities are accessed through a firecall process, which involves users that are used only in situations that require escalated privileges. Examples of this scenario are when an automated process has broken down, or when permissions need to be elevated temporarily to manually restore a service to a functional state. Table 2.6.3 lists the privileged identities used in example.com that should be accessed through a firecall process.

Identity	Description	Role
`gcp-superadmin@example.com`	Identity that has Google Cloud super administrator permissions.	Super Administrator
`gcp-orgadmin@example.com`	Identity that has organization administrator permissions within example.com.	Organization Administrator
`gcp-billingcreator@example.com`	Identity that can create billing accounts.	Billing Account Creator
`gcp-billingadmin@example.com`	Identity that has billing administrator permissions within example.com.	Billing Account Administrator
`gcp-`*`environment`*`-folderadmin@example.com`	Identity (one per folder) that has folder administrator permissions.	Folder Administrator
`gcp-`*`businesscode-environment-code`*`-editor@example.com`	Identity (one per business code or per environment) that has project editor permissions for that group's projects in the environment.	Editor

Table 2.6.3 Privileged identities in example.com

Because of the high-level permissions available to privilege identities, access to the privilege identities should require the consent of at least two people within your organization. For example, one person controls access to a privileged identity password, while another person controls access to the associated MFA token.

Note: Cloud Identity users with the super admin role, such as `gcp-superadmin@example.com`, bypass the organization's SSO settings and authenticate directly to Cloud Identity. This is to provide access to the Google Admin console in the event of an SSO misconfiguration or outage. For details, see the Super administrator account best practices documentation and the Security best practices for administrator accounts support article.

7. Networking

Networking is fundamental to creation of an organization within Google Cloud. This section describes the structure in the example.com reference architecture for VPC networks, projects, IP address space, DNS, firewalls, and connectivity to the example.com on-premises environment.

7.1) Shared VPC

Shared VPC is a networking construct that significantly reduces the amount of complexity in network design. With Shared VPC, network policy and control for all networking resources are centralized and easier to manage. Service project departments can configure and manage non-network resources, enabling a clear separation of responsibilities for different teams in the organization.

Resources in Shared VPC networks can communicate with each other securely and efficiently across project boundaries using internal IP addresses. You can manage shared network resources—such as subnets, routes, and firewalls—from a central host project, so you can enforce consistent network policies across the projects.

As shown in Figure 2.7.1, the example.com architecture uses two Shared VPC networks, base and restricted, as the default networking construct for each environment. Each Shared VPC network is contained within a single project. The base VPC network is used for deploying services that contain non-sensitive data, and the restricted VPC network uses VPC Service Controls to limit access to services that contain sensitive data. Communication to the restricted VPC network is controlled by Access Context Manager, which has an access policy that allows IP address flows based on access levels. By creating an access level, you can also use a base VPC network for hosting services that require access to the sensitive data that's stored in the restricted VPC network.

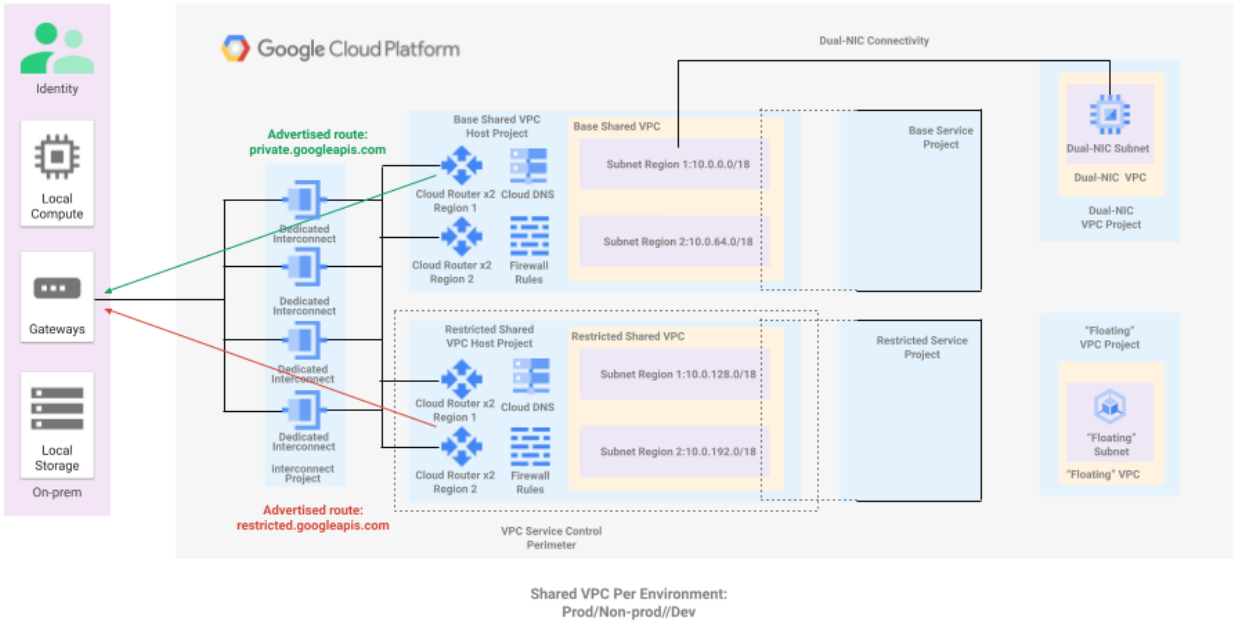

Figure 2.7.1 The example.com VPC network structure

Each Shared VPC network has two subnets, with each subnet in a distinct region that spans multiple zones. This infrastructure provides a mechanism for high availability and for disaster-recovery workload configurations. As described in Section 7.3, connectivity with on-premises resources is enabled through four connections for each Shared VPC, using four Cloud Routers (two per region). Table 2.7.2, Table 2.7.3, and Table 2.7.4 list the IP address ranges of each subnet in the example.com design.

Note: The example.com network architecture represents one possible architecture for your VPC network design. An alternative design is presented in the next section, but you might also want to consider an alternative architecture for your VPC design.

7.1.1) Project deployment patterns

Within the reference example.com organization, there are four patterns for deploying projects:

- **Service projects** are attached to a Shared VPC host project. Service projects are used by example.com for projects that require access to on-premises resources and that need to share resources with other service projects.

- **Floating projects** have no private network connectivity to the example.com on-premises environment; they require access solely to Google Cloud resources. You can modify the floating projects pattern to configure a Shared VPC host project that has no connectivity to on-premises and then have service projects that have no on-premises connectivity.

- **Peered VPC projects** use VPC Network Peering to enable you to connect VPC networks so that workloads in different VPC networks can communicate internally. Traffic stays within Google's network and doesn't traverse the public internet; this allows you to make services available across VPC networks by using internal IP addresses. Peered VPC networks can't have a subnet CIDR range in one peered VPC network that overlaps with a static route in another peered VPC network. Only directly peered networks can communicate, because transitive peering is not supported.

- **Dual-NIC connected projects** are connected to either the base VPC network or to the restricted VPC network through a dual-NIC enabled Compute Engine instance. The Compute Engine instance acts as a NAT, allowing resources in the dual-NIC VPC network to communicate to the on-premises environment through the Shared VPC network or with resources within the Shared VPC network. You use dual-NIC connected projects when you need connectivity either to on-premises or to the Shared VPC network, but the IP address space requirements of the dual-NIC-enabled VPC network is large.

Note: For your workloads that require egress traffic to the internet, the example.com repository contains code that allows you to optionally deploy Cloud NAT instances. For workloads that require ingress traffic from the internet, you should consider using Cloud Load Balancing.

7.2) Hub-and-spoke

You can implement the model described in the preceding section independently for each of the four environments (`common`, `development`, `non-production`, and `production`). This model provides the highest level of network segmentation between environments, where the Shared VPC networks from each environment connect to on-premises networks using either Cloud Interconnect or Cloud VPN HA. If cross-environment network traffic is expected, the traffic needs to flow through the on-premises network, which can incur extra operational effort and extra costs.

As an evolution of the independent Shared VPC model described in Section 7.1, the example.com architecture provides you with a hub-and-spoke reference network model. In this model, you connect the `development`, `non-production`, and `production` environments (spokes) to the `common` environment (hub), and you use the Shared VPCs in the `common` environment as the transit point to the on-premises network.

For this scenario, all spoke environments can directly communicate with shared resources in the `common` environment hub, and can also use this path to share connectivity to on-premises networks. The `common` environment can host tooling that requires connectivity to all other environments, like CI/CD infrastructure, directories, and security and configuration management tools. As in the previous independent Shared VPC model, the hub-and-spoke scenario is also composed of base and restricted VPC networks. A base Shared VPC hub connects the base Shared VPCs spokes in `development`, `non-production`, and `production`, while the restricted Shared VPC hub connects the restricted Shared VPCs spokes in these same environments. The choice between base and restricted Shared VPCs here also depends on whether VPC Service Controls are required. For workloads with strong data exfiltration mitigation requirements, the hub-and-spoke associated to the restricted Shared VPCs is preferred.

Figure 2.7.2 shows the organization of the Shared VPCs across environments for the hub-and-spoke model. As the diagram shows, connectivity to on-premises terminates only on the base and restricted Shared VPCs that are associated to the common environment. From there, networking is extended to the three Shared VPCs in `development`, `non-production`, and `production` using VPC peering. The routes to the on-premises networks are learned by the Cloud Routers on the hub VPCs. You can export these routes to the spokes to allow workloads on those Shared VPCs to directly communicate to the on-premises networks.

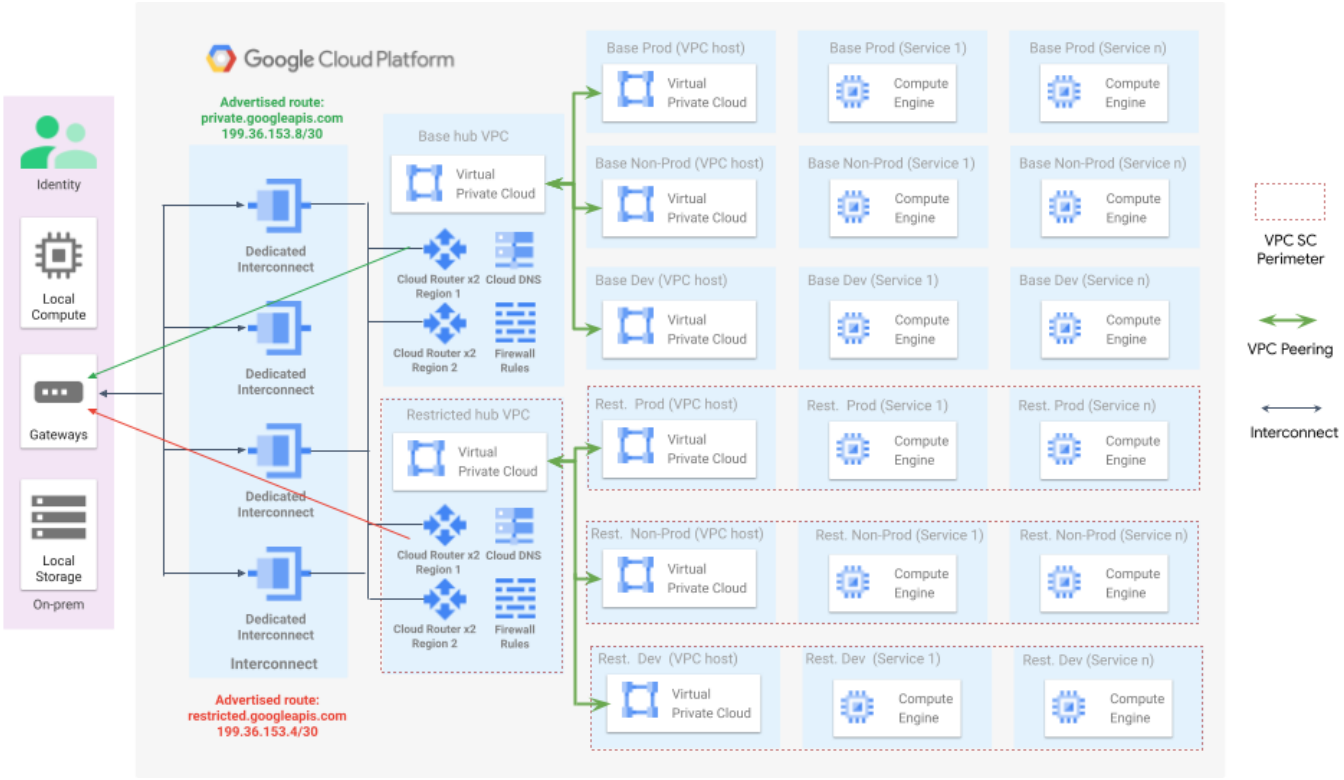

Figure 2.7.2 The example.com VPC network structure when using hub-and-spoke connectivity based on VPC peering

When you use VPC peering to attach spokes to the hub VPC, however, spokes can't directly communicate with each other because VPC peering isn't transitive. This also impacts connectivity from the hub VPC into managed Google services like GKE, Cloud SQL, and Apigee X, because private connectivity to components of these services is also implemented using VPC peering. The lack of transitivity for VPC peering can bring additional complexity—for example, to overcome this limitation, you might need to deploy VMs as gateways or as proxies, as described in the following section.

7.2.1) Hub-and-spoke transitivity

As noted, VPC peering isn't transitive, hence Shared VPC spokes can't directly communicate with each other. This gap can be considered a feature because it can provide full network separation between the `development`, `non-production`, and `production` environments. Realistically, there are frequent use cases that require some sort of connectivity across environments. To overcome the gap, you can deploy network virtual appliances (NVAs) on the hub Shared VPC that act as gateways for the spoke-to-spoke traffic. You can deploy these NVAs behind internal load balancers that are used as a next hop. These load balancers can provide high availability and scalability for the spoke-to-spoke traffic.

The example.com reference architecture includes NVAs on both regions. This approach provides regional independence with respect to failures. It also enforces that intra-region traffic is handled by an NVA that's within the same region as the destination, forcing packets to take the shortest route and helping to keep latency low. Traffic is steered to the NVAs in the region of the destination by VPC routes

that cover the regional aggregate IP address ranges of the target region. These routes are exchanged from hub to spoke VPC networks through custom routes exchange.

In this setup, the NVAs perform source network address translation (SNAT) as they forward packets to the spokes. The translation serves two purposes. First, it ensures that return traffic is directed to the same NVA that handles the forward packet flow, which ensures that the NVAs can handle flows statefully. Second, it simplifies network access controls that are implemented using VPC firewalls on the target spoke by allowing traffic only from the NVA IP address ranges. In this scenario, you implement network access controls between spokes through the firewall functionality of the corresponding NVAs.

Figure 2.7.3 shows two network flows between resources in the difference spokes. The first one is a connection started by an instance in `Region 2` of the `development` environment to another instance (in the same region) in the `production` environment. In this example, the routing at the source matches the route for the `Region 2` aggregate range, and the flow is forwarded across the VPC peering to the internal load balancer that's responsible for distributing traffic to the NVAs for `Region 2`. The load balancer picks one of the NVA backends to handle the traffic, and the NVA forwards the packets to the destination in `production`.

The second flow originates from an instance in `Region 2` of the production spoke, but it's directed to an instance in `Region 1` of the `development` spoke. This time, the routing at the source matches the route for the `Region 1` aggregate, and the traffic is directed to the NVAs of `Region 1` after crossing the internal load balancer. From there, traffic reaches its destination through the VPC peering.

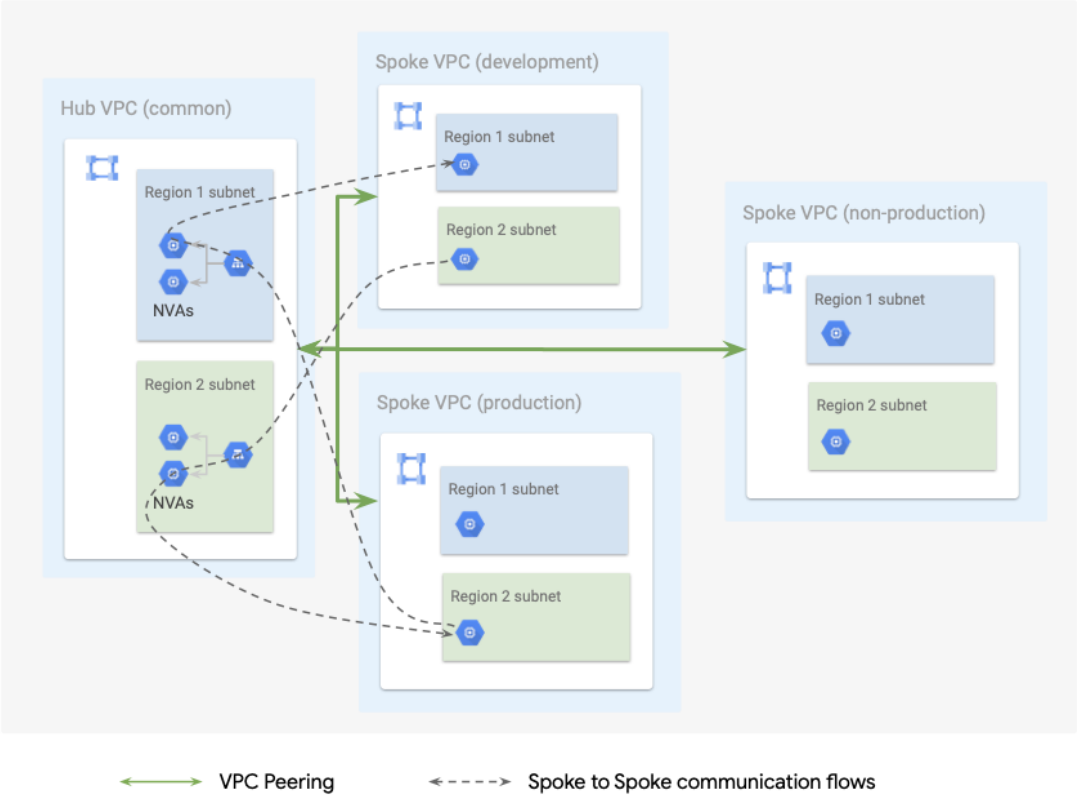

Figure 2.7.3 Spoke-to-spoke connectivity using network virtual appliances (NVAs)

7.3) Enterprise-to-Google Cloud connectivity

To establish connectivity between the on-premises environment and Google Cloud, the example.com reference architecture uses Dedicated Interconnect to maximize security and reliability. A Dedicated Interconnect connection is a direct link between your enterprise's network and Google Cloud.

As shown in Figure 2.7.4, the example.com architecture relies on four Dedicated Interconnect connections in two different metro regions to achieve a 99.99% SLA. The connections are divided into two pairs, with each pair connected to a separate on-premises data center. All connections are hosted in the `Interconnect` project of the organization. VLAN attachments are used to connect each Dedicated Interconnect instance to Cloud Routers that are attached to the Shared VPC structure described in Section 7.1. Each Shared VPC network has four Cloud Routers, two in each region, with the dynamic routing mode set to `global`. This enables every Cloud Router to announce all subnets, independent of region. Figure 2.7.4 also depicts the ASN and IP addresses of the Cloud Routers.

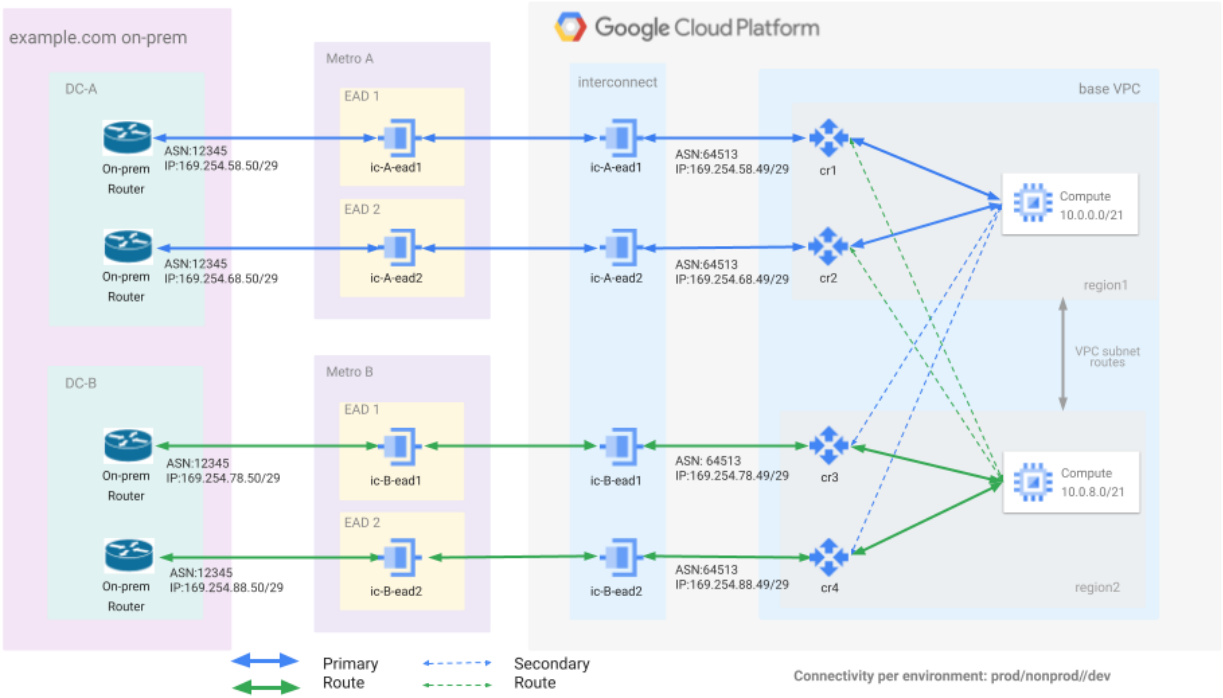

Figure 2.7.4 The example.com Dedicated Interconnect connection structure

When you configure filters on the on-premises routers to specify which routes to accept through Border Gateway Protocol (BGP), add the internal IP address space that you configured in the previous step. Also accept `199.36.153.8/30` for private Google access and `199.36.153.4/30` for restricted private Google access. For inbound DNS forwarding queries from Google Cloud, you also need to accept `35.199.192.0/19`, as discussed in Section 7.5. Although all these ranges are public IP addresses, they are not routed over the public internet.

Table 2.7.1 shows the mapping of Cloud Routers, VLANs, VPC networks, connections, regions, and points of presence (POPs) for the production environment.

When global routing is enabled and advertised route priority values are left at their default, every Cloud Router announces all subnet routes within its VPC network using BGP. However, routes from remote regions are assigned a higher advertised route-priority value. This means that on-premises routing tables prefer to send traffic over the connection that's closer to the compute region that's being used. In the reverse direction, to keep things symmetrical, Google Cloud penalizes routes that are learned if the routes originate from remote regions. Between two Cloud Routers in the same region, BGP path selection is left to the configuration of your on-premises routers and BGP sessions.

Traffic flowing in the other direction—from Google Cloud to your on-premises network—also egresses through the connection and through the Cloud Router that's closest to compute resources. Within a single region, there are multiple routes available to on-premises networks that have the same MED value. In those cases, Google Cloud uses ECMP to distribute egress traffic between all possible routes, keeping an affinity based on the hash of a 5-tuple.

Router name	POP-EAD	IC	VLAN attachment	VPC	Region
cr-p-shared-base-region1-cr1	metroA-zone1	ic-a-ead1	vl-ic-a-ead1-p-shared-base-region1-cr1	base	region1
cr-p-shared-base-region1-cr2	metroA-zone2	ic-a-ead2	vl-ic-a-ead2-p-shared-base-region1-cr2	base	region1
cr-p-shared-base-region2-cr3	metroB-zone1	ic-b-ead1	vl-ic-b-ead1-p-shared-base-region2-cr3	base	region2
cr-p-shared-base-region2-cr4	metroB-zone2	ic-b-ead2	vl-ic-b-ead2-p-shared-base-region2-cr4	base	region2
cr-p-shared-restricted-region1-cr5	metroA-zone1	ic-a-ead1	vl-ic-a-ead1-p-shared-restricted-region1-cr5	restricted	region1
cr-p-shared-restricted-region1-cr6	metroA-zone2	ic-a-ead2	vl-ic-a-ead2-p-shared-restricted-region1-cr6	restricted	region1
cr-p-shared-restricted-region2-cr7	metroB-zone1	ic-b-ead1	vl-ic-b-ead1-p-shared-restricted-region2-cr7	restricted	region2
cr-p-shared-restricted-region2-cr8	metroB-zone2	ic-b-ead2	vl-ic-b-ead2-p-restricted-region2-cr8	restricted	region2

Table 2.7.1 The example.com connection topology

Note: This section describes using Dedicated Interconnect to connect Google Cloud to the example.com on-premises data centers. Google Cloud also supports Partner Interconnect and Cloud VPN for private connectivity options. You should consider whether those services meet your connectivity requirements.

7.4) IP address space allocation

To properly lay out a Google Cloud foundation, you must allocate IP address ranges for the Google Cloud environment. Google Cloud supports a variety of valid IP ranges, including both RFC 1918 and non-RFC 1918 spaces. Note that there are reserved and restricted IP ranges that can't be used within a subnet. In the example.com reference architecture, IP ranges are assigned to a subnet during project creation.

Table 2.7.2 provides a breakdown of the IP address space that's allocated for the Shared VPC networks in the example.com reference architecture. The `common` environment corresponds to the hub for the hub-and-spoke model, while the `development`, `non-production`, and `production` environments are either independent Shared VPCs for each environment, or they're spokes for hub-and-spoke. The IP address allocation of the example.com reference architecture assigns a continuous RFC 1918 `/16` IP address range to each region and to each Shared VPC network. Because there are three environments in the example.com architecture (or, four, if the hub Shared VPC is included), each environment is allocated a `/18` address range from within the regional `/16` range for each Shared VPC network.

For each Shared VPC network, six `/18` aggregates are left unallocated for future use in new or existing regions. From the allocated IP address space, you can break down the allocated `/18` range into more specific ranges for different subnets, with each subnet spanning a region. If RFC 1918 IP address space is limited, you can use non-RFC 1918 address spaces, provided that the on-premises networking can support such topologies.

VPC	Region	Environment			
		Common (Hub)	Production	Non-production	Development
Base	Region 1	10.0.0.0/18	10.0.192.0/18	10.0.128.0/18	10.0.64.0/18
	Region 2	10.1.0.0/18	10.1.192.0/18	10.1.128.0/18	10.1.64.0/18
	Unallocated	10.{2-7}.0.0/18	10.{2-7}.192.0/18	10.{2-7}.128.0/18	10.{2-7}.64.0/18
Restricted	Region 1	10.8.0.0/18	10.8.192.0/18	10.8.128.0/18	10.8.64.0/18
	Region 2	10.9.0.0/18	10.9.192.0/18	10.9.128.0/18	10.9.64.0/18
	Unallocated	10.{10-15}.0.0/18	10.{10-15}.192.0/18	10.{10-15}.128.0/18	10.{10-15}.64.0/18

Table 2.7.2 Primary IP address space allocation for Shared VPC networks

Some use cases, such as container-based workloads, can require additional aggregates. These need to be defined as subnet secondary ranges. For these cases, you can use address ranges that are taken from the reserved RFC 6598 address space, as listed in Table 2.7.3. If you use these aggregates, you must make sure that they are announced and that they are allowed to reach on-premises.

VPC	Region	Environment			
		Common (Hub)	Production	Non-production	Development
Base	Region 1	100.64.0.0/18	100.64.192.0/18	100.64.128.0/18	100.64.64.0/18
	Region 2	100.65.0.0/18	100.65.192.0/18	100.65.128.0/18	100.65.64.0/18
	Unallocated	100.{66-71}.0.0/18	100.{66-71}.192.0/18	100.{66-71}.128.0/18	100.{66-71}.64.0/18
Restricted	Region 1	100.72.0.0/18	100.72.192.0/18	100.72.128.0/18	100.72.64.0/18
	Region 2	100.73.0.0/18	100.73.192.0/18	100.73.128.0/18	100.73.64.0/18
	Unallocated	100.{74-79}.0.0/18	100.{74-79}.192.0/18	100.{74-79}.128.0/18	100.{74-79}.64.0/18

Table 2.7.3 Secondary IP address space allocation for Shared VPC networks

For private services access, a `/16` range is reserved globally for each of the Shared VPC networks. This range is used to allocate addresses for private connectivity to managed services such as Cloud SQL. Table 2.7.4 shows the addresses reserved for private service access use.

VPC	Region	Environment		
		Production	Non-production	Development
Base	Global	10.16.192.0/16	10.16.128.0/16	10.16.64.0/16
Restricted	Global	10.24.192.0/16	10.24.128.0/16	10.24.64.0/16

Table 2.7.4 IP address space allocation for private services access (Service Networking)

7.5) DNS setup

Cloud DNS requires the `35.199.192.0/19` address space to be advertised to the on-premises network in order to pass DNS information between Google Cloud and on-premises. To accommodate multiple DNS instances, Cloud DNS has native peering functionality that can be used to create a central hub DNS instance with other DNS instances that are peered to the hub, as shown in Figure 2.7.5. Cloud DNS peering is a logical construct and doesn't require network connectivity between projects.

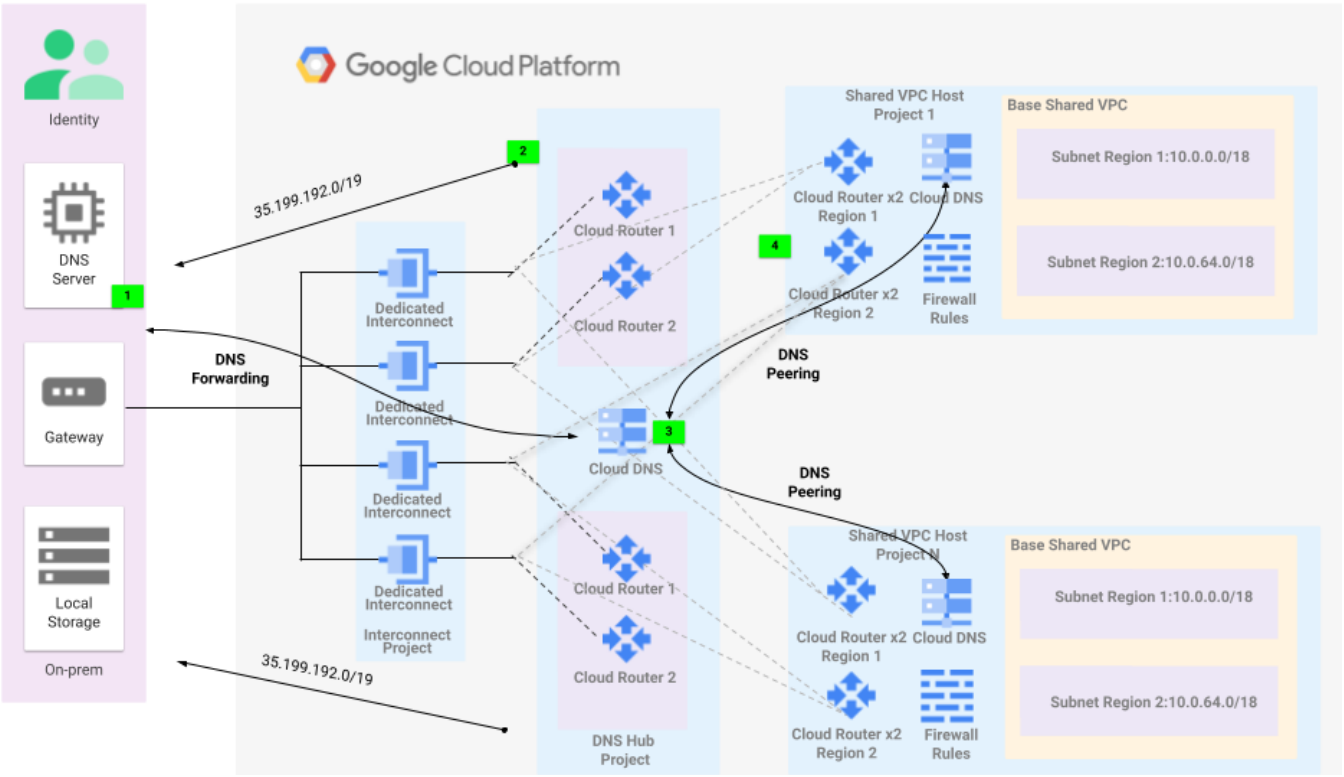

Figure 2.7.5 Cloud DNS setup for example.com

The components highlighted in the diagram include the following:

1. On-premises DNS Server: In the on-premises environment, configure DNS forwarding to the DNS inbound endpoints in the hub VPC.
2. Hub VPC - DNS Egress Proxy: Advertise the Google Cloud DNS egress proxy range `35.199.192.0/19` to the on-premises network.
3. Hub VPC - DNS Forwarding/DNS Peering: Configure DNS forwarding to on-premises in order to resolve on-premises hosts. Configure DNS peering from the hub VPC to each Shared VPC network in order to resolve the zone records in the Shared VPC networks.
4. Shared VPC Cloud DNS - DNS: Shared VPC 1 can resolve zone records in Shared VPC N by taking the DNS peering hop to the hub VPC and then a second DNS peering hop to Shared VPC N.

The DNS hub VPC network in the DNS hub project is used as a central VPC network for DNS configuration. The DNS hub project requires connectivity to the on-premises example.com infrastructure, which is enabled through Dedicated Interconnect VLAN attachments and a BGP session that's established from the hub VPC network to on-premises. These VLAN attachments are separate from the VLAN attachments that are used in the Shared VPC networks for the `production`, `non-production`, and `development` environments. The hub VPC network is not attached to the Shared VPC network on the basic Shared VPC topology. For the hub-and-spoke transitivity topology, the transit Shared VPC host project on the common environment also acts as the DNS hub.

After the DNS hub has been established, the Google Cloud DNS egress proxy range `35.199.192.0/19` needs to be advertised through the Cloud Routers to the on-premises network. In the on-premises environment, you need to configure the enterprise DNS with DNS forwarding to the DNS inbound endpoints in the DNS hub VPC network.

You can configure DNS peering from the DNS hub VPC network to each Shared VPC network so that it resolves the zone records in the Shared VPC networks. Cloud DNS supports transitive DNS peering, but only through a single transitive hop. In other words, no more than three VPC networks can be involved, with the network in the middle as the transitive hop.

For example, consider the architecture in Figure 2.7.5 and consider two arbitrary VPC networks that need to exchange DNS information: `Shared VPC 1` and `Shared VPC 16`. The `Shared VPC 1` network can resolve zone records in `Shared VPC 16` by taking the DNS peering hop to the DNS hub VPC and then taking a second DNS peering hop to `Shared VPC 16`.

7.6) On-premises access to Google Cloud APIs through a private IP address using Dedicated Interconnect

In the example.com architecture, in order to set up private Google Cloud API access from on-premises hosts that have only private IP addresses, Google Cloud provides two distinct `/30` IP address endpoints known as virtual IP addresses (VIPs). These VIPs can be accessed over the Dedicated Interconnect connections.

For Google Cloud APIs that are accessible in the base VPC network, on-premises hosts can access those APIs by resolving `private.googleapis.com` to `199.36.153.8/30`. The `private.googleapis.com` VIP enables access to most Google APIs. Although this is a public IP address, it's not announced over the internet by Google. As shown earlier in Figure 2.7.1, the base VPC network's Cloud Routers announce the `private.googleapis.com` route over the Dedicated Interconnect connection to on-premises hosts.

For Google Cloud APIs that are accessible in the restricted VPC network, on-premises hosts can access those APIs by resolving `restricted.googleapis.com` to `199.36.153.4/30`. The `restricted.googleapis.com` VIP enables access to Google APIs that are accessible in a service perimeter. Although this is a public IP address, it's not announced over the internet by Google. As shown earlier in Figure 2.7.1, the restricted VPC Cloud Routers announce the `restricted.googleapis.com` route over the interconnect connection to on-premises hosts.

For VMs that run in Google Cloud without external IP addresses, private access to Google Cloud APIs is enabled during project creation through Google Private Access.

Note: To support both `private.googleapis.com` and `restricted.googleapis.com`, on-premises DNS redirection is required and must be managed on a workload-by-workload basis, with on-premises workloads directing API calls to either the `private` or `restricted` VIP. In addition, the same DNS indirection and routing needs to be set up in the restricted VPC network so that cloud-based workloads can take full advantage of VPC Service Controls.

7.7) Hierarchical firewall policies and VPC firewall rules

Hierarchical firewall policies let you enforce firewall rules at the organization and folder level in the Google Cloud resource hierarchy. Security administrators at different levels in the hierarchy can define and deploy consistent firewall rules across a number of projects so that they are applied to all VMs in existing and future projects.

This ability gives organization-level security administrators assurance that all VMs in their organization include the most critical rules, such as blocking traffic from specific IP ranges or ensuring that certain traffic can reach all VMs.

Managing the most critical firewall rules in one place also frees project-level administrators (project owners, editors, or security administrators) from having to keep up with changes to organization-wide policies.The goal is to apply a baseline of VPC firewall rules to the entire organization or specific business units by using folder-level firewall policies.

VPC firewall rules can be used for VPC-specific configurations, which ideally represents minimal configuration at the VPC level. An example of this approach is to use VPC firewalls for application-specific requirements and leave security enforcements in the hierarchy.

7.7.1) Hierarchical firewall policies

The organization is the highest level in the hierarchy, therefore any security policies that you add to this level affect the entire organization. The example.com architecture provides you with firewall policies that are defined at the folder level. These policies are then attached to the `production`, `non-production`, `development`, `bootstrap`, and `common` folders. In the example.com architecture, the organization folder is left empty in case a need arises that requires you to set a firewall policy at the topmost level.

The example.com reference architecture includes the following hierarchical firewall policies:

- **Configured health checks from Cloud Load Balancing**. The well-known ranges that are used for health checks are allowed.
    - For most Cloud Load Balancing instances (including Internal TCP/UDP Load Balancing, Internal HTTP(S) Load Balancing, TCP Proxy Load Balancing, SSL Proxy Load Balancing, and HTTP(S) Load Balancing), a security policy is defined that allows ingress traffic from the IP ranges `35.191.0.0/16` and `130.211.0.0/22` for ports `80` and `443`.

    - For Network Load Balancing, a security policy is defined that enables legacy health checks by allowing ingress traffic from IP ranges `35.191.0.0/16`, `209.85.152.0/22`, and `209.85.204.0/22` for ports `80` and `443`.

  For both of these scenarios, the security policy is attached to all folders in the hierarchy.

- **Configured IAP for TCP forwarding**. IAP for TCP forwarding is allowed through a security policy that permits ingress traffic from IP range `35.235.240.0/20` for TCP ports `22` and `3389`. The security policy is attached to all folders in the hierarchy.

- **Restricting network traffic**. In order to better control the network perimeter, a delegation for RFC 1918 is configured to lower levels in the hierarchy. This lets administrators define the appropriate access for hybrid and VPC connectivity. To do this, a firewall rule using the `goto_next` option for RFC 1918 IP ranges is configured and attached to all folders in the hierarchy.

7.8) VPC firewall rules

The example.com reference architecture uses VPC firewall rules with network tags to restrict traffic to VMs. VPC firewall rules reside in the Shared VPC networks and are deployed through the project deployment pipeline that's described in Section 5. Network tags are added to Google Cloud instances, and traffic is evaluated against those tags.

By default, example.com uses a deny-all firewall rule with priority `65530` to ensure that all traffic that's not explicitly allowed is denied. For your resources to be able access the `private.googleapis.com` VIP or the `restricted.googleapis.com` VIP, you need to create firewall rules that allow traffic to flow to those IP addresses.

Figure 2.7.6 illustrates how the VPC firewalls with network tags work in the example.com environment.

The example.com default deny-all firewall rule (`Rule 1`) is applied to all instances. Another firewall rule with a higher priority (`Rule 2`) allows port `443` traffic ingress to any compute instances that have the tag `https`. That means that the instance named `instance-2` can receive inbound traffic on port `443` because it's tagged with `https`. In contrast, the instance named `instance-3` doesn't receive inbound traffic, because it doesn't have a tag that matches a firewall rule.

By default, network tags can be added or removed from VMs by users with Owner, Editor, or Compute Instance Admin roles. However, to ensure that network tags are used in a more secure manner, the example.com reference architecture restricts Owner, Editor, and Compute Instance Admin roles from being provided to users or groups except by the firecall process that's described in Section 6.3.

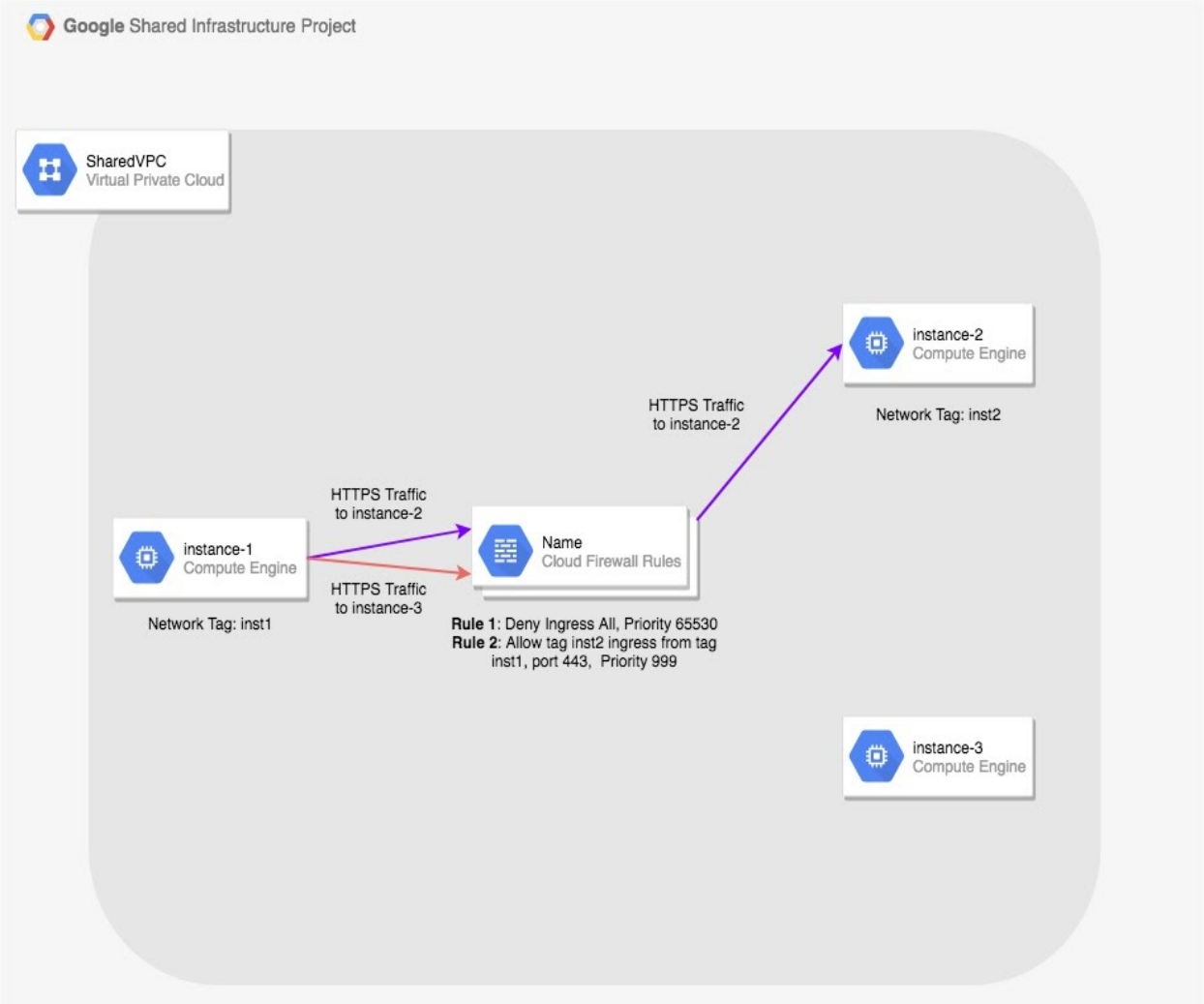

Figure 2.7.6 Firewall rules in example.com

In cases where the Compute Instance Admin (v1) role (`roles/compute.instanceAdmin.v1`) is required by users or groups, you can create a custom role that has all the permissions of the Compute Instance Admin (v1) role with the exception of the `compute.instances.setTags` permission. Someone with this custom role can't add or remove tags on VMs; the tags must be added by a controlled mechanism, depending on workload type.

Note: You need to add firewall rules in order to enable different workloads to run within the foundation. For example, if you want to run a Google Kubernetes Engine-based workload with ingress traffic, you need to enable certain IP ranges to be able to access your Google Kubernetes Engine. Firewall Insights provides visibility into your firewall rule usage and detects firewall rule configuration issues. You can use it to analyze deployed firewall rules, detecting overly permissive firewall rules.

8. Key and secret management

Secrets in your organization encompass cryptographic keys, credential sets, API keys, OAuth tokens, and other sensitive data that's needed in order to run your applications. Google Cloud offers the Cloud Key Management Service (Cloud KMS) for key management and Secret Manager for managing secrets. You can use these two services to help secure your Google Cloud environment and help you avoid insecure practices such as storing secrets in code.

8.1) Cloud Key Management Service

Cloud KMS is a unified API for performing cryptographic operations and managing cryptographic keys. The keys can be backed by software, hardware (HSM), or external systems (EKM). Cloud KMS is designed so that key material never leaves the service. Cloud KMS also offers customer-managed encryption key (CMEK) integrations that give you control over how Google Cloud services encrypt data at rest. Cloud KMS resources include key rings, keys, and key versions as defined in the following list:

- **Key ring**: Organizes keys in a specific Google Cloud location.
- **Key**: Contains zero or more key versions. This named resource exists on exactly one key ring. Keys specify parameters such as type and purpose.
- **Key version**: Contains key material that's used for performing cryptographic information. A key version can be assigned as the primary version for a key.

Cloud KMS is not a general-purpose secret store where you can expect to securely store secret material and retrieve it later. For that capability, you should use Secret Manager, as explained in Section 8.2.

8.1.1) Cloud KMS resource organization and access control

Key access and key ring access is managed by organizing keys into key rings and projects, and by granting IAM roles on the keys, key rings, and projects. As you build out your cloud environment, follow the guidance in the following list for how to design your key resource hierarchy to reduce risk:

1. Create a dedicated project for Cloud KMS that's separate from workload projects.
2. Add key rings into the dedicated Cloud KMS project. Create key rings as needed to impose separation of duties.

The following list provides guidance for how to apply IAM roles to support your security and administrative needs:

- Avoid the basic project-wide roles like `owner`, `editor`, and `viewer` on projects that host keys, or on enclosing folders or organizations. Instead, designate an organization administrator that's granted at the organization level, as noted on the Cloud KMS separation of duties page. The Organization Admin functions as the administrator for all the organization's cryptographic keys. Grant other IAM roles at the project level. If you have further concerns about separation of duties, grant IAM roles at the key ring level or key level.

- Use Cloud KMS predefined roles for least privilege and separation of duties. For example:
  - Separate administration roles (`roles/cloudkms.admin` and `roles/cloudkms.importer`) and usage roles for keys.
  - Limit the use of `roles/cloudkms.admin` to members of the security administration team and to service accounts that are responsible for creating key rings and key objects through tools such as Terraform.
  - For asymmetric keys, grant roles that need private key access (`roles/cloudkms.cryptoKeyDecrypter` and `roles/cloudkms.signer`) separately from roles that don't need private key access (`roles/cloudkms.publicKeyViewer` and `roles/cloudkms.signerVerifier`).
  - In general, grant the most limited set of permissions to the lowest object in the resource hierarchy.

As shown in Figure 2.4.1, the example.com reference architecture provides you with projects for storing keys and secrets at the organization level and at the environment (folder) level. The intent of having a separate secret repository per environment is to create a clear separation between organization-level and environment-level secrets.

8.1.2) Cloud KMS infrastructure decisions

When you're setting up Cloud KMS for a project, you must make several decisions regarding your keys. Table 2.8.1 provides you with guidance on what factors to consider when you create keys and key rings.

Key attribute	Key attribute guidance
Key location	Choose the location that's geographically closest to the data on which you will be performing cryptographic operations. Use the same key ring location for CMEK keys that's used for the data that they are encrypting. For more information, see choosing the best type of location in the Cloud KMS documentation.
Key protection level	Use protection level `EXTERNAL` when your workloads require keys to be maintained outside of the Google Cloud infrastructure in a partner system.
	Use protection level `HSM` when your workloads require keys to be protected in FIPS 140-2 Level 3-certified hardware security modules (HSMs).
	Use protection level `SOFTWARE` when your workload doesn't require keys to be maintained outside of the Google Cloud and doesn't require keys to be protected with FIPS 140-2 Level 3-certified hardware.
	Choose appropriate protection levels for development, staging, and production environments. Because the Cloud KMS API is the same regardless of protection level, you can use different protection levels in different environments, and you can relax protection levels where there is no production data.

Key source	Allow Cloud KMS to generate keys unless you have workload requirements that require keys to be generated in a specific manner or environment. For externally generated keys, use Cloud KMS key import to import them as described in Section 8.1.7.
Key rotation	For symmetric encryption, configure automation key rotation by setting a key rotation period and starting time when you create the key
	For asymmetric encryption, you must always manually rotate keys, because the new public key must be distributed before the key pair can be used. Cloud KMS doesn't support automatic key rotation for asymmetric keys.
	If you have indications that keys have been compromised, manually rotate the keys and re-encrypt data that was encrypted by the compromised keys as soon as possible. To re-encrypt data, you typically download the old data, decrypt it with the old key, encrypt the old data using the new key, and then re-upload the re-encrypted data.
Key destruction	Destroy old keys when there is no data encrypted by those keys.

Table 2.8.1 Cloud KMS key attribute guidance

8.1.3) Application data encryption

Your application might use Cloud KMS by calling the API directly to encrypt, decrypt, sign, and verify data. Applications that handle data encryption directly should use the envelope encryption approach, which provides better application availability and scaling behavior.

Note that to perform application data encryption in this way, your application must have Cloud IAM access to both the key and the data.

8.1.4) Integrated Google Cloud encryption

By default, Google Cloud encrypts all your data at rest and in transit without requiring any explicit setup by you. This default encryption for data at rest, which is transparent to you, uses Cloud KMS behind the scenes and manages Cloud IAM access to the keys on your behalf.

8.1.5) Customer-managed encryption keys (CMEK)

For more control over the keys for encrypting data at rest in a Google Cloud project, you can use several Google Cloud services that offer the ability to protect data related to those services by using encryption keys managed by the customer within Cloud KMS. These encryption keys are called customer-managed encryption keys (CMEK).

Google Cloud products that offer CMEK integration might require the keys to be hosted in the same location as the data used that's with the key. Cloud KMS might use different names for some locations than other services use. For example, the Cloud KMS multi-regional location `europe` corresponds to the Cloud Storage multi-region location `EU`. Cloud KMS also has some locations that are not available in all other services. For example, the Cloud KMS dual-regional location `eur5` has no counterpart in Cloud

Storage. You need to identify these requirements before you create the Cloud KMS key ring so that the key ring is created in the correct location. Keep in mind that you cannot delete a key ring.

The security foundation provides you with an example of CMEK with Cloud Storage. In the `4-projects` folder of the foundation, there's an example project where a key is created in the key ring that's in the environment folder's `secrets` project. That key is set as the default key for a bucket that's created in the `example` project.

8.1.6) Importing keys into Cloud KMS

Your workloads might require you to generate the keys outside of Cloud KMS. In this case, you can import key material into Cloud KMS. Furthermore, you might need to provide assurance to relying parties on the key generation and import processes. These additional steps are referred to as a *key ceremony*.

You use a key ceremony to help people trust that the key is being stored and used securely. Two examples of key ceremonies are the DNSSEC root key signing key (KSK) ceremony and the ceremony used by Google to create new root CA keys. Both of these ceremonies support high transparency and high-assurance requirements because the resulting keys must be trusted by the entire internet community.

The people involved in a key ceremony might include the following (using the role names in the US NIST publication A Profile for U.S. Federal Cryptographic Key Management Systems):

- System authority
- System administrator
- Cryptographic officer
- Key custodian
- Key owner
- Audit administrator
- Key-recovery agent
- Cryptographic key management system operator
- Internal or external witnesses

Before the ceremony, the participants and the audience must all agree to a script for the ceremony. During the ceremony, the participants follow the script and document any exceptions that occur.

Depending on the trust required for the key, you might need to perform the ceremony in a sensitive compartmented information facility (SCIF) or other secure facility that you should identify and reserve as part of preparing for the ceremony.

You need to document the performance of the key ceremony, and you might also need to record a video of the ceremony.

During the key ceremony, you generate the key material and encrypt a known plaintext into ciphertext. You then import the key material into Cloud KMS and use the ciphertext to verify the imported key. After you've successfully completed the key ceremony, you can enable the key in Cloud KMS and use it for cryptographic operations.

Because key ceremonies require a lot of setup and staffing, you should carefully choose which keys require ceremonies.

Note that this is a high-level description of the key ceremony. Depending on the key's trust requirements, you might need more steps.

8.1.7) Key lifecycle

During the course of operating your workloads, you need to manage the lifecycle of your keys. The US National Information Standards Institute (NIST) special publication (SP) 800-57, Part 1 describes a key management lifecycle that's divided into four phases: pre-operational, operational, post-operational, and destroyed. Table 2.8.2 provides a mapping of the NIST lifecycle functions from the publication to Cloud KMS lifecycle functions.

Lifecycle phase	NIST operations	Cloud KMS operations
Pre-operational	8.1.4 Keying-Material Installation Function	Key import
	8.1.5 Key Establishment Function	Key creation (symmetric, asymmetric)
Operational	8.2.1 Normal Operational Storage Function	Key creation in `SOFTWARE`, `HSM`, or `EXTERNAL` protection levels (symmetric, asymmetric)
	8.2.3 Key Change Function	Key rotation
Post-operational	8.3.4 Key Destruction Function	Key destruction (and recovery)

Table 2.8.2 Cloud KMS key lifecycle

In addition, Cloud KMS offers the capability to disable a key. Typically, you do this to the old key after rotating to a new key. This gives you a period to confirm that no additional data needs to be re-encrypted before the key is destroyed. You can re-enable a disabled key if you discover data that needs to be re-encrypted. When you're sure that there's no more data that was encrypted by using the old key, the key can be scheduled for destruction.

8.2) Secret Manager

Secret Manager is a secure and convenient storage system for API keys, passwords, certificates, and other sensitive data. Secret Manager provides a central place and single source of truth to manage, access, and audit secrets across Google Cloud. Secret Manager lets you set access policies for each secret and to configure audit logs for each secret access.

Secret Manager resources include secrets and secret versions as defined in the following list:

- **Secret**: A named resource that contains zero or more secret versions. Secrets let you manage access control and replication policy for your secret versions.
- **Secret versions**: The actual secret material being stored. Secret Manager allows multiple secret versions to be active at one time and lets you control the lifecycle of the stored secret material (video).

8.2.1) Secret Manager infrastructure decisions

When you're setting up Secret Manager for a project, you need to make several decisions regarding your secrets. Table 2.8.3 provides you with guidance on what factors you should consider.

Infrastructure	Recommended best practice
Resource hierarchy location	For secrets that are used by workloads in a single project, create secrets in the project where the workload resides.
	For secrets that are used by workloads in multiple projects, use a separate project for secrets that's distinct from the workload projects.
Access control	Grant Cloud IAM roles directly on secrets, rather than at the organization, folder, or project level. Using an infrastructure as code (IaC) approach can help with managing secret-level IAM role grants.
	Avoid using the basic project-wide roles like `owner`, `editor`, and `viewer` on projects that host secrets or on enclosing folders or organizations.
	Use Secret Manager predefined roles in order to enforce least privilege (video) and separation of duties.
	Create separate administration and access roles for keys: • Most accounts only need to access secret versions for specific secrets. Grant these accounts `roles/secretmanager.secretAccessor` for those specific secrets. • For processes that add new secret versions, grant `roles/secretmanager.secretVersionAdder`. • For processes that handle timed disabling and destruction of secret versions, grant `roles/secretmanager.secretVersionManager`.

Replication policy	Choose the `automatic` replication policy unless your workload has specific location requirements. The `automatic` policy meets the availability and performance needs of most workloads (video).
Audit logging	Secret Manager writes Admin Activity audit logs, which record operations that modify the configuration or metadata of a resource. You can't disable Admin Activity audit logs. You should enable data access logs at the folder or organization level so you can obtain and analyze AccessSecretVersion requests. For more information, see the Secret Manager audit logging documentation.
Secret rotation	Rotate secrets automatically and have emergency rotation procedures available in case of a compromise.
Accessing secret versions	Reference secrets by their version number rather than by using the `latest` alias (video). Configure your application with a specific secret version that's read on startup. Deploy updates to version numbers using your existing release processes. Although using the `latest` alias can be convenient, if there is a problem with the new version of the secret, your workload might be left unable to use the secret version. If you pin to a version number, the configuration can be validated and rolled back using your existing release processes.
Service account keys	Don't store Google Cloud service account keys in Secret Manager. Instead, use alternatives that use short-lived credentials rather than stored keys, such as service account impersonation or GKE workload identity. This reduces the risk of an attacker capturing credentials from storage.
Secrets for high-availability workloads	For high-availability processes that use secrets, prevent flag-day events by storing multiple sets of credentials (identifier and secret), not just the secret. This helps prevent downtime when a secret is rotated. For example, a workload uses Secret Manager to store a database password. When the password is rotated, process restarts might be required across the workload. If there's only one set of credentials, this can cause an outage. To help mitigate this type of outage, have multiple database user accounts with the same access, and rotate one account's password at a time.

Table 2.8.3 Secret Manager infrastructure guidance

8.2.2) Secret Manager content decisions

Secret material that's stored in secret versions must be no larger than the size indicated in the Secret Manager quotas and limits documentation. Other than this specification, Secret Manager doesn't restrict the format of the secret material.

For secrets like API keys, SSH keys, and cryptographic keys that can't reside in Cloud KMS, the secrets are generated in external systems that impose formatting constraints. For secrets like passwords where the format and length don't have such constraints, use randomly generated, long, complex values, and use cryptographically strong random number generators.

8.2.3) Secret Manager lifecycle

Table 2.8.4 describes the lifecycle of secrets.

State	Trigger	Next state
(Secret does not exist)	Creation	Created
Created	Deletion	(Secret does not exist)

Table 2.8.4 Secret Manager lifecycle for secrets

Table 2.8.5 describes the lifecycle of secret versions.

State	Trigger	Next state
(Secret version does not exist)	Creation	Enabled
Enabled	Disable	Disabled
Disabled	Enable	Enabled
Enabled	Destroy	(Secret version does not exist)
Disabled	Destroy	(Secret version does not exist)

Table 2.8.5 Secret Manager lifecycle for secret versions

Follow the guidance in Table 2.8.6 for best practices that you can use for different secret and secret version lifecycle states.

Secret lifecycle stage	Secret lifecycle guidance
Enabled	Access Secret Manager using the API or a client library unless programmatic access isn't feasible, because that's the most secure method.
	Use the API or client library to access secrets. You might be unable to use the API because your workload involves legacy apps or closed source tools, or because it requires multi-cloud portability. If you can't use the API or client library directly, use environment variables or file paths, but be aware that those methods increase the risk that attackers can gain unauthorized access to your secrets.
Disabled	If you're going to disable a secret version after rotating it, then wait until no other processes are using the old secret version. Secret Manager allows multiple secrets to be enabled at once, but you should disable old secret versions as soon as your workloads aren't using them. You can use Data Access audit logs to help identify unused secret versions.
Destroyed	Destroy secret versions after a reasonable period following disablement. Because destroying secret versions is irreversible, you shouldn't destroy them until you're sure the secret version is no longer in use.

Table 2.8.6 Secret Manager lifecycle best practices

9. Logging

Logging provides important functionality to development organizations, audit organizations, and security organizations, as well as helping to satisfy regulatory compliance. As shown in Figure 2.9.1, there are a number logging sources in the example.com organization that are aggregated by Google Cloud Logging.

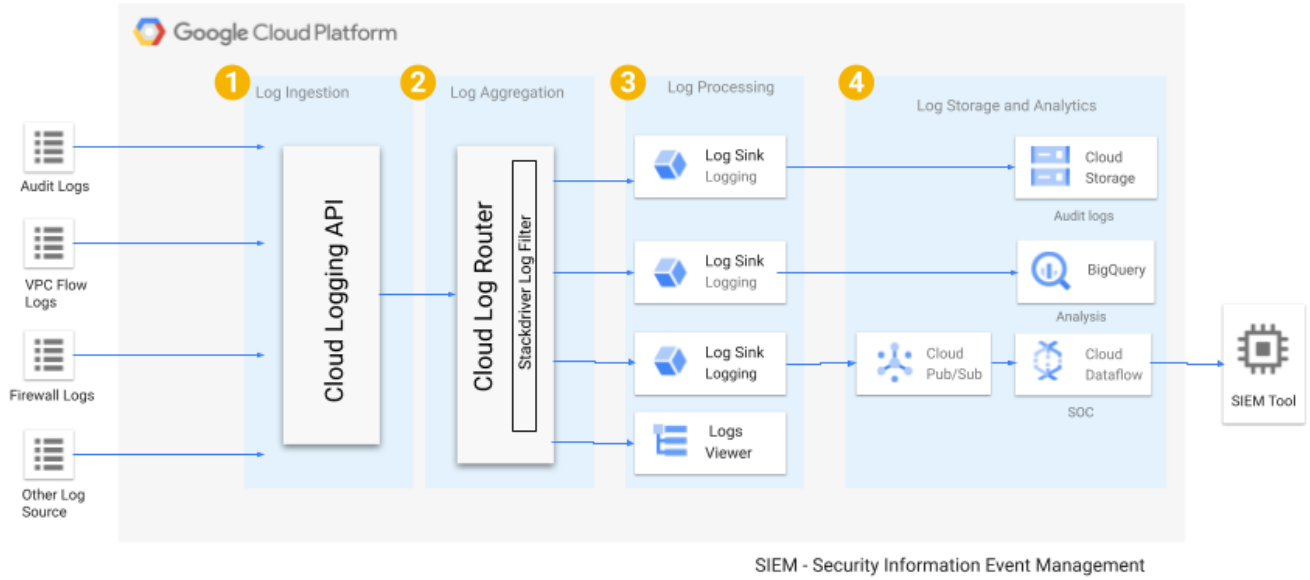

Figure 2.9.1 Logging structure for example.com

Within the reference example.com organization, logging functions are split into two projects:

- A standalone logging project named `Logging`. This project contains Pub/Sub topics, Cloud Functions and a BigQuery instance used for collecting and processing logs generated by the example.com organization.
- A SCC Alert Center project named SCC. This project contains the notification Pub/Sub topics from the Security Command Center. It's separate from the logging project so that you have a clear separation of duties between operations teams that might need general log access and the security team that needs access to specific security information and events.

In Google Cloud, logs are sent to the Cloud Logging API, where they pass through the Logs Router. The Logs Router checks each log entry against existing rules to determine which log entries to discard, which log entries to ingest (store) in Cloud Logging, and which log entries to export through log sinks. Table 2.9.1 shows the logs that are collected as part of example.com foundation and how those logs are enabled.

Log source	Description	Management
Audit Logs	Google Cloud services write audit log entries to these logs to answer the question of "who did what, where, and when?" with Google Cloud resources.	Enabled at an organization level and configured by the pipeline during the initial organization setup.
VPC Flow Logs	VPC Flow Logs records a sample of network flows that are sent from and received by VM instances, including instances that are used as GKE nodes. The sample is typically 50% or less of the VPC network flows.	Enabled for each VPC subnet and configured by the pipeline during project creation.
Firewall Rules Logging	Firewall logging creates a record each time a firewall rule allows or denies traffic.	Enabled for each firewall rule and configured by the pipeline during firewall creation.
Google Workspace audit logging	Google Workspace allows its logs to be shared with the Google Cloud logging service. Google Workspace collects Login logs, Admin logs, and Group logs.	Enabled at an organization level and configured through the Cloud Identity Admin Console.
Data Access audit logs	Data Access audit logs record API calls that read the configuration or metadata of resources, as well as user-driven API calls that create, modify, or read user-provided resource data.	Enabled at an organization level and configured by the pipeline during the initial organization setup.
Access Transparency logs	Access Transparency provides logs that capture the actions Google personnel take when accessing your content.	Enabled at an organization level and configured by raising a support case with Google Cloud.

Table 2.9.1 Log sources in example.com

Exporting to a sink involves writing a query that selects the log entries that you want to export and choosing a destination in Cloud Storage, BigQuery, or Pub/Sub. Logging information can be excluded by changing the query. Table 2.9.2 describes the sinks that are used in the example.com architecture.

Sink	Description	Purpose
`sk-c-logging-bq`	Sends the aggregate logs from the organization to a BigQuery table in the logging project.	The aggregated logs can be analyzed in BigQuery and used as part of the detective control architecture that's described in Section 10.
`sk-c-logging-bkt`	Sends the aggregate logs from the organization to a Cloud Storage bucket in the logging project.	The aggregated logs can be used for compliance, audit, and incident-tracking purposes. For regulatory purposes, you can apply Cloud Storage Bucket Lock.
`sk-c-logging-pub`	Sends the aggregated logs from the organization to a Pub/Sub topic in the logging project.	The aggregated logs are sent from Pub/Sub to a Dataflow job and from there to an external SIEM, as described in Section 10.

Table 2.9.2 Log sinks in example.com

The sinks and sink destinations are created through the deployment pipeline when the organization objects are first created and configured. When a sink is created, a service account identity is returned; that service account must have permission to write to the specified destination. Those permissions are also configured during setup.

Note: In conjunction with logging, you should use SRE concepts (SLI, SLO, SLA) in order to monitor and maintain your environment.

10. Detective controls

Detective controls use platform telemetry to detect misconfigurations, vulnerabilities, and potentially malicious activity in the cloud environment. As shown in Figure 2.10.1, the example.com reference architecture has extensive detective controls to support its overall security posture.

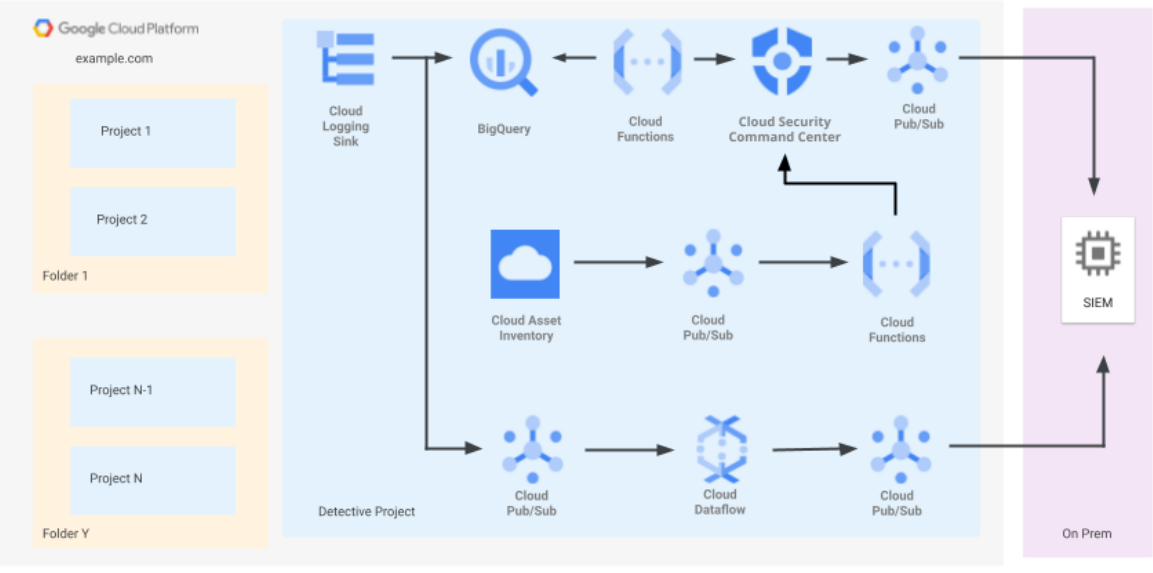

Figure 2.10.1 Detective control architecture in example.com

The example.com architecture includes the following detective controls:

- Google security sources through Security Command Center, including security misconfiguration information and vulnerability changes. (For details, see the next section.)
- Connection of Google Cloud logs to Chronicle.
- Asset changes and policy compliance notifications from Cloud Asset Inventory.
- Logs from Google Cloud Logging (including Google Workspace Audit logs) that are sent to the SIEM tool.
- Custom findings from a serverless BigQuery-based tool.

In addition, if you have existing event sources from any of our third-party providers or have created your own custom sources and custom findings, you can integrate those findings into Security Command Center.

10.1) Security Command Center

Security Command Center in Google Cloud provides a single user interface and data platform to aggregate and manage security findings. The example.com architecture uses Security Command Center to aggregate native findings from Google security sources as listed in Section 10.1.2 and custom findings from the BigQuery analysis solution. It then sends Security Command Center notifications to a Pub/Sub topic for the SIEM to consume.

10.1.1) Premium and Standard

Google Security Command Center is available in two versions:

- Security Command Center Premium. This is focused on customers with strong security requirements, especially enterprise customers. It includes a full suite of vulnerability and threat detection capabilities as well as integration options that simplify both the connection to third-party SIEMs and the automation of response and remediation actions. Security Command Center Premium is a paid service.
- Security Command Center Standard. This provides a basic subset of the Security Command Center Premium features for all customers. Its focus is on enabling customers to discover, identify, alert on, manage, and respond to the critical vulnerabilities in their estate. Security Command Center Standard is available to customers free of charge.

Whether you choose Premium or Standard, you can enable the selected Security Command Center capabilities and manage the setup and configuration of the native detectors in the Security Command Center in a unified, centralized way.

The built-in security sources in both Security Command Center Premium and Standard are set by default to apply to all existing and future folders and projects in your Google Cloud organization. Unless there is a specific special security situation, we strongly recommend that you keep this default setting.

10.1.2) Security sources

Security Command Center aggregates findings from a wide variety of security sources, consisting of native Google provided findings, third-party findings, and customer-generated custom findings. Table 2.10.1 lists the security event sources that are enabled for the example.com reference architecture.

Security event source	Description
Security Health Analytics	Detects common vulnerabilities, misconfigurations, and drift from secure configurations.
Anomaly Detection	Uses behavior signals from outside your system to display granular information that helps you detect cloud abuse behavior for your projects and VM instances.
Event Threat Detection	Automatically scans various types of logs for suspicious activity in your Google Cloud environment.
Web Security Scanner	Scans public web applications for common application vulnerabilities.
Container Threat Detection	Provides runtime security for GKE containers to help detect reverse shells, suspicious binaries, and libraries.

Table 2.10.1 Security event sources for example.com

Note: Security Command Center is also integrated with a number of third-party security sources. In addition, your own applications can generate custom findings in Security Command Center through the findings and sources APIs. If your organization uses third-party security sources or custom findings, you can manage the resulting findings through the Security Command Center dashboard. The findings can also be shared with another system like an enterprise SIEM using Security Command Center API notifications.

10.1.3) Setting up basic security alerting

Beyond detection, it's important for you to be able to take action to respond and remediate detected vulnerabilities and threats. You should leverage the built-in Security Command Center Pub/Sub topic to connect Security Command Center findings to your downstream alerting systems, ticketing, workflow, SOAR systems, and SIEMs. Security Command Center Pub/Sub notifications are managed through a set of notification configs. You can create your notification configs using either the `gcloud` command-line tool or client libraries and the API. To start, you should create a dedicated SCC Alerts project and restrict IAM access to just the users and services that are allowed to change and manage notification configurations. Within the project, you can then set up the Pub/Sub topics to which you will be publishing events. For each topic, you should further restrict access to that topic to only the necessary individuals and services whose duties require access to the security findings information in the topic.

10.1.3.1) Configuring notifications

After the security-alerting project is created and the topics have been created, you should define your notification configs. For detailed instructions, see Setting up finding notifications in the Security Command Center documentation. For information about managing notification configs, see Creating and managing Notification Configs, and for information about filtering notifications, see Filtering notifications.

The default quota on notification configuration counts is 500 per organization. If you need more, you should submit a request to increase your Security Command Center API quota using the Google Cloud Console, as shown in Figure 2.10.2.

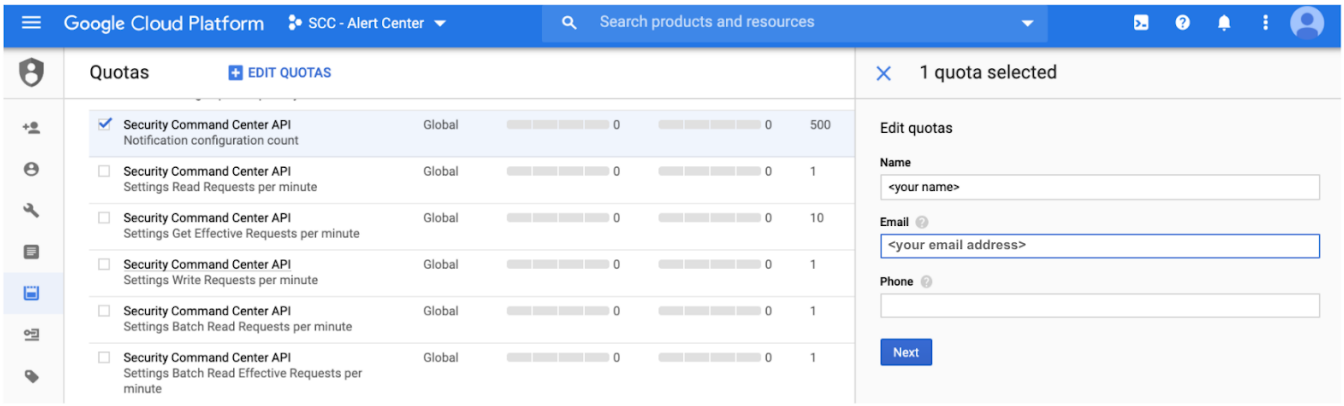

Figure 2.10.2 Request in Cloud Console for additional notification configs quota

The first notification config you should create is one that sends all Security Command Center findings to a Pub/Sub topic that you can then connect to your SIEM. You need the following roles:

- Security Center Admin (broad permissions)
  or
  Security Center Notification Config Editor (specific permissions)
- Pub/Sub Admin on the Pub/Sub topic

After you've set up the roles, follow these instructions:

1. In the Cloud Console, open Cloud Shell, or open a terminal at a computer where you have the Cloud SDK installed and configured.
2. Set up the Pub/Sub topic:

```
# topic-id = siem-export
gcloud pubsub topics create topic-id
export TOPIC_ID=topic-id

# subscription-id = siem-subscription
gcloud pubsub subscriptions create subscription-id --topic topic-id
```

3. Set up the notification config with the filter:

```
export ORGANIZATION_ID=organization-id
export PUBSUB_PROJECT=project-id
export GCLOUD_ACCOUNT=your-username@email.com
```

4. Grant the `gcloud` account the required roles and permissions:

```
gcloud pubsub topics add-iam-policy-binding \
    projects/$PUBSUB_PROJECT/topics/$TOPIC_ID \
    --member="user:$GCLOUD_ACCOUNT" \
    --role='roles/pubsub.admin'

gcloud organizations add-iam-policy-binding $ORGANIZATION_ID \
    --member="user:$GCLOUD_ACCOUNT" \
    --role='roles/securitycenter.notificationConfigEditor'

# The topic to which the notifications are published
PUBSUB_TOPIC="projects/project-id/topics/topic-id"

# The description for the NotificationConfig
DESCRIPTION="Notifies for active findings"
```

```
# Filters for all active findings - both vulnerabilities and threats
FILTER="state=\"ACTIVE\""

gcloud alpha scc notifications create notification-name \
    --organization "$ORGANIZATION_ID" \
    --description "$DESCRIPTION" \
    --pubsub-topic $PUBSUB_TOPIC \
    --filter $FILTER
```

For the full set of instructions, including information about how to set up notifications configs using the client libraries and API, see Setting up finding notifications.

10.1.3.2) Matching the notification configurations to your organization's hierarchy

You can use the definition and organization of notification topics to configure the routing of security alerts to the appropriate teams within your organization. Security Command Center enables you to create and store a set of notification configurations, each of which is a unique combination of selection filter and Pub/Sub topic. You match your specific requirements and enforce IAM access policies to security findings by controlling the following:

- The number of notification configs that you create.
- The selection filter in each config.
- The name of the topic where the selected and filtered findings events are published.

The following sections describe four examples of successful notification configuration and topic patterns.

10.1.3.2.1) One security queue

The simplest pattern is just one notification config, with the selection filter set for all projects and all findings categories and the target set to just one topic (for example, `gcp-security-alerts`). This pattern is for customers who want all Google cloud security vulnerability and threat findings to be routed to a single centralized Security Operations Center (SOC).

10.1.3.2.2) By line of business

You should create multiple separate notification configs when you've adopted the approach where each line of business has its own SOC and therefore owns the end-to-end security responsibilities for your infrastructure, workloads, and applications. Then you should set up a unique Security Command Center notification config and a dedicated topic for each line-of-business SOC. The selection filter in the notification config should select all projects within the scope of the line of business and select all findings categories. This configuration enables IAM control and separation on the dedicated topics between each line-of-business team so that each team has access to and receives only the vulnerability and threat signals relevant to them.

10.1.3.2.3) Cloud-native DevSecOps

Similarly, you should create multiple separate notification configs when you've adopted the approach where each application team owns the end-to-end security responsibilities for their infrastructure,

workloads, and applications. Then you should set up a unique Security Command Center notification config along with a dedicated topic for each application team. The selection filter in the notification config should select all projects within the scope of the application team and all findings categories. This configuration enables IAM control and separation on the dedicated topics between each application team so each application team has access to and receives only the vulnerability and threat signals relevant to them.

10.1.3.2.4) By Security finding category

You should create multiple separate notification configs when you've adopted the approach where different types of security findings are handled by different customer teams. The most common example of this is when you've chosen to separate the response and remediation of vulnerabilities and misconfigurations from those of live, active threats. We've seen many customers route the first to their cloud security team and the second to their SOC. In this example case, you can set up a unique Security Command Center notification config and a dedicated Pub/Sub topic for the two teams. The selection filter in the notification config for misconfigurations and vulnerabilities should select all projects and all the finding sources for vulnerabilities (for example, Security Health Analytics and Web Security Scanner). The selection filter in the notification config for threats should select all projects and all the finding sources for threats (for example, Event Threat Detection, Container Threat Detection, and Abuse Anomaly Detection).

10.2) Vulnerability and drift detection

Configuration drift refers to changes in configuration away from a predefined standard or baseline. Over time, as a solution evolves, a certain amount of drift is necessary, but for security or stability reasons, some configuration details should not be violated. Managing drift can be a manual process where an administrator is notified and then takes action where needed. But it can also be automated for specific changes, reverting a change that violates a policy.

10.2.1) Built-in drift detection using Security Command Center Premium

Security Command Center Premium includes Security Health Analytics, which automatically runs over 100 different misconfiguration and vulnerability checks daily against all the Security Command Center-supported resources in your organization. The resulting assessments are listed in the Vulnerabilities tab of the Security Command Center dashboard. You can view and explore the data either at the organization level or for specific sets of projects. You can also further filter the data by status, category, recommendation, severity, active or inactive status, and compliance benchmark. Then in the Compliance tab dashboards, the assessment results are specifically matched against CIS 1.0, NIST-800-53, PCI-DSS, and ISO 27001 benchmarks and summarized for either the full organization or for a specific set of customer-selected projects. Examples of both tabs are shown in Figure 2.10.3 and Figure 2.10.4.

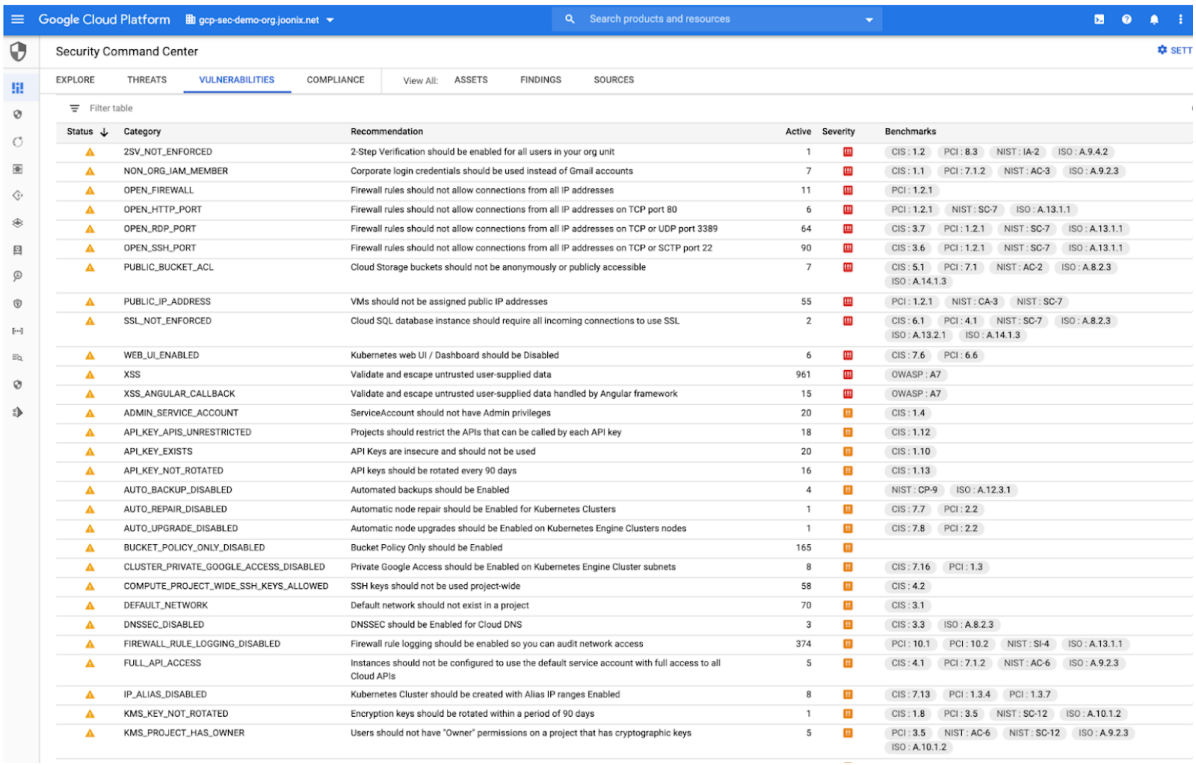

Figure 2.10.3 The Vulnerability tab in Security Command Center Premium

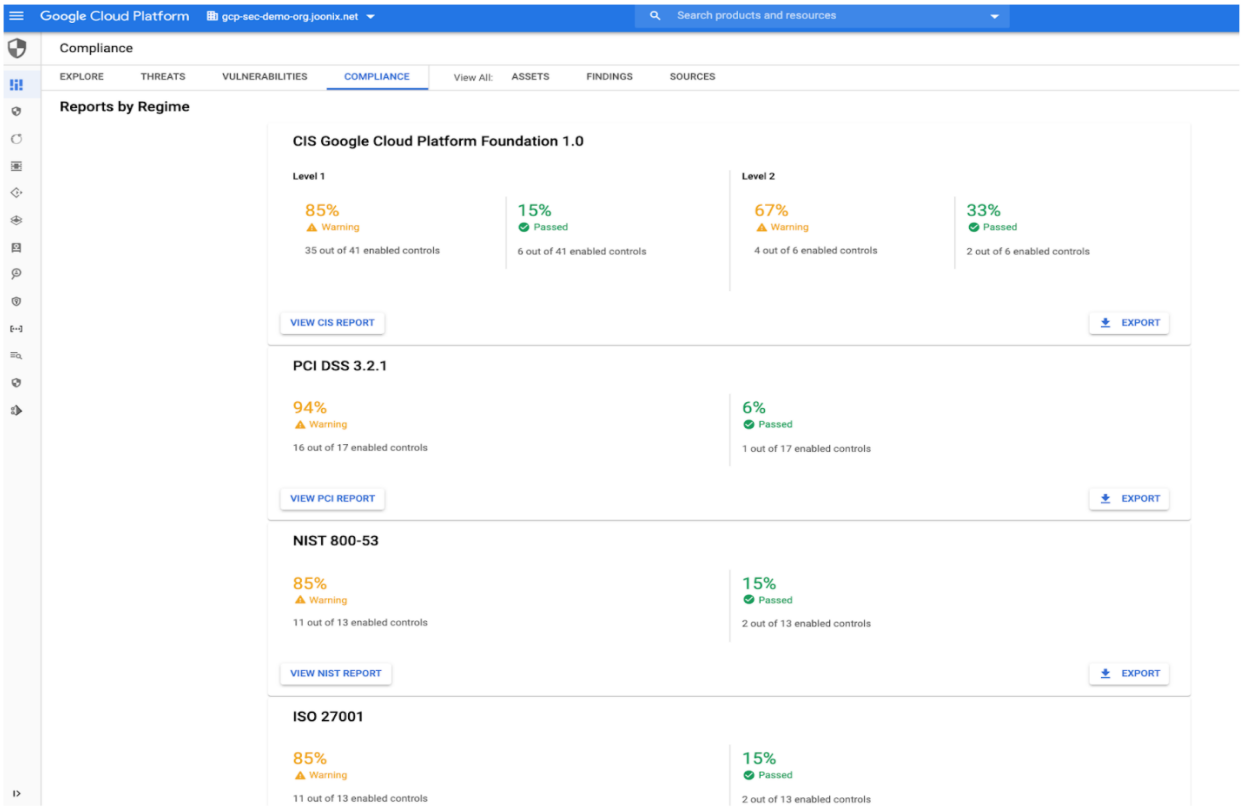

Figure 2.10.4 The Compliance tab in Security Command Center Premium

With the Security Command Center, you can set up different notification configs to generate alerts on changes in the Security Health Analytics findings to help you identify drift in the configurations of the resources that you care about. In addition, in the Security Command Center dashboard, you can use the Asset Changed view and Findings Changed view to discover and explore changes in user-selected time ranges (for example, last hour, last day, last 7 days, last 30 days, and so on). Examples of both tabs are shown in Figure 2.10.5 and Figure 2.10.6.

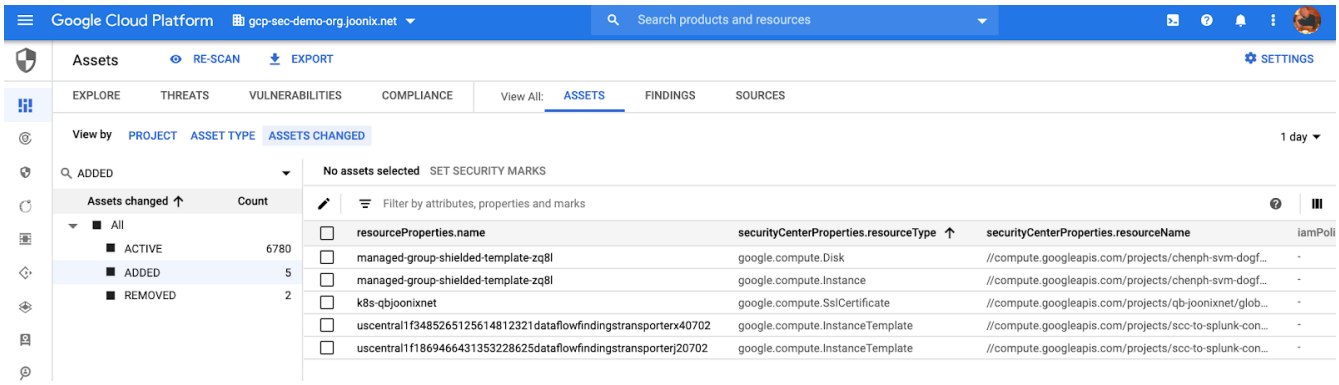

Figure 2.10.5 The Asset Changed tab in the Security Command Center dashboard

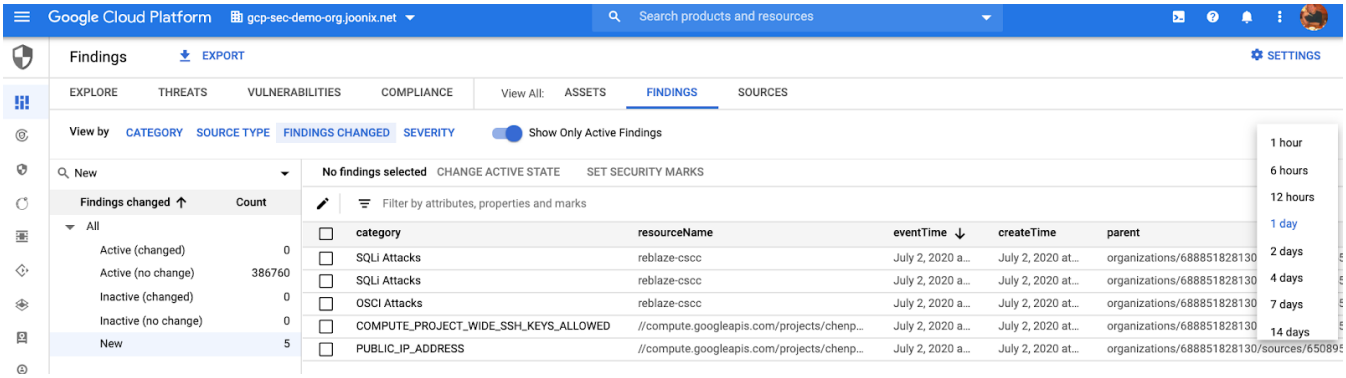

Figure 2.10.6 The Findings Changed tab in the Security Command Center dashboard

In the dashboards, you can explore the time ranges relative to now—that is, to the current time. If instead you want the reference point to be a different time, you can use either the Security Command Center command-line interface or the API or client libraries to set your own reference point.

10.2.2) Managed web vulnerability scans

If you deploy web applications, you should also leverage Security Command Center Premium's built-in managed web vulnerability scans. Web Security Scanner currently provides 13 built-in web vulnerability detections that cover 4 of the OWASP top 10. Managed scans automatically run once each week to detect and scan public web endpoints.

10.3) Active threat detection

In addition to enabling built-in vulnerability and misconfiguration assessments and threat prevention, you should also leverage Security Command Center Premium's built-in active threat detection capabilities. The initial set that's built in to Security Command Center Premium includes Event Threat Detection and Container Threat Detection.

10.3.1) Event Threat Detection

Event Threat Detection is a built-in service with Security Command Center Premium that detects threats that target your Google Cloud assets. It provides near real-time detection of threats from platform logs, network logs, and compute logs, and it leverages native Google threat intelligence and detection algorithms. Findings are automatically written to the Security Command Center and can also be exported to Cloud Logging. Event Threat Detection performs basic deduplication on findings for users and is enabled by default when you're using Security Command Center Premium.

10.3.2) Container Threat Detection

Container Threat Detection is a built-in service with Security Command Center Premium that detects threats that target Google Kubernetes Engine (GKE) containers. It provides near real-time detection of reverse shell execution, suspicious binary execution, and the linking of suspicious libraries. Container Threat Detection is uniquely able to make the connection between containers and underlying physical nodes at the time of threat detection.

To use Container Threat Detection, you must run the latest Container-Optimized OS image for your GKE environment. Container Threat Detection automatically deploys and manages a daemonset container on every node to transmit data plus additional information that identifies the container. Findings are automatically written to Security Command Center and Cloud Logging.

10.4) Real-time compliance monitoring of custom policies

To monitor compliance of custom policies in real time, the example.com architecture uses Cloud Asset Inventory real-time notifications. Cloud Asset Inventory can send a Pub/Sub message for each configuration change to a specific asset name or to an asset type. The message triggers a Cloud Function to examine the change. If that change represents a policy violation, the Cloud Function can take action such as reverting the change or notifying an administrator.

One policy that's monitored in example.com is if external Gmail accounts are being granted any permissions to example.com projects. A Cloud Function is used to check when an IAM policy is created or modified that has a Gmail address as a member. If the Cloud Function detects that the policy has been violated, the Cloud Function reverts the change and sends a custom finding to the Security Command Center. Figure 2.10.7 shows this configuration.

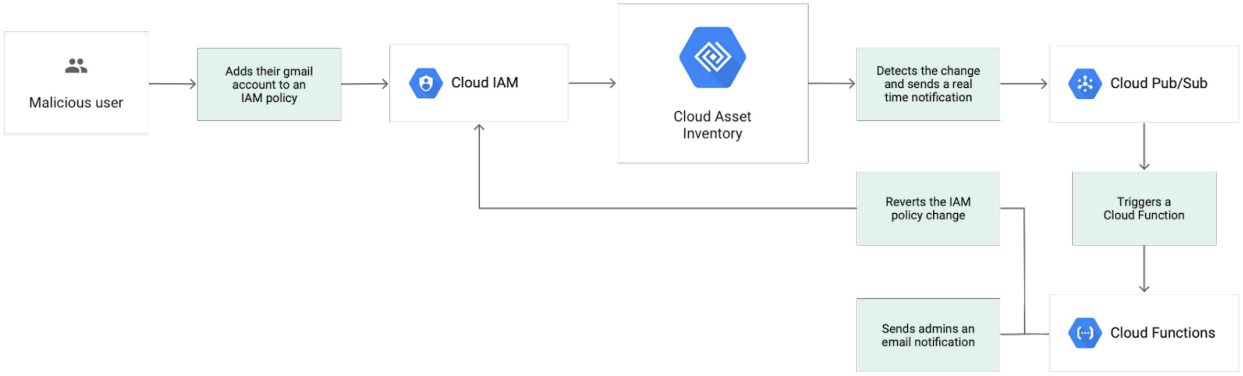

Figure 2.10.7 Automatically reverting an IAM policy change and sending a notification

Cloud Asset Inventory also enables you to export configuration metadata to a BigQuery table for in-depth analysis. When the data is in BigQuery, you can use SQL queries to explore the configuration and build reports. Listing 2.10.1 shows an example to detect IAM policies that grant access to a Gmail account.

```
SELECT name, asset_type, bindings.role
FROM `PROJECT_ID.DATASET_ID.TABLE_NAME`
JOIN UNNEST(iam_policy.bindings) AS bindings
JOIN UNNEST(bindings.members) AS members
WHERE members like "%@gmail.com"
```

Listing 2.10.1 SQL query to identify IAM bindings with @gmail.com members

10.5) Integration with Chronicle

Cloud Logging and Security Command Center events can be exported to Chronicle, which is purpose-built for security threat detection and investigation. Chronicle is built on the Google infrastructure, which lets you take advantage of Google's scale to reduce your time to investigate and triage potential threats. It correlates data from multiple security sources and threat feeds, providing a single timeline-based aggregated view of events and detections. You can leverage Chronicle's detection rules or use real-time search and custom rules to create your own detections.

10.6) SIEM solutions integrations

An enterprise SIEM product can be useful for the overall aggregation and visibility of security events, but sometimes the SIEM can be inefficient when dealing with cloud-scale logging. Both Security Command Center and Cloud Logging can output configurable built-in Pub/Sub event streams. You can therefore choose the option that best fits your needs.

As shown earlier in Figure 2.10.1, the example.com reference architecture includes a mechanism for a SIEM tool to ingest Google Cloud logs through a Pub/Sub subscription. A log sink in Cloud Logging is used to put all required logs into a raw logs Pub/Sub topic. A Dataflow job subscribes to the raw logs topic and aggregates the logs before putting them into a processed-logs Pub/Sub topic to which the SIEM can subscribe.

Note: Different enterprise SIEM tools can integrate with Google Cloud in a number of ways to receive logs, depending on the tool and the log source. Some other integration methods that you can use are the following:

- Direct ingestion from the Security Command Center-native Pub/Sub topic.
- Direct ingestion from a Pub/Sub log sink (if volumes are manageable).
- Flat-file ingestion from Cloud Storage or from a file server.
- Cloud Functions to take Pub/Sub events and turn them into API calls to the SIEM.
- Pulling logs from Google Cloud using Google APIs.

10.6.1) Integrations with Splunk

Both Cloud Logging and Security Command Center events can be directly exported to Splunk and can leverage the Splunk Add-on for Google Cloud for integration. For Cloud Logging, you can set up a logging export to Splunk. For Security Command Center, you can set up a notification config. In both cases, after the Pub/Sub event stream is set up, you can leverage the approved Pub/Sub Splunk Dataflow template to complete the integration.

10.7) Analyzing your security data using BigQuery

In cases where exporting to an enterprise SIEM is inefficient, an option is to build your own security analysis solution that's made up of cloud-native tools, including BigQuery, Pub/Sub, Cloud Functions, and Security Command Center. This solution is capable of working with massive log volumes. It can identify security events and forward the events to an enterprise SIEM as findings, rather than as raw logs.

10.7.1) Building your own analysis solution

The example.com reference architecture has some cloud-specific security events that are monitored through the BigQuery-based solution. These security event logs are aggregated and surfaced as custom findings in Security Command Center, and are forwarded to the enterprise SIEM tool through the notifications Pub/Sub topic. This BigQuery-based architecture can handle log sources that have a high log volume, such as data-access logs and VPC Flow Logs. For details, see Figure 2.10.8.

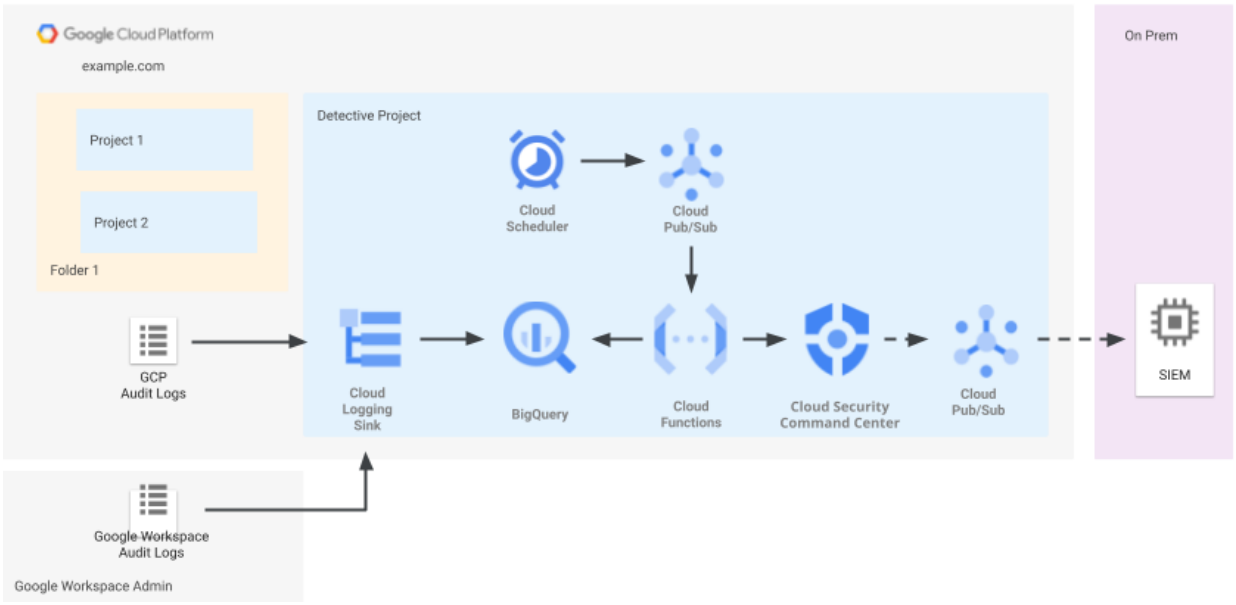

Figure 2.10.8 BigQuery-based analysis architecture

The BigQuery-based solution works as follows:

1. Logs from Cloud Logging are stored to a raw-data dataset in BigQuery using a log sink. This dataset can be configured to expire on a regular basis in order to help manage costs.
2. Views are added to a BigQuery dataset that represent the security events being monitored. A view is like a stored query, and in this case, it should produce one entry for each incident of the security event being monitored. Monitored use cases are described in Section 10.7.2.
3. Cloud Scheduler pushes an event to a scheduling Pub/Sub topic every 15 minutes. A Cloud Function is configured to trigger on every scheduling event. The Cloud Function queries the views for new events in the last 20 minutes to ensure no missed events; if it finds events, it pushes them to the Security Command Center API as custom findings.
4. The Security Command Center notifications feature pushes all new findings, including custom and Google or third-party security sources, to a Pub/Sub findings topic.
5. The enterprise SIEM subscribes to this findings topic to receive the security events.

10.7.2) Examples of alerting use cases

Alerting use cases are implemented as SQL queries that are saved as views in BigQuery. The following list specifies conditions that generate alerts using the BigQuery-based architecture.

- A login occurs from a high-privilege account login (Super Admin, Organization Admin, and so on).
- Cloud IAM permissions are granted to a user who is from a non-allowed domain.
- Changes are made to logging settings.
- VPC Flow Logs with IP addresses that are outside of expected ranges are detected.
- Changes are made to permissions on critical encryption keys.
- Non-approved Google services are used.

You can extend these use cases by writing a query and saving it as a view in the BigQuery dataset so that it gets queried regularly for new results with views for other use cases. As an example, the following query (Listing 2.10.2) addresses the case where logging settings are changed.

```
SELECT
  receiveTimestamp,
  timestamp AS eventTimestamp,
  protopayload_auditlog.requestMetadata.callerIp,
  protopayload_auditlog.authenticationInfo.principalEmail,
  protopayload_auditlog.resourceName,
  protopayload_auditlog.methodName
FROM
  `${project}.${dataset}.cloudaudit_googleapis_com_activity_*`
WHERE
  protopayload_auditlog.serviceName = "logging.googleapis.com"
```

Listing 2.10.2 SQL query to detect logging setting changes

11. Billing

You can use billing as a mechanism to enhance governance and security by monitoring Google Cloud usage and alerting on unexpected consumption. Google Cloud charges are handled by associating projects with billing accounts. The example.com reference architecture uses a single billing account that all projects are associated with when projects are created, as described in Section 5. Table 2.6.2 lists the groups and roles that are associated with the example.com billing account roles.

11.1) Billing alerts

You can set budgets at a project level or organization level, either as a fixed amount to suit steady-state expenditure, or as a percentage of the previous month's spend to suit variable costs. Billing alerts can be applied to different scopes within the organization. For example.com, budgets and associated billing alerts are created when the project is created. Billings alerts are also applied to the example.com billing account to provide organization-level budget alerts.

The example.com reference architecture has billing alerts that are triggered when the project or organization consumption reaches 50%, 75%, 90%, and 95% of the budget threshold. By default, billing alerts are sent to the billing administrator and to billing user email accounts. For example.com, these are a service account and firecall account, so emails to the service account and firecall account need to be forwarded. It's important to note that billing alerts on budget spend don't stop usage when the limit has been met; they are only a notification.

Note: If more than the default recipients for billing alerts need to be alerted, you can configure additional recipients through the Cloud Console.

11.2) Billing exports and chargeback

The Cloud Console has extensive cost management tools that you can use to view and forecast costs in a variety of formats. In the example.com model, these cost management tools are augmented by exporting all billing records to a BigQuery dataset in the Billing project. The export needs to be enabled through the Cloud Console. The exported billing records include the project label metadata that's assigned to the project during project creation, as listed in Table 2.5.3. As specified, each project has an associated billing code and contact points that can be used for chargeback. A simple SQL query on the exported dataset, as shown in Listing 2.11.1, can be used to provide billing information for each business unit within example.com.

```
#standardSQL
SELECT
   (SELECT value from UNNEST(labels) where key = 'business-code') AS bu,
   service.description AS description,
   SUM(cost) AS charges,
   SUM((SELECT SUM(amount) FROM UNNEST(credits))) AS credits
FROM `PROJECT_ID.DATASET_ID.TABLE_NAME`
GROUP BY bu, description
ORDER BY bu ASC, description ASC
```

Listing 2.11.1 Example query that groups charges by business unit and service

12. Creating and deploying secured applications

The Bank of Anthos secured application example is a container-based web application that lets you simulate an online bank. The example provides you with a reference cloud technology stack that includes the secured foundation, the Google Cloud services that are necessary in order to support the Bank of Anthos, and the Bank of Anthos application code.

The Bank of Anthos secured application example is deployed using the pipelines (foundation pipeline, infrastructure pipeline, and application pipeline) that are described in Section 5 and using Anthos Config Management.

The example architecture is based on design patterns that are defined in the Hardening your cluster's security guide, the GKE safer cluster repository, the Anthos Multicloud Workshop repository, the Anthos security blueprint, and the Bank of Anthos application. The complete code for the Bank of Anthos secured application example can be found in `terraform-example-foundation-app` GitHub repository.

12.1) The Bank of Anthos secured application platform architecture

Figure 2.12.1 shows how the Bank of Anthos secured application example builds on the secured foundation for its deployment. In this example, the Bank of Anthos is deployed across the three foundation environments: `development`, `production`, and `non-production`. The hub-and-spoke VPC networking topology that's detailed in Section 7.2 is used by the application for networking. The application uses Google Cloud projects as a logical security boundary.

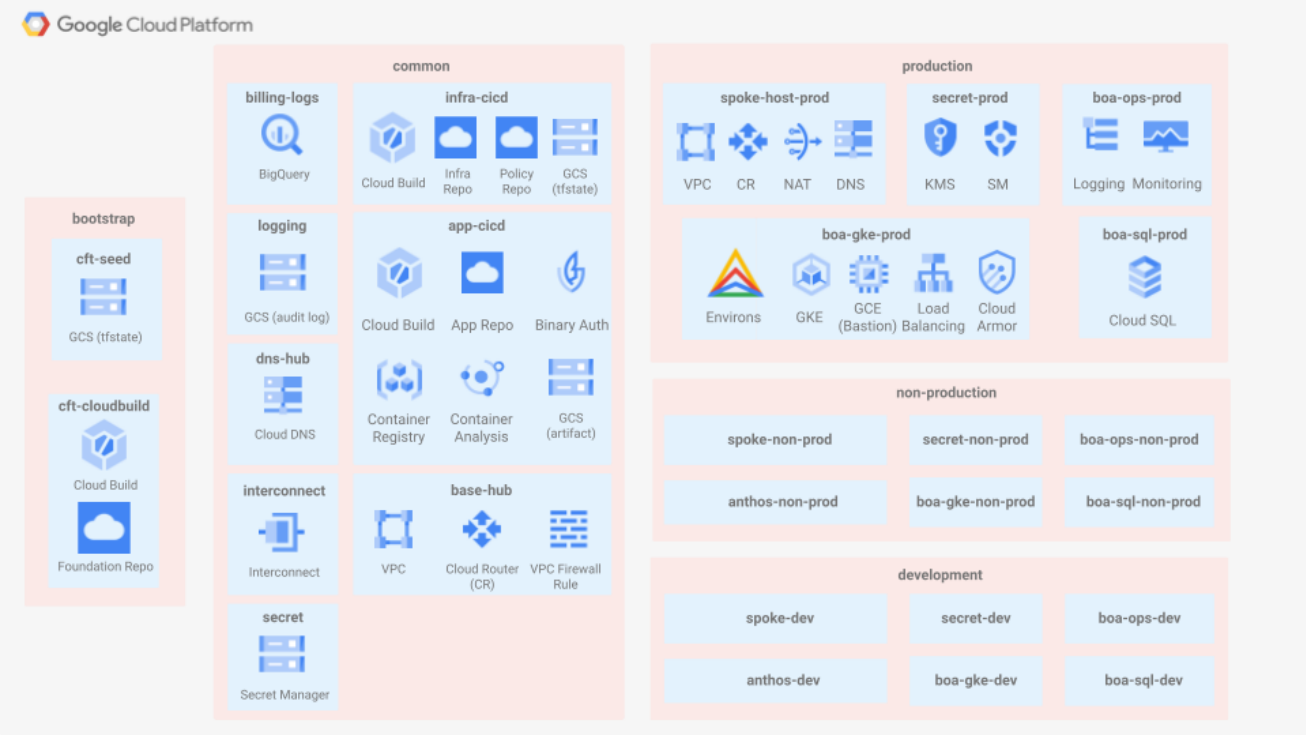

Figure 2.12.1 Bank of Anthos project structure on the secured foundation

The Bank of Anthos application adds several projects to the projects that are already defined for the foundation. The additional projects are for deploying resources, configurations, and applications, which are all necessary to support the Bank of Anthos, as shown in Table 2.12.1.

Folder	Project	Description
common	infra-cicd	Contains the infrastructure pipeline that's described in Section 5.7. This project also contains the Anthos Config Management policy repository that's used for GKE cluster configuration.
	app-cicd	Contains the application pipeline that's described in Section 5.7.
development, non-production, production	boa-gke	Contains the Bank of Anthos GKE clusters and MCI GKE cluster. Also used as the environ host project for Anthos.
	boa-sql	Contains the Cloud SQL for PostgreSQL databases that are used by the Bank of Anthos application.
	secret	Contains Secret Manager and KMS instances for secrets that are specific to the Bank of Anthos application.
	monitoring	Used for storing environment logs as well as monitoring the environment instance of the Bank of Anthos application.

Table 2.12.1 Additional Bank of Anthos projects

As shown in Figure 2.12.2, a total of three Google Kubernetes Engine (GKE) clusters are created in each environment to deploy the Bank of Anthos. Two clusters (gke-boa-region1-01 and gke-boa-region2) act as identical GKE private clusters in two different regions to provide multi-region resiliency. The third cluster (gke-mci-region1-01) acts as the config cluster for Multi-cluster Ingress (MCI).

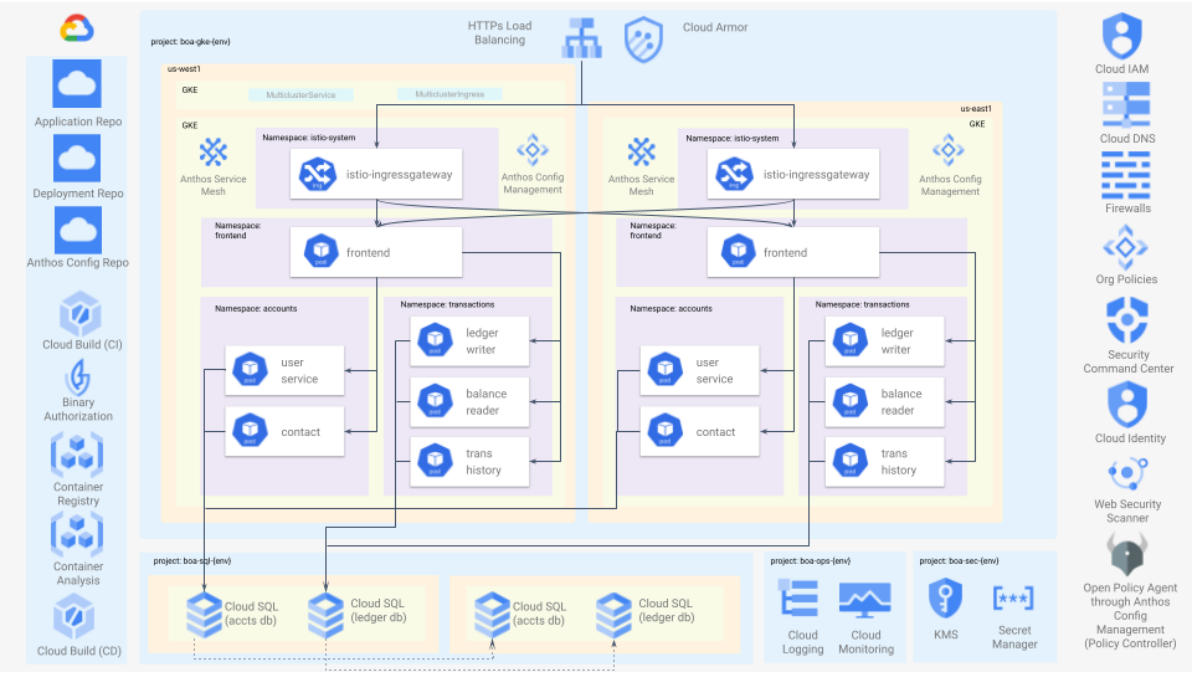

Figure 2.12.2 Detailed Bank of Anthos architecture

12.1.1) Bank of Anthos application components

The Bank of Anthos application example lets users create login accounts, log in to their account, see their transaction history, make deposits, and transfer money to other users' accounts. The application is composed of six services that are described in Table 2.12.2. The services run as containers that connect to each other over REST APIs and gRPC APIs.

Service	Language	Description
`frontend`	Python	Exposes an HTTP server to serve the website. Contains a login page, a signup page, and a home page.
`ledger-writer`	Java	Accepts and validates incoming transactions before writing them to the ledger.
`balance-reader`	Java	Provides an efficient readable cache of user balances, as read from `ledger-db`.
`transaction-history`	Java	Provides an efficient readable cache of past transactions, as read from `ledger-db`.
`user-service`	Python	Manages user accounts and authentication. The service signs JWTs that are used for authentication by other services.
`contacts`	Python	Stores a list of additional accounts that are associated with a user. These accounts are listed in the application's **Send Payment** and **Deposit** forms.

Table 2.12.2 Bank of Anthos software components

The application runs on Anthos, Google's managed hybrid multi-cloud application platform. This platform allows you to build, deliver, and manage the lifecycle of your applications. Table 2.12.3 details the Anthos components that are used in the deployment of the Bank of Anthos application.

Anthos component	Use
Anthos GKE	Container management
Anthos Configuration Management	Policy management and enforcement
Anthos Service Mesh	Service management
Cloud Operations	Observability and platform management
Binary Authorization	Container image attestation
Multi-cluster Ingress (MCI)	Multi-cluster Ingress controller for Anthos GKE clusters

Table 2.12.3 Anthos components deployed in the Bank of Anthos application

12.1.2) Distributed services and Anthos Service Mesh

In the Bank of Anthos application, Anthos Service Mesh adds security by providing encryption, mutual authentication, and authorization for all communications between workloads. For this deployment, the Anthos Service Mesh uses its own managed private certificate authority (Mesh CA) for issuing TLS certificates to authenticate peers and to help ensure that only authorized clients can access a service. Using mutual TLS (mTLS) for authentication also helps ensure that all TCP communications are encrypted in transit. For service ingress traffic into the mesh, the Bank of Anthos uses an Istio ingress gateway (`istio-ingressgateway`).

The Bank of Anthos application runs as a distributed service, which means that it runs as a Kubernetes Service on two or more Kubernetes clusters. The Kubernetes Service runs in the same namespace on each cluster and acts as a single logical service. A distributed service continues running even if one or more GKE clusters are down, as long as the healthy clusters are able to serve the load. In order to create a distributed service across clusters, Anthos Service Mesh provides L4/L7 connectivity between the services running on each cluster and enables them to act as a single logical service.

12.1.3) Bank of Anthos cluster protection

The secured application uses private GKE clusters to increase its security posture; neither the cluster nodes nor the control plane has a public endpoint. The clusters in the secured application are protected by VPC firewall rules and hierarchical firewall policies. As part of the secured application deployment, one Cloud NAT instance is used for each environment to give Pods a mechanism to access resources that have public IPs. Further protection of the cluster is provided by GKE Sandbox, which helps protect the host kernel of the nodes. In addition, the cluster uses Shielded GKE Nodes to limit the ability of an attacker to impersonate a node, and the nodes run Container-Optimized OS to limit their attack surface.

The web frontend of the Bank of Anthos application is exposed to the internet by using MCI. MCI creates a load balancer for the Bank of Anthos application using Cloud Load Balancing and also creates and manages Network Endpoint Groups (NEGs). The load balancer has the `istio-ingressgateway` service in both clusters as backends and the NEGs dynamically track the service endpoints of the two `istio-ingressgateway` services to ensure the load balancer has healthy backends. Cloud Load Balancing uses Google's network infrastructure to provide the Bank of Anthos frontend with an anycast IP address that enables low-latency access to the Bank of Anthos application and helps protect the application frontend from DDoS attacks. The web interface to the Bank of Anthos application is further protected against attacks through the use of Cloud Armor.

Cloud Domains is used to register the public domain (`boaongcp.cloud`) on which the secured application example operates. Cloud DNS is used to provide low-latency and high-availability DNS zone serving.

12.1.4) Bank of Anthos namespaces

The GKE clusters that are used by the Bank of Anthos application are segregated into four namespaces in order to provide separation of services. The traffic flows between namespaces are restricted through network policies. The namespaces are as follows:

- `istio-system`, which contains the Ingress gateway.
- `frontend`, which contains the Bank of Anthos frontend service.
- `accounts`, which contains Bank of Anthos user services.
- `transactions`, which contains Bank of Anthos transaction services.

All namespaces that participate in the service mesh are injected with the `istio.io/rev=REVISION` label. The label allows the resource controller to automatically inject the sidecar Envoy proxy into every Pod within the labeled namespace.

12.1.5) Bank of Anthos identity and access control

The services that constitute the Bank of Anthos application run in Pods that use Workload Identity, which provides each Pod with a unique identity that has only the minimal permissions necessary for the service to operate. Workload Identity lets a Kubernetes service account act as a Google service account by creating a secure mapping between the Kubernetes service account and the Google service account. Pods that run as the Kubernetes service account automatically authenticate as the Google service account when they access Google Cloud APIs because Workload Identity allows Identity and Access Management (IAM) to trust Kubernetes service account credentials.

Access to each GKE control plane is enabled through a bastion host, with one host in each environment. Each bastion is protected by Identity-Aware Proxy. Access to the GKE clusters is controlled by Google Groups for GKE-based Kubernetes role-based access control (RBAC). Groups let you control identities using a central identity management system that's controlled by identity administrators. Instead of updating the RBAC configuration whenever a change is needed for permissions, an administrator can modify group membership.

12.1.6) Bank of Anthos database structure

The Bank of Anthos application uses two PostgreSQL databases. One database (`ledger-db`) is used for transactions, and the other database (`accounts-db`) is used for user accounts. The databases are deployed using Google's managed database service, Cloud SQL for PostgreSQL. The databases are configured with cross-region replicas for disaster recovery, and they are encrypted using customer-managed encryption keys. Cloud SQL proxy is used to connect Bank of Anthos microservices to Cloud SQL for PostgreSQL by using service account authentication.

12.2) Deployment roles for the Bank of Anthos secured application

The complete application stack of the Bank of Anthos secured application, from foundation to software application, is deployed through a series of automated systems that are described in Section 5. These systems are intended to be operated by different groups, as shown in the model in Figure 2.5.3. Table 2.12.4 lists the groups, their roles, the deployment mechanism they use, and the resources that they are responsible for deploying.

Operator team	Role	Deployment systems	Resources deployed
Platform team	Responsible for centrally creating and managing the resources that are associated with the security foundation.	Foundation pipeline	Table 2.12.6
Infrastructure service team	Responsible for deploying managed services and configuring the application platform where applications are deployed.	Infrastructure pipeline, Anthos Config Management	Table 2.12.7, Table 2.12.8, Table 2.12.9
Application team	Responsible for developing application code.	Application pipeline	Table 2.12.2

Table 2.12.4 Operator teams, roles, and the corresponding deployment systems

Using automated pipelines to deploy the secured application stacks makes it possible to build security, auditability, traceability, repeatability, controllability, and compliance into the deployment process. Using different systems that have different permissions and putting different people into different operating groups creates a separation of responsibilities. This lets you follow the principle of least privilege.

12.2.1) Anthos Config Management

The secured application uses Anthos Config Management to manage GKE workload policies and resources. Anthos Config Management uses a Git repository that acts as the single source of truth for declared policies that are stored in a config. Configs are applied to the environments (`development`, `production`, and `non-production`) by using a branching strategy on the Git repository that stores the configs.

Anthos Config Management uses a declarative approach to policy and resource management. It continuously checks cluster state and applies the state that's declared in the config in order to enforce policies.

Anthos Config Management works in conjunction with Policy Controller. Policy Controller is a Kubernetes dynamic admission controller that checks, audits, and enforces your clusters' compliance with policies related to security, regulations, and business rules. Policy Controller enforces your clusters' compliance by using policies called *constraints*. Policy Controller is built from the Gatekeeper open source project.

12.3) Logging and monitoring

Cloud Operations for GKE is used to provide Cloud Logging and Cloud Monitoring services for the Bank of Anthos application. Cloud Operations for GKE provides predefined monitoring dashboards for GKE. Cloud Operations for GKE also allows you to collect system and application logs into central log buckets.

The secured application has a project in each environment folder that's used for storing logs and for a monitoring workspace. The security foundation has a separate logging project where the aggregate Cloud Audit Logs logs from across the entire Google Cloud organization are exported. The logging mechanism described in this section is specific to the secured application.

Security Command Center provides insight into the overall security posture of the secured application. Security Command Center provides the secured application with Container Threat Detection. This service continuously monitors the state of container images, evaluating all changes and remote access attempts to help detect runtime attacks in near real time. Configuration of Security Command Center and Container Threat Detection is described in Section 10.1.

The secured application uses Web Security Scanner to detect vulnerabilities in the Bank of Anthos application. It does this by crawling the application, following all links starting at the base URL (`boaongcp.cloud`). It then attempts to exercise as many user inputs and event handlers as possible.

12.4) Mapping BeyondProd security principles to the secured application

Google pioneered container-based architectures over 15 years ago when we moved to containers and container orchestration to achieve higher resource utilization, to build highly available applications, and to simplify work for Google developers. This container-based architecture required a different security mode.This security model is termed BeyondProd and encompasses several key security principles that map to the Bank of Anthos application architecture as shown in Table 2.12.5.

Security principle	Mapping to secured application architecture
Protection of the network at the edge	Cloud Load Balancing
	Cloud Armor
	VPC with private GKE clusters
	Firewall policy
No inherent mutual trust between services	Anthos Service Mesh
	Workload Identity
	Network Policy
Trusted machines running code with known provenance	Binary Authorization
	Shielded GKE Nodes
Choke points for consistent policy enforcement across services	Anthos Config Management
	Policy Controller
Simple, automated, and standardized change rollout	Foundation pipeline
	Infrastructure pipeline
	Application pipeline
	Anthos Config Management
Isolation between workloads that share an operating system	GKE Sandbox
	Container-Optimized OS

Table 2.12.5 Mapping BeyondProd principles to the secured application

12.5) Pipelines used to deploy Bank of Anthos architectural components

As mentioned at the beginning of Section 12, the Bank of Anthos application is deployed using a combination of the foundation pipeline, infrastructure pipeline, manual processes, and Anthos Config Management. Table 2.12.6, Table 2.12.7, Table 2.12.8, and Table 2.12.9 show which components are deployed using which methods.

Architectural component	Purpose
Project	Provides a logical boundary for Google Cloud resources and services. The boundary is used to segment the Bank of Anthos deployment.
Virtual Private Cloud network	Provides networking services to Compute Engine and GKE resources through regional subnets that define the IP address ranges for Bank of Anthos GKE clusters.
VPC firewall rules	Defines allow and deny rules that are applied to networking traffic to control access to the Bank of Anthos GKE clusters.
IAM roles	Defines permissions that are used within the projects by the Bank of Anthos.
Cloud APIs	Enables APIs to support the Bank of Anthos deployment.
Cloud DNS	Publishes the Bank of Anthos domain name to the global DNS.

Table 2.12.6 Components deployed by the foundation pipeline

Architectural component	Purpose
Google Kubernetes Engine	Provides hosting for the microservices of the containerized Bank of Anthos application.
Cloud SQL for PostgreSQL	Provides hosting for the data backend for the Bank of Anthos application.
Cloud Key Management	Provides a key encryption key for encryption based on customer-managed encryption keys (CMEK) for Cloud SQL for PostgreSQL, GKE, and Secret Manager.
Secret Manager	Provides a secret store for the RSA key pair that's used for JWT-based user authentication.
Compute Engine	Provides a bastion host to access the GKE control plane (API server) to bootstrap Anthos Config Management and Anthos Service Mesh.
Static external IP address	Provides a reserved IP address that MCI binds to a load balancer.
Cloud Armor	Provides a web-application firewall and DDoS protection.

Table 2.12.7 Components deployed by the infrastructure pipeline

Architectural component	Purpose
Anthos Config Management	Provides configuration management for the Bank of Anthos application. Anthos Config Management version 1.1 and higher include Policy Controller as one of its components.
Anthos Service Mesh	Provides service mesh capability to secure the communication (using mTLS) between microservices of the Bank of Anthos application.

Table 2.12.8 Components deployed through a manual bootstrap process from the bastion host

Architectural component	Purpose	Configuration area
VirtualService	Provides configuration that enables name-based routing and canary rollouts.	Istio custom resource
DestinationRule	Defines policies, load balancing, mTLS, and circuit breakers.	
AuthorizationPolicy	Defines access control on workloads in the service mesh.	
Service	Defines the virtual IP address/DNS name that's used to access the logical set of Pods.	Kubernetes workload definitions
Deployment	Provides a declarative template for Pods and replica sets.	
RBAC (Roles and Bindings)	Defines what authorization a Kubernetes service account has at the cluster level or namespace level.	Kubernetes identity and authorization
Workload Identity	Defines the Google Cloud service account that's used to access Google Cloud resources.	
Kubernetes Service Account	Defines an identity that's used by a Kubernetes Service.	
Namespace	Defines the logical clusters within the physical cluster.	
Policy Controller	Defines constraints that are used to enforce compliance on Kubernetes clusters.	Kubernetes hardening

Table 2.12.9 Components deployed by Anthos Config Management

12.6) Bank of Anthos resource IP address ranges

The Bank of Anthos secured application example requires multiple IP address ranges to be assigned in the `development`, `non-production`, and `production` environments. Each GKE cluster that's used by the Bank of Anthos needs separate IP address ranges allocated for the nodes, Pods, Services, and control plane. Both the Cloud SQL instances and bastion hosts also require separate IP address ranges. If you need to design your own IP address allocation scheme, you should ensure that you follow the GKE guides and recommendations.

The following tables show the spoke VPC subnets and IP address ranges that are used in the different environments to deploy the Bank of Anthos secured application example:

- `development` environment: Table 2.12.10
- `non-production` environment: Table 2.12.11
- `production` environment: Table 2.12.12

Resource/subnet	Primary IP range	Pod IP range	Service IP range	GKE control plane IP range
MCI GKE cluster / `subnet-usw1-01`	`10.0.64.0/29`	`100.64.64.0/22`	`100.64.68.0/26`	`100.64.70.0/28`
Application GKE cluster region 1 / `subnet-usw1-02`	`10.0.65.0/29`	`100.64.72.0/22`	`100.64.76.0/26`	`100.64.78.0/28`
Bastion host / `subnet-usw1-03`	`10.0.66.0/29`			
`development` application GKE cluster region 2 / `subnet-use4-01`	`10.1.64.0/29`	`100.65.64.0/22`	`100.65.68.0/26`	`100.65.70.0/28`
Cloud SQL	`10.16.64.0/24`			

Table 2.12.10 Bank of Anthos resource IP address ranges for the `development` environment

Resource/subnet	Primary IP range	Pod IP range	Service IP range	GKE control plane IP range
MCI GKE cluster / `subnet-usw1-01`	`10.0.128.0/29`	`100.64.128.0/22`	`100.64.132.0/26`	`100.64.134.0/28`
`non-production` application GKE cluster region 1 / `subnet-usw1-02`	`10.0.129.0/29`	`100.64.136.0/22`	`100.64.140.0/26`	`100.64.142.0/28`
`non-production` bastion host / `subnet-usw1-03`	`10.0.130.0/29`			
`non-production` application GKE cluster region 2 / `subnet-use4-01`	`10.1.128.0/29`	`100.65.128.0/22`	`100.65.132.0/26`	`100.65.134.0/28`
`non-production` Cloud SQL	`10.16.128.0/24`			

Table 2.12.11 Bank of Anthos resource IP address ranges for the `non-production` environment

Resource/subnet	Primary IP range	Pod IP range	Service IP range	GKE control plane IP range
`production` MCI GKE cluster / `subnet-usw1-01`	`10.0.192.0/29`	`100.64.192.0/22`	`100.64.196.0/26`	`100.64.198.0/28`
`production` App GKE cluster region 1 / `subnet-usw1-02`	`10.0.193.0/29`	`100.64.200.0/22`	`100.64.204.0/26`	`100.64.206.0/28`
Bastion host/ `subnet-usw1-03`	`10.0.194.0/29`			
App GKE cluster region 2/ `subnet-use4-01`	`10.1.192.0/29`	`100.65.192.0/22`	`100.65.196.0/26`	`100.65.198.0/28`
Cloud SQL instance	`10.16.192.0/24`			

Table 2.12.12 Bank of Anthos resource IP address ranges the production environment

13. General security guidance

The default role automatically assigned for Compute Engine service accounts is Editor, which is very broad. Instead of using these default service accounts, create new service accounts for your projects with tightly scoped permissions that are matched to their use cases. After you've confirmed that the new service accounts work for their respective applications, disable the default Compute Engine service accounts. You can find the default Compute Engine service account in the project-level Cloud IAM section of the Cloud Console. The service account name has the following format:

`project-number-compute@developer.gserviceaccount.com`

Similarly, if you use the App Engine service, create a new service account with tightly scoped permissions that are matched to your use case and override the default flow. After you've confirmed that the new service account works for your application, disable the App Engine default service account that was automatically created. You can find the default App Engine service account in the project-level Cloud IAM section of the Cloud Console. The service account name has the following format:

`your-project-id@appspot.gserviceaccount.com`

Each Kubernetes Engine node has a Cloud IAM service account associated with it. By default, nodes are given the Compute Engine default service account, which you can find by navigating to the Cloud IAM section of the Cloud Console. As previously discussed, this account has broad access by default, making it useful to a wide variety of applications, but it has more permissions than are required to run your Kubernetes Engine cluster. Instead, strongly consider creating and using a minimally privileged service account to run your Kubernetes Engine cluster. For more information about n hardening your Kubernetes Engine cluster, see the documentation for hardening your cluster's security and the Container Security Blog series.

14. Updates for the next version

The list of possible security controls you might care about for your specific business scenario can be quite broad. We recognize that this version of the security foundations blueprint doesn't cover everything. Future updates will include topics about data residency; data classification; and additional native guard rail services for security and compliance.

III. Summary

This guide provided our opinionated step-by-step guidance for configuring and deploying your Google Cloud estate. With the v2.5 version, we added guidance for restricting service and data location and for leveraging Assured Workloads when you are subject to regulatory compliance. We identified key decision points and areas of focus, and for each of those we provided both background considerations and discussions of the tradeoffs and motivations for each of the decisions we made. We recognize that these choices might not match every individual company's requirements and business context. You are free to adopt and modify the guidance we've provided.

We will continue to update this blueprint to stay current with new product capabilities, customer feedback, and the needs of and changes to the security landscape.

A core Google security principle covered in our BeyondProd paper is to use simple, automated, and standardized change rollout, which emphasizes the use of automation to limit exception cases and to minimize the need for human action. We therefore include a full Terraform repository of the scripts that you can use to automate the majority of the curated steps in this guide. You can run the scripts end-to-end to deploy the full opinionated foundation blueprint. The scripts can also be used and modified individually so that you can leverage parts of the blueprint that are the most relevant for your use case.

Finally, we can provide you with access to a demonstration Google organization using the repository of Terraform automation from start to finish. This gives you a view of what the full security foundations are like after they've been configured and are operational.

If you would like viewer access to the demonstration organization, please contact your Google Cloud sales team.

For support questions, or if you need a copy of the previous version of this document, contact us at security-foundations-blueprint-support@google.com.

For advisory and implementation services, Google Cloud is collaborating with Deloitte's industry-leading cyber practice to deliver end-to-end architecture, design, and deployment services to support your cloud security journey.