Best practices and reference architectures for VPC design

Last reviewed 2023-05-08 UTC

This guide introduces best practices and typical enterprise architectures for the design of virtual private clouds (VPCs) (/vpc/docs) with Google Cloud. This guide is for cloud network architects and system architects who are already familiar with Google Cloud networking concepts.

General principles and first steps

Best practices:
Identify decision makers, timelines, and pre-work (#prework).
Consider VPC network design early (#early).
Keep it simple (#simple).
Use clear naming conventions (#naming).

Identify decision makers, timelines, and pre-work

As a first step in your VPC network design, identify the decision makers, timelines, and pre-work necessary to ensure that you can address stakeholder requirements.

Stakeholders might include application owners, security architects, solution architects, and operations managers. The stakeholders themselves might change depending on whether you are planning your VPC network for a project, a line of business, or the entire organization.

Part of the pre-work is to get the team acquainted with concepts and terminology around VPC network design. Useful documents include the following:

- Resource Manager documentation (/resource-manager/docs)
- Cloud Identity and Access Management documentation (/iam/docs)
- Virtual Private Cloud documentation (/vpc/docs)

Consider VPC network design early

Make VPC network design an early part of designing your organizational setup in Google Cloud. It can be disruptive to your organization if you later need to change fundamental things such as how your network is segmented or where your workloads are located.

Different VPC network configurations can have significant implications for routing, scale, and security. Careful planning and deep understanding of your specific considerations helps you to create a solid architectural foundation for incremental workloads.

Keep it simple

Keeping the design of your VPC network topology simple is the best way to ensure a manageable, reliable, and well-understood architecture.

Use clear naming conventions

Make your naming conventions simple, intuitive, and consistent. This ensures that administrators and end users understand the purpose of each resource, where it is located, and how it is differentiated from other resources.

Commonly accepted abbreviations of long words help with brevity. Using familiar terminology where possible helps with readability.

Consider the components illustrated in the following example when establishing your naming conventions:

- **Company name**: Acme Company: `acmeco`
- **Business unit**: Human Resources: `hr`
- **Application code**: Compensation system: `comp`
- **Region code**: northamerica-northeast1: `na-ne1`, europe-west1: `eu-we1`
- **Environment codes**: `dev`, `test`, `uat`, `stage`, `prod`

In this example, the development environment for the human resources department's compensation system is named `acmeco-hr-comp-eu-we1-dev`.

For other common networking resources, consider patterns like these:

- **VPC network**
  syntax: `{company name}-{description(App or BU)-label}-{environment-label}-{seq#}`
  example: `acmeco-hr-dev-vpc-1`
- **Subnet**
  syntax: `{company-name}-{description(App or BU)-label}-{region/zone-label}`
  example: `acmeco-hr-na-ne1-dev-subnet`
- **Firewall rule**
  syntax: `{company-name}-{description(App or BU)-label}{source-label}-{dest-`

`label}-{protocol}-{port}-{action}`
example: `acmeco-hr-internet-internal-tcp-80-allow-rule`

- **IP route**
  syntax: `{priority}-{VPC-label}-{tag}-{next hop}`
  example: `1000-acmeco-hr-dev-vpc-1-int-gw`

Addresses and subnets

Best practices:
Use custom mode subnets in your enterprise VPC networks (#custom-mode).
Group applications into fewer subnets with larger address ranges (#fewer-subnets).

Use custom mode VPC networks

When you start your first project, you begin with the default network (/vpc/docs/vpc#default-network) , which is an auto mode VPC network (/vpc/docs/vpc#subnet-ranges) named `default`. Auto mode networks automatically create subnets and corresponding subnet routes (/vpc/docs/routes#subnet-routes) whose primary IP ranges are `/20` CIDRs in each Google Cloud region using a predictable set of RFC 1918 address ranges (/vpc/docs/subnets#ip-ranges). The `default` network also automatically includes some pre-populated firewall rules (/vpc/docs/firewalls#more_rules_default_vpc).

Though auto mode networks can be useful for early exploration, custom mode VPC networks are better suited for most production environments.

We recommend that enterprises use VPC networks in custom mode from the beginning for the following reasons:

- Custom mode VPC networks better integrate into existing IP address management schemes. Because all auto mode networks use the same set of internal IP ranges, auto mode IP ranges might overlap when connected with your on-premises corporate networks.
- You can't connect two auto mode VPC networks together using VPC Network Peering because their subnets use identical primary IP ranges.
- Auto mode subnets all have the same name as the network. You can choose unique, descriptive names for custom mode subnets, making your VPC networks more understandable and maintainable.
- When a new Google Cloud region is introduced, auto mode VPC networks automatically get a new subnet in that region. Custom mode VPC networks only get new subnets if you

specify them. This can be important for both sovereignty and IP address management reasons.

If it has no resources, you can delete (/vpc/docs/create-modify-vpc-networks#deleting_a_network) the `default` network. You cannot delete a VPC network until you have removed all resources, including Virtual Machine (VM) instances, that depend on it.

For more details of the differences between auto mode and custom mode VPC networks, see the VPC network overview (/vpc/docs/vpc).

Group applications into fewer subnets with larger address ranges

Conventionally, some enterprise networks are separated into many small address ranges for a variety of reasons. For example, this might have been done to identify or isolate an application or keep a small broadcast domain.

However, we recommend that you group applications of the same type into fewer, more manageable subnets with larger address ranges in the regions you want to operate.

Unlike other networking environments in which a subnet mask is used, Google Cloud uses a software-defined networking (SDN) approach to provide a full mesh of reachability between all VMs in the global VPC network. The number of subnets does not affect routing behaviour.

You can use service accounts or network tags to apply specific routing policies or firewall rules. Identity in Google Cloud is not based solely on the subnet IP address.

Some VPC features—including Cloud NAT (/nat/docs/overview), Private Google Access (/vpc/docs/configure-private-google-access), VPC Flow Logs (/vpc/docs/using-flow-logs), and alias IP ranges (/vpc/docs/alias-ip)—are configured per subnet. If you need more fine-grained control of these features, use additional subnets.

Single VPC network and Shared VPC

Best practices:

Start with a single VPC network for resources that have common requirements (#single-vpc).
Use Shared VPC for administration of multiple working groups (#shared-vpc).
Grant the network user role at the subnet level (#network-user).
Use a single host project if resources require multiple network interfaces (#single-host).
Use multiple host projects if resource requirements exceed the quota of a single project (#multiple-host-quota).
Use multiple host projects if you need separate administration policies for each VPC (#multiple-host-admin).

Start with a single VPC network for resources that have common requirement

For many simple use cases, a single VPC network provides the features that you need, while being easier to create, maintain, and understand than the more complex alternatives. By grouping resources with common requirements and characteristics into a single VPC network, you begin to establish the VPC network border as the perimeter for potential issues.

For an example of this configuration, see the single project, single VPC network (#single-project-single-vpc) reference architecture.

Factors that might lead you to create additional VPC networks include scale, network security, financial considerations, operational requirements, and identity and access management (IAM).

Use Shared VPC for administration of multiple working groups

For organizations with multiple teams, Shared VPC (/vpc/docs/shared-vpc) provides an effective tool to extend the architectural simplicity of a single VPC network across multiple working groups. The simplest approach is to deploy a single Shared VPC host project with a single Shared VPC network and then attach team service projects to the host project network.

In this configuration, 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 those projects can communicate with each other more 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.

For an example of this configuration, see the Single host project, multiple service projects, single Shared VPC (#single-host-project-multiple-service-projects-single-shared-vpc) reference architecture.

Grant the network user role at the subnet level

The centralized Shared VPC administrator can grant IAM members the network user (`networkUser` (/iam/docs/understanding-roles#compute-engine-roles)) role either at a subnet level, for fine-grained service-project authorization, or for all subnets at the host project level.

Following the principle of least privilege, we recommend granting the network user role at the subnet level to the associated user, service account, or group.

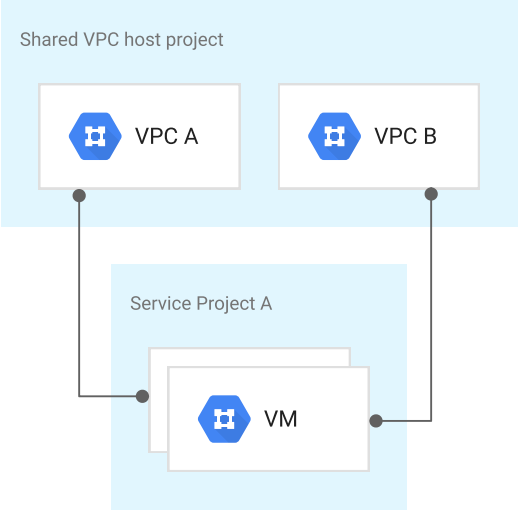
Because subnets are regional, this granular control allows you to specify which regions each service project can use to deploy resources.

With Shared VPC architectures, you also have the flexibility to deploy multiple Shared VPC host projects within your organization. Each Shared VPC host project can then accommodate a single or multiple Shared VPC networks. In this configuration, different environments can easily enforce different policy concerns.

For more information, see IAM roles for networking (/iam/docs/job-functions/networking).

Use a single host project if resources require multiple network interfaces

If you have a service project that needs to deploy resources to multiple isolated VPC networks—for example, VM instances with multiple network interfaces—your host project must contain all of the VPC networks that provide the services. This is because a service project is allowed to attach to only one host project.



Use multiple host projects if resource requirements exceed the quota of a sin project

In cases where the aggregate resource requirements of all VPC networks can't be met within a project's quota, use an architecture with multiple host projects with a single Shared VPC network per host project, rather than a single host project with multiple Shared VPC networks. It's important to evaluate your scale requirements, because using a single host project requires multiple VPC networks in the host project, and quotas are enforced at the project level.

For an example of this configuration, see the Multiple host projects, multiple service projects, multiple Shared VPC reference architecture (#multiple-host-project-multiple-service-projects-multiple-shared-vpc).

Use multiple host projects if you need separate administration policies for eac VPC network

Because each project has its own quota, use a separate Shared VPC host project for every VPC network to scale aggregate resources. This allows each VPC network to have separate IAM permissions for networking and security management, because IAM permissions are also implemented at the project level. For example, if you deploy two VPC networks (VPC network A and VPC network B) into the same host project, the network administrator (`networkAdmin` (/iam/docs/understanding-roles#compute-engine-roles)) role applies to both VPC networks.

Deciding whether to create multiple VPC networks

Best practices:
Create a single VPC network per project to map VPC network quotas to projects (#vpc-per-project).
Create a VPC network for each autonomous team, with shared services in a common VPC network (#shared-common-vpc).
Create VPC networks in different projects for independent IAM controls (#independent-iam).
Isolate sensitive data in its own VPC network (#isolate-data).

Create a single VPC network per project to map VPC resource quotas to proje

Quotas are constraints applied at the project or network level. All resources have an initial default quota meant to protect you from unexpected resource usage. However, many factors might lead you to want more quota. For most resources, you can request additional quota (/vpc/docs/quota#requesting-additional-quota).

We recommend creating a single VPC network per project if you expect to grow beyond the default VPC resource quotas. This makes it easier to map project-level quota increases to each VPC network rather than to a combination of VPC networks in the same project.

Limits are designed to protect system resources in aggregate. Limits generally can't be raised easily, although Google Cloud support and sales teams can work with you to increase some limits.

See VPC resource quotas and limits (/vpc/docs/quota#quotas_and_limits) for current values.

Google Support can increase some scaling limits, but there might be times when you need to build multiple VPC networks to meet your scaling requirements. If your VPC network has a requirement to scale beyond the limits, discuss your case with Google Cloud sales and support teams about the best approach for your requirements.

Create a VPC network for each autonomous team, with shared services in a common VPC network

Some large enterprise deployments involve autonomous teams that each require full control over their respective VPC networks. You can meet this requirement by creating a VPC network for each business unit, with shared services in a common VPC network (for example, analytic tools, CI/CD pipeline and build machines, DNS/Directory services). For more information about creating a common VPC network for shared services, see the shared services (#shared-service) section.

Create VPC networks in different projects for independent IAM controls

A VPC network is a project-level resource with fine-grained, project-level identity and access management (IAM) controls (/iam/docs/understanding-roles#compute-engine-roles), including the following roles:

- `networkAdmin`
- `securityAdmin`
- `networkUser`
- `networkViewer`

By default, IAM controls are deployed at the project level and each IAM role applies to all VPC networks within the project.

If you require independent IAM controls per VPC network, create your VPC networks in different projects.

If you require IAM roles scoped to specific Compute Engine resources such as VM instances, disks, and images, use IAM policies for Compute Engine resources (/compute/docs/access#resource-policies).

Isolate sensitive data in its own VPC network

For companies that deal with compliance initiatives, sensitive data, or highly regulated data that is bound by compliance standards such as HIPAA or PCI-DSS, further security measures often make sense. One method that can improve security and make it easier to prove compliance is to isolate each of these environments into its own VPC network.

Connecting multiple VPC networks

Best practices:
Choose the VPC connection method that meets your cost, performance, and security needs (#choose-method).
Use VPC Network Peering if you won't exceed resource limits (#vpc-peering-limits).
Use external routing if you don't need private IP address communication (#external-routing).

Use Cloud VPN to connect VPC networks that would otherwise exceed aggregate peering group limits (#cloud_vpn_aggregate_peering_limits).
Use Cloud Interconnect to control traffic between VPC networks through an on-premises device (#cloud_interconnect_on-premises_device).
Use multi-NIC virtual appliances to control traffic between VPC networks through a cloud device (#multi-nic).
Create a shared services VPC if multiple VPC networks need access to common resources but not each other (#shared-service).

Choose the VPC connection method that meets your cost, performance, and security needs

The next step after deciding to implement multiple VPC networks is connecting those VPC networks. VPC networks are isolated tenant spaces within Google's Andromeda (https://cloudplatform.googleblog.com/2017/11/Andromeda-2-1-reduces-GCPs-intra-zone-latency-by-40-percent.html)
SDN, but there are several ways that you can enable communication between them. The subsequent sections provide best practices for choosing a VPC connection method.

The advantages and disadvantages of each are summarized in the following table, and subsequent sections provide best practices for choosing a VPC connection method.

	Advantages	Disadvantages
VPC Network Peering (/vpc/docs/vpc-peering)	• Easy to configure. • Low management overhead. • High bandwidth. • Low egress charges (same as single VPC network). • Each VPC network maintains its own distributed firewall. • Each VPC network maintains its own IAM accounts and permissions.	• Non-transitive. • Scaling numbe *aggregate* grou networks. This number of VM internal forwar • Requires non-c space. • Static and dyn propagated. • Source tags ar accounts of th not propagate Network Peeri
External routing (/vpc/docs/routes) (public IP or NAT gateway)	• No configuration needed. • Full isolation between VPC networks. • Overlapping IP address space is possible. • High bandwidth.	• Egress charge same zone are other options, Network Peeri • VMs need to b external IP adc

	Advantages	Disadvantages
		• No firewalling addresses. • Static and dyn propagated. • Source tags ar accounts of th not honored by
Cloud VPN (/network-connectivity/docs/vpn)	• Cloud VPN enables transitive topologies for hub and spoke. • Scalable through ECMP. • 99.99% service availability SLA on HA VPN.	• Management c • Billed at intern • Slightly higher • Throughput lin tunnel. • Lower MTU be tunnel encaps • Source tags ar accounts of th lost across the

Cloud Interconnect (/network-connectivity/docs/interconnect) - Dedicated or Partner	• Supported SLAs based on redundancy in deployment:
	  • 99.99% (details (/network-connectivity/docs/interconnect/tutorials/dedicated-creating-9999-availability))
	  • 99.9% (details (/network-connectivity/docs/interconnect/tutorials/dedicated-creating-999-availability))
	• Each link of a Dedicated Interconnect is a 10 Gbps or 100 Gbps connection. Multiple interconnect links can be bundled to increase throughput, with a maximum of 8 x 10 Gbps circuits (80 Gbps), or 2 x 100 Gbps (200 Gbps) circuits for each Dedicated Interconnect connection.
	• Possible to use ECMP across multiple interconnects to increase throughput.
		• Additional cos charges for tra VPC networks Interconnect c
		• Traffic is not e
		• Additional late native solution
		• Additional hard path that can f

	Advantages	Disadvantages
Multiple network interfaces (/vpc/docs/create-use-multiple-interfaces) (Multi-NIC)	• Scalable through managed instance groups and ECMP routes across instances. • Individual VMs have [bandwidth limits] (/compute/docs/network-bandwidth).	• Limit of 8 inter • The internal pa Load Balancer traffic to [any i balancing/doc service-multi-r load balancers default networ of a VM.

Use VPC Network Peering if you won't exceed resource limits

We recommend using VPC Network Peering (/vpc/docs/vpc-peering) when you need to connect VPC networks and can't use a single Shared VPC, as long as the totals of the resources needed for all directly connected peers do not exceed the limits (/vpc/docs/using-vpc-peering#limits) on VM instances, number of peering connections, and internal forwarding rules.

VPC Network Peering enables two VPC networks to connect with each other internally over Google's SDN, whether or not they belong to the same project or the same organization. VPC Network Peering merges the control plane and flow propagation between each peer, allowing the same forwarding characteristics as if all the VMs were in the same VPC network. When VPC networks are peered, all subnets, alias IP ranges, and internal forwarding rules are accessible, and each VPC network maintains its own distributed firewall. VPC Network Peering is not transitive.

VPC Network Peering is the preferred method for connecting VPC networks for the following reasons:

- No gateway bottleneck—Traffic forwards across peers as if the VMs were in the same VPC network.
- Billing—There are no additional charges; you are billed as if the VMs were in the same VPC network.

Use external routing if you don't need private IP address communication

If you don't need private IP address communication, you can use external routing with external IP addresses or a NAT gateway.

When a VPC network is deployed, a route to Google's default internet gateway is provisioned with a priority of 1000. If this route exists, and a VM is given an external IP address, VMs can send outbound (egress) traffic through Google's internet gateway. You can also deploy services behind

one of Google's many public load-balancing offerings, which allows the services to be reached externally.

Externally addressed VMs communicate with each other privately over Google's backbone, regardless of region and Network Service Tiers (/network-tiers). Use Google Cloud firewall rules to control external inbound (ingress) traffic to your VMs.

External routing is a good option for scaling purposes, but it's important to understand how public routing affects costs. For details, see the Network pricing documentation (/vpc/network-pricing#externaliptraffic).

Use Cloud VPN to connect VPC networks that would otherwise exceed aggregate peering group limits

HA VPN provides a managed service to connect VPC networks by creating IPsec tunnels between sets of endpoints. If you configure your Cloud Routers with custom route advertisements, you can enable transitive routing across VPC networks and hub-and-spoke topologies as described later in this document. Using Network Connectivity Center, you can use HA VPN tunnels as a transit network between on-premises networks, as explained in the Cloud VPN documentation (/network-connectivity/docs/vpn/concepts/overview#vpn-as-data-transfer-network).

Cloud VPN does not support large MTU. For details, see MTU considerations (/network-connectivity/docs/vpn/concepts/mtu-considerations).

Use Cloud Interconnect to control traffic between VPC networks through an c premises device

Cloud Interconnect extends your on-premises network to Google's network through a highly available, low-latency connection. You can use Dedicated Interconnect to connect directly to Google or use Partner Interconnect to connect to Google through a supported service provider.

Dedicated Interconnect provides high-speed L2 service between Google and a colocation provider or on-premises location. This allows you to use on-premises routing equipment to route between VPC networks and use existing on-premises security and inspection services to filter all traffic between VPC networks. All traffic between the two VPC networks has extra latency because of an additional round trip through the on-premises system.

Partner Interconnect provides similar capabilities, as well as the ability to peer directly with a provider at L3 and have the provider route between VPC networks. This enables access to internal IP addresses across your on-premises network and Google Cloud VPC networks without a NAT device or VPN tunnel.

Because many enterprise security appliances can be used on Google Cloud using VMs with multiple network interfaces, using Cloud Interconnect to route traffic between VPC networks is

not necessary except in very few cases where all traffic has to flow through an on-premises appliance because of corporate policies.

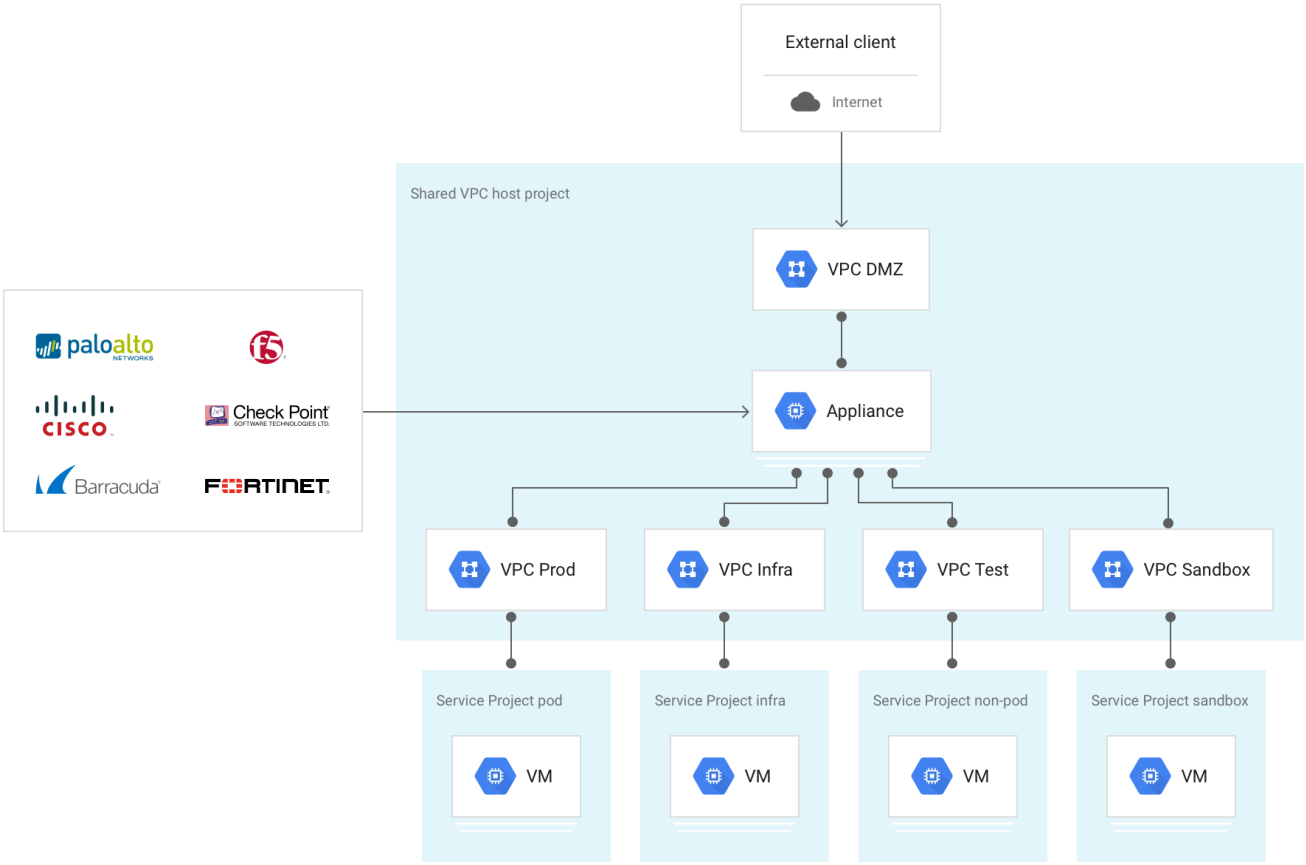
Use virtual appliances to control traffic between VPC networks through a clou device

Multiple network interface VMs are common for VPC networks that require additional security or services between them, because multiple network interface VMs enable VM instances to bridge communication between VPC networks.

A VM is allowed to have only one interface for each VPC network that it connects to. To deploy a VM into multiple VPC networks, you must have the appropriate IAM permission for each VPC network to which the VM connects.

Keep the following characteristics in mind when deploying a multi-NIC VM:

- A multi-NIC VM can have a maximum of 8 interfaces.
- The subnet IP ranges of the interfaces must not overlap.



Create a shared services VPC if multiple VPC networks need access to comm resources but not each other

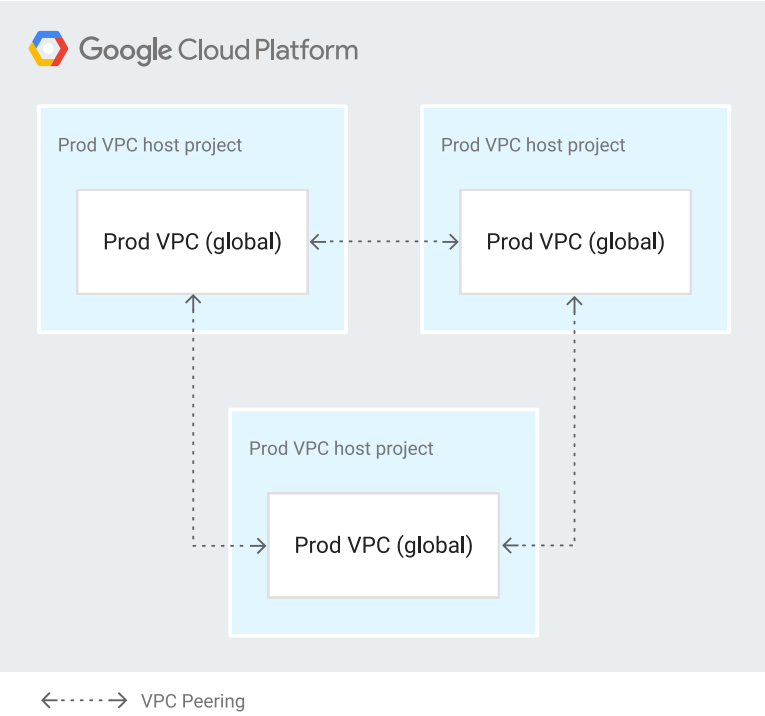
A VPC network provides a full mesh of global reachability. For this reason, shared services and continuous integration pipelines residing in the same VPC network don't require special consideration when it comes to connectivity—they are inherently reachable. Shared VPC (/vpc/docs/shared-vpc) extends this concept, allowing shared services to reside in an isolated project while providing connectivity to other services or consumers.

The approaches to shared services include Private Service Connect, VPC Network Peering, and Cloud VPN. Use Private Service Connect unless you aren't able to for some reason. You can use VPC Network Peering to connect to a shared services VPC network if you won't exceed aggregate resource limits and don't want to set up endpoints for every service. You can also use Cloud VPN to connect shared service VPC networks that would otherwise exceed aggregate peering group limits.

Private Service Connect lets you make a service hosted in one network available to VMs in another network by creating a special endpoint in the consumer network. The Private Service Connect configuration then channels traffic for that service between the two networks.

Consumers can use their own internal IP addresses to access services without leaving their VPC networks or using external IP addresses. Traffic remains entirely within Google Cloud. Private Service Connect provides service-oriented access between consumers and producers with granular control over how services are accessed.

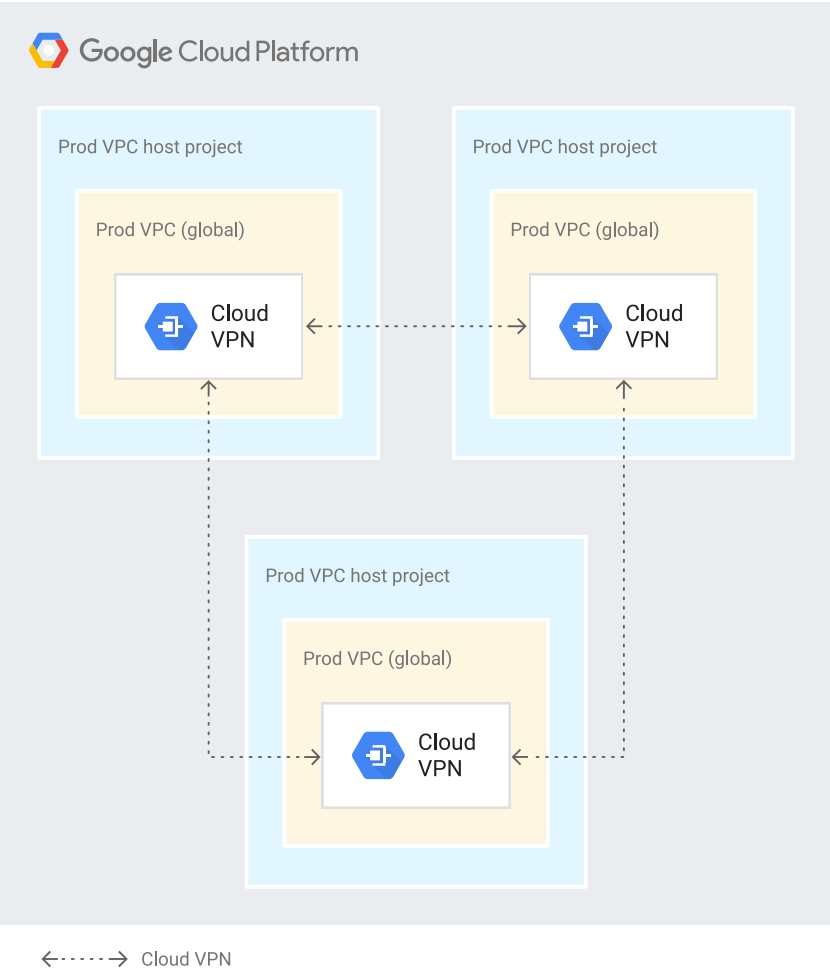
In the VPC Network Peering model, each VPC network creates a peering relationship with a common shared services VPC network to provide reachability. VPC Network Peering introduces scaling considerations, because scaling limits apply to the aggregate resource use of all peers.



VPC Network Peering can also be used in conjunction with private service access (/service-infrastructure/docs/enabling-private-services-access) and the Service Networking API

(/service-infrastructure/docs/service-networking/getting-started). Using the Service Networking API, you can let your customers in the same organization or another organization make use of a service you provide, but let them choose the IP address range that gets connected using VPC Network Peering.

Cloud VPN is another alternative. Because Cloud VPN establishes reachability through managed IPsec tunnels, it doesn't have the aggregate limits of VPC Network Peering. Cloud VPN uses a VPN Gateway for connectivity and doesn't consider the aggregate resource use of the IPsec peer. The drawbacks of Cloud VPN include increased costs (VPN tunnels and traffic egress), management overhead required to maintain tunnels, and the performance overhead of IPsec.



Hybrid design: connecting an on-premises environment

Best practices:

Use dynamic routing when possible (#dynamic-routing).

Use a connectivity VPC network to scale a hub-and-spoke architecture with multiple VPC networks (#scale-hub-spoke).

After you have identified the need for hybrid connectivity (/hybrid-connectivity) and have chosen a solution that meets your bandwidth, performance, and security requirements, consider how to integrate it into your VPC design.

Using Shared VPC (/vpc/docs/shared-vpc) alleviates the need for each project to replicate the same solution. For example, when you integrate a Cloud Interconnect solution into a Shared VPC, all VMs—regardless of region or service project—can access the Cloud Interconnect connection.

Dynamic routing

Dynamic routing is available on all hybrid solutions, including HA VPN, Classic VPN, Dedicated Interconnect, and Partner Interconnect. Dynamic routing uses Google's Cloud Router as a Border Gateway Protocol (BGP) speaker to provide dynamic External BGP (eBGP) routing.

The Cloud Router is not in the data plane; it only creates routes in the SDN.

Dynamic routing does not use tags, and the Cloud Router never re-advertises learned prefixes.

You can enable either of the Cloud Router's two modes, regional or global, on each VPC network. If you choose regional routing, the Cloud Router only advertises subnets that co-reside in the region where the Cloud Router is deployed. Global routing, on the other hand, advertises all subnets of the given VPC network, regardless of region, but does penalize routes that are advertised and learned outside of the region. This maintains symmetry within the region by always preferring a local interconnect, and is calculated by adding a penalty metric (MED) equal to 200 + TTL in milliseconds between regions.

Static routing

Static routing is only available on Classic VPN. Static routing offers the ability to set a next-hop route pointing at a Cloud VPN tunnel.

By default, a static route applies to all VMs in the network regardless of region. Static routing also lets network administrators selectively set which VMs the route applies to by using instance tags, which can be specified when you create a route.

Static routes apply globally within the VPC network, with the same route priority as each other. Therefore, if you have multiple tunnels in multiple regions to the same prefix with the same priority, a VM will use 5-tuple hash-based ECMP across all tunnels. To optimize this setup, you can create a preferred in-region route by referencing instance tags for each region and creating preferred routes accordingly.

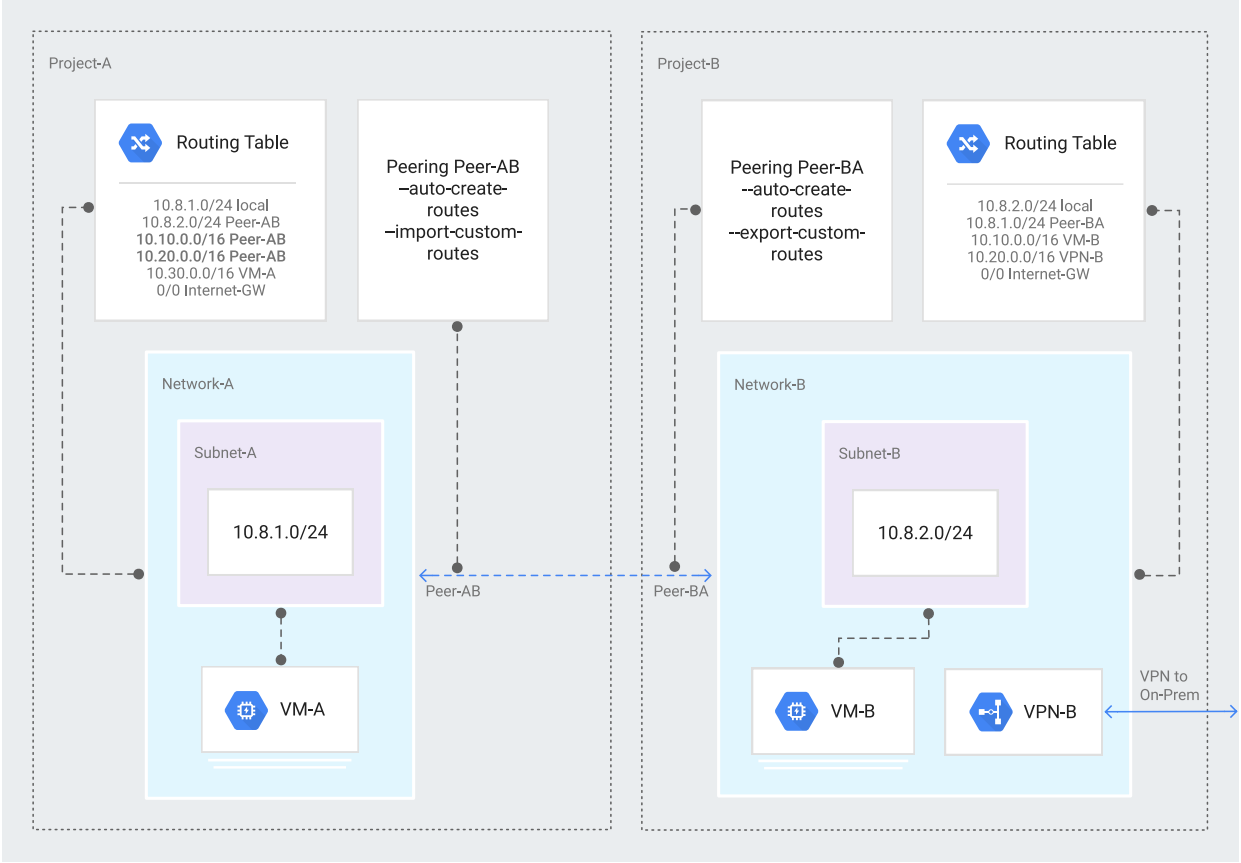
If you don't want outbound (egress) traffic to go through Google's default internet gateway, you can set a preferred default static route to send all traffic back on-premises through a tunnel.

Use a connectivity VPC network to scale a hub-and-spoke architecture with multiple VPC networks

If you need to scale a hub-and-spoke architecture with multiple VPC networks, configure a centralized hybrid connectivity in a dedicated VPC network and peer to other projects using custom advertised routes. This allows static or dynamically learned routes to be exported to peer VPC networks, to provide centralized configuration and scale to your VPC network design.

This is in contrast to conventional hybrid connectivity deployment, which uses VPN tunnel or VLAN attachments in each individual VPC network.

The following diagram illustrates centralized hybrid connectivity with VPC Network Peering custom routes:



Network security

Best practices:

Identify clear security objectives (#security-objectives).

Limit external access (#limit-access).

Define service perimeters for sensitive data (#service-perimeters).

Google Cloud provides robust security features across its infrastructure and services, from the physical security of data centers and custom security hardware to dedicated teams of researchers. However, securing your Google Cloud resources is a shared responsibility. You must take appropriate measures to help ensure that your apps and data are protected.

Identify clear security objectives

Before evaluating either cloud-native or cloud-capable security controls, start with a set of clear security objectives that all stakeholders agree to as a fundamental part of the product. These objectives should emphasize achievability, documentation, and iteration, so that they can be referenced and improved throughout development.

Limit external access

When you create a Google Cloud resource that uses a VPC network, you choose a network and subnet where the resource reside. The resource is assigned an internal IP address from one of the IP ranges associated with the subnet. Resources in a VPC network can communicate among themselves through internal IP addresses if firewall rules permit.

Limit access to the internet to only those resources that need it. Resources with only a private, internal IP address can still access many Google APIs and services through Private Service Connect or Private Google Access. Private access enables resources to interact with key Google and Google Cloud services while remaining isolated from the public internet.

Additionally, use organizational policies (/resource-manager/docs/organization-policy/org-policy-constraints) to further restrict which resources are allowed to use external IP addresses.

Before blocking internet access, however, consider the impact on your VM instances. Blocking internet access can reduce your risk of data exfiltration, but it can also block legitimate traffic, including essential traffic for software updates and third-party APIs and services. Without internet access, you are only able to access your VM instances through an on-premises network connected through a Cloud VPN tunnel, an Cloud Interconnect connection, or Identity-Aware Proxy. Using Cloud NAT, virtual machines can initiate egress connections to the internet for specific essential traffic without exposing public ingress connections.

Define service perimeters for sensitive data

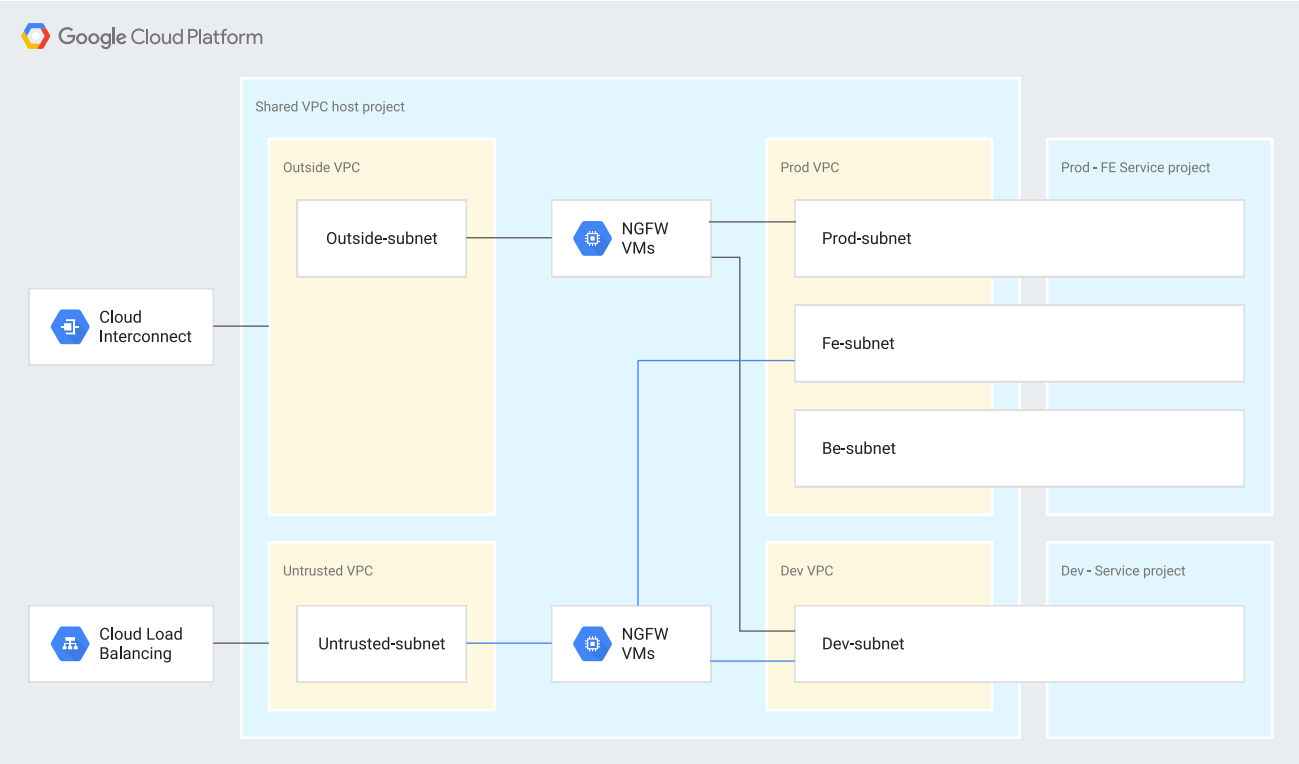
For workloads involving sensitive data, use VPC Service Controls (/vpc-service-controls/docs) to configure service perimeters around your VPC resources and Google-managed services and control the movement of data across the perimeter boundary. Using VPC Service Controls, you can group projects and your on-premises network into a single perimeter that prevents data access through Google-managed services. Service perimeters can't contain projects from different organizations, but you can use perimeter bridges to allow projects and services in different service perimeters to communicate.

Manage traffic with Google Cloud native firewall rules when possible

Google Cloud VPC includes an L3/L4 stateful firewall that is horizontally scalable and applied to each VM in a distributed manner. This firewall is configured using Hierarchical firewall policies, global and regional network firewall policies, and VPC firewall rules. See the Cloud Firewall overview (/vpc/docs/about-firewalls) for details.

Google Cloud Marketplace features a large ecosystem of third-party solutions, including VMs that do the following: provide advanced security, such as protection from information leakage, application exploits, and escalation of privileges; detect known and unknown threats; and apply URL filtering. There are also operational benefits to having a single vendor implement policy across cloud service providers and on-premises environments.

Traffic is typically routed to these VMs by specifying routes, either with the same priority (to distribute the traffic using a 5-tuple hash) or with different priorities (to create a redundant path), as shown in the multiple paths to the Dev-subnet in the following diagram.



Most solutions require multiple network interface VMs. Because a VM can't have more than one interface per VPC network, when you create a multiple network interface VM, each interface must attach to a separate VPC network.

Scale is also an important consideration when deploying third-party solutions into your VPC network for the following reasons:

- Limits: Most VM-based appliances must be inserted into the data path. This requires a multiple network interface VM that bridges multiple VPC networks that reside in the same project. Because VPC resource quotas are set at the project level, the aggregate resource needs across all VPC networks can become limiting.
- Performance: Introducing a single VM-based chokepoint into the fully horizontal scalability attributes of a VPC network goes against cloud design methodologies. To mitigate this, you can place multiple network virtual appliances (NVAs) into a managed instance group behind an internal passthrough Network Load Balancer.

To account for these factors in high-scale requirement architectures, push security controls to your endpoints. Start by hardening your VMs and using Google Cloud firewall rules. This strategy can also involve introducing host-based endpoint inspection agents that do not change the forwarding architecture of your VPC network through multiple network interface VMs.

For an additional example of this configuration, see the Stateful L7 firewall between VPC networks reference architecture (#l7).

Use fewer, broader firewall rule sets when possible

Only a certain number of rules can be programmed on any VM. However, you can combine many rules into one complex rule definition. For example, if all VMs in the VPC network need to explicitly allow 10 ingress TCP ports, you have two options: write 10 separate rules, each defining a single port, or define a single rule that includes all 10 ports. Defining a single rule that includes all 10 ports is the more efficient option.

Create a generic rule set that applies to the entire VPC network, and then use more specific rule sets for smaller groupings of VMs using targets (/vpc/docs/firewalls#rule_assignment). In other words, start by defining broad rules, and progressively define rules more narrowly as needed:

1. Apply firewall rules that are common across all VMs in the VPC network.
2. Apply firewall rules that can be grouped across several VMs, like a service instance group or subnet.
3. Apply firewall rules to individual VMs, such as a NAT gateway or bastion host.

Isolate VMs using service accounts when possible

Many organizations have environments that require specific rules for a subset of the VMs in a VPC network. There are two common approaches that you can take in these cases: subnet isolation and target filtering.

Subnet isolation

With subnet isolation, the subnet (/vpc/docs/vpc#subnets_vs_subnetworks) forms the security boundary across which Google Cloud firewall rules are applied. This approach is common in on-premises networking constructs and in cases where IP addresses and network placement form part of the VM identity.

You can identify the VMs on a specific subnet by applying a unique Tag (/vpc/docs/use-tags-for-firewalls), network tag, or service account to those instances. This allows you to create firewall rules that only apply to the VMs in a subnet—those with the associated Tag, network tag, service account. For example, to create a firewall rule that permits all communication between VMs in the same subnet, you can use the following rule configuration on the **Firewall rules page** (/vpc/docs/using-firewalls):

- **Targets**: **Specified target tags**
- **Target tags**: **subnet-1**
- **Source filter**: **Subnets**
- **Subnets**: Select subnet by name (example: **subnet-1**).

Target filtering

With target filtering, all VMs either reside on the same subnet or are part of an arbitrary set of subnets. With this approach, subnet membership is not considered part of the instance identity for firewall rules. Instead, you can use Tags, network tags, or service accounts to restrict access between VMs in the same subnet. Each group of VMs that uses the same firewall rules has the same network tag applied.

To illustrate this, consider a three-tier (web, app, database) application for which all of the instances are deployed in the same subnet. The web tier can communicate with end users and the app tier, and the app tier can communicate with the database tier, but no other communication between tiers is allowed. The instances running the web tier have a network tag of `web`, the instances running the app tier have a network tag of `app`, and the instances running the database tier have a network tag of `db`.

The following firewall rules implement this approach:

Rule 1: Permit end-users → web tier	**Targets**: **Specified target tags** **Target tags**: **web**

	Source filter: **IP ranges** **Source IP ranges**: **0.0.0.0/0**
Rule 2: Permit web tier → app tier	**Targets**: **Specified target tags** **Target tags**: **app** **Source filter**: **Source tags** **Source tags**: **web**
Rule 3: Permit app tier → database tier	**Targets**: **Specified target tags** **Target tags**: **db** **Source filter**: **Source tags** **Source tags**: **app**

However, even though it is possible to use network tags for target filtering in this manner, we recommend that you use Tags or service accounts where possible. Target tags are not access-controlled and can be changed by someone with the `instanceAdmin` role while VMs are in service. Tags and service accounts are access-controlled, meaning that a specific user must be explicitly authorized to use a service account. There can only be one service account per instance, whereas there can be multiple tags. Also, service accounts assigned to a VM can only be changed when the VM is stopped.

Use automation to monitor security policies when using tags

If you use network tags, remember that an instance administrator can change those tags. This can circumvent security policy. Therefore, if you do use network tags in a production environment, use an automation framework to help you overcome the lack of IAM governance over the network tags.

Use additional tools to help secure and protect your apps

In addition to firewall rules, use these additional tools to help secure and protect your apps:

- Use a Google Cloud global HTTP(S) load balancer (/load-balancing/docs/load-balancing-overview) to support high availability and protocol normalization.
- Integrate Google Cloud Armor (/armor) with the HTTP(S) load balancer to provide DDoS protection and the ability to block or allow IP addresses at the network edge.
- Control access to apps by using IAP (/iap) (IAP) to verify user identity and the context of the request to determine if a user should be granted access.
- Provide a single interface for security insights, anomaly detection, and vulnerability detection with Security Command Center (/security-command-center).

Network services: NAT and DNS

Best practices:
Use fixed external IP addresses with Cloud NAT (#use_fixed_outbound_ip_addresses_with_cloud_nat).
Use Private DNS zones for name resolution (#private-dns).

Use fixed external IP addresses with Cloud NAT

If you need fixed external IP addresses from a range of VMs, use Cloud NAT (/nat/docs). An example of why you might need fixed outbound IP addresses is the case in which a third party allows requests from specific external IP addresses. Using Cloud NAT allows you to have a small number of NAT IP addresses for each region that are used for outbound communications.

Cloud NAT also allows your VM instances to communicate across the internet without having their own external IP addresses. Google Cloud firewall rules are stateful. This means that if a connection is allowed between a source and a target or a target and a destination, then all subsequent traffic in either direction will be allowed as long as the connection is active. In other words, firewall rules allow bidirectional communication after a session is established.

Use private DNS zones for name resolution

Use private zones on Cloud DNS (/dns/docs/overview#concepts) to allow your services to be resolved with DNS within your VPC network using their internal IP addresses without exposing this mapping to the outside.

Use split horizon DNS (/dns/docs/overview#split-horizon) to map services to different IP addresses from within the VPC network than from the outside. For example, you can have a service exposed through network load balancing from the public internet, but have internal load balancing provide the same service using the same DNS name from within the VPC network.

API access for Google managed services

Best practices:
Use the default internet gateway where possible (#default-gateway).
Add explicit routes for Google APIs if you need to modify the default route (#explicit-routes).
Deploy instances that use Google APIs on the same subnet (#deploy-same-subnet).

Use the default internet gateway where possible

Access from resources within the VPC network to Google APIs follows the default internet gateway next-hop. Despite the next-hop gateway's name, the traffic path from instances to the Google APIs remains within Google's network.

By default, only VM instances with an external IP address can communicate with Google APIs and services. If you need access from instances without an external IP address, set up Private Service Connect endpoints or use the Private Google Access feature for each subnet. This doesn't slow down communications for Google APIs.

Google Kubernetes Engine (GKE) automatically enables Private Google Access on subnets where nodes are deployed. All nodes on these subnets without an external IP address are able to access Google managed services.

Add explicit routes for Google APIs if you need to modify the default route

If you need to modify the default route, then add explicit routes for Google API destination IP ranges.

In environments where the default route (`0.0.0.0/0`) doesn't use the default internet gateway next-hop, configure explicit routes for the destination IP address ranges (/vpc/docs/configure-private-google-access#config-routing-custom) used by Google APIs. Set the next-hop of the explicit routes to the default internet gateway. An example of such a scenario is when you need to inspect all traffic through an on-premises device.

Deploy instances that use Google APIs on the same subnet

Deploy instances that require access to Google APIs and services on the same subnet and enable Private Google Access for instances without external IP addresses. Alternatively, set up Private Service Connect endpoints.

If you are accessing Google APIs from your on-premises environment using Private Google Access, use Configuring Private Google Access for on-premises hosts (/vpc/docs/configure-private-google-access-hybrid) to access some Google services over private IP address ranges. Check to see which services are supported (/vpc/docs/private-google-access-hybrid#supported-services-onprem) before activating this feature, because access to other Google APIs through the IP addresses provided by this service will be unreachable. Activating this feature can require additional DNS configuration, such as configuring DNS Views.

If you are accessing Google APIs from your on-premises environment using Private Service Connect endpoints, see Access the endpoint from on-premises hosts (/vpc/docs/configure-private-service-connect-apis#on-premises) for details.

Logging, monitoring, and visibility

Best practices:

Tailor logging for specific use cases and intended audiences (#tailor-logging).

Increase the log aggregation interval for VPC networks with long connections (#log-aggregation-interval).

Use VPC Flow Logs sampling to reduce volume (#vpc-flow-log-sampling).

Remove additional metadata when you only need IP and port data (#remove-metadata).

Tailor logging for specific use cases and intended audiences

Use VPC Flow Logs (/vpc/docs/using-flow-logs) for network monitoring, forensics, real-time security analysis, and expense optimization. You can enable or disable VPC Flow Logs at the subnet level. If VPC Flow Logs are enabled for a subnet, it collects data from all VM instances in that subnet.

These logs record a sample of network flows that VM instances send and receive. Your logging use cases help to determine which subnets you decide require logging, and for how long.

Flow logs are aggregated by connection at 5-second intervals from Compute Engine VMs and then exported in real time. You can view flow logs in Cloud Logging and export them to any destination that Cloud Logging export supports.

The logs ingestion page in Logging tracks the volume of logs in your project and lets you disable all logs ingestion or exclude (discard) log entries you're not interested in, so that you can minimize any charges for logs over your monthly allotment.

Logs are a critical part of both operational and security success, but they aren't useful unless you review them and take action. Tailor logs for their intended audience, which helps to ensure operational and security success for your VPC networks.

For details, see Using VPC Flow Logs (/vpc/docs/using-flow-logs).

Increase log aggregation interval for VPC networks with long connections

For VPC networks with mostly long-lived connections, set the log aggregation interval to 15 minutes to greatly reduce the number of logs generated and to enable quicker and simpler analysis.

An example workflow for which increasing the log aggregation interval is appropriate is network monitoring, which involves the following tasks:

- Performing network diagnosis
- Filtering the flow logs by VMs and by applications to understand traffic changes

- Analyzing traffic growth to forecast capacity

Use VPC Flow Logs sampling to reduce volume

Use VPC Flow Logs sampling to reduce the volume of VPC Flow Logs, but still be able to see low-level samples and aggregated views.

An example workflow for which using sampling to reduce volume is appropriate is understanding network usage and optimizing network traffic expense. This workflow involves the following tasks:

- Estimating traffic between regions and zones
- Estimating traffic to specific countries on the internet
- Identifying top talkers

Remove additional metadata when you only need IP and port data

In security use cases where you are only interested in IP addresses and ports, remove the additional metadata to reduce the volume of data consumed in Cloud Logging.

An example workflow for which removing metadata is appropriate is network forensics, which involves the following tasks:

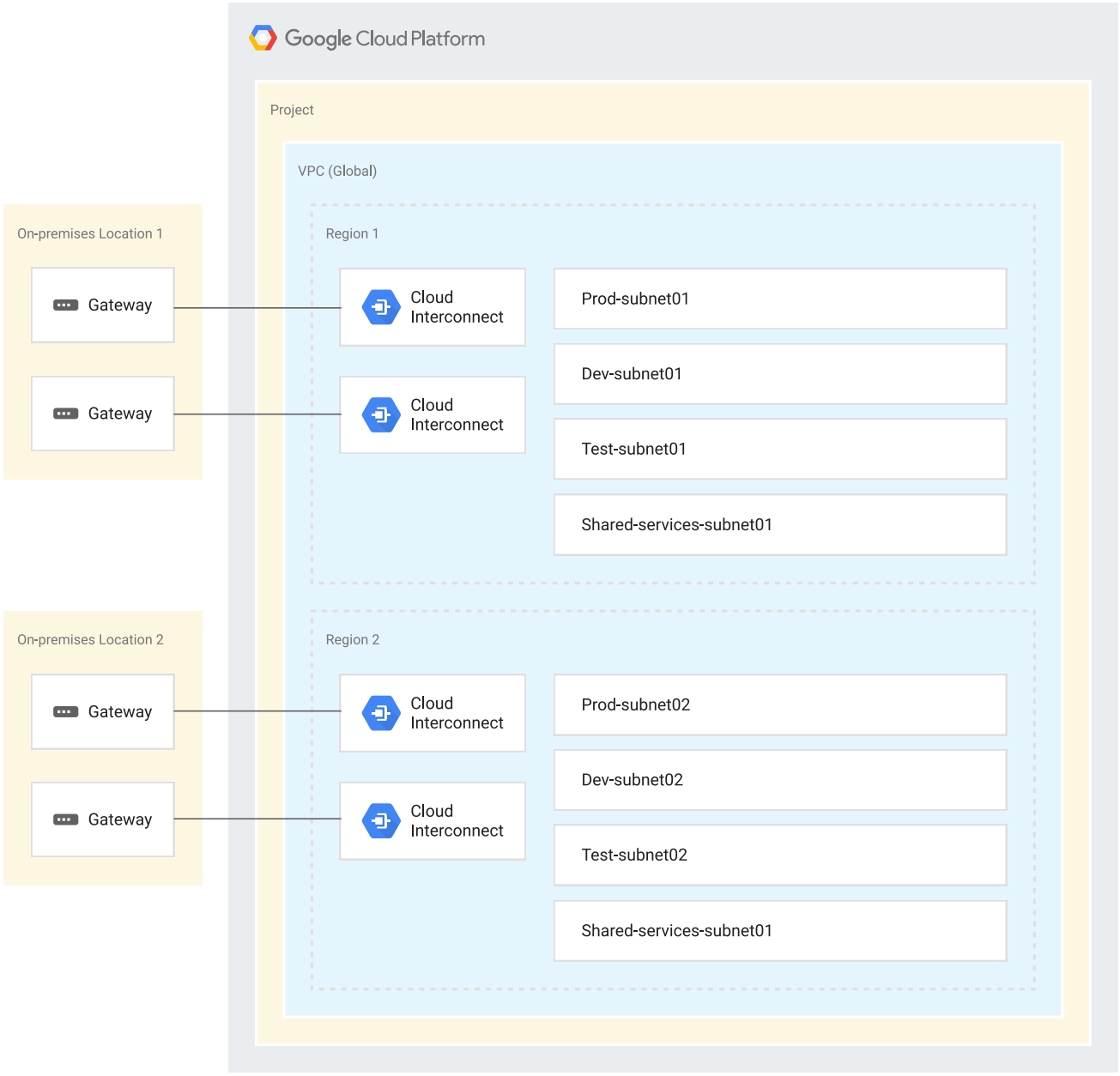
- Determining which IPs talked with whom and when
- Identifying any compromised IP addresses, found by analyzing network flows

Reference architectures

This section highlights a few architectures that illustrate some of the best practices in this document.

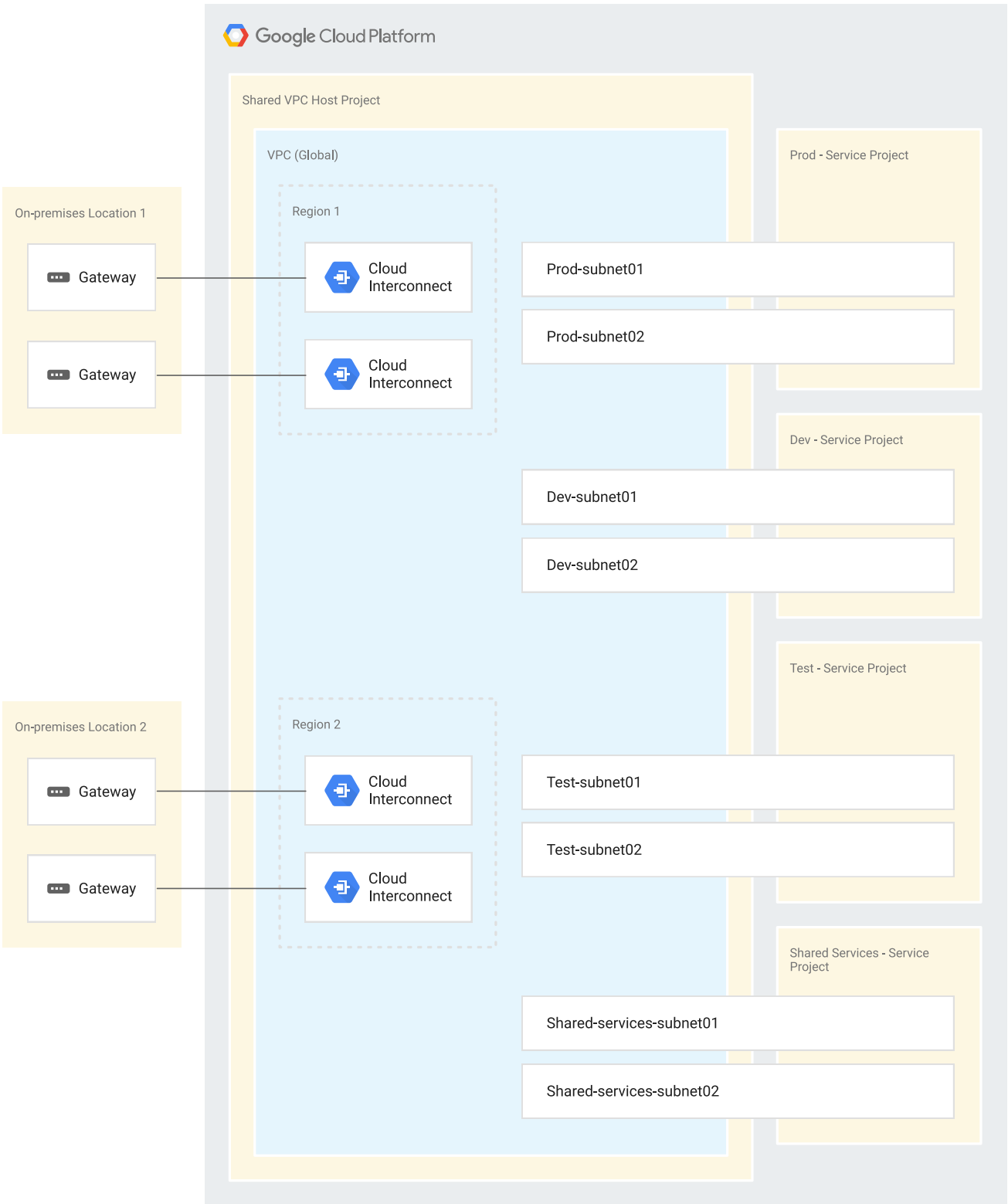
Single project, single VPC network

This initial reference architecture includes all of the components necessary to deploy highly available architectures across multiple regions, with subnet-level isolation and a 99.99% SLA connecting to your on-premises data centers.



Single host project, multiple service projects, single Shared VPC

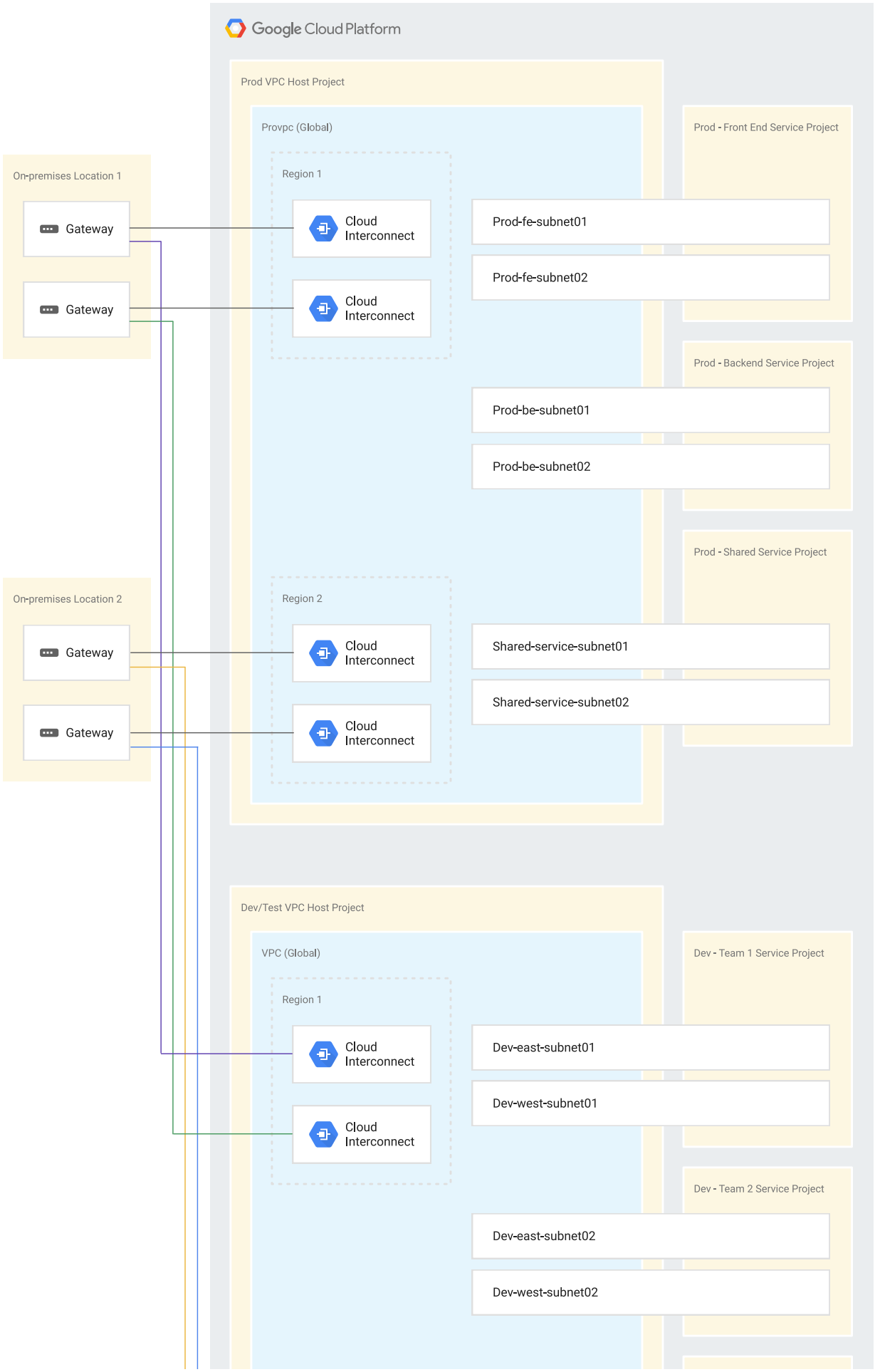
Building on the initial reference architecture, Shared VPC host projects and multiple service projects let administrators delegate administrative responsibilities—such as creating and managing instances—to Service Project Admins while maintaining centralized control over network resources like subnets, routes, and firewalls.

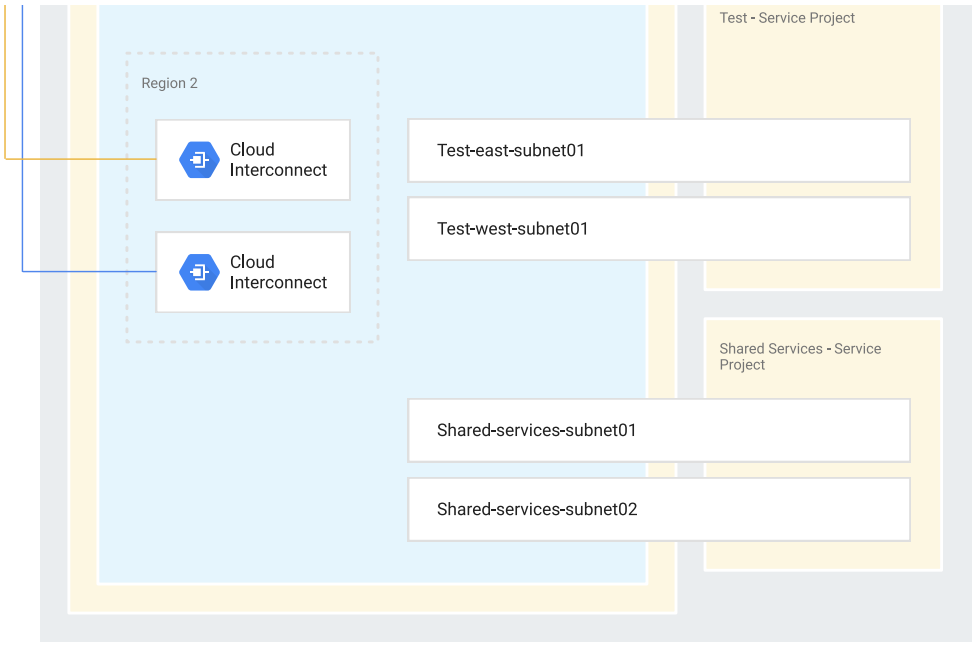


Multiple host projects, multiple service projects, multiple Shared VPC

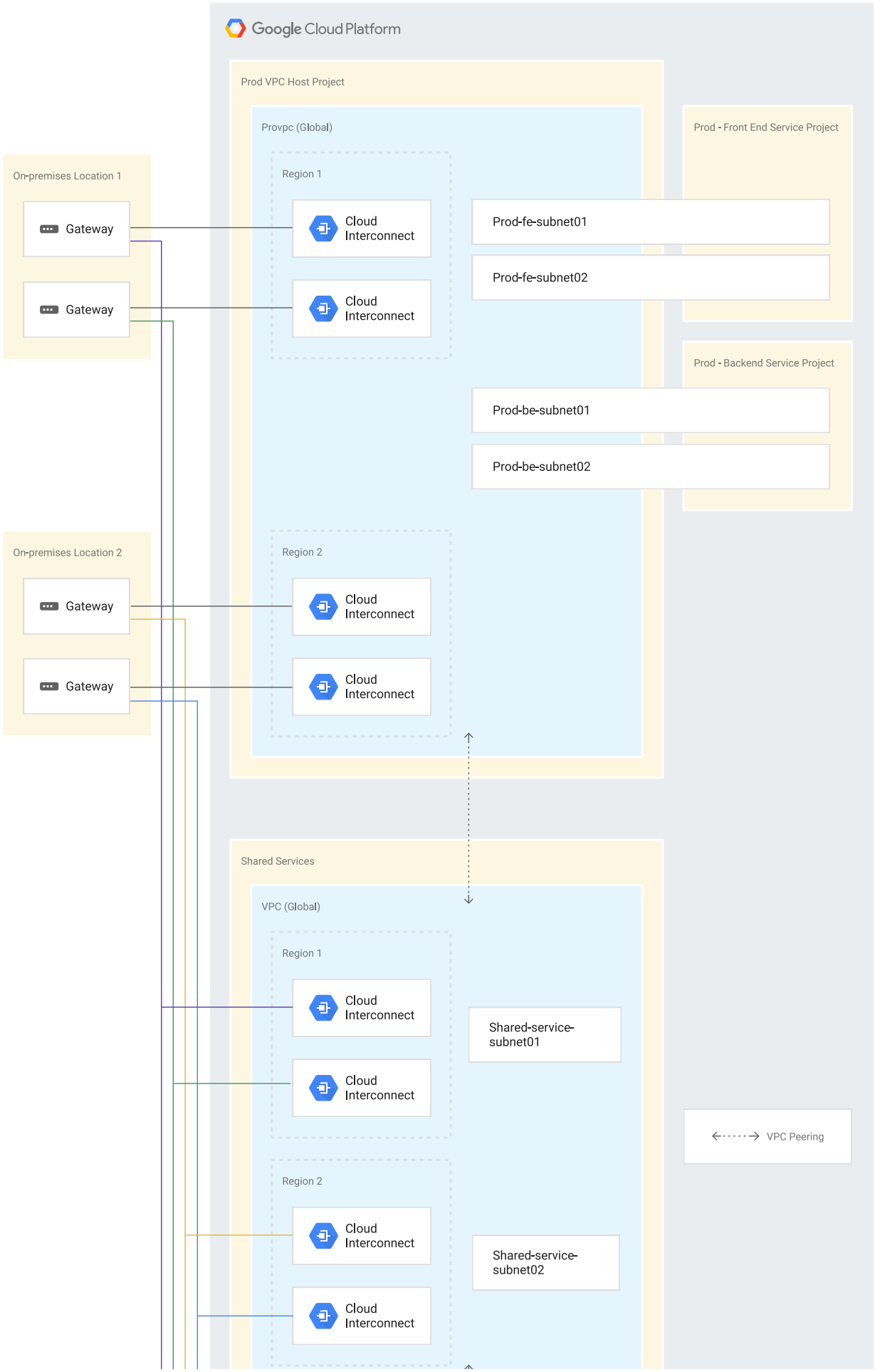
The following diagram illustrates an architecture for VPC isolation, which builds on our high-availability design while separating prod from other projects. There are many reasons to consider VPC isolation, including audit requirements (such as PCI), quota considerations between environments, or just another layer of logical isolation. You only require two interconnects (for

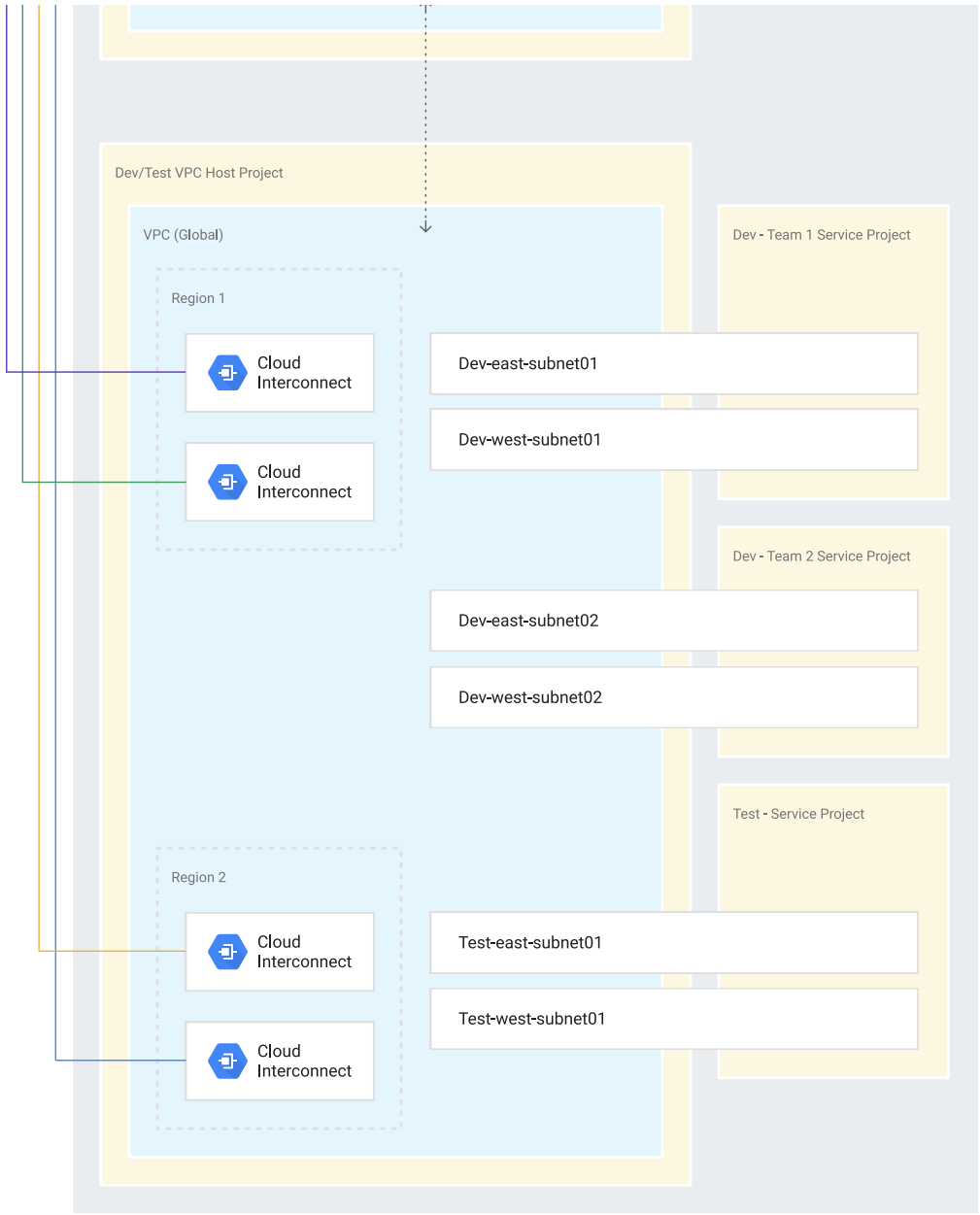
redundancy) per location but can add multiple Interconnect attachments to multiple VPC networks or regions from those.


Google Cloud Platform
Prod VPC Host Project
Provpc (Global)
Region 1
Region 2
Cloud Interconnect
Cloud Interconnect
Cloud Interconnect
Cloud Interconnect
On-premises Location 1
Gateway
Gateway
On-premises Location 2
Gateway
Gateway
Prod - Front End Service Project
Prod-fe-subnet01
Prod-fe-subnet02
Prod - Backend Service Project
Prod-be-subnet01
Prod-be-subnet02
Prod - Shared Service Project
Shared-service-subnet01
Shared-service-subnet02
Dev/Test VPC Host Project
VPC (Global)
Region 1
Cloud Interconnect
Cloud Interconnect
Dev - Team 1 Service Project
Dev-east-subnet01
Dev-west-subnet01
Dev - Team 2 Service Project
Dev-east-subnet02
Dev-west-subnet02



Using isolation can also introduce the need for replication, as you decide where to place core services such as proxies, authentication, and directory services. Using a Shared Services VPC network can help to avoid this replication, and allow you to share these services with other VPC networks through VPC Network Peering, while at the same time centralizing administration and deployment.


Google Cloud Platform
Prod VPC Host Project
Provpc (Global)
Region 1
Cloud Interconnect
Cloud Interconnect
Region 2
Cloud Interconnect
Cloud Interconnect
Shared Services
VPC (Global)
Region 1
Cloud Interconnect
Cloud Interconnect
Region 2
Cloud Interconnect
Cloud Interconnect
On-premises Location 1
Gateway
Gateway
On-premises Location 2
Gateway
Gateway
Prod - Front End Service Project
Prod-fe-subnet01
Prod-fe-subnet02
Prod - Backend Service Project
Prod-be-subnet01
Prod-be-subnet02
Shared-service-subnet01
Shared-service-subnet02
VPC Peering

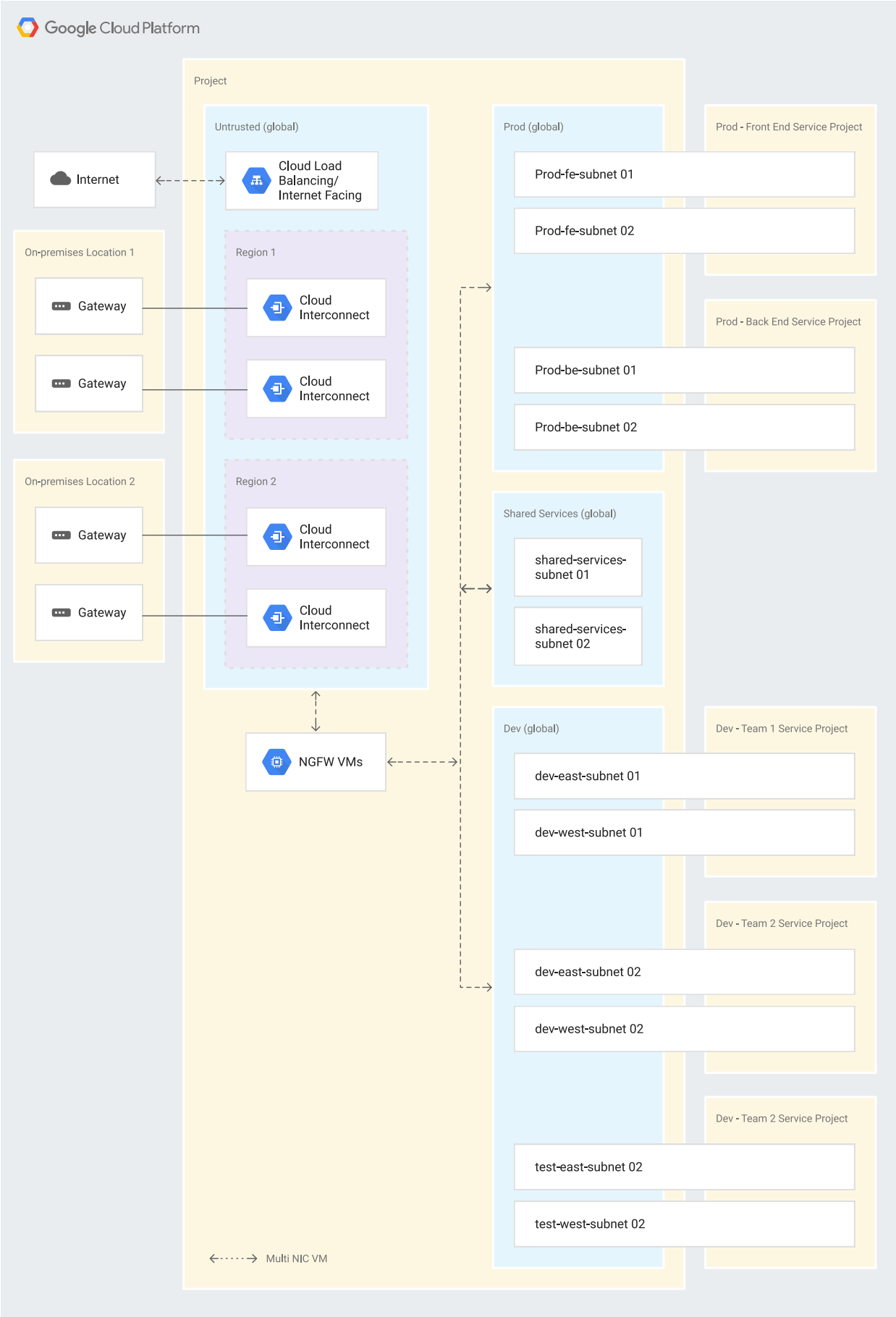


Stateful L7 firewall between VPC networks

This architecture has multiple VPC networks that are bridged by an L7 next-generation firewall (NGFW) appliance, which functions as a multi-NIC bridge between VPC networks.

An untrusted, outside VPC network is introduced to terminate hybrid interconnects and internet-based connections that terminate on the outside leg of the L7 NGFW for inspection. There are many variations on this design, but the key principle is to filter traffic through the firewall before the traffic reaches trusted VPC networks.

This design requires each VPC network to reside in the project where you insert the VM-based NGFW. Because quotas are enforced at the project level, you must consider the aggregate of all VPC resources.


Google Cloud Platform
Project
Untrusted (global)
Internet
Cloud Load Balancing/ Internet Facing
On-premises Location 1
Gateway
Gateway
Region 1
Cloud Interconnect
Cloud Interconnect
On-premises Location 2
Gateway
Gateway
Region 2
Cloud Interconnect
Cloud Interconnect
NGFW VMs
Prod (global)
Prod-fe-subnet 01
Prod-fe-subnet 02
Prod - Front End Service Project
Prod - Back End Service Project
Prod-be-subnet 01
Prod-be-subnet 02
Shared Services (global)
shared-services-subnet 01
shared-services-subnet 02
Dev (global)
dev-east-subnet 01
dev-west-subnet 01
Dev - Team 1 Service Project
Dev - Team 2 Service Project
dev-east-subnet 02
dev-west-subnet 02
Dev - Team 2 Service Project
test-east-subnet 02
test-west-subnet 02
Multi NIC VM

What's next

- VPC deep dive and best practices (Cloud NEXT'18 video) (https://www.youtube.com/watch?v=wmP6SQe5J7g)
- Hybrid and multi-cloud network topologies (/solutions/hybrid-and-multi-cloud-network-topologies)
- Best practices for network design in the Google Cloud Architecture Framework (/architecture/framework/system-design/networking)
- Best practices for Compute Engine region selection (/solutions/best-practices-compute-engine-region-selection)
- Google Cloud for data center professionals: networking (/docs/compare/data-centers/networking)
- VPC documentation (/vpc/docs)
- GKE networking overview (/kubernetes-engine/docs/concepts/network-overview)
- Explore reference architectures, diagrams, and best practices about Google Cloud. Take a look at our Cloud Architecture Center (/architecture).

Last updated 2023-05-08 UTC.