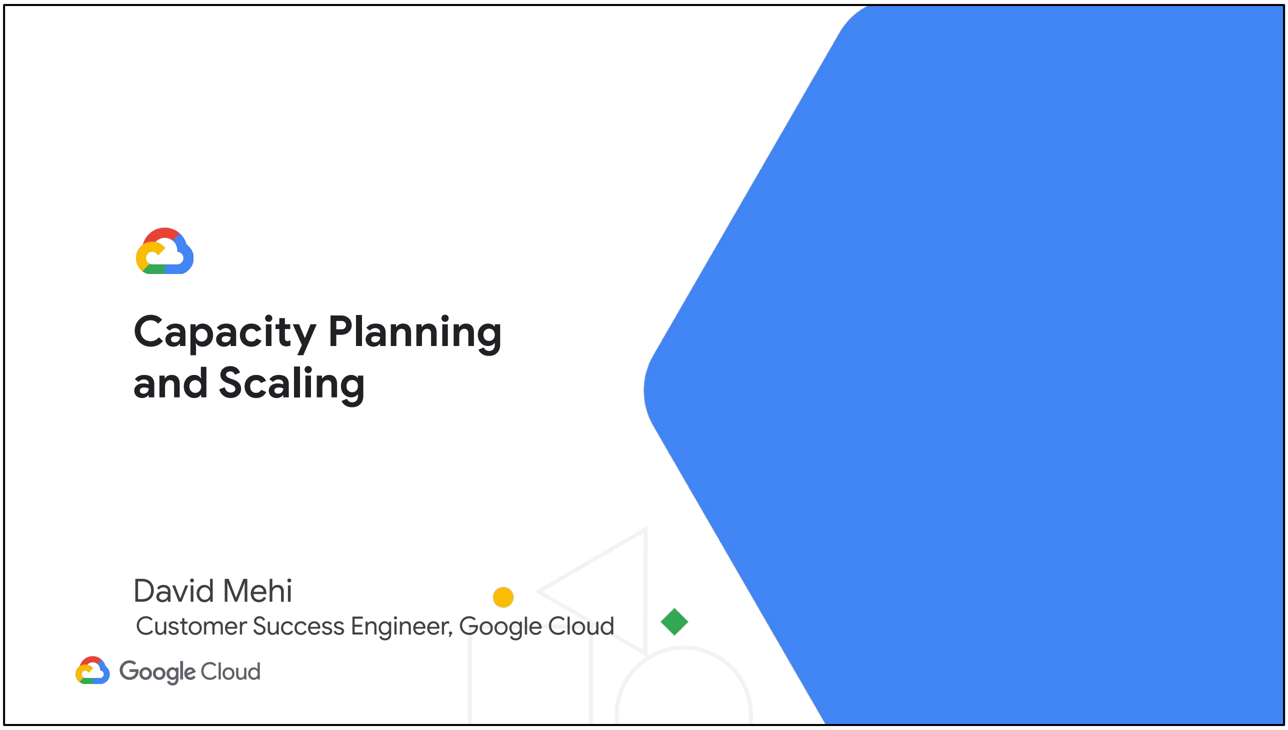
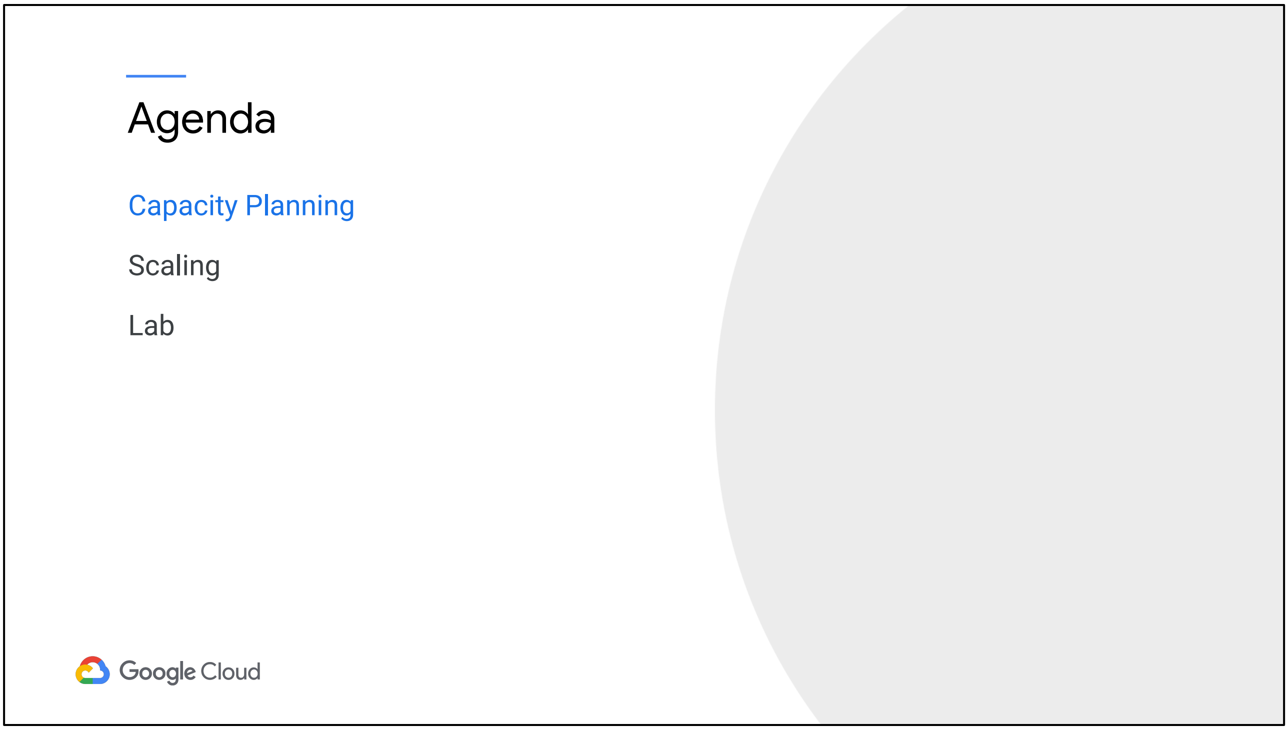


Capacity Planning and Scaling

David Mehi
Customer Success Engineer, Google Cloud

Google Cloud

In this module, you will learn about capacity planning for Apigee hybrid and how you can scale the runtime plane to support higher throughput for your API proxies.



Agenda

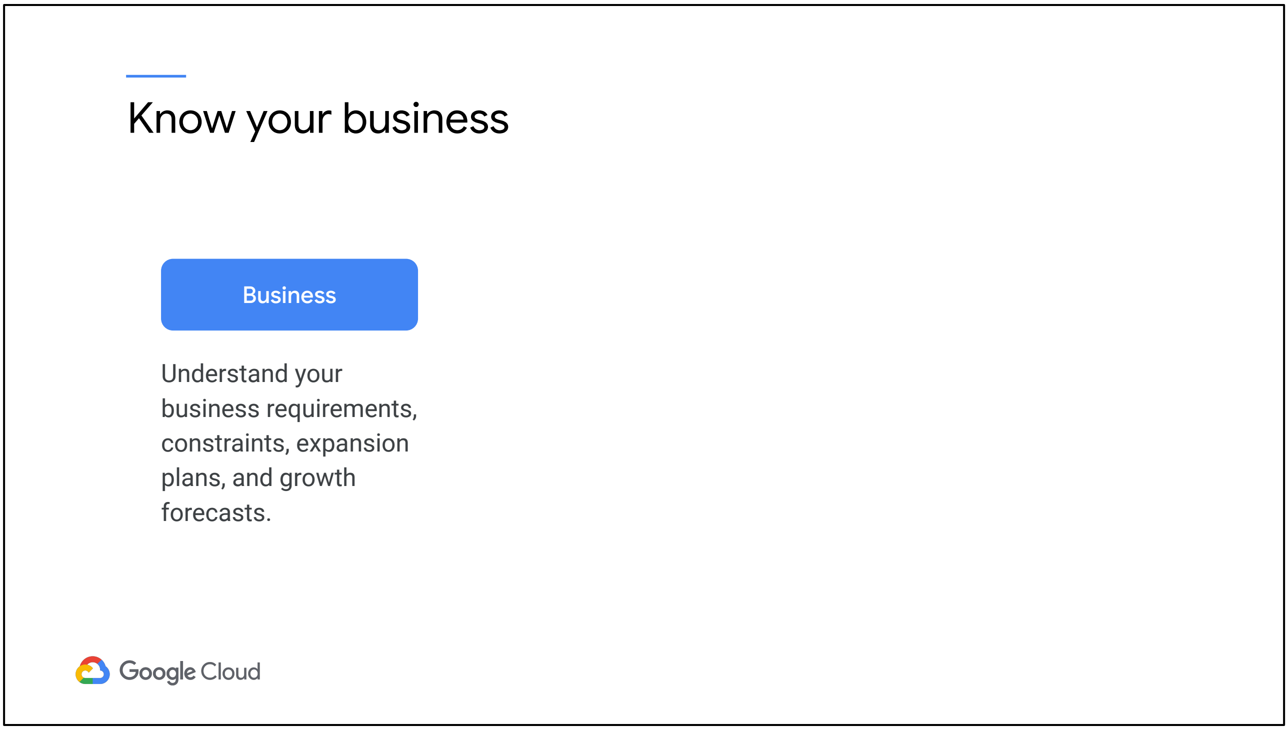
Capacity Planning

Scaling

Lab

Let’s discuss capacity planning for Apigee hybrid.

You will learn how to scale the runtime plane in a later lecture and learn to scale your runtime plane components in a lab.



Know your business

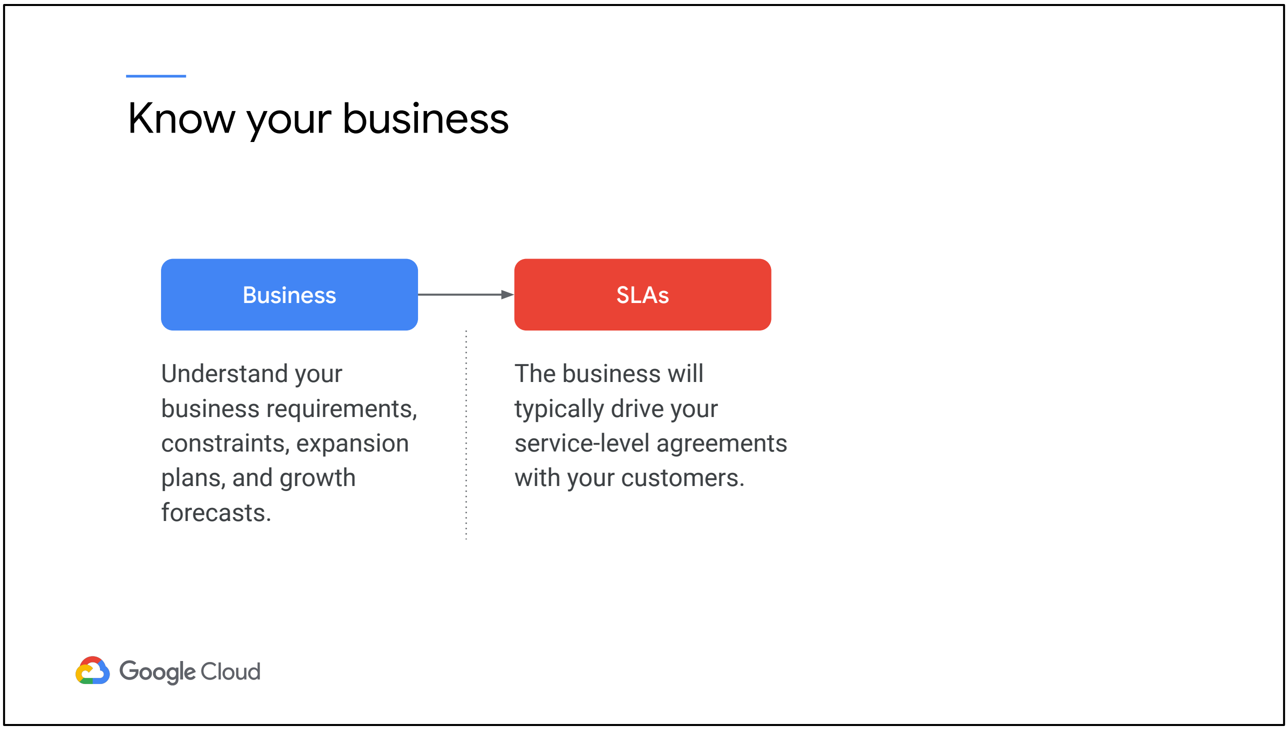
Business

Understand your business requirements, constraints, expansion plans, and growth forecasts.

Google Cloud

One of the fundamental concepts when planning for capacity of your Apigee hybrid platform is to understand your business requirements.

The business provides inputs into transaction volume and processing throughput for both current needs and future growth, taking into account any expansion plans.



Know your business

Business → **SLAs**

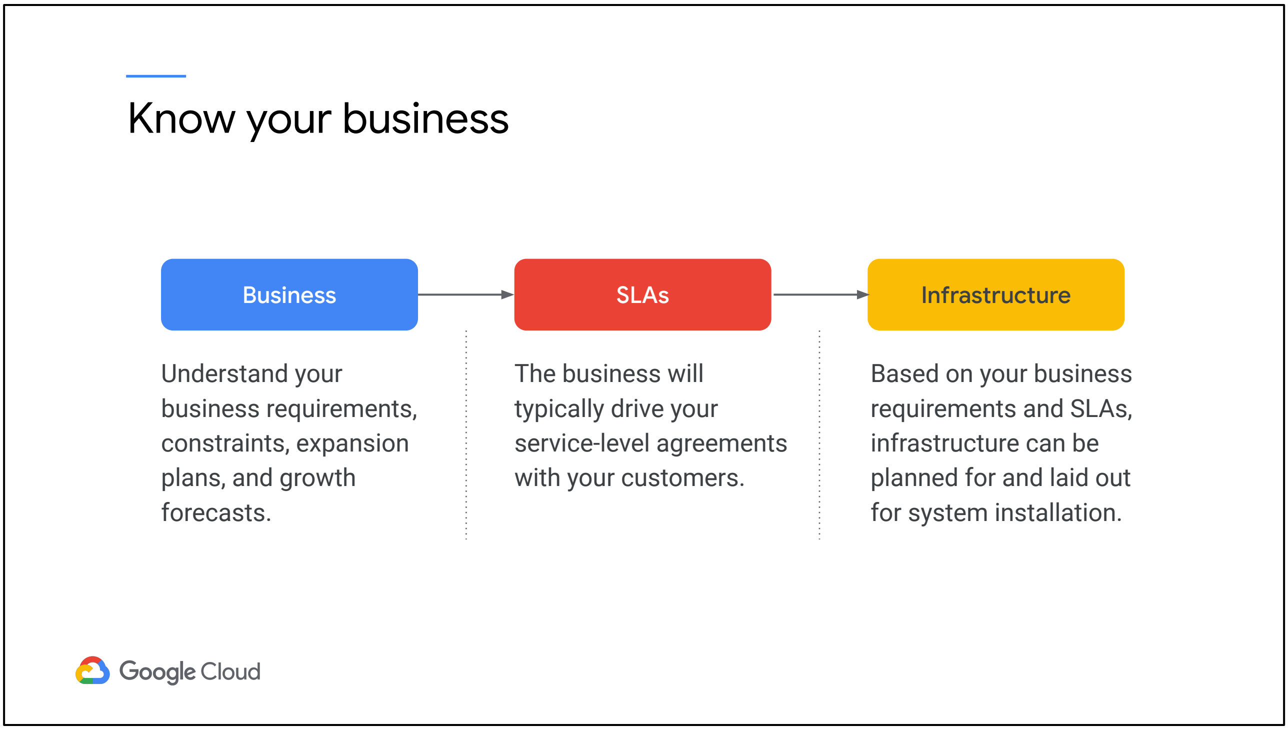
Understand your business requirements, constraints, expansion plans, and growth forecasts.

The business will typically drive your service-level agreements with your customers.

Google Cloud

Business requirements typically drive service level agreements that you set with your customers.

SLAs include service availability in terms of uptime percentage, response times, problem resolution time, etc.



Know your business

Business

Understand your business requirements, constraints, expansion plans, and growth forecasts.

SLAs

The business will typically drive your service-level agreements with your customers.

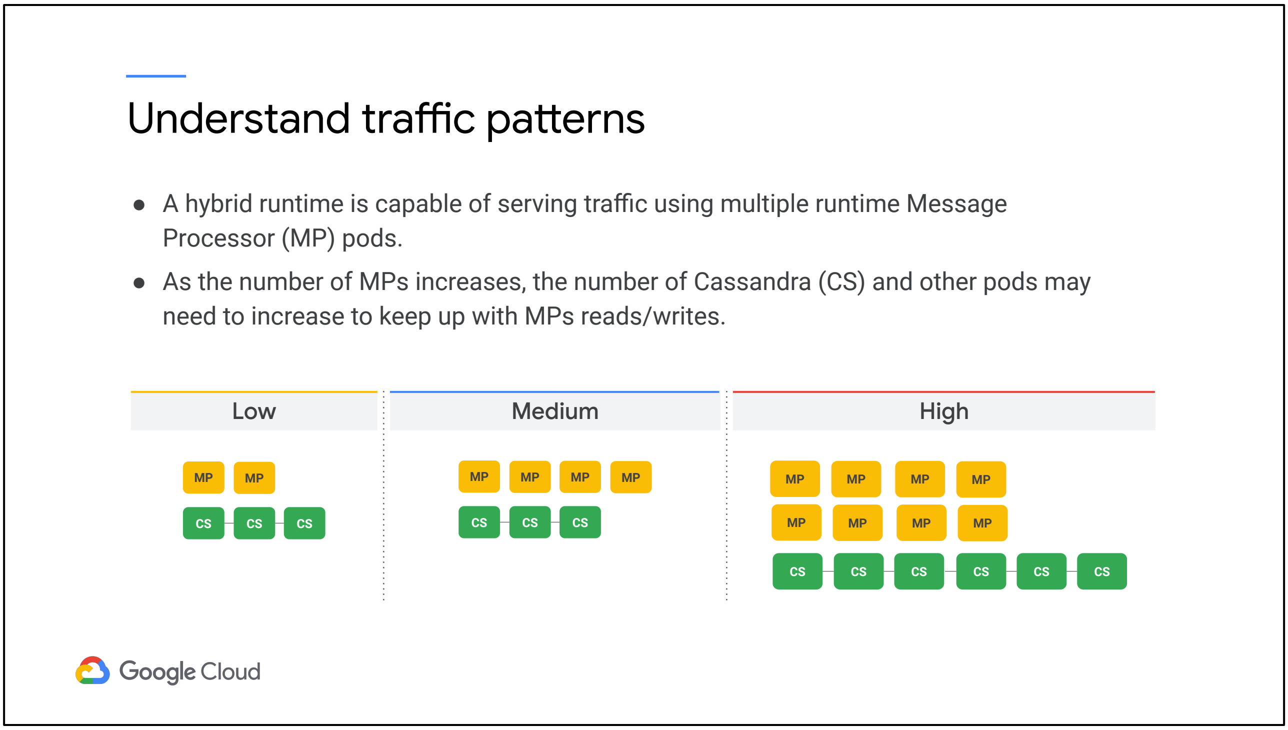
Infrastructure

Based on your business requirements and SLAs, infrastructure can be planned for and laid out for system installation.

Google Cloud

Business requirements and service level agreements drive your infrastructure requirements.

This helps in capacity planning for your Apigee hybrid installation.



Understand traffic patterns

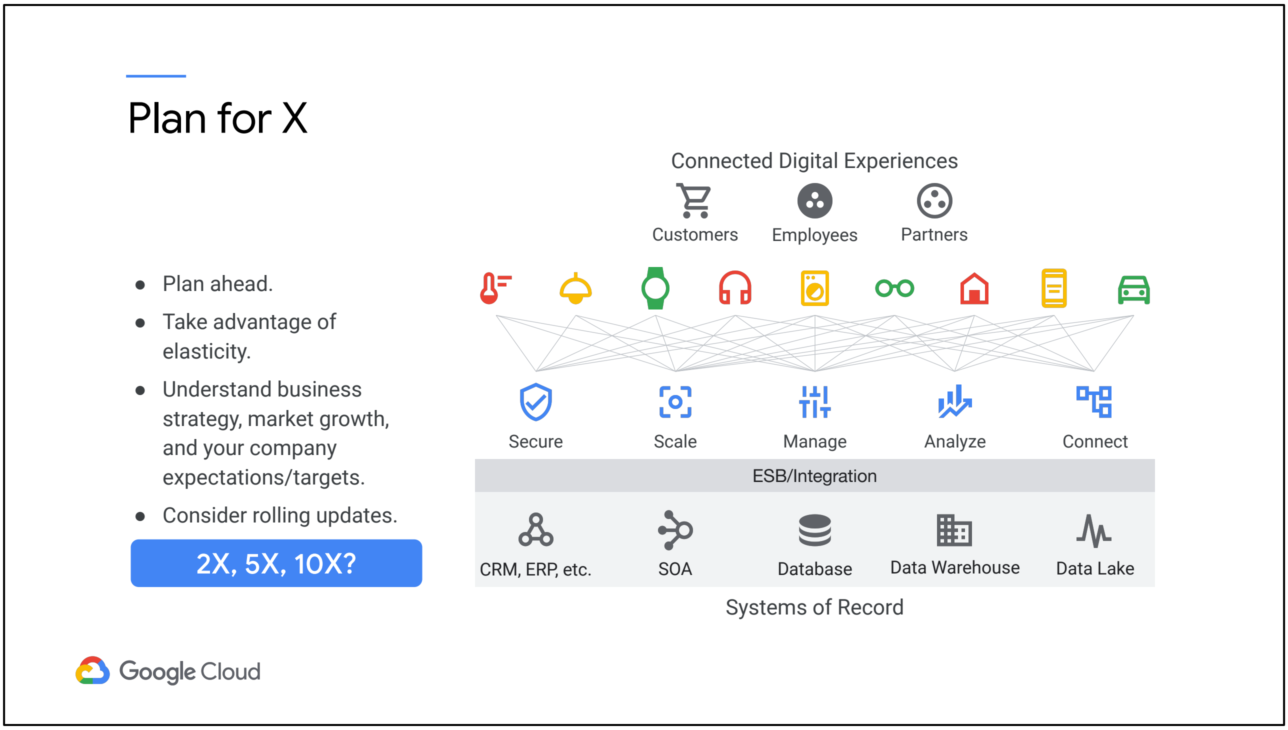
- A hybrid runtime is capable of serving traffic using multiple runtime Message Processor (MP) pods.
- As the number of MPs increases, the number of Cassandra (CS) and other pods may need to increase to keep up with MPs reads/writes.

Google Cloud

It is important to understand your API traffic patterns when planning your hybrid runtime installation.

The number of message processors or runtime pods will scale based on CPU usage and the amount of traffic processed.

The Cassandra pods in your cluster might also need to scale to handle the increased traffic volume.



Plan for X

- Plan ahead.
- Take advantage of elasticity.
- Understand business strategy, market growth, and your company expectations/targets.
- Consider rolling updates.

2X, 5X, 10X?

Connected Digital Experiences

Customers

Employees

Partners

Secure

Scale

Manage

Analyze

Connect

ESB/Integration

CRM, ERP, etc.

SOA

Database

Data Warehouse

Data Lake

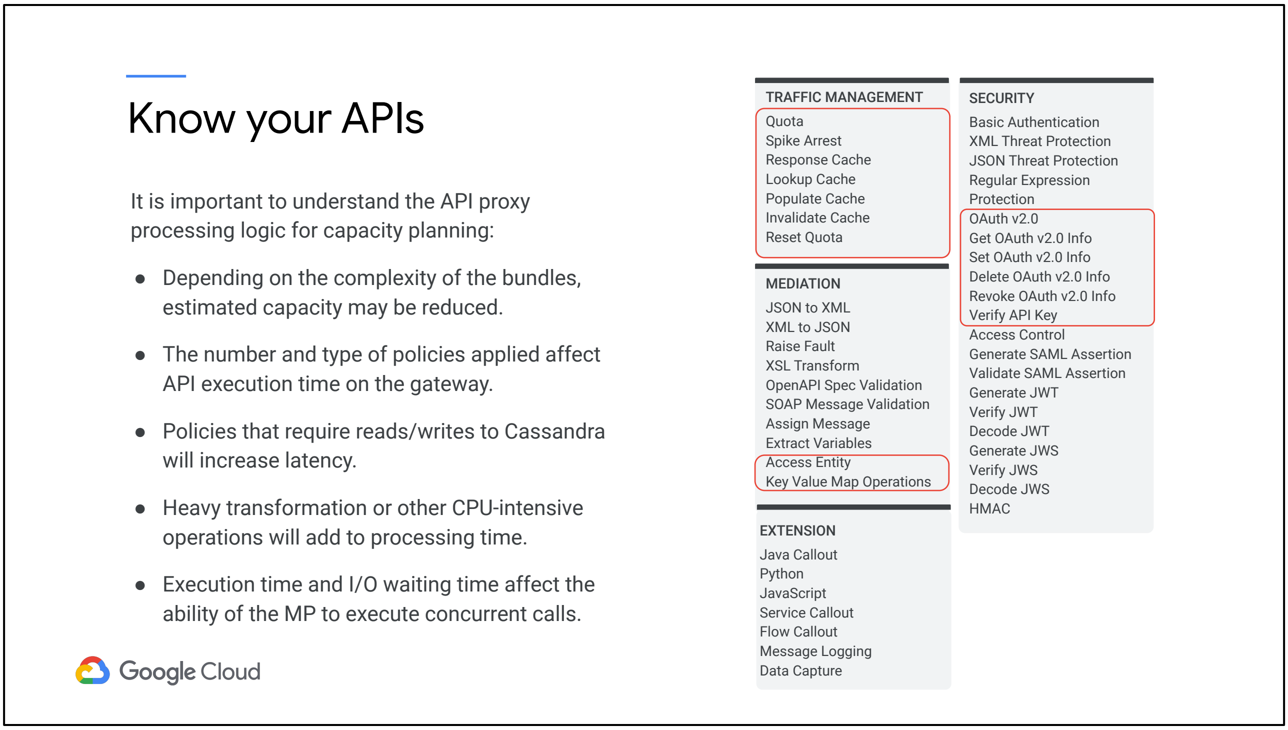
Systems of Record

Google Cloud

As mentioned earlier, understanding your business strategy, growth plans, and target expectations is key to planning capacity for your hybrid installation.

It’s important to plan ahead when determining the amount of API traffic your runtime would be expected to handle.

Also, consider rolling updates of the runtime components in your cluster. You must have sufficient cluster resources available to create new pods with the updated containers during rolling updates.



Know your APIs

It is important to understand the API proxy processing logic for capacity planning:

- Depending on the complexity of the bundles, estimated capacity may be reduced.
- The number and type of policies applied affect API execution time on the gateway.
- Policies that require reads/writes to Cassandra will increase latency.
- Heavy transformation or other CPU-intensive operations will add to processing time.
- Execution time and I/O waiting time affect the ability of the MP to execute concurrent calls.

TRAFFIC MANAGEMENT

Quota
Spike Arrest
Response Cache
Lookup Cache
Populate Cache
Invalidate Cache
Reset Quota

MEDIATION

JSON to XML
XML to JSON
Raise Fault
XSL Transform
OpenAPI Spec Validation
SOAP Message Validation
Assign Message
Extract Variables
Access Entity
Key Value Map Operations

EXTENSION

Java Callout
Python
JavaScript
Service Callout
Flow Callout
Message Logging
Data Capture

SECURITY

Basic Authentication
XML Threat Protection
JSON Threat Protection
Regular Expression Protection
OAuth v2.0
Get OAuth v2.0 Info
Set OAuth v2.0 Info
Delete OAuth v2.0 Info
Revoke OAuth v2.0 Info
Verify API Key
Access Control
Generate SAML Assertion
Validate SAML Assertion
Generate JWT
Verify JWT
Decode JWT
Generate JWS
Verify JWS
Decode JWS
HMAC

Google Cloud

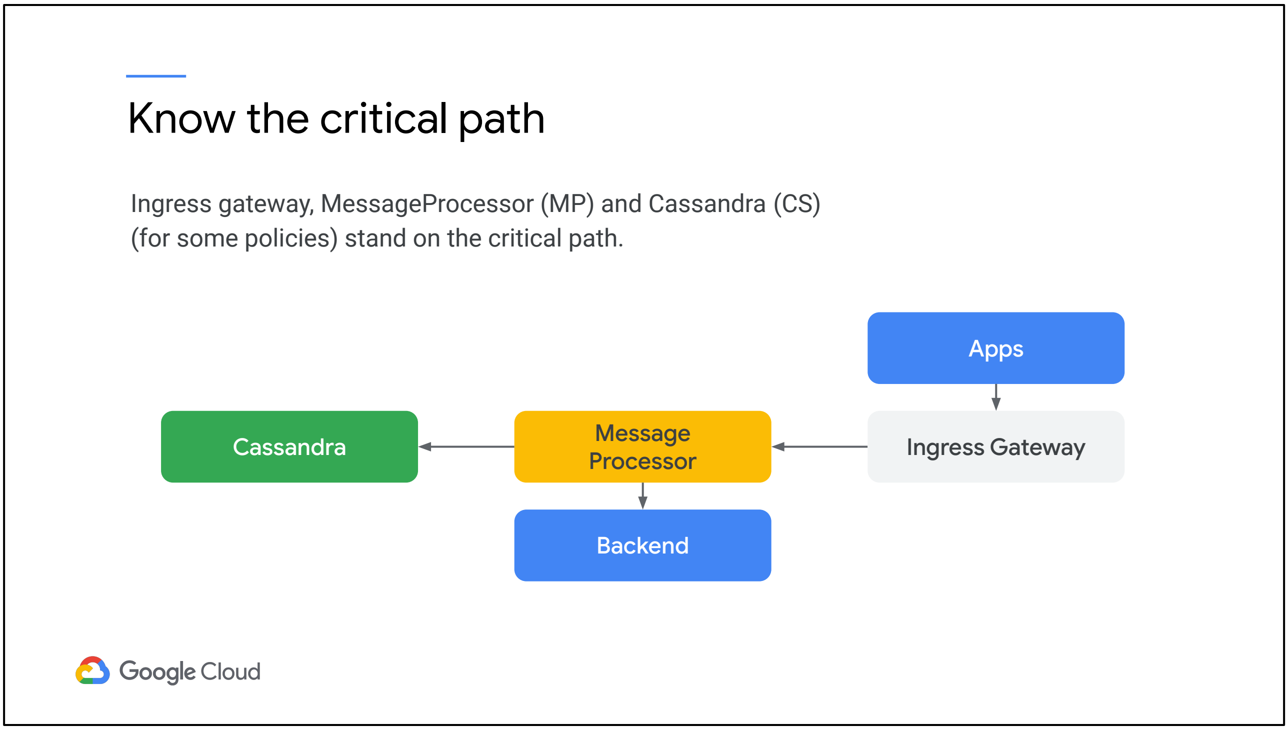
When considering API throughput and latency for capacity planning purposes, you must have a good understanding of proxy logic.

The number of API proxies in a hybrid environment and the complexity of the API processing logic may increase your response latency.

Also, API proxies that require reading and writing to the runtime Cassandra data store increase proxy execution time.

Policy types that involve Cassandra lookups and updates, such as OAuth, Quota, and KVM, will add to proxy latency.

You can use cache policies in your API proxies to improve proxy latency.



Know the critical path

Ingress gateway, MessageProcessor (MP) and Cassandra (CS)
(for some policies) stand on the critical path.

Apps

Ingress Gateway

Message Processor

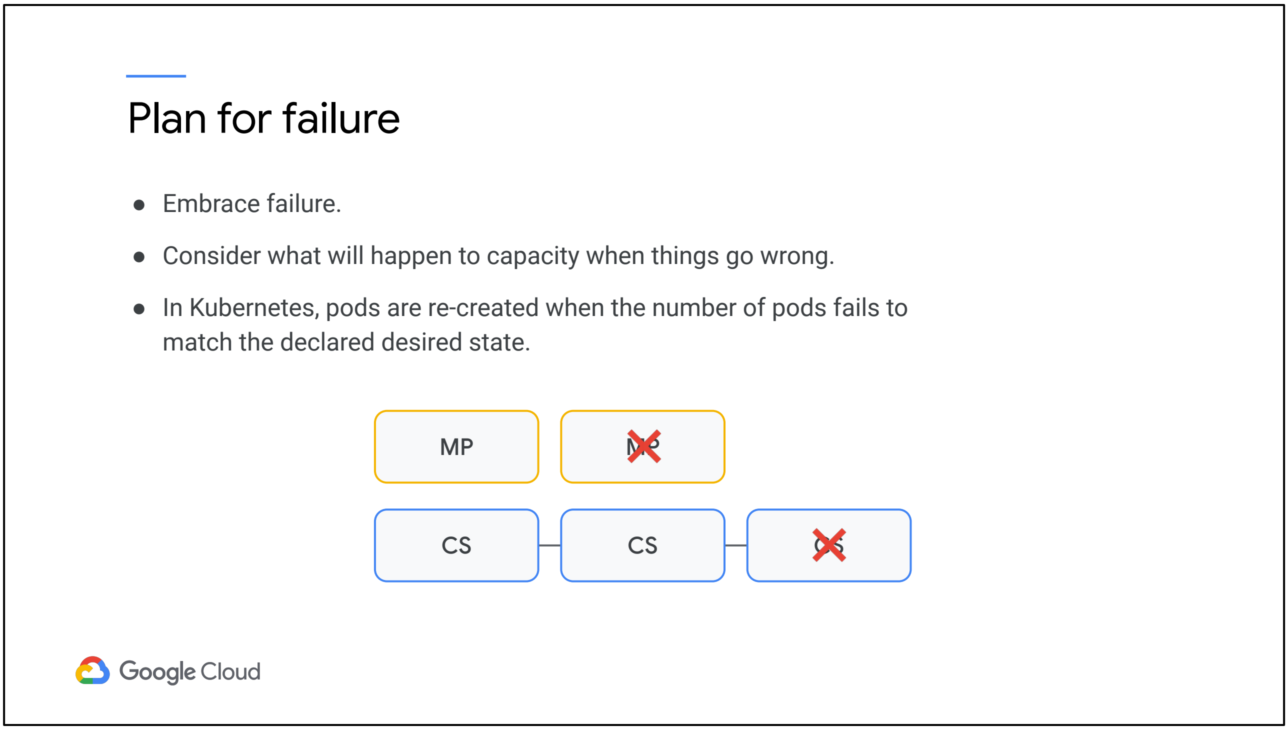
Cassandra

Backend

It is important to identify which Apigee hybrid components are in the critical path of execution of an API proxy.

The runtime message processor receives requests from client apps via the ingress gateway and processes them by executing the logic in the API proxy.

The Cassandra component also belongs on the critical path for API proxies that need read or write access to runtime data.



Plan for failure

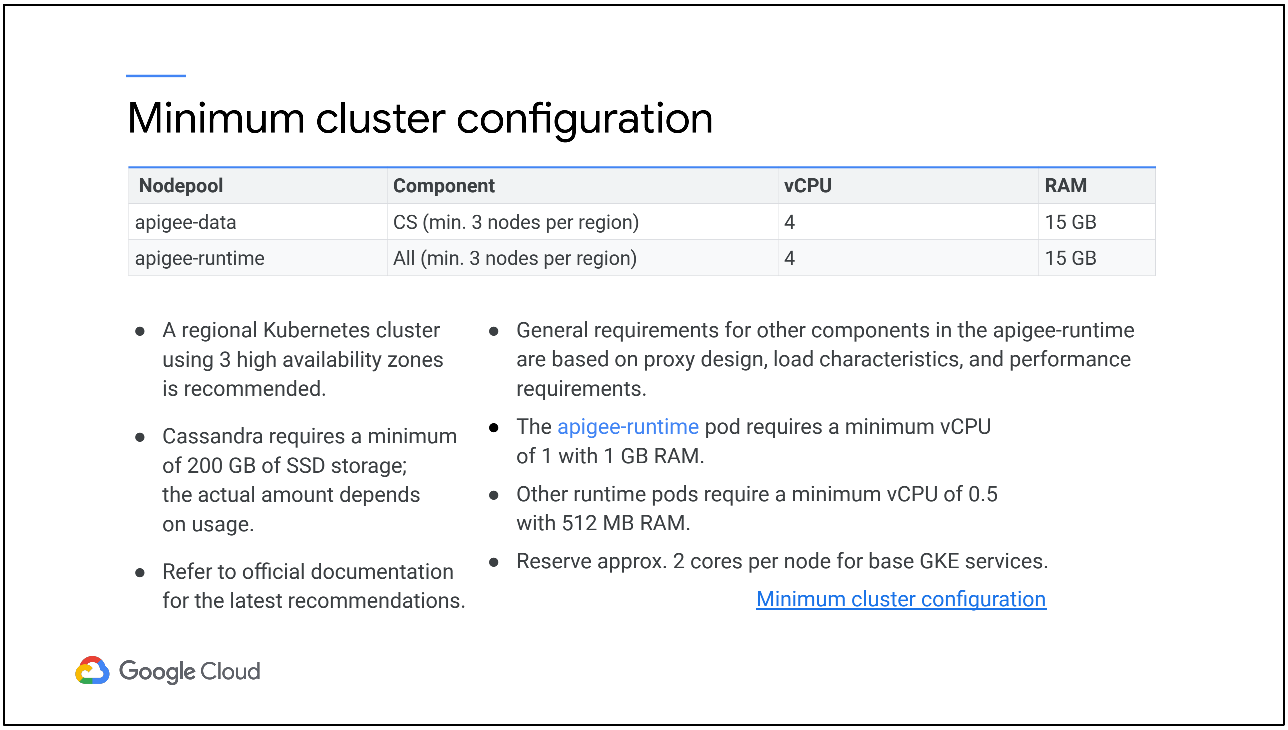
- Embrace failure.
- Consider what will happen to capacity when things go wrong.
- In Kubernetes, pods are re-created when the number of pods fails to match the declared desired state.

MP
~~MP~~
CS
CS
~~CS~~

Component failure must always be considered as a factor when planning your runtime capacity.

You should test how your runtime plane handles your API traffic load under reduced capacity and adjust your cluster configuration accordingly.

In a containerized environment like Kubernetes, the failed pods are re-created automatically and take some time to boot up before they are available to handle runtime traffic.



Minimum cluster configuration

Nodepool	Component	vCPU	RAM
apigee-data	CS (min. 3 nodes per region)	4	15 GB
apigee-runtime	All (min. 3 nodes per region)	4	15 GB

- A regional Kubernetes cluster using 3 high availability zones is recommended.
- Cassandra requires a minimum of 200 GB of SSD storage; the actual amount depends on usage.
- Refer to official documentation for the latest recommendations.

- General requirements for other components in the apigee-runtime are based on proxy design, load characteristics, and performance requirements.
- The apigee-runtime pod requires a minimum vCPU of 1 with 1 GB RAM.
- Other runtime pods require a minimum vCPU of 0.5 with 512 MB RAM.
- Reserve approx. 2 cores per node for base GKE services.

Minimum cluster configuration

Google Cloud

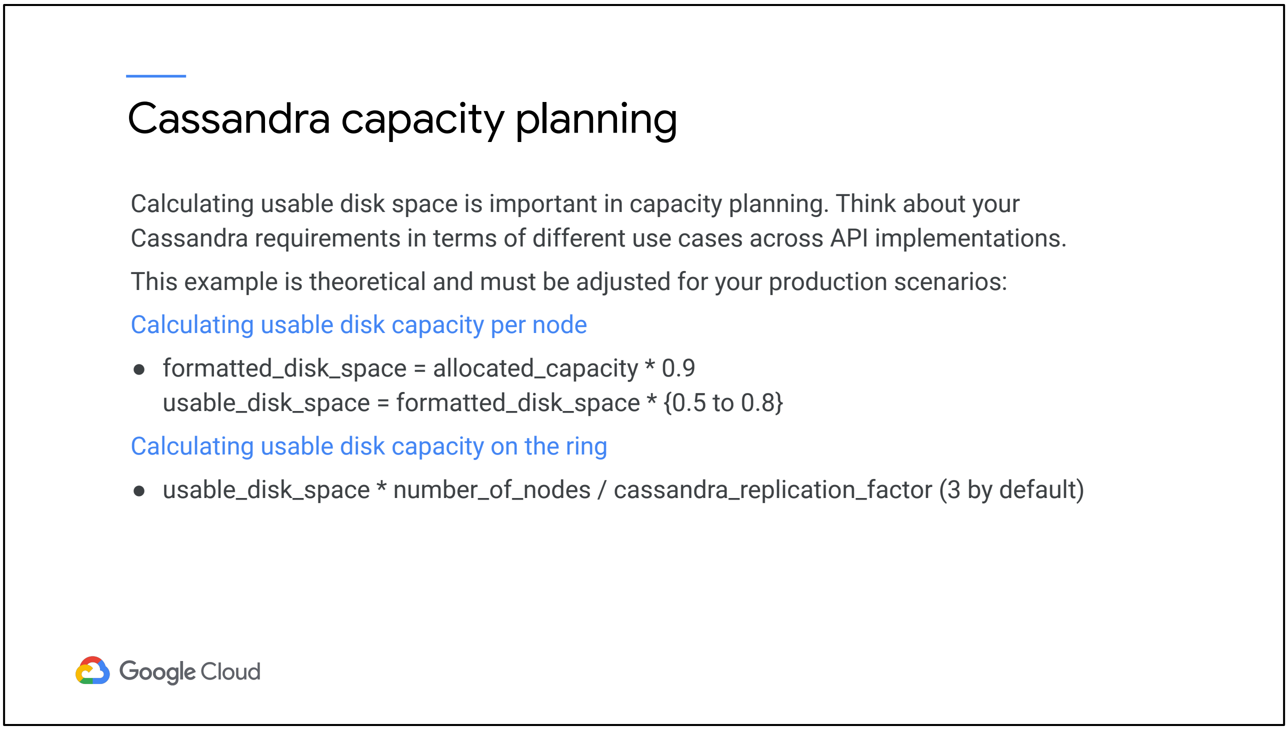
Here are some general requirements for your Apigee hybrid runtime plane installation on Google Kubernetes Engine.

Apigee recommends creating two nodepools: one for the runtime Cassandra data store, and the other nodepool for all of the other hybrid runtime components.

The Cassandra data store requires a minimum of 200 to 500 GB SSD storage per pod, depending on your proxy usage.

A regional cluster with 3 availability zones is recommended.

It is always a good practice to refer to the official hybrid documentation for the latest sizing recommendations.



Cassandra capacity planning

Calculating usable disk space is important in capacity planning. Think about your Cassandra requirements in terms of different use cases across API implementations.

This example is theoretical and must be adjusted for your production scenarios:

Calculating usable disk capacity per node

- formatted_disk_space = allocated_capacity * 0.9
  usable_disk_space = formatted_disk_space * {0.5 to 0.8}

Calculating usable disk capacity on the ring

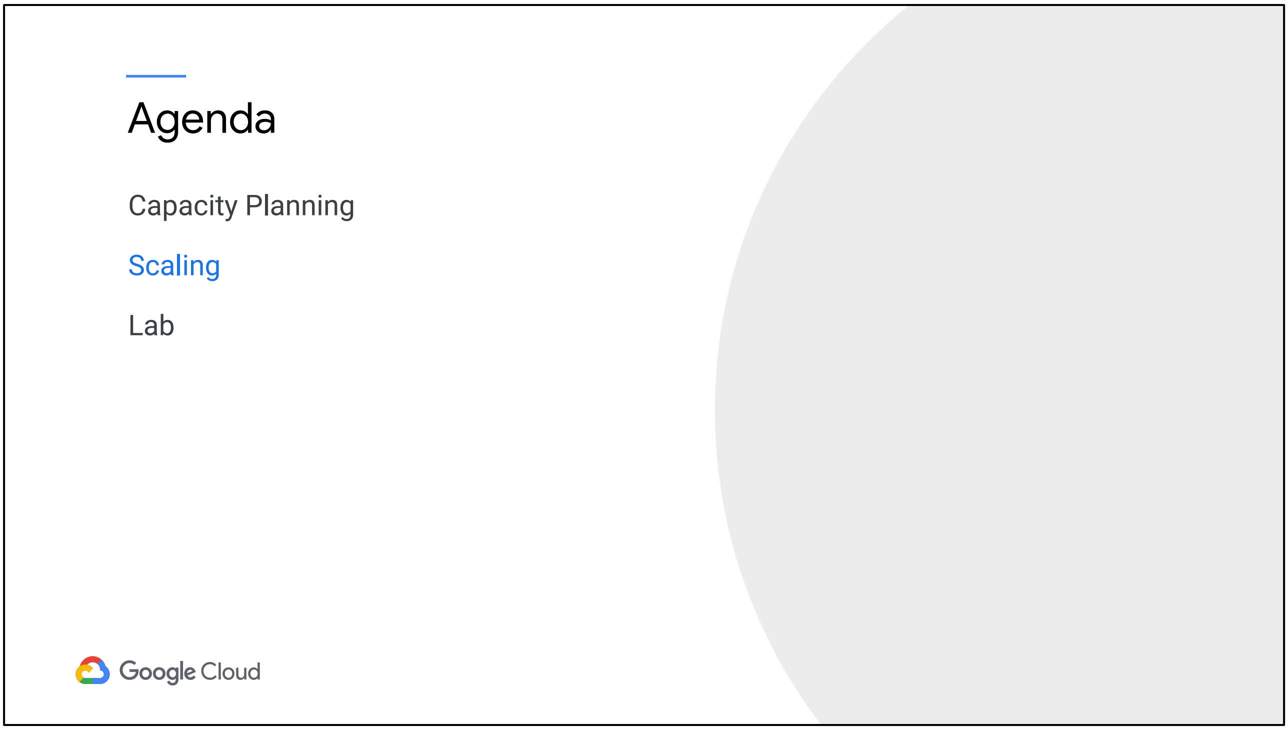
- usable_disk_space * number_of_nodes / cassandra_replication_factor (3 by default)

While determining your Cassandra storage requirements, take into account the usable disk capacity.

This is typically around 50 to 70 percent of your allocated storage capacity that is available for use. This is because, during normal operations, Cassandra requires disk capacity for compaction and repair operations.

Also note that, because data is replicated, the total usable disk capacity is a function of the number of nodes in the ring divided by the replication factor.

Disk space calculation formula: Calculating usable disk capacity



Agenda

Capacity Planning

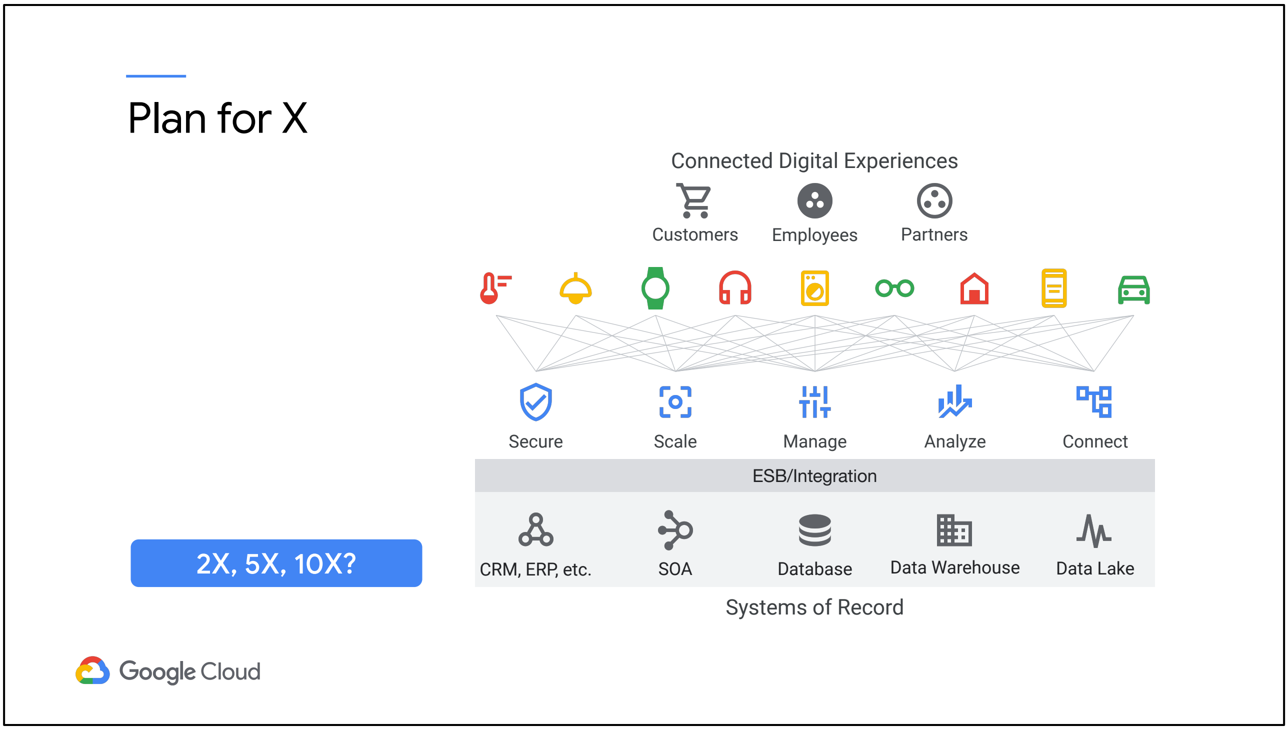
Scaling

Lab

Google Cloud

In this lecture, we will discuss how the Apigee hybrid runtime components can be scaled to handle your API runtime traffic.

Scaling is an integral feature of Kubernetes that can be configured for your hybrid runtime cluster.



Plan for X

Connected Digital Experiences

Customers Employees Partners

Secure Scale Manage Analyze Connect

ESB/Integration

CRM, ERP, etc. SOA Database Data Warehouse Data Lake

Systems of Record

2X, 5X, 10X?

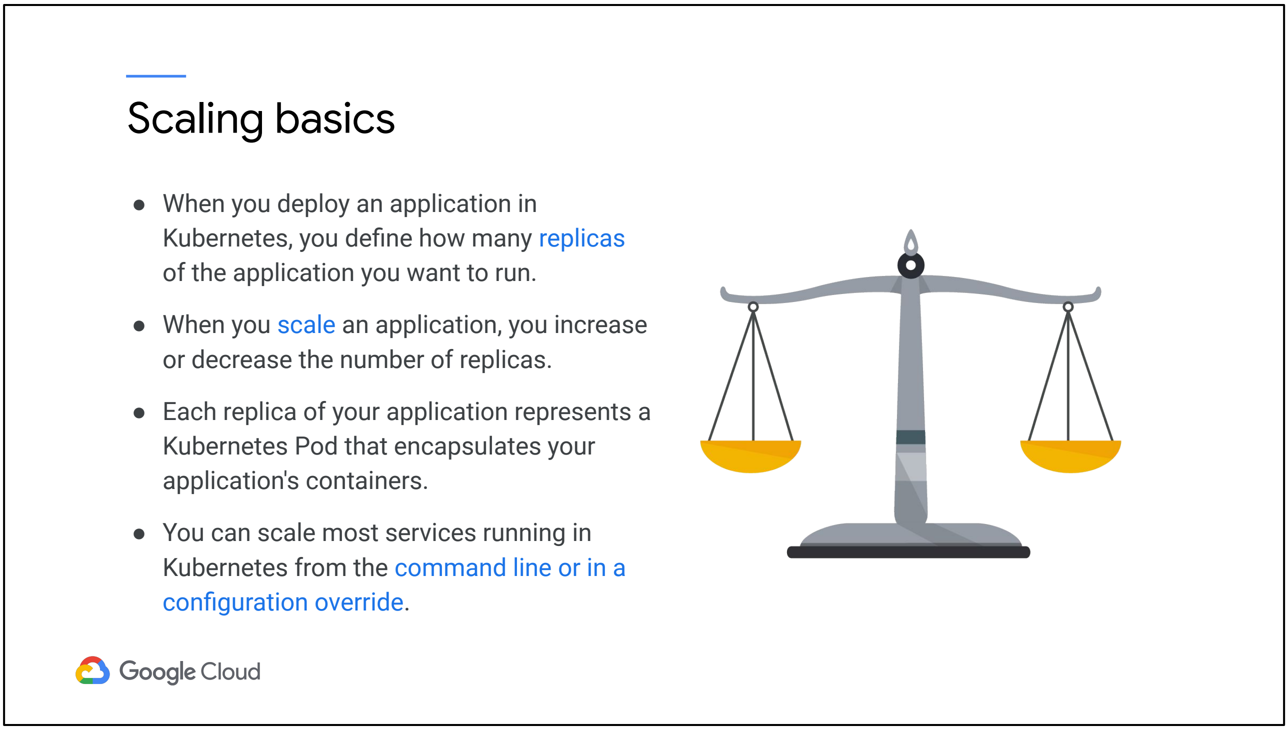
Google Cloud

You've seen this picture before in the context of capacity planning.

Scaling allows you to handle increases in traffic volume by spinning up additional processing capacity in your cluster.

It also conserves resources when they are no longer needed by removing the extra capacity.

Based on your business requirements, expected API traffic and backend capabilities, you can configure your runtime cluster so that it can scale efficiently when needed.



Scaling basics

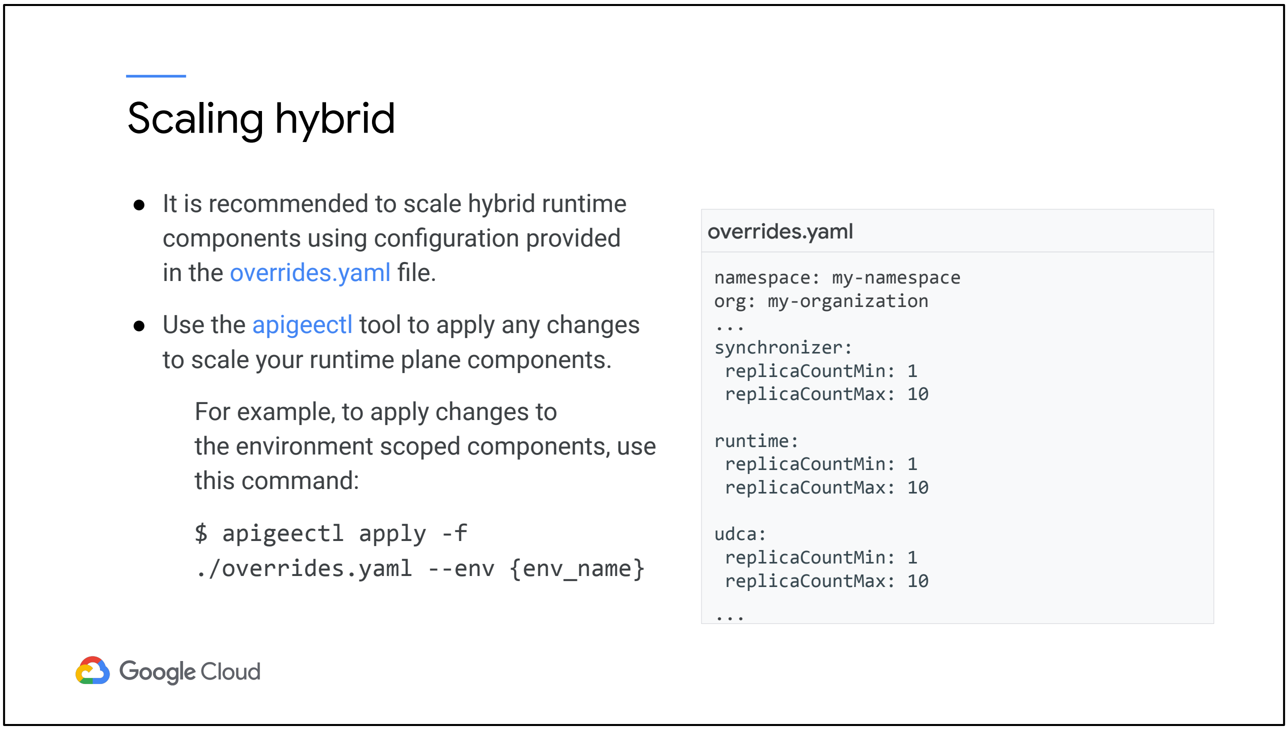
- When you deploy an application in Kubernetes, you define how many replicas of the application you want to run.
- When you scale an application, you increase or decrease the number of replicas.
- Each replica of your application represents a Kubernetes Pod that encapsulates your application's containers.
- You can scale most services running in Kubernetes from the command line or in a configuration override.

Google Cloud

You can scale most services running in Kubernetes from the command line or in a configuration override.

Scaling generally involves increasing or decreasing the number of replica pods that make up the service or application.

The exact method of scaling and autoscaling hybrid runtime services depends on the type of service.



Scaling hybrid

- It is recommended to scale hybrid runtime components using configuration provided in the overrides.yaml file.
- Use the apigeectl tool to apply any changes to scale your runtime plane components.

  For example, to apply changes to the environment scoped components, use this command:

  ```
  $ apigeectl apply -f
  ./overrides.yaml --env {env_name}
  ```

overrides.yaml

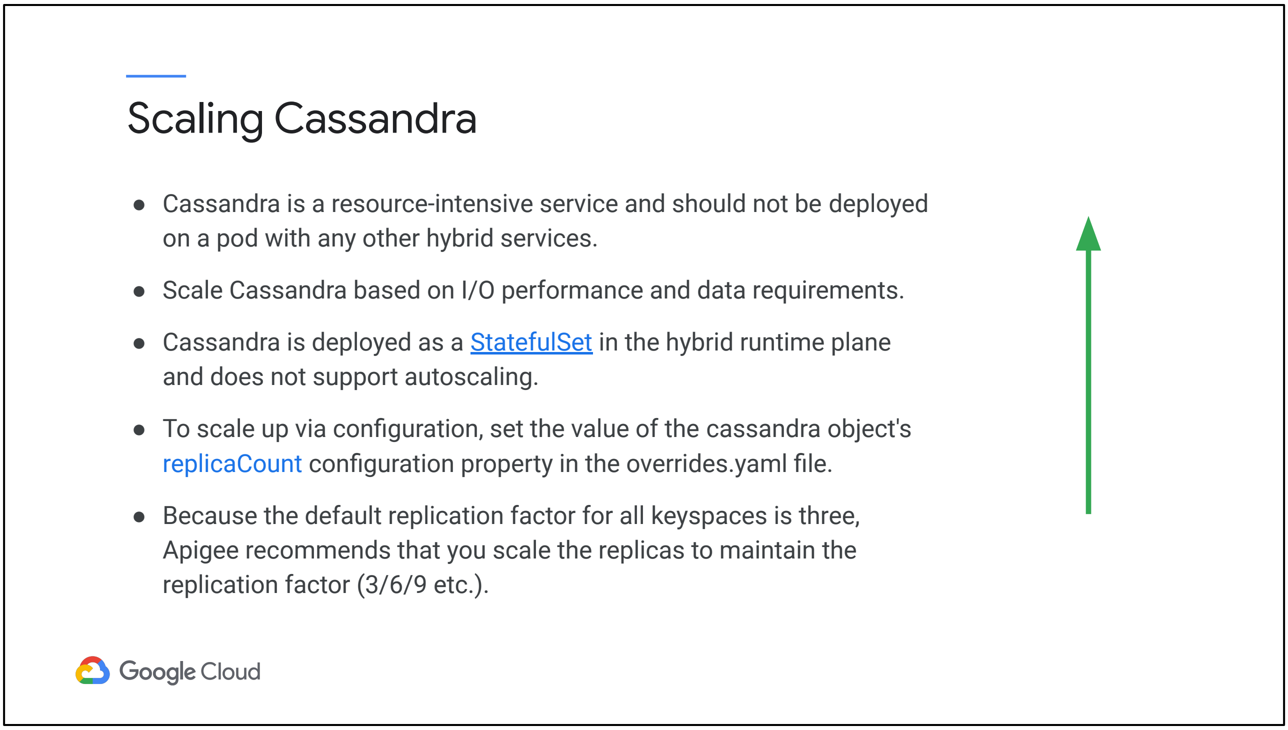
```
namespace: my-namespace
org: my-organization
...
synchronizer:
 replicaCountMin: 1
 replicaCountMax: 10

runtime:
 replicaCountMin: 1
 replicaCountMax: 10

udca:
 replicaCountMin: 1
 replicaCountMax: 10
...
```

Google Cloud

The recommended method of scaling hybrid runtime services is by updating the appropriate properties in the overrides.yaml configuration file.

This is usually done so any configuration changes are not overridden by future updates to the overrides file, or during hybrid software upgrades.

The changes are applied to the cluster by using the apigeectl apply command.



Scaling Cassandra

- Cassandra is a resource-intensive service and should not be deployed on a pod with any other hybrid services.
- Scale Cassandra based on I/O performance and data requirements.
- Cassandra is deployed as a StatefulSet in the hybrid runtime plane and does not support autoscaling.
- To scale up via configuration, set the value of the cassandra object's replicaCount configuration property in the overrides.yaml file.
- Because the default replication factor for all keyspaces is three, Apigee recommends that you scale the replicas to maintain the replication factor (3/6/9 etc.).

Google Cloud

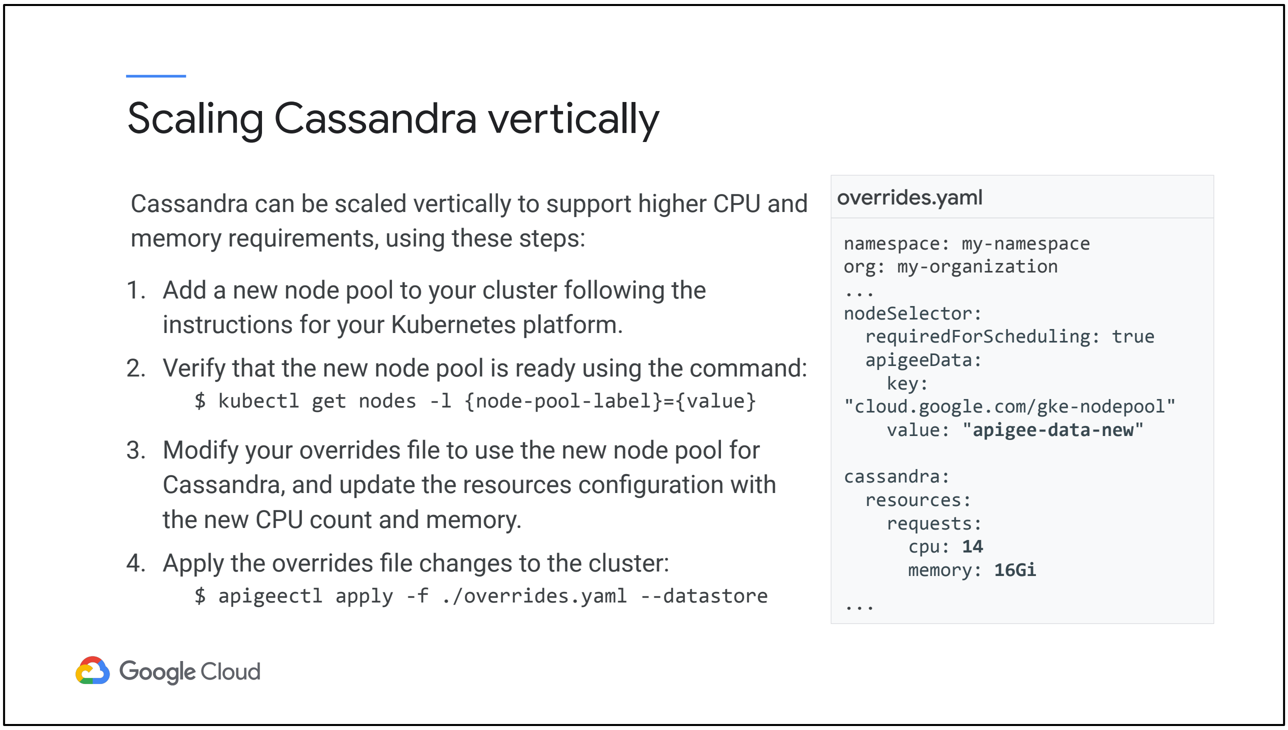
The Cassandra runtime data store is deployed as a StatefulSet in your Kubernetes cluster on its own nodepool.

StatefulSets do not support autoscaling but can be scaled manually.

To scale up the Cassandra StatefulSet in your hybrid runtime, set the value of the Cassandra object’s replicaCount property in the overrides.yaml configuration file.

Apigee recommends that you maintain the replicator factor of 3 when scaling the Cassandra component.

Though not recommended, you can scale up Cassandra using the command line with the kubectl scale command: $kubectl scale statefulset cassandra --replicas 6



Scaling Cassandra vertically

Cassandra can be scaled vertically to support higher CPU and memory requirements, using these steps:

1. Add a new node pool to your cluster following the instructions for your Kubernetes platform.
2. Verify that the new node pool is ready using the command:
   ```
   $ kubectl get nodes -l {node-pool-label}={value}
   ```
3. Modify your overrides file to use the new node pool for Cassandra, and update the resources configuration with the new CPU count and memory.
4. Apply the overrides file changes to the cluster:
   ```
   $ apigeectl apply -f ./overrides.yaml --datastore
   ```

overrides.yaml

```
namespace: my-namespace
org: my-organization
...
nodeSelector:
  requiredForScheduling: true
  apigeeData:
    key:
"cloud.google.com/gke-nodepool"
    value: "apigee-data-new"

cassandra:
  resources:
    requests:
      cpu: 14
      memory: 16Gi
...
```

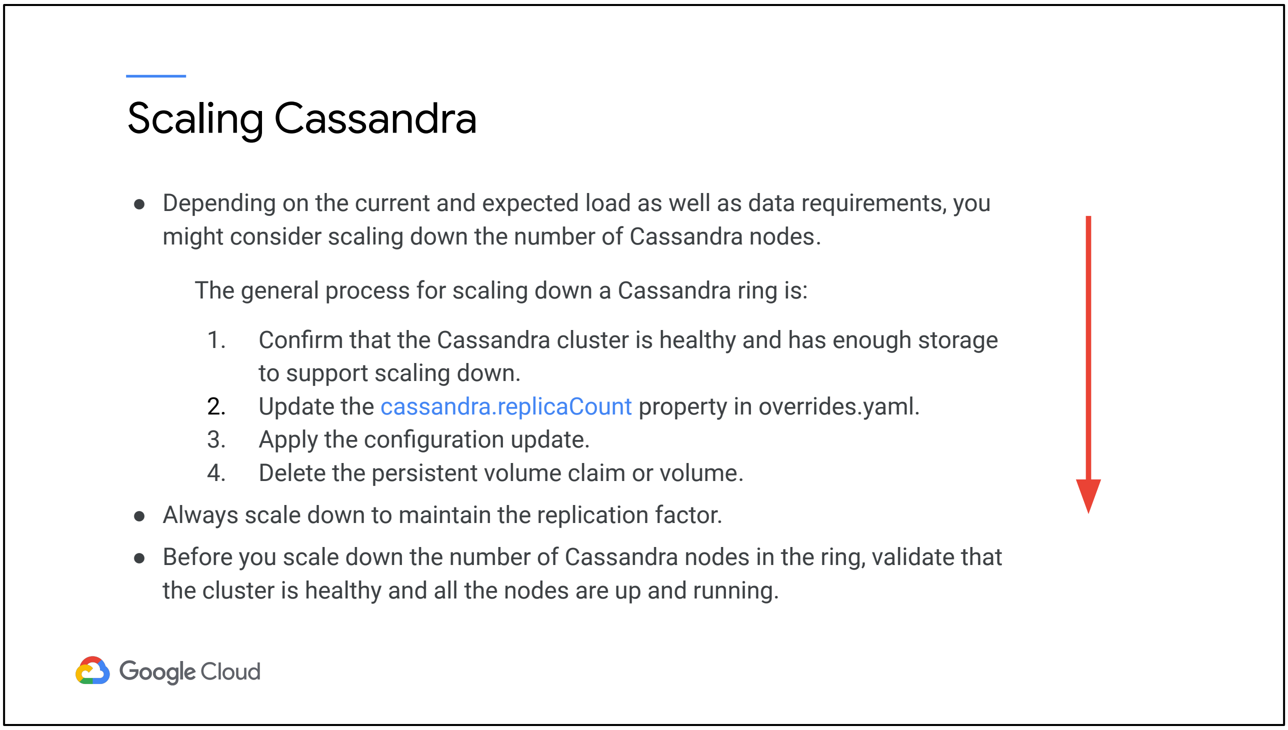
Google Cloud

The Cassandra runtime component can be scaled vertically to support higher CPU and memory requirements.

To implement this, you create a new node pool in your Kubernetes cluster with the higher sizing requirements.

Once the node pool is available, modify your hybrid runtime plane overrides configuration file to use the new node pool and update the Cassandra resource configuration with the new CPU and memory sizes.

Apply the changes to your cluster using the apigeectl apply command.



Scaling Cassandra

- Depending on the current and expected load as well as data requirements, you might consider scaling down the number of Cassandra nodes.

  The general process for scaling down a Cassandra ring is:

  1. Confirm that the Cassandra cluster is healthy and has enough storage to support scaling down.
  2. Update the cassandra.replicaCount property in overrides.yaml.
  3. Apply the configuration update.
  4. Delete the persistent volume claim or volume.
- Always scale down to maintain the replication factor.
- Before you scale down the number of Cassandra nodes in the ring, validate that the cluster is healthy and all the nodes are up and running.

Google Cloud

Depending on your current API traffic load, you may want to scale down the Cassandra ring in your runtime plane.

Scaling down your Cassandra pods is a multi-step process.

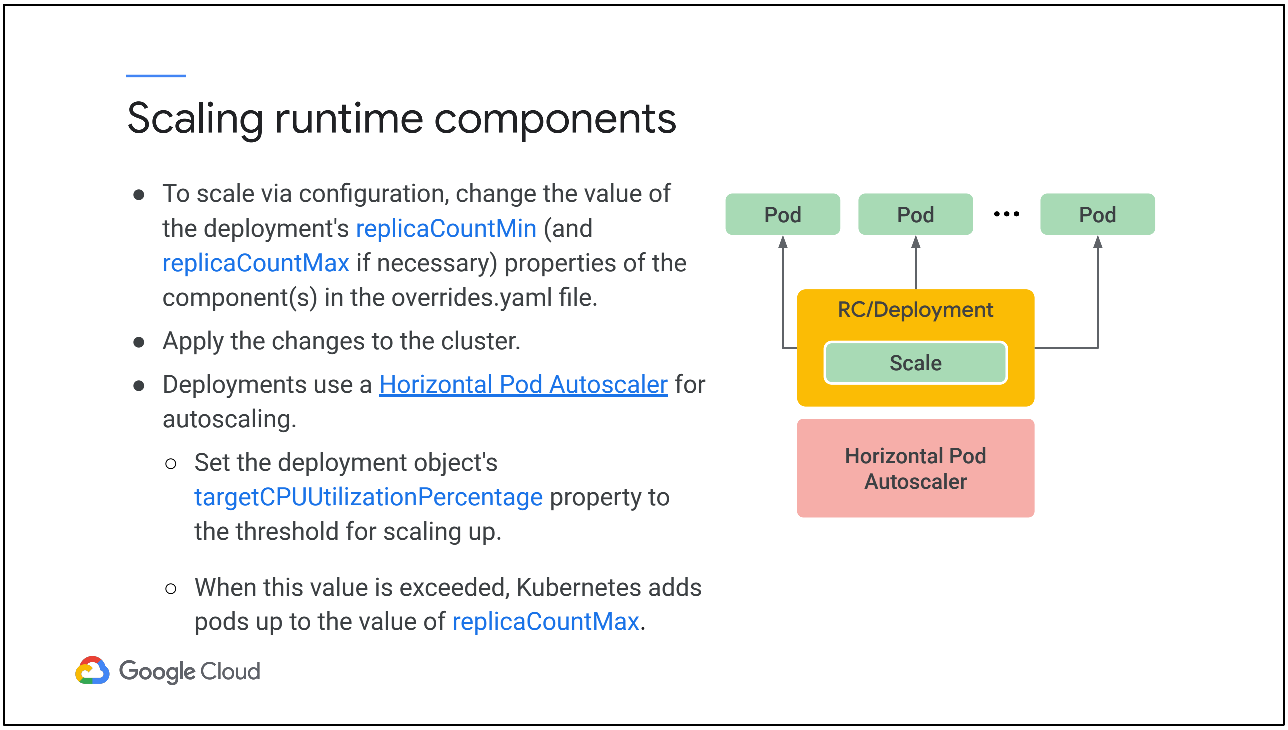
You must first confirm that the nodes in the Cassandra cluster are healthy, and there is enough storage to support your data requirements with the reduced number of Cassandra pods.

To scale down, update the Cassandra replicaCount property in your overrides.yaml configuration file. Make sure to maintain the replication factor of 3.

Use the apigeectl command to apply the change. The runtime cassandra pods are then scaled down in the cluster.

Finally, delete the persistent volume claims that are no longer used.

Scale down Cassandra | Apigee



Scaling runtime components

- To scale via configuration, change the value of the deployment's replicaCountMin (and replicaCountMax if necessary) properties of the component(s) in the overrides.yaml file.
- Apply the changes to the cluster.
- Deployments use a Horizontal Pod Autoscaler for autoscaling.
  - Set the deployment object's targetCPUUtilizationPercentage property to the threshold for scaling up.
  - When this value is exceeded, Kubernetes adds pods up to the value of replicaCountMax.

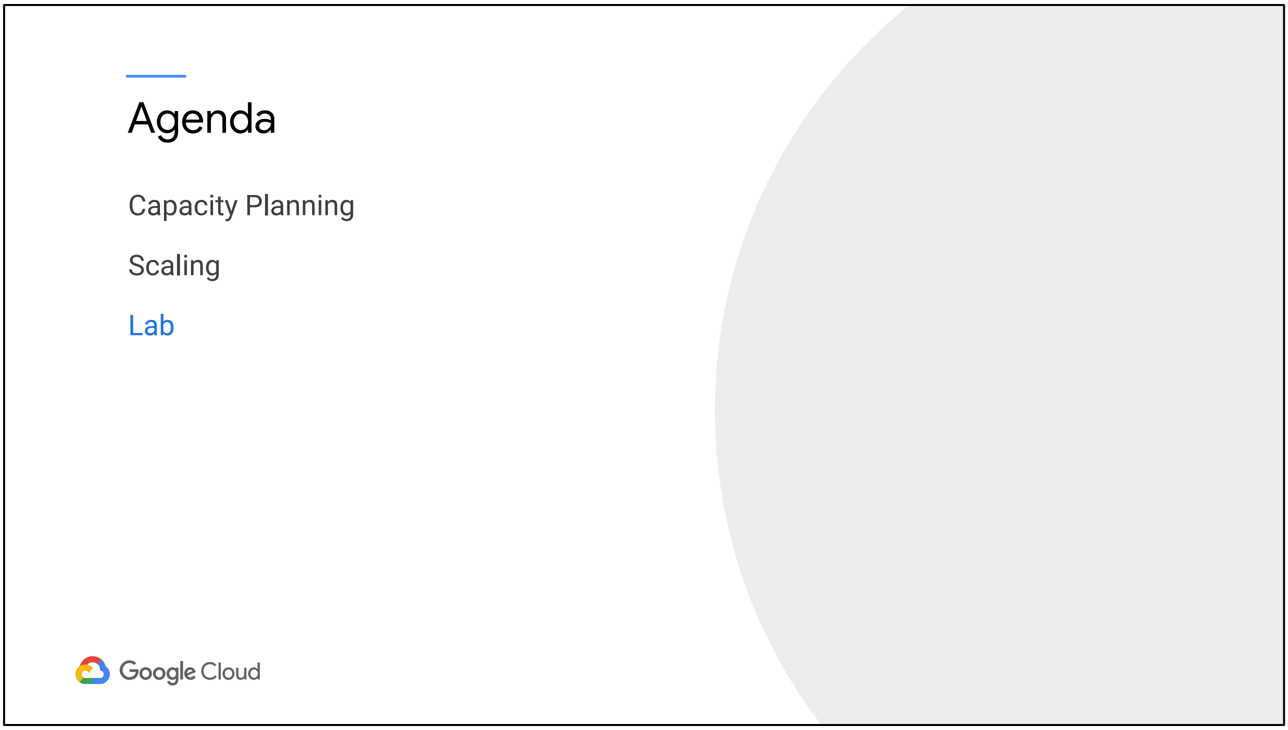
To manually scale the hybrid components, update the value of their corresponding Deployment’s replicaCountMin and replicaCountMax properties in the overrides.yaml configuration file.

Then run the apigeectl command to apply your changes to the cluster.

To autoscale these components, set the Deployment object's targetCPUUtilizationPercentage property to the threshold for scaling up. When this value is exceeded, Kubernetes adds the required number of pods up to the value of replicaCountMax.

Deployments use a Horizontal Pod Autoscaler for autoscaling.

[Scale and autoscale runtime services | Apigee]



Agenda

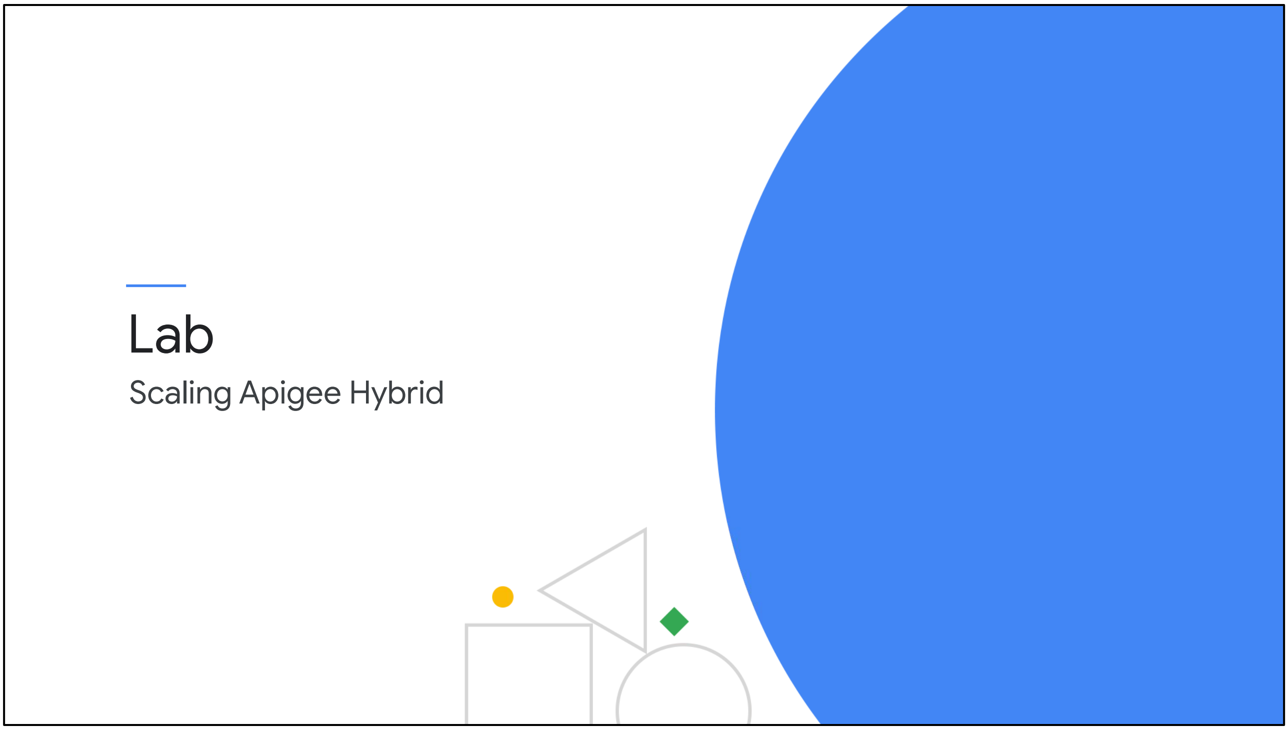
Capacity Planning

Scaling

Lab

Google Cloud

Lets now do a lab on scaling your Apigee hybrid runtime.

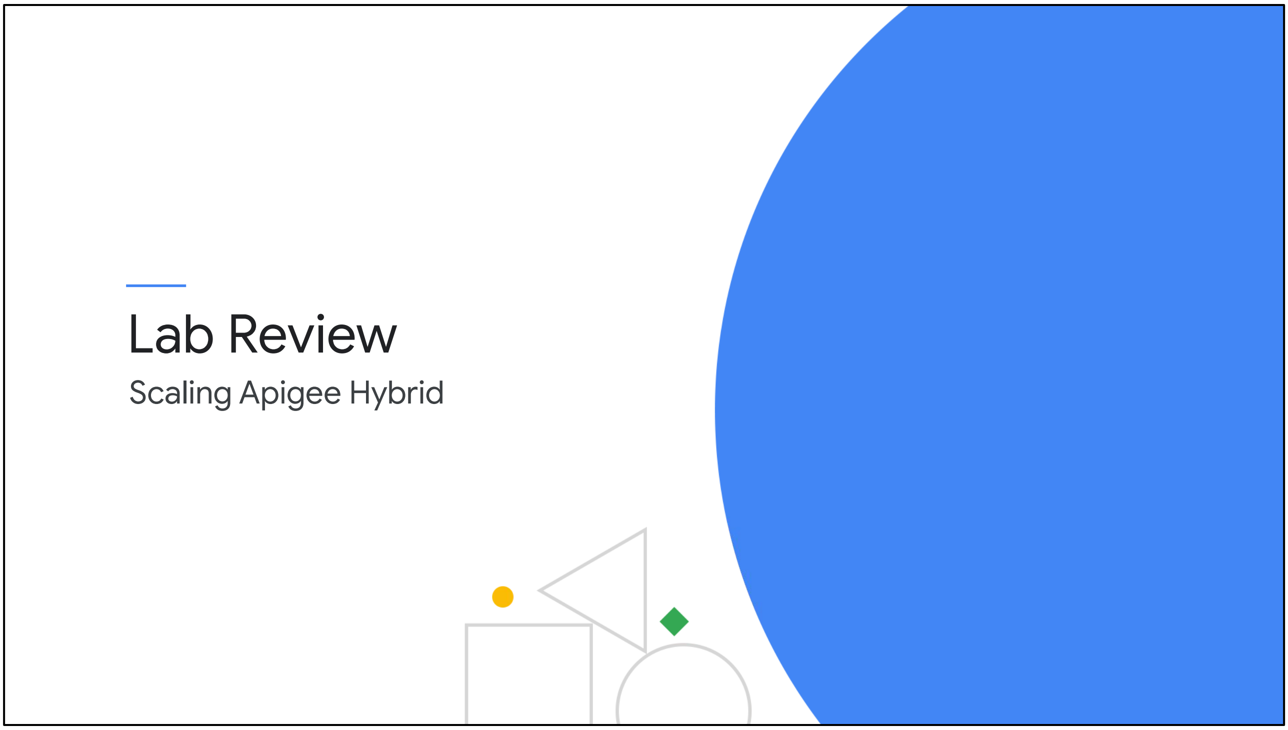


Lab

Scaling Apigee Hybrid

In this lab, you will learn how to manually scale the runtime message processor component in the cluster and use autoscaling to scale the runtime component based on CPU usage.

Specifically, you will configure your Apigee hybrid runtime plane to manually scale the runtime component and then verify that the number of runtime pods in your cluster increases to match the minimum configuration. You will then configure your Apigee hybrid runtime plane to autoscale the runtime component based on CPU usage.
You will send API request traffic to your test API proxy to simulate load and then verify that your runtime pods are automatically scaled.



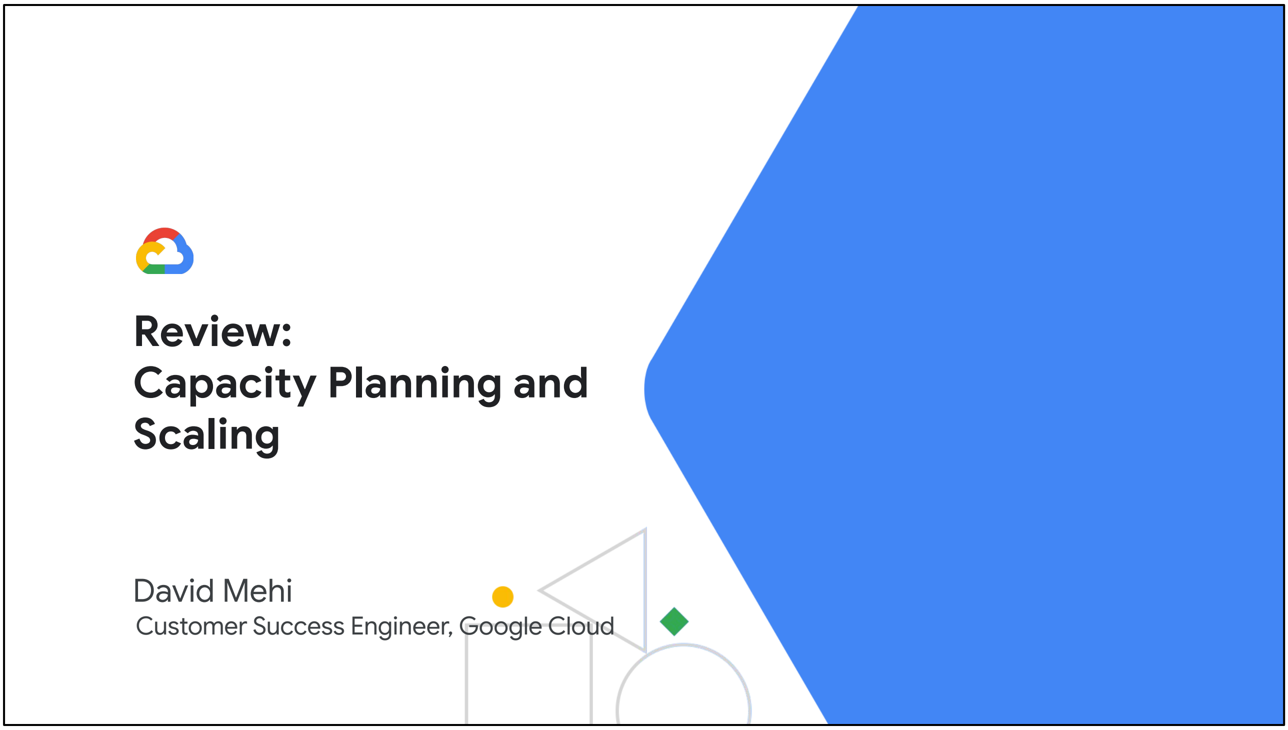
Lab Review

Scaling Apigee Hybrid

In this lab, you configured your Apigee hybrid runtime plane to manually scale the runtime component and then verified that the number of runtime pods in your cluster increased to match the minimum configuration.

You also configured hybrid to autoscale the runtime component based on CPU usage.

You sent API request traffic to your test API proxy to simulate load and then verified that the runtime pods scaled automatically.



Review: Capacity Planning and Scaling

David Mehi
Customer Success Engineer, Google Cloud

In this module, you learned about capacity planning for Apigee hybrid and items to consider when planning your runtime plane installation.

You also learned how to scale your runtime plane components using both manual and autoscaling.

You completed a lab to scale the runtime message processor component manually and with autoscaling.