



Capacity Planning and Scaling

Hansel Miranda
Technical Curriculum Developer, Google



Welcome to the module on capacity planning and expansion.

In this module, we discuss how to plan for capacity and manage changes to your infrastructure requirements for the Apigee private cloud platform.

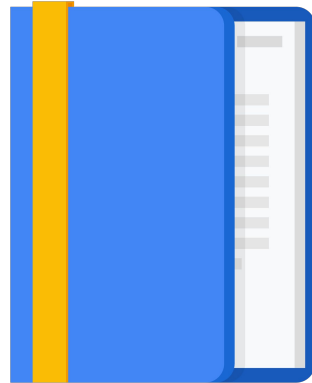
Agenda

Capacity planning

Provisioning

Expansion

Lab



We discuss capacity planning and how to provision and expand capacity on the platform.

You will complete a lab to add and remove API processing capacity for Apigee private cloud on Google Compute Engine.

Let's get started.

Know your business

Business

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

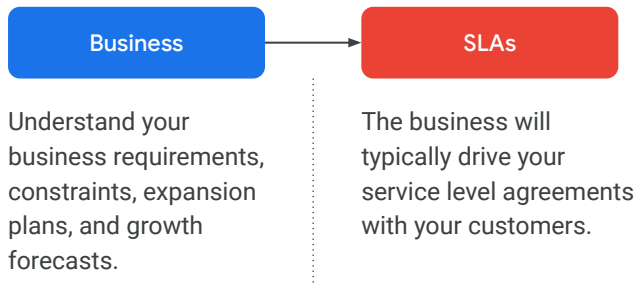


Understanding your business is probably the most important aspect of capacity planning.

You must be aligned with how your business intends to use the API platform and the business requirements, constraints, and growth forecasts.

You must be aware of marketing programs and any seasonality in the business that could impact platform requirements.

Know your business

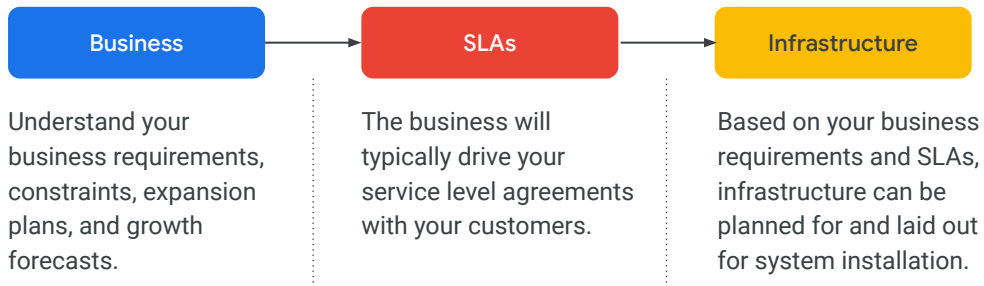


The business operations typically drive the platform SLA requirements.

Service level agreements play an important role in deciding the infrastructure requirements of the platform.

Platform resiliency, downtime, and regional failure are some of the factors to consider when planning your infrastructure requirements.

Know your business

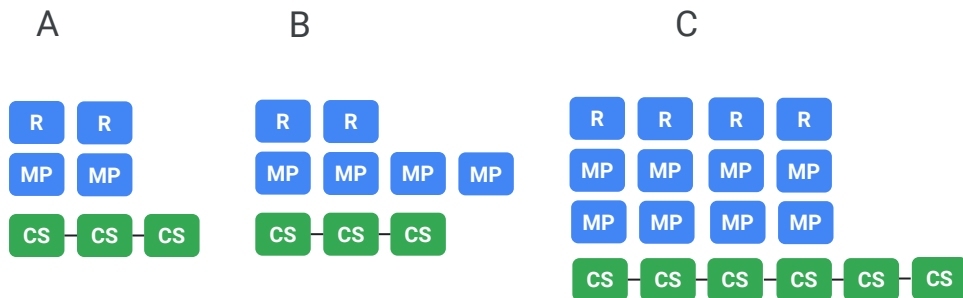


Business requirements and service level agreements drive your infrastructure requirements.

These requirements will help you determine whether you need to provision infrastructure in advance or scale elastically as demand grows.

All of these factors helps in capacity planning for your Apigee private cloud installation.

Understand traffic patterns



It is important to understand your API traffic patterns when planning your private cloud installation.

As the volume of traffic increases, you will eventually need to increase the number of servers in the underlying infrastructure.

Note that not all components scale at the same rate.

In the first example A, we have a typical topology that contains an equal number of routers and message processors and a standard Cassandra cluster. Routers can serve traffic to either message processor.

We may have an excess of capacity in the router layer, but we do have redundancy in all layers.

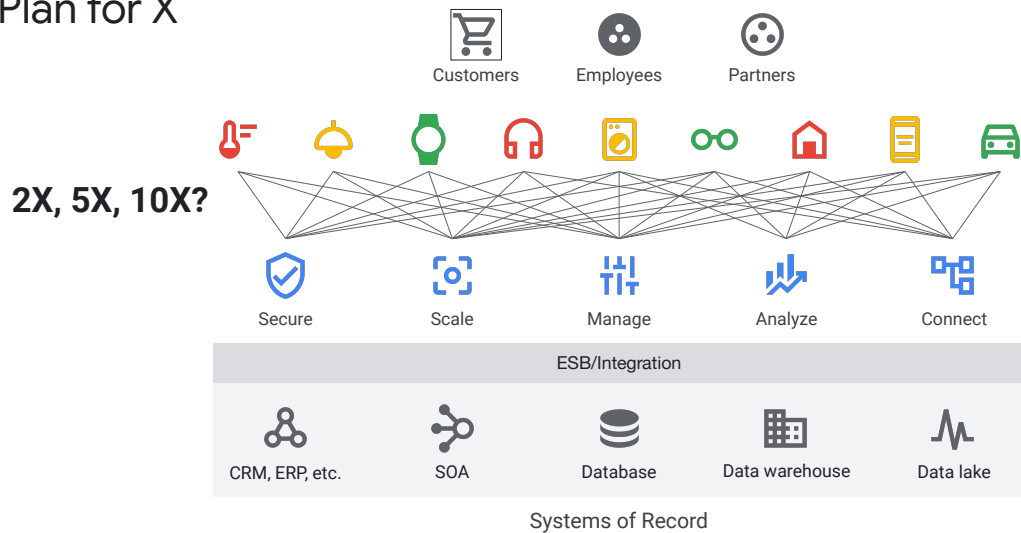
In the second example B, we've scaled out the message processor layer. A good benchmark is to use one router for every four message processors.

In practice, we would never use a single router in front of the message processors because this would give us a single point of failure.

In example C, we've doubled the number of routers and message processors to handle an increase in the number of API transactions.

With this configuration, you will probably need to expand the Cassandra ring. This is normally done either by doubling the size of the ring or increasing it by a multiple of the replication factor.

Plan for X



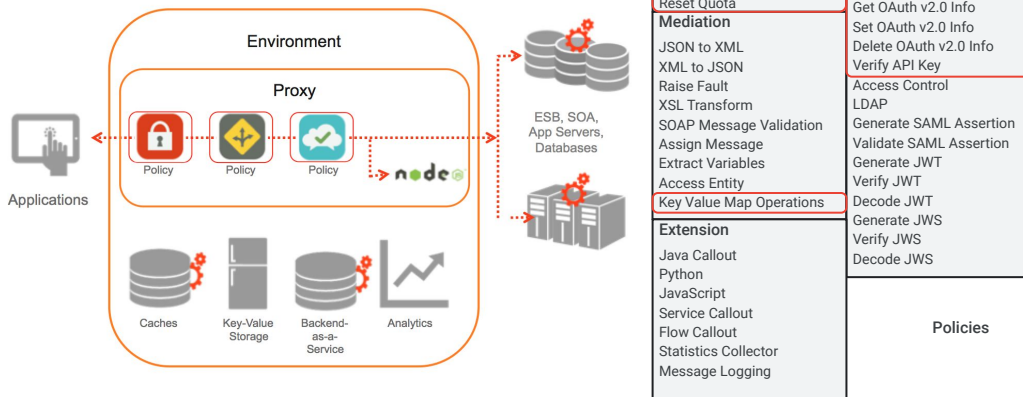
When you plan for a certain amount of platform capacity, you should use everything you know about your business to also plan ahead.

Where possible, think about adopting the cloud where you can take advantage of elastic scaling capabilities.

If you are using traditional infrastructure, an upfront investment in capacity is often far more cost-effective than reacting to accelerated change.

Understanding your business strategy, market growth, and your company's expectations and targets for the platform is crucial for its operational success.

API proxy considerations



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

The number of API proxies deployed to an environment and the complexity of the API processing logic may increase your response latency and reduce available capacity.

Also, API proxies that implement complex payload transformation or 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.

Execution time and I/O wait time have a direct impact on the ability of a router or a message processor to execute concurrent calls.

All these factors must be considered when going through a capacity planning exercise.

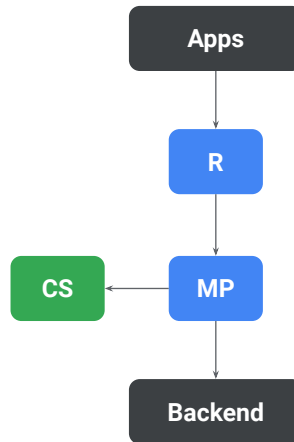
It's essential for operations teams to work closely with development and Q&A teams to understand the impact of the APIs on platform capacity and to determine optimization strategies (like caching) that can result in efficient use of platform resources.

In this way, you improve your ability to plan future releases and ensure that the

platform is always optimized effectively.

Know the critical path

Router (R), Message Processor (MP) and Cassandra (CS) (for some policies) stand on the critical path.



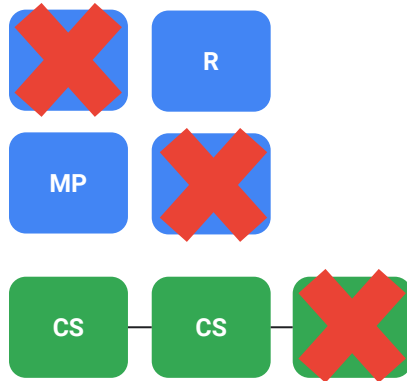
It is important to identify which Apigee private cloud components are on the critical path of execution of an API proxy.

The router and message processor receives requests from client apps 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 components fail.



Know that things fail all the time, so failure should be embraced.

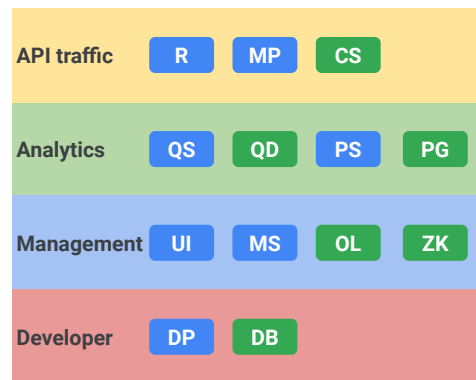
Component failure must always be considered as a factor when planning your runtime capacity.

The Apigee private cloud platform has been designed so that, with careful planning, you can ensure that failures do not impact the API runtime.

When planning for failure, consider also what happens to capacity when components fail.

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

Independent scalability



The Apigee private cloud components that handle API traffic, analytics, management, and developer portal functionality can and should scale independently.

Because of this, the platform enables you to focus your infrastructure investment and resources to achieve operational excellence and generate the most effective outcome.

Hardware characteristics and requirements

Server/VM	Components	CPU	RAM	Hard Disk
Router, Message Processor	R, MP	8 Core	16 GB	100 GB
Cassandra	CS, ZK	8 Core	16 GB	250 GB local storage with SSD or fast HDD supporting 2000 IOPS
Analytics - Qpid	QD, QIS	4 Core	8 GB	30 GB to 50 GB local storage with SSD The default Qpid queue size is 20 GB. To add capacity, use additional Qpid nodes.
Management	UI, MS, OL	4 Core	8 GB	60 GB
Analytics - Postgres	PG, PS	8 Core	16 GB	500 GB to 1TB network storage, preferably with SSD backend, supporting 1000 IOPS or higher.
Developer portal	DP	2 Core	1 GB	10 GB

For the latest hardware requirements, see

<https://docs.apigee.com/private-cloud/v4.50.00/installation-requirements#hardwarerequirements>

Here are the the minimum hardware requirements for the Apigee private cloud components.

Note that the hard disk requirements are in addition to the disk space required by the operating system.

Depending on your API logic and traffic volume, your installation may require more or fewer resources than those that are listed.

Cassandra and analytics components such as Qpid and PostgreSQL have fairly high performance requirements.

These components make heavy use of disk and thus benefit from high performing disk subsystems.

The random read and write nature of Cassandra makes SSDs an ideal choice.

Additionally, Cassandra's compaction process can substantially increase the amount of disk space consumed while the process is in progress.

PostgreSQL capacity planning

Adjust Postgres system requirements based on throughput:

PostgreSQL Machine Requirements

Throughput	CPU	RAM	Storage IOPS
< 250 TPS	4 Core	8 GB	> 1000
> 250 TPS	8 Core	16 GB	> 1000
> 1000 TPS	8 Core	16 GB	> 2000
> 2000 TPS	16 Core	32 GB	> 2000
> 4000 TPS	32 Core	64 GB	> 4000

Calculation:

Amount of storage needed (in bytes)
=
(number of bytes per analytics message) x
(number of requests per day) x
(number of days of data retention)

Example:

(2048 bytes per analytics message) x
(6480000 request per day) x
(90 days of data retention) =
119439360000 bytes, which is
equivalent to
[1194.3936 GB](#).

Analytics is a core component of Apigee Edge for private cloud. The capacity of the PostgreSQL servers that are used to store analytics data requires careful planning.

Here are the PostgreSQL system requirements.

As the number of API transactions flowing through the platform increases, the CPU memory and IOPS requirements increase proportionally.

The PostgreSQL machines process analytics events data from API requests, perform aggregation of raw data, and also generate reports.

When planning actual storage requirements, consider the amount of data that Apigee Edge generates for each analytics event.

Each analytics event generates approximately 2 KB of data in addition to any custom values that may be stored.

The calculation shown can be used to determine how much storage is required based on the size of the analytics message and your data retention requirements.

For every API request, the message processor generates one analytics message.

Amount of storage needed (in bytes) = (the number of bytes per analytics message) multiplied by (the number of requests per day) multiplied by (the number of days of

data retention)

It is recommended to use network storage for the PostgreSQL database because it provides the ability to dynamically scale storage size when required.

Network IOPS can also be adjusted in most storage network subsystems, and you can enable storage- level snapshots as part of a backup and recovery solution.

Cassandra capacity planning

- Calculating usable disk space per node/server:

```
raw_capacity = disk_size x number_of_data_disks
formatted_disk_space = raw_capacity x 0.9
usable_disk_space = formatted_disk_space x (0.5 to 0.8)
```

- Calculating usable disk capacity on the ring:

Replication factor / number of Cassandra nodes = % of data stored on each Cassandra node

Example:

$3(RP)/3(CS) = 100\%$ data per Cassandra node

$3(RP)/6(CS) = 50\%$ data per Cassandra node

RP = Replication factor

When calculating usable disk space for Cassandra, start your estimation based on factors like API traffic and the usage of Edge policies such as OAuth, Cache, KVM and other policies that store data in the Cassandra database.

You must also consider the operational processes that are executed automatically by Cassandra in the background.

The calculation shows how to estimate usable space per Cassandra node.

Usable space is equal to the final formatted disk space multiplied by a number between 0.5 and 0.8.

The multiplier reduces the available disk space and allows space needed for compaction, hint file handling, snapshots, and other Cassandra processes.

You also need to calculate the capacity for the overall ring taking into account the replication factor.

The replication factor for Cassandra in Apigee Edge is 3. If we have 500 gigabytes of usable disk per node and 3 nodes in the ring, our usable space remains at 500 gigabytes and not 1500 gigabytes.

This is because a data element is copied to each node of the ring. As the number of nodes is increased beyond the replication factor, additional free space may be gained.

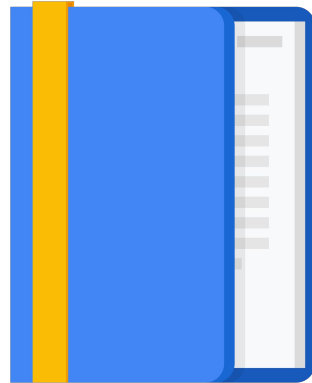
Agenda

Capacity planning

Provisioning

Expansion

Lab

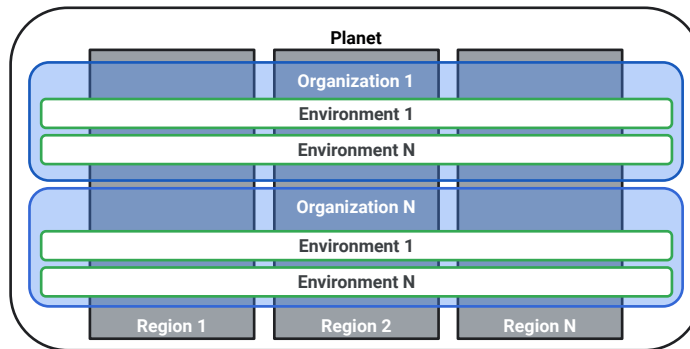


In this lesson we discuss how to expand the logical capacity of the Apigee private cloud platform.

You will learn how to add and delete organizations, environments, virtual hosts, and TargetServers in Apigee Edge.

Multitenancy

- A planet can expand to multiple regions (data centers). An organization and environment expands across the planet.
- A planet can contain multiple organizations and represent tenants on the platform. An organization can have multiple environments.



Apigee Edge for private cloud uses the concept of a planet, which corresponds to a single installation of Apigee private cloud.

A common approach is to install separate planets for production and non-production environments.

Planets can span multiple data centers, and organizations can span the entire planet.

Planets can also contain multiple organizations. An organization in Apigee Edge represents a tenant on the platform.

Each organization can contain one or more environments. An environment provides a runtime execution context for API proxies.

Organizations and environments represent strong conceptual boundaries that allow for strong data and process partitioning.

Organization provisioning

Naming convention

An organization name:

- Must be unique across the planet.
- Can only contain lowercase letters, numbers, and the hyphen character.



The organization name must be unique within the planet, so you cannot create two organizations with the same name.

Organization names should only contain lowercase characters, numbers, and certain special characters.

Organization provisioning

Onboarding process

During the onboarding process, the apigee-service utility:

- Creates the organization.
- Optionally assigns a new user to function as the organization administrator.
- Associates the organization with a gateway pod.
- Creates an environment.
- Creates a virtual host for the environment.
- Associates the environment with all Message Processors.
- Enables analytics.

Usually, organizations are created as part of the onboarding process following a successful installation.

This process is used to provision all the elements that make up a fully functional organization.

Onboarding is performed using the setup-org function of the apigee service utility and must be executed on the management server.

The utility creates the organization and can optionally assign a new user to function as the organization administrator. The user must already exist in the system.

The utility also creates an environment and a virtual host for that environment.

It also associates the environment with all the message processors and enables analytics on the platform.

Organization provisioning

Onboarding process

```
/opt/apigee/apigee-service/bin/apigee-service apigee-provision setup-org  
-f response_file
```

Sample configuration file:

<pre>IP1=<node 1> MSIP="\$IP1" ADMIN_EMAIL="<admin@example.com>" APIGEE_ADMINPW="<password>" NEW_USER="y" USER_NAME="<orgadmin@example.com>" FIRST_NAME="OrgAdminName" LAST_NAME="OrgAdminLastName" USER_PWD="<password>"</pre>	<pre>ORG_NAME="<org name>" ORG_ADMIN="\$USER_NAME" ENV_NAME="prod" VHOST_PORT="9001" VHOST_NAME="default" VHOST_ALIAS="<virtual-host-name>"</pre>
--	---

To execute the onboarding process, run apigee service with the apigee-provision setup-org command options, and provide the configuration file that configures the organization and environment.

In the example, the MSIP property is the IP address of the Management Server.

The ADMIN_EMAIL and APIGEE_ADMINPW properties are the username and password for the Apigee sysadmin user that was created during the installation.

You can and should create a separate user to serve as the organization administrator and provide the required user properties.

The organization and environment names are also provided in the configuration file, along with the details of the first environment.

These include the virtual host name and port and one or more aliases that the virtual host should listen on for API requests to proxies deployed to the environment.

Organization provisioning

Onboarding verification

- List users:
`curl -u <adminEmail> http://<ms_IP>:8080/v1/users`
- List organizations:
`curl -u <adminEmail> http://<ms_IP>:8080/v1/organizations`
- Describe an organization:
`curl -u <adminEmail>:<adminPassword> http://<ms_IP>:8080/v1/organizations/<orgname>`
- Get analytics provisioning status:
`curl -u <adminEmail>:<adminPassword> \`
`http://<ms_IP>:8080/v1/o/<orgname>/e/<envname>/provisioning/axstatus`
- Describe analytics tables:
`psql -h /opt/apigee/var/run/apigee-postgresql -U apigee apigee`
`apigee=# : \d analytics."<orgname>.<envname>.fact"`
`apigee=# \q`

After the onboarding process is complete, you should verify the details of the objects that were created using management APIs.

The `/users` API should return a list of users that include the original sysadmin user and the newly created organization admin user.

The `/organizations` API should return a list of organizations that exist. Typically, the list will only contain the single organization created so far.

By invoking the same API and appending the organization name to the URL, you can retrieve details about the organization, including any environments that belong to the organization.

The `/axstatus` API returns the status of the analytics components along with their servers UUIDs.

Finally, you can use the PSQL utility to confirm that the facts tables for the new organization and environment have been created in the PostgreSQL database.

Creating an organization

apigee-provision

To create an organization separately, use the create-org function:

```
/opt/apigee/apigee-service/bin/  
apigee-service apigee-provision  
create-org -f response_file
```

Sample configuration file:

```
APIGEE_ADMINPW="<password>"  
ORG_NAME="<orgname>"  
ORG_ADMIN="<user_name>"
```

<http://docs.apigee.com/private-cloud/latest/creating-organization-environment-and-virtual-host>

It is possible to create an organization separately from the private cloud installation and provisioning process.

To do this, run the apigee service utility using the create-org function with the apigee-provision command.

Note that the configuration file is much shorter.

You only supply the sysadmin password, the new organization name, and the organization admin username. The organization admin should already exist.

Alternatively, you can use management API calls to create organizations. More details are documented at the link provided.

Deleting an organization

Overview



On rare occasions, you may need to delete an organization.

Before attempting this, you should fully understand what an organization represents in its entirety.

An organization represents a tenant on Apigee Edge and represents a strong logical data and process partitioning boundary.

It is used by the Edge security model to control access to resources associated with the tenant it represents.

Because of this, deleting an organization will not perform a cascading delete of all relationships and objects owned by that organization.

That is why, before deleting the organization, you must disassociate or delete all related items and objects in that organization.

In the next few slides we discuss the steps needed to delete an organization that has a single environment.

If the organization has more than one environment, you need to repeat these steps for each environment.

Deleting an organization

Deleting keystores

- API to get the keystores defined for an environment:
`curl -v -u <adminEmail> http://<ms_IP>:8080/v1/o/<org-name>/e/<env-name>/keystores`
- API to delete the keystores defined in an environment:
`curl -v -u <adminEmail> -X DELETE \`
`http://<ms_IP>:8080/v1/o/<org-name>/e/<env-name>/keystores/<keystore-name>`

First you must delete any keystores that are defined in the environment.

Get a list of keystores in the environment using the /keystores management API call and provide the organization and environment name in the URL.

You then delete each keystore using the same API.

Deleting an organization

Delete virtual hosts

- API call to get the virtual hosts defined for an Environment:

```
curl -v -u <adminEmail>  
http://<ms_IP>:8080/v1/o/<org-name>/e/<env-name>/virtualhosts
```

- API call to delete the virtual hosts:

```
curl -v -u <adminEmail> -X DELETE \  
http://<ms_IP>:8080/v1/o/<org-name>/e/<env-name>/virtualhosts/<vhost>
```

Next, you delete the virtual hosts that are defined in the environment.

Get a list of virtual hosts in the environment using the /virtualhosts management API call.

You then delete each virtual host using the same API.

Deleting an organization

Disassociate servers

- API call to get the UUIDs of the servers:

```
curl -v -u <adminEmail> http://<ms_IP>:8080/v1/o/<org-name>/e/<env-name>/servers
```

- For each of the servers, execute:

```
curl -v -u <adminEmail> -X POST \  
http://<ms_IP>:8080/v1/o/<org-name>/e/<env-name>/servers \  
-d "action=remove&uuid=<uuid>&region=<region-name>&pod=<pod-name>" \  
-H "Content-Type: application/x-www-form-urlencoded"
```

Next, you disassociate the servers from the environment.

Get a list of servers in the environment using the /servers management API call.

You then disassociate each server from the environment using the same API and specify the UUID of the server, the region and pod names, and the remove action.

Deleting an organization

Delete the environment

- Use the following management API call to delete the environment:

```
curl -v -u <adminEmail> -X DELETE \  
http://<ms_IP>:8080/v1/o/<org-name>/e/<env-name>
```

After you have deleted all the objects within the environment and disassociated all servers, you can delete the environment itself.

Use the /environments management API to delete the environment, and provide the organization and environment name.

Deleting an organization

Disassociate the organization from pods

- Get pods associated to org:
`curl -v -u <adminEmail> http://<ms_IP>:8080/v1/o/<org-name>/pods`
- Disassociate organization from pods:
`curl -v -u <adminEmail> -X DELETE http://<ms_IP>:8080/v1/o/<org-name>/pods \`
`-d "action=remove®ion=<region-name>&pod=<pod-name>" \`
`-H "Content-Type: application/x-www-form-urlencoded"`

If there are multiple environments within the organization, repeat the previous set of steps to delete the objects in each environment and the environment itself.

After all the environments are deleted, you must then disassociate the organization from all the pods in the planet.

To do this, first get a list of pods associated to the organization using the /pods management API call.

You then disassociate the organization from each pod using the same API and specify the region and pod name and the remove action.

Deleting an organization

Delete the organization

- Use the following management API call to delete the organization:

```
curl -v -u <adminEmail> -X DELETE http://<ms_IP>:8080/v1/o/<org-name>
```

Finally, with all the objects, servers and associations removed, you delete the organization.

To delete the organization, use the /organizations management API and provide the organization name.

Environment provisioning

Environment naming convention

An environment name:

- Must be unique within an organization.
- Can only contain lowercase letters, numbers, and the hyphen character.

An environment name must be unique within the organization.

Environment names can contain lowercase and uppercase characters, numbers, and certain special characters.

Environment provisioning

apigee-provision

To create an environment:

```
/opt/apigee/apigee-service/bin/ap  
igee-service apigee-provision  
add-env -f response_file
```

Sample configuration file:

```
APIGEE_ADMINPW="<password>"  
ORG_NAME="<orgname>"  
ENV_NAME="<envname>"  
VHOST_PORT="9001"  
VHOST_NAME="<virtualhostname>"  
VHOST_ALIAS="<virtualhostalias>"
```

<http://docs.apigee.com/private-cloud/latest/creating-organization-environment-and-virtual-host>

When creating a new environment, the easiest way is to use the apigee-service apigee provision command with the add-env function and provide a configuration file.

In the configuration file, provide the Apigee sysadmin password, the name of the organization that the environment should be associated with, the name for the new environment, the virtual host port, and the name of the virtual host to be created.

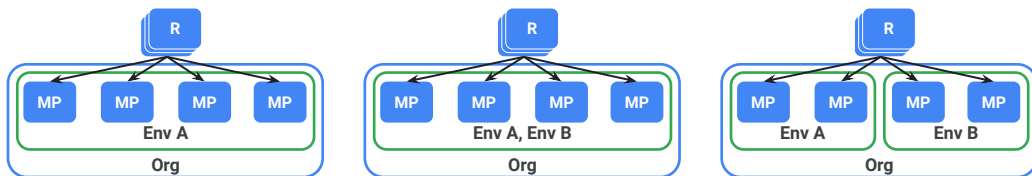
Note that the combination of virtual host name and port should be unique.

A virtual host alias that will receive API traffic to this environment is also provided.

You can also create an environment using the management API. Details are documented at the link provided.

Environment scope

- Environments define logical processing scope. Message processors are associated to environments.
- Routers are multi-tenant aware. They are capable of sending traffic to any message processor, given the associated scope.



An environment provides a logical execution context or scope for API proxies and configuration objects.

Message processors can be associated with one or more environments.

Routers are multi-tenant aware and are capable of sending traffic to any message processor given the scope of organization and environment.

Consider three typical scenarios.

In this first scenario, we have an organization that contains a single environment, A.

In the second scenario, we have an organization that contains two environments, A and B. The message processors have been associated with both environments.

All routers send traffic to all message processors.

In the third scenario, we have an organization that contains two environments, A and B. Two of the message processors have been associated with Env A, and the other two with Env B.

In this setup, the message processors for environment A will never process traffic destined for environment B, and vice versa.

This ability to associate message processors with different environments enables partitioning of resources.

For example, you may have a staging environment and a production environment within the same organization.

To ensure that one doesn't impact the other, separate pools of message processors can be associated with each environment.

Environment provisioning

Associate a message processor to an environment

- Get the UUID of the message processor:

```
curl -v http://<mp_IP>:8082/v1/servers/self/uuid
```

- For each message processor to be associated with an environment:

```
curl -v -u <sysAdminEmail> -X POST \  
http://<ms_IP>:8080/v1/organizations/<org-name>/environments/<env-name>/servers \  
-d "action=add&uuid=<MP_UUID>" \  
-H "Content-Type: application/x-www-form-urlencoded"
```

To associate a message processor with an environment, you need the UUID of the message processor. The UUID is a unique ID assigned to each Apigee private cloud component in the cluster.

To get the UUID, make an API call to the message processor's /servers API, which listens on port 8082.

After you have the UUID, make a second API call, this time to the management API specifying the organization and environment that the message processor is to be associated with.

The add action and UUID must be supplied as query parameters in the API call.

Environment provisioning

Delete an environment

To delete an environment, follow these steps:

1. List and delete all keystores associated with the environment.
2. List and delete all virtual hosts associated with the environment.
3. List and disassociate all message processors associated with the environment.
4. Delete the environment.

Use the same management API calls as those used when deleting an environment as part of the process to delete the organization.

Before deleting an environment, you must remove any objects that are associated with it.

You first delete any keystores and virtual hosts that are defined for the environment.

To delete these objects, you invoke the corresponding management API to get a list of the objects of that type, then invoke the delete API on that same object type, providing the name of each object to be deleted.

If you're decommissioning or moving resources, you may need to disassociate a message processor from an environment.

To do this, first obtain a list of all server UUIDs for your organization and environment name using the `/servers` management API.

You then disassociate each server from the environment using the same `/servers` API and specify the UUID of the server, the region and pod names, and the `remove` action.

Note that to delete an environment you have to disassociate all message processors that are associated with the environment.

Finally, when there are no objects or servers associated with the environment, you can delete the environment itself by invoking the `/environments` management API and

providing the environment name.

Note that the management API calls to delete an environment and its associated objects and servers are the same APIs that were discussed earlier in this section as part of deleting an organization.

Virtual host provisioning

Virtual host definition

- A virtual host definition requires three values:
 - Name (allowable characters are: a-zA-Z0-9._\-\$ %)
 - Virtual host alias
 - Port

Environment Configuration test -			
Caches Key Value Maps Target Servers Virtual Hosts			
Name	Port	Alias	Actions
default		80 apigee-docs-test.apigee.net	
Name secure	Port 443	Alias apigee-docs-test.apigee.net	
SSL Info			
Enabled	<input checked="" type="checkbox"/>	Client Auth Disabled	Hide
Key Store	freetrial	Trust Store	
Key Alias	freetrial	Ciphers	

A virtual host is a configuration object in Apigee Edge that is used to serve traffic for your environment.

To define a virtual host, you need to specify three values. The first is a name that identifies the virtual host and is unique within the environment.

For the name, you can use upper or lower case characters, numbers, and a few special characters.

In the example shown, there are two virtual hosts. One is named *default* and serves regular HTTP traffic; the other is named *secure* and serves HTTPS traffic.

The second value needed to define a virtual host is the virtual host alias. A virtual host can have multiple virtual host aliases.

Most commonly, a fully qualified domain name that is used to access API services is used as a virtual host alias. You can also specify a combination of the router or load-balancer IP address and the listening port as a virtual host alias.

The third value needed to define a virtual host is the port number. The port number specified here is exposed on the routers for that virtual host.

Virtual hosts must be able to be uniquely identified.

This means that you can create virtual hosts for the same alias as long as you use different ports, or virtual hosts with different aliases on the same ports, or virtual hosts with a combination of different aliases and ports.

Virtual host provisioning

Create a virtual host

The following management API call creates a virtual host:

```
curl -H "Content-Type:application/xml" -u <adminEmail> -X POST \
http://<ms_IP>:8080/v1/o/<org-name>/e/<env-name>/virtualhosts \
-d @<payload_xml_file>
```

Sample payload:

```
<VirtualHost name="default">
  <HostAliases>
    <HostAlias>myorg-test.apigee.net</HostAlias>
  </HostAliases>
  <Interfaces/>
  <Port>9002</Port>
</VirtualHost>
```

<https://docs.apigee.com/private-cloud/latest/creating-organization-environment-and-virtual-host>

<http://docs.apigee.com/api-services/content/virtual-hosts>

Virtual hosts are normally created during the environment provisioning process.

A virtual host can also be created by using the /virtualhosts management API and providing the organization and environment name for which it is being created.

The API also requires a payload that contains the virtual host name, the host alias, and port.

If you need to create the virtual host with multiple aliases, simply include additional host alias elements in the payload.

When you are creating a virtual host that serves HTTPS traffic, additional TLS elements are required in the payload.

Refer to the documentation for more details on provisioning virtual hosts at the link provided.

Virtual host provisioning

Delete virtual hosts

- API call to get the virtual hosts defined in the Environment:

```
curl -v -u <adminEmail>  
http://<ms_IP>:8080/v1/o/<org-name>/e/<env-name>/virtualhosts
```
- API call to delete the virtual hosts:

```
curl -v -u <adminEmail> -X DELETE \  
http://<ms_IP>:8080/v1/o/<org-name>/e/<env-name>/virtualhosts/<vhost>
```

Deleting a virtual host that is associated to an environment follows the same pattern as seen earlier.

First use a management API call to get a list of all the virtual hosts defined for an environment.

Then use a separate API call to delete each virtual host that is no longer required.

TargetServer provisioning

Naming convention

Allowable characters in a TargetServer name are: a-zA-Z0-9._\-\$ %

Environment Configuration **test** ▾

[Caches](#) [Flow Hooks](#) [Key Value Maps](#) [References](#) **Target Servers** [TLS Keystores](#) [Virtual Hosts](#)

Name	Host	Port	Enabled	Actions
httpbin	<input type="text" value="httpbin.org"/>	<input type="text" value="80"/>	<input checked="" type="checkbox"/>	x Delete
httpbin2	<input type="text" value="httpbin.org"/>	<input type="text" value="443"/>	<input checked="" type="checkbox"/>	x Delete

[+ Target Server](#)

[Cancel](#) [Save](#)

When API proxies are deployed to multiple environments, TargetServers can be used to avoid the hardcoding of backend server hosts.

You create a TargetServer object with the same name in each environment. The same TargetServer definition in each environment would typically target a different backend server.

The API proxy simply references the TargetServer by the same name in each environment that the proxy is deployed to.

The TargetServer name must be unique within the environment.

As shown in the example, two TargetServers are defined in the test environment. Each TargetServer forwards API requests to the httpbin.org backend host on different ports.

You can manage TargetServers using the enterprise UI or management API.

TargetServer provisioning

Create TargetServer

Management API to create a TargetServer:

```
curl -v -u <adminEmail> -X POST \  
-H "Content-Type: application/json" \  
http://<ms_IP>:8080/v1/o/<org_name>/e/  
<env_name>/targetservers \  
-d @<json_payload_file>
```

Sample json payload:

```
{  
  "name" : "{target}",  
  "host" : "{hostname}",  
  "isEnabled" : {true | false},  
  "port" : {port_num},  
  "sSLInfo": {  
    "enabled": "true | false",  
    "clientAuthEnabled": "true | false",  
    "keyStore": "{keystore_name}",  
    "trustStore": "{truststore_name}",  
    "keyAlias": "{key_alias}",  
    "ignoreValidationErrors": "true | false",  
    "ciphers": [ "{cipher_1}", "{cipher_2}", ... ],  
    "protocols": [ "{protocol_1}", "{protocol_2}", ... ]  
  }  
}
```

<https://apidocs.apigee.com/docs/targetservers/1/overview>

You create a TargetServer definition by using the management API /targetservers and providing the organization and environment name in which the TargetServer is being created.

You supply a payload to the API that contains the name of the TargetServer, the backend server hostname, and port.

If the backend server is TLS enabled, you can also specify the keyStore, trustStore, and keyAlias names, the clientAuthEnabled flag for 2-way TLS, and the supported ciphers and protocols.

For more details on managing TargetServers, view the documentation at the link provided.

TargetServer provisioning

Delete TargetServer

- API call to get the TargetServers for an environment:

```
curl -v -u <adminEmail> \  
http://<ms_IP>:8080/v1/o/<org_name>/e/<env_name>/targetservers
```

- API call to delete the TargetServer:

```
curl -v -u <adminEmail> -X DELETE \  
http://<ms_IP>:8080/v1/o/<org_name>/e/<env_name>/targetservers/<target_server_name>
```

To delete a TargetServer, you first obtain a list of TargetServers in the environment using the /targetservers management API call and provide the organization and environment name in the URL.

You then delete a specific TargetServer using the same API and provide the TargetServer name in the URL.

You can also delete a TargetServer using the enterprise UI.

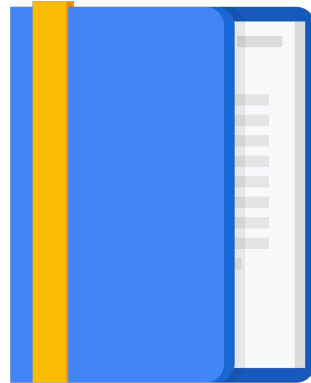
Agenda

Capacity planning

Provisioning

Expansion

Lab



In the previous lesson, we discussed how to add and remove logical capacity to Apigee Edge for private cloud by adding and deleting organizations, environments, virtual hosts, and TargetServers.

We will now discuss how to expand physical capacity for the Apigee private cloud platform.

In this lesson you learn how to add and remove physical components such as routers, message processors, and other Apigee private cloud components.

We also discuss adding a new regional data center.

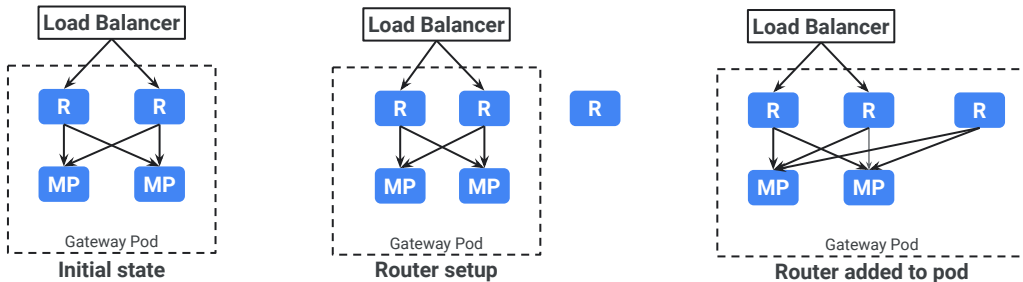
Add and remove a Router

Add Router

1. Set up Router:

```
/opt/apigee/apigee-setup/bin/setup.sh -p r -f <configfile>
```

2. The script automatically adds the router to the gateway pod.
3. The Ops team is responsible for adding the new router to the Load Balancer.



Let's first discuss how you add routers to and remove them from the platform.

In the example shown, the initial state consists of a load balancer that serves traffic to two routers.

To add a router, run the setup utility to install the router software component and specify the r, or router, profile and configuration file.

The router is added automatically to the gateway pod during setup.

After the router is added, it should also be added to the load balancing pod.

Add and remove a Router

Remove Router

1. Obtain Router UUID:
`curl -v http://<r_IP>:8081/v1/servers/self/uuid`
2. Stop the Router service:
`apigee-service edge-router stop`
3. De-register server's type:
`curl -v -u <adminEmail> -X POST http://<ms_IP>:8080/v1/servers -d \
"type=router®ion=<regionName>&pod=<podName>&uuid=<uuid>&action=remove"`
4. Delete the server:
`curl -v -u <adminEmail> -X DELETE http://<ms_IP>:8080/v1/servers/<uuid>`
5. If required, uninstall the component as described in the documentation:
<https://docs.apigee.com/private-cloud/latest/uninstalling-edge>

The process to delete a router has several steps.

First, obtain the UUID for the router using its server API as shown.

Next, stop the router service on the node on which the router is running.

Then, deregister the server type and provide the router UUID, type, region, pod name and the remove action to the management API call.

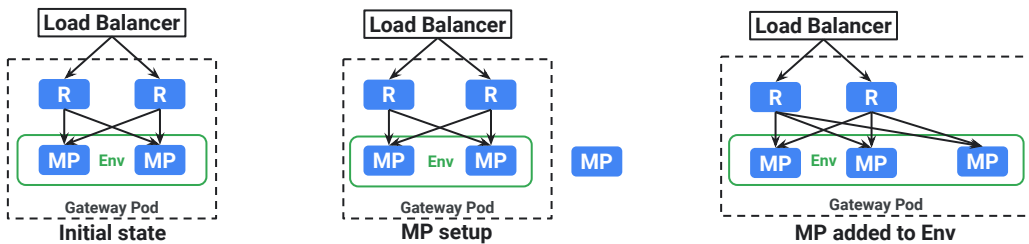
Finally, delete the server using the /servers management API call.

At this point, you can optionally uninstall the router software and decommission the hardware by following the Apigee private cloud documentation at the link provided.

Add and remove Message Processor

Add Message Processor

1. Set up Message Processor:
`/opt/apigee/apigee-setup/bin/setup.sh -p mp -f <configfile>`
2. Obtain Message Processor UUID:
`curl -v http://<mp_IP>:8082/v1/servers/self/uuid`
3. For each applicable Environment, add Message Processor to environment:
`curl -v -u <adminEmail> -H "Content-Type: application/x-www-form-urlencoded" \`
`-X POST http://<ms_IP>:8080/v1/o/<org_name>/e/<env_name>/servers -d "action=add&uuid=<mp_UUID>"`



Adding a message processor involves several steps.

First, you install the message processor software component using the Apigee setup utility and provide the mp, or message processor, profile and configuration file.

Next, obtain the UUID of the new server using the /servers API on the host.

At this point, the message processor is set up, but it's not associated with any environment.

Message processors can be associated with all environments or a subset of environments within a planet.

In this final step, you add the message processor to each of the environments that you need to process API traffic for.

Use the /servers management API and provide the organization and environment name, the server UUID, and add action to add the new message processor and associate it with the environment.

This step must be done for each environment that the message processor processes API traffic for.

Add and remove Message Processor

Remove Message Processor

1. Stop the Message Processor service:
`apigee-service edge-message-processor stop`
2. Deregister Message Processor from the organization's environments:
`curl -v -u <adminEmail> -X POST
http://<ms_IP>:8080/v1/o/<orgName>/e/<envName>/servers \
-d "uuid=<uuid>®ion=<regionName>&pod=<podName>&action=remove"`
3. Deregister server's type:
`curl -v -u <adminEmail> -X POST http://<ms_IP>:8080/v1/servers -d \
"type=message-processor®ion=<regionName>&pod=<podName>&uuid=<uuid>&action=remove"`
4. Delete the server:
`curl -v -u <adminEmail> -X DELETE http://<ms_IP>:8080/v1/servers/<uuid>`
5. If required, uninstall the message processor component as described in the documentation:
<https://docs.apigee.com/private-cloud/latest/uninstalling-edge>

The process of deleting a message processor also involves several steps.

First, stop the message processor service on the node.

Next, deregister the message processor from each of the environments that it is associated with using the /servers management API.

Provide the organization and environment name, the server UUID, region and pod name, and the remove action.

You then deregister the server type using the /servers management API and provide the message processor type and the same information as before.

Finally, you delete the server using the /servers management API and provide the server UUID.

At this point, you can also uninstall the message processor software and decommission the hardware.

Uninstalling Edge components is described in the Edge documentation at the link provided.

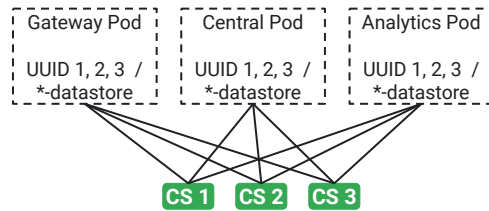
Add and remove Cassandra

Cassandra server registration

List server registration by Pod and Region using the following API:

```
curl -v -u <adminEmail> http://<ms_IP>:8080/v1/servers?region=<region>&pod=<pod>
```

```
{
  "internalIP" : "xxx.xxx.xxx.xxx",
  "isUp" : true,
  "pod" : "gateway",
  "reachable" : true,
  "region" : "dc-1",
  "tags" : {
    "property" : [ ]
  },
  "type" : [ "dc-datastore", "cache-datastore",
    "keyvaluemap-datastore", "kms-datastore",
    "counter-datastore" ],
  "uUID" : "c6d1c0d2-ffa5-4ffe-b950-9b06b75e41eb"
}
```



Cassandra servers along with references to specific keyspaces are registered to each pod in Apigee Edge.

The gateway, central, and analytics pods all contain references to Cassandra.

The example displays a partial output of a management API call to describe the Gateway pod.

The JSON response contains the pod name, the region, and the server UUID. It contains all keyspaces, listed as types, associated with that server and pod.

This server registration process is performed during platform installation.

Add and remove Cassandra

Cassandra node configuration

In addition to the logical server registration performed during Edge installation, Cassandra components use a local configuration file that describes the topology.

When nodes are added to or removed from the ring, these files must be updated.

```
/opt/apigee/apigee-cassandra/conf/cassandra-topology.properties
```

```
/opt/apigee/apigee-cassandra/conf/cassandra.yaml
```

Cassandra uses local configuration files to describe its topology. The files are named `cassandra-topology.properties` and `cassandra.yaml` and are located in the `/opt/apigee/apigee-cassandra/conf` directory.

If nodes are added to or removed from the Cassandra ring, these files must be updated.

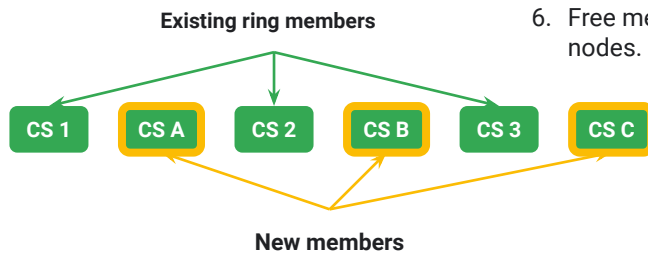
Changes to these files are managed by the `setup.sh` script and are updated when changes are made to the private cloud topology.

Manual updates to these files should never be required.

Add and remove Cassandra

Cassandra ring expansion process:

1. Re-configure the existing Cassandra nodes.
2. Install Cassandra on the new nodes.
3. Rebuild the data on the new Cassandra nodes from existing nodes.
4. Re-configure the Management Server.
5. Restart Routers and Message Processors.
6. Free memory on the existing Cassandra nodes.



Horizontal scalability is one of the key reasons that Cassandra is used as the underlying storage technology for Apigee Edge.

When you expand the Cassandra ring, the ring is either doubled or increased in size by an increment of the replication factor. The default replication factor used in Apigee Edge is three.

Expanding a regular three-node Cassandra ring implies that you will either double the number of nodes to six, or increase the number of nodes to 6, 9, 12, etc.

The example shows a three-node Cassandra ring being expanded to six nodes. The original three nodes are labeled CS1, CS2, and CS3. The new nodes are labeled CSA, CSB, and CSC.

The new nodes are placed between the existing nodes, with the goal being to split ranges and data ownership between the existing and newly added nodes.

To expand the ring, you must first re-configure the existing Cassandra nodes.

Then, install the Cassandra software on the new nodes using the Apigee setup utility.

Next, you re-build the data on the new nodes using data from the existing nodes.

You then re-configure the Edge management server with the new topology and restart

all the routers and message processors.

Finally, you free up memory on the existing Cassandra nodes.

Add and remove Cassandra

Adding Cassandra nodes: update configuration file

Current configuration file:

```
IP1=<node 1>
IP2=<node 2>
IP3=<node 3>

HOSTIP="$(hostname -i)"
...
CASS_HOSTS="$IP1:1,1 $IP2:1,1 $IP3:1,1"
...
BIND_ON_ALL_INTERFACES="y"
```



Updated configuration file:

```
IP1=<node 1>
IP2=<node 2>
IP3=<node 3>
IP10=<node 10>
IP11=<node 11>
IP12=<node 12>

HOSTIP="$(hostname -i)"
...
CASS_HOSTS="$IP1:1,1 $IP10:1,1 $IP2:1,1 $IP11:1,1
$IP3:1,1 $IP12:1,1"
...
BIND_ON_ALL_INTERFACES="y"
```

To reconfigure the existing Cassandra nodes, update the configuration file that was used during the original installation to include the new Cassandra nodes.

The sample partial configuration file shows the three additional nodes whose IP addresses are configured as the value of the properties IP10, IP11, and IP12.

The CASS_HOSTS property is also updated to include the IP addresses of the new nodes, which are placed between the original Cassandra nodes whose IP addresses were provided in the properties IP1, IP2, and IP3.

Add and remove Cassandra

Adding Cassandra nodes: reconfigure the existing Cassandra nodes

Execute the command to update one server at a time:

```
/opt/apigee/apigee-setup/bin/setup.sh -p c -f <updatedconfigfile>
```

To update the existing Cassandra nodes, run the apigee setup utility, specifying c as the Cassandra profile, and provide the path to the updated configuration file.

Add and remove Cassandra

Adding Cassandra nodes: install new Cassandra nodes

Install the new nodes using the following commands:

1. `curl https://software.apigee.com/bootstrap_<version>.sh -o /tmp/edge/bootstrap_<version>.sh`
2. `bash /tmp/edge/bootstrap_<version>.sh apigeeuser=<uName> apigeepassword=<pWord>`
3. `/opt/apigee/apigee-service/bin/apigee-service apigee-setup install`
4. `/opt/apigee/apigee-setup/bin/setup.sh -p c -f <updatedconfigfile>`

To install the new Cassandra nodes, follow almost the same process as used during installation.

Download the bootstrap script if you didn't keep a copy of the one used during the original installation.

Make the file executable and run the bootstrap script.

After the bootstrap script has been executed, install the apigee setup script using the apigee service utility.

Finally, run the setup script using the Cassandra profile and provide the path to the updated configuration file.

Add and remove Cassandra

Adding Cassandra nodes: rebuild the data on new Cassandra nodes

Rebuild the new Cassandra nodes, specifying the region name.

On the first node, run:

```
/opt/apigee/apigee-cassandra/bin/nodetool -h <nodeIP> rebuild <regionName>
```

where:

nodeIP is the IP address of the Cassandra node.

regionName is the value of the REGION property in the configuration file.

Repeat this step on the remaining new Cassandra nodes, one at a time.

Next, rebuild the data on the new Cassandra nodes.

To do this, run the Cassandra nodetool command on the first node and provide its IP address and the region name.

The region is used as a source region for streaming data during the rebuild and is the value of the REGION property in the configuration file.

Perform this step on each of the new Cassandra nodes.

Add and remove Cassandra

Adding Cassandra nodes: re-configure Management Server

On a management server node, execute `setup.sh` to update the management server for the newly added Cassandra nodes:

```
/opt/apigee/apigee-setup/bin/setup.sh -p ms -f <updatedconfigfile>
```

After the new nodes have been rebuilt, the management server must be re-configured to be aware of the new topology.

This is done using the apigee setup utility script.

Run the setup script, specifying the management server ms profile, and provide the path to the updated configuration file that contains the new Cassandra topology.

Add and remove Cassandra

Adding Cassandra nodes: restart all Routers and Message Processors

On all Routers:

```
/opt/apigee/apigee-service/bin/apigee-service edge-router restart
```

On all Message Processors:

```
/opt/apigee/apigee-service/bin/apigee-service edge-message-processor restart
```

After the topology is updated, you must restart all the routers and message processors so that they become aware of the new Cassandra topology.

This is done one server at a time using the apigee-service utility.

Add and remove Cassandra

Adding Cassandra nodes: clean up existing Cassandra nodes

On the existing Cassandra nodes, run the nodetool cleanup command to free up memory:

```
/opt/apigee/apigee-cassandra/bin/nodetool -h <nodeIP> cleanup
```

As the final step in the process, remove any unwanted data following the addition of new nodes to the cluster.

This is done using the nodetool command with the clean up option.

When new nodes are added, existing nodes lose parts of the partition range they are responsible for.

Cassandra does not automatically clean this data up, which has an unwanted effect during rebalancing because Cassandra includes the old data as part of the load on the node.

Note that during the cleanup process, there will be an increase in disk usage that is proportional to the size of the largest SS table and an increase in disk I/O.

Add and remove Cassandra

Removing Cassandra nodes: update configuration file

Current configuration file:

```
IP1=<node 1>
IP2=<node 2>
IP3=<node 3>
IP10=<node 10>
IP11=<node 11>
IP12=<node 12>

HOSTIP="$(hostname -i)"
...
CASS_HOSTS="$IP1:1,1 $IP10:1,1 $IP2:1,1 $IP11:1,1
$IP3:1,1 $IP12:1,1
...
BIND_ON_ALL_INTERFACES="y"
```



Updated configuration file:

```
IP1=<node 1>
IP2=<node 2>
IP3=<node 3>

HOSTIP="$(hostname -i)"
...
CASS_HOSTS="$IP1:1,1 $IP2:1,1 $IP3:1,1"
...
BIND_ON_ALL_INTERFACES="y"
```

Let's now discuss the process to remove Cassandra nodes.

First, we update the configuration file used during the original installation.

To illustrate the process, we will remove the nodes marked IP10, IP11, and IP12 and update the configuration file accordingly.

Add and remove Cassandra

Removing Cassandra nodes: update Management Server

Update the topology on the Management Server:

```
/opt/apigee/apigee-setup/bin/setup.sh -p ms -f <updatedconfigfile>
```

We use the new configuration file to update the Cassandra topology on the management server by running the apigee setup utility.

Use the MS profile for the management server, and provide the path to the updated configuration file.

Add and remove Cassandra

Removing Cassandra nodes: restart components

Perform a rolling restart of all Edge components (R, MP, QS, PS):

```
/opt/apigee/apigee-service/bin/apigee-all restart
```

The restart forces components to:

- Read the latest pod wiring information
- Remove references to the deleted Cassandra nodes

After the management server has been updated, perform a rolling restart of all Edge components except Cassandra and Zookeeper.

The restart forces components to reread the latest pod wiring information and to remove references to the deleted Cassandra nodes.

Add and remove Cassandra

Removing Cassandra nodes: decommission Cassandra nodes

To decommission a node, run the nodetool command:

```
/opt/apigee/apigee-cassandra/bin/nodetool -h <nodeIP> decommission
```

The decommission command deactivates a node by streaming its data to another node.

Repeat the process on the other nodes to be removed, one at a time.

Next, we decommission the Cassandra nodes that are being removed.

This process is performed on one node at a time.

To decommission a node, run the nodetool command and provide the node IP address with the decommission option.

This deactivates the node by streaming its data to the next node in the ring.

Add and remove Cassandra

Removing Cassandra nodes: update remaining Cassandra nodes

Update configuration on the remaining Cassandra nodes:

```
/opt/apigee/apigee-setup/bin/setup.sh -p c -f <updatedconfigfile>
```

Finally, update the local configuration file on the remaining Cassandra nodes.

To do this, use the apigee setup script with the Cassandra c profile and provide the path to the updated configuration file.

Add and remove Cassandra

Removing Cassandra nodes: uninstall Cassandra

If uninstalling the component is required, follow the documentation:

<https://docs.apigee.com/private-cloud/latest/uninstalling-edge>

Optionally, we can now completely decommission the Cassandra nodes by uninstalling the Cassandra software component from the nodes.

To do this, follow the documentation at the link provided.

Add and remove ZooKeeper

Add ZooKeeper node: update configuration file

Original configuration file:

```
IP1=<node 1>
IP2=<node 2>
IP3=<node 3>

HOSTIP=$(hostname -i)
ADMIN_EMAIL=opdk@apigee.com
...
ZK_HOSTS="$IP1 $IP2 $IP3"
ZK_CLIENT_HOSTS="$IP1 $IP2 $IP3"
...
BIND_ON_ALL_INTERFACES="y"
```



Updated configuration file:

```
IP1=<node 1>
IP2=<node 2>
IP3=<node 3>
IP14=<node 14>
IP15=<node 15>
IP16=<node 16>

HOSTIP=$(hostname -i)
ADMIN_EMAIL=opdk@apigee.com
...
ZK_HOSTS="$IP1 $IP2 $IP3 $IP14 $IP15 $IP16:observer"
ZK_CLIENT_HOSTS="$IP1 $IP2 $IP3 $IP14 $IP15 $IP16"
...
BIND_ON_ALL_INTERFACES="y"
```

Over time, as your planet expands with more servers, additional organizations and environments, API proxies and other objects, you may need to increase the number of ZooKeeper nodes in your private cloud cluster.

Typically, the ZooKeeper components are added to the server that hosts the Cassandra components.

The first step in the process to add new ZooKeeper nodes is to create an updated configuration file.

In the example, we create three new ZooKeeper nodes whose IP addresses are added to the configuration file as values of the properties IP14, IP15, and IP16.

With ZooKeeper, it is crucial to remember that an odd number of voters must be maintained.

Here, the node IP16 is marked as an observer, which indicates that it can respond to read requests, but will not vote during a write request.

Add and remove ZooKeeper

Add ZooKeeper node: install new ZooKeeper nodes

Install new ZooKeeper nodes using the updated configuration file:

```
/opt/apigee/apigee-setup/bin/setup.sh -p zk -f <updatedconfigfile>
```

With the updated config file in place, set up the new ZooKeeper nodes using the apigee-setup utility, with the ZooKeeper profile ZK, and provide the path to the configuration file.

Add and remove ZooKeeper

Add ZooKeeper node: update existing ZooKeeper nodes

Update configuration on each of the existing ZooKeeper nodes:

```
/opt/apigee/apigee-setup/bin/setup.sh -p zk -f <updatedconfigfile>
```

After the new nodes are installed, the existing ZooKeeper nodes must be updated to reflect the new configuration.

On each existing ZooKeeper node, run the apigee-setup utility and provide the ZooKeeper ZK profile and path to the updated configuration file.

Add and remove ZooKeeper

Add ZooKeeper node: restart all ZooKeeper nodes

Restart all ZooKeeper nodes one at a time:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-zookeeper restart
```

After all the existing ZooKeeper nodes are updated, perform a rolling restart of all ZooKeeper nodes in the cluster using the apigee-service utility.

Add and remove ZooKeeper

Add ZooKeeper node: update Apigee Edge components

For each of the components listed, reconfigure the component and restart it:

- edge-management-server(profile: [ms](#))
- edge-router (profile: [r](#))
- edge-message-processor (profile: [mp](#))
- edge-qpuid-server (profile: [qs](#))
- edge-postgres-server (profile: [ps](#))

Reconfigure components:

```
/opt/apigee/apigee-setup/bin/setup.sh -p <component> -f <updatedconfigfile>
```

Restart components:

```
/opt/apigee/apigee-service/bin/apigee-service <component> restart
```

In addition to the ZooKeeper nodes, the other Apigee Edge components must also be reconfigured and restarted.

To re-configure each component, run apigee-setup and provide the appropriate component profile and path to the updated configuration file.

To restart the component, run the apigee-service utility with the component name and use the restart command option.

The components should be restarted serially in order to prevent any service interruption, starting with the management server.

Add and remove ZooKeeper

Remove ZooKeeper node: update configuration file

Original configuration file:

```
IP1=<node 1>
IP2=<node 2>
IP3=<node 3>
IP14=<node 14>
IP15=<node 15>
IP16=<node 16>

HOSTIP=$(hostname -i)
ADMIN_EMAIL=opdk@apigee.com
...
ZK_HOSTS="$IP1 $IP2 $IP3 $IP14 $IP15 $IP16:observer"
ZK_CLIENT_HOSTS="$IP1 $IP2 $IP3 $IP14 $IP15 $IP16"
...
BIND_ON_ALL_INTERFACES="y"
```



Updated configuration file:

```
IP1=10.10.0.1
IP2=10.10.0.2
IP3=10.10.0.3

HOSTIP=$(hostname -i)
ADMIN_EMAIL=opdk@apigee.com
...
ZK_HOSTS="$IP1 $IP2 $IP3"
ZK_CLIENT_HOSTS="$IP1 $IP2 $IP3"
...
BIND_ON_ALL_INTERFACES="y"
```

To remove ZooKeeper nodes, you first update the configuration file to remove the ZooKeeper node IP address properties and their references.

In the updated configuration file, we remove three of the ZooKeeper nodes.

Note that with only three nodes, there is no need to tag a node as an observer.

Add and remove ZooKeeper

Remove ZooKeeper node: update remaining ZooKeeper nodes

Update ZooKeeper configuration on each of the remaining nodes:

```
/opt/apigee/apigee-setup/bin/setup.sh -p zk -f <updatedconfigfile>
```

Removing nodes follows a similar pattern to adding nodes.

First, update each of the remaining nodes with the new configuration by running apigee-setup with the ZooKeeper ZK profile and providing the path to the updated configuration file.

Add and remove ZooKeeper

Remove ZooKeeper node: restart ZooKeeper nodes

Restart all ZooKeeper nodes one at a time:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-zookeeper restart
```

We then restart all the ZooKeeper nodes one after the other using the apigee-service utility apigee-zookeeper restart command.

Add and remove ZooKeeper

Remove ZooKeeper node: stop deleted ZooKeeper nodes

Stop all deleted ZooKeeper nodes:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-zookeeper stop
```

After all the remaining nodes have been reconfigured and restarted, stop the ZooKeeper service on each of the deleted ZooKeeper nodes.

This is done using the apigee-service utility `apigee-zookeeper stop` command.

Add and remove ZooKeeper

Remove ZooKeeper node: uninstall ZooKeeper

If uninstalling the component is required, follow the documentation:

<https://docs.apigee.com/private-cloud/latest/uninstalling-edge>

After being stopped, the ZooKeeper software component can be uninstalled and the node decommissioned.

To uninstall the component, follow the documentation at the link provided.

Add and remove Zookeeper

Remove Zookeeper node: update Edge components

For each of the components listed, reconfigure the component and restart it:

- Edge-management-server (profile: [ms](#))
- Edge-router (profile: [r](#))
- Edge-message-processor (profile: [mp](#))
- Edge-qpuid-server (profile: [qs](#))
- Edge-postgres-server (profile: [ps](#))

Reconfigure components:

```
/opt/apigee/apigee-setup/bin/setup.sh -p <component> -f <updatedconfigfile>
```

Restart components:

```
/opt/apigee/apigee-service/bin/apigee-service <component> restart
```

The final step in the process is to reconfigure and restart the other Apigee Edge components.

To re-configure each component, run `apigee-setup` and provide the appropriate component profile and path to the updated configuration file.

To restart the component, run the `apigee-service` utility with the component name and use the restart command option.

As mentioned previously, the components should be restarted serially in order to prevent any service interruption.

Add and remove Qpid

Analytics groups

```
curl -u <sysAdminEmail> "http://<ms_IP>:8080/v1/analytics/groups/ax"
[ {
  "name" : "axgroup001",
  "properties" : {
    "consumer-type" : "ax"
  },
  "scopes" : [ "traininglab-prod" ],
  "uuids" : {
    "postgres-server" : [ "baf7d9d5-618c-4ce4-8c5b-23ad2c62eb51:489aa9b0-b764-4ade-8615-27db09e9deca" ],
    "qpid-server" : [ "708cf21f-2efe-41c9-97b5-809e56676347", "4f1c20e5-0dc5-450c-8619-3cd9fe75ce4b" ]
  },
  "consumer-groups" : [ {
    "name" : "consumer-group-001",
    "consumers" : [ "708cf21f-2efe-41c9-97b5-809e56676347", "4f1c20e5-0dc5-450c-8619-3cd9fe75ce4b" ],
    "datastores" : [ "baf7d9d5-618c-4ce4-8c5b-23ad2c62eb51:489aa9b0-b764-4ade-8615-27db09e9deca" ],
    "properties" : {
    }
  } ],
  "data-processors" : {
  }
} ]
```

Qpid is the message broker used by Apigee Edge's analytics platform.

Analytics uses a data structure called analytics groups that contains information on the association of Qpid and Postgres servers to a given organization and environment.

The sample analytics group shown has a single associated scope that specifies the prod environment within the training lab organization.

The scope lists two Qpid servers and two Postgres servers as its members.

Add and remove Qpid

Add Qpid server: get analytics group configuration

By default, the name of the analytics group is `axgroup-001`, and the name of the consumer group is `consumer-group-001`.

To list the names of the analytics and consumer groups, use the `apigee-adminapi` script:

```
apigee-adminapi.sh analytics groups list --admin <adminEmail> --pwd <adminPword>
--host localhost
```

The script returns the analytics group name in the `name` field and the consumer group name in the `consumer-groups` field.

Adding new Qpid nodes to the Apigee private cloud platform involves several steps.

The first step is to determine the analytics and consumer group names.

In the default installation, these are set to `axgroup-001` and `consumer-group-001`.

To list the names that are used, use the `apigee-adminapi.sh` script as shown.

The script returns the names of the analytics and consumer groups that are used.

Add and remove Qpid

Add Qpid server: install new Qpid server

Using the configuration file used to install other components on the planet, install the new Qpid server:

```
/opt/apigee/apigee-setup/bin/setup.sh -p qs -f <configfile>
```

Get the UUID of the Qpid server:

```
curl http://<qs_IP>:8083/v1/servers/self/uuid
```

The next step in the process is to install the Qpid software component.

Use the apigee-setup utility and provide the Qpid QS profile and the configuration file that was used for the Apigee private cloud installation.

After the setup is complete, invoke the /servers API on the Qpid server to obtain its UUID, which is used in the next step.

Add and remove Qpid

Add Qpid server: add Qpid nodes

- Add the Qpid server to the analytics group:

```
curl -v -u <sysAdminEmail> -H "Content-Type: application/json" -X POST \
"http://<ms_IP>:8080/v1/analytics/groups/ax/<AX_GROUP>/servers?uuid=<QPID_UUID>&type=qpid-server"
```

- Add the Qpid server to the consumer group:

```
curl -v -u <sysAdminEmail> -H "Content-Type: application/json" -X POST \
"http://<ms_IP>:8080/v1/analytics/groups/ax/<AX_GROUP>/consumer-groups/<CONSUMER_GROUP>/consumers?uuid=<QPID_UUID>"
```

Using the information gathered, add the new Qpid server to the analytics and consumer groups.

This is done using the /analytics/groups management API.

We first add the Qpid server to the analytics group by invoking the API's servers endpoint, and provide the name of the analytics group, the UUID of the Qpid server, and the qpid-server type.

We then add the Qpid server to the consumer group using the API's consumer-groups endpoint, and provide the name of the analytics group, the name of the consumer group, and the UUID of the Qpid server.

At this point, the new Qpid server will start to receive analytics events from the message processors that are associated with the scope served by the analytics group.

Add and remove Qpid

Add Qpid server:

Restart Qpid server:

```
/opt/apigee/apigee-service/bin/apigee-service edge-qpid-server restart
```

The last step is to restart all the edge-qpid-server components on all nodes so that the changes are picked up by those components.

To restart the component, use apigee-service with the edge-qpid-server component name and the restart command option.

Add and remove Qpid

Remove Qpid server: get analytics group configuration

To list the names of the analytics and consumer groups, use the apigee-adminapi script:

```
apigee-adminapi.sh analytics groups list --admin <adminEmail> --pwd <adminPword>
--host localhost
```

The script returns the analytics group name in the name field and the consumer group name in the consumer-groups field.

Get the UUID of the Qpid server to be removed:

```
curl http://<ms_IP>:8083/v1/servers/self/uuid
```

To remove a Qpid server from the cluster, you follow the same process to first determine the analytics and consumer group names.

Next, use the servers API on the Qpid server that is being removed to get its UUID.

The UUID will be used in the next set of steps.

Add and remove Qpid

Remove Qpid server: remove Qpid nodes

- Remove Qpid server from the consumer group:

```
curl -v -u <sysAdminEmail> -H "Content-Type: application/json" -X DELETE \
"http://<ms_IP>:8080/v1/analytics/groups/ax/<AX_GROUP>/consumer-groups/<CONSUMER_GROUP>/consumers/<QPID_UUID>"
```
- Remove Qpid server from the analytics group:

```
curl -v -u <sysAdminEmail> -X DELETE
"http://<ms_IP>:8080/v1/analytics/groups/ax/<AX_GROUP>/servers?uuid=<QPID_UUID>&type=qpid-server"
```
- Remove Qpid server from the planet:

```
curl -v -u <sysAdminEmail> -X DELETE "http://<ms_IP>:8080/v1/servers/<QPID_UUID>"
```

Using the group names and the UUID information, first delete the Qpid server from the consumer group using the /analytics/groups management API /consumer-groups endpoint.

Next, delete the Qpid server from the analytics group using the same management API's /servers endpoint.

In the last step, delete the Qpid server from the Apigee Edge planet.

Add and remove Qpid

Remove Qpid server: restart all Qpid servers

Restart all Qpid servers one at a time:

```
/opt/apigee/apigee-service/bin/apigee-service edge-qpid-server restart
```

Finally, restart all the edge-qpid-server components on all nodes so that the change is picked up by those components.

To restart the component, use apigee-service with the edge-qpid-server component name and the restart command option.

Add and remove Qpid

Remove Qpid server:

If uninstalling the component is required, follow the documentation:

<https://docs.apigee.com/private-cloud/latest/uninstalling-edge>

If required, the Qpid software component can be uninstalled and the server decommissioned.

To do this, follow the steps documented at the link provided.

Adding a region

Overview

- The steps to add a new region are very similar to the individual steps required to horizontally scale the planet by adding one component at a time.
- In this section, we focus on the “process” required to add a new region, instead of the individual commands needed for each step.
- A detailed example of adding a region is available as part of the official documentation:
<https://docs.apigee.com/private-cloud/latest/adding-data-center>.



Let's now explore how to add a region to your Apigee Edge planet.

We'll start with an overview and discuss the prerequisites, describe a scenario, create a configuration file, and discuss the process to add a new region.

The steps to add a region are very similar to the individual steps needed to horizontally scale the planet one component at a time.

In this section, we focus on the process required to add the region, instead of the specific commands for adding each component.

For details on adding individual components, refer to the previous topics in this module. There is also detailed documentation on adding a region at the link provided.

We will extend an example by describing the prerequisites, processes, and considerations needed to successfully add a new region.

Adding a region

Prerequisites

Many of the design principles discussed as part of topology design apply to the addition of regions to an existing planet.

Before adding a new region you must:

- Produce a topology design diagram that clearly describes all regions (old and new) on the Edge planet.
- Understand the basic principles that govern Apigee Edge data replication:
 - Openldap replication
 - Cassandra ring topology requirements
 - ZooKeeper ensemble requirements
 - Postgres replication
- Consider the steps involved to:
 - add components in the new region,
 - reconfigure components in the existing regions,
 - and update server registrations.
- Comply with cross-region component connectivity requirements as described in the Edge Installation Guide: <https://docs.apigee.com/private-cloud/latest/installation-requirements>

Before you plan to add additional regions to your Apigee Edge private cloud installation, there are several prerequisites to consider.

When you add new regions, many of the previously discussed principles on topology design apply.

You must produce a topology diagram that clearly describes all the existing regions, and include the new regions to be added.

Understand the basic principles governing data replication in Apigee Edge.

These include multi-master replication for Openldap and Cassandra ring topology requirements. Every region, datacenter, or availability zone must have an equal number of Cassandra nodes.

You must understand ZooKeeper cluster requirements, because ZooKeeper nodes across all regions are part of the same ensemble or cluster. There should be an odd number of voting nodes across the cluster.

The original topology may have had one or more PostgreSQL servers deployed in an active or active standby configuration. As part of the region expansion, you must decide whether new PostgreSQL servers will be required.

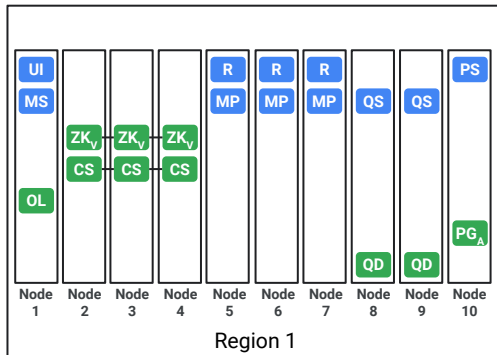
You must also consider that adding a new region requires the addition of new

components, the reconfiguration of the existing components, and updating server registrations.

And finally, you must ensure that the new topology complies with all cross-region component connectivity requirements. Documentation describing this is at the link provided.

Add region

Scenario: current topology



To illustrate the process of adding a new region, we:

- Start with a single region (current topology)
- Add a second region (target topology) with:
 - Similar configuration
 - Same number of nodes

Let's now discuss a scenario that describes adding a region.

The scenario illustrates the process to expand our topology of a single region planet by adding a new region, which is our target topology.

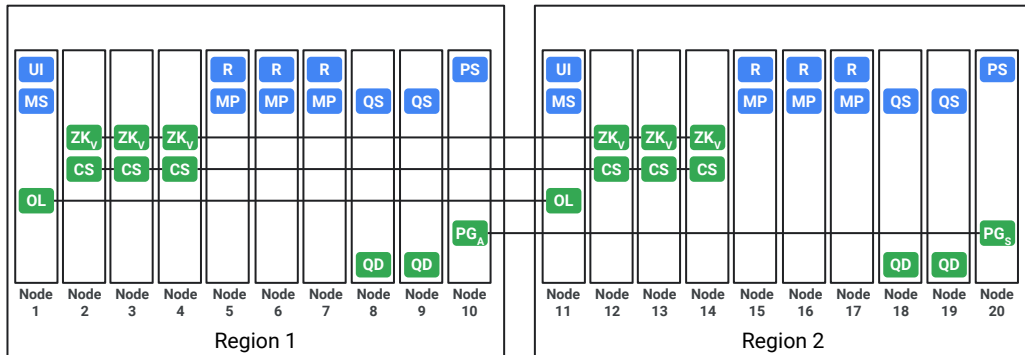
The second region will have a similar configuration and an equal number of nodes as in the current region.

The current topology has a single management server, UI, and an Openldap server. It also has ZooKeeper and Cassandra clusters, each of which is installed on three nodes.

It also contains three router message processor nodes, two Qpid daemons, two Qpid servers, a single PostgreSQL database, and Postgres server.

Add region

Scenario: target topology



This is the target topology or our desired state.

We will use an equal number of nodes of the same type in the new region, region two.

As shown in the diagram, connectivity is required between the Openldap servers, between the Zookeeper and Cassandra clusters, and between the active and standby PostgreSQL servers across the regions.

As previously mentioned, we maintain single ZooKeeper and Cassandra cluster configurations across both regions.

Add region

Create configuration files

- Create a configuration file per region. If the current planet's configuration file is available, modify it to add references to:
 - Openldap peer
 - ZooKeeper hosts in region 2
 - Cassandra hosts in region 2

/tmp/apigee/edge-config-dc1.txt

```
...
MSIP="$DC1IP1"
...
LDAP_TYPE="2"
LDAP_SID="1"
LDAP_PEER="$DC2IP1"
...
REGION="dc-1"
ZK_HOSTS="$DC1IP2 $DC1IP3 $DC1IP4 $DC2IP2 $DC2IP3 $DC2IP4:observer"
ZK_CLIENT_HOSTS="$DC1IP2 $DC1IP3 $DC1IP4"
CASS_HOSTS="$DC1IP2:1,1 $DC1IP3:1,1 $DC1IP4:1,1 $DC2IP2:2,1 $DC2IP3:2,1
$DC2IP4:2,1"
...
```

/tmp/apigee/edge-config-dc2.txt

```
...
MSIP="$DC2IP1"
...
LDAP_TYPE="2"
LDAP_SID="2"
LDAP_PEER="$DC1IP1"
...
REGION="dc-2"
ZK_HOSTS="$DC1IP2 $DC1IP3 $DC1IP4 $DC2IP2 $DC2IP3 $DC2IP4:observer"
ZK_CLIENT_HOSTS="$DC2IP2 $DC2IP3 $DC2IP4"
CASS_HOSTS="$DC2IP2:2,1 $DC2IP3:2,1 $DC2IP4:2,1 $DC1IP2:1,1 $DC1IP3:1,1
$DC1IP4:1,1"
...
```

With the target topology defined, we can now create the new configuration files that are needed to expand the planet.

Each region has a configuration file associated with it. Updated configuration files are required for both the new and the existing region.

Modify the original installation configuration file to add references to the Openldap peer in region two, the new ZooKeeper hosts in region two, and the new Cassandra hosts in region two.

Add region

Create configuration files

/tmp/apigee/edge-config-dc1.txt

```
HOSTIP="$(hostname -i)"
MSIP="$DC1IP1"
ADMIN_EMAIL="<email@example.com>"
APIGEE_ADMINPW="<password>"
LICENSE_FILE="/tmp/edge/license.txt"
USE_LDAP_REMOTE_HOST="n"
LDAP_TYPE="2"
LDAP_SID="1"
LDAP_PEER="$DC2IP1"
APIGEE_LDAPPW="<password>"
MP_POD="gateway"
REGION="dc-1"
ZK_HOSTS="$DC1IP2 $DC1IP3 $DC1IP4 $DC2IP2 $DC2IP3 $DC2IP4:observer"
ZK_CLIENT_HOSTS="$DC1IP2 $DC1IP3 $DC1IP4"
CASS_HOSTS="$DC1IP2:1,1 $DC1IP3:1,1 $DC1IP4:1,1 $DC2IP2:2,1 $DC2IP3:2,1 $DC2IP4:2,1"
PG_MASTER="$DC1IP5"
PG_STANDBY="$DC2IP5"
SKIP_SMT="y"
SMTPHOST="smtp.example.com"
SMTPPORT="25"
SMTPUSER="smtp@example.com"
SMTPPASSWORD="<password>"
SMTPSSL="n"
BIND_ON_ALL_INTERFACES="y"
```

/tmp/apigee/edge-config-dc2.txt

```
HOSTIP="$(hostname -i)"
MSIP="$DC2IP1"
ADMIN_EMAIL="<email@example.com>"
APIGEE_ADMINPW="<password>"
LICENSE_FILE="/tmp/edge/license.txt"
USE_LDAP_REMOTE_HOST="n"
LDAP_TYPE="2"
LDAP_SID="2"
LDAP_PEER="$DC1IP1"
APIGEE_LDAPPW="<password>"
MP_POD="gateway"
REGION="dc-2"
ZK_HOSTS="$DC1IP2 $DC1IP3 $DC1IP4 $DC2IP2 $DC2IP3 $DC2IP4:observer"
ZK_CLIENT_HOSTS="$DC2IP2 $DC2IP3 $DC2IP4"
CASS_HOSTS="$DC2IP2:2,1 $DC2IP3:2,1 $DC2IP4:2,1 $DC1IP2:1,1 $DC1IP3:1,1 $DC1IP4:1,1"
PG_MASTER="$DC1IP5"
PG_STANDBY="$DC2IP5"
SKIP_SMT="y"
SMTPHOST="smtp.example.com"
SMTPPORT="25"
SMTPUSER="smtp@example.com"
SMTPPASSWORD="<password>"
SMTPSSL="n"
BIND_ON_ALL_INTERFACES="y"
```

Here are the completed configuration files for both regions. Let's review the highlighted changes.

The MSIP variable always refers to the management server in the local region.

In the set of Openldap variables, the LDAP TYPE variable is set to two in both regions, which indicates a replicated configuration.

For a single region, this is set to one. The LDAP SID variable is a unique identifier and must be set to different values in each region.

The LDAP PEER variable contains the IP address of the remote peer to replicate with. Region one refers to the IP address of the Openldap server in region two, and vice versa.

The region variable sets an identifier for the region, normally set as DC one and DC two, but these can be re-named.

ZooKeeper has two sets of variables per region. The ZK hosts variable contains all the ZooKeeper hosts in the ensemble.

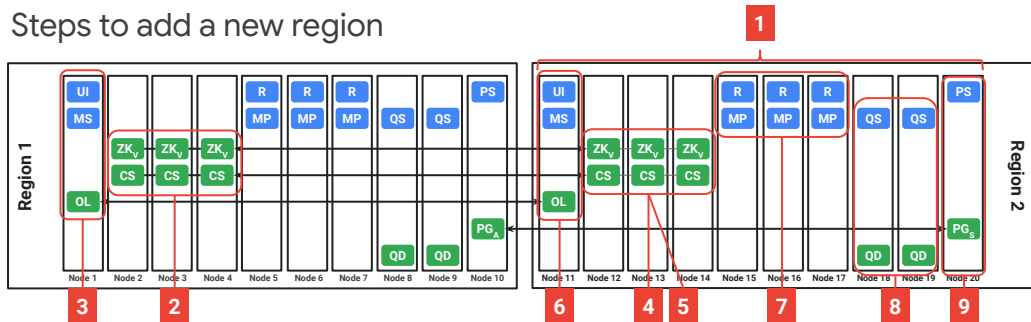
Note that the host designated DC2IP4 is listed as an observer in order to preserve an odd number of voting members. The ZK client hosts variable contains the ZooKeeper members for that region.

Cassandra has a single variable CASS HOSTS that describe its topology, but the value is different in each region. The variable lists hosts in its own region first, followed by the hosts in its peer regions.

Lastly, the PostgreSQL variables have the same value in each region.

Add region

Steps to add a new region



On region 1:

2. Rerun setup.sh datastore nodes (ZK, CS nodes).
3. Rerun setup.sh on the management server node (UI, MS, OL).

On region 2:

1. Perform bootstrap and install apigee-setup in all nodes.
4. Install datastore nodes (ZK, CS nodes).
5. In all CS nodes, perform `nodetool -h <host> rebuild dc-1`.
6. Install Management Server node (UI, MS, OL).
7. Install Routes and Message Processors (R, MP).
8. Install Qpid server (QS, QD).
9. Install Postgres server (PS, PG).

After the topology is finalized and the configuration files are created or updated, we can now add the new region. We first start the process in the new region.

In Step one, bootstrap the nodes and run apigee-setup on all nodes in region two.

With the new Cassandra and ZooKeeper member nodes in place, in step 2 rerun setup on the datastore nodes in region one.

In step three, run setup to reconfigure the management server in region one. This step updates the management server with new server registrations and also informs the Openldap server that there is a new peer to replicate with.

In step four, install the new Cassandra and ZooKeeper datastore nodes in region **two** using the apigee-setup utility.

Step five rebuilds Cassandra in region **two**. On each node, the `nodetool rebuild` command is run providing the name of region one, DC one, as the source region for streaming data. Note that this step is both disk- and network-intensive.

In step six, install the new management server, UI, and Openldap. During this process, existing data from the Openldap server in region one is copied and continuous replication initiated.

In step seven, install the router and message processor nodes in region two. This can

be done in parallel.

In step eight, install the Qpid servers. These will be added to the analytics groups in a later step.

And finally in step nine, install the PostgreSQL and Postgres servers.

Add region

Enable PostgreSQL replication

Set up Postgres active/standby for the PostgreSQL nodes.

On region 1:

- Enable replication on the active server:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-postgresql  
setup-replication-on-master -f <configfile>
```

On region 2:

- Stop the server, and then delete any existing PostgreSQL data:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-postgresql stop  
rm -rf /opt/apigee/data/apigee-postgresql/
```
- Configure the standby node:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-postgresql  
setup-replication-on-standby -f <configfile>
```

At this stage in the process our new region setup is almost complete.

A few additional steps still need to be completed.

Region one now contains the PostgreSQL active server, and region two contains a standby server. We must now enable replication for PostgreSQL.

To achieve this, first enable replication on the active PostgreSQL server using the `apigee-service` utility with the `apigee postgresql` and `setup replication` command options, and provide the path to region one's configuration file.

Next, stop the new PostgreSQL server in region two using `apigee-service` and delete any existing PostgreSQL data.

Finally, configure the standby server using the `apigee-service` utility with the `apigee postgresql` and `setup replication standby` command options, and provide the path to region two's configuration file.

Add region

Update analytics configuration

```
curl -u <sysAdminEmail>:<passwd> "http://<ms_IP>:8080/v1/analytics/groups/ax"
[ {
  "name" : "axgroup001",
  "properties" : {
    "consumer-type" : "ax"
  },
  "scopes" : [ "traininglab-prod" ],
  "uuids" : {
    "postgres-server" : [ "baf7d9d5-618c-4ce4-8c5b-23ad2c62eb51:489aa9b0-b764-4ade-8615-27db09e9deca" ],
    "qpid-server" : [ "708cf21f-2efe-41c9-97b5-809e56676347", "4f1c20e5-0dc5-450c-8619-3cd9fe75ce4b" ]
  },
  "consumer-groups" : [ {
    "name" : "consumer-group-001",
    "consumers" : [ "708cf21f-2efe-41c9-97b5-809e56676347", "4f1c20e5-0dc5-450c-8619-3cd9fe75ce4b" ],
    "datastores" : [ "baf7d9d5-618c-4ce4-8c5b-23ad2c62eb51:489aa9b0-b764-4ade-8615-27db09e9deca" ],
    "properties" : {
    }
  } ],
  "data-processors" : {
  }
}]
```

We next update the analytics configuration.

Apigee Edge uses analytics groups to map resources to organizations and environments and stores this data in PostgreSQL.

The analytics group axgroup001 has two Postgres and two Qpid servers listed in the server and the consumer groups.

To update the analytics configuration, follow the documentation on the Apigee private cloud website.

<https://docs.apigee.com/api-platform/troubleshoot/analytics/add-remove-postgres-nodes>
<https://docs.apigee.com/private-cloud/v4.50.00/adding-new-analytics-group>

Add region

Clean up Cassandra nodes

Run the nodetool utility on each Cassandra node in region 1 to recover disk space:

```
/opt/apigee/apigee-cassandra/bin/nodetool -h <cassandraIP> cleanup
```

The Cassandra ring expansion process leaves the original nodes with data they are no longer responsible for.

This data can be deleted by running the nodetool utility on each host with the cleanup command.

Add region

Re-configure organization and environments

Reconfigure organizations:

```
apigee-adminapi.sh orgs pods add -o <orgName> -r dc-2 -p gateway-2 --admin  
<adminEmail> --pwd <adminPword> --host localhost
```

Add the new Message Processors to the organization and environment:

```
apigee-adminapi.sh orgs envs servers add -o <orgName> -e <envName> -u UUID --admin  
<adminEmail> --pwd <adminPword> --host localhost
```

To support Apigee Edge organizations across both regions, we will need to re-configure our setup.

To do this, run the apigee admin API script and provide the name of the organization that needs to span the two regions, the name of the second region, and the name of the gateway pod that was specified in the configuration file for the second region.

The script is executed on the management server in region one.

Finally, we need to add the new message processors to each organization and environment.

To do this, use the apigee admin API script and provide the organization name, the environment name, and the UUID of the new message processor.

The script must be executed for each message processor in region two.

Add region

Global Load Balancer

- When two or more regions are present on the planet, a mechanism to route traffic to each data center or region must be implemented.
- Most customers use a Global Load Balancer strategy with Active/Active, Active/Passive, Geo-location, or other routing rules.
- With replication, Apigee Edge data is available across all regions on the planet.



At this point, the Apigee Edge planet has been expanded from one to two regions.

API Proxies that are deployed in either region will be seamlessly deployed in all regions.

To take advantage of this capability, a mechanism is required to route traffic to each data center or region.

This is commonly implemented using a global load balancer strategy with active/active, active/passive, geo-location, or other routing rules.

Note that, regardless of the type of routing rules implemented, Apigee Edge data is available across all the regions on the planet.

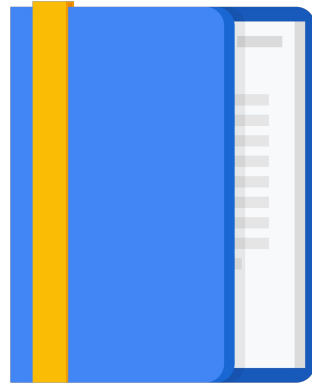
Agenda

Capacity planning

Provisioning

Expansion

Lab



We will now complete a lab to expand capacity in Apigee for private cloud.

Lab

Expanding Capacity in Apigee for Private Cloud



In this lab, you will expand the API processing capacity of your Apigee Edge private cloud installation by adding a message processor component to the cluster, and associate the message processor to an Apigee environment.

Lab

Expanding Capacity in Apigee for
Private Cloud



Lab review

Expanding Capacity in Apigee for Private Cloud



In this lab, you added a message processor to increase the API processing capacity in your cluster.

You verified the status of the message processor component after adding it to the cluster.

You also removed the component from the cluster. When you no longer need the extra capacity, you can remove components to conserve resources.



Review: Capacity Planning and Scaling

Hansel Miranda
Technical Curriculum Developer, Google



In this module, we discussed how to plan for capacity and manage changes to your infrastructure requirements for the Apigee Edge private cloud platform.

We discussed capacity planning and how to provision and expand capacity on the platform.

You completed a lab to add and remove API processing capacity for Apigee private cloud on Google Compute Engine.

