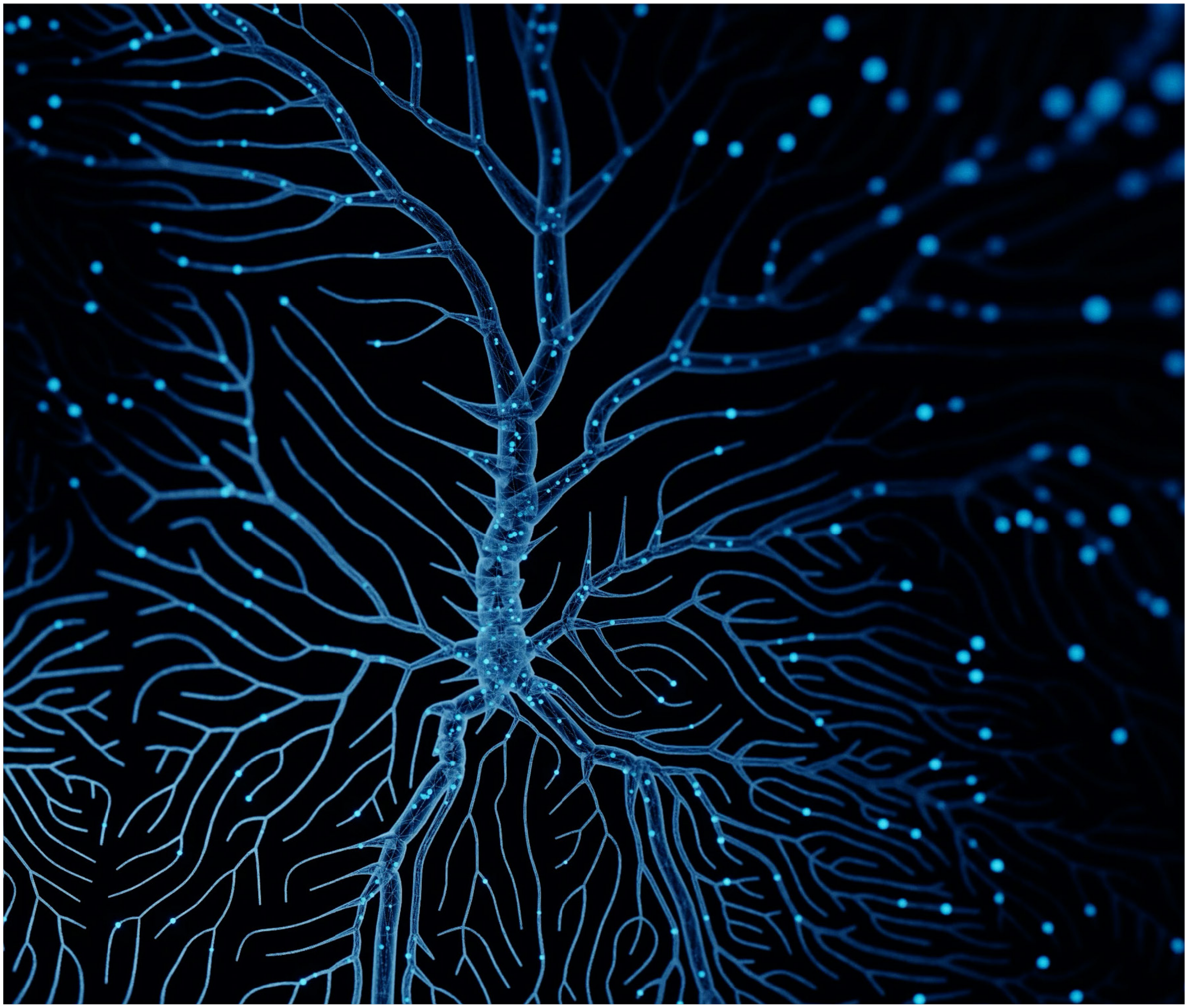
Google Cloud

Unlocking the power of GKE’s flat network: design recommendation

Scale your GKE network for gen AI and agentic AI workloads



Abstract

A Kubernetes network model describes how pod IPs integrate with the larger network. Google Kubernetes Engine's (GKE) **flat network** makes pod IPs directly routable on the wider network, simplifying external access. In contrast, **island-mode** isolates the pod network within the cluster, creating overlays or isolated 'islands' requiring network address translation (NAT) or gateways for external communication.

This guide offers recommended practices for designing, deploying, and managing GKE's flat network model, and contrasts it with the island-mode network model. It provides designs to adapt island-mode to GKE's flat networking architecture and presents the latest GKE innovations that enhance the flat network's capabilities.

The massive adoption of GKE by a wide spectrum of customers signals a definitive strategic pivot towards containerized applications for enhanced scalability and agility. GKE is increasingly being adopted for gen AI and agentic AI workloads due to its cloud-native architecture, advanced networking capabilities, hardware accelerator integration, scalability, reliability, and state management benefits.

As organizations increasingly entrust their mission-critical workloads to the GKE platform, understanding and leveraging the power of its default network model – fully integrated or flat – is key to success.

Network model: flat or fully integrated

Key network features

- Every pod in a GKE cluster gets a unique IP address
- Pods share IPs with VPC and are routable in the VPC network
- Full IP reachability across nodes, clusters, and VMs

Network model: island-mode

Key network features

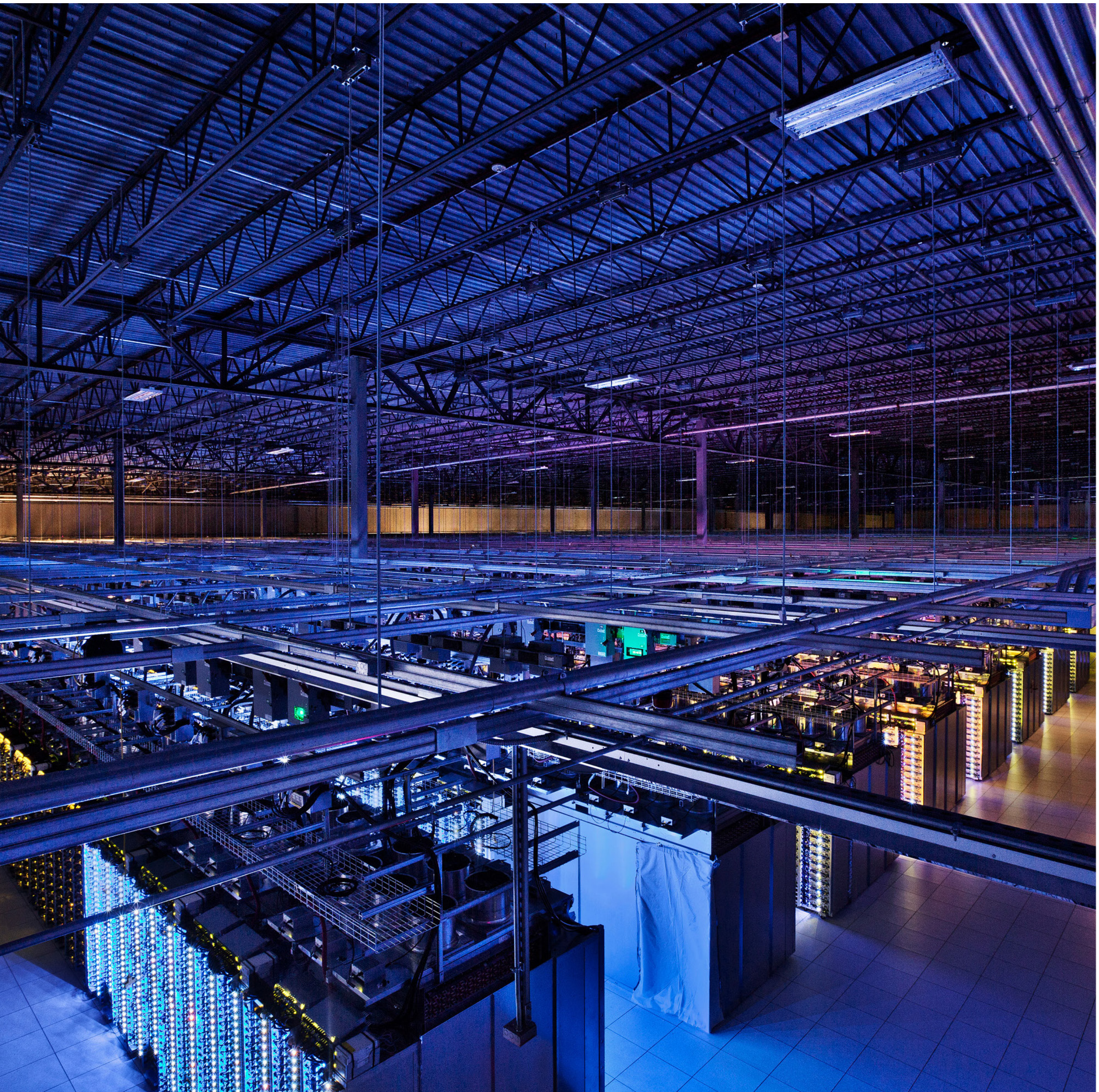
- Pods don't have unique addresses across clusters
- Typically communicate with resources outside the cluster through gateways or proxies, often involving NAT
- Pods use isolated ranges and typically use overlay networks

Overview

GKE leverages a flat network model as the default. This is sometimes also referred to as the “IP-per-pod” model.

This network model aligns elegantly with Google Cloud's global Virtual Private Cloud (VPC) concept to provide global connectivity without need for NAT or complicated overlay networks, facilitating multiregional deployments of distributed applications.

Kubernetes provides diverse network implementation options, including the alternative island-mode network model. Let’s consider their main characteristics, pros, and cons.



Feature/aspect	GKE's flat or fully integrated	Island-mode
Pod IP range	Part of the VPC IP range	Could be part of the VPC IP range or a separate subnet, isolated from VPC
Pod IP	Unique across the VPC	Typically pod IPs are reused
Pod-to-pod communication	Native or non-NAT. Allowed across all GKE clusters and VMs by default	Restricted to within the cluster
Integration with VPC	Seamless – pods are first-class citizens of the VPC	Pods are isolated from VPC by default
Load balancing	Container-native L7 load balancing that can take advantage of all the advanced algorithms offered by Google Cloud's load balancers	Load balancing happens across the VMs, then from the VM the load balancing happens across nodes since the pod IP is not directly routable within the VPC
Security boundary for cross-cluster communication	Needs network policies or firewalls to isolate.	Natural isolation due to subnet boundaries
Service discovery	Global, native with VPC DNS and GKE internal services	Local to the cluster. Needs complex IPAM co-ordination between clusters or complex NAT logic when crossing cluster boundaries. Needs extra setup for cross-cluster DNS – typically achieved via some sort of static or dynamic routing protocol to distribute the pod ranges to the nodes for communication
Compliance needs	May require more controls for regulatory isolation	Might be preferable for PCI, HIPAA, or regulated data zones
Simplicity	Easier to manage (setup) and maintain (troubleshooting, observability)	More complex to setup and maintain
Scalability	Improper IP address management (IPAM) could lead to challenges. This document details strategies to address IPAM	Can be scaled by reusing IP-address ranges for in-cluster communication

For customers with extensive on-premises networks, or that are transitioning from other clouds, who want to adapt island-mode model to GKE's flat network, this design recommendation document will help navigate and fully harness the potential of GKE's flat network via strategies that include efficient IP address planning and leverage VPC-native networking features.

Intended audience: Platform Engineers, DevOps, Network Admins, Cluster Admins, Fleet Admins

Prerequisites: This document assumes familiarity with Kubernetes concepts, networking planning, and deployments.

Customer challenges

Adopting cloud-native architectures can be incredibly powerful, but it's not without its challenges. Many organizations face hurdles with foundational networking that can impact scalability, operational efficiency, and seamless hybrid cloud integration. Thankfully, solutions exist to transform these challenges into opportunities.

One of the main concerns is the scarcity of private IPv4 address space, which can become a major headache when integrating extensive on-premises networks or migrating from other cloud environments.

Another common issue is hindered pod connectivity. In some deployments, the lack of direct, routable pod IP addresses forces a reliance on complex gateway proxies, which can introduce latency and make it difficult to deploy distributed applications.

Additionally, many teams struggle with limited network visibility, which makes it challenging to understand traffic flows at a granular level and complicates troubleshooting. When applications are built to expect a flat, directly addressable network, migrating them to some Kubernetes environments can be an arduous process.

Imagine the power of direct, routable IP connections to every pod! With a flat network model, you can unlock incredible benefits, including:

Simplified routing

Better telemetry data

Effortless migration of existing applications to Kubernetes

Intelligent load balancing and security

The ability to seamlessly bridge your cloud and on-premises environments

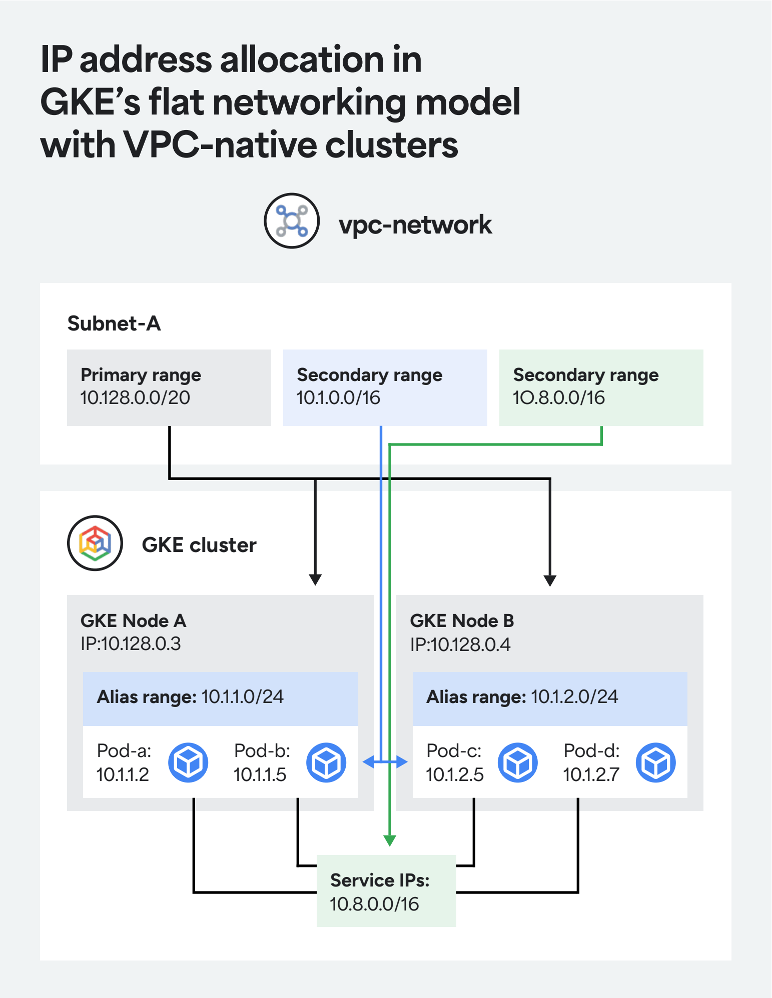
This approach tackles common networking complexities head-on, paving the way for a more efficient and powerful cloud-native journey.

GKE’s flat network model

GKE's flat network model provides every pod within the GKE cluster, and even across different GKE clusters within the same VPC network, a unique IP address. This eliminates the need for NAT or overlay networks, simplifying pod-to-pod communication both internally (within the VPC) and externally (extending Kubernetes deployments to on-prem).

The flat network model is principally compatible with the desire for Kubernetes to enable low-friction VM-to-container migration, easing legacy app transitions to GKE. For a detailed explanation on how pods, nodes, and services get IPs in a GKE cluster along with illustrations of the communication pathways within the cluster and with external resources, check out this.

GKE’s flat network model is achieved through the implementation of GKE VPC-native clusters.



In GKE VPC-native clusters:

＋

The primary IP address range of the VPC subnet is used to assign IPs to GKE nodes. This subnet should be large enough to accommodate the maximum number of nodes anticipated, as well as IP addresses for internal load balancers, if a separate subnet isn't used.

＋

A secondary IP address range within the same subnet is used by GKE to provision an alias IP range on the nodes. Pod IPs are assigned from the alias IP range associated with a node and are logically within the same subnet as the nodes.

＋

Service IP address, often referred to as the ClusterIP, is allocated by GKE from a secondary IP address range that users define specifically for services during cluster creation, and this is stable for the lifetime of the service, even if the underlying pods change.

Default communication in GKE’s flat network model deployment

Pod-to-pod	Can interact with all other pods on any nodes within the cluster without NAT
Agents on a node (e.g. kubelet)	Can communicate with all pods on that node
Pod-to-other-services within VPC	Can communicate with pod IPs without NAT
Pod-to-connected-on-premises	On-premises networks can learn routes to a pod using a cloud router configured to advertise the pod IP address ranges

Advantages of GKE’s flat network model

Performance: Reduced network hops for all internal communication within the VPC makes it faster than an island-mode network because of no address translations and no encapsulation / decapsulation overhead for pod-to-pod communication. This efficiency gain is crucial, especially for latency- and performance- sensitive workloads.

Optimized for microservices: Simplifies communication in microservices architectures and aligns with cloud-native design patterns. Note that AI inference workloads at scale are increasingly coordinated, distributed systems vs. individual, stateless servers.

Integration: Seamless integration with other Google Cloud services like load balancers, service meshes, VPC network peering, and shared VPC.

Scalability: Efficiently scales to meet your application needs, supporting up to 65,000 nodes and beyond for LLM training and inference use cases.

Enhanced features: Allows simple deployment of container-native load balancing, directing traffic directly to pods enabling the level of granularity that gen AI workloads demand.

Observability and debugging: Pod IP addresses are visible throughout the network, making telemetry data more useful for monitoring, alerting, and logging. Debugging gets simpler, even from outside the cluster with VPC Flow Logs and Connectivity Tests.

On-premises accessibility: Pod IP address ranges in VPC-native clusters can be seamlessly made accessible from on-premises networks connected with Cloud VPN or Cloud Interconnect.

Firewall rules for pod IPs: Firewall rules that specifically apply to pod IP address ranges, providing granular security control. Kubernetes network policies for finer-grained control over traffic within your cluster.

Considerations of GKE's flat network model

When GKE carves out a portion of the secondary IP address range for each node in the cluster to assign IP addresses to the pods running on that node, it inherently provisions a buffer to handle scenarios with high pod churn (e.g. frequent creation and deletion during upgrades) and to minimize IP reuse within a node.

In standard clusters: GKE reserves a /24 alias IP range (256) from the pod IP address range when the maximum pods per node is set to 110.

In autopilot clusters: GKE reserves a /26 alias IP range from the pod IP address range while allowing 32 pods to run on it.

+

IP address management: While a flat network model simplifies networking, making pod IPs routable throughout causes potential IP address exhaustion scenarios in large clusters.

+

Pod density: Limits on the number of pods per node due to IP constraints.

We will discuss how to navigate these challenges in a later section.

Island-mode: an alternative Kubernetes network model

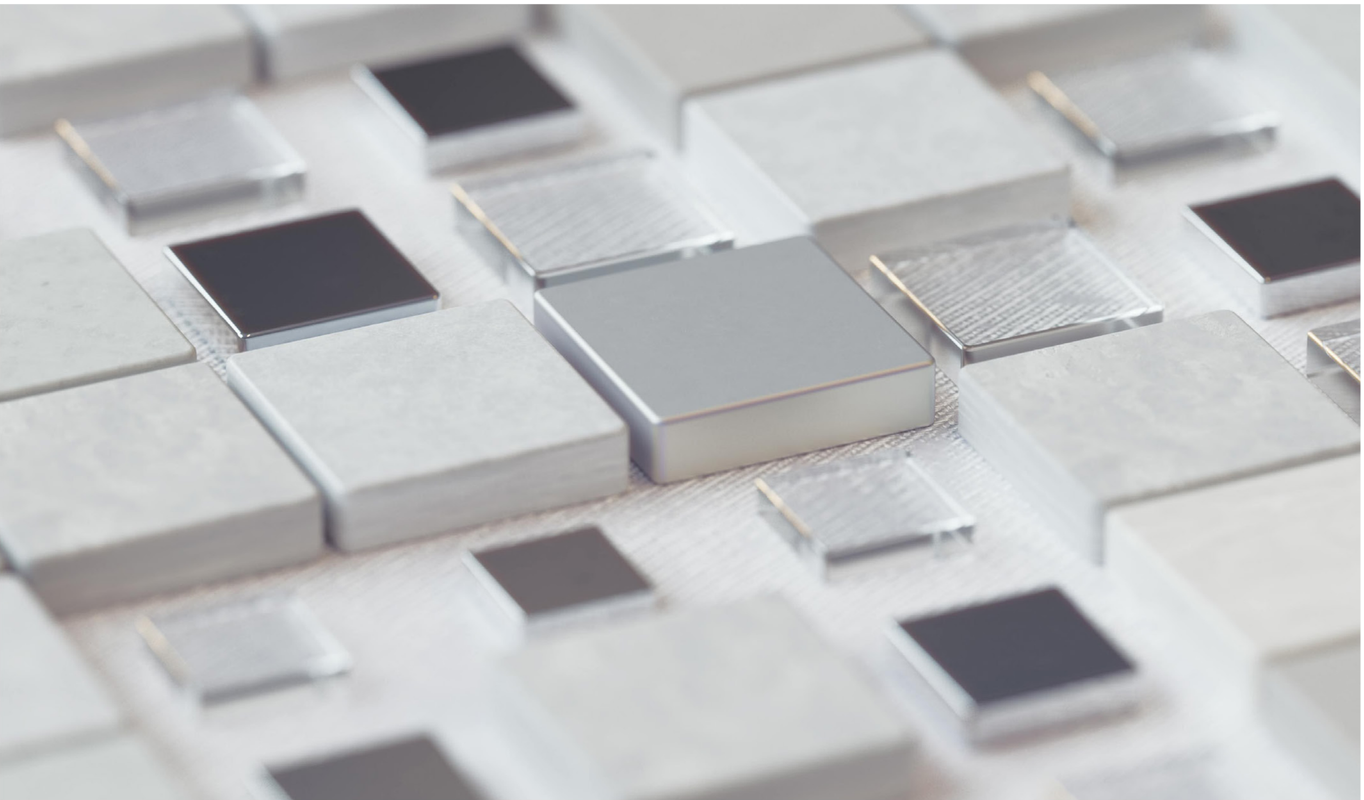
An alternative network model in Kubernetes is the island-mode. This model, commonly found in on-premises Kubernetes deployments and the default for some public clouds, isolates pod networks from the broader network, creating a hierarchical network.

The desire for an island-mode network is usually driven by IP address reuse (the next section details solutions for IPAM in GKE's flat network model) across logically separate environments or strict network segmentation that prevents direct pod-to-pod communication between 'islands'.

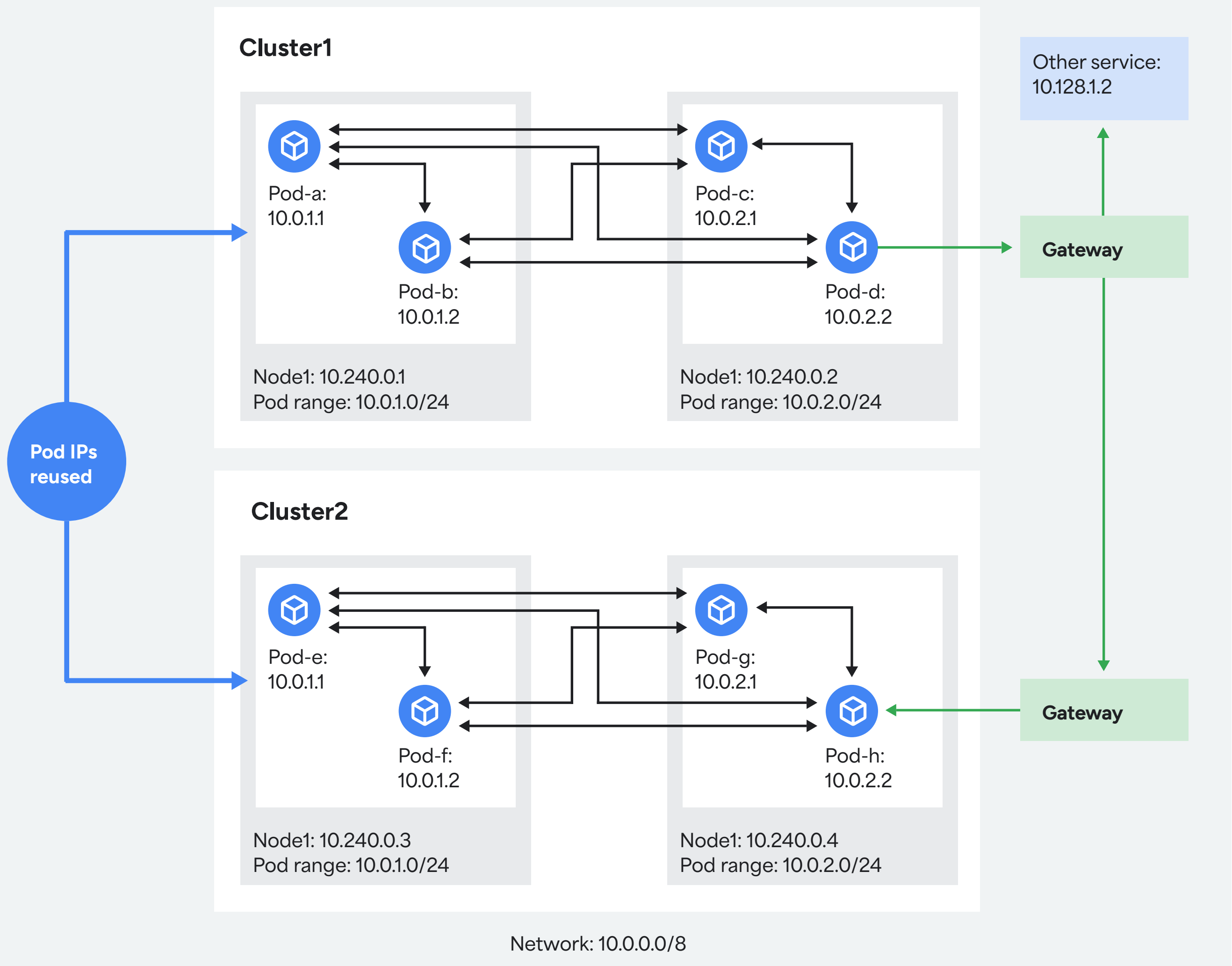
In island-mode, nodes have unique IP addresses but pods don't have unique addresses across clusters, and typically communicate with resources outside the cluster through gateways or proxies, often involving NAT. Pod ranges are typically distributed via some static or dynamic routing protocol to the nodes for communication – added complexity that makes the stack less robust. GKE does **not** support island-mode.

Some customers with IP address constraints or fragmented IP space might lean towards an island-mode where pod IP addresses can be reused across clusters. However, the island-mode network:

- Inherently incurs **performance costs due to translations and overlays**
- Is **more complex to debug** and observability of pods is tougher
- Is **less robust** due to the complexity of the static or dynamic routing protocol needed to distribute the pod range to nodes for communication



Island-mode network model

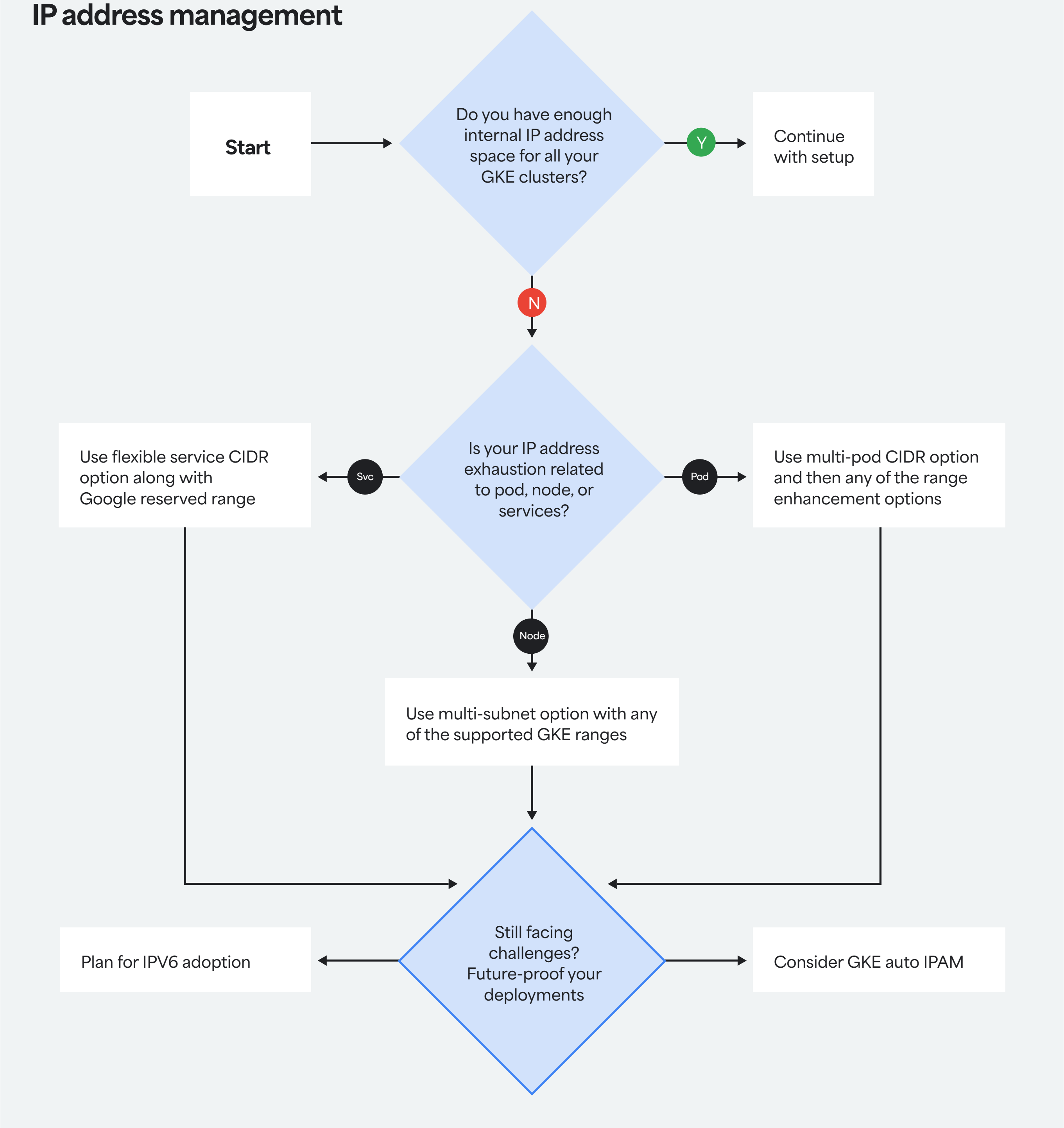


Despite the additional considerations around IPAM, we believe GKE's flat network model is superior for most cloud deployments due to its ease of communication with applications inside and outside Kubernetes, better telemetry data with visible pod IPs, simpler firewall configurations, better protocol compatibility (no NAT constraints), easier debugging with direct pod reachability, and inherent compatibility with service meshes. GKE ensures pod IP addresses are exposed in VPC Flow Logs, Packet Mirroring, firewall rules logging, and application logs.

Strategies to navigate the IPAM challenges have been detailed in the section that follows.

Navigating complexities of GKE's flat network model

IP address management


Start
Do you have enough internal IP address space for all your GKE clusters?
Y
Continue with setup
N
Is your IP address exhaustion related to pod, node, or services?
Svc
Use flexible service CIDR option along with Google reserved range
Pod
Use multi-pod CIDR option and then any of the range enhancement options
Node
Use multi-subnet option with any of the supported GKE ranges
Still facing challenges? Future-proof your deployments
Plan for IPV6 adoption
Consider GKE auto IPAM

GKE offers several strategies to improve IP capacity, flexibility, and efficiency.

Maximizing RFC 1918 addresses: Utilize the standard private IPv4 address ranges defined in RFC 1918, specifically 172.16.0.0/12 and 192.168.0.0/16, in addition to 10.0.0.0/8. These ranges are natively supported on GKE for pods, nodes, and services.

Leveraging non-RFC 1918 space: The 100.64.0.0/10 range (carrier-grade NAT or CGNAT space) can also be used alongside RFC 1918 ranges for GKE pods, nodes, and services.

Leveraging Class E IPv4 address space: The Class E IPv4 address space (240.0.0.0/4) offers a significantly larger pool of addresses (approximately 268.4 million) compared to RFC 1918 addresses. While historically reserved, Google Cloud's VPC includes Class E addresses as part of its valid address ranges for IPv4 within the VPC. However, considerations around operating system and networking equipment compatibility should be noted.

Leveraging Google reserved service IPs: Google has reserved 34.118.224.0/20 for services to help with IP exhaustion, for customers who need less than 4k IPs.

Adding additional ranges: GKE provides features to dynamically add IP ranges.

- **Multi-pod CIDR**: Enables adding additional secondary ranges for pods to new nodes in new node pools. This could also be used for scenarios where the pod needs to access more than one network.
- **Multi-subnet support**: Enables adding new subnets to an existing GKE cluster, which can then be used by new node pools, removing single-subnet limitations and allowing seamless growth and dynamic scaling beyond the initial subnet's limits.
- **Flexible service CIDR (extend service IP ranges)**: Enables the ability to increase the number of available IP addresses for services by adding a new ServiceCIDR, which allows dynamically modifying the service IP address range without downtime. This feature is available on GKE version 1.31 and later.

GKE auto IPAM: Simplifies IP address management by dynamically allocating IP address ranges for nodes and pods as your cluster grows, on-demand. It helps optimize resource allocation, enhance IP efficiency, scale with confidence, prevent IP exhaustion, and reduce administrative overhead by eliminating the need for large upfront reservations or manual intervention. It is compatible with new and existing clusters running GKE version 1.33 or greater.

IPv6 future enablement: GKE supports dual-stack (IPv4 and IPv6) clusters and is introducing future support for IPv6-only clusters for pods, nodes, and services, which can directly address IP exhaustion and meet compliance requirements. This enables end-to-end IPv6 connectivity.

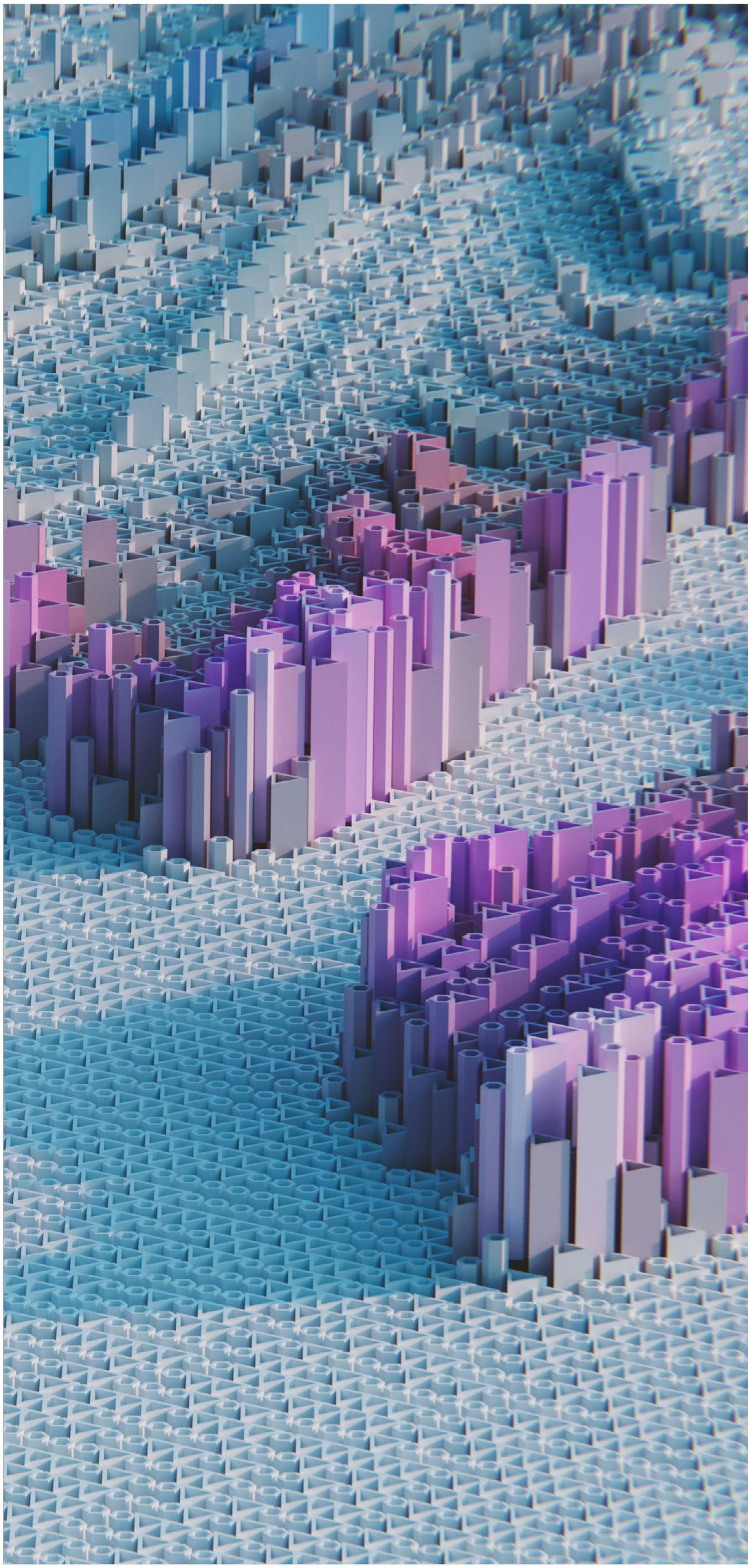
Pod density

Managing pod density within GKE is an important consideration, especially since high pod density necessitates a large number of IP addresses. Different pods have varying resource requirements, including CPU, memory, disk, and network bandwidth. Understanding these diverse resource needs is essential for effective pod density management.

GKE offers two main types of scaling mechanisms:

Horizontal scaling, where more nodes are added to the cluster within the node pool, increasing density capacity horizontally.

Vertical scaling, typically applied to the underlying VMs and involves adding more CPU and RAM to the VMs, which provides the capability to run more pods on those nodes.



Adapting to GKE's flat networking architecture

GKE does not support island-mode network setup. This means that pod ranges **cannot** be shared across clusters within the same VPC. All clusters must have unique IPs (non-overlapping), even with NAT/IP masquerade. However, there are strategies that can be leveraged to translate your current configuration to GKE's flat network model.

Adopting GKE's flat network model involves integrating the cluster's networking directly within a Google Cloud VPC.

The common considerations in this transition can be broken down as follows.

+

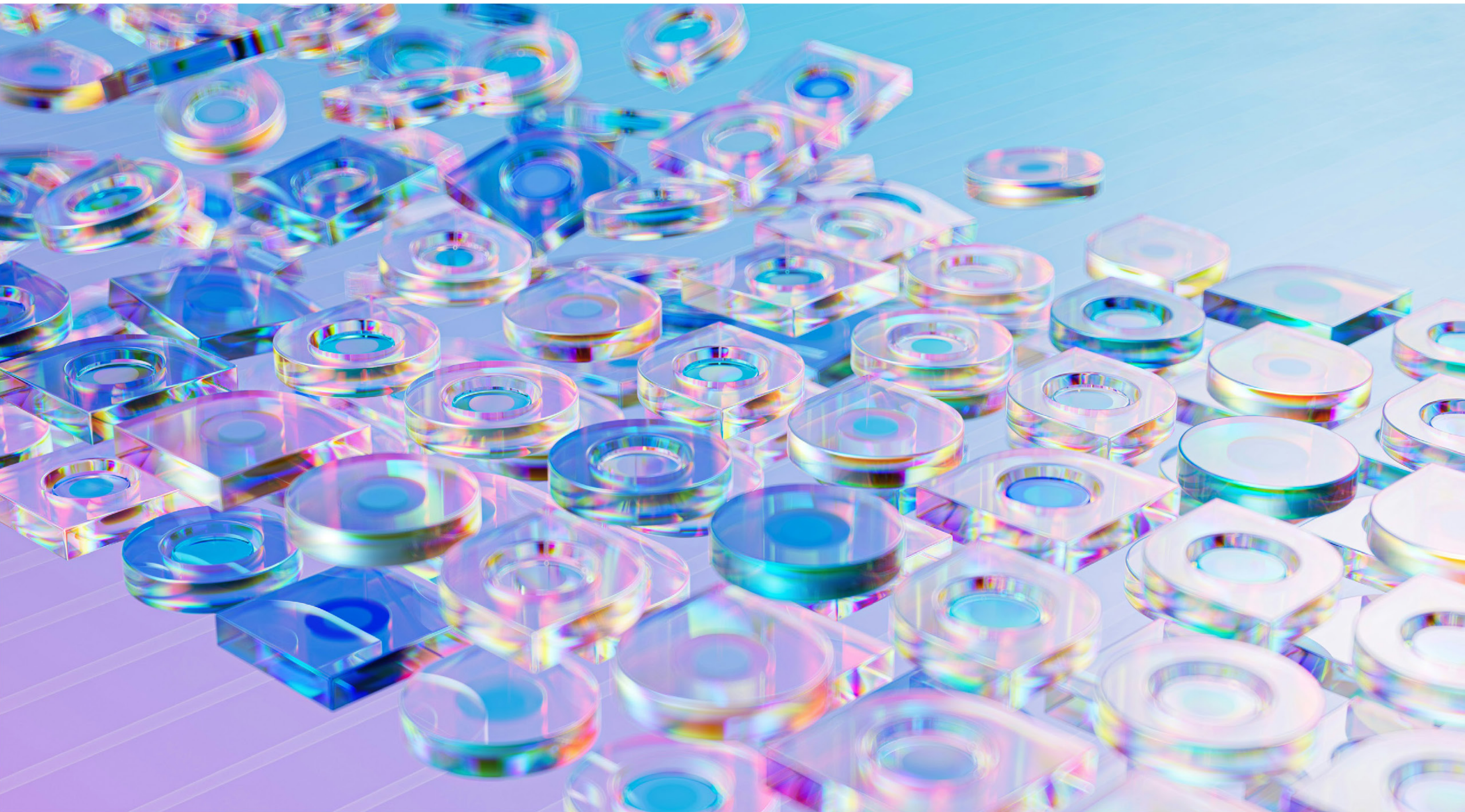
Infrastructure configuration

- Defining specific IP address spaces to be used for pods, nodes, and services
- Selecting appropriate VM type

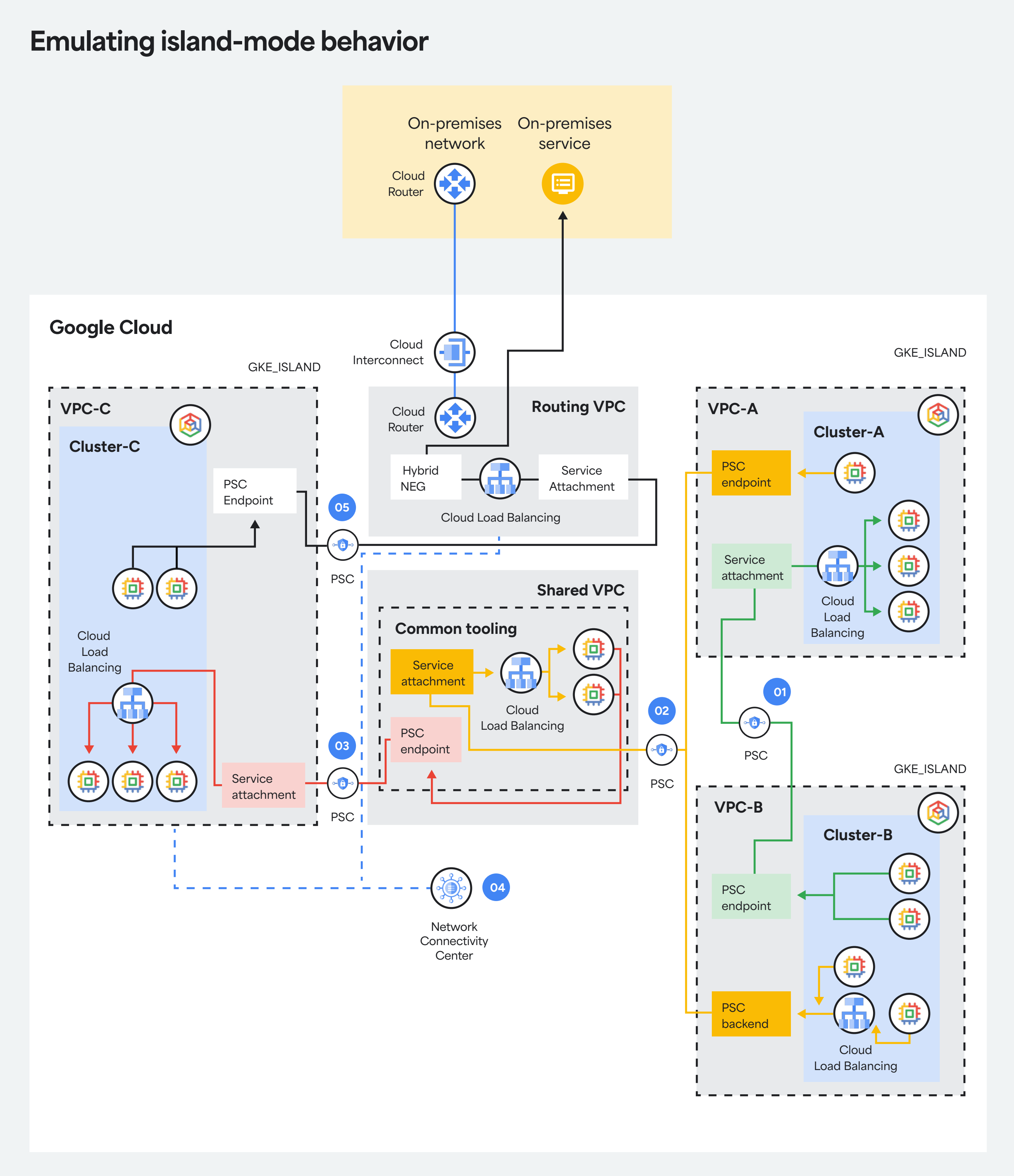
+

Shared VPC considerations

- Infrastructure tasks are managed by a network team within the host project
- Otherwise, AppDev or service teams may have the responsibility to manage their own VPCs and configurations



Option 1: Create VPC networks per 'island' and leverage Network Connectivity Center + Private Service Connect to connect GKE 'islands'



Strategy for isolation and IP reuse

Deploy each 'island' (e.g. a GKE cluster or a set of clusters) into its own dedicated VPC network. Since each VPC is a completely isolated network space, the pod IP ranges in VPC-A can overlap with pod IP ranges in VPC-B.

Strategy for inter-island communication 01

To communicate between apps in different 'islands', use Private Service Connect (PSC) for controlled, private consumption of services across VPCs without peering.

Services running on Cluster-A (producer) that need to be consumed by apps in Cluster-B (consumer) are exposed via an internal passthrough Network Load Balancer with a PSC service attachment. In the consumer VPC-B, a PSC endpoint is created. The PSC endpoint is allocated an unique IP from the VPC-B's IP ranges. Traffic is privately and securely routed between the PSC endpoint and its corresponding PSC service attachment. The consumer VPC-B only sees the PSC endpoint's IP address, not the pod IP addresses in VPC-A.

Hint: Create a Kubernetes Service of type LoadBalancer in GKE with an internal-only annotation, and GKE will automatically provision an internal passthrough Network Load Balancer.

Connectivity to common tooling and Google API services 02 03

Use PSC-based strategy for scalable and reusable service exposure and connectivity to common tooling, say in a shared VPC.

Applications in both Cluster-A and Cluster-B can connect via a single PSC service attachment in the shared VPC to access common tooling.

Alternatively, if any application in the shared VPC needs access to a service hosted in an island, (e.g. in Cluster-C), the PSC-based strategy can be extended to solve for secure, private connectivity. Note that all GKE pods within this shared VPC will have unique, routable IPs.

Strategy for hybrid connectivity 04 05

Create connectivity to on-prem via a Dedicated Interconnect or Partner Interconnect and terminate the connections in a 'Routing VPC', where all external connectivity will be handled.

Deploy a Network Connectivity Center (NCC) hub for a mesh topology to centralize hybrid connectivity. Each of the GKE 'island' VPCs, which need to communicate with on-premises environments or other public clouds (e.g. VPC-C), should be deployed as a VPC spoke connected to this NCC hub. The 'Routing VPC' should be connected as a hybrid spoke to the NCC hub.

NCC provides a centralized, managed gateway for traffic to on-premises environments or other public clouds and helps manage routing between your on-premises network spoke VPCs. Leverage export filters in NCC to advertise only specific routes.

Lastly, services running on-prem (producer) that need to be consumed by apps in Cluster-C (consumer) can be exposed via a load balancer with a PSC service attachment and a hybrid NEG.

Best practices and recommendations

01

Leverage diverse IPv4 address ranges

- Beyond standard RFC 1918 ranges, consider non-RFC 1918 ranges
- Employ Class E addresses[2] for pods and services

02

Add IP ranges dynamically as needed

- Use multi-pod CIDR to add secondary ranges for pods for new node pools
- Consider flexible service CIDR to dynamically extend the service IP range if needed
- Utilize multi-subnet support: adding additional subnets to an existing cluster

03

Plan for and adopt IPv6

- GKE supports dual-stack (IPv4 and IPv6) clusters and will introduce future support for IPv6-only clusters

04

Examine pod density

- Understand the resource requirements of your workloads for effective pod scaling (horizontal or vertical)
- Set the maximum number of pods based on requirement. Setting it too high (110) will result in IP address wastage as these IP addresses are reserved for every node
- More information: https://cloud.google.com/kubernetes-engine/docs/how-to/flexible-pod-cidr?hl=en#cidr_ranges_for_clusters

05

Leverage observability at the pod level

- The visibility of pod IP addresses simplifies debugging and enhances telemetry data, making monitoring, alerting, and logging more effective

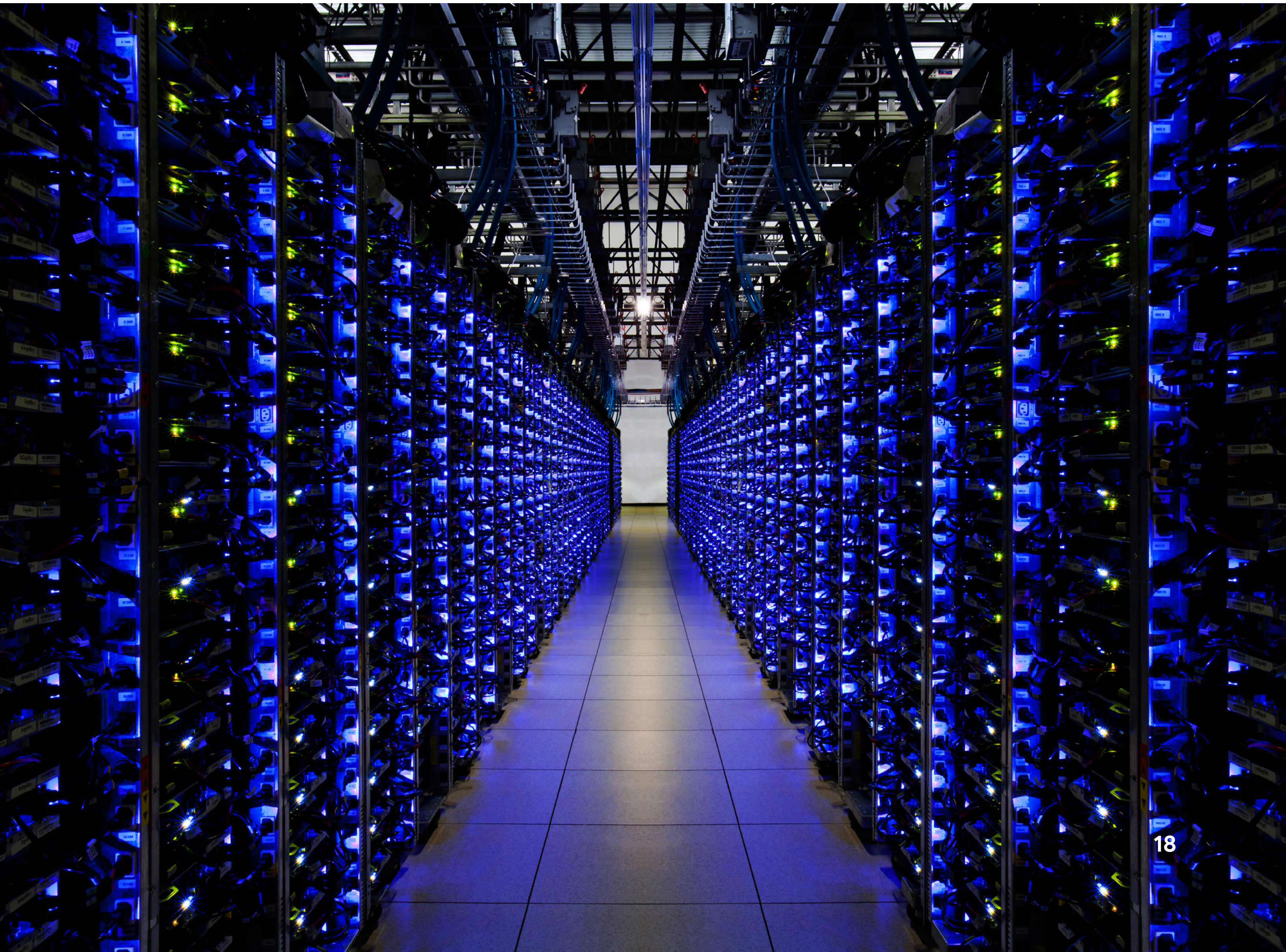
06

Implement robust security with granularity

- Utilize network segmentation with namespaces and network policies
- Configure firewall rules that specifically apply to pod IP ranges for granular control

Summary

This design recommendation document serves as a guide to best leverage GKE's default flat network model, which provides unique, routable IP addresses for every pod. This model enhances observability and simplifies communication, offering significant latency and throughput benefits, crucial for gen AI and agentic AI deployments. While large or constrained environments may face IP address management challenges, the document provides strategies and highlights recent GKE innovations to overcome these complexities, ensuring successful workload deployment and maximizing the flat network's potential for scalability, performance, and seamless integration.



Google Cloud