Apigee™

Apigee Edge for Private Cloud

v4.16.05

May 31, 2015

Operations Guide

THE CONTENTS OF THIS PUBLICATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT OF INTELLECTUAL PROPERTY.

APIGEE CORPORATION SHALL NOT UNDER ANY CIRCUMSTANCES BE LIABLE TO ANY PERSON FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES, INCLUDING WITHOUT LIMITATION, DAMAGES RESULTING FROM THE USE OF OR RELIANCE ON THE INFORMATION IN THIS PUBLICATION, LOSS OF PROFITS, REVENUE OR DATA, EVEN IF APIGEE CORPORATION HAS BEEN PREVIOUSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Contact Information

INDIA

No.17/2, 2B Cross, 7th Main, 2 & 3
Floor, Off 80 Feet Road, 3rd Block
Koramangala, Bangalore 560034
Call +91 80 67696800
www.apigee.com

USA

10 Almaden Boulevard,
16th Floor, San Jose
CA 95113
Call +1 (408) 343-7300
www.apigee.com

UK

3 Sheldon Square
London W2 6HY
Call: +44 (0) 750 123 2390
www.apigee.com/

Contents

Overview

This document provides instructions on how to perform key operational tasks for the Apigee Edge for Private Cloud.

The information is arranged according to the specific operational area being covered, with sub-sections for each key Apigee subsystem – Edge Services and Analytics.

This version of this document has details specific to **version 4.16.01**. Any references that are specific to previous versions are oversights and should be reported as bugs.

Access the Apigee Community for your questions

The Apigee Community is a free resource where you can contact Apigee as well as other Apigee customers with questions, tips, and other issues. Before posting to the community, be sure to first search existing posts to see if your question has already been answered.

Apigee Edge Architecture Overview

The following diagram illustrates the Apigee Edge architecture and how Edge components interact.

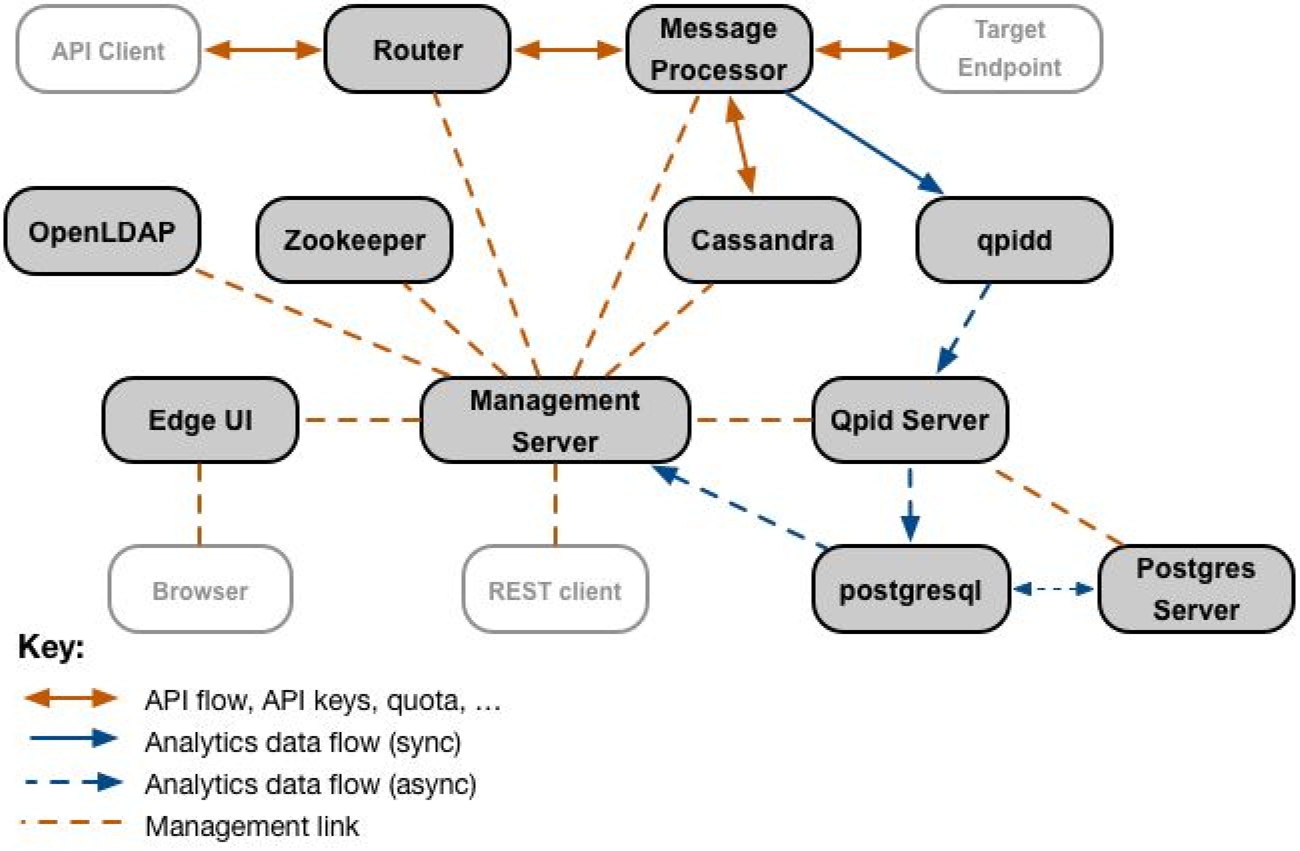


Figure 1: Conceptual Component Interactions

Component Descriptions

Components required for API processing

The following components must be up and running to process API requests:

- The **Apigee Router** receives requests from a load balancer, determines which organization and environment the request is to be routed to, and sends the request to a Message Processor configured to handle requests for that organization and environment.
- The **Apigee Message Processor** processes API requests. The Message Processor evaluates an incoming request, executes any Apigee policies, and calls the back-end systems and other systems to retrieve data. Once those responses have been received, the Message Processor formats a response and returns it to the initial client.

- **Apache Cassandra** is the runtime data repository for Apigee API processing. It is an eventually consistent, distributed database.

Components not required for API processing but for system services

The following components are not required for processing API requests, but are required to start the required Router, Message Processor, and Cassandra components:

- **Apache Zookeeper** is a configuration database, which holds information about the location and configuration of the various Apigee components. It is an instantly consistent, distributed database.

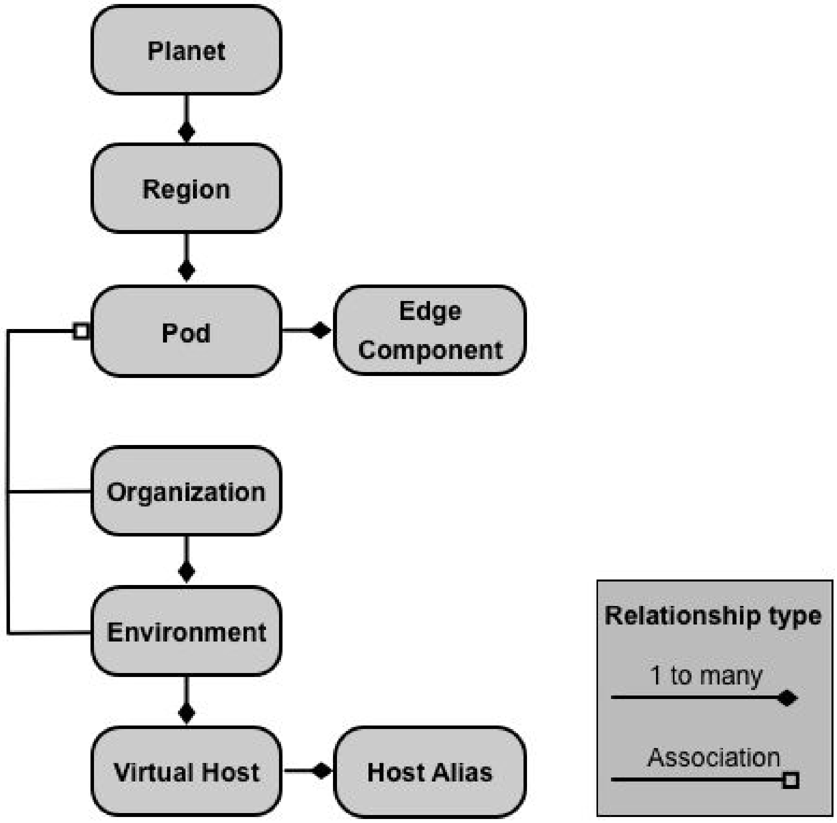
Components not required for API processing

The following components are not required for processing API requests but are necessary to enable additional Edge functionality:

- **Apache Qpid** is a queuing system that accepts analytics data from the Message Processors and writes the data to the PostgreSQL database.
- The **Apigee Postgres Server** manages the PostgresSQL analytics database.
- The **Apigee Management Server** is the endpoint for Management API requests. The Management Server also interacts with the Enterprise User Interface to display Analytics reporting requests.
- The **Apigee Edge User Interface** is the management UI for the Apigee system. This UI can be used to define and deploy APIs, manage developer and application definitions, and manage Apigee system users.
- The LDAP repository (**OpenLDAP**) contains Apigee UI user, role, and permission definitions.

About Planets, Regions, Pods, Organizations, Environments and Virtual Hosts

An on-premises installation of Edge Private Cloud, or an Edge *instance*, consists of multiple Edge components installed on a set of server nodes. The following image shows the relationship among the planet, regions, pods, organizations, environments, and virtual hosts that make up an Edge instance:



The following table describes these relationships:

	Contains	Associated with	Default
Planet	One or more regions		n/a
Region	One or more pods		"dc-1"
Pod	One or more Edge components		"central" "gateway" "analytics "

Organization	One or more environments	One or more pods containing Message Processors, and a user acting as the org admin	none
Environment	One or more virtual hosts	One or more Message Processors in a pod associated with the parent organization	none
Virtual Host	One or more host aliases		none

About Planets

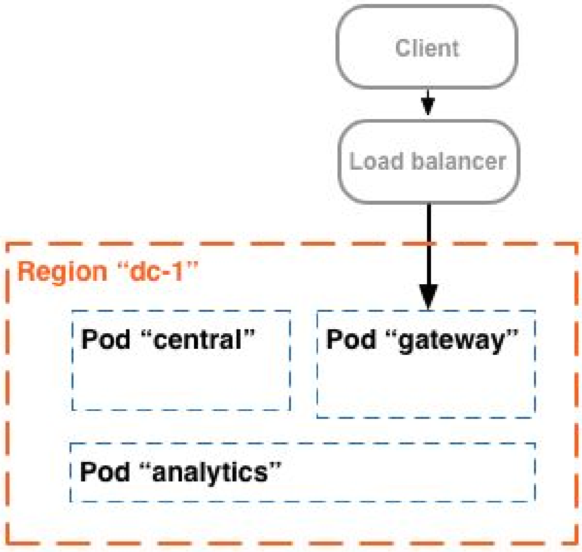
A *planet* represents an entire Edge hardware and software environment and can contain one or more regions. In Edge, a planet is a logical grouping of regions — you do not explicitly create or configure a planet as part of installing Edge.

About Regions

A *region* is a grouping of one or more pods. By default, when you install Edge, the installer creates a single region named "dc-1" containing three pods, as the following table shows:

Region	Pods in the region
"dc-1"	"gateway", "central", "analytics"

The following image shows the default regions:

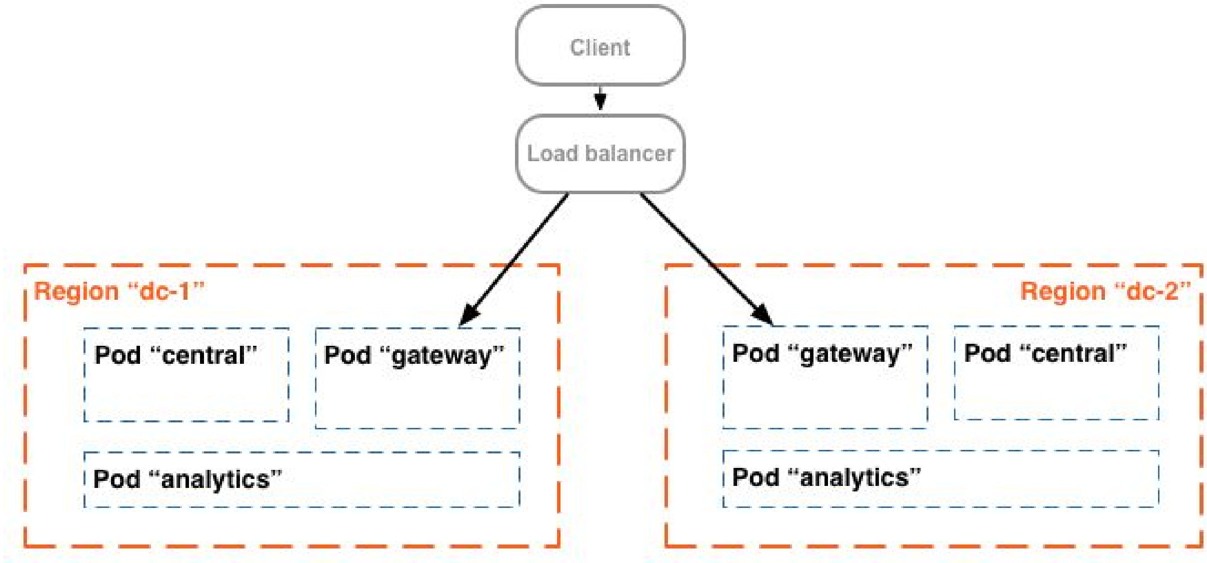


This image shows the load balancer directing traffic to the "gateway" pod. The "gateway" pod contains the Edge Router and Message Processor components that handle API requests. Unless you define multiple data centers, you should not have to create additional regions.

In a more complex installation, you can create two or more regions. One reason to create multiple regions is to organize machines geographically, which minimizes network transit time. In this scenario, you host API endpoints so that they are geographically “close” to the consumers of those APIs.

In Edge, each region is referred to as a *data center*. A data center in the Eastern US can then handle requests arriving from Boston, Massachusetts, while a data center in Singapore can handle requests originating from devices or computers in Asia.

For example, the following image shows two regions, corresponding to two data centers:



About Pods

A *pod* is a grouping of one or more Edge components and Cassandra datastores. The Edge components can be installed on the same node, but are more commonly installed on different nodes. A Cassandra datastore is a data repository used by the Edge components in the pod.

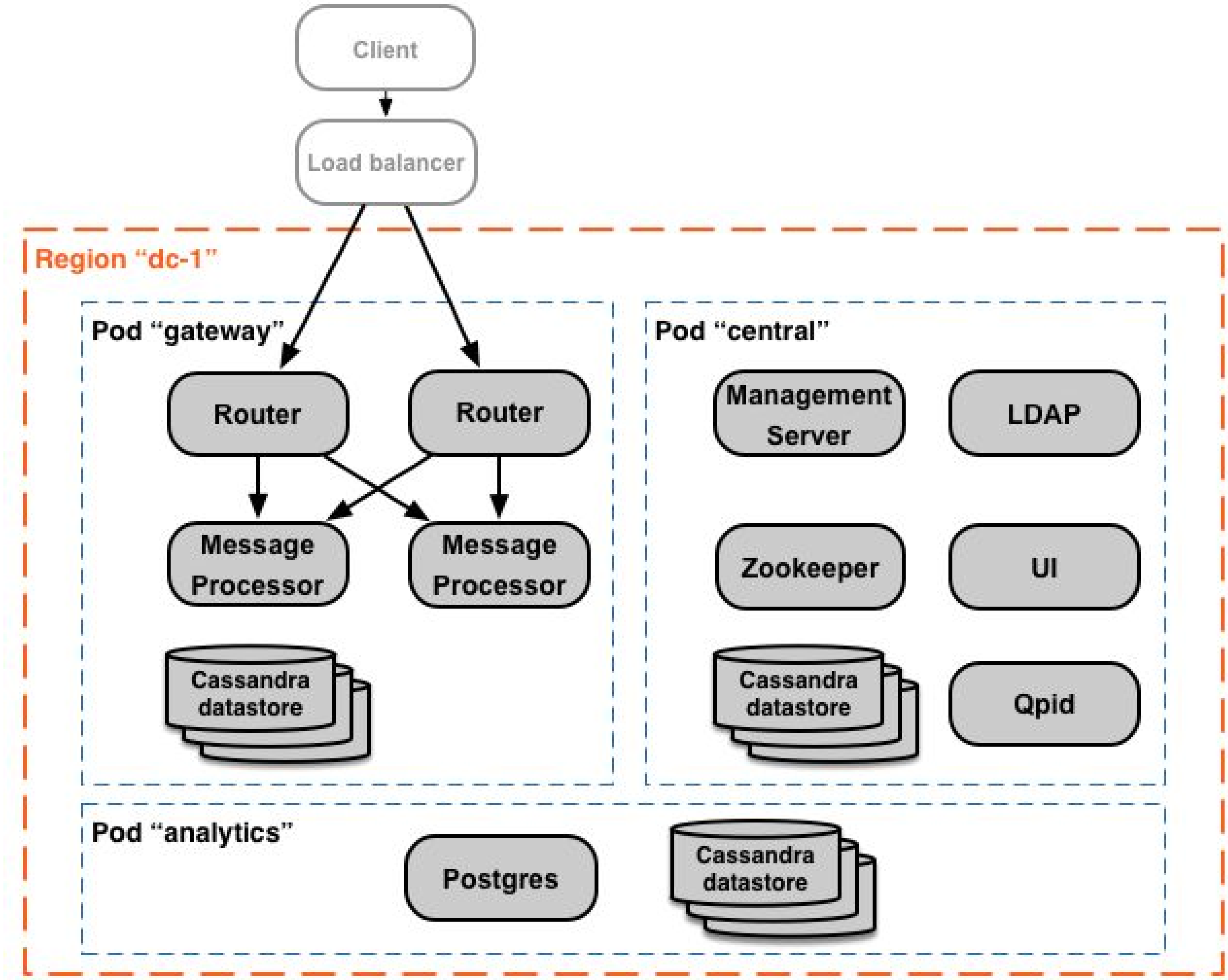
By default, when you install Edge, the installer creates three pods and associates the following Edge components and Cassandra datastores with each pod:

Pod	Edge components	Cassandra datastores	
"gateway"	Router, Message Processor	cache-datastore counter-datastore dc-datastore	keyvaluemap-datastore kms-datastore

"central"	Management Server, Zookeeper, LDAP, UI	application-datastore apimodel-datastore audit-datastore auth-datastore management-server user-settings-datastore
"analytics"	Postgres, Qpid	analytics-datastore reportcrud-datastore

The Edge components and Cassandra datastores in the "gateway" pod are required for API processing. These components and datastores must be up and running to process API requests. The components and datastores in the "central" and "analytics" pods are not required to process APIs, but add additional functionality to Edge.

The following image shows the components in each pod:



You can add additional pods for Message Processors and Routers to the three that are created by default. You can also add additional Edge components to an existing pod. For example, you can add additional Routers and Message Processors to the "gateway" pod to handle increased traffic loads.

Notice that the "gateway" pod contains the Edge Router and Message Processor components. Routers only send requests to Message Processors in the same pod and not to Message Processors in other pods.

You can use the following API call to view server registration details at the end of the installation for each pod. This is a useful monitoring tool.

```
curl -u adminEmail:pword http://<ms_IP>:8080/v1/servers?pod=podName
```

where `ms_IP` is the IP address or DNS name of the Management Server, and ***podName*** is either:

- `gateway`
- `central`
- `analytics`

For example, for the "gateway" pod:

```
> curl -u adminEmail:pword http://<ms_IP>:8080/v1/servers?pod=gateway
```

You see output in the form:

```
[ {
  "externalHostName" : "localhost",
  "externalIP" : "192.168.1.11",
  "internalHostName" : "localhost",
  "internalIP" : "192.168.1.11",
  "isUp" : true,
  "pod" : "gateway",
  "reachable" : true,
  "region" : "dc-1",
  "tags" : {
    "property" : [ {
      "name" : "jmx.rmi.port",
      "value" : "1101"
    }, ... ]
  },
  "type" : [ "message-processor" ],
  "uUID" : "276bc250-7dd0-46a5-a583-fd11eba786f8"
},
{
  "internalIP" : "192.168.1.11",
  "isUp" : true,
  "pod" : "gateway",
  "reachable" : true,
  "region" : "dc-1",
  "tags" : {
    "property" : [ ]
  },
  "type" : [ "dc-datastore", "management-server", "cache-datastore",
"keyvaluemap-datastore", "counter-datastore", "kms-datastore" ],
  "uUID" : "13cee956-d3a7-4577-8f0f-1694564179e4"
},
{
  "externalHostName" : "localhost",
  "externalIP" : "192.168.1.11",
  "internalHostName" : "localhost",
  "internalIP" : "192.168.1.11",
  "isUp" : true,
  "pod" : "gateway",
  "reachable" : true,
  "region" : "dc-1",
  "tags" : {
    "property" : [ {
      "name" : "jmx.rmi.port",
      "value" : "1100"
    }, ... ]
  },
  "type" : [ "router" ],
  "uUID" : "de8a0200-e405-43a3-a5f9-eabafdd990e2"
} ]
```

The **type** attribute lists the component type. Note that it lists the Cassandra datastores registered in the pod. While Cassandra nodes are installed in the "gateway" pod, you will see Cassandra datastores registered with all pods.

About Organizations

An *organization* is a container for all the objects in an Apigee account, including APIs, API products, apps, and developers. An organization is associated with one or more pods, where each pod must contain one or more Message Processors.

Note: In the default installation, where you only have a single "gateway" pod that contains all the Message Processors, you only associate an org with the "gateway" pod.

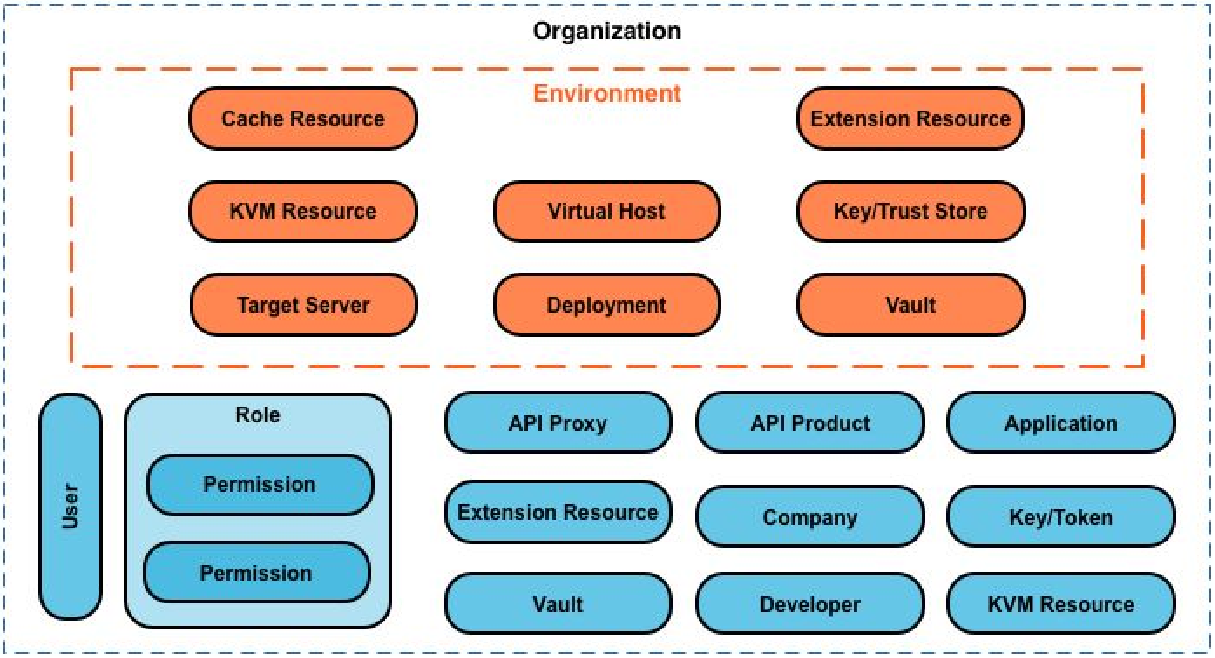
In an on-premises installation of Edge Private Cloud, there are no organizations by default. When you create an organization, you specify two pieces of information:

1. A user who functions as the organization administrator. That user can then add additional users to the organization, and set the role of each user.
2. The "gateway" pod, the pod containing the Message Processors.

An organization can contain one or more environments. The default Edge installation procedure prompts you to create two environments: "test" and "prod". However, you can create more environments as necessary, such as "staging", "experiments", etc.

Organization provides scope for some Apigee capabilities. For example, key-value-map (KVM) data is available at the organization level, meaning from all environments. Other capabilities, such as caching, are scoped to a specific environment. Apigee analytics data is partitioned by a combination of organization and environment.

Shown below are the major objects of an organization, including those defined globally in the organization, and those defined specifically to an environment:



About Environments

An *environment* is a runtime execution context for the API proxies in an organization. You must deploy an API proxy to an environment before it can be accessed. You can deploy an API proxy to a single environment or to multiple environments.

An organization can contain multiple environments. For example, you might define a "dev", "test", and "prod" environment in an organization.

When you create an environment, you associate it with one or more Message Processors. You can think of an environment as a named set of Message Processors on which API proxies run. Every environment can be associated with the same Message Processors, or with different ones.

To create an environment, specify two pieces of information:

1. The organization containing the environment.
2. The Message Processors that handle API proxy requests to the environment. These Message Processors must be in a pod associated with the environment's parent organization.

   By default, when you create an environment, Edge associates all available Message Processors in the "gateway" pod with the environment. Alternatively, you can specify a subset of the available Message Processors so that different Message Processors handle requests to different environments.

A Message Processor can be associated with multiple environments. For example, your Edge installation contains two Message Processors: A and B. You then create three environments in your organization: "dev", "test", and "prod":

- For the "dev" environment, you associate Message Processor A because you do not expect a large volume of traffic.
- For the "test" environment, you associate Message Processor B because you do not expect a large volume of traffic.
- For the "prod" environment, you associate both Message Processors A and B to handle the production-level volume.

The Message Processors assigned to an environment can all be from the same pod, or can be from multiple pods, spanning multiple regions and data centers. For example, you define the environment "global" in your organization that includes Message Processors from three regions, meaning three different data centers: US, Japan, and Germany.

Deploying an API proxy to the "global" environment causes the API proxy to run on Message Processors in all of three data centers. API traffic arriving at a Router in any one of those data centers would be directed only to Message Processors in that data center because Routers only direct traffic to Message Processors in the same pod.

About Virtual Hosts

A *virtual host* defines the port on the Edge Router on which an API proxy is exposed, and, by extension, the URL that apps use to access the API proxy. Every environment must define at least one virtual host.

Ensure that the port number specified by the virtual host is open on the Router node. You can then access an API proxy by making a request to:

```
http://<routerIP>:<port>/{proxy-base-path}/{resource-name}
```

```
https://<routerIP>:<port>/{proxy-base-path}/{resource-name}
```

where:

- `http` or `https`: If the virtual host is configured to supports SSL, use HTTPS. If the virtual host does not support SSL, use HTTP.
- `<routerIP>:<port>` is the IP address and port number of the virtual host.
- `{proxy-base-path}` and `{resource-name}` are defined when you create the API proxy.

Typically, you do not publish your APIs to customers with an IP address and port number. Instead, you define a DNS entry for the Router and port. For example:

```
http://myAPI.myCo.com/{proxy-base-path}/{resource-name}
```

```
https://myAPI.myCo.com/{proxy-base-path}/{resource-name}
```

You also must create a *host alias* for the virtual host that matches the domain name of the DNS entry. From the example above, you would specify a host alias of `myAPI.myCo.com`. If you do not have a DNS entry, set the host alias to the IP address of the Router and port of the virtual host, as `<routerIP>:port`.

For more, see http://apigee.com/docs/api-services/content/virtual-hosts.

Creating your first organization, environment, and virtual host

After you complete the Edge installation process, your first action is typically to create an organization, environment, and virtual host through the "onboarding" process. To perform onboarding, run the following command on the Edge Management Server node:

```
/<inst_root>/apigee/apigee-service/bin/apigee-service apigee-provision
setup-org -f configFile
```

This command takes as input a config file that defines a user, organization, environment, and virtual host.

For example, you create:

- A user of your choosing to function as the organization administrator
- An organization named `example`
- An environment in the organization named `prod` that is associated with all Message Processors in the "gateway" pod

- A virtual host in the environment named `default` that allows HTTP access on port 9001
- Host alias for the virtual host

After running that script, you can access your APIs by using a URL in the form:

```
http://<router-ip>:9001/{proxy-base-path}/{resource-name}
```

You can later add any number of organizations, environments, and virtual hosts.

For more information on onboarding, see http://docs.apigee.com/api-services/content/onboard-organization.

Apigee Installation File System

As part of the installation process, several directories are created which contain programs, scripts and tools that you may find useful in your day-to-day operation of the system.

Installation directory

By default, the installer writes all files to the `/<inst_root>/apigee` directory.

In the instructions in this guide, the installation directory is noted as /`<inst_root>/apigee`, where /`<inst_root>` is `/opt`.

File System Structure

Under the `/<inst_root>/apigee` directory are a number of directories, including one directory for each Edge component. Each directory is prefixed by:

- `apigee` - a third-party component used by Edge. For example, `apigee-cassandra`.
- `edge` - an Edge component from Apigee. For example, `edge-management-server`.
- `edge-mint` - a Monetization component. For example `edge-mint-management-server`.
- `baas` - an API BaaS component. For example `baas-usergrid`.

Additional directories under the `/<inst_root>/apigee` directory include:

- `customer/application` - To configure Edge components after installation, you edit .properties files in the `customer/application` directory. Changes to .properties files require you to restart the affected Edge component.

  Each component has its own .properties file in that directory. For example, `router.properties` and `management-server.properties`. The .properties files are not installed by default. Typically, you create them only when setting properties on a component.
- `data` - Data associated with each component. If you create a local mirror of the Edge repo, it is stored in the `data` directory.
- `var/log` - Log files for each component. For example, `var/log/edge-management-server` contains the log files for the Edge Management Server.

Warning: Only modify .properties files under the `customer/application` directory. Modifying any files under an Edge component's directory, under the `/<inst_root>/apigee/etc` directory, and under the `/<inst_root>/apigee/token` directory is not supported and will result in an inoperable system. Please contact Apigee Support if modification is required.

Important Data to Remember From the Installation Process

Before you can perform any maintenance tasks, you need to know a few things about your Apigee Edge environment that are set during the system installation process. Be sure that you have the following information available to your operations team:

- Gateway pod name (default "gateway")
- Central pod name (default "central")
- Analytics pod name (default "analytics")
- Region name (default "dc-1")
- Administrative user ID and password

Other important information, such as the Apigee version number or the unique identifiers (UUIDs) of an Apigee component, can be found as follows:

- Use the following command to display the status of all Edge components installed on the node:

  ```
  > /<inst_root>/apigee/apigee-service/bin/apigee-all status
  ```

- Use the following command to display the version numbers of all Edge components installed on the node:

  ```
  > /<inst_root>/apigee/apigee-service/bin/apigee-all version
  ```

- Use the following API call to display the UUID of install Edge components:

  ```
  > curl http://<ms_IP>:<port_num>/v1/servers/self -u $ADMINEMAIL:$PW
  ```

  where ***ms_IP*** is the IP address or DNS name of the Management server node, and ***port_num*** is:

  - 8081 for the Router
  - 8082 for the Message Processor
  - 8083 for Qpid
  - 8084 for Postgres

How to Configure Edge

To configure Edge after installation, you use a combination of `.properties` files and Edge utilities. For example, to configure SSL on the Edge UI, you edit `.properties` files to set the necessary properties. Changes to `.properties` files require you to restart the affected Edge component.

Apigee refers to the technique of editing `.properties` files as *code with config*. Essentially, *code with config* is a key/value lookup tool based on settings in the `.properties` files. In code with config, the keys are referred to as *tokens*. Therefore, to configure Edge, you set *tokens* in `.properties` files.

Code with config allows Edge components to set default values that are shipped with the product, lets the installation team override those settings based on the installation topology, and then lets customers override any properties they choose.

If you think of it as a hierarchy, then the settings are arranges as follows with customer settings having the highest priority to override any settings from the installer team or Apigee:

1. Customer
2. Installer
3. Apigee

How to use .properties files

As a customer, you can only modify the `.properties` files in the `/<inst_root>/apigee/customer/application` directory. Each component has its own `.properties` file in that directory. For example, `router.properties` and `management-server.properties`.

Note: If you have not set any properties for a component, the `/<inst_root>/apigee/customer/application` directory might not contain a `.properties` file for the component. In that case, create one. The only requirement on a `.properties` file is that it must be accessible or readable by the "apigee" user. For example, to create a `.properties` file:

1. Create the file in an editor as any user.
2. Chown the owner of the file to `apigee:apigee` or, if you changed the user running the Edge service from the `apigee` user, chown the file to the user who is running the Edge service.

To set a property for a component, edit the corresponding `.properties` file to set a token, and then restart the component:

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

For example, after editing `router.properties`, restart the Router:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router restart
```

When you upgrade Edge, the `.properties` files in the `/<inst_root>/apigee/customer/application` directory are read. That means the upgrade will retain any properties that you set on the component.

Location of .properties files

There are three top-level directories for `.properties` files for Edge components: installation owner, and customers. The default locations are shown in the following table:

Warning: Edge customers can only modify the `.properties` files under `<inst_root>/apigee/customer`. While you can view files under the component and installation directories, do not modify any files.

Owner	Default Token Root Directory
Component	`<inst_root>/apigee/<`***`prefix`***`>-<`***`component`***`>/conf` where /<***prefix***>-<***component***> identifies the component, such as edge-router or apigee-cassandra.
Installation	`<inst_root>/apigee/token`
Customer	`<inst_root>/apigee/customer`

Determining the current value of a token

Before you set a token in the `.properties` file for the component, you can first determine its current value by using the following command:

```
> /<inst_dir>/apigee/apigee-service/bin/apigee-service comp configure -search
token
```

where ***comp*** is the name of the component, and ***token*** is the token to inspect.

This command searches the hierarchy of `.properties` files to determine the current value of the token.

For example, to check the current value of the `conf_router_HTTP.request.line.limit` token for the Router:

```
> /<inst_dir>/apigee/apigee-service/bin/apigee-service edge-router configure
-search conf_router_HTTP.request.line.limit
```

You should see output in the form:

```
Found key conf_router_HTTP.request.line.limit, with value, 4k, in
/opt/apigee/edge-router/token/default.properties
```

Setting tokens in .properties files

To override the value of a token:

1. Edit the `.properties` file for the component to set the token value. If the file does not exist, then create it.
2. Restart the component.
3. Check the token value.

For example, to set the request line limit for the Edge Router:

1. Edit the `/<inst_root>/apigee/customer/application/router.properties` file to set the `conf_router_HTTP.request.line.limit` token:

   ```
   conf_router_HTTP.request.line.limit=8k
   ```

2. Restart the Edge Router:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router
   restart
   ```

3. Check the value of the token:

   ```
   > /<inst_dir>/apigee/apigee-service/bin/apigee-service edge-router
   configure -search conf_router_HTTP.request.line.limit
   ```

Locating a token

In most cases, the tokens you need to set are identified in this guide. However, if you need to determine the token used to override a property, perform a `grep` in the `source` folder of the component.

For example, if you know that in a previous release of Edge you set the `session.maxAge` property and want to know the token value used to set it, then `grep` for the property in the `/<inst_root>/apigee/edge-ui/source` directory:

```
> grep -ri "session.maxAge" /<inst_root>/apigee/edge-ui/source
```

You should see a result in the form:

```
/<inst_root>/apigee/edge-ui/source/conf/application.conf:session.maxAge={T}conf_
application_session.maxage{/T}
```

The string between the **`{T}{/T}`** tags is the token that you set in the `.properties` file.

Setting a token that is currently commented out

Some tokens are commented out in the Edge configuration files. If you try to set a token that is commented out, the setting is ignored.

To set a token that is commented out, you use a special syntax, in the form:

```
conf/file.extension+propertyName=propertyValue
```

For example, you want to set the property named `HTTPClient.proxy.host` on the Message Processor. You then `grep` for the property to determine its token:

```
> cd /opt/apigee/edge-message-processor
```

```
> grep -ri "HTTPClient.proxy.host" *
```

The `grep` command returns results in the form that includes the token value. Notice how the property name is commented out, as indicated by the **#** prefix on the property name:

```
source/conf/http.properties:#HTTPClient.proxy.host={T}conf_http_HTTPClient.proxy
.host{/T}
```

```
token/default.properties:conf_http_HTTPClient.proxy.host=
```

```
conf/http.properties:#HTTPClient.proxy.host=
```

To set the property, edit `/opt/apigee/customer/application/message-processor.properties` to set the property as:

```
conf/http.properties+HTTPClient.proxy.host=myhost.name.com
```

Notice how the property name is prefixed by `conf/http.properties+`, the location and name of the configuration file containing the property followed by "+".

After you restart the Message Processor, examine the file `/opt/apigee/edge-message-processor/conf/http.properties`:

```
> cat /opt/apigee/edge-message-processor/conf/http.properties
```

At the end of the file, you will see the property set, in the form:

```
conf/http.properties:HTTPClient.proxy.host=yhost.name.com
```

Using the apigee-adminapi.sh utility

Use the `apigee-adminapi.sh` utility to perform the same Edge configuration tasks that you perform by making calls to the Edge management API. The advantage to the `apigee-adminapi.sh` utility is that it:

- Use a simple command-line interface
- Implements tab-based command completion
- Provides help and usage information
- Can display the corresponding API call if you decide to try the API

The `apigee-adminapi.sh` utility is not a replacement for the `apigee-provision` utility. The `apigee-provision` utility actually uses the `apigee-adminapi.sh` utility to perform its tasks.

The main differences between the two are:

- The `apigee-adminapi.sh` utility performs atomic functions that replace individual Edge API calls. For example, to create an organization, environment, and virtual host requires three separate `apigee-adminapi.sh` commands corresponding to three API calls.
- The `apigee-provision` utility is designed to perform a complete high-level operation in a single command. For example, you can create an organization, environment, and virtual host with a single `apigee-provision` command by passing a config file with all necessary information.

The Edge documentation uses both utilities where appropriate.

Installing apigee-adminapi.sh

The `apigee-adminapi.sh` utility is automatically installed when you install the `apigee-provision` or the `apigee-validate` utility.

The utility is installed in the following location:

```
/opt/apigee/apigee-adminapi/bin/apigee-adminapi.sh
```

apigee-adminapi.sh syntax

The `apigee-adminapi.sh` utility uses a simple command line syntax. At any time, use the tab key to display a prompt that lists the available command options.

To see all possible commands, invoke the utility with no options:

```
> apigee-adminapi.sh
```

If you press the tab key after typing `apigee-adminapi.sh`, you will see the list of possible options:

```
analytics  classification  logsessions     regions    securityprofile userroles

buildinfo  GET             orgs            runtime    servers         users
```

The tab key displays options based on the context of the command. If you enter the tab key after typing:

```
> apigee-adminapi.sh orgs
```

You will see the possible options for completing the `orgs` command:

```
add  apis  apps  delete  envs  list  pods  userroles
```

Use the `-h` option to display help for any command. For example, if you use the `-h` option as shown below:

```
> apigee-adminapi.sh orgs -h
```

The utility displays complete help information for all possible options to the `orgs` command. The first item in the output shows the help for the "`orgs add`" command:

```
+++++++++++++++++++++++++++++++++++++++++++

orgs add

  Required:

    -o ORG Organization name

  Optional:

    -H HEADER add http header in request

    --admin ADMIN_EMAIL admin email address

    --pwd ADMIN_PASSWORD admin password

    --host EDGE_SERVER edge server to make request to

    --port EDGE_PORT port to use for the http request

    --ssl set EDGE_PROTO to https, defaults to http

    --debug ( set in debug mode, turns on verbose in curl )

    -h      Displays Help
```

Setting parameters using command-line switches and environment variables

You must enter all parameters to a command by using either command-line switches, or by using environment variables. Prefix the command line switches with a single dash (-) or double dash (--) as required.

Note: If you omit the sys admin password when entering a command, the `apigee-adminapi.sh` utility will promt you for it. It will not prompt you for any other parameters.

For example, from the help show above for the the "`orgs add`" command, you can specify the organization name by either:

- Using the `-o` command line switch:

  ```
  > apigee-adminapi.sh orgs -o testOrg
  ```

- Setting an environment variable named `ORG`:

  ```
  > export ORG=testOrg
  > apigee-adminapi.sh orgs
  ```

Note: You typically use environment variables to set `ADMIN_EMAIL` and `EDGE_SERVER`, and optionally `ADMIN_PASSWORD`. These parameters are used by most commands. The concern with setting other params, such as `ORG`, by using environment variables is that the setting might be correct for one command but could be incorrect for a subsequent command. For example, if you forget to reset the environment variable, you might inadvertently pass the wrong value to the next command.

If you omit any required parameters to the command, the utility displays an error message describing the missing parameters. For example, if you omit the `--host` or `EDGE_SERVER` environment variable specifying the Edge Management Server when creating an org, you see the following error message:

```
Error with required variable or parameter
ADMIN_PASSWORD....OK
ADMIN_EMAIL....OK
EDGE_SERVER....null
```

Two common parameters that you often set as environment variables are the sys admin email address and IP address of the Management Server:

```
> export ADMIN_EMAIL=foo@bar.com
> export EDGE_SERVER=192.168.56.101
```

Passing a file to the apigee-adminapi.sh utility

Some `apigee-adminapi.sh` utility commands correspond to PUT and POST API calls that take a request body. For example, creating a virtual host corresponds to a POST API call that requires information about the virtual host in the request body.

When using the `apigee-adminapi.sh` utility to create a virtual host, or any command that takes a request body, you can pass all of the necessary information on the command line as shown below:

```
> apigee-adminapi.sh orgs envs virtual_hosts add -e prod -o testOrg --host
localhost --admin foo@bar.com -v myVHostUtil -p 9005 -a 192.168.56.101:9005
```

Or, you can pass a file containing the same information as would be contained in the request body of the POST. For example, the following command takes a file defining the virtual host:

```
> apigee-adminapi.sh orgs envs virtual_hosts add -e prod -o testOrg --host
localhost --admin foo@bar.com -f vhostcreate
```

where the file `vhostcreate` contains the POST body of the call. In this example, it is a XML-formatted request body:

```
<VirtualHost name="myVHostUtil">
   <HostAliases>
     <HostAlias>192.168.56.101:9005</HostAlias>
```

```
   </HostAliases>
   <Interfaces/>
   <Port>9005</Port>
</VirtualHost>
```

Displaying debug and API information

Use the `--debug` option to the `apigee-adminapi.sh` utility to display detailed information about the command. This infomration includes the cURL command generated by the `apigee-adminapi.sh` utility to perform the operation.

For example, this command uses the `--debug` option:

```
> apigee-adminapi.sh orgs add -o testOrg2 --admin foo@bar.com --host localhost
--debug
```

And displays the following output, including the generated cURL command:

```
curl -H Content-Type: application/xml -v -X POST     -s -k -w \n==> %{http_code}
-u ***oo@bar.com:*****     http://localhost:8080/v1/o -d <Organization
name="testOrg2" type="paid"/>
```

```
* About to connect() to localhost port 8080 (#0)
*   Trying ::1... connected
* Connected to localhost (::1) port 8080 (#0)
* Server auth using Basic with user 'foo@bar.com'
> POST /v1/o HTTP/1.1
> Authorization: Basic c2dp234234NvbkBhcGlnZ2342342342342341Q5
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.19.1
Basic ECC zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: localhost:8080
> Accept: */*
> Content-Type: application/xml
> Content-Length: 43
>
} [data not shown]
< HTTP/1.1 201 Created
< Content-Type: application/json
< Date: Tue, 03 May 2016 02:08:32 GMT
< Content-Length: 291
<

{ [data not shown]
* Connection #0 to host localhost left intact
* Closing connection #0
```

Scaling Edge for Private Cloud

Scaling an instance of Edge for Private Cloud refers to the process of adding either additional processing nodes, or adding an entire data center, or region, to the installation.

Adding a Router or Message Processor node

You can add a Router or Message Processor node to an existing installation. For a list of the system requirements for a Router or Message Processor, see http://docs.apigee.com/api-services/latest/installation-requirements.

Add a Router

After you install Edge on the node, use the following procedure to add the Router:

1. Install Edge on the node using the internet or non internet procedure as described in the *Edge Installation* manual.
2. At the command prompt, run the `apigee-setup.sh` script:

   ```
   > /<inst_root>/apigee/apigee-setup/bin/setup.sh -p r -f configFile
   ```

   The “-p r” option specifies to install the Router. See http://docs.apigee.com/api-services/latest/install-edge-components-node for information on creating a ***configFile***.
3. When the installation completes, the script displays the UUID of the Router.

   If you need to determine the UUID later, use the following cURL command on the host where you installed the Router:

   ```
   > curl http://<router_IP>:8081/v1/servers/self
   ```
4. To check the configuration, you can run the following cURL command:

   ```
   > curl -v -u adminEmail:pword
   "http://<ms_IP>:8080/v1/servers?pod=<pod_name>"
   ```

   where `pod_name` is `gateway` or your custom pod name. You should see the UUIDs of all Routers, including the Router that you just added.

   If the Router UUID does not appear in the output, run the following cURL command to add it:

   ```
   > curl -v -u adminEmail:pword  -X POST
   "http://<ms_IP>:8080/regions/<region_name>/pods/<pod_name>/servers -d
   ```

```
"action=add&uuid={router_UUID}&type=router"
```

Replace *ms_IP* with the IP address of the Management Server, *region_name* with the default region name of dc-1 or your custom region name, and *pod_name* with gateway or your custom pod name.

5. To test the router, you should be able to make requests to your APIs through the IP address or DNS name of the Router. For example:

   ```
   http://<newRouter_IP>:<port>/v1/apiPath
   ```

   For example, if you completed the first tutorial where you created the weather API:

   ```
   http://<newRouter_IP>:<port>/v1/weather/forecastrss?w=12797282
   ```

Add a Message Processor

After you install Edge on the node, use the following procedure to add a Message Processor:

1. Install Edge on the node using the internet or non internet procedure as described in the *Edge Installation* manual.
2. At the command prompt, run the apigee-setup.sh script:

   ```
   > /<inst_root>/apigee/apigee-setup/bin/setup.sh -p mp -f configFile
   ```

   The “-p mp” option specifies to install the Message Processor. See http://docs.apigee.com/api-services/latest/install-edge-components-node for information on creating a ***configFile***.
3. When the installation completes, the script displays the UUID of the Message Processor. Note that UUID as you need it to complete the configuration process.

   If you need to determine the UUID, use the following cURL command on the host where you installed the Message Processor:

   ```
   > curl http://<mp_IP>:8082/v1/servers/self
   ```
4. For each environment in each organization in your installation, use the following cURL command to associate the Message Processor with the environment:

   ```
   > curl -v -u adminEmail:pword
   -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}"
   ```

Replace *ms_IP* with the IP address of the Management Server and *org_name* and *env_name* with the organization and environment associated with the Message Processor.

5. To check the configuration, you can run the following cURL command:

```
> curl -v -u adminEmail:pword
"http://<ms_IP>:8080/v1/o/{org_name}/e/{env_name}/servers"
```

   where *org_name* is the name of your organization, and *env_name* is the environment. You should see the UUIDs of all Message Processors associated with the organization and environment, including the Message Processor that you just added.

Add both a Router and a Message Processor

After you install Edge on the node, us the following procedure to add a router and Message Processor at the same time:

1. At the command prompt, run the `apigee-setup` script:

```
> /<inst_root>/apigee/apigee-setup/bin/setup.sh -p rmp -f configFile
```

   The “-p rmp” option specifies to install the Router and Message Processor. See thttp://docs.apigee.com/api-services/latest/install-edge-components-node for information on creating a ***configFile***.

2. Follow the procedures above to configure the Router and Message Processor.

Adding Cassandra nodes

This document describes how to add three new Cassandra nodes to an existing Edge for Private Cloud installation.

While you can add one or two Cassandra nodes to an existing Edge installation, Apigee recommends that you add three nodes at a time.

Existing Edge configuration

All the supported Edge topologies for a production system specify to use three Cassandra nodes. The three nodes are specified to the **CASS_HOSTS** property in the config file as shown below:

```
IP1=10.10.0.1
IP2=10.10.0.2
IP3=10.10.0.3
HOSTIP=$(hostname -i)
ADMIN_EMAIL=opdk@apigee.com
APIGEE_ADMINPW=Secret123
```

```
LICENSE_FILE=/tmp/license.txt
MSIP=$IP1
USE_LDAP_REMOTE_HOST=n
LDAP_TYPE=1
APIGEE_LDAPPW=secret
BIND_ON_ALL_INTERFACES=y
MP_POD=gateway
REGION=dc-1
ZK_HOSTS="$IP1 $IP2 $IP3"
ZK_CLIENT_HOSTS="$IP1 $IP2 $IP3"
CASS_HOSTS="$IP1:1,1 $IP2:1,1 $IP3:1,1"
SKIP_SMTP=n
SMTPHOST=smtp.example.com
SMTPUSER=smtp@example.com
SMTPPASSWORD=smtppwd
```

Note that the `REGION` property specifies the region name as "dc-1". You need that information when adding the new Cassandra nodes.

Modifying the config file to add the three new Cassandra nodes

In this example, the three new Cassandra nodes are at the following IP addresses:

- 10.10.0.14
- 10.10.0.15
- 10.10.0.16

You must first update Edge configuration file to add the new nodes:

```
IP1=10.10.0.1
IP2=10.10.0.2
IP3=10.10.0.3
# Add the new node IP addresses.
IP14=10.10.0.14
IP15=10.10.0.15
IP16=10.10.0.16
HOSTIP=$(hostname -i)
ADMIN_EMAIL=opdk@apigee.com
...
# Update CASS_HOSTS to add each new node after an existing nodes.
CASS_HOSTS="$IP1:1,1 $IP14:1,1 $IP2:1,1 $IP15:1,1 $IP3:1,1 $IP16:1,1"
```

Important: Add each new Cassandra node to `CASS_HOSTS` **after an existing node.**

This ensure that the existing nodes retain their initial token settings, and the initial token of each new node is between the token values of the existing nodes.

Configure Edge

After editing the config file, you must:

- Reconfigure the existing Cassandra nodes
- Install Cassandra on the new nodes
- Reconfigure the Management Server

Reconfigure the existing Cassandra nodes

On the existing Cassandra nodes:

1. Rerun the `setup.sh` with the "-p c" profile and the new config file:

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

Install Cassandra on the new nodes

On each new Cassandra node:

1. Install Cassandra on the three nodes:
   a. Install `apigee-setup` on the first node as described in the *Edge Installation Guide*.
   b. Install Cassandra on the first node by using the updated config file:

      ```
      > /opt/apigee/apigee-setup/bin/setup.sh -p c -f updatedConfigFile
      ```
   c. Repeat steps a and b for the remaining new Cassandra nodes.
2. Rebuild the three new Cassandra nodes, specifying the region name set in the config file by the `REGION` property. In this example, it is "dc-1":
   a. On the first node, run:

      ```
      > /opt/apigee/apigee-cassandra/bin/nodetool -h nodeIP rebuild dc-1
      ```

      where ***nodeIP*** is the IP address of the Cassandra node.
   b. Repeat step a on the remaining new Cassandra nodes.

Reconfigure the Management Server

On a Management-Server node

1. Update the `apigee-cassandra-client` component on the Management Server node:

   ```
   > /opt/apigee/apigee-service/bin/apigee-service apigee-cassandra-client
   update
   ```
2. Rerun `setup.sh` to update the Management Server for the newly added Cassandra nodes:

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

Adding ZooKeeper nodes

This document describes how to add three new ZooKeeper nodes to an existing Edge for Private Cloud installation.

You can add one or two ZooKeeper nodes to an existing Edge installation, however, you must make sure that you always have an odd number of ZooKeeper voter nodes, as described below..

Existing Edge configuration

All the supported Edge topologies for a production system specify to use three ZooKeeper nodes. The three nodes are specified to the **ZK_HOSTS** and **ZK_CLIENT_HOSTS** properties in the config file as shown below:

```
IP1=10.10.0.1
IP2=10.10.0.2
IP3=10.10.0.3
HOSTIP=$(hostname -i)
ADMIN_EMAIL=opdk@apigee.com
APIGEE_ADMINPW=Secret123
LICENSE_FILE=/tmp/license.txt
MSIP=$IP1
USE_LDAP_REMOTE_HOST=n
LDAP_TYPE=1
APIGEE_LDAPPW=secret
BIND_ON_ALL_INTERFACES=y
MP_POD=gateway
REGION=dc-1
ZK_HOSTS="$IP1 $IP2 $IP3"
ZK_CLIENT_HOSTS="$IP1 $IP2 $IP3"
CASS_HOSTS="$IP1:1,1 $IP2:1,1 $IP3:1,1"
SKIP_SMTP=n
SMTPHOST=smtp.example.com
SMTPUSER=smtp@example.com
SMTPPASSWORD=smtppwd
```

where:

- **ZK_HOSTS** specifies the IP addresses or DNS names of the ZooKeeper nodes. The IP addresses or DNS names must be listed in the same order on all ZooKeeper nodes. In a multi-data center environment, list all ZooKeeper nodes from both data centers.

- **ZK_CLIENT_HOSTS** specifies the IP addresses or DNS names of the ZooKeeper nodes used by this data center. The IP addresses or DNS names must be listed in the same order on all ZooKeeper nodes.

In a single data center installation, these are the same nodes as specified by **ZK_HOSTS**. In a multi-data center environment, list only the ZooKeeper nodes in this data center.

Modifying the config file to add the three new ZooKeeper nodes

In this example, the three new ZooKeeper nodes are at the following IP addresses:

- 10.10.0.14
- 10.10.0.15
- 10.10.0.16

You must first update Edge configuration file to add the new nodes:

```
IP1=10.10.0.1
IP2=10.10.0.2
IP3=10.10.0.3
# Add the new node IP addresses.
IP14=10.10.0.14
IP15=10.10.0.15
IP16=10.10.0.16
HOSTIP=$(hostname -i)
ADMIN_EMAIL=opdk@apigee.com
...
# Update ZK_HOSTS to add each new node after an existing nodes.
ZK_HOSTS="$IP1 $IP2 $IP3 $IP14 $IP15 $IP16:observer"
# Update ZK_Client_HOSTS to add each new node after an existing nodes.
ZK_CLIENT_HOSTS="$IP1 $IP2 $IP3 $IP14 $IP15 $IP16"
```

Mark the last node in **ZK_HOSTS** with the with “:observer” modifier. Nodes without the “:observer” modifier are called "voters". You must have an odd number of "voters" in your configuration. Therefore, in this configuration, you have 5 ZooKeeper voters and one observer.

Note: While you can configure three observers and three voters, Apigee recommends that you use five voters.

Make sure to add the nodes to both **ZK_HOSTS** and **ZK_CLIENT_HOSTS** in the same order. However, omit the “:observer” modifier when setting **ZK_CLIENT_HOSTS**.

Configure Edge

After editing the config file, you must perform all of the following tasks.

Install ZooKeeper on the new nodes

Install Zookeeper on the new nodes:

1. Install `apigee-setup` on the first node as described in the *Edge Installation Guide*.
2. Install ZooKeeper on the first node by using the following commands:

   ```
   > /opt/apigee/apigee-service/bin/apigee-service apigee-zookeeper install
   > /opt/apigee/apigee-service/bin/apigee-service apigee-zookeeper setup -f
   updatedConfigFile
   ```
3. Repeat steps 1 and 2 for the remaining new ZooKeeper nodes.

Reconfigure the existing ZooKeeper nodes

On the existing ZooKeeper nodes:

1. Rerun the `setup.sh` with the "-p c" profile and the new config file:

   ```
   > /opt/apigee/apigee-service/bin/apigee-service apigee-zookeeper setup -f
   updatedConfigFile
   ```

Restart all Zookeeper nodes

On all ZooKeeper nodes:

1. Restart the node:

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

   You must restart all ZooKeeper nodes, but order of restart does not matter.

Reconfigure the Management Server node

On the Management Server node:

1. Run the setup command:

   ```
   > /opt/apigee/apigee-service/bin/apigee-service edge-management-server
   setup -f updatedConfigFile
   ```
2. Restart the Management Server:

   ```
   > /opt/apigee/apigee-service/bin/apigee-service edge-management-server
   restart
   ```

Reconfigure all the Routers

On all Router nodes:

1. Run the setup command:

```
> /opt/apigee/apigee-service/bin/apigee-service edge-router setup -f
updatedConfigFile
```

2. Restart the Router:

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

Reconfigure all the Message Processors

On all Message Processor nodes:

1. Run the setup command:

```
> /opt/apigee/apigee-service/bin/apigee-service edge-message-processor
setup -f updatedConfigFile
```

2. Restart the Message Processor:

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

Reconfigure all Qpid nodes

On all Qpid nodes:

1. Run the setup command:

```
> /opt/apigee/apigee-service/bin/apigee-service edge-qpid-server setup -f
updatedConfigFile
```

2. Restart Qpid:

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

Reconfigure all Postgres nodes

On all Postgres nodes:

1. Run the setup command:

```
> /opt/apigee/apigee-service/bin/apigee-service edge-postgres-server setup
-f updatedConfigFile
```

2. Restart Postgres:

```
> /opt/apigee/apigee-service/bin/apigee-service edge-postgres-server
restart
```

Validate the installation

You can validate the installation of the new ZooKeeper nodes by sending commands to port 2181 using netcat (nc) or telnet. For more info on ZooKeeper commands, see: http://zookeeper.apache.org/doc/r3.1.2/zookeeperAdmin.html#sc_zkCommands

To validate:

1. If it is not installed on the ZooKeeper node, install nc:

   ```
   > sudo yum install nc
   ```
2. Run the following nc command:

   ```
   > echo stat | nc localhost 2181
   ```
3. Repeat steps 1 and 2 on each ZooKeeper node.

   In the **Mode** line of the output for the nodes, one node should be designated as observer, one node as leader, and the rest as followers.

Adding a data center

You can add a data center (also called a *region*) to an existing data center.

Considerations before adding a data center

Before you install add a data center, you must understand how to configure OpenLDAP, ZooKeeper, Cassandra, and Postgres servers across the data centers. You must also ensure that the necessary ports are open between the nodes in the two data centers.

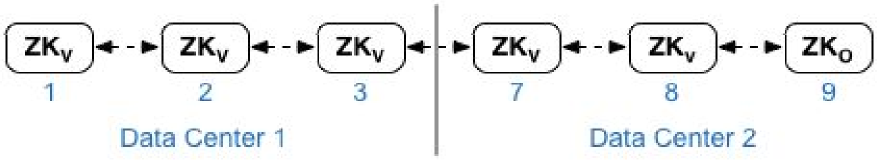
- **OpenLDAP**

  Each data center has its own OpenLDAP server configured with replication enabled. When you install the new data center, you must configure OpenLDAP to use replication, and you must reconfigure the OpenLDAP server in the existing data center to use replication.

- **ZooKeeper**

  For the `ZK_HOSTS` property for both data centers, specify the IP addresses or DNS names of all ZooKeeper nodes from both data centers, in the same order, and mark any nodes with the with ":observer" modifier. Nodes without the ":observer" modifier are called "voters". You must have an odd number of "voters" in your configuration.

  In this topology, the ZooKeeper host on host 9 is the observer:



  In the example configuration file shown below, node 9 is tagged with the ":observer" modifier so that you have five voters: Nodes 1, 2, 3, 7, and 8.

For the `ZK_CLIENT_HOSTS` property for each data center, specify the IP addresses or DNS names of only the ZooKeeper nodes in the data center, in the same order, for all ZooKeeper nodes in the data center.

- **Cassandra**

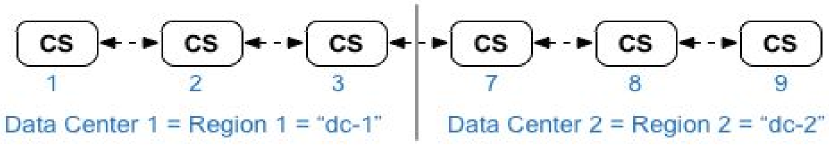
  All data centers must to have the same number of Cassandra nodes.

  For `CASS_HOSTS` for each data center, ensure that you specify all Cassandra IP addresses or DNS names for both data centers. For data center 1, list the Cassandra nodes in that data center first. For data center 2, list the Cassandra nodes in that data center first. List the Cassandra nodes in the same order for all Cassandra nodes in the data center.

  All Cassandra nodes must have a suffix ':<d>,<r>', for example '<ip>:1,1 = data center 1 and rack/availability zone 1 and '<ip>:2,1 = data center 2 and rack/availability zone 1.
  For example, "192.168.124.201:1,1 192.168.124.202:1,1 192.168.124.203:1,1 192.168.124.204:2,1 192.168.124.205:2,1 192.168.124.206:2,1"

  The first node in rack/availability zone 1 of each data center will be used as the seed server.

  In this deployment model, Cassandra setup will look like this:


CS
1
CS
2
CS
3
CS
7
CS
8
CS
9
Data Center 1 = Region 1 = "dc-1"
Data Center 2 = Region 2 = "dc-2"

- **Postgres**

  By default, Edge installs all Postgres nodes in master mode. However, when you have multiple data centers, you configure Postgres nodes to use master-standby replication so that if the master node fails, the standby node can continue to server traffic. Typically, you configure the master Postgres server in one data center, and the standby server in the second data center.

  If the existing data center is already configured to have two Postgres nodes running in master/standby mode, then as part of this procedure, deregister the existing standby node and replace it with a standby node in the new data center.

  The following table shows the before and after Postgres configuration for both scenarios:

Before	**After**
Single Master Postgres node in **dc-1**	Master Postgres node in **dc-1** Standby Postgres node in **dc-2**
Master Postgres node in **dc-1** Standby Postgres node in **dc-1**	Master Postgres node in **dc-1** Standby Postgres node in **dc-2** Deregister old Standby Postgres node in **dc-1**

- **Port requirements**

  You must ensure that the necessary ports are open between the nodes in the two data centers. For a port diagram, see the *Edge Installation Guide*.

Updating the existing data center

Adding a data center requires you to perform the steps to install and configure the new data center nodes, but it also requires you to update nodes in the original data center. These modifications are necessary because you are adding new Cassandra and ZooKeeper nodes in the new data center that have to be accessible to the existing data center, and you have to reconfigure OpenLDAP to use replication.

Creating the configuration files

Shown below are the silent configuration files for the two data centers, where each data center has 6 nodes. Notice that the config file for dc-1 adds additional settings to:

- Configure OpenLDAP with replication across two OpenLDAP nodes.
- Add the new Cassandra and ZooKeeper nodes from dc-2 to the config file for dc-1.

```
# Datacenter 1
IP1=IPorDNSnameOfNode1
IP2=IPorDNSnameOfNode2
IP3=IPorDNSnameOfNode3
IP7=IPorDNSnameOfNode7
IP8=IPorDNSnameOfNode8
IP9=IPorDNSnameOfNode9
HOSTIP=$(hostname -i)
MSIP=$IP1
ADMIN_EMAIL=opdk@apigee.com
APIGEE_ADMINPW=Secret123
LICENSE_FILE=/tmp/license.txt
USE_LDAP_REMOTE_HOST=n
LDAP_TYPE=2
LDAP_SID=1
LDAP_PEER=$IP7
APIGEE_LDAPPW=secret
BIND_ON_ALL_INTERFACES=y
MP_POD=gateway-1
REGION=dc-1
ZK_HOSTS="$IP1 $IP2 $IP3 $IP7 $IP8
$IP9:observer"
ZK_CLIENT_HOSTS="$IP1 $IP2 $IP3"
CASS_HOSTS="$IP1:1,1 $IP2:1,1 $IP3:1,1
$IP7:2,1 $IP8:2,1 $IP9:2,1"
```

```
# Datacenter 2
IP1=IPorDNSnameOfNode1
IP2=IPorDNSnameOfNode2
IP3=IPorDNSnameOfNode3
IP7=IPorDNSnameOfNode7
IP8=IPorDNSnameOfNode8
IP9=IPorDNSnameOfNode9
HOSTIP=$(hostname -i)
MSIP=$IP7
ADMIN_EMAIL=opdk@apigee.com
APIGEE_ADMINPW=Secret123
LICENSE_FILE=/tmp/license.txt
USE_LDAP_REMOTE_HOST=n
LDAP_TYPE=2
LDAP_SID=2
LDAP_PEER=$IP1
APIGEE_LDAPPW=secret
BIND_ON_ALL_INTERFACES=y
MP_POD=gateway-2
REGION=dc-2
ZK_HOSTS="$IP1 $IP2 $IP3 $IP7 $IP8
$IP9:observer"
ZK_CLIENT_HOSTS="$IP7 $IP8 $IP9"
CASS_HOSTS="$IP7:2,1 $IP8:2,1 $IP9:2,1
$IP1:1,1 $IP2:1,1 $IP3:1,1"
```

SKIP_SMTP=n
SMTPHOST=smtp.example.com
SMTPUSER=smtp@example.com
SMTPPASSWORD=smtppwd
SMTPSSL=n
SMTPPORT=25
	SKIP_SMTP=n
	SMTPHOST=smtp.example.com
	SMTPUSER=smtp@example.com
	SMTPPASSWORD=smtppwd
	SMTPSSL=n
	SMTPPORT=25

Procedure to add a new data center

In this procedure, the data centers are named:

- **dc-1**: the existing data center
- **dc-2**: the new data center

1. **On dc-2**, install `apigee-setup` on all nodes. See the *Edge Installation Guide* for more info.
2. **On dc-2**, install Cassandra and ZooKeeper on the appropriate nodes:

   ```
   /opt/apigee/apigee-setup/bin/setup.sh -p ds -f configFile2
   ```

3. **On dc-1**, rerun `setup.sh` on the original Cassandra nodes with the new dc-1 confiig file that includes the Cassandra nodes from dc-2:

   ```
   /opt/apigee/apigee-setup/bin/setup.sh -p ds -f configFile1
   ```

4. **On dc-1**, rerun `setup.sh` on the Management Server node:

   ```
   > /opt/apigee/apigee-setup/bin/setup.sh -p ms -f configFile1
   ```

5. **On dc-2**, run the rebuild command on all Cassandra nodes, specifying the region name of dc-1:

   ```
   > /opt/apigee/apigee-cassandra/bin/nodetool -h cassIP rebuild dc-1
   ```

6. **On dc-2**, install the Management Server on the appropriate node:

   ```
   > /opt/apigee/apigee-setup/bin/setup.sh -p ms -f configFile2
   ```

7. **On dc-2**, install the Routes and Message Processors on the appropriate nodes:

   ```
   > /opt/apigee/apigee-setup/bin/setup.sh -p rmp -f configFile2
   ```

8. **On dc-2**, install Qpid on the appropriate nodes:

```
> /opt/apigee/apigee-setup/bin/setup.sh -p qs -f configFile2
```

9. **On dc-2**, install Postgres on the appropriate node:

   ```
   > /opt/apigee/apigee-setup/bin/setup.sh -p ps -f configFile2
   ```

10. Setup Postgres master/standby for the Postgres nodes. The Postgres node in **dc-1** is the master, and the Postgres node in **dc-2** is the standby server.

    Note: If **dc-1** is already configured to have two Postgres nodes running in master/standby mode, then as part of this procedure, use the **existing master Postgres node in dc-1** as the master, and the **Postgres node in dc-2** as the standby server. Later in this procedure, you will deregister the existing Postgres standby server in **dc-1**.

    a. On the master node in **dc-1**, edit the config file to set:

       ```
       PG_MASTER=IPorDNSofNewMaster
       PG_STANDBY=IPorDNSofOldMaster
       ```

    b. Enable replication on the new master:

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

    c. On the standby node in **dc-2**, edit the config file to set:

       ```
       PG_MASTER=IPorDNSofNewMaster
       PG_STANDBY=IPorDNSofOldMaster
       ```

    d. On the standby node in **dc-2**, stop the server and then delete any existing Postgres data:

       ```
       > /opt/apigee/apigee-service/bin/apigee-service apigee-postgresql
       stop
       > rm -rf /opt/apigee/data/apigee-postgresql/
       ```

       Note: If necessary, you can backup this data before deleting it.

    e. Configure the standby node in **dc-2**:

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

11. **On dc-1**, update analytics configuration and configure the organizations.

    a. **On the Management Server node of dc-1,** get the UUID of the Postgres node:

       ```
       > apigee-adminapi.sh servers list -r dc-1 -p analytics -t
       ```

```
postgres-server --admin adminEmail --pwd adminPword --host localhost
```

The UUID appears at the end of the returned data. Save that value.

Note: If **dc-1** is configured to have two Postgres nodes running in master/standby mode, you see two IP addresses and UUIDs in the output. Save both UUIDs. From the IPs, you should be able to determine which UUID is for the master and which is for the standby node.

b. **On the Management Server node of dc-2,** get the UUID of the Postgres node as shown in the previous step. Save that value.

c. **On the Management Server node of dc-1,** determine the name of the analytics and consumer groups. Many of the commands below require that information.

By default, the name of the analytics group is `axgroup-001`, and the name of the consumer group is `consumer-group-001`. In the silent config file for a region, you can set the name of the analytics group by using the `AXGROUP` property.

If you are unsure of the names of the analytics and consumer groups, use the following command to display them:

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

This command returns the analytics group name in the **name** field, and the consumer group name in the **consumer-groups** field.

d. **On the Management Server node of dc-1,** remove the existing Postgres server from the analytics group:

   i. Remove the Postgres node from the consumer-group:

```
> apigee-adminapi.sh analytics groups consumer_groups
datastores remove -g axgroup-001 -c consumer-group-001 -u UUID
-Y --admin adminEmail --pwd adminPword --host localhost
```

   If **dc-1** is configured to have two Postgres nodes running in master/standby mode, remove both:

```
> apigee-adminapi.sh analytics groups consumer_groups
datastores remove -g axgroup-001 -c consumer-group-001 -u
UUID_1,UUID_2 -Y --admin adminEmail --pwd adminPword --host
localhost
```

ii. Remove the Postgres node from the analytics group:

```
> apigee-adminapi.sh analytics groups postgres_server remove -g
axgroup-001 -u UUID -Y --admin adminEmail --pwd adminPword
--host localhost
```

If **dc-1** is configured to have two Postgres nodes running in master/standby mode, remove both:

```
> apigee-adminapi.sh analytics groups consumer_groups
datastores remove -g axgroup-001 -c consumer-group-001 -u
UUID1,UUID2 -Y --admin adminEmail --pwd adminPword --host
localhost
```

e. **On the Management Server node of dc-1,** add the new master/standby Postgres servers to the analytics group:

i. Add both Postgres servers to the analytics group:

```
> apigee-adminapi.sh analytics groups postgres_server add -g
axgroup-001 -u "UUID_1,UUID_2" --admin adminEmail --pwd
adminPword --host localhost
```

where ***UUID_1*** corresponds to the master Postgres node in **dc-1**, and ***UUID_2*** corresponds to the standby Postgres node in **dc-2.**

ii. Add the PG servers to the consumer-group as master/standby:

```
> apigee-adminapi.sh analytics groups consumer_groups
datastores add -g axgroup-001 -c consumer-group-001 -u
"UUID_1,UUID_2" --admin adminEmail --pwd adminPword --host
localhost
```

f. Add the Qpid servers from **dc-2** to the analytics group:

i. **On the MS node of dc-1,** get the UUIDs of the Qpid nodes in **dc-2**:

```
> apigee-adminapi.sh servers list -r dc-2 -p central -t
qpid-server --admin adminEmail --pwd adminPword --host localhost
```

The UUIDs appear at the end of the returned data. Save those values.

ii. **On the Management Server node of dc-1,** add the Qpid nodes to the analytics group:

```
 > apigee-adminapi.sh analytics groups qpid_server add -g
axgroup001 -u UUID_1,UUID_2 --admin adminEmail --pwd adminPword
```

```
--host localhost
```

iii. **On the Management Server node of dc-1,** add the Qpid nodes to the consumer group:

```
> apigee-adminapi.sh analytics groups consumer_groups consumers
add -g axgroup-001 -c consumer-group-001 -u UUID_1,UUID_2
--admin adminEmail --pwd adminPword --host localhost
```

g. Deregister and delete the old Postgres standby server from **dc-1**:

i. Deregister the existing **dc-1** Postgres standby server:

```
> apigee-adminapi.sh servers deregister -u UUID -r dc-1 -p
analytics -t postgres-server -Y --admin adminEmail --pwd
adminPword --host localhost
```

where ***UUID*** is the old standby Postgres node in **dc-1**.

ii. Delete the existing dc-1 Postgres standby server:

Note: This command does not uninstall the Postgres server node. It only removes it from the list of Edge nodes. You can later uninstall Postgres from the node, if necessary.

```
> apigee-adminapi.sh servers delete -u UUID --admin adminEmail
--pwd adminPword --host localhost
```

12. For each organization and for each environment that you want to support across data centers:

a. **On the Management Server node of dc-1,** add the new MP_POD to the Organization:

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

where ***gateway-2*** is the name of the gateway pod as defined by the `MP_POD` property in the dc-2 config file.

b. Add the new Message Processors to the org and environment:

i. **On the Management Server node of dc-2,** get the UUIDs of the Message Processor nodes in **dc-2**:

```
> apigee-adminapi.sh servers list -r dc-2 -p gateway-2 -t
message-processor --admin adminEmail --pwd adminPword --host
localhost
```

The UUIDs appear at the end of the returned data. Save those values.

ii. **On the Management Server node of dc-1,** for each Message Processor in dc-2, add the Message Processor to an environment for the org:

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

c. **On the Management Server node of dc-1,** check the organization:

```
> apigee-adminapi.sh orgs apis deployments -o orgName -a apiProxyName
--admin adminEmail --pwd adminPword --host localhost
```

where ***apiProxyName*** is the name of an API proxy deployed in the organization.

Moving Apigee Servers

Apigee components use IP addresses to communicate with each other. Moving components from one machine to another may cause a configuration mismatch. To fix configuration mismatches, follow the relevant instructions below.

Changing the IP Address of a Cassandra Node

To change the IP address of a Cassandra node, perform the following steps:

For configurations with a single Cassandra node

1. Edit `/<inst_root>/apigee/customer/application/cassandra.properties` on the system being modified. If the file does not exist, create it.
2. Change the following parameters:
   - Set the `conf_cassandra_seeds` and `conf_cassandra_listen_address` parameters to specify the system's new IP address.
   - Change the `conf_cassandra_rpc_address` to use either the new IP address or 0.0.0.0 (which allows Cassandra Thrift to listen on all interfaces).
3. Open `/<inst_root>/apigee/apigee-cassandra/conf/cassandra-topology.properties` in an editor.

   You should see the old IP address and default setting in the form:

   ```
   192.168.56.101=dc-1:ra-1
   default=dc-1:ra-1
   ```

   Save that information.
4. Edit `/<inst_root>/apigee/customer/application/cassandra.properties` to change the old IP address specified to the new IP address:

   ```
   conf_cassandra-topology_topology=192.168.56.103=dc-1:ra-1\ndefault=dc-1:ra
   -1\n
   ```

   Ensure that you insert "\n" after the IP address, and specify the same default settings as you found above in Step 3.
5. Restart Cassandra:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
   restart
   ```

6. If necessary also repair ZooKeeper (see below), else restart every Apigee platform component starting with Management Server.

For configurations with multiple Cassandra nodes (ring)

1. If the node being changed is a seed node, edit `/<inst_root>/apigee/customer/application/cassandra.properties` file on each system in the ring, and change the `conf_cassandra_seeds` parameter to include the modified system's new IP. If the `cassandra.properties` file does not exist, create it.
2. Edit `/<inst_root>/apigee/customer/application/cassandra.properties` on the system being modified, and change the following parameters:
   - Set the `conf_cassandra_listen_address` to use the new IP address.
   - Set the `conf_cassandra_rpc_address` to use either the new IP address or 0.0.0.0 (which allows Cassandra Thrift to listen on all interfaces).
3. Open `/<inst_root>/apigee/apigee-cassandra/conf/cassandra-topology.properties` in an editor.

   You should see all Cassandra IP addresses and default setting in the form:

   ```
   192.168.56.101=dc-1:ra-1
   192.168.56.102=dc-1:ra-1
   192.168.56.103=dc-1:ra-1
   default=dc-1:ra-1
   ```

   Save that information.
4. Edit `/<inst_root>/apigee/customer/application/cassandra.properties` to change the old IP address specified to the new IP address:

   ```
   conf_cassandra-topology_topology=192.168.56.101=dc-1:ra-1\n192.168.56.102=
   dc-1:ra-1\n192.168.56.104=dc-1:ra-1\ndefault=dc-1:ra-1\n
   ```

   Ensure that you insert "\n" after each IP address, and use the same default settings as you recorded above in Step 3.
5. Restart Cassandra on the modified system. If the modified system is a seed node, also restart each system that used the modified seed node.

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
   restart
   ```

6. Run the `nodetool ring` command on the modified node to ensure that the ring is complete. The utility can be found at `<inst_root>/apigee/apigee-cassandra/bin`.

   ```
   > nodetool -h localhost ring
   ```

7. Run `nodetool repair` on the modified node. Note that this process may take some time, so it is highly recommended that this not be done during peak API traffic hours.

   ```
   > nodetool -h localhost repair
   ```

8. If necessary, repair ZooKeeper (see below), then restart every Apigee platform component starting with Management Server

Update datastore registrations

1. Find the UUIDs of datastore registrations specifying the old IP address by using the commands below. Take note of the "type" and "UUID" parameters:

   -
     ```
     curl -u $ADMINEMAIL:$PW  "http://$MSIP:$port/v1/servers?pod=central" |
     egrep -i '(type|internalip|uuid|region)'
     ```

   -
     ```
     curl -u $ADMINEMAIL:$PW  "http://$MSIP: $port/v1/servers?pod=gateway" |
     egrep -i '(type|internalip|uuid|region)'
     ```

   -
     ```
     curl -u $ADMINEMAIL:$PW  "http://$MSIP :$port/v1/servers?pod=analytics"
     | egrep -i '(type|internalip|uuid|region)'
     ```

2. Register the new IP addresses using one of the commands below. The command needed will depend on the type of the changed node.

   Note: The `REGION` parameter below refers to the datacenter that the cluster is in. For example, for high availability you would generally have a cluster in dc-1 (Dater Center 1) and a cluster in dc-2 (Data Center 2). This parameter is defined at installation time. The default value is dc-1.

   - For type="application-datastore":

     ```
     curl -u $ADMINEMAIL:$PW  "http://$MSIP:$port/v1/servers -d
     "Type=application-datastore&Type=audit-datastore&InternalIP=${NEWIP}&re
     gion=${REGION}&pod=central" -H 'content-type:
     application/x-www-form-urlencoded' -X POST
     ```

   - For type="kms-datastore":

     ```
     curl -u $ADMINEMAIL:$PW  "http://$MSIP:$port/v1/servers -d
     "Type=kms-datastore&Type=dc-datastore&Type=keyvaluemap-datastore&Type=c
     ounter-datastore&Type=cache-datastore
     &InternalIP=${NEWIP}&region=${REGION}&pod=${GATEWAY_POD}" -H
     'content-type: application/x-www-form-urlencoded' -X POST
     ```

   - For type="reportcrud-datastore":

     ```
     curl -u $ADMINEMAIL:$PW "http://$MSIP:$port/v1/servers" -d
     "Type=reportcrud-datastore&InternalIP=${NEW_IP}&region=${REGION}&pod=an
     alytics" -H 'content-type: application/x-www-form-urlencoded' -X POST
     ```

3. Delete old registrations for the UUID of the system on which the IP address was changed
   - For each of these UUIDs issue:

     ```
     curl -u $ADMINEMAIL:$PW "http://$MSIP:$port/v1/servers/${OLD_UUID}" -X
     DELETE
     ```

Changing the IP Address of a ZooKeeper Node

Follow the steps below to change the IP address of a ZooKeeper node:

Change the IP Address and restart the ZooKeeper ensemble (for multi-node ensemble configurations only)

1. Open `/<inst_root>/apigee/apigee-zookeeper/conf/zoo.cfg` in an editor.

   You should see all ZooKeeper IP addresses and default setting in the form:

   ```
   server.1=192.168.56.101:2888:3888
   server.2=192.168.56.102:2888:3888
   server.3=192.168.56.103:2888:3888
   ```

   Save that information.

2. On each ZooKeeper node, edit the file `/<inst_root>/apigee/customer/application/zookeeper.properties` file to set the `conf_zoo_quorum` property to the correct IP addresses. If the file does not exist, create it.

   ```
   conf_zoo_quorum=server.1=192.168.56.101:2888:3888\nserver.2=192.168.56.102
   :2888:3888\nserver.3=192.168.56.104:2888:3888\n
   ```

   Ensure that you insert “\n” after each IP address and that entries are in the same order on every node.

3. Find the leader of the ZooKeeper ensemble by using the following command (replace `<node>` with the IP address of the Zookeeper machine):

   ```
   echo srvr | nc <node> 2181
   ```

   The `Mode` line in the output should say "leader".

4. Restart one ZooKeeper after the other starting with the leader and ending with the node on which the IP address was changed. If more than one zookeeper node changed IP addresses it may be necessary to restart all nodes.

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-zookeeper
   restart
   ```

5. Use the `echo` command described above to verify each ZooKeeper node.

Inform the Apigee nodes of the changed configuration

1. On each Router node, edit the file `/<inst_root>/apigee/customer/application/router.properties` as follows. If the file does not exist, create it.
   - Change the `conf_zookeeper_connection.string` parameter to include the new IP address
   - Change the `conf_zookeeper_zk1.host` parameter to include the new IP address
2. On every Message Processor node, edit the file `/<inst_root>/apigee/customer/application/message-processor.properties` as follows. If the file does not exist, create it.
   - Change the `conf_zookeeper_connection.string` parameter to include the new IP address
   - Change the `conf_zookeeper_zk1.host` parameter to include the new IP address
3. On the Management Server node, edit the file `/<inst_root>/apigee/customer/application/management-server.properties` as follows. If the file does not exist, create it.
   - Change the `conf_zookeeper_connection.string` parameter to include the new IP address
   - Change the `conf_zookeeper_zk1.host` parameter to include the new IP address
4. Restart all Apigee platform component by running the following command on each node:

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-all restart
   ```

Changing the IP Address of a LDAP Server (OpenLDAP)

To change the IP address of an OpenLDAP node, do the following:

1. On the Management Server node, edit the file `/<inst_root>/apigee/customer/application/management-server.properties` file. If the file does not exist, create it.
2. In the `management-server.properties` file, set the `conf_security_ldap.server.host` parameter to the new IP address.
3. Restart the Management Server:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-management-server restart
   ```

Changing the IP Address of Other Apigee Node Types

To change the IP address of any of these node types (Router, Message Processor, Postgres Server (not postgresql) and Qpid Server (not qpidd):

1. Use the following cURL command to register the new internal and external IP address:

   ```
   curl -u $ADMINEMAIL:$PW -X PUT http://$MSIP:8080/v1/servers/<uuid> -d
   ExternalIP=<ip>

   curl -u $ADMINEMAIL:$PW -X PUT http://$MSIP:8080/v1/servers/<uuid> -d
   InternalIP=<ip>
   ```

   where `uuid` is the UUID of the node.

If you do not know the UUID of the node, you can use the following command to display it:

- Router: `curl http://<router_IP>:8081/v1/servers/self`
- Message Processor: `curl http://<mp_IP>:8082/v1/servers/self`
- Qpid: `curl http://<qp_IP>:8083/v1/servers/self`
- Postgres: `curl http://<pg_IP>:8084/v1/servers/self`

Post-Installation Tasks

This section discusses tasks or additional configuration steps which can best be accomplished after the primary site installation is complete.

Resetting Passwords

You can reset the OpenLDAP, Apigee Edge system administrator, Edge organization user, and Cassandra passwords after the installation is complete.

Reset OpenLDAP Password

Depending on your Edge configuration, OpenLDAP can be installed as:

- A single instance of OpenLDAP installed on the Management Server node. For example, in a 2-node, 5-node, or 9-node Edge configuration.
- Multiple OpenLDAP instances installed on Management Server nodes, configured with OpenLDAP replication. For example, in a 12-node Edge configuration.
- Multiple OpenLDAP instances installed on their own nodes, configured with OpenLDAP replication. For example, in a 13-node Edge configuration.

The way you reset the OpenLDAP password depends on your configuration.

For a single instance of OpenLDAP installed on the Management Server, perform the following:

1) On the Management Server node, run the following command to create the new OpenLDAP password:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-openldap
change-ldap-password -o oldPword -n newPword
```

2) Run the following command to store the new password for access by the Management Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-management-server store_ldap_credentials -p newPword
```

This command restarts the Management Server.

In an **OpenLDAP replication setup with OpenLDAP installed on Management Server nodes**, follow the above steps on both Management Server nodes to update the password.

Note: You must set *identical new password* on both LDAP servers in order to achieve successful OpenLDAP replication.

In an **OpenLDAP replication setup with OpenLDAP being on a node other than Management Server**, ensure that you first change the password on both OpenLDAP nodes, then on both Management Server nodes.

Reset System Admin Password

Resetting the system admin password requires you to reset the password in two places:

- Management Server
- UI

Note: You cannot reset the system admin password from the Edge UI. You must use the procedure below to reset it.

Warning: You should stop the Edge UI before resetting the system admin password. Because you reset the password first on the Management Server, there can be a short period of time when the UI is still using the old password. If the UI makes more than three calls using the old password, the OpenLDAP server locks out the system admin account for three minutes.

To reset the system admin password:

1) On the UI node, stop the Edge UI:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui stop
```

2) On the Management Server, run the following command to reset the password:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-management-server change_sysadmin_password -o currentPW -n newPW
```

3) Edit the silent config file that you used to install the Edge UI to set the following properties:

```
APIGEE_ADMINPW=newPW
SMTPHOST=smtp.gmail.com
SMTPPORT=465
SMTPUSER=foo@gmail.com
SMTPPASSWORD=bar
SMTPSSL=y
```

Note that you have to include the SMTP properties when passing the new password because all properties on the UI are reset.

4) Use the `apigee-setup` utility to reset the password on the Edge UI from the config file:

```
> /<inst_root>/apigee/apigee-setup/bin/setup.sh -p ui -f configFile
```

5) Start the Edge UI on the UI node:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui start
```

In an **OpenLDAP replication** environment with multiple Management Servers, resetting the password on one Management Server updates the other Management Server automatically. However, you have to update all Edge UI nodes separately.

Reset Organization User Password

To reset the password for an organization user, use the `apigee-servce` utility to invoke `apigee-setup`:

```
/<inst_root>/apigee/apigee-service/bin/apigee-service apigee-setup
reset_user_password
 [-h]
 [-u USER_EMAIL]
 [-p USER_PWD]
 [-a ADMIN_EMAIL]
 [-P APIGEE_ADMINPW]
 [-f configFile]
```

For example:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-setup
reset_user_password -u user@myCo.com -p foo12345 -a admin@myCo.com -P
adminPword
```

Note: If you omit the admin email, then it uses the default admin email set at installation time. If you omit the admin password, then the script prompts for it. All other parameters must be set on the command line, or in a config file.

Shown below is an example config file that you can use with the "-f" option:

```
USER_NAME= user@myCo.com
USER_PWD= "foo12345"
APIGEE_ADMINPW= adminPword
```

You can also use the following curl command to invoke the Edge API to change the user password:

```
curl -u <adminEmail>:<passwd> -X POST \
http://<management-ip>:8080/v1/users/{userEmail} \
-d '<User>
  <Password>{newPW}</Password>
  <FirstName>{firstName}</FirstName>
  <LastName>{lastName}</LastName>
  <EmailId>{userEmail}</EmailId>
</User>' \
-H "Content-Type: application/xml"
```

where *`userEmail`* is the email address of the organization user.

Sys Admin and Organization User Password Rules

Use this section to enforce a desired level of password length and strength for your API management users. The settings use a series of preconfigured (and uniquely numbered) regular expressions to check password content (such as uppercase, lowercase, numbers, and special characters). Write these settings to `/<inst_root>/apigee/customer/application/management-server.properties` file. If that file does not exist, create it.

After editing `management-server.properties`, restart the management server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-management-server
restart
```

You can then set password strength ratings by grouping different combinations of regular expressions. For example, you can determine that a password with at least one uppercase and one lowercase letter gets a strength rating of "3", but that a password with at least one lowercase letter and one number gets a stronger rating of "4".

Table 5: Password Validation - management-server.properties

Properties	Description

conf_security_password.validation.minimum.password.length=8
conf_security_password.validation.default.rating=2
conf_security_password.validation.minimum.rating.required=3
	Use these to determine the overall characteristics of valid passwords. The default minimum rating for password strength (described later in the table) is 3.
	Notice that the `password.validation.default.rating=2` is lower than the minimum rating required, which means that if a password entered falls outside of the rules you configure, the password is rated a 2 and is therefore invalid (below the minimum rating of 3).

Following are regular expressions that identify password characteristics. Note that each one is numbered. For example, "password.validation.regex.**5**=…" is expression number 5. You'll use these numbers in a later section of the file to set different combinations that determine overall password strength.	
conf_security_password.validation.regex.1=^(.)\\1+$	1 – All characters repeat
conf_security_password.validation.regex.2=^.*[a-z]+.*$	2 – At least one lowercase letter

<table>
<tr><td>conf_security_password.validation.regex.3=^.*[A-Z]+.*$</td><td>3 – At least one uppercase letter</td></tr>
<tr><td>conf_security_password.validation.regex.4=^.*[0-9]+.*$</td><td>4 – At least one digit</td></tr>
<tr><td>conf_security_password.validation.regex.5=^.*[^a-zA-z0-9]+.*$</td><td>5 – At least one special character (not including underscore _)</td></tr>
<tr><td>conf_security_password.validation.regex.6=^.*[_]+.*$</td><td>6 – At least one underscore</td></tr>
<tr><td>conf_security_password.validation.regex.7=^.*[a-z]{2,}.*$</td><td>7 – More than one lowercase letter</td></tr>
<tr><td>conf_security_password.validation.regex.8=^.*[A-Z]{2,}.*$</td><td>8 – More than one uppercase letter</td></tr>
<tr><td>conf_security_password.validation.regex.9=^.*[0-9]{2,}.*$</td><td>9 – More than one digit</td></tr>
<tr><td>conf_security_password.validation.regex.10=^.*[^a-zA-z0-9]{2,}.*$</td><td>10 – More than one special character (not including underscore)</td></tr>
<tr><td>conf_security_password.validation.regex.11=^.*[_]{2,}.*$</td><td>11 – More than one underscore</td></tr>
<tr><td colspan="2">The following rules determine password strength based on password content. Each rule includes one or more regular expressions from the previous section and assigns a numeric strength to it. The numeric strength of a password is compared to the <code>conf_security_password.validation.minimum.rating.required</code> number at the top of this file to determine whether or not a password is valid.</td></tr>
<tr><td>conf_security_password.validation.rule.1=1,AND,0
conf_security_password.validation.rule.2=2,3,4,AND,4
conf_security_password.validation.rule.3=2,9,AND,4
conf_security_password.validation.rule.4=3,9,AND,4
conf_security_password.validation.rule.5=5,6,OR,4
conf_security_password.validation.rule.6=3,2,AND,3
conf_security_password.validation.rule.7=2,9,AND,3
conf_security_password.validation.rule.8=3,9,AND,3</td><td>Each rule is numbered. For example, "password.validation.rule.<b>3</b>=…" is rule number 3.
Each rule uses the following format (right of the equals sign):
<regex-index-list>,<AND|OR>,<rating>
<b>regex-index-list</b> is the list of regular expressions (by number from the previous section), along with an <b>AND|OR</b> operator (meaning, consider all or any of the expressions listed).</td></tr>
</table>

	rating is the numeric strength rating given to each rule. For example, rule 5 means that any password with at least one special character OR one underscore gets a strength rating of 4. With password.validation.minimum. rating.required=3 at the top of the file, a password with a 4 rating is valid.
conf_security_rbac.password.validation.enabled=true	Set role-based access control password validation to false when single sign-on (SSO) is enabled. Default is true.

Resetting Cassandra password

By default, Cassandra ships with authentication disabled. If you enable authentication, it uses a predefined user named '`cassandra`' with a password of '`cassandra`'. You can use this account, set a different password for this account, or create a new Cassandra user. Add, remove, and modify users by using the Cassandra CREATE/ALTER/DROP USER statements.

For information on how to enable Cassandra authentication, see http://docs.apigee.com/api-services/latest/enable-cassandra-authentication.

To reset the Cassandra password, you have to:

- Set the password on each Cassandra node
- Update the Edge configuration files on each node with the new password

For more information, see http://www.datastax.com/documentation/cql/3.0/cql/cql_reference/cqlCommandsTOC.html.

To reset the Cassandra password:

1. Log in to the first Cassandra node.

2. Log into Cassandra using the `cqlsh` tool and the default credentials:

```
> /<inst_root>/apigee/apigee-cassandra/bin/cqlsh cassIP cassPort -u
cassandra -p cassandra
```

Where:

- `cassIP` is the IP address of the Cassandra node.
- `cassPort` is the Cassandra port, which by default is 9160.
- The default user is `cassandra`.

- The default password is `cassandra`. If you changed the password previously, use the current password.

3. Run the following command as the `cqlsh>` prompt to update the password:

   ```
   cqlsh> ALTER USER cassandra WITH PASSWORD 'NEW_PASSWORD';
   ```

4. Exit the `cqlsh` tool:

   ```
   cqlsh> exit
   ```

5. Repeat on all Cassandra nodes, ensuring that you use the same password on all nodes.

6. On the Management Server node, run the following command:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-management-server store_cassandra_credentials -u CASS_USERNAME -p
   CASS_PASSWORD
   ```

   Optionally, you can pass a file to the command containing the new username and password:

   ```
   > apigee-service edge-management-server store_cassandra_credentials -f
   configFile
   ```

   Where the `configFile` contains the following:

   ```
   conf_credentials_cassandra.user=CASS_USERNAME
   conf_credentials_cassandra.password=CASS_PASSWROD
   ```

   This command automatically restarts the Management Server.

7. Repeat this process on the:

   - All Message Processors
   - All Routers
   - All Qpid servers (edge-qpid-server)
   - Postgres servers (edge-postgres-server)

The Cassandra password is now changed.

Managing the Default LDAP Password Policy for API Management

The Apigee system uses OpenLDAP to authenticate users in your API management environment. OpenLDAP makes this LDAP password policy functionality available.

This section describes how to configure the delivered default LDAP password policy. Use this password policy to configure various password authentication options, such as the number of consecutive failed login attempts after which a password can no longer be used to authenticate a user to the directory.

This section also describes how to use a couple of APIs to unlock user accounts that have been locked according to attributes configured in the default password policy.

Configuring the Default LDAP Password Policy

To configure the default LDAP password policy:

1. Connect to your LDAP server using an LDAP client, such as Apache Studio or ldapmodify. By default OpenLDAP server listens on port 10389 on the OpenLDAP node.

   To connect, specify the Bind DN or user of `cn=manager,dc=apigee,dc=com` and the OpenLDAP password that you set at the time of Edge installation.
2. Use the client to navigate to the password policy attributes for:
   a. Edge users: `cn=default,ou=pwpolicies,dc=apigee,dc=com`
   b. Edge sysadmin: `cn=sysadmin,ou=pwpolicies,dc=apigee,dc=com`
3. Edit the password policy attribute values as desired.
4. Save the configuration.

Default LDAP Password Policy Attributes

Attribute	Description	Default
`pwdExpireWarning`	The maximum number of seconds before a password is due to expire that expiration warning messages will be returned to a user who is authenticating to the directory.	`604800` (Equivalent to 7 days)
`pwdFailureCountInterval`	Number of seconds after which old consecutive failed bind attempts are purged from the failure counter. In other words, this is the number of seconds after which the count of consecutive failed login attempts is reset. If `pwdFailureCountInterval` is set to `0`, only a successful authentication can reset the counter. If `pwdFailureCountInterval` is set to >0, the attribute defines a duration after which the count of	`300`

	consecutive failed login attempts is automatically reset, even if no successful authentication has occurred. We suggest that this attribute be set to the same value as the `pwdLockoutDuration` attribute.	
`pwdInHistory`	Maximum number of used, or past, passwords for a user that will be stored in the `pwdHistory` attribute. When changing her password, the user will be blocked from changing it to any of her past passwords.	`3`
`pwdLockout`	If `TRUE`, specifies to lock out a user when their password expires so that the user can no longer log in.	`False`
`pwdLockoutDuration`	Number of seconds during which a password cannot be used to authenticate the user due to too many consecutive failed login attempts. In other words, this is the length of time during which a user account will remain locked due to exceeding the number of consecutive failed login attempts set by the `pwdMaxFailure` attribute. If `pwdLockoutDuration` is set to `0`, the user account will remain locked until a system administrator unlocks it. See Unlocking a User Account. If `pwdLockoutDuration` is set to >0, the attribute defines a duration for which the user account will remain locked. When this time period has elapsed, the user account will be automatically unlocked. We suggest that this attribute be set to the same value as the `pwdFailureCountInterval` attribute.	`300`
`pwdMaxAge`	Number of seconds after which a user (non-sysadmin) password expires. A value of 0 means passwords do not expire. The default value of 2592000 corresponds to 30 days from the time the password was created.	user: 2592000 sysadmin: 0

`pwdMaxFailure`	Number of consecutive failed login attempts after which a password may not be used to authenticate a user to the directory.	3
`pwdMinLength`	Specifies the minimum number of characters required when setting a password.	8

Unlocking a User Account

A user's account may be locked due to attributes set in the password policy. A user with the `sysadmin` Apigee role assigned can use the following API call to unlock the user's account. Replace values in curly braces with actual values.

To unlock a user:

```
/v1/users/{userEmail}/status?action=unlock -X POST -u {adminEmail}:{password}
```

Note: To ensure that only system administrators can call this API, the `/users/*/status` path is included in the `conf_security_rbac.restricted.resources` property for the Management Server:

```
> cd /opt/apigee/edge-management-server
```

```
> grep -ri "conf_security_rbac.restricted.resources" *
```

The output contains the following:

```
token/default.properties:conf_security_rbac.restricted.resources=/environments,
/environments/*,/environments/*/virtualhosts,/environments/*/virtualhosts/*,/po
ds,/environments/*/servers,/rebuildindex,/users/*/status
```

Setting the Session Timeout in the Edge UI

By default, a session in the Edge UI expires one day after login. That means Edge UI users have to log in again after the session expires.

You can configure the session timeout be setting the `conf_application_session.maxAge` property in the `/<inst_root>/apigee/customer/application/ui.properties` file. To set this property:

1. Open the `ui.properties` file in an editor. If the file does not exist, create it:

   ```
   > vi /<inst_root>/apigee/customer/application/ui.properties
   ```

2. Set `conf_application_session.maxAge` as a time value and time unit. Time units can be:
   - `m`, minute, minutes
   - `h`, hour, hours

- d, day, days

  For example, set conf_application_session.maxAge as: 30m, 12h, 2d.

3. Save your changes.
4. Restart the Edge UI:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui restart
```

Enabling/Disabling Server (Message Processor/Router) Reachability

It is a good practice to set reachability to false on a server during maintenance, such as for a server restart or upgrade. When reachability is false, no traffic is directed to the server. For example, when reachability is false on a Message Processor, Routers will not direct any traffic to that Message Processor.

To upgrade a Message Processor, you can use the following procedure:

1) Set reachability to false on the Message Processor.
2) Wait a few seconds for the Message Processor to finish any current requests.
3) Upgrade the Message Processor and restart it.
4) Set reachability to true on the Message Processor.

Note: In some configurations there may be a number of Routers behind a Load Balancer (ELB). The ELB might be configured to monitor the reachability flag of the Routers, in which case setting reachability to false on a Router would allow you to restart the Router.

The following API call allows load balancers to configure a node as reachable or unreachable. Note that it does not remove the server physically in a setup:

```
> curl -u adminEmail:pWord -X POST "http://<ms_IP>:8080/v1/servers/UUID" -d
"reachable=true|false"
```

where ***UUID*** is the UUID of the Message Processor or Router, and reachable is set to either true or false.

If you need to determine the UUID of the Router, use the following cURL command:

```
> curl http://<routerIP>:8081/v1/servers/self
```

If you need to determine the UUID of the Message Processor, use the following cURL command:

```
> curl http://<mpIP>:8082/v1/servers/self
```

Enabling reachability on a node

To enable any node (Router/Message Processor):

```
> curl -u adminEmail:pWord -X POST "http://<ms_IP>:8080/v1/servers/UUID" -d
"reachable=true"
```

Disabling reachability on a node

To disable any node (Router/Message Processor):

```
> curl -u adminEmail:pWord -X POST "http://<ms_IP>:8080/v1/servers/UUID" -d
"reachable=false"
```

Checking reachability status

To get reachable status of any node (Router/Message Processor):

```
> curl -u adminEmail:pWord "http://<ms_IP>:8080/v1/servers/UUID/reachable"
```

Removing a Server (Management Server/Message Processor/Router)

Follow the steps below to remove a server from an on-premises installation of Apigee Edge for Private Cloud.

1) **(Message Processor only)** Deregister Message Processor from the organization's environments

```
curl -X POST
http://<management-ip>:8080/v1/o/<orgName>/e/<envName>/servers
-d "uuid={uuid}&region={regionName}&pod={podName}&action=remove"
```

2) Deregister server's type

```
curl http://<management-ip>:8080/v1/servers -X POST -d
"type={message-processor|router|management-server}&region={regionName}&pod
={podName}&uuid={uuid}&action=remove"
```

3) Delete the server

```
curl http://<management-ip>:8080/v1/servers/{uuid} -X DELETE
```

Setting Server Autostart

An on-premises installation of Edge Private does not restart automatically during a reboot. You can use the following commands to enable/disable autostart on any node.

For all components on the node:

- `/<inst_root>/apigee-service/bin/apigee-all enable_autostart`
- `/<inst_root>/apigee-service/bin/apigee-all disable_autostart`

For a specific component:

- `/<inst_root>/apigee-service/bin/apigee-service comp enable_autostart`
- `/<inst_root>/apigee-service/bin/apigee-service comp disable_autostart`

The script only affects the node on which you run it. If you want to configure all nodes for autostart, run the script on all nodes.

Note: If you run into issues where the OpenLDAP server does not start automatically on reboot, try disabling SELinux or setting it to permissive mode.

Note that the order of starting the components is very important:

1) First start ZooKeeper, Cassandra, LDAP (OpenLDAP)
   If ZooKeeper and Cassandra are installed as cluster, the complete cluster must be up and running before starting any other Apigee component.
2) Then, any Apigee component (Management Server, Router, UI, etc.)
   For Postgres Server first start `postgresql` and for Qpid Server first start `qpidd`.

Implications:

- For a complete restart of an Apigee Edge environment, the nodes with ZooKeeper and Cassandra need to be booted completely prior to any other node.
- If any other Apigee component is running on one or more ZooKeeper and Cassandra nodes, it is not recommended to use autostart. Instead, start the components in the order described below in Starting, Stopping, and Restarting Apigee Edge.

Troubleshooting autostart

If you configure autostart, and Edge encounters issues with starting the OpenLDAP server, you can try disabling SELinux or setting it to permissive mode on all nodes. To configure SELinux:

1. Edit the `/etc/sysconfig/selinux` file:

```
> sudo vi /etc/sysconfig/selinux
```

2. Set `SELINUX=disabled` or `SELINUX=permissive`
3. Save your edits.
4. Restart the machine and then restart Edge:

```
> /<inst_root>/apigee/bin/all-start.sh
```

Starting, Stopping, and Restarting Apigee Edge

The order of stopping and starting the subsystems is important. Start and stop scripts are provided that take care of that for you for Edge components running on the same node.

Stop order: If you install Edge on multiple nodes, then you should **stop** Edge components on those nodes in the following order:

1. Apigee UI
2. Management Server
3. Router
4. Message Processor

5. Qpid Server
6. Postgres Server
7. OpenLDAP
8. Qpidd
9. PostgreSQL database
10. Cassandra
11. ZooKeeper

Start order: If you install Edge on multiple nodes, then you should **start** Edge components on those nodes in the following order:

1. ZooKeeper
2. Cassandra
3. OpenLDAP
4. Qpidd
5. PostgreSQL database
6. Management Server
7. Router
8. Message Processor
9. Qpid Server
10. Postgres Server
11. Apigee UI

The following scripts detect the Apigee components configured to run on the system on which the script is executed, and will start or stop only those components in the correct order for that node.

- To stop Apigee Edge, Apache Cassandra, and Apache ZooKeeper:

  ```
  /<inst_root>/apigee/apigee-service/bin/apigee-all stop
  ```

- To start Apache ZooKeeper, Apache Cassandra, and Apigee Edge:

  ```
  /<inst_root>/apigee/apigee-service/bin/apigee-all start
  ```

- To check if the server is running:

  ```
  /<inst_root>/apigee/apigee-service/bin/apigee-all status
  ```

Starting, Stopping and Restarting an Individual Service

You can use the following tool to start/stop/restart an individual Apigee service on any specific server.

```
/<inst_root>/apigee/apigee-service/bin/apigee-service <service> <command>
```

where:

- `<service>` is one of the following: `edge-management-server, edge-ui, edge-router, edge-message-processor, apigee-qpid-server, apigee-postgres-server, apigee-zookeeper, apigee-cassandra, apigee-openldap, apigee-qpidd, apigee-postgresql`
- `<command>` is one of the following: `start, stop, restart`

For example, to start or stop or restart Management Server, run the following commands:

```
apigee-service edge-management-server start

apigee-service edge-management-server stop

apigee-service edge-management-server restart
```

You can also check the status of an individual Apigee service by using the following command:

```
apigee-service edge-management-server status
```

Uninstalling Edge

To uninstall a component, use the `apigee-service` utility in the form:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service compName uninstall
```

For example, to uninstall the Edge UI:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui uninstall
```

To uninstall all Apigee components on the node, uninstall the `apigee-service` utility:

```
> /opt/apigee/apigee-service/bin/apigee-service apigee-service uninstall
```

This command does not delete any data or log files. It only deletes the components.

If you want to completely remove Edge from your system:

1. Stop all Edge services running on the machine:

   ```
   > /opt/apigee/apigee-service/bin/apigee-all stop
   ```

2. Clear the yum cache:

   ```
   > sudo yum clean all
   ```

3. Remove all the Apigee RPMs:

   ```
   > sudo rpm -e $(rpm -qa | egrep "(apigee-|edge-|baas)")
   ```

4. Remove the installation root directory:

   ```
   > sudo rm -rf /opt/apigee
   ```

Viewing Pod Details

You can use the following API call to view server registration details at the end of the installation for each pod. This is a useful monitoring tool.

```
curl -u adminEmail:pword http://<ms_IP>:8080/v1/servers?pod=podName
```

where *`ms_IP`* is the IP address or DNS name of the Management Server, and ***podName*** is either:

- `gateway`
- `central`
- `analytics`

For example, for the "gateway" pod:

```
> curl -u adminEmail:pword http://<ms_IP>:8080/v1/servers?pod=gateway
```

You see output in the form:

```
[ {
  "externalHostName" : "localhost",
  "externalIP" : "192.168.1.11",
  "internalHostName" : "localhost",
  "internalIP" : "192.168.1.11",
  "isUp" : true,
  "pod" : "gateway",
  "reachable" : true,
  "region" : "dc-1",
  "tags" : {
    "property" : [ {
      "name" : "jmx.rmi.port",
      "value" : "1101"
    }, ... ]
  },
  "type" : [ "message-processor" ],
  "uUID" : "276bc250-7dd0-46a5-a583-fd11eba786f8"
},
{
  "internalIP" : "192.168.1.11",
  "isUp" : true,
  "pod" : "gateway",
  "reachable" : true,
  "region" : "dc-1",
  "tags" : {
    "property" : [ ]
  },
  "type" : [ "dc-datastore", "management-server", "cache-datastore",
"keyvaluemap-datastore", "counter-datastore", "kms-datastore" ],
  "uUID" : "13cee956-d3a7-4577-8f0f-1694564179e4"
},
{
  "externalHostName" : "localhost",
  "externalIP" : "192.168.1.11",
  "internalHostName" : "localhost",
  "internalIP" : "192.168.1.11",
  "isUp" : true,
  "pod" : "gateway",
  "reachable" : true,
  "region" : "dc-1",
  "tags" : {
    "property" : [ {
      "name" : "jmx.rmi.port",
      "value" : "1100"
    }, ... ]
  },
  "type" : [ "router" ],
  "uUID" : "de8a0200-e405-43a3-a5f9-eabafdd990e2"
} ]
```

Setting the port number of the Edge UI

By default, the Edge UI listens on port 9000 of the Edge node on which it is installed. Use the following procedure to change the port number:

1. Ensure that the desired port number is open on the Edge node.
2. Open `/<inst_root>/apigee/customer/edge-ui.d/ui.sh` in an editor. If the file and directory does not exist, create it.
3. Set the following propertty in `ui.sh`:

   ```
   export JAVA_OPTS="-Dhttp.address=IP -Dhttp.port=PORT"
   ```

   Where IP is the IP address of the Edge UI node, and PORT is the new port number.
4. Save the file.
5. Restart the Edge UI:

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui restart
   ```

You can now access the Edge UI on the new port number.

Handling a PostgresSQL Database Failover

Perform the following during a PostgreSQL database failover:

1) Stop `apigee-postgresql` on the current master if it is still running:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
   stop
   ```

2) Go to standby node and invoke the following command to make it the master:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
   promote-standby-to-master IPorDNSofOldMaster
   ```

If old master is restored at some time in the future, make it a standby node:

1) On the current master, edit the config file to set:

   ```
   PG_MASTER=IPorDNSofNewMaster
   PG_STANDBY=IPorDNSofOldMaster
   ```

2) Enable replication on the new master:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
setup-replication-on-master -f configFIle
```

3) On the old master, edit the config file to set:

```
PG_MASTER=IPorDNSofNewMaster
PG_STANDBY=IPorDNSofOldMaster
```

4) On the old master, clean out any old Postgres data:

```
> rm -rf /<inst_root>/apigee/data/apigee-postgresql/
```

Note: If necessary, you can backup this data before deleting it.

5) Configure the old master as a standby:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
setup-replication-on-standby -f configFile
```

6) On completion of replication, verify the replication status by issuing the following scripts on both servers. The system should display identical results on both servers to ensure a successful replication:

   a. On the master node, run:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service
   apigee-postgresql postgres-check-master
   ```

   Validate that it says it is the master.

   b. On the standby node:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service
   apigee-postgresql postgres-check-standby
   ```

   Validate that it says it is the standby.

Customizing Edge Email Templates

Edge automatically sends emails in response to certain events. For each of these events, Edge defines a default email template in `/<inst_root>/apigee/edge-ui/email-templates`. To customize these emails, you can edit the default templates.

Note: These events are triggered when modifying users in the Edge management UI. If you use a script to add or modify a user, no email is sent.

The events where an email is sent, and the default template file for the generated email, are:

- New user is added: `user-added-new.html`
- Existing user requests a password reset: `password-reset.html`
- A user is added to an organization: `user-added-existing.html`

Note: The next installation of Edge overwrites these files, so make sure to backup any customizations that you make.

The "From" field of the email can be customized by setting the `conf_apigee_apigee.mgmt.mailfrom` property in `/<inst_root>/apigee/customer/application/ui.properties` (if that file does not exist, create it). For example:

```
conf_apigee_apigee.mgmt.mailfrom="My Company <myCo@company.com>"
```

The email subjects are customizable by editing the following properties in `/<inst_root>/apigee/customer/application/ui.properties` (if that file does not exist, create it):

- `conf_apigee-base_apigee.emails.passwordreset.subject`
- `conf_apigee-base_apigee.emails.useraddedexisting.subject`
- `conf_apigee-base_apigee.emails.useraddednew.subject`

Several email properties reference the `{companyNameShort}` placeholder, which defaults to a value of "Apigee". You can set the value of the placeholder by using the `conf_apigee_apigee.branding.companynameshort` property in `ui.properties`. For example:

```
conf_apigee_apigee.branding.companynameshort="My Company"
```

After setting any properties in `/<inst_root>/apigee/customer/application/ui.properties`, you must restart the Edge UI:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui restart
```

Configuring the Edge SMTP server

An SMTP server allows the Edge UI to send emails. For example, the Edge UI can send an email to assist a user in recovering their UI password.

At install time, you can specify the SMTP information. To change the SMTP information, or if you deferred SMTP configuration at the time of installation, you can use the `apigee-setup` utility to modify the SMTP configuration.

You can change any or all of the following information:

- SMTP host, such as `smtp.gmail.com`
- SMTP port, such as 465 if you are using SSL
- SMTP username and password

To update the SMTP information:

1. Edit the configuration file that you used to install Edge, or create a new configuration file with the following information:

   ```
   SKIP_SMTP=n
   SMTPHOST=smtp.example.com
   SMTPUSER=smtp@example.com  # 0 for no username
   SMTPPASSWORD=smtppwd       # 0 for no password
   SMTPSSL=n
   SMTPPORT=25
   ```

2. Run the `apigee-setup` utility on all the Edge UI nodes to update the SMTP settings:

   ```
   > /<inst_root>/apigee/apigee-setup/bin/setup.sh -p ui -f configFile
   ```

   The `apigee-setup` utility restarts the Edge UI.

Configuring SMTP for the Apigee BaaS SMTP Server

Apigee BaaS requires that you configure an SMTP server. When you install the Apigee BaaS API Backend Stack, you enter the SMTP information, including the password of the SMTP user. That password is then encrypted before it is stored.

If you later want to change the SMTP information, edit `/<inst_root>/apigee/customer/application/usergrid.properties` on all BaaS Stack nodes.

To encrypted a new password so you can set it in `usergrid.properties`, use the `apigee-service` utility:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service baas-usergrid newPword
```

To change SMTP information:

1. Open `/<inst_root>/apigee/customer/application/usergrid.properties` in an editor. Create this file if it does not exist.

2. Set the following properties as necessary:

   ```
   # SMTP Properties
   usergrid-deployment_mail.smtp.host=smtp.gmail.com
   usergrid-deployment_mail.smtp.port=465
   usergrid-deployment_mail.smtp.auth=true
   usergrid-deployment_mail.smtp.username=your@email.com
   usergrid-deployment_mail.smtp.password=SECURE:74c57edacd3242f0ba1b1413890e17c22a5
   usergrid-deployment_mail.smtp.quitwait=false
   ```

   ```
   # SMTPS Properties
   usergrid-deployment_mail.smtps.host=smtp.gmail.com
   usergrid-deployment_mail.smtps.port=465
   usergrid-deployment_mail.smtps.auth=true
   usergrid-deployment_mail.smtps.username=your@email.com
   ```

```
usergrid-deployment_mail.smtps.password=SECURE:74c57edacd3242f0ba1b1413890e17c22a52
usergrid-deployment_mail.smtps.quitwait=false
```

3. After editing this information, you must restart the API Backend Stack by using the command:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service baas-usergrid
   restart
   ```

4. Repeat on all API BaaS Stack nodes.

Modifying Java Settings

Depending on your traffic and processing requirements you may need to increase the heap size for one or more Apigee components.

To change the heap settings, edit the properties file for the component. If the component is installed on multiple machines, such as the Edge Router, then edit the properties file on all machines hosting the component.

To set these values, set the following properties:

- `bin_setenv_min_mem` - set the minimum heap size. The default is 256 MB.
- `bin_setenv_max_mem` - set the maximum heap size. The default is 512 MB.
- `bin_setenv_max_permsize` - set the maximum permgen size. The default is 128 MB. On the Message Processor, Apigee recommends that you set this value to 256 MB or 512 MB, depending on your traffic.

Set these properties for each component on the machine. For example, for the Router, set them in the `/<inst_dir>/apigee/customer/application/router.properties` file. For the Message Processor, set them in the `/<inst_dir>/apigee/customer/application/message-processor.properties` file.

After setting the values, restart the component:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service comp restart
```

For example:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router restart
```

Setting HTTP request/response header limits

The Edge Router and Message Processor have predefined limits to the size of request/response headers and to the line size.

For the Router, which handles incoming requests from your APIs, edit the following properties in `/<inst_root>/apigee/customer/application/router.properties` to change these default values:

```
conf_router_HTTP.request.line.limit=4k
conf_router_HTTP.request.header.limit =8k
conf_router_HTTP.response.line.limit=4k
conf_router_HTTP.response.header.limit =8k
```

If that file does not exist, create it.

You must restart the Router after changing these properties:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router restart
```

For the Message Processor, which handles outgoing requests to your backend services, edit the following properties in `/<inst_root>/apigee/customer/application/message-processor.properties` to change these default values:

```
conf/http.properties+HTTPRequest.line.limit=7k
conf/http.properties+HTTPRequest.headers.limit=25k
conf/http.properties+HTTPResponse.line.limit=2k
conf/http.properties+HTTPResponse.headers.limit=25k
```

Note: The syntax is slightly different for the Message Processor because the properties are commented out by default.

If that file does not exist, create it.

You must restart the Message Processor after changing these properties:

```
> /<inst_root>/apigee/bin/apigee-service message-processor restart
```

Setting the URL of the Developer Services portal

Apigee provides you with a Developer Services portal that you can use to build and launch your own customized website to provide all of these services to your development community. Edge customers can create their own developer portal, either in the cloud or on-prem. See What is a developer portal? for more information.

The Edge UI displays the **DevPortal** button on the **Publish > Developers** page that, when clicked, opens the portal associated with an organization. By default, that button opens the following URL:

```
http://live-{orgname).devportal.apigee.com
```

where `{orgname}` is the name of the organization.

You can set this URL to a different URL, for example if your portal has a DNS record, or disable the button completely. Use the following properties of the organization to control the button:

- `features.devportalDisabled`: Set to false (default) to enable the button and to true to disable it.
- f`eatures.devportalUrl`: Set to the URL of the developer portal.

You set these properties separately for each organization. To set these properties, you first use the following API call to determine the current property settings on the organization:

```
curl -H "Content-Type:application/json" \
-u adminEmail:pword -X GET \
http://<ms-IP>:8080/v1/organizations/{orgname}
```

This call returns an object describing the organization in the form:

```
{
  "createdAt" : 1428676711119,
  "createdBy" : "me@my.com",
  "displayName" : "orgname",
  "environments" : [ "prod" ],
  "lastModifiedAt" : 1428692717222,
  "lastModifiedBy" : "me@my.com",
  "name" : "organme",
  "properties" : {
    "property" : [{
      "name" : "foo",
      "value" : "bar"}]
  },
  "type" : "paid"
}
```

Note any existing properties in the `properties` area of the object. When you set properties on the organization, the value in `properties` overwrites any current properties. Therefore, if you want to set `features.devportalDisabled` or `features.devportalUrl` in the organization, make sure you copy any existing properties when you set them.

Use the following PUT call to set `properties` on the organization:

```
curl -H "Content-Type:application/json" \
-u adminEmail:pword -X PUT \
http://<ms-IP>:8080/v1/organizations/{orgname} \
-d '{
  "displayName" : "orgname",
  "name" : "orgname",
  "properties" : {
    "property" : [ {
      "name" : "foo",
      "value" : "bar"},
    {
      "name": "features.devportalUrl",
      "value": http://dev-orgname.devportal.apigee.com/},
    {
      "name": "features.devportalDisabled",
      "value": "false"}]
  }
}'
```

In the PUT call, you only have to specify `displayName`, `name`, and `properties`. Note that this call incudes the "foo" property that was originally set on the organization.

Allowing the Trace Tool Access to Local IP Addresses

The Trace tool in the Edge UI has the ability to send and receive API request to any specified URL. In certain deployment scenarios where Edge components are co-hosted with other internal services, a malicious user may misuse the power of the Trace tool.

The `conf_apigee-base_apigee.feature.enabletraceforinternaladdresses` property is disabled by default to prevent the Trace tool from sending requests to the following private IP addresses:

- Loopback address (127.0.0.1 or localhost)
- Site-Local Addresses (For IPv4 - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16)
- Any Local Address (any address resolving to localhost).

If the Apigee Routers are reachable only over the above private IP ranges, Apigee recommends that you set the `conf_apigee-base_apigee.feature.enabletraceforinternaladdresses` property to true.

To set the property to true:

1. Open the `ui.properties` file in an editor. If the file does not exist, create it.

   ```
   > vi /<inst_root>/apigee/customer/application/ui.properties
   ```

2. Set the following property to true:

   ```
   conf_apigee-base_apigee.feature.enabletraceforinternaladdresses ="true"
   ```

3. Save your changes to `ui.properties`.
4. Restart the Edge UI:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui restart
   ```

   You can now access local IP addresses in the Trace tool.

Replacing the Edge license file

If your license expires, use the following procedure to replace it:

1. Overwrite the existing license file with the new license file:

   ```
   > cp <customer_license_file> /<inst_root>/apigee/customer/conf/license.txt
   ```

2. Change ownership of the new license file to the "apigee" user:

```
> chown apigee:apigee /<inst_root>/apigee/customer/conf/license.txt
```

Specify a different user if you are running Edge as a user other than "apigee".

3. Restart the Edge Management Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-management-server restart
```

Allow custom reports longer than 31 days in the Edge UI

By default, the Edge UI only lets you create custom reports for a maximum of 31 days.

To override this default:

1. Open the `ui.properties` file in an editor. If the file does not exist, create it.

   ```
   > vi /<inst_root>/apigee/customer/application/ui.properties
   ```

2. Set the following property to true:

   ```
   conf_apigee-base_apigee.feature.enableforcerangelimit="true"
   ```

3. Save your changes to `ui.properties`.
4. Restart the Edge UI:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui restart
   ```

Configuring the Router to retry connections to a Message Processor

The Router makes a health check to the Message Processor every five seconds to determine if the Message Processor is able to service requests. If a Message Processor goes down, the Router automatically forwards requests to another Message Processor.

You can configure how the Router reacts when the Message Processor goes down by setting the `conf_load_balancing_load.balancing.driver.nginx.server.retry` property on the Router. That property takes a space-delimited set of values that can include:

- `off`: Disable retry, the Router returns a failure code upon a request.
- `http_599`: (Default) If the Router receives an HTTP 599 response from the Message Processor, the Router forwards the request to the next Message Processor.

  HTTP 599 is a special response code that is generated by a Message Processor when it is being shutdown. The Message Processor tries to complete all existing requests, but for any new requests it responds with HTTP 599 to signal to the Router to retry the request on the next Message Processor.

- `error`: If an error occurred while establishing a connection with the Message Processor, passing a request to it, or reading the response header from it, the Router forwards the request to the next Message Processor.
- `timeout`: If a timeout occurs while establishing a connection with the Message Processor, passing a request to it, or reading the response header from it, the Router forwards the request to the next Message Processor.
- `invalid_header`: If the Message Processor returned an empty or invalid response, the Router forwards the request to the next Message Processor.
- `http_`***`XXX`***: If the Message Processor returned a response with HTTP code ***XXX***, the Router forwards the request to the next Message Processor.

To configure the Router:

1. Edit the `/<inst_root>/apigee/customer/application/router.properties` file (if the file does not exist, create it).
2. Set the `conf_load_balancing_load.balancing.driver.nginx.server.retry` property.

   ```
   conf_load_balancing_load.balancing.driver.nginx.server.retry =http_599 error
   ```

3. Restart the Router:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router
   restart
   ```

Setting log file location

By default, the log files for an Edge component are written to the `/<inst_root>/apigee/var/log/`***`componentName`*** directory. Use the following procedure to change the default location for all Edge components except the Edge UI:

1. Create the following file:

   ```
    /<inst_root>/apigee/etc/componentName.d/APIGEE_APP_LOGDIR.sh
   ```

2. Add the following property to the file:

   ```
   APIGEE_APP_LOGDIR=/foo/bar
   ```

   where ***/foo/bar*** specifies the directory for the log files.

3. Restart the component:

   ```
   > /<install_dir>/apigee/apigee-service/bin/apigee-service compName restart
   ```

Set the expiration time for user activation links in activation emails

When a user registers a new account, or requests to reset a password, they receive an email that contain an activation link. By default, that activation link expires after 600 seconds, or ten minutes.

You can configre the exiration time by setting the `conf_apigee_apigee.forgotpassword.emaillinkexpiryseconds` token in the `ui.properties` file.

To set the expiry time:

1. On the Edge UI node, edit `/<install_dir>/apigee/customer/application/ui.properties`. If that file does not exist, create it.
2. Add the following property to the file to set the expiry time in seconds:

   ```
   conf_apigee_apigee.forgotpassword.emaillinkexpiryseconds=timeInSeconds
   ```
3. Restart the Edge UI:

   ```
   > /<install_dir>/apigee/apigee-service/bin/apigee-service edge-ui restart
   ```

Enable access to OAuth 2.0 tokens by user ID and app ID

This section describes how to enable retrieval and revocation of OAuth 2.0 access tokens by end user ID, app ID, or both.

App IDs are automatically added to an OAuth access token. Therefore, after you use the procedure below to enable token access for an organization, you can access tokens by app ID.

To retrieve and revoke OAuth 2.0 access tokens by end user ID, an end user ID must be present in the access token. The procedure below describes how add an end user ID to an existing token or to new tokens.

By default, when Edge generates an OAuth 2.0 access token, the token has the format:

```
{
  "issued_at" : "1421847736581",
  "application_name" : "a68d01f8-b15c-4be3-b800-ceae8c456f5a",
  "scope" : "READ",
  "status" : "approved",
  "api_product_list" : "[PremiumWeatherAPI]",
  "expires_in" : "3599",
  "developer.email" : "tesla@weathersample.com",
  "organization_id" : "0",
  "token_type" : "BearerToken",
  "client_id" : "k3nJyFJIA3p62DWOkLO6OJNi87GYXFmP",
```

```
  "access_token" : "7S22UqXGJDTuUADGzJzjXzXSaGJL",
  "organization_name" : "myorg",
  "refresh_token_expires_in" : "0",
  "refresh_count" : "0"
}
```

Note the following:

- The `application_name` field contains the UUID of the app associated with the token. If you enable retrieval and revocation of OAuth 2.0 access tokens by app ID, this is the app ID you use.
- The `access_token` field contains the OAuth 2.0 access token value.

To enable retrieval and revocation of OAuth 2.0 access tokens by end user ID, configure the OAuth 2.0 policy to include the user ID in the token, as described in the procedure below.

The end user ID is the string that Edge uses as the developer ID, not the developer's email address. You can determine the developer's ID from the developer's email address by using the Get Developer API call.

After you configure Edge to include the end user ID in the token, it is included as the `app_enduser` field, as shown below:

```
{
  "issued_at" : "1421847736581",
  "application_name" : "a68d01f8-b15c-4be3-b800-ceae8c456f5a",
  "scope" : "READ",
  "app_enduser" : "6ZG094fgnjNf02EK",
  "status" : "approved",
  "api_product_list" : "[PremiumWeatherAPI]",
  "expires_in" : "3599",
  "developer.email" : "tesla@weathersample.com",
  "organization_id" : "0",
  "token_type" : "BearerToken",
  "client_id" : "k3nJyFJIA3p62DWOkLO6OJNi87GYXFmP",
  "access_token" : "7S22UqXGJDTuUADGzJzjXzXSaGJL",
  "organization_name" : "myorg",
  "refresh_token_expires_in" : "0",
  "refresh_count" : "0"
}
```

APIs to retrieve and revoke OAuth 2.0 access tokens by user ID and app ID

Use the following APIs to access OAuth tokens by user ID, app ID, or both:

- Get OAuth 2.0 Access Token by End User ID or App ID
- Revoke OAuth 2.0 Access Token by End User ID or App ID

Procedure for enabling token access

Use the following procedure to enable retrieval and revocation of OAuth 2.0 access tokens by end user ID and app ID.

Step 1: Enable token access support for an organization

You must enable token access for each organization separately. Call the PUT API below for each organization on which you want to enable the retrieval and revocation of OAuth 2.0 access tokens by end user ID or app ID.

The user making the following call must be in the role **orgadmin** or **opsadmin** for the organization. Replace the values in *{curly braces}* with your org-specific values:

```
> curl -H "Content-type:text/xml" -X POST \
  https://<ms-ip>:8080/v1/organizations/{org_name} \
  -d '<Organization name="{org_name}">
      <Properties>
        <Property name="features.isOAuthRevokeEnabled">true</Property>
        <Property name="features.isOAuth2TokenSearchEnabled">true</Property>
      </Properties>
    </Organization>' \
  -u {userEmail}:{mypassword}
```

Step 2: Set permissions for opsadmin role in the organization

Only the **orgadmin** and **opsadmin** roles in an organization should be given permissions to retrieve (HTTP GET) and revoke (HTTP PUT) OAuth 2.0 tokens based on user ID or app ID. To control access, set get and put permissions on the `/oauth2` resource for an organization. That resource has a URL in the form:

```
https://<ms-ip>:8080/v1/organizations/{org_name}/oauth2
```

The **orgadmin** role should already have the necessary permissions. For the **opsadmin** role for the `/oauth2` resource, the permissions should look like this:

```
<ResourcePermission path="/oauth2">
    <Permissions>
        <Permission>get</Permission>
        <Permission>put</Permission>
    </Permissions>
</ResourcePermission>
```

You can use the Get Permission for a Single Resource API call to see which roles have permissions for the `/oauth2` resource.

Based on the response, you can use the Add Permissions for Resource to a Role and Delete Permission for Resource API calls to make any necessary adjustments to the /oauth2 resource permissions.

Use the following cURL command to give the **opsadmin** role `get` and `put` permissions for the `/oauth2` resource. Replace the values in {*curly braces*} with your org-specific values:

```
> curl -X POST -H 'Content-type:application/xml' \
  http://<ms-ip>:8080/v1/organizations/{org}/userroles/opsadmin/permissions \
  -d '<ResourcePermission path="/oauth2">
      <Permissions>
        <Permission>get</Permission>
        <Permission>put</Permission>
      </Permissions>
    </ResourcePermission>' \
  -u {USEREMAIL}:{PWD}
```

Use the following cURL command to revoke `get` and `put` permissions for the `/oauth2` resource from roles other than **orgadmin** and **opsadmin**. Replace the values in {*`curly braces`*} with your org-specific values:

```
> curl -X DELETE -H 'Content-type:application/xml' \
  http://<msip>:8080/v1/organizations/{org-name}/userroles/{roles}/permissions \
  -d '<ResourcePermission path="/oauth2">
      <Permissions></Permissions>
    </ResourcePermission>' \
   -u {USEREMAIL}:{PWD}
```

Step 3: Set the oauth_max_search_limit property

Ensure that the `conf_keymanagement_oauth_max_search_limit` property in `customer/application/management-server.properties` file is set to 100:

```
conf_keymanagement_oauth_max_search_limit = 100
```

If this file does not exist, create it.

This property sets the page size used when fetching tokens. Apigee recommends a value of 100, but you can set it as you see fit.

On a new installation, the property should be already set to 100. If you have to change the value of this property, restart the Management Server and Message Processor by using the commands:

```
> /<inst_rt>/apigee/apigee-service/bin/apigee-service edge-management-server
restart
```

```
> /<inst_rt>/apigee/apigee-service/bin/apigee-service edge-message-processor
restart
```

Step 4: Configure the OAuth 2.0 policy that generates tokens to include end user ID

Note: This task is optional. You only need to perform it if you want to retrieve and revoke OAuth 2.0 access tokens by end user ID.

Configure the OAuth 2.0 policy used to generate access tokens to include the end user ID in the token. By including end user IDs in the access token, you can retrieve and revoke tokens by ID.

To configure the policy to include an end user ID in an access token, the request that creates the access token must include the end user ID and you must specify the input variable that contains the end user ID.

The OAuth 2.0 policy below, named GenerateAccessTokenClient, generates an OAuth 2.0 access token. Note the addition of the `<AppEndUser>` tag in bold that specifies the variable containing the end user ID:

```
<OAuthV2 async="false" continueOnError="false" enabled="true"
name="GenerateAccessTokenClient">
      <DisplayName>OAuth 2.0.0 1</DisplayName>
      <ExternalAuthorization>false</ExternalAuthorization>
      <Operation>GenerateAccessToken</Operation>
      <SupportedGrantTypes>
          <GrantType>client_credentials</GrantType>
      </SupportedGrantTypes>
      <GenerateResponse enabled="true"/>
      <GrantType>request.queryparam.grant_type</GrantType>
<AppEndUser>request.header.appuserID</AppEndUser><ExpiresIn>960000</ExpiresIn>
</OAuthV2>
```

You can then use the following cURL command to generate the OAuth 2.0 access token, passing the user ID as the `appuserID` header:

```
> curl -H "appuserID:6ZG094fgnjNf02EK" \
https://myorg-test.apigee.net/oauth/client_credential/accesstoken?grant_type=client_credentials \
  -X POST \
  -d 'client_id=k3nJyFJIA3p62TKIkLO6OJNXFmP&client_secret=gk5K5lIp943AY4'
```

In this example, the `appuserID` is passed as a request header. You can pass information as part of a request in many ways. For example, as an alternative, you can:

- Use a form parameter variable: `request.formparam.appuserID`
- Use a flow variable providing the end user ID

Configuring SSL

Secure Socket Layer (SSL) is a best practice for ensuring secure, encrypted messaging across your API environment, from apps to Apigee Edge to your back-end services.

Regardless of the environment configuration for your management API—for example, whether you're using a proxy, a router, and/or a load balancer in front of your management API (or not)— Edge lets you enable and configure SSL, giving you control over message encryption in your on-premise API management environment.

For an on-premises installation of Edge Private Cloud, there are several places where you can configure SSL:

1. Between a Router and Message Processor
2. For access to the Edge management API
3. For access to the Edge management UI
4. For access from an app to your APIs
5. For access from Edge to your backend services

Configuring SSL for the first three items is described below. All of these procedures assume that you have created a JKS file containing your SSL certification and private key.

To configure SSL for access from an app to your APIs, #4 above, see Creating a virtual host for an on-premises Edge installation. To configure SSL for access from Edge to your backend services, #5 above, see Configuring SSL from Edge to the backend service.

For a complete overview of configuring SSL on Edge, see SSL.

Creating a JKS file

You represent the keystore as a JKS file, where the keystore contains your SSL certificate and private key. There are several ways to create a JKS file, but one way is to use the `openssl` and `keytool` utilities.

Note: If you have a certificate chain, all certs in the chain must be appended in order into a single PEM file, where the last certificate is signed by a CA.

For example, you have a PEM file named `server.pem` containing your SSL certificate and a PEM file named `private_key.pem` containing your private key. Use the following commands to create the PKCS12 file:

```
> openssl pkcs12 -export -clcerts -in server.pem -inkey private_key.pem -out
keystore.pkcs12
```

You have to enter the passphrase for the key, if it has one, and an export password. This command creates a PKCS12 file named `keystore.pkcs12`.

Use the following command to convert it to a JKS file named `keystore.jks`:

```
> keytool -importkeystore -srckeystore keystore.pkcs12 -srcstoretype pkcs12
-destkeystore keystore.jks -deststoretype jks
```

You are prompted to enter the new password for the JKS file, and the existing password for the PKCS12 file.

Generating an obfuscated password

Some parts of the Edge SSL configuration procedure require you to enter an obfuscated password in a configuration file. An obfuscated password is a more secure alternative to entering your password in plain text.

You can generate an obfuscated password in Java by using the Jetty .jar files installed with Edge. enerate the obfuscated password by using a command in the form:

```
> java -cp /<inst_root>/apigee/edge-gateway/lib/thirdparty/jetty-http-
x.y.z.jar:/<inst_root>/apigee/edge-gateway/lib/thirdparty/jetty-util- x.y.z.jar
org.eclipse.jetty.http.security.Password yourPassword
```

where **x.y.z** specifies the version number of the Jetty .jar files, such as 8.0.4.v20111024. This command returns the password in the form:

```
yourPassword
OBF:58fh40h61svy156789gk1saj
MD5:902fobg9d80e6043b394cb2314e9c6
```

Use the obfuscated password specified by `OBF` when configuring SSL.

For more information, see this article.

Configuring SSL between a Router and a Message Processor

By default, SSL between the Router and Message Processor is disabled.

Note: Port 8082 on the Message Processor has to be open for access by the Router to perform health checks when you configure TLS/SSL between the Router and Message Processor. If you do not configure TLS/SSL between the Router and Message Processor, the default configuration, port 8082 still must be open on the Message Processor to manage the component, but the Router does not require access to it.

Use the following procedure to enable SSL encryption between a Router and the Message Processor:

1. Ensure that port 8082 on the Message Processor is accessible by the Router.
2. Generate the keystore JKS file containing your SSL certification and private key. For more, see Creating a JKS file.
3. Copy the keystore JKS file to the `/tmp` directory on the Message Processor server.
4. Change permissions and ownership of the JKS file:

   ```
   > chown apigee:apigee /tmp/keystore.jks
   > chmod 600 /tmp/keystore.jks
   ```

   where *`keystore`*`.jks` is the name of your keystore file.

5. Edit the file `/<inst_root>/apigee/customer/application/message-processor.properties`. If the file does not exist, create it.
6. Set the following properties in the `message-processor.properties` file:

```
conf_message-processor-communication_local.http.ssl=true
conf_message-processor-communication+local.http.port=8443
conf_message-processor-communication+local.http.ssl.keystore.type=jks
conf_message-processor-communication+local.http.ssl.keystore.path=/tmp/keyStore.jks
conf_message-processor-communication+local.http.ssl.keyalias=apigee-devtest
# Enter the obfuscated keystore password below.
conf_message-processor-communication+local.http.ssl.keystore.password=OBF:obsPword
```

   where ***keyStore.jks*** is your keystore file, and ***obsPword*** is your obfuscated keystore and keyalias password. See Generating an obfuscated password for information on generating an obfuscated password.

7. Stop the Message-Processors and Routers:

```
/<install_dir>/apigee/apigee-service/bin/apigee-service
edge-message-processor stop
/<install_dir>/apigee/apigee-service/bin/apigee-service edge-router stop
```

8. On the Router, delete any files in /opt/nginx/conf.d:

```
> rm -f /opt/nginx/conf.d/*
```

9. Start the Message-Processors and Routers:

```
/<install_dir>/apigee/apigee-service/bin/apigee-service
edge-message-processor start
/<install_dir>/apigee/apigee-service/bin/apigee-service edge-router start
```

10. Repeat for any additional Message Processors.

The following table lists all of the available properties in `message-processor.properties`:

Properties	**Description**
conf_message-processor-communication_local.http.host=<localhost or IP address>	Optional. Hostname to listen on for router connections. This will override the host name configured at registration.

conf_message-processor-communication_local.http.port=8998	Optional. Port to listen on for router connections. Default is 8998.
conf_message-processor-communication_local.http.ssl=<false | true>	Set this to true to enable SSL. Default is false. When SSL is enabled, you must set local.http.ssl.keystore.path and local.http.ssl.keyalias.
conf_message-processor-communication_local.http.ssl.keystore.path=	Local file system path to the keystore (JKS or PKCS12). Mandatory when local.http.ssl=true.
conf_message-processor-communication_local.http.ssl.keyalias=	Key alias from the keystore to be used for SSL connections. Mandatory when local.http.ssl=true.
conf_message-processor-communication_local.http.ssl.keyalias.password=	Password used for encrypting the key inside the keystore. Use an obfuscated password in this format: OBF:xxxxxxxxxx
conf_message-processor-communication_local.http.ssl.keystore.type=jks	Keystore type. Only JKS and PKCS12 are currently supported. Default is JKS.
conf_message-processor-communication_local.http.ssl.keystore.password=	Optional. Obfuscated password for the keystore. Use an obfuscated password in this format: OBF:xxxxxxxxxx
conf_message-processor-communication_local.http.ssl.ciphers=<cipher1,cipher2>	Optional. When configured, only the ciphers listed are allowed. If omitted, use all ciphers supported by the JDK.

Configuring SSL for the management API

By default, SSL is disabled for the management API and you access the Edge management API over HTTP by using the IP address of the Management Server node and port 8080. For example:

```
http://ms_IP:8080
```

Alternatively, you can configure SSL access to the management API so that you can access it in the form:

```
https://ms_IP:8443
```

In this example, you configure SSL access to use port 8443. However, that port number is not required by Edge - you can configure the Management Server to use other port values. The only requirement is that your firewall allows traffic over the specified port.

To ensure traffic encryption to and from your management API, configure the settings in the `/<install_dir>/apigee/customer/application/management-server.properties` file.

In addition to SSL configuration, you can also control password validation (password length and strength) by modifying the `management-server.properties` file.

Note: After modifying these files, restart your management API service.

Ensure that your SSL port is open

The procedure in this section configures SSL to use port 8443 on the Management Server. Regardless of the port that you use, you must ensure that the port is open on the Management Server. For example, you can use the following command to open it:

```
$ iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 8443 -j ACCEPT
--verbose
```

Configure SSL

Edit the `/<install_dir>/apigee/customer/application/management-server.properties` file to control SSL use on traffic to and from your management API. If this file does not exist, create it.

Use the following procedure to configure SSL access to the management UI:

1. Generate the keystore JKS file containing your SSL certification and private key. For more see Creating a JKS file.
2. Copy the keystore JKS file to a directory on the Management Server node, such as /tmp.
3. Change ownership of the JKS file to `apigee`:

   ```
   $ chown apigee:apigee keystore.jks
   ```

   where ***keystore*****.jks** is the name of your keystore file.
4. Edit `/<install_dir>/apigee/customer/application/management-server.properties` to set the following properties. If that file does not exist, create it:

   ```
   conf_webserver_ssl.enabled=true
   # Set next to true if all communications should be over HTTPS.
   #  Not recommended because many Edge internal calls use HTTP.
   conf_webserver_http.turn.off=false
   conf_webserver_ssl.port=8443
   conf_webserver_keystore.path=/tmp/keystore.jks
   # Enter the obfuscated keystore password below.
   ```

```
conf_webserver_keystore.password=OBF:obfuscatedPassword
conf_webserver_cert.alias=apigee-devtest
```

where ***keyStore.jks*** is your keystore file, and ***obfuscatedPassword*** is your obfuscated keystore password. See Generating an obfuscated password for information on generating an obfuscated password.

5. Restart the Edge Management Server by using the commands:

```
$ /<install_dir>/apigee/apigee-service/bin/apigee-service edge-management-server
restart
```

The management API now supports access over SSL.

The following table lists all of the SSL properties that you can set in `management-server.properties`:

Properties	**Description**
conf_webserver_http.port=8080	Default is 8080.
conf_webserver_ssl.enabled=false	To enable/disable SSL. With SSL enabled (true), you must also set the ssl.port and keystore.path properties.
conf_webserver_http.turn.off=true	To enable/disable http along with https. If you want to use only HTTPS, leave the default value to true.
conf_webserver_ssl.port=8443	These are mandatory when SSL is enabled (ssl.enabled=true)
conf_webserver_keystore.path=<path>	Modify the SSL port to match your server's SSL port, and provide the path to your keystore file.
conf_webserver_keystore.password=	Use an obfuscated password in this format: OBF:xxxxxxxxxx
conf_webserver_cert.alias=	Optional keystore certificate alias
conf_webserver_keymanager.password=	If your key manager has a password, enter an obfuscated version of the password in this format: OBF:xxxxxxxxxx

conf_webserver_trust.all= <false | true>
conf_webserver_trust.store.path=<path>
conf_webserver_trust.store.password=
	Configure settings for your trust store. Determine whether you want to accept all SSL certificates (for example, to accept non-standard types). The default is false. Provide the path to your trust store, and enter an obfuscated trust store password in this format: OBF:xxxxxxxxxx

conf_webserver_exclude.cipher.suites=<CIPHER_SUITE_1 CIPHER_SUITE_2> conf_webserver_include.cipher.suites=	Indicate any cipher suites you want to include or exclude. For example, if you discover vulnerability in a cipher, you can exclude it here. Separate multiple ciphers with a space. For information on cypher suites and cryptography architecture, see: http://docs.oracle.com/javase/8/docs/technotes/guides/security/SunProviders.html#SunJSSE
conf_webserver_ssl.session.cache.size= conf_webserver_ssl.session.timeout=	Integers that determine: ● The SSL session cache size (in bytes) for storing session information for multiple clients. ● The amount of time SSL sessions can last before they time out (in milliseconds).

Configuring SSL for the management UI

By default, you access the Edge management UI over HTTP by using the IP address of the Management Server node and port 9000. For example:

```
http://ms_IP:9000
```

Alternatively, you can configure SSL access to the management UI so that you can access it in the form:

```
https://ms_IP:9443
```

In this example, you configure SSL access to use port 9443. However, that port number is not required by Edge - you can configure the Management Server to use other port values. The only requirement is that your firewall allows traffic over the specified port.

Ensure that your SSL port is open

The procedure in this section configures SSL to use port 9443 on the Management Server. Regardless of the port that you use, you must ensure that the port is open on the Management Server. For example, you can use the following command to open it:

```
$ iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 9443 -j ACCEPT
--verbose
```

Configure SSL

Use the following procedure to configure SSL access to the management UI:

1. Generate the keystore JKS file containing your SSL certification and private key and copy it to the Management Server node. For more information, see Creating a JKS file.
2. Run the following command to configure SSL:

   ```
   $ /<inst_root>/<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-ui configure-ssl
   ```
3. Enter the HTTPS port number, for example, 9443.
4. Specify if you want to disable HTTP access to the management UI. By default, the management UI is accessible over HTTP on port 9000.
5. Enter the keystore algorithm. The default is JKS.
6. Enter the absolute path to the keystore JKS file.

   The script copies the file to the `/<install_dir>/apigee/customer/conf` directory on the Management Server node, and changes the ownership of the file to `apigee`.
7. Enter the keystore password.
8. The script then restarts the Edge management UI. After the restart, the management UI supports access over SSL.

Recurring Edge Services Maintenance Tasks

In order to ensure optimal day-to-day operation of the Apigee system, certain tasks should be performed when the system is originally installed and/or on a periodic basis.

Maintenance Tool Set

The following tools are used to communicate with, or maintain various components of, the Apigee system. The variable $APROOT refers to the directory in which the Apigee system is installed.

Tool	Used For	System Location
nodetool	Apache Cassandra maintenance	/<inst_root>/apigee/apigee-cassandra/bin
cassandra-cli	Apache Cassandra command line	/<inst_root>/apigee/apigee-cassandra/bin
zkCli.sh	Apache ZooKeeper command line utility	/<inst_root>/apigee/apigee-zookeeper/bin
nc	Arbitrary TCP/IP and UDP commands; invocation of ZooKeeper "four-letter commands"	/usr/bin/nc or another location dependent on your operating system

In situations where the "`nc`" or "`telnet`" commands might be considered a security risk, the following Python script can be used:

```
import time

import socket

import sys

if len(sys.argv) <> 4:

      print "Usage: %s address port 4-letter-cmd" % sys.argv[0]

else:

      c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

      c.connect((sys.argv[1], int(sys.argv[2])))

      c.send(sys.argv[3])

      time.sleep(0.1)

      print c.recv(512)
```

Apache Cassandra Maintenance Tasks

Anti-Entropy Maintenance

The Apache Cassandra ring nodes require periodic maintenance to ensure consistency across all nodes. To perform this maintenance, use the Cassandra "`nodetool -h localhost repair`" command.

Note: This maintenance must be run on every Cassandra node at least every ten days in order to eliminate problems related to Cassandra "forgotten deletes". Running " `nodetool -h localhost repair` " imposes a significant load on the system, so Apigee recommends that this process only be run during periods of relatively low workload.

In a single-region or single-data-center Apigee installation, use the command " `nodetool -h localhost repair` " on one Cassandra node to ensure consistency across all nodes in the ring. In a multi-region or multi-data-center Apigee installation, use "`nodetool -h localhost repair -pr`" on every node in the ring (across all regions or data centers).

For more information on "forgotten deletes" and Cassandra consistency, and for instructions on how to use "`nodetool`", see:

http://wiki.apache.org/cassandra/Operations - Consistency

Important Note: Apigee does not recommend adding, moving or removing Cassandra nodes without contacting Apigee Customer Success. The Apigee system tracks Cassandra nodes using their IP address, and performing ring maintenance without performing corresponding updates on the Apigee environment metadata will cause undesirable results.

Log File Maintenance

Cassandra logs are stored in the `/<inst_root>/apigee/var/log/cassandra` directory on each node. By default, a maximum of 50 log files, each with a maximum size of 20 MB, can be created; once this limit is reached older logs are deleted when newer logs are created.

If you should find that Cassandra log files are taking up excessive space, you can modify the amount of space allocated for log files by editing the log4j settings.

1. Edit `/<install_dir>/apigee/customer/application/cassandra.properties` to set the following properties. If that file does not exist, create it:

```
conf_log4j-server_log4j.appender.r.maxfilesize=20MB  # max file size
conf_log4j-server_log4j.appender.r.maxbackupindex=50 # max open files
```

2. Restart Cassandra by using the commands:

```
$ /<install_dir>/apigee/apigee-service/bin/apigee-service apigee-cassandra restart
```

Apache Zookeeper Maintenance Tasks

Four-Letter Commands

Apache ZooKeeper has a number of "four-letter commands" that can be helpful in determining the current status of ZooKeeper voter and observer nodes. These commands can be invoked using "`nc`", "`telnet`" or another utility that has the ability to send commands to a specific port. Details on the four-letter commands can be found at:

http://zookeeper.apache.org/doc/r3.1.2/zookeeperAdmin.html#sc_zkCommands.

Removing Old Snapshot Files

Apache ZooKeeper also requires periodic maintenance to remove old snapshot files which accumulate as updates to the system are made. Instructions on how to clean up old snapshots can be found at

http://zookeeper.apache.org/doc/r3.1.2/zookeeperAdmin.html#sc_maintenance.

Log File Maintenance

Apache Zookeeper log files are kept in `/<inst_root>/apigee/var/log/zookeeper`. Normally, log file maintenance should not be required, but if you find that there are an excessive number of ZooKeeper logs or that the logs are very large you can modify ZooKeeper's log4j properties to set the maximum file size and file count.

1. Edit `/<install_dir>/apigee/customer/application/zookeeper.properties` to set the following properties. If that file does not exist, create it:

   ```
   conf_log4j_log4j.appender.rollingfile.maxfilesize=10MB  # max file size
   conf_log4j_log4j.appender.rollingfile.maxbackupindex=50 # max open files
   ```

2. Restart ZooKeeper by using the commands:

   ```
   $ /<install_dir>/apigee/apigee-service/bin/apigee-service apigee-zookeeper restart
   ```

OpenLDAP Maintenance Tasks

OpenLDAP log files are contained in the directory `/<inst_root>/apigee/var/log`**.** These files can be periodically archived and removed in order to ensure that they do not take up excessive disk space. Information on maintaining, archiving and removing OpenLDAP logs can be found in Section 19.2 of the OpenLDAP manual at http://www.openldap.org/doc/admin24/maintenance.html.

Recurring Analytics Services Maintenance Tasks

Many Apigee Analytics Services tasks can be performed using standard Postgres utilities. The routine maintenance tasks you would perform on the Analytics database – such as database reorganization using VACUUM, reindexing and log file maintenance - are the same as those you would perform on any PostgreSQL database. Information on routine Postgres maintenance can be found at http://www.postgresql.org/docs/9.1/static/maintenance.html.

Important Note: Apigee does not recommend moving the PostgreSQL database without contacting Apigee Customer Success. The Apigee system PostgreSQL database servers using their IP address, and moving the database or changing its IP address without performing corresponding updates on the Apigee environment metadata will cause undesirable results.

For more on maintaining PostgreSQL database, see http://www.postgresql.org/docs/9.1/static/maintenance.html.

Note: No pruning is required for Cassandra. Expired tokens are automatically purged after 180 days.

Pruning Analytics Data

As the amount of analytics data available within the Apigee repository increases, you may find it desirable to "prune" the data beyond your required retention interval. Run the following command to prune data for a specific organization and environment:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
pg-data-purge <Org> <Env> <NoOfDaysToPurgeBackFromCurrentDate>
```

This command interrogates the "childfactables" table in the "analytics" schema to determine which raw data partitions cover the dates for which data pruning is to be performed, then drops those tables. Once the tables are dropped, the entries in "childfactables" related to those partitions are deleted.

Childfactables are daily-partitioned fact data. Every day new partitions are created and data gets ingested into the daily partitioned tables. So at a later point in time, when the old fact data will not be required, you can purge the respective childfactables.

Note: If you have configured master-standby replication between Postgres servers, then you only need to run this script on the Postgres master.

Organization and Environment Maintenance

This section covers various administrative operations, for example, creation, management and removal of Apigee organizations, environments and virtual hosts in an Apigee Edge for Private Cloud installation.

For an introduction to organizations, environments, and virtual hosts, see About Regions, Pods, Organizations, Environments and Virtual Hosts.

Checking Status of Users, Organization and Environment

Management Server plays a vital role in holding all other components together in an on-premises installation of Edge Private Cloud. You can check for user, organization and deployment status on the Management Server by issuing the following CURL commands:

```
curl -u <adminEmail>:<admin passwd> http://localhost:8080/v1/users
```

```
curl -u <adminEmail>:<admin passwd> http://localhost:8080/v1/organizations
```

```
curl -u <adminEmail>:<admin passwd>
http://localhost:8080/v1/organizations/<orgname>/deployments
```

The system should display 200 HTTP status for all calls. If these fail, do the following:

1. Check the Management Server logs (at `<inst_root>/apigee/var/log/apigee/management-server`) for any errors.
2. Make a call against Management Server to check whether it is functioning properly.
3. Remove the server from the ELB and then restart the Management Server.

   ```
   /<inst_root>/apigee/bin/apigee-service management-server restart
   ```

About using config files

The commands shown below take a config file as input. For example, you pass a config file to the `setup-org` command to define all the properties of the organization, including the environment and virtual host.

For a complete config file, and information on the properties that you can set in the config file, see http://docs.apigee.com/api-services/content/onboard-organization.

About setting up a virtual host

A virtual host on Edge defines the domains and Edge Router ports on which an API proxy is exposed, and, by extension, the URL that apps use to access an API proxy. A virtual host also defines whether the API proxy is accessed by using the HTTP protocol, or by the encrypted HTTPS protocol.

Use the scripts and API calls shown below to create a virtual host. When you create the virtual host, you must specify the following information:

- **The name** of the virtual host that you use to reference it in your API proxies.
- **The port** on the Router for the virtual host. Typically these ports start at 9001 and increment by one for every new virtual host.
- **The host alias** of the virtual host. Typically the DNS name of the virtual host.

  The Edge Router compares the `Host` header of the incoming request to the list of host aliases as part of determining the API proxy that handles the request. When making a request through a virtual host, either specify a domain name that matches the host alias of a virtual host, or specify the IP address of the Router and the `Host` header containing the host alias.

For example, if you created a virtual host with a host alias of **myapis.apigee.net** on port **9001**, then a cURL request to an API through that virtual host could use one of the following forms:

- If you have a DNS entry for **myapis.apigee.net**:

  ```
  curl http://myapis.apigee.net:9001/{proxy-base-path}/{resource-path}
  ```

- If you do not have a DNS entry for **myapis.apigee.net**:

  ```
  curl http://<routerIP>:9001/{proxy-base-path}/{resource-path}
       -H 'Host: myapis.apigee.net'
  ```

  In the second form, you specify the IP address of the Router, and pass the host alias in the `Host` header.

  Note: The curl command, most browsers, and many other utilities automatically append the `Host` header with the domain as part of the request, so you can actually use a curl command in the form:

  ```
  curl http://<routerIP>:9001/{proxy-base-path}/{resource-path}
  ```

Options when you do not have a DNS entry for the virtual host

One option when you do not have a DNS entry is to set the host alias to the IP address of the Router and port of the virtual host, as `<routerIP>:port`. For example:

```
192.168.1.31:9001
```

When you make a curl command in the form below:

```
curl http://<routerIP>:9001/{proxy-base-path}/{resource-path}
```

This option is preferred because it works well with the Edge UI.

If you have multiple Routers, add a host alias for each Router, specifying the IP address of each Router and port of the virtual host.

Alternatively, you can set the host alias to a value, such as **temp.hostalias.com**. Then, you have to pass the `host` header on every request:

```
curl -v http://<routerIP>:9001/{proxy-base-path}/{resource-path}
     -H 'Host: temp.hostalias.com'
```

Or, add the host alias to your `/etc/hosts` file. For example, add this line to `/etc/hosts`:

```
192.168.1.31   temp.hostalias.com
```

Then you can make a request as if you had a DNS entry:

```
curl -v http://myapis.apigee.net:9001/{proxy-base-path}/{resource-path}
```

Creating an Organization, Environment, and Virtual Host

Before you create an API proxy on Apigee Edge, you must create at least one organization and, within each organization, one or more environments and virtual hosts.

Typically, organizations and environments are created together. To simplify the process, use the `apigee-provision` utility. Invoke it from the command line on the Edge Management Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-provision
setup-org -f configFile
```

The config file contains:

```
     APIGEE_ADMINPW=adminPword # If omitted, you are prompted for it.
     NEW_USER="y"
     USER_NAME=orgAdmin@myCo.com
     FIRST_NAME=foo
     LAST_NAME=bar
     USER_PWD="userPwrod"
     ORG_NAME=example  # lowercase only, no spaces, underscores, or periods.
     ENV_NAME=prod
     VHOST_PORT=9001
     VHOST_NAME=default
     VHOST_ALIAS="$IP1:9001"
     # Optionally configure SSL for virtual host.
     # VHOST_SSL=y      # Set to "y" to enable SSL on the virtual host.
     # KEYSTORE_JAR=    # JAR file containing the cert and private key.
     # KEYSTORE_NAME=   # Name of the keystore.
     # KEYSTORE_ALIAS= # The key alias.
     # KEY_PASSWORD=    # The key password, if it has one.
     # AXGROUP=axgroup-001 # Default name is axgroup-001
```

Note: For SSL configuration, see Keystores and Truststores and Configuring SSL access to an API for the Private Cloud for more information on creating the JAR file, and other aspects of configuring SSL.

The command then:

- Create the organization

  Note: You cannot rename an organization after you create it.

- Associate the organization with a pod, by default is associates it with the "gateway" pod
- Add the specified user as the org admin. If the user does not exist, you can create one.
- Create one or more environments
- Create one or more virtual host for each environment
- Associate the environment with all Message Processor(s)
- Enable analytics

For a complete silent config file, see http://docs.apigee.com/api-services/content/onboard-organization.

By default, the maximum length of the organization name and environment name is 20 characters when using the `apigee-provision` utility. This limit does not apply if you use the Edge API directly to create the organization or environment.

Note: Rather than using the `apigee-provision` utility, you can use a set of additional scripts, or the Edge API itself, to create and configure an organization. For example, use thee scripts to add an organization and associate it with multiple pods. The following sections describe those scripts and APIs in more detail.

Create an Organization

Use the `create-org` command to create an organization:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-provision
create-org -f configFile
```

This script creates the organization, but does not add or configure the environments and virtual hosts required by the organization to handle API calls.

Note: You cannot rename an organization after you create it.

The config file contains the name of the org and the email address of the org admin. The characters you can use in the name attribute are restricted to: `a-z0-9\-$%`. Do not use spaces, periods or upper-case letters in the name:

```
APIGEE_ADMINPW=adminPword  # If omitted, you are prompted for it.
ORG_NAME=example  # lowercase only, no spaces, underscores, or periods.
ORG_ADMIN=orgAdmin@myCo.com
```

The command then:

- Creates the organization
- Associates the organization with a pod, by default is associates it with the "gateway" pod
- Adds the specified user as the org admin. The user must already exist; otherwise the script issues an error.

Note: You cannot create two organizations with the same name. In that case, the second create will fail, like this:

```
<Error>
    <Code>organizations.OrganizationAlreadyExists</Code>
    <Message>Organization : test already exists</Message>
    <Contexts/>
</Error>
```

Create an organization by using API calls

Alternatively, you can use the following API calls to create an org. The first call creates the org:

```
curl -H "Content-Type:application/xml" -u <adminEmail>:<admin passwd> \
-X POST http://<ms-ip>:8080/v1/organizations \
-d '<Organization name="<org-name>" type="paid"/>'
```

The next call associates the org with a pod:

```
curl -H "Content-Type:application/x-www-form-urlencoded" \
-u <adminEmail>:<admin passwd> -X POST \
http://<ms-ip>:8080/v1/organizations/<org-name>/pods \
-d "region=default&pod=gateway"
```

You can make this call multiple times to associate the organization with multiple pods.

The final call adds an existing user as the org admin for the org:

```
curl -H "Content-Type:application/xml" -u <adminEmail>:<admin passwd> \
-X POST \
http://<ms-ip>:8080/v1/organizations/<org-name>/users/<user-email>/userroles/ \
-d '<Roles><Role name="orgadmin"/></Roles>'
```

Note: Currently, Apigee Edge for Private Cloud supports the roles – orgadmin, readonlyadmin, opsadmin, user, and businessuser, all having a default permission of full access to entities (APIs, API products, apps, developers, and reports) in an Apigee organization. Depending on the customer's needs, you can customize the pre-defined or configure more complex roles and permissions using RBAC API. See http://apigee.com/docs/api/user-roles (or http://apigee.com/docs/content/documentation-archives to find the docs that correspond to earlier versions of the product).

If the user does not exist, you can use the following call to create the user as described in Adding a user.

Create an Environment

Use the `add-env` command to create an environment in an existing organization:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-provision
add-env -f configFile
```

This config file contain the information necessary to create the environment and virtual host:

```
APIGEE_ADMINPW=adminPword  # If omitted, you are prompted for it.
ORG_NAME=example  # lowercase only, no spaces, underscores, or periods.
ENV_NAME=prod
VHOST_PORT=9001
VHOST_NAME=default
VHOST_ALIAS="$IP1:9001"
# Optionally configure SSL for virtual host.
# VHOST_SSL=y     # Set to "y" to enable SSL on the virtual host.
# KEYSTORE_JAR=   # JAR file containing the cert and private key.
# KEYSTORE_NAME=  # Name of the keystore.
# KEYSTORE_ALIAS= # The key alias.
# KEY_PASSWORD=   # The key password, if it has one.
# AXGROUP=axgroup-001 # Default name is axgroup-001
```

Note: For SSL configuration, see Keystores and Truststores and Configuring SSL access to an API for the Private Cloud for more information on creating the JAR file, and other aspects of configuring SSL.

The command:

- Creates the environment
- Creates a single virtual host for the environment
- Associates the environment with all Message Processor(s) in the pod associated with the organization containing the environment.
- Enables analytics

  Note: If you enable analytics for one environment in an organization, you must enable analytics for all environments in the organization.

Create an environment by using API calls

Alternatively, you can use the following API calls to create an environment. The first call creates the environment:

```
curl -H "Content-Type:application/xml" -u <admin user>:<admin passwd> \
-X POST http://<ms-ip>:8080/v1/organizations/<org-name>/environments \
-d  '<Environment name="<env-name>"/>'
```

Note: The characters you can use in the `name` attribute are restricted to: `A-Z0-9._\-$ %.`

The next call associates the environment with a Message Processor. Make this call for each Message Processor that you want to associate with the environment:

```
curl -H "Content-Type:application/x-www-form-urlencoded" \
-u <admin user>:<admin passwd> -X POST \
http://<ms-ip>:8080/v1/organizations/<org-name>/environments/<env-name>/servers
```

```
\
-d "action=add&uuid=<uuid>"
```

Where "`<uuid>`" is the UUID of Message Processor. You can obtain the UUID by using the command:

```
> curl http://<mp-ip>:8082/v1/servers/self
```

Where "`<mp-ip>`" is the IP address of the Message Processor.

The next API call enables Analytics for a given environment. It validates the existence of Qpid and Postgres Servers in the PODs of all datacenters. Then it starts the Analytics onboarding for the given organization and environment.

Note: Rather than use the following API call, use the following command to enable Analytics:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-provision
enable-ax -f configFile
```

This config file contains:

```
ORG_NAME=orgName
```

```
ENV_NAME=envName
```

Note: If you enable analytics for one environment in an organization, you must enable analytics for all environments in the organization.

```
curl -H "Content-Type:application/json" -u <admin user>:<admin passwd>
-X POST
http://<ms-ip>:8080/v1/organizations/<org-name>/environments/<env-name>/analyti
cs/admin -d "@sample.json"
```

Where `sample.json` contains entries for Analytics onboarding. Contact Apigee Support to obtain the JSON file.

Create a Virtual Host

You can create a virtual host in an existing environment in an organization. Often an environment supports multiple virtual hosts. For example, one virtual host might support the HTTP protocol, while another virtual host in the same environment supports the encrypted HTTPS protocol.

Use the following API call to create additional virtual hosts, or to create a virtual host for an environment with no virtual host:

```
curl -H "Content-Type:application/xml" -u <admin user>:<admin passwd> \
-X POST \
http://<ms-ip>:8080/v1/organizations/<org-name>/environments/<env-name>/virtual
hosts \
-d '<VirtualHost name="default"> \
    <HostAliases> \
      <HostAlias>myorg-test.apigee.net</HostAlias> \
    </HostAliases> \
    <Interfaces/> \
    <Port>443</Port> \
  </VirtualHost>'
```

Note: The characters you can use in the `name` attribute are restricted to: `A-Z0-9._\-$ %.`

For a complete description of creating a virtual host, including creating a secure virtual host that uses SSL over HTTPS, see http://apigee.com/docs/api-services/content/creating-virtual-host.

Deleting a Virtual Host/Environment/Organization

This section explains the removal of organizations, environments, and virtual hosts. Note that the order of API calls is very important– that means, for example, the step to remove an organization can only be executed after you remove all associated environments in the organization.

Delete a Virtual Host

Before you can delete a virtual host from an environment, you must update any API proxies that reference the virtual host to remove the reference. See Virtual hosts for more.

Use the following API to delete a virtual host:

```
curl -u <admin user>:<admin passwd> -X DELETE \
"http://<ms-ip>:8080/v1/organizations/<org-name>/environments/<env-name>/virtua
lhosts/{virtualhost_name}"
```

Delete an Environment

You can only delete an environment after you have:

1. Deleted all virtual hosts in the environment as described above
2. Disassociated the environment from all Message Processors
3. Cleaned up analytics

Disassociate an Environment from Message Processor

Use the following API to remove an association of an environment with a Message Processor. If you want to delete the environment, you must disassociate it from all Message Processors:

```
curl -H "Content-Type: application/x-www-form-urlencoded" \
-u <admin user>:<admin passwd> -X POST \
"http://<ms-ip>:8080/v1/organizations/<org-name>/environments/<env-name>/server
s" \
-d "action=remove&uuid=<uuid>"
```

Where "`<uuid>`" is the UUID of Message Processor.

Note: To retrieve the UUID of the Message Processor, run the following command:

```
> curl http://<mp-ip>:8082/v1/servers/self
```

Where "< mp-ip>" is the IP address of the Message Processor.

Cleanup analytics

Remove analytics information about the organization:

```
curl -u <admin user>:<admin passwd> -X DELETE \
http://<ms-IP>:8080/v1/analytics/groups/ax/analytics/scopes?org=<org-name>&env=
<env-name>
```

Delete the environment

Use the following API to delete an environment:

```
curl -u <admin user>:<admin passwd> \
http://<ms-ip>:8080/v1/organizations/<org-name>/environments/<env-name> \
-X DELETE
```

Delete an Organization

You can only delete an organization after you have:

1. Deleted all virtual hosts in all environments in the organization as described above
2. Deleted all environments in the organization as described above
3. Disassociated the organization from all pods

Disassociate an Organization from Pod

Use the following API to delete disassociate an organization from a pod:

```
curl -H "Content-Type: application/x-www-form-urlencoded" \
-u <admin user>:<admin passwd> -X POST \
"http://<ms-ip>:8080/v1/organizations/<org-name>/pods" \
-d "action=remove&<region-name>=default&pod=<pod-name>"
```

Note: To find out what region and pods an organization is associated with, run the following command:

```
curl -u <admin user>:<admin passwd> \
"http://<ms-ip>:8080/v1/organizations/<org-name>/pods"
```

Delete the organization

Use the following API to delete an organization:

```
curl -u <admin user>:<admin passwd> -X DELETE \
"http://<ms-ip>:8080/v1/organizations/<org-name>"
```

Managing Users, Roles, and Permissions

The Apigee documentation site has extensive information on managing user roles and permissions. Users can be managed using both the Edge UI and the Management API; roles and permissions can be managed only with the Management API.

For information on users and creating users, see:

- About global users
- Creating global users

Many of the operations that you perform to manage users requires system administrator privileges. In a Cloud based installation of Edge, Apigee functions in the role of system administrator. In an Edge for the Private Cloud installation, your system administrator must perform these tasks as described below.

Adding a user

You can create a user either by using the Edge API, the Edge UI, or Edge commands. This section describes how to use Edge API and Edge commands. For information on creating users in the Edge UI, see Creating global users.

After you create the user in an organization, you must assign a role to the user. Roles determine the access rights of the user on Edge.

Note: The user cannot log in to the Edge UI, and does not appear in the list of users in the Edge UI, until you assign it to a role in an organization.

Use the following command to create a user with the Edge API:

```
curl -H "Content-Type:application/xml" -u <sysAdminEmail>:<passwd> \
-X POST http://<ms_IP>:8080/v1/users \
-d '<User> \
   <FirstName>New</FirstName> \
   <LastName>User</LastName> \
   <Password>newUserPWord</Password> \
   <EmailId>foo@bar.com</EmailId> \
</User>'
```

Or use the following Edge command to create a user:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-provision
create-user -f configFile
```

Where the ***configFile*** contain the information necessary to create the user:

```
APIGEE_ADMINPW=sysAdminPW      # If omitted, you will be prompted.
USER_NAME=foo@bar.com
FIRST_NAME=New
```

```
      LAST_NAME=User
      USER_PWD="newUserPWord"
      ORG_NAME=myorg  # lowercase only, no spaces, underscores, or periods.
```

You can then use this call to view information about the user:

```
curl -u <sysAdminEmail>:<passwd> \
http://<ms_IP>:8080/v1/users/foo@bar.com
```

Assigning the user to a role in an organization

Before a new user can do anything, they have to be assigned to a role in an organization. You can assign the user to different roles, including: `orgadmin`, `businessuser`, `opsadmin`, `user`, or to a custom role defined in the organization.

Assigning a user to a role in an organization automatically adds that user to the organization. Assign a user to multiple organizations by assigning them to a role in each organization.

Use the following command to assign the user to a role in an organization:

```
curl -X POST -H "Content-Type:application/x-www-form-urlencoded" /
-u <sysAdminEmail>:<passwd> /
http://<ms_IP>:8080/v1/o/<org_name>/userroles/<role>/users?id=foo@bar.com
```

You can view the user's roles by using the following command:

```
curl -u <sysAdminEmail>:<passwd>
http://<ms_IP>:8080/v1/users/foo@bar.com/userroles
```

To remove a user from an organization, remove all roles in that organization from the user. Use the following command to remove a role from a user:

```
curl -X DELETE -u <sysAdminEmail>:<passwd>
http://<ms_IP>:8080/v1/o/<org_name>/userroles/<role>/users/foo@bar.com
```

Adding a system administrator

A system administrator can:

- Create orgs
- Add Routers, Message Processors, and other components to an Edge installation
- Configure SSL
- Create additional system administrators
- Perform all Edge administrative tasks

While only a single user is the default user for administrative tasks, there can be more than one system administrator. Any user who is a member of the `sysadmin` role has full permissions to all resources.

Note: The `sysadmin` role is unique in that a user assigned to that role does not have to be part of an organization. However, you typically assign it to an organization, otherwise that user cannot log in to the Edge UI.

You can create the user for the system administrator in either the Edge UI or API. However, you must use the Edge API to assign the user to the role of `sysadmin`. Assigning a user to the `sysadmin` role cannot be done in the Edge UI.

To add a system administrator:

1. Create a user in the Edge UI or API.
2. Add user to `sysadmin` role:

   ```
   curl -u <sysAdminEmail>:<passwd> \
   -X POST http://<ms_IP>:8080/v1/userroles/sysadmin/users \
   -d 'id=foo@bar.com'
   ```

3. Make sure new user is in `sysadmin` role:

   ```
   curl -u <sysAdminEmail>:<passwd>
   http://<ms_IP>:8080/v1/userroles/sysadmin/users
   ```

   Returns the user's email address:

   ```
   [ " foo@bar.com " ]
   ```

4. Check permissions of new user:

   ```
   curl -u <sysAdminEmail>:<passwd>
   http://<ms_IP>:8080/v1/users/foo@bar.com/permissions
   ```

   Returns:

   ```
   {
     "resourcePermission" : [ {
     "path" : "/",
       "permissions" : [ "get", "put", "delete" ]
     } ]
   }
   ```

5. After you add the new system administrator, you can add the user to any orgs.

   Note: The new system administrator user cannot log in to the Edge UI until you add the user to at least one org.

6. If you later want to remove the user from the system administrator role, you can use the following API:

   ```
   curl -X DELETE -u <sysadminEmail:pword>
   http://<ms_IP>:8080/v1/userroles/sysadmin/users/foo@bar.com
   ```

   Note that this call only removes the user from the role, it does not delete the user.

Specifying the email domain of a system administrator

As an extra level of security, you can specify the required email domain of an Edge system administrator. When adding a system administrator, if the user's email address is not in the specified domain, then adding the user to the `sysadmin` role fails.

By default, the required domain is empty, meaning you can add any email address to the `sysadmin` role.

To set the email domain:

1. Open in an editor `management-server.properties:`

   ```
   vi /<inst_root>/apigee/customer/application/management-server.properties
   ```

   If this file does not exist, create it.

2. Set the `conf_security_rbac.global.roles.allowed.domains` property to the comma-separated list of allowed domains. For example:

   ```
   conf_security_rbac.global.roles.allowed.domains=myCo.com,yourCo.com
   ```

3. Save your changes.

4. Restart the Edge Management Server:

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-management-server restart
   ```

   If you now attempt to add a user to the `sysadmin` role, and the email address of the user is not in one of the specified domains, the add fails.

Deleting a user

You can create a user either by using the Edge API or the Edge UI. However, you can only delete a user by using the API.

To see the list of current users, including email address, use the following cURL command:

```
curl -u <sysAdminEmail>:<passwd> http://<ms-IP>:8080/v1/users
```

Use the following cURL command to delete a user:

```
curl -u <sysAdminEmail>:<passwd> \
-X DELETE http://<ms-IP>:8080/v1/users/<userEmail>
```

Backup and Restore

This section describes the backup and restore tasks in an on-premises installation of Apigee Edge. It is recommended that you should always create a backup of Apigee Edge components, i.e. configuration and data, at regular intervals and ensure that recovery is performed in an event of a system failure. Backup and restore procedures enable you to restore the state of an entire system (including all components), without affecting other parts of the system.

What to Backup

In an on-premises deployment of Apigee Edge, you must backup the following Edge components:

- Apache ZooKeeper (apigee-zookeeper)
- Apache Cassandra (apigee-cassandra)
- Postgres Server (edge-postgres-server)
- PostgreSQL database (apigee-postgresql)

  Note: In a Postgres Master/Standby configuration, you only backup the Master. You do not have to backup the slave.
- Qpid Server (edge-qpid-server)
- Qpidd (apigee-qpidd)
- OpenLDAP (apigee-openldap)
- Management Server (edge-management-server)
- Message Processor (edge-message-processor)
- Router (edge-router)
- Edge UI (edge-ui)

Recovery time objective (RTO) vs. Recovery point objective (RPO)

The RTO is the duration of time and a service level within which a business process must be restored after a disaster (or disruption) in order to avoid unacceptable consequences associated with a break in business continuity.

A RPO is the maximum tolerable period in which data might be lost from an IT service due to a major incident. Both objectives must be taken into consideration before you implement a backup plan for your recovery strategy.

Before You Start: Useful Facts

You may observe that installation data is distributed across several systems, for example organizations are in LDAP, ZooKeeper and Cassandra. Make sure that you take care of the following notes about backup and restore:

- If you have multiple Cassandra nodes, back them up one at a time. The backup process temporarily shuts down Cassandra, so you do not want to run it at the same time for all Cassandra nodes.
- If you have multiple ZooKeeper nodes, back them up one at a time. The backup process temporarily shuts down ZooKeeper.
- If you have multiple Postgres nodes, back them up one at a time.
- You can back up all other Edge components at the same time on all nodes by using tools such as Ansible or Chef.
- When you restore one of ZooKeeper, Cassandra or LDAP nodes it is recommended to restore all three nodes in order to achieve consistency (especially when organizations/environments have been created since backup was created).

  Note: The above does not affect restoration of one Cassandra or ZooKeeper node in a datastore cluster, since no backup is used.
- If LDAP or global administrator passwords are lost/corrupted, a complete backup is required in order to get the same credentials for the last backup and running system.
- All backup files, except for PostreSQL, are named in the form:

  ```
  backup-(year).(month).(day),(hour).(min).(seconds).tar.gz
  ```

  PostreSQL backup files are named:

  ```
  (year).(month).(day),(hour).(min).(seconds).dump
  ```

How to Perform a Backup

Use the following command to perform a backup:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service comp backup
```

where ***comp*** is the name of the component. For example:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra backup
```

Note: When backing up a complete Edge installation, you must back up all components. See What to Backup for more.

The backup command:

1. Stops the component (except for PostgreSQL which must be running to backup).

2. Creates a tar file of the following directories and files, where ***comp*** is the name of the component:
    a. Directories
        i. `/<inst_root>/apigee/data/`***comp***
        ii. `/<inst_root>/apigee/etc/`***comp***`.d`
    b. Files if they exists
        i. `/<inst_root>/apigee/token/application/`***comp***`.properties`
        ii. `/<inst_root>/apigee/customer/application/`***comp***`.properties`
        iii. `/<inst_root>/apigee/customer/defaults.sh`
        iv. `/<inst_root>/apigee/customer/conf/license.txt`
3. For all components except PostgreSQL, writes the tar file to `/<inst_root>/apigee/backup/`***comp***. The name is in the form:

    ```
    backup-(year).(month).(day),(hour).(min).(seconds).tar.gz
    ```

    For example:

    ```
    backup-2016.03.17,14.40.41.tar.gz
    ```

    For PostgreSQL, the file is named:

    ```
    (year).(month).(day),(hour).(min).(seconds).dump
    ```

4. Restarts the component (except for PostgreSQL which must be running to backup).

If you have multiple Edge components installed on the same node, you can back up them all with a single command:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-all backup
```

This command creates a backup file for each component on the node.

How to Perform a Restore

Use the following command to perform a restore:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service comp restore backupFile
```

where ***comp*** is the name of the component and ***backupFile*** is the name of the backup file. For example:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
restore backup-2016.03.17,14.40.41.tar.gz
```

The backup file is optional. If omitted, it uses the latest file in `/<inst_root>/apigee/backup/`***comp***.

Note: There is no "all" restore command. You must restore component individually.

The restore command:

1. Gets the latest backup file, if a filename was not passed in, or uses the specified backup file.
2. Checks to see if the directories are empty:

   a. `/<inst_root>/apigee/data/`***`comp`***

   b. `/<inst_root>/apigee/etc/`***`comp`***`.d`
3. If the above directories are not empty, restore stops and ask you to remove them.
4. Stops the component.
5. Restore the component from the backup.

   Note: You must explicitly restart the component after a restore.

How to Restore a Component to an Existing Environment

This section covers restoration of any Edge component to an existing environment without having to re-install the component. This means the node where you are restoring the component has the same IP address or DNS name as when you performed the backup.

If you have to re-install the component see How to Reinstall and Restore Components.

Apache ZooKeeper

Restore one standalone node

1) Remove old ZooKeeper directories:

```
/<inst_root>/apigee/data/apigee-zookeeper
/<inst_root>/apigee/etc/apigee-zookeeper.d
```

2) Restore ZooKeeper data from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-zookeeper
restore backup-2016.03.17,14.40.41.tar.gz
```

3) Restart all components to establish synchronization with the new restored ZooKeeper.

Restore one cluster node

1) If a single ZooKeeper node fails, that is part of an ensemble, you can create a new node with the same hostname/IP address (follow the re-install steps mentioned here) and when it joins the ZooKeeper ensemble it will get the latest snapshots from the Leader and start to serve clients. You do not need to restore data in this instance.

Restore a complete cluster

1) Stop the complete cluster.
2) Restore all ZooKeeper nodes from the backup file.
3) Start the ZooKeeper cluster.
4) Restart all components.

Apache Cassandra

Restore one standalone node

1) Remove old Cassandra directories:

   ```
   /<inst_root>/apigee/data/apigee-cassandra
   /<inst_root>/apigee/etc/apigee-cassandra.d
   ```

2) Restore the Cassandra node from the backup file:

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
   restore backup-2016.03.17,14.40.41.tar.gz
   ```

3) Restart all components.

Restore one cluster node

1) If a single Cassandra node fails, that is part of an ensemble, you can create a new node with the same hostname/IP address (follow the re-install steps mentioned here). You only need to re-install Cassandra, you do not need to restore the data.

   When performing a restore on a non-seed node, ensure that at least one Cassandra seed node is up.

   After installing Cassandra, and the node is up, (given that RF>=2 for all keyspaces) execute the following `nodetool` command to initialize the node:

   ```
   <inst_root>/apigee/apigee-cassandra/bin/nodetool -h localhost repair
   ```

Restore a complete cluster

1) Stop the complete cluster.
2) Restore all Cassandra nodes from the backup file.
3) Start the Cassandra cluster.
4) Restart all components.

PostgreSQL database

PosgreSQL running standalone or as Master

1) Stop the Management Server, Qpid Server, and Postgres Server on **all nodes**:

   Note: Your system can still handle requests to API proxies while these components are stopped.

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-management-server stop

   > /<inst_root>/apigee/apigee-service/bin/apigee-service edge-qpid-server
   stop

   > /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-postgres-server stop
   ```

2) Make sure PostgreSQL database is running:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
   status
   ```

3) Restore PostgreSQL database from the backup file:

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
   restore 2016.03.17,14.40.41.dump
   ```

4) Start the Management Server, Qpid Server, and Postgres Server on **all nodes**:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-management-server start

   > /<inst_root>/apigee/apigee-service/bin/apigee-service edge-qpid-server
   start

   > /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-postgres-server start
   ```

PosgreSQL running as Standby

1) Reconfigure PostgreSQL database using the same config file you used to install it:

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
   setup -f configFile
   ```

2) Start PostgreSQL:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
start
```

Postgres Server

1) Remove old Postgres Server directories:

```
/<inst_root>/apigee/data/edge-postgres-server
/<inst_root>/apigee/etc/edge-postgres-server.d
```

2) Restore Postgres Server from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-postgre-server restore backup-2016.03.17,14.40.41.tar.gz
```

3) Start Postgres Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-postgres-server start
```

Qpidd database

1) Remove old Qpidd directories:

```
/<inst_root>/apigee/data/apigee-qpidd
/<inst_root>/apigee/etc/apigee-qpidd.d
```

2) Restore Qpidd:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-qpidd
restore backup-2016.03.17,14.40.41.tar.gz
```

3) Start Qpidd:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-qpidd start
```

Qpid Server

1) Remove old Qpid Server directories:

```
/<inst_root>/apigee/data/edge-qpid-server
/<inst_root>/apigee/etc/edge-qpid-server.d
```

2) Restore Qpid Server from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-qpid-server
restore backup-2016.03.17,14.40.41.tar.gz
```

3) Start Qpid Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-qpid-server
start
```

OpenLDAP

1) Remove old OpenLDAP directories:

```
/<inst_root>/apigee/data/apigee-openldap
/<inst_root>/apigee/etc/apigee-openldap.d
```

2) Restore OpenLDAP from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-openldap
restore backup-2016.03.17,14.40.41.tar.gz
```

3) Restart OpenLDAP:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-openldap
start
```

Management Server

1) Remove old Management Server directories:

```
/<inst_root>/apigee/data/edge-management-server
/<inst_root>/apigee/etc/edge-management-server.d
```

2) Restore Management Server from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-management-server restore backup-2016.03.17,14.40.41.tar.gz
```

3) Restart the Management Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-management-server start
```

Message Processor

1) Remove old Message Processor directories:

```
/<inst_root>/apigee/data/edge-message-processor
/<inst_root>/apigee/etc/edge-message-processor.d
```

2) Restore Message Processor from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-
message-processor restore backup-2016.03.17,14.40.41.tar.gz
```

3) Restart Message Processor:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-message-processor start
```

Router

1) Remove old Router directories:

```
/<inst_root>/apigee/data/edge-router
/<inst_root>/apigee/etc/edge-router.d
```

2) Restore Router from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router
restore backup-2016.03.17,14.40.41.tar.gz
```

3) Restart Router:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router start
```

Edge UI

1) Remove old UI directories:

```
/<inst_root>/apigee/data/edge-ui
/<inst_root>/apigee/etc/edge-ui.d
```

2) Restore UI from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui restore
backup-2016.03.17,14.40.41.tar.gz
```

3) Restart UI:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui start
```

How to Reinstall and Restore Components

This section covers re-installation and restoration of an Edge component. Use this procedure if you have to re-install the Edge component before you restore the backup.

Note: These procedures all assume that you are re-installing the component onto a node with the same IP address or DNS name as the node that failed.

Apache ZooKeeper

Restore one standalone node

1) Stop ZooKeeper:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-zookeeper
stop
```

2) Remove old ZooKeeper directories:

```
/<inst_root>/apigee/data/apigee-zookeeper
/<inst_root>/apigee/etc/apigee-zookeeper.d
```

3) Re-install ZooKeeper:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-zookeeper
install
```

4) Restore ZooKeeper.

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-zookeeper
restore backup-2016.03.17,14.40.41.tar.gz
```

5) Restart all components:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-all restart
```

Restore one cluster node

If a single ZooKeeper node fails that is part of an ensemble, you can create a new node with the same hostname/IP address and re-install ZooKeeper. When the new ZooKeeper node joins the ZooKeeper ensemble it will get the latest snapshots from the Leader and start to serve clients. You do not need to restore data in this instance.

1) Re-install ZooKeeper:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-zookeeper
install
```

2) Run setup on the ZooKeeper node using the same config file used when installing the original node:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-zookeeper
setup -f configFile
```

3) Start ZooKeeper:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-zookeeper
start
```

Restore a complete cluster

1) Stop the complete cluster.
2) Restore all ZooKeeper nodes from the backup file as described above for a single node.
3) Start the ZooKeeper cluster.
4) Restart all components.

Apache Cassandra

Restore one standalone node

1) Stop Cassandra:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
stop
```

2) Remove old Cassandra directories:

```
/<inst_root>/apigee/data/apigee-cassandra
/<inst_root>/apigee/etc/apigee-cassandra.d
```

3) Re-install Cassandra:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
install
```

4) Restore Cassandra:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
restore backup-2016.03.17,14.40.41.tar.gz
```

5) Restart all components:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-all restart
```

Restore one cluster node

If a single Cassandra node fails, that is part of an ensemble, you can create a new node with the same hostname/IP address. You only need to re-install Cassandra, you do not need to restore the data.

Note: When performing a re-install on a non-seed node, ensure that at least one Cassandra seed node is up.

1) Re-install Cassandra:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
   install
   ```

2) Run setup on the Cassandra node using the same config file used when installing the original node:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee- cassandra
   setup -f configFile
   ```

3) Start Cassandra:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee- cassandra
   start
   ```

4) After installing Cassandra, and the node is up, (given that RF>=2 for all keyspaces) execute the following `nodetool` command to initialize the node:

   ```
   <inst_root>/apigee/apigee-cassandra/bin/nodetool -h localhost repair
   ```

Restore a complete cluster

1. Stop the complete cluster.
2. Restore all Cassandra nodes from the backup file.
3. Start the Cassandra cluster.
4. Restart all components.

PostgreSQL database

PosgreSQL running standalone or as Master

1) Stop the Management Server, Qpid Server, and Postgres Server on **all nodes**:

   Note: Your system can still handle requests to API proxies while these components are stopped.

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-management-server stop

   > /<inst_root>/apigee/apigee-service/bin/apigee-service edge-qpid-server
   stop

   > /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-postgres-server stop
   ```

2) Re-install PostgreSQL database:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
install
```

3) Start PostgreSQL:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
start
```

4) Restore PostgreSQL database from the backup file:

```
/<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
restore 2016.03.17,14.40.41.dump
```

5) Start the Management Server, Qpid Server, and Postgres Server on **all nodes**:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-management-server start

> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-qpid-server
start

> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-postgres-server start
```

PosgreSQL running as Standby

1) Re-install PostgreSQL database:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
install
```

2) Reconfigure PostgreSQL database using the same config file you used to install it:

```
/<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
setup -f configFile
```

3) Start PostgreSQL:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-postgresql
start
```

Postgres Server

1) Stop Postgres Server on all master and standby nodes:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-postgres-server stop
```

2) Remove old Postgres Server directories:

```
/<inst_root>/apigee/data/edge-postgres-server
/<inst_root>/apigee/etc/edge-postgres-server.d
```

3) Re-install Postgres Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-postgres-server install
```

4) Restore Postgres Server from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-postgre-server restore backup-2016.03.17,14.40.41.tar.gz
```

5) Start Postgres Server on all master and standby nodes:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-postgres-server start
```

Qpid Server and Qpidd

1) Stop Qpidd, Qpid Server, and Postgres Server on all nodes:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-qpid-server
stop

> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-postgres-server stop

> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-qpidd stop
```

2) Remove old Qpid Server and Qpidd directories:

```
/<inst_root>/apigee/data/edge-qpid-server
/<inst_root>/apigee/etc/edge-qpid-server.d
/<inst_root>/apigee/data/apigee-qpidd
/<inst_root>/apigee/etc/apigee-qpidd.d
```

3) Re-install Qpidd:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-qpidd
install
```

4) Restore Qpidd:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-qpidd
restore backup-2016.03.17,14.40.41.tar.gz
```

5) Start Qpidd:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-qpidd start
```

6) Re-install Qpid Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-qpid-server
install
```

7) Restore Qpid Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-qpid-server
restore backup-2016.03.17,14.40.41.tar.gz
```

8) Restart Qpid Server, Qpidd, and Postgres Servers on all nodes:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-qpidd
restart

> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-qpid-server
restart

> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-postgres-server restart
```

OpenLDAP

1) Stop OpenLDAP:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-openldap
stop
```

2) Re-install OpenLDAP:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-openldap
install
```

3) Remove old OpenLDAP directories:

```
/<inst_root>/apigee/data/apigee-openldap
/<inst_root>/apigee/etc/apigee-openldap.d
```

4) Restore OpenLDAP:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-openldap
restore backup-2016.03.17,14.40.41.tar.gz
```

5) Restart OpenLDAP:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-openldap
start
```

6) Restart all Management Servers:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-management-server restart
```

Management Server

1) Stop Management Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-management-server stop
```

2) Remove old Management Server directories:

```
/<inst_root>/apigee/data/edge-management-server
/<inst_root>/apigee/etc/edge-management-server.d
```

3) Re-install Management Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-management-server install
```

4) Restore Management Server from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-management-server restore backup-2016.03.17,14.40.41.tar.gz
```

5) Restart Management Server:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-management-server start
```

Message Processor

1) Stop Message Processor:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-message-processor stop
```

2) Remove old Message Processor directories:

```
/<inst_root>/apigee/data/edge-message-processor
/<inst_root>/apigee/etc/edge-message-processor.d
```

3) Re-install Message Processor:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-message-processor install
```

4) Restore Message Processor from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-message-processor restore backup-2016.03.17,14.40.41.tar.gz
```

5) Restart Message Processor:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service
edge-message-processor start
```

Router

1) Stop Router:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router stop
```

2) Remove old Router directories:

```
/<inst_root>/apigee/data/edge-router
/<inst_root>/apigee/etc/edge-router.d
```

3) Re-install Router:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router
install
```

4) Restore Router from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router
restore backup-2016.03.17,14.40.41.tar.gz
```

5) Restart Router:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router start
```

Edge UI

1) Stop UI:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui stop
```

2) Remove old UI directories:

```
/<inst_root>/apigee/data/edge-ui
/<inst_root>/apigee/etc/edge-ui.d
```

3) Re-install UI:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui install
```

4) Restore UI from the backup file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui restore
backup-2016.03.17,14.40.41.tar.gz
```

5) Restart UI:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-ui start
```

Complete Site Recovery

Restore a complete site

1) Stop all component nodes. Note that the order of stopping the subsystems is important - first all Edge nodes and then all datastores nodes.
2) Restore all components as described above.
3) Now start all components in the following order. Note that the order of starting the subsystems is important:
    a) Start the ZooKeeper cluster
    b) Start the Cassandra cluster
    c) Ensure that OpenLDAP is up and running
    d) Start qpid
    e) Ensure that the PostgreSQL database is up and running
    f) Start Management Server
    g) Start Routers and Message Processors
    h) Start Qpid Server
    i) Start Postgres Server
    j) Start Apigee UI

Monitoring

This section provides an introduction to the monitoring tasks that you should perform at specific intervals to maintain your on-premise installation. Monitoring helps you to perform audits, troubleshoot and diagnose problems.

Before monitoring your Apigee Edge for Private Cloud, you should have a good working knowledge of Apigee Edge. You must also be familiar with overall organizational components of Apigee Edge and their functional differences.

For information on overview and installing Apigee Edge for Private Cloud, see http://docs.apigee.com/api-services/latest/installing-edge-private-cloud.

What to Monitor

Generally in a production setup, there is a need to enable monitoring mechanisms within an Apigee Edge for Private Cloud deployment. These monitoring techniques warn the network administrators (or operators) of an error or a failure. Every error generated is reported as an alert in Apigee Edge. For more information on alerts, see Monitoring Best Practices.

For ease, Apigee components are classified mainly into two categories:

- **Apigee-specific Java Server Services** – These include Management Server, Message Processor, Qpid Server, and Postgres Server.
- **Third-party Services** – These include Nginx Router, Apache Cassandra, Apache ZooKeeper, OpenLDAP, PostgresSQL database, and Qpid.

In an on-premise deployment of Apigee Edge, the following table provides a quick glance into the parameters that you can monitor:

Component		**System Checks**	**Process-Level Stats**	**API-Level Checks**	**Message Flow Checks**	**Component Specific**
Apigee-specific Java Services	Management Server	☑	☑	☑		
	Message Processor	☑	☑	☑	☑	
	Qpid Server	☑	☑	☑		

	Postgres Server	☑	☑	☑		
Third-party Services	Apache Cassandra	☑				☑
	Apache ZooKeeper	☑				☑
	OpenLDAP	☑				☑
	PostgreSQL database	☑				☑
	Qpid	☑				☑
	Nginx Router	☑	☑		☑	

In general, after Apigee Edge has been installed, you can perform the following monitoring tasks to track the performance of an Apigee Edge for Private Cloud installation.

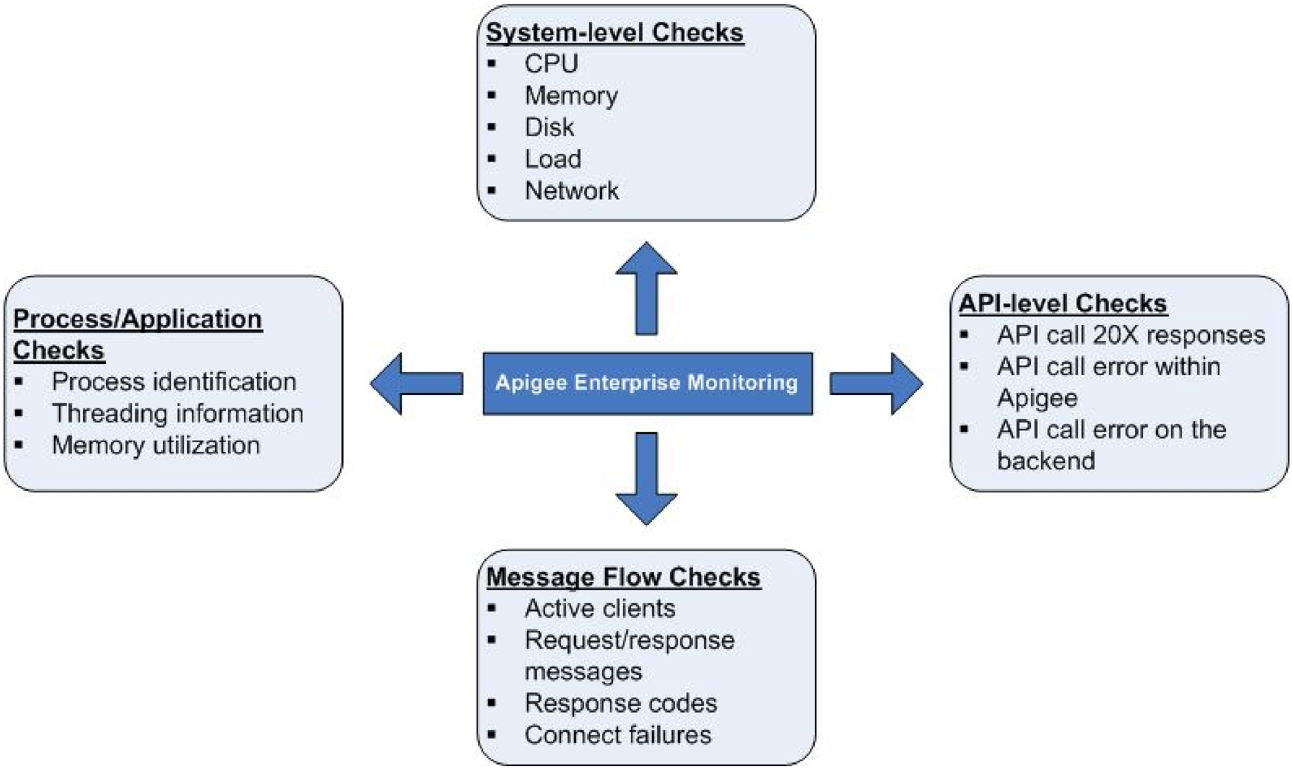


Figure 2: Apigee Edge for Private Cloud - Monitoring Overview

System health checks

It is very important to measure the system health parameters such as CPU utilization, memory utilization and port connectivity at a higher level. You can monitor the following parameters to get the basics of system health.

- **CPU Utilization** – Specifies the basic statistics (User/System/IO Wait/Idle) about the CPU utilization. For example, total CPU used by the system.

- **Free/Used Memory** – Specifies the system memory utilization as bytes. For example, physical memory used by the system.
- **Disk Space Usage** – Specifies the file system information based on the current disk usage. For example, hard disk space used by the system.
- **Load Average** – Specifies the number of processes waiting to run.
- **Network Statistics** – Network packets and/or bytes transmitted and received, along with the transmission errors about a specified component.

Processes/Application checks

At the process level, you can view important information about all the processes that are running. For example, these include memory and CPU usage statistics that a process or application utilizes. For processes like qpidd, postgres postmaster, java and so on, you can monitor the following:

- **Process identification**: Identify a particular Apigee process. For example, you can monitor for the existence of an Apigee server java process.
- **Thread statistics**: View the underlying threading patterns that a process uses. For example, you can monitor peak thread count, thread count for all the processes.
- **Memory utilization**: View the memory usage for all the Apigee processes. For example, you can monitor the parameters like heap memory usage, non-heap memory usage used by a process.

API-level checks

At the API level, you can monitor whether server is up and running for frequently used API calls proxied by Apigee. For example, you can perform API check on the Management Server, Router, and Message Processor by invoking the following cURL command:

```
curl http://<host>:<port>/v1/servers/self/up
```

Where <`host`> is the IP address of the Apigee Edge component. The `<port>` number is specific to each Edge component. For example:

- Management Server: 8080
- Router: 8081
- Message Processor: 8082
- etc.

See the individual sections below for information on running this command for each component

This call returns the "true" and "false". For best results, you can also issue API calls directly on the backend (with which Apigee software interacts) in order to quickly determine whether an error exists within the Apigee software environment or on the backend.

Note: To monitor your API proxies, you can also use Apigee's API Health. API Health makes scheduled calls to your API proxies and notifies you when they fail and how. When calls succeed, API Health shows you response times and can even notify you when response latency is high. API Health can make calls from different locations around the world to compare API behavior between regions.

Message flow checks

You can collect data from Routers and Message Processors about message flow pattern/statistics. This allows you to monitor the following:

- Number of active clients
- Number of responses (10X, 20X, 30X, 40X and 50X)
- Connect failures

This helps you to provide dashboards for the API message flow.

How to Monitor

This section describes the monitoring techniques of components supported by an on-premise deployment of Apigee Edge.

Enabling JMX authentication and setting the JMX password

The monitoring process for the Management Server, Message Processor, Qpid, and Postgres all use JMX. JMX is enabled by default and remote JMX access does not require a password.

To enable JMX authentication, each component has a `change_jmx_auth` action that you use to enable/disable authentication and to set the JMX credentials.

To enable JMX authentication, use the following command:

```
>  /<inst_root>/apigee/apigee-service/bin/apigee-service comp change_jmx_auth
optionsOrConfigFile
```

where:

- ***comp*** is either `edge-management-server`, `edge-message-processor`, `edge-router`, `edge-qpid-server`, or `edge-postgres-server`.
- Options are:
    - o `-u`: username
    - o `-p`: password
    - o `-e`: y (enable) or n (dsiable)
- Config file includes:
    - o JMX_USERNAME=username
    - o JMX_ENABLED=y/n
    - o JMX_PASSWORD=password (if not set or not passed in with -p, you are prompted)

For example, to use options on the command line:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-management-server
change_jmx_auth -u foo -p bar -e y
```

If you have a config file:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-management-server
change_jmx_auth -f configFile
```

If you are running Edge on multiple nodes, run this command on all nodes, specifying the same username and password.

To later disable JMX authentication, use the command:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service edge-management-server
change_jmx_auth -e n
```

Management Server

Using JConsole – monitor system health check and process information

Use JConsole (a JMX compliant tool) to manage and monitor health check and process statistics. Using JConsole, you can consume JMX statistics exposed by Management Server (or any server) and display them in a graphical interface. For more information on JConsole usage, see http://docs.oracle.com/javase/8/docs/technotes/guides/management/jconsole.html

Use **JConsole** and the following service URL to monitor the JMX attributes (MBeans) offered via JMX.

```
service:jmx:rmi:///jndi/rmi://<ip address>:<port>/platform
```

where <`ip address`> is the IP address of Management Server (or respective server). By default the port is 1099 for the Management Server.

Generic JMX statistics

JMX MBeans	JMX Attributes
Memory	HeapMemoryUsage
	NonHeapMemoryUsage
	Usage
Note: Attribute values will be displayed in four values: committed, init, max, and used.	

Using Edge Application – API checks

You can perform API check on the Management Server (or any server) by invoking the following CURL command:

```
curl http://<host>:8080/v1/servers/self/up
```

Where, host is the IP address of Management Server.

This call returns the "true" and "false". If true, that means node is up and Java service is running.

If you do not receive a HTTP 200 (OK) response, the Edge is unable to respond to port 8080 requests.

Troubleshooting

1. Login to the server and run the following command:

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-management-server status
   ```

2. If the service is not running start the service

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-management-server start
   ```

Using Edge Application – Users, organization and deployment checks

Management Server plays a vital role in holding all other parcels together in each on-premises installation. You can check for user, organization and deployment status on the management server by issuing the following commands:

```
curl -u userEmail:password http://localhost:8080/v1/users

curl -u userEmail:password http://localhost:8080/v1/organizations

curl -u userEmail:password
http://localhost:8080/v1/organizations/orgname/deployments
```

The system should display "deployed" status for all calls. If these fail, do the following:

1. Check the Management Server logs (at `<inst_root>/apigee/var/log/edge-management-server`) for any errors.
2. Make a call against Management Server to check whether it is functioning properly.
3. Remove the server from the ELB and then restart the Management Server.

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-service
   edge-management-server restart
   ```

Router

You can perform API check on the Router (or any server) by invoking the following CURL command:

```
curl http://<host>:8081/v1/servers/self/up
```

Where, host is the IP address of Router.

This call returns the "true" and "false". If true, that means node is up and Java service is running.

If you do not receive a HTTP 200 (OK) response, Edge is unable to respond to port 8081 requests.

Troubleshooting

1. Login to the server and run the following commands:

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router status
   ```

2. If the service is not running start the service

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router start
   ```

3. If the service is running, test that it is functioning. You monitor the cluster status by checking the memberCount against the reachableCount and alert all the instances with 'memberCount != reachableCount'

   ```
   curl -v -u <userEmail>:<password> http://localhost:port/v1/cluster
   ```

   Where port – 8081 for Router and 8082 for Message Processor.

   The output of above CURL command is shown below.

   ```
   {
    "memberCount" : 12,
    "pod" : "rea1gw001",
    "reachableCount" : 12,
    "region" : "us-east-1",
    "types" : [ "management-server" ]
   * Connection #0 to host ms05apigee left intact
   * Closing connection #0
   }
   ```

4. If it is not functioning, run the following command to examine the failure or find the offending member.

   ```
   curl http://localhost:port/v1/cluster/members
   ```

   Where port – 8081 for Router and 8082 for Message Processor.

   The output of above CURL command will like this

   ```
   {
     "lastChange" : 0,
     "latency" : 0,
     "state" : "CONNECTED",
     "uuid" : "9c4c8bde-0015-4dc5-82d2-59fb326c4074"
   }, {
     "address" : "/192.168.5.209:4526",
     "clusterType" : "router,message-processor",
   ```

```
    "lastChange" : 1350658037228,
    "latency" : 3,
    "pod" : "rea1gw001",
    "region" : "us-east-1",
    "serverType" : "message-processor",
    "state" : "CONNECTED",
    "uuid" : "f1c663a1-2bb8-469f-b5fd-69a5c5aa91c5"
  }, {
    "address" : "/192.168.5.29:4526",
    "clusterType" : "router,message-processor",
    "lastChange" : 1350623005057,
    "latency" : 1,
    "pod" : "rea1gw001",
    "region" : "us-east-1",
    "serverType" : "message-processor",
    "state" : " DISCONNECTED ",
    "uuid" : "4cfe932b-f644-4581-b1ae-df338af9c7ce"
  }, {
    "address" : "/192.168.4.182:4526",
    "clusterType" : "router,message-processor",
    "lastChange" : 1350657730535,
    "latency" : 1,
    "pod" : "rea1gw001",
    "region" : "us-east-1",
    "serverType" : "message-processor",
    "state" : "CONNECTED",
    "uuid" : "cba063d5-b8a4-409f-9e0b-f5d403e02091"
  }
```

5. Notice that the IP address 192.168.5.29 is disconnected. Restart the server

   ```
   /<inst_root>/apigee/apigee-service/bin/apigee-service edge-router restart
   ```

Note: If a Router has a disconnected state, remove the router from the ELB and then restart it.

6. After restart check that it is functioning

```
curl -v http://localhost:port/v1/cluster
```

Where port – 8081 for Router and 8082 for Message Processor.

Message Processor

Using JConsole – monitor system health check and process information

Follow the same as described for Management Server.

Note: Ensure that you use port – 1101.

Using Edge Application – API checks

Follow the same as described for Router.

Note: Ensure that you use port – 8082.

Using JMX – Message flow checks

Follow the same as described for Management Server.

Note: Ensure that you use port – 1101.

Qpid Server

Using JConsole – monitor system health check and process information

Follow the same as described for Management Server.

Note: Ensure that you use port – 1102.

Using Edge Application – API checks

Follow the same as described for Management Server.

Note: Ensure that you use port – 8083. The following CURL command is also supported for Qpid Server:

```
curl http://<qpid_IP>:8083/v1/servers/self
```

Postgres Server

Using JConsole – monitor system health check and process information

Follow the same as described for Management Server.

Note: Ensure that you use port – 1103.

Using Edge Application – API checks

Follow the same as described for Management Server.

Note: Ensure that you use port – 8084. The following CURL command is also supported for Postgres Server:

```
curl http://<postgres_IP>:8084/v1/servers/self
```

Using Edge Application – organization and environment checks

You can check for organization and environment name that are onboarded on the Postgres Server by issuing the following CURL commands:

```
curl http:// <postgres_IP>:8084/v1/servers/self/organizations
```

Note: Ensure that you use port – 8084.

The system should display the organization and environment name.

Using Edge Application – axstatus check

You can verify the status of the analytics servers by issuing the following CURL command.

```
curl -u userEmail:password
http://<host>:<port>/v1/organizations/<orgname>/environments/<envname>/pro
visioning/axstatus
```

The system should display SUCCESS status for all analytics servers. The output of above CURL command is shown below:

```
{
"environments" : [ {
"components" : [ {
"message" : "success at Thu Feb 28 10:27:38 CET 2013",
"name" : "pg",
"status" : "SUCCESS",
"uuid" : "[c678d16c-7990-4a5a-ae19-a99f925fcb93]"
}, {
"message" : "success at Thu Feb 28 10:29:03 CET 2013",
"name" : "qs",
"status" : "SUCCESS",
"uuid" : "[ee9f0db7-a9d3-4d21-96c5-1a15b0bf0adf]"
} ],
"message" : "",
"name" : "prod"
} ],
"organization" : "acme",
"status" : "SUCCESS"
}
```

PostgresSQL Database

Using Script – check_postgres.pl

To monitor the PostgresSQL database, you can use a standard monitoring script, *check_postgres.pl* which is available at http://bucardo.org/wiki/Check_postgres.

Note: The script, *check_postgres.pl* needs to be installed in each Postgres node.

Before you run the script:

1. Ensure that you have installed `perl-Time-HiRes.x86_64`, a Perl module that implements high resolution alarm, sleep, gettimeofday, and interval timers. For example, you can install it by using the following command:

   ```
   yum install perl-Time-HiRes.x86_64
   ```

The default output of the API calls using the script, *check_postgres.pl* is Nagios compatible. After you install the script, do the following checks:

1. Database size – check the database size:

   ```
   check_postgres.pl -H 10.176.218.202 -db apigee -u apigee -dbpass postgres
   -include=apigee -action database_size --warning='800 GB' --critical='900
   GB'
   ```

2. Incoming connection to the database – checks the number of incoming connections to the database and compares with maximum allowed connections:

   ```
   check_postgres.pl -H 10.176.218.202 -db apigee -u apigee -dbpass postgres
   -action backends
   ```

3. Database availability and performance – checks if database is running and available:

   ```
   check_postgres.pl -H 10.176.218.202 -db apigee -u apigee -dbpass postgres
   -action connection
   ```

4. Disk space – checks the disk space:

   ```
   check_postgres.pl -H 10.176.218.202 -db apigee -u apigee -dbpass postgres
   -action disk_space --warning='80%' --critical='90%'
   ```

5. Onboarded organizations/environments – checks the number of organization and environment onboarded in a Postgres node:

   ```
   check_postgres.pl -H 10.176.218.202 -db apigee -u apigee -dbpass postgres
   -action=custom_query --query="select count(*) as result from pg_tables
   where schemaname='analytics' and tablename like '%fact'" --warning='80'
   --critical='90' --valtype=integer
   ```

Note: Please refer to the http://bucardo.org/check_postgres/check_postgres.pl.html in case you need any help on using the above commands.

DB Checks

You can verify that the proper tables are created in PostgresSQL database.

Login to PostgreSQL database using:

```
psql -U apigee -d apigee
```

and then run:

```
\d analytics."<org>.<env>.fact"
```

API check – health status of postgres process

You can perform API check on the postgres machine by invoking the following CURL command:

```
http://<postgres_IP>:8084/v1/servers/self/health/
```

Note: Ensure that you use port 8084.

It returns the 'ACTIVE' status when postgres process is active. If the postgres process is not up and running, it returns the 'INACTIVE' status.

Postgres Resources

http://www.postgresql.org/docs/9.0/static/monitoring.html

http://www.postgresql.org/docs/9.0/static/diskusage.html

http://bucardo.org/check_postgres/check_postgres.pl.html

Apache Cassandra

Using JConsole – monitor task statistics

Use **JConsole** and the following service URL to monitor the JMX attributes (MBeans) offered via JMX.

```
service:jmx:rmi:///jndi/rmi://<ip address>:7199/jmxrmi
```

where <`ip address`> is the IP of the Cassandra server.

JMX is enabled by default for Cassandra and remote JMX access to Cassandra does not require a password.

To enable JMX authentication to add a password:

1. Edit `/<inst_root>/apigee/customer/application/cassandra.properties`. If the file does not exist, create it.
2. Add the following to the file:

   ```
   conf_cassandra-env_com.sun.management.jmxremote.authenticate=true
   ```
3. Save the file.
4. Copy the following files from your `$JAVA_HOME` directory to `/<inst_root>/apigee/data/apigee-cassandra/`:

```
cp ${JAVA_HOME}/lib/management/jmxremote.password.template
$APIGEE_ROOT/data/apigee-cassandra/jmxremote.password

cp ${JAVA_HOME}/lib/management/jmxremote.access
$APIGEE_ROOT/data/apigee-cassandra/jmxremote.access
```

5. Edit `jmxremote.password` and add username and password to the file:

   ```
   cassandra password
   ```

   where ***password*** is the JMX password.

6. Edit `jmxremote.access` and add the following role:

   ```
   cassandra     readwrite
   ```

7. Make sure the files are owned by "apigee" and that the file mode is 400:

   ```
   > chown apigee:apigee
   /<inst_root>/apigee/data/apigee-cassandra/jmxremote.*
   > chmod 400 /<inst_root>/apigee/data/apigee-cassandra/jmxremote.*
   ```

8. Run `configure` on Cassandra:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
   configure
   ```

9. Restart Cassandra:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
   restart
   ```

To later disable authentication:

1. Edit `/<inst_root>/apigee/customer/application/cassandra.properties`.
2. Remove the following line in the file:

   ```
   conf_cassandra-env_com.sun.management.jmxremote.authenticate=true
   ```

3. Run `configure` on Cassandra:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
   configure
   ```

4. Restart Cassandra:

   ```
   > /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-cassandra
   ```

```
    restart
```

Cassandra JMX statistics

JMX MBeans	JMX Attributes
ColumnFamilies/apprepo/environments ColumnFamilies/apprepo/organizations ColumnFamilies/apprepo/apiproxy_revisions ColumnFamilies/apprepo/apiproxies ColumnFamilies/audit/audits ColumnFamilies/audit/audits_ref	PendingTasks
	MemtableColumnsCount
	MemtableDataSize
	ReadCount
	RecentReadLatencyMicros
	TotalReadLatencyMicros
	WriteCount
	RecentWriteLatencyMicros
	TotalWriteLatencyMicros
	TotalDiskSpaceUsed
	LiveDiskSpaceUsed
	LiveSSTableCount
	BloomFilterFalsePositives
	RecentBloomFilterFalseRatio
	BloomFilterFalseRatio

Using nodetool utility – manage cluster nodes

The nodetool utility, which is a command line interface for Cassandra, is used to manage cluster nodes. The utility can be found at `<inst_root>/apigee/apigee-cassandra/bin`.

For more info on nodetool utility, see http://www.datastax.com/docs/1.0/references/nodetool

The following calls can be made on all Cassandra cluster nodes:

1. **General ring info** (also possible for single Cassandra node): Look for the "Up" and "Normal" for all nodes.

   ```
   [host]# nodetool -h localhost ring
   ```

   The output of the above command looks as shown below:

   ```
   Address         DC      Rack     Status State   Load       Owns    Token

   113427455640312821154458202477256070484
   ```

```
192.168.124.201 dc1     ra1      Up     Normal  1.67 MB    33,33%  0
192.168.124.202 dc1     ra1      Up     Normal  1.68 MB    33,33%
 56713727820156410577229101238628035242
192.168.124.203 dc1     ra1      Up     Normal  1.67 MB    33,33%
 113427455640312821154458202477256070484
```

2. **General info about nodes** (call per node)

   ```
   nodetool -h localhost info
   ```

   The output of the above command looks as shown below:

   ```
   Token             : 0
   Gossip active     : true
   Load              : 1.67 MB
   Generation No     : 1361968765
   Uptime (seconds) : 78108
   Heap Memory (MB) : 46,80 / 772,00
   Data Center       : dc1
   Rack              : ra1
   Exceptions        : 0
   ```

3. **Status of the thrift server** (serving client API)

   ```
   host]# nodetool -h localhost statusthrift
   ```

   The output of the above command displays status as "running".

4. **Status of data streaming operations:** Observe traffic for cassandra nodes

   ```
   nodetool -h localhost netstats 192.168.124.203
   ```

   The output of the above command looks as shown below:

   ```
   Mode: NORMAL
   Nothing streaming to /192.168.124.203
   Nothing streaming from /192.168.124.203
   Pool Name          Active   Pending      Completed
   Commands              n/a         0           1688
   Responses             n/a         0         292277
   ```

Cassandra Monitoring (UI)

Refer to the datastax opscenter URL: http://www.datastax.com/products/opscenter

Cassandra Resource

Refer to the following URL: http://www.datastax.com/docs/1.0/operations/monitoring

Apache ZooKeeper

Checking ZooKeeper status

1. Ensure the ZooKeeper process is running. ZooKeeper writes a PID file to `<inst_root>/apigee/var/run/apigee-zookeeper/apigee-zookeeper.pid`.
2. Test ZooKeeper ports to ensure that you can establish a TCP connection to ports 2181 and 3888 on every ZooKeeper server.
3. Ensure that you can read values from the ZooKeeper database. Connect using a ZooKeeper client library (or `/<inst_root>/apigee/apigee-zookeeper/bin/zkCli.sh`) and read a value from the database.
4. Check the status:

```
> /<inst_root>/apigee/apigee-service/bin/apigee-service apigee-zookeeper
status
```

Using ZooKeeper Four Letter Words

ZooKeeper can be monitored via a small set of commands (four-letter words) that are sent to the port 2181 using netcat (nc) or telnet.

For more info on ZooKeeper commands, see:
http://zookeeper.apache.org/doc/r3.1.2/zookeeperAdmin.html#sc_zkCommands

For example:

- `srvr`: Lists full details for the server.
- `stat`: Lists brief details for the server and connected clients.

The following commands can be issued to the ZooKeeper port:

1. Run the four-letter command `ruok` to test if server is running in a non-error state. A successful response returns "imok".

```
echo ruok | nc <host> 2181

imok
```

2. Run the four-letter command, `stat` to list server performance and connected clients statistics.

```
echo stat | nc <host> 2181

Zookeeper version: 3.4.5-1392090, built on 09/30/2012 17:52 GMT

Clients:

/0:0:0:0:0:0:0:1:33467[0](queued=0,recved=1,sent=0)

/192.168.124.201:42388[1](queued=0,recved=8433,sent=8433)
```

```
/192.168.124.202:42185[1](queued=0,recved=1339,sent=1347)

/192.168.124.204:39296[1](queued=0,recved=7688,sent=7692)

Latency min/avg/max: 0/0/128

Received: 26144

Sent: 26160

     Connections: 4

     Outstanding: 0

     Zxid: 0x2000002c2

     Mode: follower

     Node count: 283
```

Note: It is sometimes important to see whether a ZooKeeper is in Mode: leader, follower or observer.

3. If netcat (nc) is not available, you can use the python as an alternative. Create a file named zookeeper.py that contains the following:

```
import time, socket,
sys c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

c.connect((sys.argv[1], 2181))

c.send(sys.argv[2])

time.sleep(0.1)

print c.recv(512)
```

Now run the following python lines:

```
python zookeeper.py 192.168.124.201 ruok

python zookeeper.py 192.168.124.201 stat
```

OpenLDAP

LDAP Level Test

You can monitor the OpenLDAP to see whether the specific requests are served properly. In other words, check for a specific search that returns the right result.

1. Use `ldapsearch` (`yum install openldap-clients`) to query the entry of the system admin. This entry is used to authenticate all API calls.

```
ldapsearch -b "uid=admin,ou=users,ou=global,dc=apigee,dc=com" -x -W -D
"cn=manager,dc=apigee,dc=com" -H ldap://localhost:10389 -LLL
```

You are then prompted for the LDAP admin password:

```
Enter LDAP Password:
```

After entering the password, you see a response in the form:

```
dn: uid=admin,ou=users,ou=global,dc=apigee,dc=com

objectClass: organizationalPerson

objectClass: person

objectClass: inetOrgPerson

objectClass: top

uid: admin

cn: admin

sn: admin

userPassword::
e1NTSEF9bS9xbS9RbVNXSFFtUWVsU1F0c3BGL3BQMkhObFp2eDFKUytmZVE9PQ=
 =

mail: opdk@apigee.com
```

2. Check whether Management Server is still connected to LDAP issue

```
curl -u <userEMail>:<password> http://localhost:8080/v1/users/<ADMIN>
```

Returns:

```
{

 "emailId" : <ADMIN>,

 "firstName" : "admin",

 "lastName" : "admin"

}
```

You can also monitor the OpenLDAP caches, which help in reducing the number of disk accesses and hence improve the performance of the system. Monitoring and then tuning the cache size in the OpenLDAP server can heavily impact the performance of the directory server. You can view the log files (`<inst_root>/apigee/var/log`) to obtain information about cache.

Monitoring Best Practices

Monitoring Alerts

Apigee Edge allows you to forward alerts to syslogs or external monitoring systems/tools when an error or a failure occurs due to failure of an event. These alerts can be system-level or application-level alerts/events. Application level alerts are mostly custom alerts that are created based on events generated. The network administrator usually configures the custom conditions. For more information on alerts, contact Apigee Support.

Setting Alert Thresholds

Set a threshold after which an alert needs to be generated. What you set depends on your hardware configuration? Threshold should be set in relation to your capacity. For example, Apigee Edge might be too low if you only have 6GB capacity. You can assign threshold with equal to (=) or greater than (>) criterion. You can also specify a time interval between two consecutive alerts generation. You can use the hours/minutes/seconds option.

Criteria for Setting System-level Alerts

Alert	Suggested Threshold	Description
Low memory	500MB	Memory is too low to start a component
Low disk space (/var/log)	8GB	Disk space has fallen too low.
High load	3+	Processes waiting to run have increased unexpectedly
Process stopped	N/A, a Boolean value of true or false	Apigee Java process in the system has stopped

Note: For disk space, monitor all mounted partitions.

Checking on Apigee-specific and Third-party Ports

Monitor the following ports to make sure they're active

- Port 4526, 4527 and 4528 on Management Server, Router and Message Processor
- Port 1099, 1100 and 1101 on Management Server, Router and Message Processor
- Port 8081 on Routers
- Port 8082 and 8998 on Message Processors

Port 8080 on Management Server Check the following third-party ports to make sure they're active:

- Qpid port 5672
- Postgres port 5432
- Cassandra port 7000, 7199, 9042, 9160
- ZooKeeper port 2181
- OpenLDAP port 10389

In order to determine which port each Apigee component is listening for API calls on, issue the following API calls to the Management Server (which is generally on port 8080):

```
curl -v -u <username>:<password>
http://<host>:<port>/v1/servers?pod=gateway&region=dc-1

curl -v -u <username>:<password> http://
<host>:<port>/v1/servers?pod=central&region=dc-1

curl -v -u <username>:<password> http://
<host>:<port>/v1/servers?pod=analytics&region=dc-1
```

The output of these commands will contain sections similar to that shown below. The "`http.management.port`" section gives the port number for the specified component.

```
{
  "externalHostName" : "localhost",
  "externalIP" : "111.222.333.444",
  "internalHostName" : "localhost",
  "internalIP" : "111.222.333.444",
  "isUp" : true,
  "pod" : "gateway",
  "reachable" : true,
  "region" : "default",
  "tags" : {
    "property" : [ {
      "name" : "Profile",
      "value" : "Router"
    }, {
      "name" : "rpc.port",
      "value" : "4527"
    }, {
      "name" : "http.management.port",
      "value" : "8081"
    }, {
      "name" : "jmx.rmi.port",
      "value" : "1100"
    } ]
  },
  "type" : [ "router" ],
  "uUID" : "2d4ec885-e20a-4173-ae87-10be38b35750"
}
```

Viewing Logs

Log files keep track of messages regarding the event/operation of the system. Messages appear in the log when processes begin and complete or when an error condition occurs. By viewing log files, you can obtain information about system components, for example, CPU, memory, disk, load, processes, so on, before and after attaining a failed state. This also allows you to identify and diagnose the source of current system problems or help you predict potential system problems.

For example, a typical system log of a component contains following entries as seen below:

```
TimeStamp = 25/01/13 19:25 ; NextDelay = 30
Memory
HeapMemoryUsage = {used = 29086176}{max = 64880640} ;
NonHeapMemoryUsage = {init = 24313856}{committed = 57278464} ;
Threading
PeakThreadCount = 53 ; ThreadCount = 53 ;
OperatingSystem
SystemLoadAverage = 0.25 ;
```

You can edit the `/<inst_root>/apigee/conf/logback.xml` file to control the logging mechanism without having to restart a server. The logback.xml file contains the following property that sets the frequency that the logging mechanism checks the logback.xml file for configuration changes:

```
<configuration scan="true" scanPeriod="30 seconds" >
```

By default, the logging mechanism checks for changes every minute. If you omit the time units to the `scanPeriod` attribute, it defaults to milliseconds.

Note: Setting the `scanPeriod` attribute to a short interval might affect system performance.

The following table tells the log files location of Apigee Edge Private Cloud components.

Components	Location
Management Server	<inst_root>/apigee/var/log/edge-management-server
Router	<inst_root>/apigee/var/log/edge-router
Message Processor	<inst_root>/apigee/var/log/edge-message-processor
Qpid Server	<inst_root>/apigee/var/log/apigee-qpid-server
Apigee Postgres Server	<inst_root>/apigee/var/log/apigee-postgres-server
Edge UI	<inst_root>/apigee/var/log/edge-ui
ZooKeeper	<inst_root>/apigee/var/log/apigee-zookeeper
OpenLDAP	<inst_root>/apigee/var/log/apigee-openldap
Cassandra	<inst_root>/apigee/var/log/apigee-cassandra
Qpidd	<inst_root>/apigee/var/log/apigee-qpidd
PostgreSQL database	<inst_root>/apigee/var/log/apigee-postgresql

Enabling debug logs for the Message Processor and Edge UI

To enable debug logs for Message Processor:

1. On the Message Processor node, edit `/<install_dir>/apigee/customer/application/messsage-processor.properties`. If that file does not exist, create it.
2. Add the following property to the file:

   ```
   conf_system_log.level=DEBUG
   ```

3. Restart the Message Processor:

   ```
   > /<install_dir>/apigee/apigee-service/bin/apigee-service
   edge-message-processor restart
   ```

To enable debug logs for Edge UI:

4. On the Edge UI node, edit `/<install_dir>/apigee/customer/application/ui.properties`. If that file does not exist, create it.
5. Add the following property to the file:

   ```
   conf_application_logger.application=DEBUG
   ```

6. Restart the Edge UI:

```
> /<install_dir>/apigee/apigee-service/bin/apigee-service edge-ui restart
```

Monitoring Tools

The open-source monitoring tools such as Nagios, Collectd, Graphite, Splunk, Sumologic, and Monit can help you monitor your entire enterprise environment and business processes.

Component		Nagios	Collectd	Splunk
System-level checks	CPU utilization	☑	☑	
	Free/used memory	☑	☑	
	Disk space usage	☑	☑	
	Network statistics	☑	☑	
Processes		☑		
API checks		☑		
JMX		☑		
Java			☑	
Log files				☑
Critical events	Rate Limit hit			☑
	Backend server (Hybris or SharePoint) cannot be reached			☑
	FaaS (STS) cannot be reached			☑
Warning events	SMTP server cannot be reached			☑
	SLA’s violated			☑

Managing LDAP Resources

When using the LDAP policy for authentication or DN queries (http://apigee.com/docs/api-services/content/ldap-policy), the policy uses an Apigee-side LDAP resource that contains the connection details to your LDAP provider. This section describes how to create and manage LDAP resources via an API.

To find the docs that correspond to earlier versions of the product, see http://apigee.com/docs/content/documentation-archives.

Create an LDAP resource

Following is the API for creating an LDAP resource:

```
/v1/organizations/{org_name}/environments/{environment}/ldapresources
```

Following is an annotated XML payload that describes the LDAP resource configuration you'll send to create the resource:

```
<LdapResource name="ldap1">
  <Connection>
    <Hosts>
      <Host port="636">foo.com</Host> <!-- port is optional: defaults to 389 for ldap:// and
636 for ldaps:// -->
    </Hosts>
    <SSLEnabled>false</SSLEnabled> <!-- optional, defaults to false -->
    <Version>3</Version> <!-- optional, defaults to 3-->
    <Authentication>simple</Authentication> <!-- optional, only simple supported -->
    <ConnectionProvider>jndi|unboundid</ConnectionProvider> <!-- required -->
    <ServerSetType>single|round robin|failover</ServerSetType> <!-- not applicable for jndi
-->
    <LdapConnectorClass>com.custom.ldap.MyProvider</LdapConnectorClass> <!-- If using a
custom LDAP provider, the fully qualified class -->
  </Connection>
  <ConnectPool enabled="true"> <!-- enabled is optional, defaults to true -->
    <Timeout>30000</Timeout> <!-- optional, in milliseconds; if not set, no timeout -->
    <Maxsize>50</Maxsize> <!-- optional; if not set, no max connections -->
    <Prefsize>30</Prefsize> <!-- optional; if not set, no pref size -->
    <Initsize></Initsize> <!-- optional; if not set, defaults to 1 -->
    <Protocol></Protocol> <!-- optional; if not set, defaults to 'ssl plain' -->
  </ConnectPool>
  <Admin>
    <DN>cn=admin,dc=apigee,dc=com</DN>
    <Password>secret</Password>
  </Admin>
</LdapResource>
```

Example

The following example creates an LDAP resource named **ldap1**.

```
curl -X POST -H "Content-Type: application/xml" \
https://api.enterprise.apigee.com/v1/organizations/myorg/environments/test/ldap
resources \
-u {apigee_email}:{password} -d \
'<LdapResource name="ldap1">
  <Connection>
    <Hosts>
      <Host>foo.com</Host>
    </Hosts>
    <SSLEnabled>false</SSLEnabled>
    <Version>3</Version>
    <Authentication>simple</Authentication>
    <ConnectionProvider>unboundid</ConnectionProvider>
    <ServerSetType>round robin</ServerSetType>
  </Connection>
  <ConnectPool enabled="true">
    <Timeout>30000</Timeout>
    <Maxsize>50</Maxsize>
    <Prefsize>30</Prefsize>
    <Initsize></Initsize>
    <Protocol></Protocol>
  </ConnectPool>
  <Admin>
    <DN>cn=admin,dc=apigee,dc=com</DN>
    <Password>secret</Password>
  </Admin>
</LdapResource>'
```

List all LDAP Resources

```
curl
https://api.enterprise.apigee.com/v1/organizations/myorg/environments/test/ldap
resources \
-u {apigee_email}:{password}
```

Get the Details of an LDAP Resource

```
curl
https://api.enterprise.apigee.com/v1/organizations/myorg/environments/test/ldap
```

```
resources/ldap1 \
-u {apigee_email}:{password}
```

Update an LDAP resource

```
curl -X POST -H "Content-Type: application/xml" \
https://api.enterprise.apigee.com/v1/organizations/myorg/environments/test/ldap
resources/ldap1 \
-u {apigee_email}:{password} -d \
'<LdapResource name="ldap1">
  <Connection>
    <Hosts>
      <Host>foo.com</Host>
    </Hosts>
    <SSLEnabled>false</SSLEnabled>
    <Version>3</Version>
    <Authentication>simple</Authentication>
    <ConnectionProvider>unboundid</ConnectionProvider>
    <ServerSetType>round robin</ServerSetType>
  </Connection>
  <ConnectPool enabled="true">
    <Timeout>50000</Timeout>
    <Maxsize>50</Maxsize>
    <Prefsize>30</Prefsize>
    <Initsize></Initsize>
    <Protocol></Protocol>
  </ConnectPool>
  <Admin>
    <DN>cn=admin,dc=apigee,dc=com</DN>
    <Password>secret</Password>
  </Admin>
</LdapResource>'
```

Delete an LDAP Resource

```
curl -X DELETE \
https://api.enterprise.apigee.com/v1/organizations/myorg/environments/test/ldap
resources/ldap1 \
-u {apigee_email}:{password}
```

10 Almaden Boulevard, 16th Floor, San Jose
CA 95113
USA

No. 17/2, 2B Cross, 7th Main,
2 & 3 Floor, Off 80 Feet Road, 3rd Block
Koramangala, Bangalore 560034
INDIA

3 Sheldon Square
London W2 6HY
UK

www.apigee.com