Today's agenda

01 Introduction

02 Clusters and services

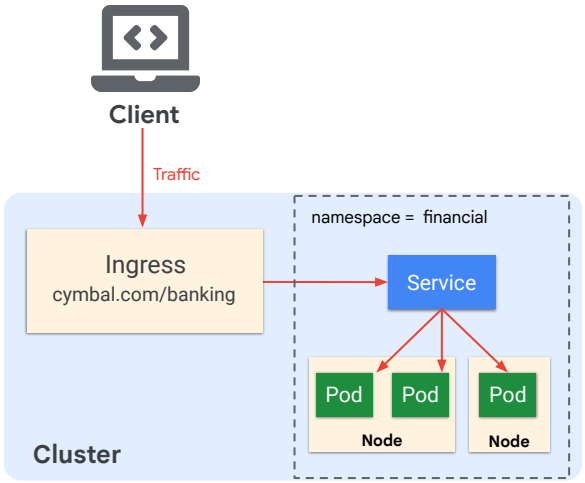
03 Lab: Creating Services and Ingress Resources

04 Pod DNS

05 Quiz

Sasha decides to use Ingress

- Sasha decides to use Ingress to make the services available.
- Ingress creates one LoadBalancer which can be used by all services in a namespace.
- This approach is cheaper than creating LoadBalancers for each service.
- Ingress creates one load balancer, which routes a client request to a service.
- The service sends traffic to a pod.



Google Cloud

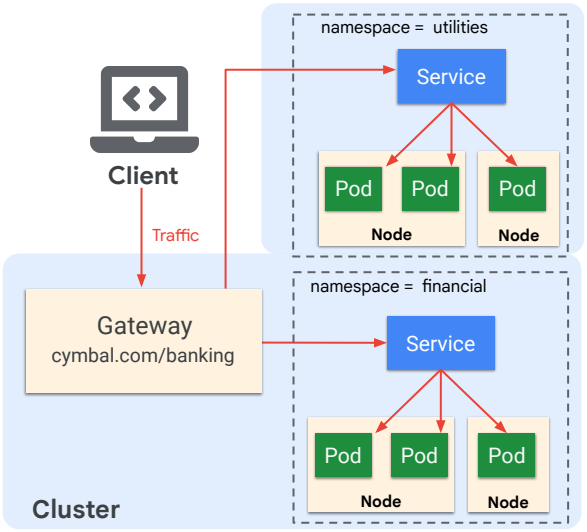
A cluster contains the main GKE infrastructure, including Kubernetes Services. Kubernetes Services allow easy communication with requests from outside of the cluster. Ingress is built on top of the Service construct.

Ingress is not a Service, or even a type of Service. It's a collection of rules that direct external inbound connections to a set of Services within the cluster. In GKE, an Ingress resource exposes these Services using a single public IP address bound to an HTTP or HTTPS load balancer provisioned within Google Cloud.

For multi-cluster deployments, you can use Multi-cluster Ingress. For more information, refer to Multi-Cluster Ingress in the Google Cloud documentation.

Gateway

- Sasha could also use Gateway to take advantage of its additional features.
- Gateway is like Ingress but also provides:
  - Increased extensibility and robustness.
  - Advanced routing features.
  - Cross-Namespace routing.
- A separate Gateway is available for single cluster or multi-cluster use.



Google Cloud

Comparison of Ingress and Gateway

Gateway and Ingress are both open source standards for routing traffic. Gateway was designed by the Kubernetes community, drawing on lessons learned from Ingress and service mesh environments. Gateway is an evolution of Ingress that provides the same function, delivered as a superset of the Ingress capabilities. Both can be used simultaneously without conflict. Over time, Gateway and Route resources will deliver more functionality not available in Ingress, compelling some organizations to use Gateway instead of Ingress.

Kubernetes Gateway as a superset of Ingress. The additional features provided by Gateway and mentioned on the slide are described in the slides that follow.

Multi-cluster Gateways

The graphic on this slide shows an example that uses Gateway with a single cluster deployment. Gateway can also be used with multiple clusters. For more information, refer to Enable multi-cluster Gateways in the Google Cloud documentation.

Gateway

Increased extensibility and robustness

- Unlike Ingress controllers, the Gateway controllers are not hosted on GKE control planes or in the user project.
- This separation enables them to be more extensible and robust.
- The Gateway controllers themselves are not a networking data plane and they do not process any traffic.
- They sit out of band from traffic and manage various data planes that process traffic.

Google Cloud

Both Gateway controllers are Google-hosted controllers that watch the Kubernetes API for GKE clusters. Unlike the GKE Ingress controller, Gateway controllers aren't hosted on GKE control planes or in user projects, enabling them to be more extensible and robust. For more information, refer to GKE Gateway Controller in the Google Cloud documentation.

The Gateway controllers themselves are not a networking data plane and they do not process any traffic. They sit out of band from traffic and manage various data planes that process traffic.

Gateway

Advanced routing features

- A Gateway can route traffic to a Kubernetes service - or to Cloud Storage buckets, Cloud Functions.
  - The routing rules are similar to the advanced traffic routing on Cloud Load Balancing.
- A Gateway also enables connections from other networks and from the public internet.

Google Cloud

Advanced traffic routing includes:

- Traffic-based autoscaling: autoscaling Pods within a Service based on HTTP requests received per second.
- Multi-cluster load balancing: the ability to load balance to Services hosted across multiple GKE clusters or multiple regions.
- Traffic splitting: explicit, weight-based traffic distribution across backends. Traffic splitting is supported with single cluster Gateways in GA.

For more information, refer to Gateway Traffic Management in the Google Cloud documentation.

Gateway

Cross-namespace routing

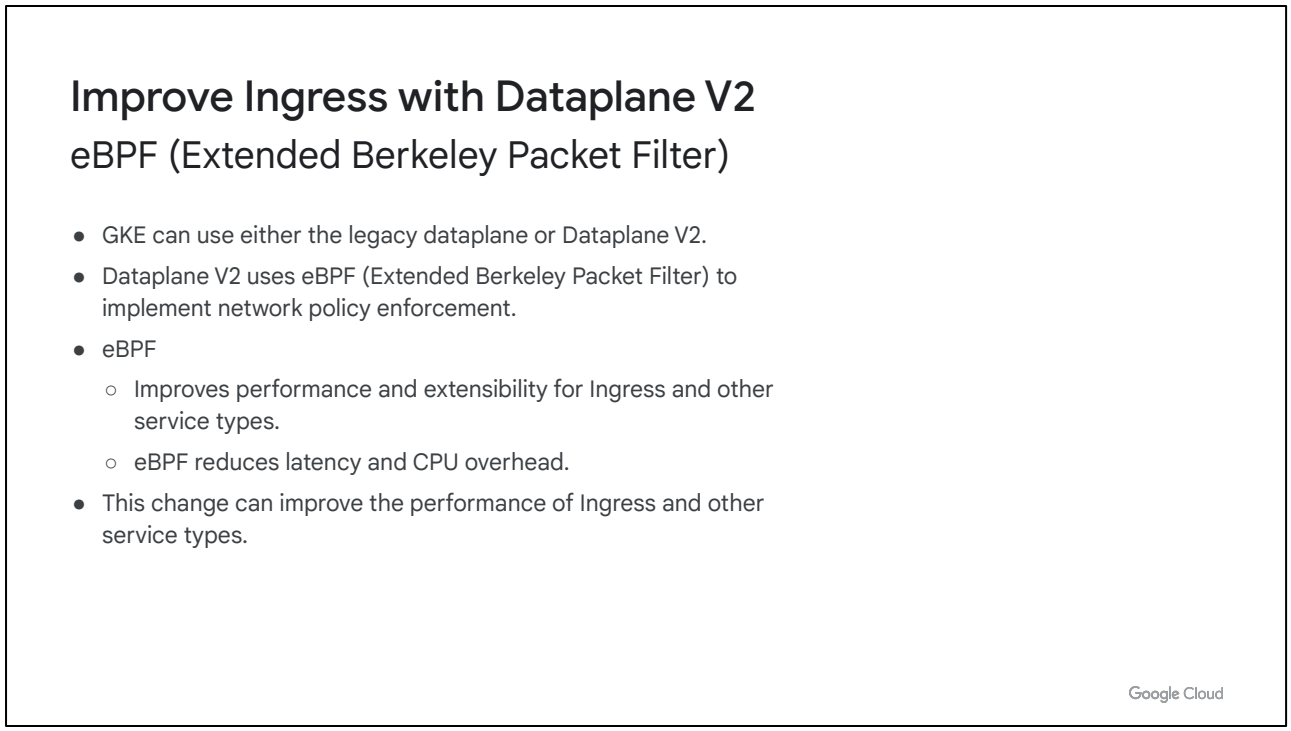
- A single-cluster gateway can be shared across namespaces.
- A multi-cluster gateway can be shared across namespaces, clusters, and regions.
- The Gateway can be a separate namespace managed by the platform team.
- The application teams manage their own namespaces.

What Sasha can do

If Sasha's team decides that it needs Gateway features - like advanced routing - they may decide to use Gateway instead of Ingress.

Google Cloud

For more information, refer to Gateway ownership and usage patterns in the Google Cloud documentation.



Dataplane V2

The choice of dataplane affects how networking functions with your cluster - for Ingress and also for other service types.

Dataplane V2 uses eBPF, a technology that allows developers to write code that runs in the kernel. Network policy enforcement is done more efficiently in Dataplane V2 than the previous dataplane. As packets arrive at a GKE node, eBPF programs installed in the kernel decide how to route and process the packets. Unlike packet processing with iptables, eBPF programs can use Kubernetes-specific metadata in the packet. This lets GKE Dataplane V2 process network packets in the kernel more efficiently and report annotated actions back to user space for logging.

Dataplane V2 uses Cilium eBPF. The previous dataplane was based on Calico.

For more information, refer to GKE Dataplane V2 in the Google Cloud documentation.

Other service types

You learned about other services types in prerequisites to this training. These service types will be summarized later in this module.

eBPF

Ingress controllers no longer need to be responsible for enforcing network policies, which can significantly improve the performance and extensibility of Ingress controllers. Dataplane V2 cluster doesn't include kube-proxy - and therefore no longer uses iptables for service routing. Dataplane V2 uses a Cilium plugin instead for service routing. The plugin is useful for scaling large clusters; iptables had been a major bottleneck. eBPF:

- Is more efficient than traditional network stack implementations, which can reduce the CPU overhead of Ingress controllers.
- Automatically configures fair queueing rules. These rules optionally allow for EDT-based Pod rate-limiting, and optimize TCP stack settings for server workloads to strike the best possible balance between the two.
- Executes in the kernel, which means that they can have lower latency than programs that are run in userspace.

Improve Ingress with Dataplane V2

Direct Server Return (DSR)

- Dataplane V2 supports Direct Server Return (DSR) mode.
- With Direct Server Return (DSR)
  - Ingress traffic goes through a load balancer to a pod.
  - Response traffic from a pod bypasses the load balancer, going directly to the source.

Google Cloud

When response traffic is greater than the request traffic, a load balancer could become a bottleneck. DSR helps to prevent that bottleneck.

Clusters

Modes of operation

GKE offers the following modes of operation for clusters:

- Autopilot mode: GKE manages the underlying infrastructure such as node configuration, autoscaling, and auto-upgrades.
- Standard mode: You manage the underlying infrastructure, including configuring the individual nodes.

Google Cloud

Autopilot and standard mode clusters are also described in *Architecting with Google Kubernetes Engine*, which is a prerequisite for this course.

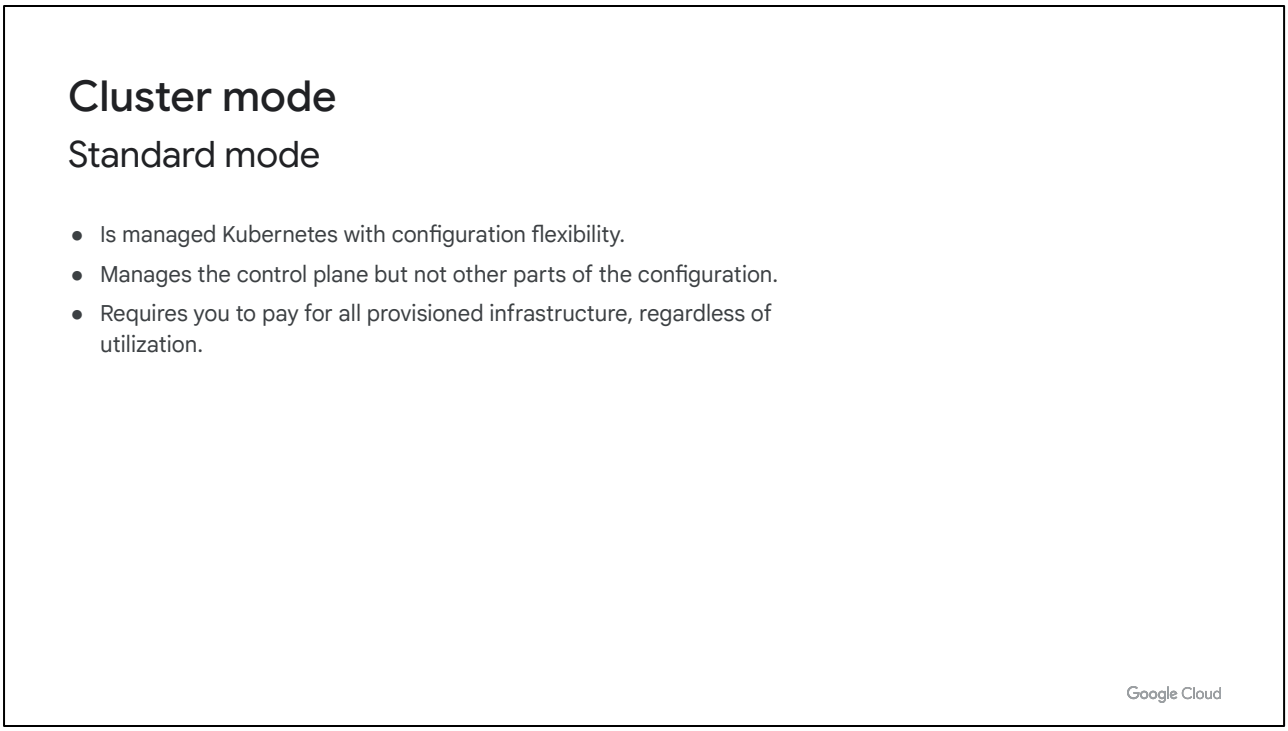
Cluster mode

Autopilot mode

- Provides optimized, managed Kubernetes, with fewer areas to configure.
  - You configure the network, node subnet, pod and service address ranges, network tags, and a few other things.
  - You don't set the control plane version, GKE release channel, the dataplane to use, or DNS provider.
- Features pod pricing - only pay for what you use.
- Features VPC-native clusters using Dataplane V2.

Cymbal uses Autopilot mode clusters for this application.

The services are in autopilot mode clusters, which use Dataplane V2. The cluster mode and Dataplane V2 affect how Ingress works, providing additional features to make Sasha's job easier.



Cluster mode was described in Architecting with Google Kubernetes Engine, which is a prerequisite for this course.

In Standard mode, you manage every configuration setting in your cluster and nodes, including managing groups of nodes called node pools that share characteristics. Google still manages your control plane, but you manage and configure:

- Node pools: You create and manage groups of nodes that have similar configuration settings.
- Security: Standard clusters have default hardening measures applied, but many GKE security features are not enabled by default, such as Workload Identity and Shielded GKE Nodes. You can enable these features manually and configure the settings.
- Scheduling: You monitor and design workloads so that GKE can schedule them efficiently to minimize unused resources (bin-packing).
- Scaling: Set up and configure node auto-provisioning, configure automatic scaling settings, and ensure that your nodes don't have too many resources or too few resources.
- Resource management: You must evaluate the resource needs of each workload that you run on standard mode clusters to ensure that the resource requests meet the workload requirements.
- Version management: Best practices such as automatic GKE version upgrades and release channel enrollment are off by default in standard mode

- clusters. You can configure auto-upgrades and GKE versions when you create or update the cluster.

Dataplane V2 and standard mode clusters

- If you're using a standard mode cluster, you can't enable Dataplane V2.
- You will need to
  1. Create a new standard mode cluster and explicitly enable Dataplane V2.
  2. Redeploy the application to the new cluster.

Google Cloud

Suppose you have a standard mode cluster that uses the legacy dataplane. If you use Google Cloud Console to try to edit the cluster settings, you'll see that you can't change the dataplane. You must create a new standard cluster and enable Dataplane V2. For that reason, you must also redeploy your application to the new cluster.

For more information about Dataplane V2, refer to Using GKE Dataplane V2 in the Google Cloud documentation.

Making microservices more visible

- One type of pod in Cymbal's mortgages application is having intermittent communication problems with other pods.
- The mortgages application runs on a cluster in standard mode.
- Sasha has been asked to troubleshoot the communications problems.
- Sasha looks at the VPC logs and does not see any information about internal communication between pods.
- What can Sasha do to initiate troubleshooting?

What Sasha can do

In order to view communication between nodes in a cluster, Sasha must enable:

- Intranode visibility in the cluster.
- VPC flow logs in the cluster subnetwork.

Sasha may also find it helpful to enable firewall rule logging.

Note

Intranode visibility node is enabled in autopilot mode clusters by default.

Google Cloud

For more information about intranode visibility, refer to Setting up intranode visibility in the Google Cloud documentation.

With intranode visibility and VPC flow logs enabled, you will be able to see traffic sent between pods and processed by VPC network. However, without enabling firewall rule logging, if a firewall rule denies traffic Sasha won't see anything in the logs. Enabling firewall rule logging will help Sasha to discover when a firewall rule blocked the communication between the pods.

Routes-based clusters and latency issues

- Artichoke, an application owned by a Cymbal subsidiary company, is experiencing high latency.
- The application uses routes-based clusters.
- Sasha has been asked to fix the latency problem.
- Before considering what Sasha can do to troubleshoot the latency issue, let's first review the GKE cluster types.

Cluster type

Routes-based

- Routes-based clusters use static routes to route traffic to pods.
- Each pod is assigned a static IP address VPC network that the cluster is using.
- When a client sends a request to the pod IP address, the traffic is routed to the pod.
- The GKE control plane maintains the routes to the pods.

Cluster type

VPC-native

- A pod in a VPC-native cluster doesn't have a static IP address.

  The pod IP address is taken from the cluster subnet's secondary IP address range.
- Pod traffic from external sources doesn't need to go through the control plane - which can decrease latency.
- In the Artichoke application, Sasha decides to use VPC-native clusters.

Google Cloud

Cluster types are modifiable.

Setting up a new VPC-native cluster

1. Create a VPC network and subnet for your cluster.
2. In the subnet, create a secondary IP address range for your pods.
3. Create a new cluster; VPC-native is enabled by default.

What Sasha can do

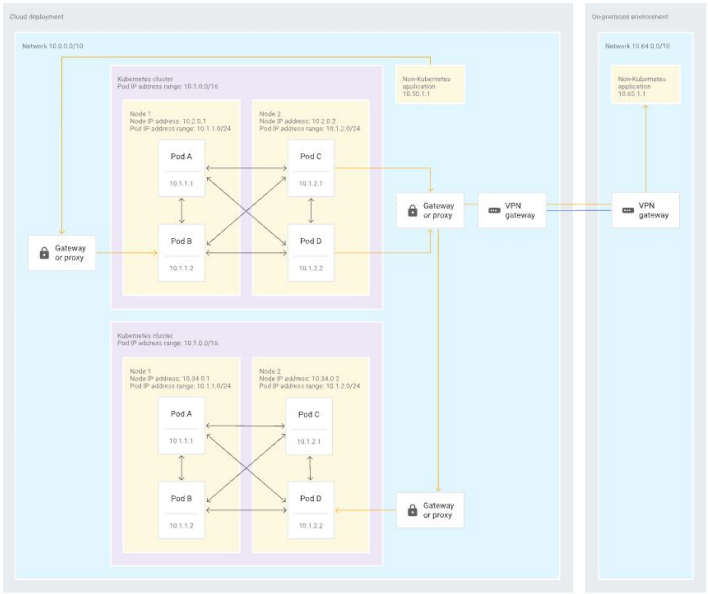
Since Sasha can't change the cluster type, they must:

1. Create a new VPC-native cluster.
2. Deploy applications from the routes-based cluster to the new cluster.
3. Test to make sure the applications run correctly on the new cluster.
4. Delete the routes-based cluster.

You can't change the cluster type, so Sasha created a new cluster.

Fully integrated network model

- GKE uses a fully integrated network model.
- Clusters are deployed in a VPC network that can also contain other applications.
- This model offers ease of communications with applications outside Kubernetes and in other Kubernetes clusters.



Google Cloud

The fully integrated network (or flat) model offers ease of communications with applications outside Kubernetes and in other Kubernetes clusters. Major cloud service providers commonly implement this model because those providers can tightly integrate their Kubernetes implementation with their software-defined network (SDN).

When you use the fully integrated model, the IP addresses that you use for Pods are routed within the network in which the Kubernetes cluster sits. Also, the underlying network knows on which node the Pod IP addresses are located. In many implementations, Pod IP addresses on the same node are from a specific, pre-assigned Pod IP address range. But this pre-assigned address range is not a requirement.

Refer to Fully integrated network model in the Google Cloud documentation for more information.

Other networking models

Island-mode network model

- This model can be used for on-premises Kubernetes implementations
- Pods in the cluster use a gateway to communicate to resources outside the cluster.

Isolated network model

- Is used for clusters that access the larger corporate network only through public-facing APIs.
- Each Kubernetes cluster is isolated and can't use internal IP addresses to communicate with the rest of the network.
- The cluster sits on its own private network.

Google Cloud

For more information, refer to the Island-mode network model and the Isolated network model in the Google Cloud documentation.