

Google Cloud

Partner Certification Academy



Professional Cloud Network Engineer

pls-academy-pcne-student-slides-6-2409

The information in this presentation is classified:

Google confidential & proprietary

 This presentation is shared with you under NDA.

- Do **not** record or take screenshots of this presentation.
- Do **not** share or otherwise distribute the information in this presentation with anyone **inside** or **outside** of your organization.

Thank you!



Source materials

Some of this program's content has been sourced from the following resources:

- [Google Cloud certification site](#)
- [Google Cloud documentation](#)
- [Google Cloud console](#)
- [Google Cloud courses and workshops](#)
- [Google Cloud white papers](#)
- [Google Cloud Blog](#)
- [Google Cloud YouTube channel](#)
- [Google Cloud partner-exclusive resources](#)



This material is shared with you under the terms of your Google Cloud Partner Non-Disclosure Agreement.

Google Cloud

Partner Advantage

- Cloud Foundations: GKE

Session logistics



Questions

In Google Meet, click the raise hand button or add your question to the Q&A section.

Answers may be deferred until the end of the session.



Slide availability

These slides are available in the Student Lecture section of your Qwiklabs classroom.



Recording

The session is **not** recorded.



Chat

As Google Meet does not have persistent chat, you will lose chat history if you get disconnected. Save URLs as they appear.

When you have a question, please:

Click the Raise hand button in Google Meet.

Or add your question to the Q&A section of Google Meet.

Please note that answers may be deferred until the end of the session.

These slides are available in the Student Lecture section of your Qwiklabs classroom.

The session is not recorded.

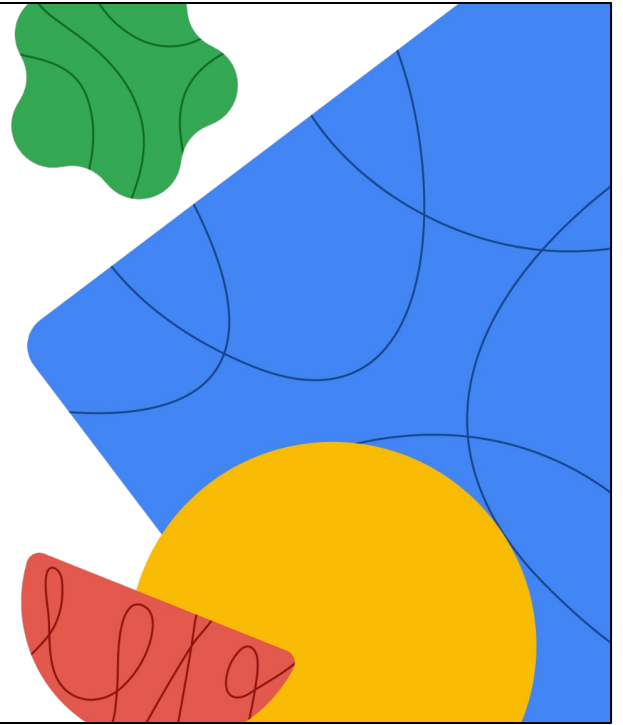
Google Meet does not have persistent chat.

If you get disconnected, you will lose the chat history.


Please copy any important URLs to a local text file as they appear in the chat.




GKE Networking



COURSE TITLE



Today's agenda



- 01 GKE basic concepts
- 02 GKE network planning
- 03 PCNE exam registration and exam-taking procedures

AGENDA

Objectives

- 01 Identify GKE basic concepts
- 02 Discuss GKE network planning
- 03 Outline the exam registration and exam-taking procedures



Objectives

GKE basic concepts

TL;DR / Purpose of the slide:

- Subtopic slide

Key points:

- **HA and DR:** Customer is probably well familiar with these terms
- **SLI SLA SLO**
 - Considered as unique to the **cloud ecosystem** compared to on-premises data centers.
 - Normally you will **depend on them** when **architecting solutions**, so it's important to understand **what they really mean**.

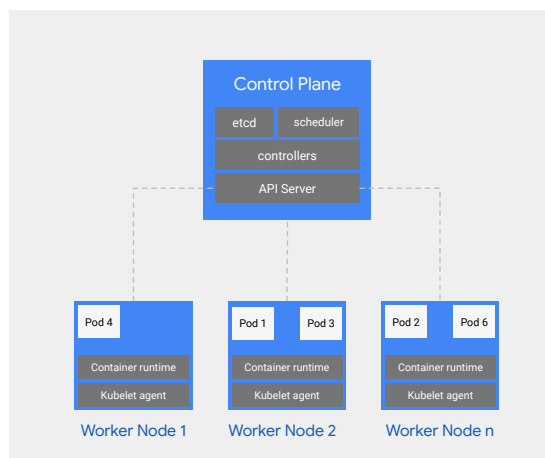
Probing questions (optional):

- None

Kubernetes architecture

A Kubernetes cluster is composed of several pieces:

- A **control plane**, responsible for handling the overall status of the cluster. Includes components such as:
 - etcd database
 - scheduler
 - controllers
 - API server
- Worker **nodes**, responsible for running user workloads and management components (container runtime, kubelet)



TL;DR / Purpose of the slide:

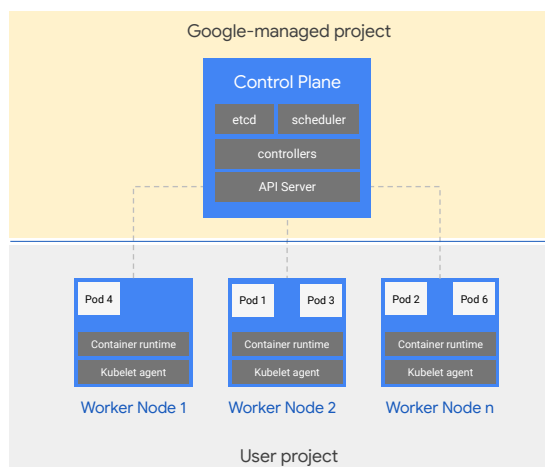
Explain the overall structure and components of a Kubernetes cluster

Key points:

- Control plane handles the overall global decisions about the cluster.
- The control plane requires/is composed of several services:
 - API server, the main entry point to interact with the cluster
 - etcd, distributed database storing the cluster's state
 - Several controllers, ensure the cluster's state is consistent with this desired state
 - A scheduler, responsible for deciding in which nodes to schedule user workloads
- The workers nodes
 - Host user workload through pods (a set of containers that work together)
 - Includes a container runtime
 - The kubelet agent, which talks to the k8s api server.

Standard GKE on Google Cloud architecture

GKE is a managed service. Google takes care of the **control plane creation and maintenance**.



TL;DR / Purpose of the slide:

GKE takes care of the control plane part. This is true for GKE standard offering, next slide covers the Autopilot flavor.

Key points:

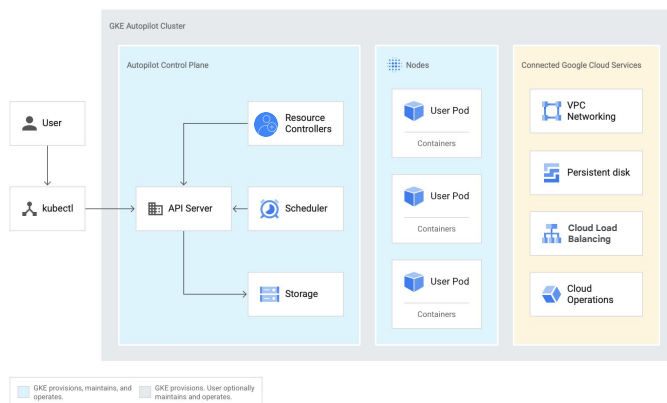
- In the previous slide we saw all the components of the control plane. GKE takes care of most of it.
- When a cluster is created, Google creates a project to host the control plane for your cluster.
- For the user, the control plane is a black box, managed using kubectl through an IP.
- Worker nodes are deployed in the user's project.

GKE Autopilot flavor - Key benefits

Google manages the entire cluster including cluster nodes

- ✓ Rich, powerful UI
- ✓ SRE monitoring
- ✓ Automated app repair
- ✓ Resource optimized app deployments
- ✓ Load balancing/autoscaling
- ✓ Global VPC
- ✓ Autopilot SLA up to 99.99%

↳ Depending on the configuration chosen



Autopilot is the second option for using GKE. Google manages everything, the customer only pays for the resources that pods consume.

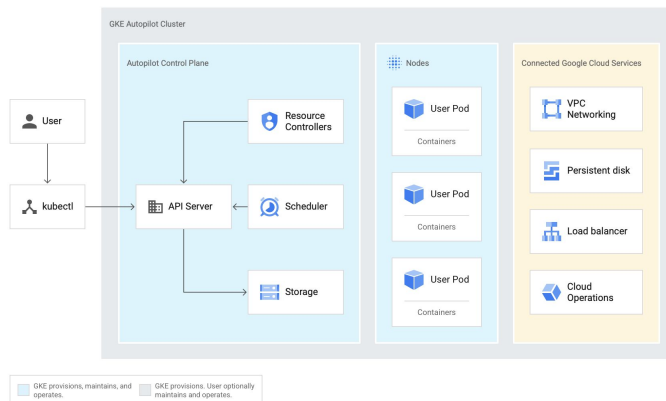
Autopilot comes with some limitations

<https://cloud.google.com/kubernetes-engine/docs/concepts/autopilot-overview#limits>

GKE Autopilot flavor - Limitations

Google manages the entire cluster including cluster nodes

- ✓ Cannot run privileged pods
- ✓ Doesn't support Service Mesh (istio/ASM)
- ✓ HostPort and hostNetwork not supported



Autopilot is the second option for using GKE. Google manages everything, the customer only pays for the resources that pods consume.

Autopilot comes with some limitations

<https://cloud.google.com/kubernetes-engine/docs/concepts/autopilot-overview#limits>

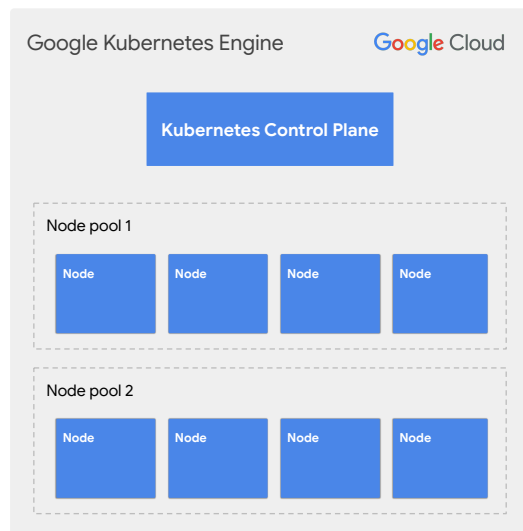
Node pools

A node pool is a **group of nodes that have the same configuration**

Node pools run in the **customer's project**

You can have **one or more node pools** per cluster

Allows you to **mix-and-match machine configurations** (for example, local SSDs, GPU, preemptible VMs, specific node images, larger instance sizes, and more)



Google Cloud

TL;DR / Purpose of the slide:

Explain the concept of node pools

Key points:

- A node pool is a group of nodes within a cluster that all have the same configuration
- A cluster can have one or more node pools
- Node pools provide a way to define different node configurations
- Node pools can be created, updated, and deleted without affecting the rest of the cluster

Node pool definition criteria

Specialized hardware	Isolation	Different configuration	Experimentation
Provide different hardware profiles for specific workloads (such as, more CPU, more memory, local SSDs, GPU, etc.).	Meet workload isolation requirements for compliance or performance reasons.	Create nodes with diverging configuration profiles (e.g. Windows nodes, preemptible VMs, custom node image, different max PPN, etc.).	Try new GKE features (such as enabling / disabling node-level configurations) or Kubernetes features (new version).

TL;DR / Purpose of the slide:

Discuss some of the reason to have different node pools

Key points:

- Specialized hardware:
 - Workloads might need specific hardware configuration. For example, AI might need a GPU, video processing might require more CPU, etc
- Isolation
 - Some industries requires certain workloads to be run in dedicated hardware
 - Some workloads might cause disruption to other pods in the same node, moving them to their own nodes can help the overall performance
- Different configuration
 - Different node-level (such as OS, vm type, etc) configurations require different node pools
- Experimentation
 - Some things can be tested without too much risk in a new node pools
 - For example, changing container runtime
 - Trying a new k8s version in worker nodes
 - Enabling disabling trusted computing features

Working with node pools

Use a **nodeSelector** together with labels to constrain in which nodes a pod can be scheduled. As a convenience, GKE automatically populates the **gke-nodepool** label for every node, which contains the name of the node's node pool.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
  nodeSelector:
    cloud.google.com/gke-nodepool: nodepool-gpu
```

TL;DR / Purpose of the slide:

Kubernetes provides a way to schedule pods in specific nodes

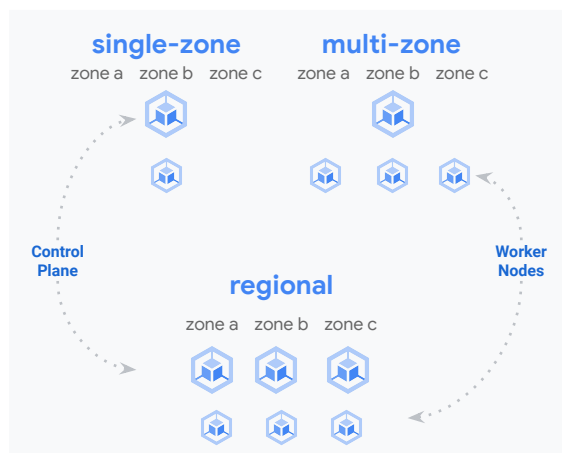
Key points:

- nodeSelector matches a pod's labels with potential nodes where
- By GKE node include the `gke-nodepool` label which is populated with the name of the node pool "owning" the node
- Monitor for under utilized nodes and rethink node pool setup if needed

Working with node pools (Cont.)

To achieve **high availability**, the Kubernetes control plane and its nodes need to be spread across different zones.

- **Zonal clusters** have a single control plane in a single zone.
- **Multi-zonal** clusters have a single replica of the control plane running in a single zone, and has nodes running in multiple zones.
- **Regional clusters (best practice)** have multiple replicas of the control plane, running in multiple zones within a given region. Nodes can run in multiple zones.



Google Cloud

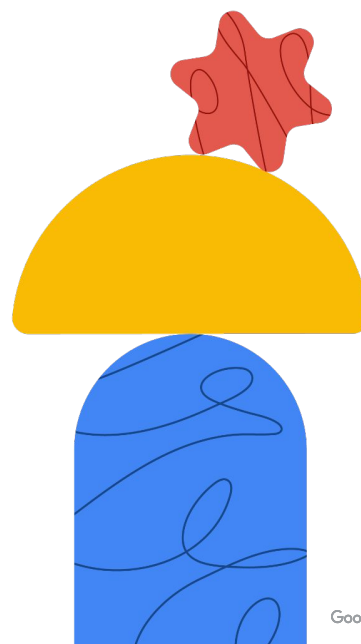
TL;DR / Purpose of the slide

Kubernetes has 3 types of cluster: single-zone, multi-zone and regional

Key points:

- Multi-zonal: During an upgrade of the cluster or an outage of the zone where the control plane runs, workloads still run
- Use regional clusters to run your production workloads, as they offer higher availability than zonal clusters.

GKE network planning



Google Cloud

TL;DR / Purpose of the slide:

- Subtopic slide

Key points:

- **HA and DR:** Customer is probably well familiar with these terms
- **SLI SLA SLO**
 - Considered as unique to the **cloud ecosystem** compared to on-premises data centers.
 - Normally you will **depend on them** when **architecting solutions**, so it's important to understand **what they really mean**.

Probing questions (optional):

- None

GKE cluster types

Routes-based

- Can only use RFC 1918 addresses
- Uses VPC custom routes for pod-to-pod traffic
- Little control over service range

Best practice

VPC-native

- Default and recommended for new clusters
- Can leverage a larger set of private IP ranges
- Pods get VPC-native endpoints, no need for additional custom routes
- Separate, VPC-routable IP ranges for pods, nodes and services

TL;DR / Purpose of the slide:

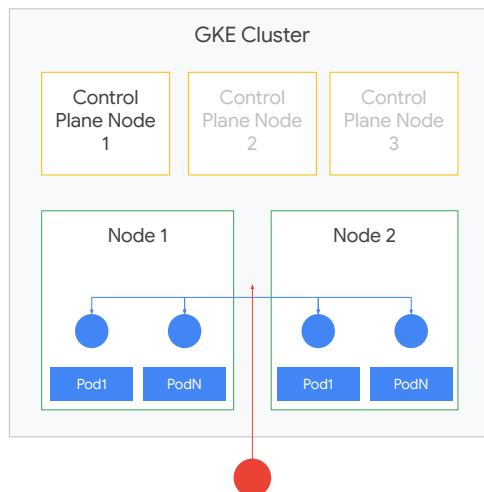
- Explain the difference between routes-based and VPC-native clusters

Key points:

- VPC-native clusters should be used for new clusters
- From here on we'll only talk about VPC-native clusters

IP management in GKE clusters

- 01 **Nodes range:** IPs for the VMs backing the cluster.
- 02 **Pods range:** IPs for pods created by the cluster. Each node gets a slice (subnet) within this range.
- 03 **Services range:** IPs for Kubernetes services (ClusterIPs).
- 04 **Control plane range:** IP address range for control plane node (private clusters only).
- 05 **Internal load balancer range:** (Optional) IP address range for internal load balancers.



TL;DR / Purpose of the slide:

- Explain the IP ranges required by a VPC-native cluster

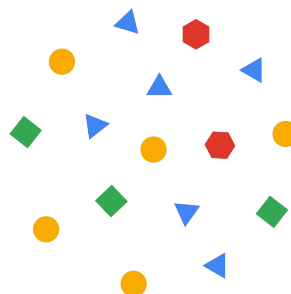
Key points

- Kubernetes uses IPs liberally.
- **Nodes** themselves get a unique CIDR block. This maps to a Subnet's Primary IP.
- The whole cluster is assigned a CIDR block from where **Pods** are allocated an IP. This maps to a Subnet's Secondary range IP.
- **Services** also get a unique CIDR block. This also maps to a Subnet's Secondary range IP.
- For private clusters, an additional range is required. This range is used by the peered network containing the clusters control plane nodes. Always /28
- Internal load balancer needs additional ranges. tcp/udp optional, http required
- Advantages of having separate cidr blocks.
 - Better efficiency for routing purposes (i.e. we can send traffic directly to the pod, this is what NEGs actually do.)
 - distinct ranges for different resources making it more intuitive for applying policies and easier for tracking
 - Easier to secure
 - Each resource can scale independently

- We don't recommend to share IP Ranges across clusters, even if it's possible to share PODS and Services IP ranges it's not a good practice.

Why do GKE clusters require so many ranges?

- Better integration with the Google Cloud networking stack
- Reserve ranges for pods and services
- **Pod and service IP addresses are natively routable within the cluster's VPC network**
- Better efficiency for routing purposes
- Each resource can scale independently



TL;DR / Purpose of the slide:

- Explain the motivation behind the additional configuration complexity of VPC-native clusters

Key points

- Pod IP address ranges do not depend on custom static routes
- Better efficiency for routing purposes (i.e. we can send traffic directly to the pod—this is what NEG's actually do)
- distinct ranges for different resources making it more intuitive for applying policies and easier for tracking
- Easier to secure
- Each resource can scale independently
- **IPAM is key for a successful GKE strategy**

Sizing ranges (high-level view)

Node	Pods	Services	Control plane
<p>1 IP per Node</p> <p>Assigned from the primary IP address range of the subnet associated with your cluster.</p>	<p>$\text{Nodes} \times \text{Max PPN}^* \times 2$</p> <p>Assigned from a secondary range for pods.</p> <p>* PPN = Pods Per Node, default 110</p>	<p>1 IP per Service</p> <p>Taken from a secondary IP for services.</p>	<p>16 IPs</p> <p>Always a /28 range.</p>

TL;DR / Purpose of the slide:

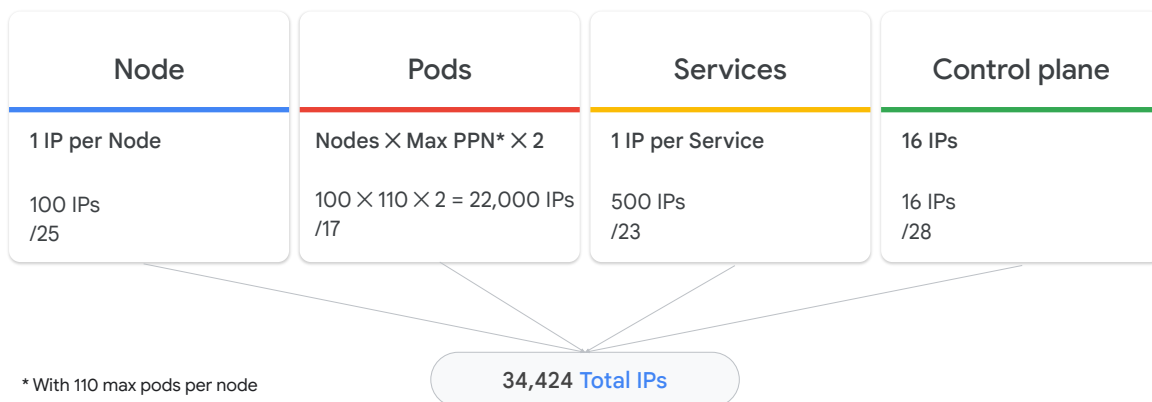
Quick review of number of IP required for each range

Key points

- Nodes
 - Count one per maximum number of node
 - Also used to allocate ILB IPs if the annotation mentioned later is not used
- Pods
 - $\text{Max nodes} \times \text{Max pods per node} \times 2$
 - $\text{Max PPN} \times 2$ to allow for rolling updates. Default is 110 and is defined by k8s (not GKE)
- Services
 - One per service
- Control plane
 - Always 16 IPs

Sizing ranges (high-level view)

For cluster with at most **100 nodes*** and with up to **500 services** you need:



TL;DR / Purpose of the slide:


Give an example with real number. Highlight that, generally, the pod range dominates the total required IPs for a given cluster.

Key points

- Pods per node can be changed. Later we'll see how.
- ~~We have a tool that can help with the math:~~
<https://googlecloudplatform.github.io/gke-ip-address-management/>

See also

~~<https://cloud.google.com/kubernetes-engine/docs/concepts/planning-large-clusters#limits-best-practices-large-scale-clusters>~~

 <https://cloud.google.com/>

Google Cloud

Limits and best practices



For limits and best practices for large GKE clusters, visit the ‘Limits and best practices’ documentation.

 <https://googlecloudplatform.github.io/>

Google Cloud

GKE IP Address Management



A tool that can help with the math, titled 'GKE IP Address Management' is available in the official Google documentation.

IP management options

Optimize IP address usage

Size node/pod/service **ranges** correctly

Optimize pod density

Define a custom **pods-per-node** value

Running out of RFC1918 addresses

Use **non-RFC 1918** private addresses

Running out of private IP addresses

Consider using Privately Used Public IPs (**PUPI**)

Need additional pod addresses

Use **discontinuous multi-pod CIDR**

TL;DR / Purpose of the slide:

- Introduce the tools available to control GKE IP address usage. We'll cover them in details in the following slides.

Custom pods-per-node value

By default GKE uses 110 pods per node, which requires 220 IPs per node (a /24)

By changing the pods per node value we can optimize the number of IPs assigned to each node .

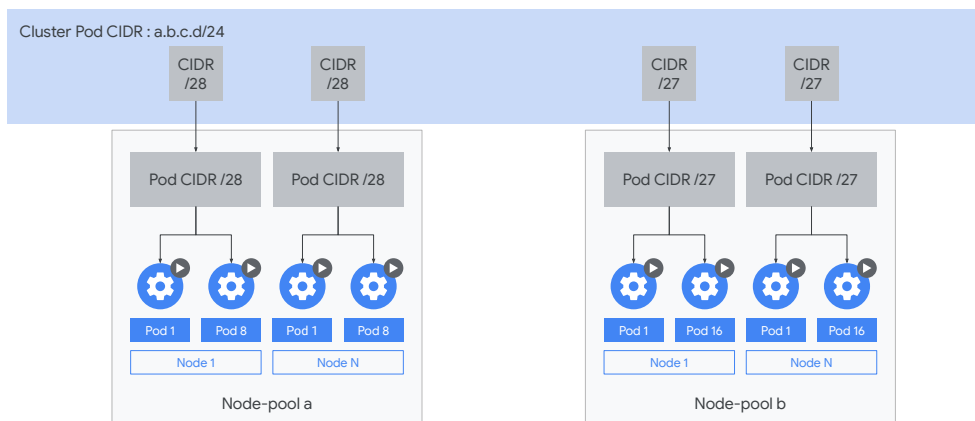
Maximum Pods per Node	CIDR Range per Node	Number of IPs
8	/28	16
9-16	/27	32
17-32	/26	64
33-64	/25	128
65-110	/24	256

TL;DR / Purpose of the slide:

Key points

- Motivation: each node gets 2*Pods-per-node IPs. If we can change the pods per node value, we optimize how pod ips are required by the cluster
- The whole idea is to decrease the rate at which the Pod range is exhausted
- Point back to the slide mentioning pod range requires Nodes*Pods-per-node*2 ips
- Why multiply by 2 pod-per-node? What are the extra ips for? → rolling updates
- Can be defined at the cluster or nodepool level
- AKA Flexible Pod CIDR

Custom pods-per-node value (cont.)



TL;DR / Purpose of the slide:

Show how the pod range is split between nodes and node pools

Using non-RFC 1918 and PUIP addresses

If your network is running out of RFC1918 addresses, GKE can use any of the following ranges for nodes, pods, and services.

CIDR Range	Description	Number of IPs
100.64.0.0/10	Shared address space (RFC 6598)	~4.2 million
192.0.2.0/24 198.51.100.0/24 203.0.113.0/24	Documentation (RFC 5737)	768
192.88.99.0/24	IPv6 to IPv4 relay (deprecated) (RFC 7526)	256
198.18.0.0/15	Benchmark testing (RFC 2544)	130k
240.0.0.0/4	Reserved for future use (RFC 5735 and RFC 1112)	~268 million
PUIP	Privately used public IP addresses	-

TL;DR / Purpose of the slide:

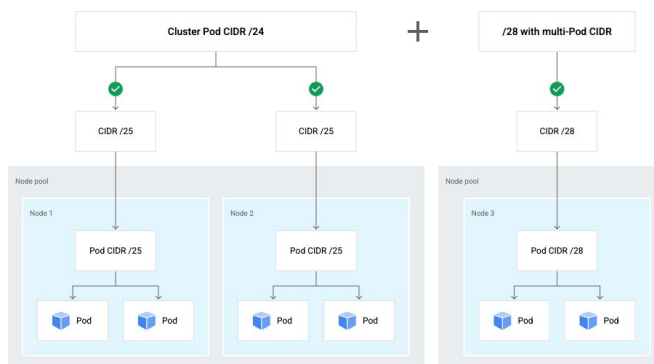
- Explain the motivation behind the additional configuration complexity of VPC-native clusters

Key points

- Sometimes there's just not enough RFC 1918 addresses available.
- GKE clusters can use private IP address ranges outside of the RFC 1918 ranges for nodes, pods, and services.
- When defining the subnet that will contain the GKE cluster, any of these ranges can be used.
- This is actually a VPC feature (configured at subnet creation time), but it's particularly useful for GKE
- Caveats
 - Some OSs (Windows, on-premises routers) don't support the /4 listed at the end.
 - PUIP addresses must not overlap any Google-owned range, or any restricted range

Using discontinuous multi-pod CIDR

- Discontinuous multi-pod CIDR allows adding pod range for a new node pool
- No need to recreate cluster to grow pod IP space
- Can fit clusters into **fragmented address space** (multiple discontinuous /24)
- Allows creating smaller clusters that can be expanded later, which results in **more efficient IP allocation**

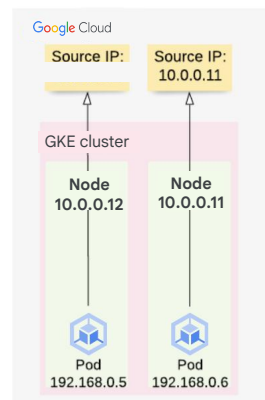


IP masq agent

- IP masq SNAT is the pod IP behind the Node IP. This is handled by ip-masq-agent.
- By default the ip-masq-agent is installed if:
 - The cluster has a network policy
 - The pod's CIDR range is not within 10.0.0.0/8
 - The cluster was created without the --disable-default-snat flag, and has Workload Identity enabled

When installed by default the ip-masq-agent is configured to NOT SNAT pod IPs for the following ranges:

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16
- 100.64.0.0/10
- 192.0.0.0/24
- 192.0.2.0/24
- 192.88.99.0/24
- 198.18.0.0/15
- 198.51.100.0/24
- 203.0.113.0/24
- 240.0.0.0/4



- IP Masq agent SNAT POD IPs based on the destination of the traffic.
- Installed by default if the customer matches one of the 3 conditions above.
- By Default configured not to masq traffic going to the destination ranges above. Customer can configure it by adding a configMap that has a **nonMasqueradeCIDRs** config

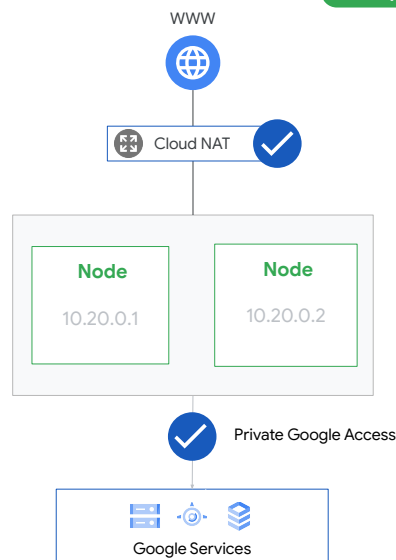
<https://cloud.google.com/kubernetes-engine/docs/how-to/ip-masquerade-agent>

Private clusters

Best practice

Private clusters isolate nodes from having inbound and outbound connectivity to the public internet

- Nodes have only private IP addresses
- Nodes use Private Google Access to communicate with Google APIs
- Nodes can use Cloud NAT to reach the internet
- Control Plane gets an additional private endpoint for the cluster nodes to talk to the control plane



Google Cloud

TL;DR / Purpose of the slide:

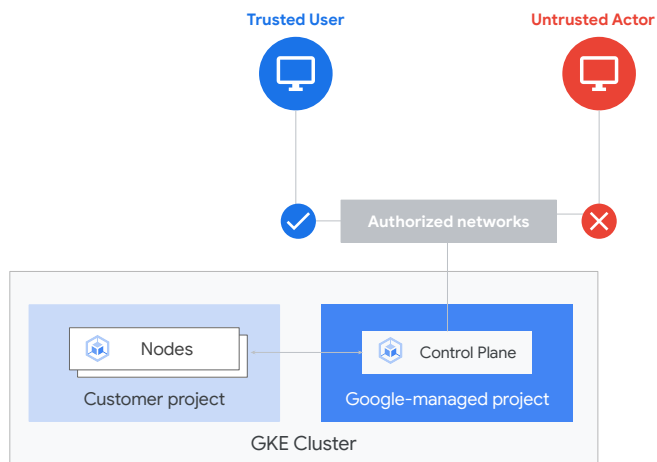
By default, nodes get external IP addresses. Private clusters removes node external IPs.

Key points

- In public clusters, the cluster master (primary) talks to the node over their public IPs, but since these are Google IPs, the communication happens over our Backbone.
- With private clusters, nodes don't get external IPs. Inbound and outbound access to internet is restricted (just like regular VMs without external IP)
- A VPC peering connection is created between customer's VPC network and new VPC network in a Google-owned project.
- Control plane has both an internal IP address and an external IP address (see next slide to control access to primary)
- Nodes talk to the primary via an Internal Load Balancer (ILB).
- For private clusters, the primary that is located in a Google-managed project has a public endpoint (public IP) and a private endpoint (private IP) that is behind an internal google load balancer.
- Cloud NAT for internet egress

Controlling access to the control plane

- Authorized networks restrict access to the control plane to trusted CIDR ranges. Mandatory for private clusters.
- By default nodes and pods ranges are allowed.
- Google recommends activating it for all clusters.
- Control plane access via public IP can be disabled (recommended). This is called Private endpoint.



Google Cloud

TL;DR / Purpose of the slide:

With both public and private Clusters, you can control access to your control plane through Authorized Networks.

Key points

- Authorized networks is disabled by default when using public endpoint. Google recommends activating it for all clusters.
- Configuring Master Authorized Networks is mandatory if private endpoint is enabled.

Specify subnet for (TCP/UDP) internal load balancers

By default, GKE uses the node range for internal TCP/UDP load balancers

Use the [internal-load-balancer-subnet](#) annotation to force the creation of the ILB in a particular subnet

Benefits



Prevent inadvertently exhausting node IPs.



Apply different firewall rules to nodes and ILBs.

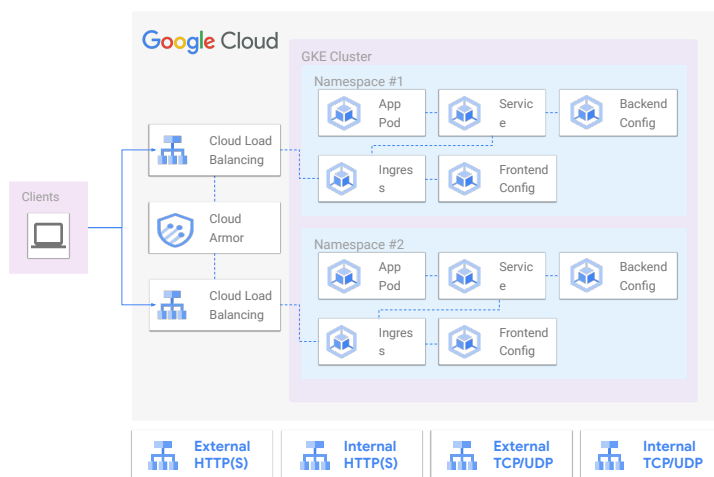
TL;DR / Purpose of the slide:

Recommend to always deploy GKE ILBs in a subnet dedicated to that purpose.

Key points

- Exposed services can inadvertently prevent the cluster from scaling if node IPs are exhausted
- Separate subnet for ILB allows different set of firewall rules for LBs and nodes
- ILB subnet is specified through an annotation in the Service. Different services can use different subnets (for example different subnets for each namespace)

Exposing applications



- Native integration with **Google Cloud Load Balancing (GCLB)**
- Broad support for **different types of load balancers** and **parameterizable**
- Compute Engine Ingress is **namespace bound**
- Control the exposure by **Org Policy** `"compute.restrictLoadBalancerCreationForTypes"`
- Use **alternative Ingress controllers** (nginx) in a self-managed manner
- **GKE Gateway**^(preview) is a more flexible evolution of Ingress

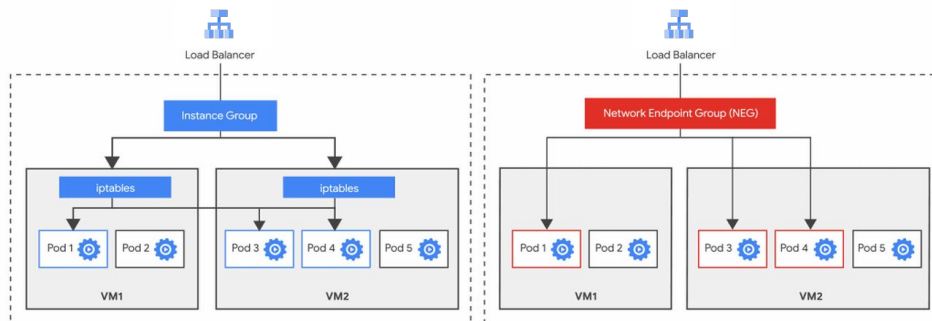
TL;DR / Purpose of the slide:

Provide an overview of GKE Ingress and mention the alternatives.

Key points

- By default, when creating an Ingress, GKE uses native Compute Engine ingress controller and creates a GCLB.
- The ingress parameters can be configured - refer to [Configuring Ingress features](#) for the full list .
- A Ingress is bound to a namespace - alternative ingress controllers can be used instead (like istio, nginx). This allows a centralized approach of Ingress, but introduces more management overhead and loss of native ingress functionalities and integrations. The Gateway Controller will also address that in the future (see next slide).

Use Container Native Load Balancing (aka NEGs) to expose services



```
kind: Service
metadata:
  annotations:
    cloud.google.com/neg: '{"ingress": true}'
```

Google Cloud

TL;DR / Purpose of the slide:

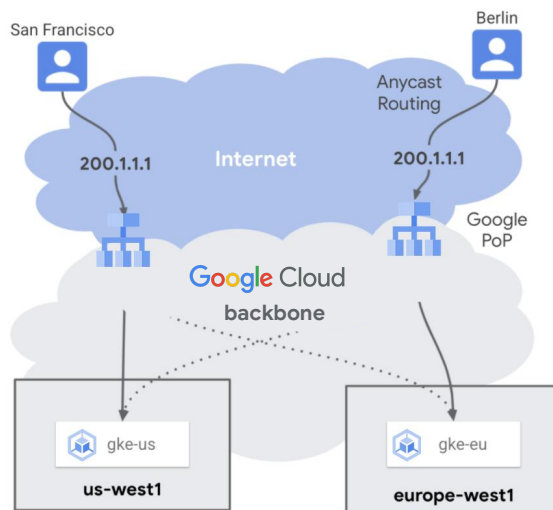
Recommend using container load balancing for ingress

Key points

- Container native load balancing leverages Network Endpoint groups
- NEG send traffic directly to Pod addresses (this one one of the reasons to prefer VPC native clusters)
- Prevents additional hop through the node (lower latency since the node->pod hop from the left diagram can be in different zones)
- NEG is fully managed by GKE.

Multi-cluster ingress

- A single VIP for globally available, low-latency, multi-cluster Kubernetes applications
- Connection is terminated at closest PoP and routed to closest cluster with healthy backends
- Supports all configuration options of regular load balancers (affinity, health checks, etc.)
- Builds on the architecture of the External HTTP(S) Load Balancing



Google Cloud

TL;DR / Purpose of the slide:

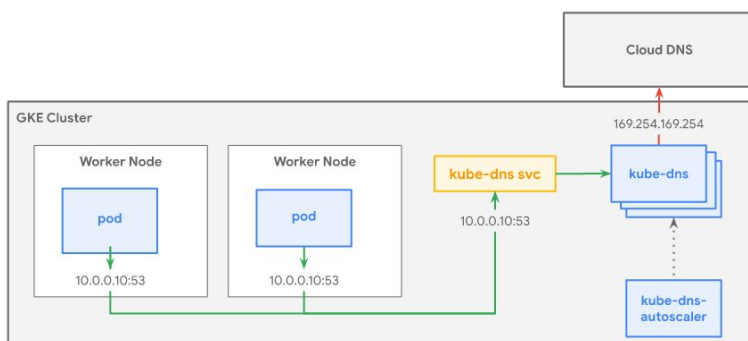
Introduce the concept of multi cluster ingress.

Key points

- MCI offers a cloud hosted multi-cluster controller that load balances resources across clusters and across regions. It's part of the Anthos entitlement but customers can use it independently as well. See [Multi Cluster Ingress](#) for the pricing details.
- Refer to [Configuring Ingress features](#) for configuration options for MCI

Kube-DNS vs Cloud DNS

- In-cluster DNS servers
- Resolves cluster FQDN (default: namespace.svc.cluster.local) and forwards to upstream for other FQDNs
- Autoscales based on the size of the cluster and nodes.



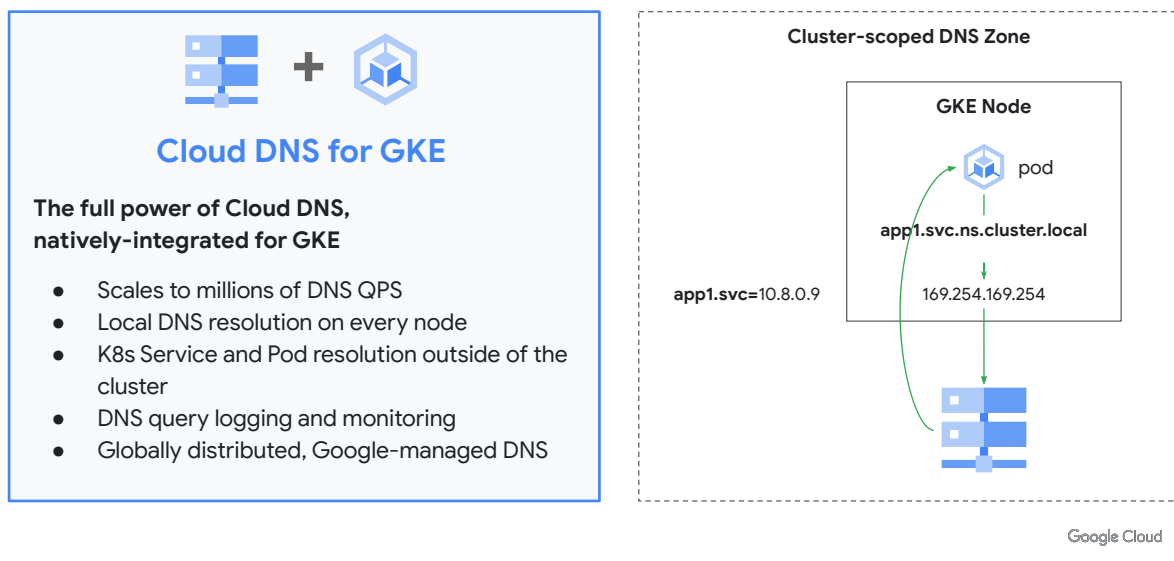
Google Cloud

This slide shows the architecture of KubeDNS. Cloud DNS is shown here as an upstream server. Pods and Services IP's are resolved via Kube-dns and everything else is routed upstream to Cloud DNS.

Key Considerations:

- Default configuration
- Kube-dns doesn't scale based on the number of requests. It scales based on a mathematical formula which takes into account the cluster size and nodes size. Kube-dns-autoscaler is responsible for autoscaling.
- Each Kube-dns can handle a max of 1500 concurrent requests.
- Kube-dns has some known limitations.
- Changes to the configuration of kube-dns-autoscaler affects the whole cluster. This is specially important to keep in mind in a multi-tenant setup


Kube-DNS vs Cloud DNS (cont.)



Cloud DNS for GKE replaces in cluster kube-dns with Cloud DNS

Key Considerations:

- Cloud DNS for GKE can be configured to be cluster scoped (similar to kube-dns) or VPC scoped.
- Cloud DNS for GKE is only used for pods and services IP's. Load Balancer IP's will have to still be registered manually with Cloud DNS.

 <https://cloud.google.com/>

Google Cloud

Using Cloud DNS for GKE

[Redacted text block]

[Redacted text block]

[Redacted text block]



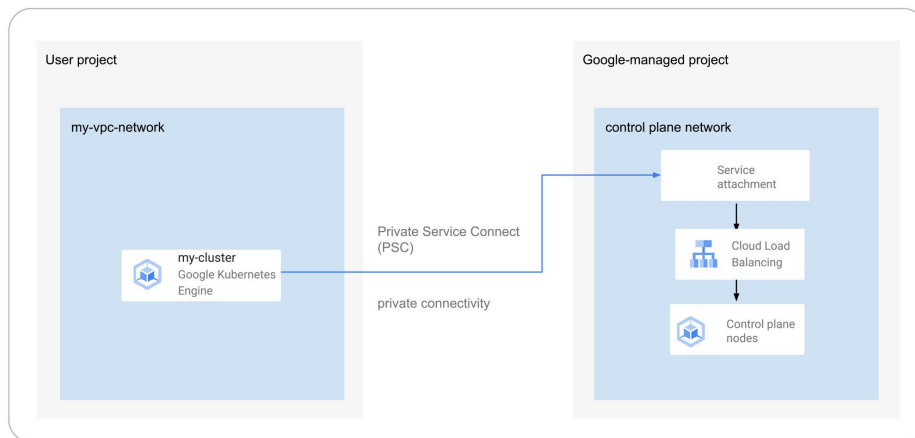
For more information visit the user guide titled 'Using Cloud DNS for GKE'.

Key decisions

- 01 What are the sizing requirements (nodes, pods, services, max pods per node) of your GKE clusters?
- 02 What are the IP ranges for your GKE clusters?
- 03 From where are your users going to connect to the control plane?
- 04 Do you need access to the Internet from the cluster?
- 05 How will you expose your services outside of the cluster?

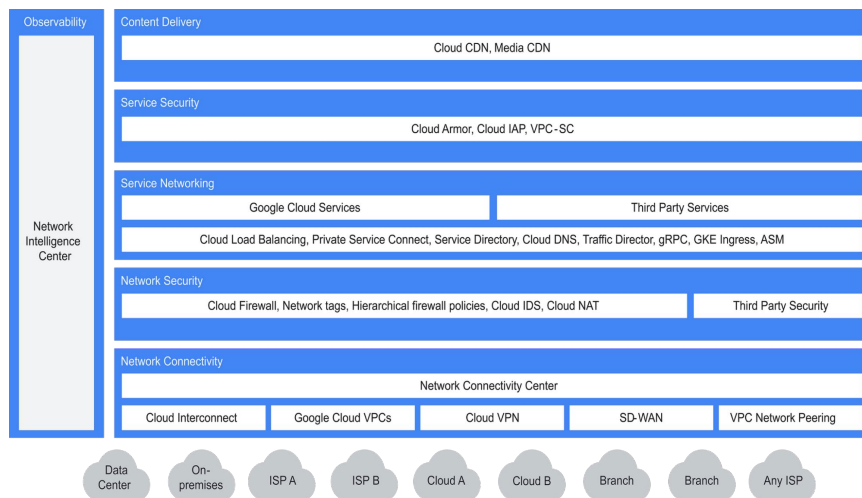
New Control Plane Connectivity for GKE

All new GKE public clusters created on or after March 15th, 2022 began using Google Cloud's **PSC (Private Service Connect)** infrastructure to communicate between the GKE cluster control plane and nodes.



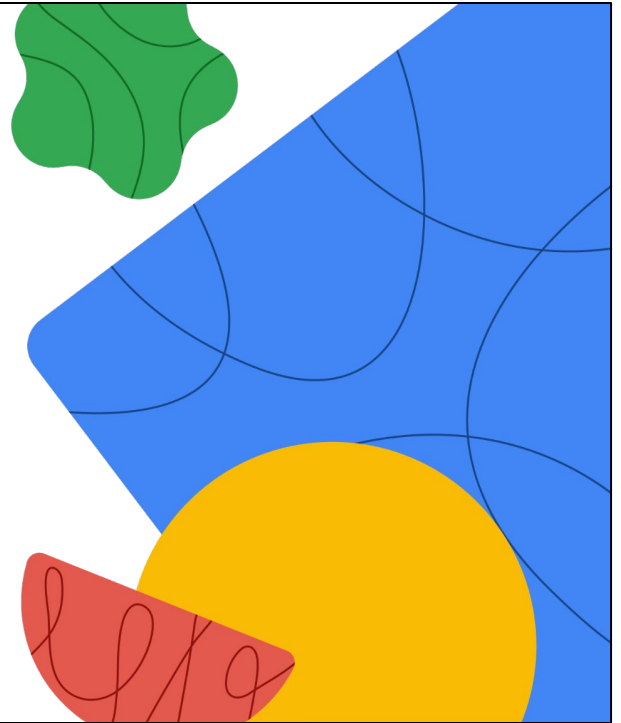
See [New Control Plane connectivity and isolation options for your GKE cluster](#)

Cloud Networking Review



See [6 Building blocks for cloud networking - Networking Architecture](#)

PCNE Exam registration and exam-taking procedures



TL;DR / Purpose of the slide:

- Subtopic slide

Key points:

- **HA and DR:** Customer is probably well familiar with these terms
- **SLI SLA SLO**
 - Considered as unique to the **cloud ecosystem** compared to on-premises data centers.
 - Normally you will **depend on them** when **architecting solutions**, so it's important to understand **what they really mean**.

Probing questions (optional):

- None



You have now completed the GKE Networking Module.