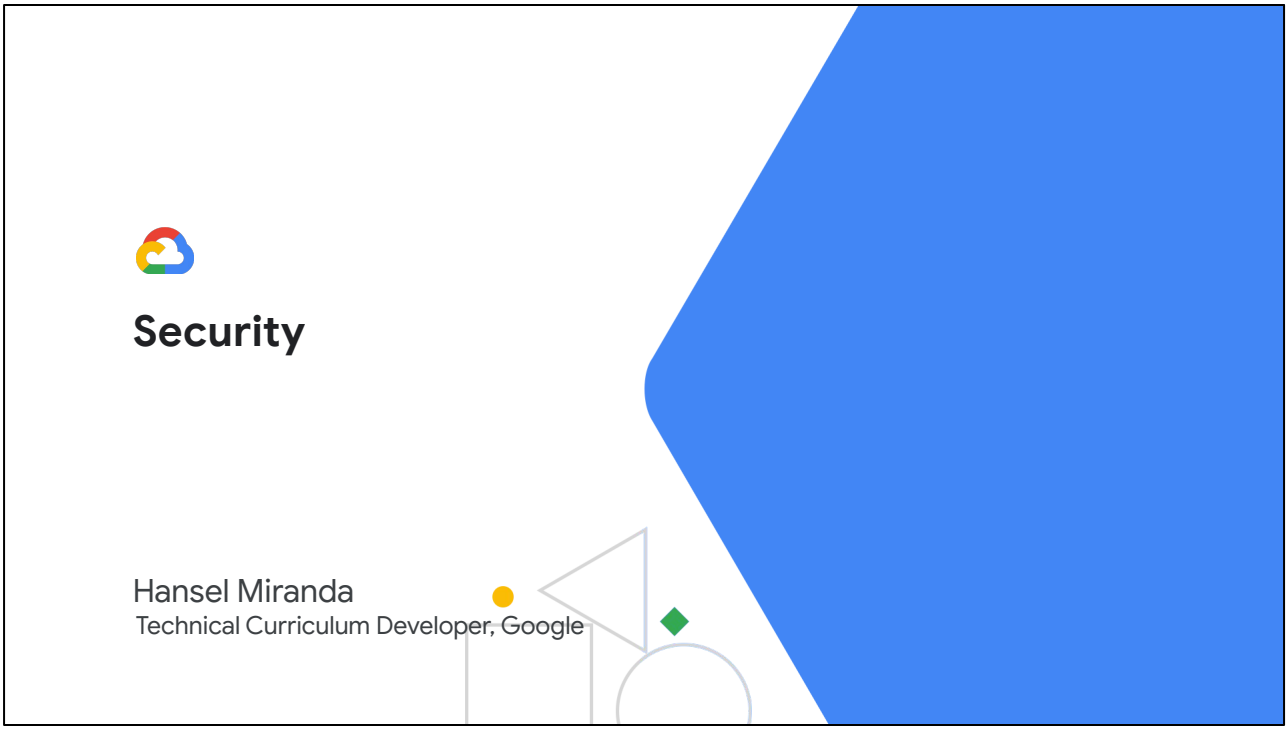


Welcome to the security module of the On Premises Management, Security, and Upgrade course.

Security is one of the key features of the Apigee API platform for private cloud.

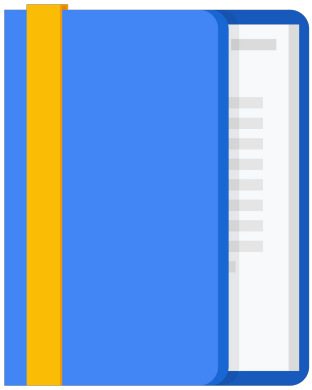
Agenda

TLS overview

Configuring TLS

Apigee mTLS

SSO integration



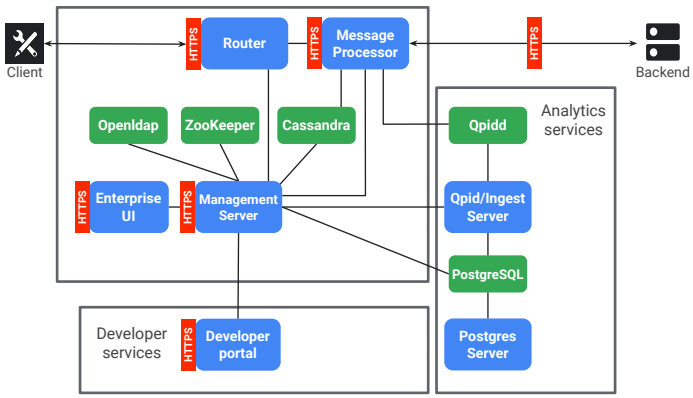
In this module, we discuss how to configure TLS or transport layer security in Apigee Edge for private cloud.

We will take you through the process of configuring TLS on the router and message processor components, and the process to secure the management server and enterprise UI using TLS.

You will also learn how to configure mutual TLS on the Apigee private cloud platform, and how to configure Apigee Edge to use an identity provider for authentication and single sign on.

Let’s get started first with a brief introduction to TLS.

Overview



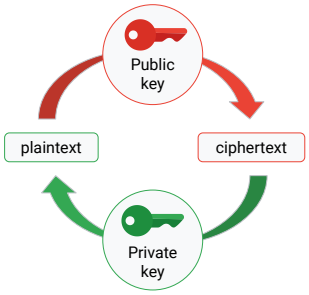
This lesson will cover an introduction to TLS and introduce the basic concepts needed to prepare to implement TLS on the components used in Apigee Edge.

This module refers to certificates, keys and keystores - let's discuss the basics of creating each of these elements.

Public key cryptography

There are two types of keys:

- Public keys are publicly shared.
- Private keys are used to decrypt messages encrypted with the public key.



Public Key Cryptography allows messages to be encrypted and sent securely between parties.

It consist of two types of keys, public and private, which are asymmetric or different.

The keys are generated in pairs and are mathematically linked.

Digital certificate

Attributes of the digital certificate:

- Common Name
- Validity Start Date
- Validity End Date
- Subject
- Issuer
- Serial Number
- ...

Sample certificate

CN=*.google.com

Validity

Not Before: Tue, 22 Jun 2021 13:37:09 UTC
Not After: Tue, 14 Sep 2021 13:37:08 UTC

Subject: C=US, ST=California, L=Mountain View, O=Google LLC
Issuer: C=US, O=Google Trust Services, CN=GTS CA 1O1

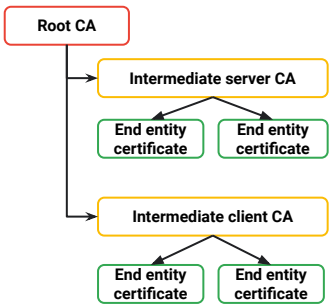
Serial Number: 39252e20e18cd90c0a00000000e8323f

A digital certificate is a file that contains a cryptographic public key and additional attributes.

The public key combined with the attributes serve as a digital identity of a person, service or an entity.

Certificate authority

- A certificate authority (CA) is an entity that issues digital certificates.
- The CA acts as a third party trusted by both the owner of the certificate and the client relying upon the certificate.



The format of a digital certificate is specified by the X.509 standard.

A root CA certificate is the base certificate that signs and issues one or more intermediate CA certificates with varying validation requirements.

Intermediate certificates in turn sign and issue certificates to end-entities like individuals, services or things.

Certificate Authorities maintain a publicly accessible certificate revocation status list of revoked certificates in either OCSP or CRL formats.

Transport Layer Security (TLS)

- TLS is the standard security technology for secure messaging across your API environment.
- Apigee supports one-way TLS and two-way TLS in both cloud and on-premises deployments.

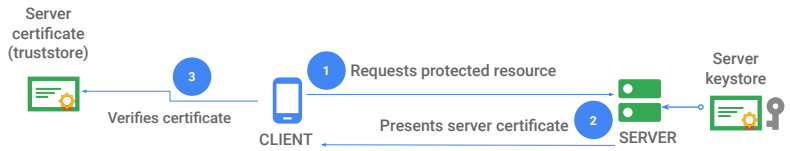
TLS is the leading standard for secure communications.

Apigee supports one-way TLS and two-way TLS in both cloud and on-premises deployments.

Various Apigee components support native TLS which can be enabled with additional configuration.

Transport Layer Security (TLS)

- One-way TLS enables the client to verify the identity of the server.



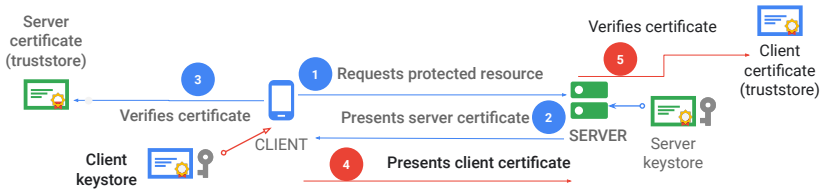
One-way TLS enables a TLS client to verify the identity of the TLS server.

For example, a client app running on your phone can verify the identity of the Apigee Edge gateway serving your APIs.

The client application sends the initial request to the server. The server then sends it the server certificate. The client is then able to verify the server credentials based on that certificate.

Transport Layer Security (TLS)

- Two-way TLS enables the client to verify the identity of the server and enables the server to verify the identity of the client.



With two-way TLS, the client application verifies the server application identity as in one-way TLS. In addition, the server also verifies the identity of the client application.

Two-way TLS is also often called client or mutual TLS because the client identifies itself once the TLS server application has authenticated itself to the client.

In this process, a client requests access to a protected resource. The server then presents its certificate to the client. The client verifies the server certificate. If successful, the client then sends its certificate to the server. The server then verifies the client's credentials. If successful, the server grants access to the resource requested by the client.

Private key

- Private keys are used when creating certificate requests.
- They should be secured properly and not shared with anyone.

Private keys are used to generate certificate signing requests which are used for certificate generation.

It is extremely important to keep the private key properly secured.

Any compromise of a private key invalidates the security measures and protocols where the key is used.

Private key

How to create a key

On Linux, use openssl to create a private key:

- Create a directory to store the private key:

```
mkdir -p ~/certs/servers/certdemo.yourdomain.xyz
chmod 700 -R ~/certs
```

- Create and secure the key:

```
openssl genrsa -out "certs/servers/certdemo.yourdomain.xyz/private.pem" 2048
chmod 600 -R certs/servers/certdemo.yourdomain.xyz/private.pem
```

You can use the OpenSSL tool to create a private key on linux systems.

First create a directory on the server and set the minimal access permissions on the directory path that will contain your key.

Then, run the openssl genrsa command that creates an RSA key providing the path and filename where the key is stored and the length of the key.

Finally, set the appropriate access permissions to secure the key file.

Certificate signing request (CSR)

A CSR is used to request your public certificate from a certificate authority.

Use the openssl tool and the private key to create a CSR:

```
openssl req -new -key
certs/servers/certdemo.yourdomain.xyz/private.pem -out
certs/tmp/certdemo.yourdomain.xyz.csr.pem -subj
"/C=AU/ST=NSW/L=Sydney/O=Cert
Demo/CN=prodapi.certdemo.yourdomain.xyz"
```

After you have generated a private key you create a certificate signing request or CSR.

To create a CSR, use OpenSSL with the private key generated earlier.

The command sets the action to create a new certificate signing request, provides the path to the private key and the location of the output file that will contain the CSR.

You must also provide information about the subject of the certificate that includes the country, state, locality, organization and the common name of the certificate.

This command creates a certificate signing request that can be sent to your certificate authority to issue a trusted certificate.

Java keystores

- Use openssl to create a keystore:

```
openssl pkcs12 -export -clcerts -in
certs/signed/prodapi.certdemo.yourdomain.xyz.cert.pem
-inkey certs/servers/certdemo.yourdomain.xyz/private.pem
-out /tmp/prodapi_keystore.pkcs12 -name prodapi
```

- Use the Java keytool utility to convert to JKS format:

```
keytool -importkeystore -srckeystore
/tmp/prodapi_keystore.pkcs12 -srcstoretype pkcs12
-destkeystore /tmp/prodapi_keystore.jks -deststoretype
jks -alias prodapi
```

In Apigee for private cloud, you mostly use Java Keystores when securing your applications.

When you create a keystore, it is a good practice to give the keystore an easily identifiable alias.

We again use the OpenSSL tool and private key, as well as our issued certificate.

To create the keystore, use the openssl command with the keystore type and provide the export action, the type of certificates to be stored, and the path to the certificate.

You also provide the private key, a path to the exported keystore and an alias by which to reference it.

Once the keystore has been created, you then use the Java keytool utility to convert it to JKS format.

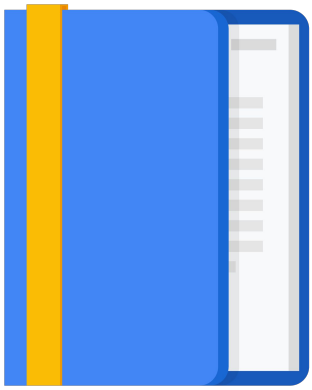
Agenda

TLS overview

Configuring TLS

Apigee mTLS

SSO integration



In this lesson, we discuss how TLS is configured on the components used in the Apigee private cloud platform.

Configuring TLS on a Router

Overview 1/2

To enable TLS for incoming API requests, you must configure the virtual host associated with an environment to accept encrypted connections.

TLS encryption can be terminated at the Apigee Edge Router or at the load balancer that is closest to the data center boundary.

To configure the router component for TLS, you create a new secure virtual host that is configured to accept encrypted connections for each environment that you need to secure.

You create TLS keys and certificates and store them in keystores using the Apigee management API.

Keys and certificates must be first bundled into a specially-formatted JAR file and then uploaded to the platform.

Once uploaded, the keystore and key name are referenced in the virtual host definition.

Configuring TLS on a Router

Overview 2/2

A keystore contains one or more JAR files, where the JAR file contains a:

- TLS certificate as a PEM file
- Private key as a PEM file

Creating the keystore is a two-step process:

You first create a JAR file containing the certificate and private key.

Then, create the keystore and upload the JAR file using the management API.

Configuring TLS on a Router

Creating a keystore: create the JAR

JAR file must contain:

- Certificate (or chain if including CA/Intermediate CA) - mycert.pem
- Private key - myKey.pem
- Manifest - META-INF/descriptor.properties

Package using JAR:

- jar -cf myKeystore.jar myCert.pem myKey.pem
- jar -uf myKeystore.jar META-INF/descriptor.properties

To create the JAR file, you need a certificate, or a full certificate chain if you're including the CA or intermediate CA. In the sample command, this file is named myCert.pem.

You also need the private key file. In the sample command, this file is named myKey.pem.

And, you need a manifest file. To create one, first create the META-INF directory and touch or create a file named descriptor.properties.

To package the JAR, you follow a two-step process. First, add the certificates and private key using the JAR command. The sample command creates a jar file named myKeystore.jar.

Next, we update the JAR to add in the descriptor.properties manifest file.

Your JAR file is now ready to be uploaded to a keystore on the Apigee Edge platform. Note that only one certificate or chain is allowed per JAR file.

Apigee for private cloud supports self-signed certificates and key lengths of up to 2048 bits.

Configuring TLS on a Router

Creating a keystore: upload the JAR

- Create the keystore:

```
curl -H "Content-Type: text/xml" \
https://api.enterprise.apigee.com/v1/o/{org_name}/environments/{env_name}/keystores \
-X POST -d '<KeyStore name="myKeystore"/>' -u email
```

- Upload the jar file:

```
curl -X POST -H "Content-Type: multipart/form-data" -F file="@myKeystore.jar" \
"https://api.enterprise.apigee.com/v1/o/{org_name}/e/{env_name}/keystores/myKeystore/k
eys?alias={key_alias}&password={key_pass}" -u email
```

To create the keystore, use the /keystores management API.

The API is invoked with a POST method and the URL that contains the organization and environment name where the keystore is being created.

The payload contains an XML fragment containing the keystore name.

Once the keystore is created, it can be populated by uploading the JAR file that was created in the previous step.

A POST call to the same management API is used to upload the JAR file to the named keystore endpoint.

Query parameters are also provided that specify an alias for the keystore and the private key passphrase.

Configuring TLS on a Router

Create a secure virtual host

```
curl -X POST -H "Content-Type:application/xml" -u email
http://<ms-IP>:8080/v1/o/{org}/e/{env}/virtualhosts
-d '<VirtualHost  name="securevhost">
      <HostAliases>
        <HostAlias>tlsdemo.training.apigee.com</HostAlias>
      </HostAliases>
      <Interfaces/>
      <Port>9005</Port>
      <SSLInfo>
        <Enabled>true</Enabled>
        <ClientAuthEnabled>false</ClientAuthEnabled>
        <KeyStore>myKeystore</KeyStore>
        <KeyAlias>myKeyAlias</KeyAlias>
      </SSLInfo>
  </VirtualHost>'
```

The keystore reference is then added to the configuration when creating or updating a secure virtual host.

To configure the virtual host, use the /virtualhosts management API and provide the payload.

The payload specifies the host alias or DNS name and port that the virtual host will listen on. SSL is enabled and the keystore and keyAlias is set.

Once this API executes successfully, the virtual host is configured and the Apigee Edge routers are updated.

Configuring TLS on the Message Processor

Overview

- Enable TLS between the Router and Message Processor.

To secure internal communication between the Router and Message Processor components, you must enable TLS on the Message Processor.

Configuring TLS on the Message Processor

Preparation and checks

From the Router:

```
curl http://<message_processor_ip>:8082/v1/servers/self/uuid
```

To configure TLS on the message processor, you must open port 8082 between each router and message processor in each region.

To test connectivity, use the /servers management API between each router and message processor.

The API should return the UUID for each message processor.

Configuring TLS on the Message Processor

Preparation and checks

- Example with output:

```
java -cp \
/opt/apigee/edge-gateway/lib/thirdparty/jetty-http-9.4.0.v20161208.jar:/opt/apigee
/edge-gateway/lib/thirdparty/jetty-util-9.4.0.v20161208.jar
org.eclipse.jetty.http.security.Password TestMe
```

- Output:

TestMe

```
OBF:1ppq1od31yta1ytc1obr1pqo
MD5:2c539f7b23fc3fb3b8ec52d3bfa58834
```

Some sections of the TLS configuration procedures require you to enter an obfuscated password in a configuration file.

Using an obfuscated password is a more secure alternative to a password in plain text.

You can generate an obfuscated password in Java using the jetty utilities that are installed with Apigee for private cloud.

Note that the actual jar file versions may vary based on the installed version of Apigee.

Configuring TLS on the Message Processor

Create and secure the Java keystore

- Store the keystore file:

```
cp message_processor_keystore.jks /opt/apigee/customer/application/mp_keystore.jks
```

- Secure the keystore:

```
chown apigee:apigee /opt/apigee/customer/application/mp_keystore.jks
chmod 600 /opt/apigee/customer/application/mp_keystore.jks
```

Next, create the keystore as previously described using the private key for the server and an appropriate certificate.

Copy the keystore to a location on the message processor that is accessible by the Apigee user.

Make sure to secure the keystore file with appropriate permissions so that it is accessible only by the Apigee user.

Configuring TLS on the Message Processor

Configure the Message Processor

Append the following properties to the file:

`/opt/apigee/customer/application/message-processor.properties:`

Properties:

```
conf_message-processor-communication_local.http.ssl=true
conf/message-processor-communication.properties+local.http.port=8443
conf/message-processor-communication.properties+local.http.ssl.keystore.type=jks
conf/message-processor-communication.properties+local.http.ssl.keystore.path=/opt
/apigee/customer/application/keyStore.jks
conf/message-processor-communication.properties+local.http.ssl.keyalias=apigee-de
vtest

# Enter the obfuscated keystore password below.
conf/message-processor-communication.properties+local.http.ssl.keystore.password=
OBF:obsPword
```

After creating and securing the keystore file, and creating an obfuscated password, you now configure the Message Processor to use TLS.

The configuration properties shown are appended to the message-processor.properties file in the customer application directory.

If the file does not already exist, then it should be created.

Stepping through the configuration properties, first set the SSL property to true. Next, specify the port that the message processor should run on, the keystore type as JKS, and the full path to the keystore.

The key alias is set to the same value as when the keystore was created. The obfuscated version of the keystore password is also configured in the properties file.

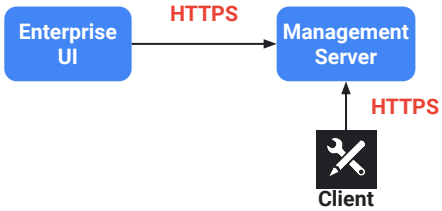
This configuration should be applied to all message processors.

Once completed, a rolling restart of the routers and message processors is required.

Configuring TLS on the Management Server

Overview

- Client apps such as curl or Postman can use HTTPS to make secure API management calls.
- The Apigee enterprise UI can be configured to use HTTPS to improve security.



TLS can be configured on the management server to permit access over HTTPS.

By default, TLS is disabled on the management server. Management API calls are made over HTTP using the management server IP address on port 8080.

It is also possible to configure the enterprise UI component to make management API calls only over HTTPS (HTTP is the default).

Configuring TLS on the Management Server

Preparation and checks

- Example with output:

```
java -cp \
/opt/apigee/edge-gateway/lib/thirdparty/jetty-http-9.4.0.v20161208.jar:/opt/apigee
/edge-gateway/lib/thirdparty/jetty-util-9.4.0.v20161208.jar
org.eclipse.jetty.http.security.Password TestMe
```

- Output:

TestMe

```
OBF:1ppq1od31yta1ytc1obr1pqo
MD5:2c539f7b23fc3fb3b8ec52d3bfa58834
```

The first step to configure TLS on the management server is to create an obfuscated password.

You create an obfuscated password in Java by using the Jetty .jar files that are installed with Apigee for private cloud.

Configuring TLS on the Management Server

Enable TLS

Append the following properties to the file:
`/opt/apigee/customer/application/management-server.properties`

Properties:

```
conf_webserver_ssl.enabled=true

# http.turn.off=true if all comms should be over HTTPS.
# Not recommended as many Edge internal calls use HTTP.
conf_webserver_http.turn.off=false
conf_webserver_ssl.port=8443
conf_webserver_keystore.path=/opt/apigee/customer/application/keystore.jks

# Enter the obfuscated keystore password below.
conf_webserver_keystore.password=OBF:obfuscatedPassword
conf_webserver_cert.alias=apigee-devtest
```

Configuring TLS for the management server follows the same pattern as for the message processor.

First, create a JKS keystore containing the server private key and a signed certificate or certificate chain.

Copy that keystore to a directory accessible by the Apigee user. It is recommended to use the customer application directory to keep all custom configuration in one place.

Next, edit or create the management server properties file in the customer application directory to add the configuration properties.

The properties enable SSL, set the SSL port to 8443, sets the full path to the keystore and the obfuscated password, and sets the certificate alias.

Once the configuration is complete, restart the management server using the apigee service utility.

If the configuration was successful, the management API should now be accessible over HTTPS.

Configuring TLS on the Management Server

Test

```
curl https://mapi.certdemo.yourdomain.xyz:8443/v1/o -u <sysadmin@email.xyz>

[ "VALIDATE", "tisdemo" ]
```

Note: If using your own CA, you may have to append the following code to the curl command:

```
--cacert /path/to/ca/chain/cert
```

To test accessing the management server over HTTPS, invoke the /organizations API on port 8443.

The API should return a list of configured organizations for your Apigee private cloud installation.

Configuring TLS on the Enterprise UI

Overview

- Web browser or client to Enterprise UI
- Enterprise UI to Management Server

By default TLS is disabled on the Apigee enterprise UI and you access it via HTTP on port 9000.

You can enable TLS on the UI and access it over HTTPS.

TLS can also be configured on the management server to permit access by the UI over HTTPS.

And, you can configure the UI component to make management API calls over HTTPS instead of over HTTP which is the default.

Configuring TLS on the Enterprise UI

Create and secure the Java keystore

- Create keystore using openssl and Java keytool.
- Store the keystore file:

```
cp ui_keystore.jks /opt/apigee/customer/application/ui_keystore.jks
```

- Secure the keystore:

```
chown apigee:apigee /opt/apigee/customer/application/ui_keystore.jks
chmod 600 /opt/apigee/customer/application/ui_keystore.jks
```

Create the keystore as previously described using the private key for the server and an appropriate certificate.

Copy the keystore to a location accessible by the Apigee user, and secure the keystore with appropriate file permissions so it can only be accessed by the Apigee user.

Configuring TLS on the Enterprise UI

Configuring TLS using the apigee-service utility

- Configure the component:

```
apigee-service edge-ui configure-ssl

Checking for optional variables
Configuring HTTPS for UI services
Enter UI HTTPS port: 9443
Do you want to disable HTTP y/n (y): n
Enter SSL keystore algorithm (JKS): JKS
Enter SSL keystore absolute file path: /opt/apigee/customer/application/ui_keystore.jks
Enter SSL keystore password: TestMe
Enter SSL keystore password for verification: TestMe
```

- Restart the component:

```
apigee-service edge-ui restart
```

TLS configuration for the Edge UI is done via the apigee-service utility.

You run the utility passing in the edge ui component name and the configure ssl options.

The utility first runs some internal checks and then prompts you through the configuration process.

First, you set the port for HTTPS. Next, you are prompted to optionally disable HTTP access.

The keystore algorithm is set to JKS and the full path to the keystore file is provided on the next set of prompts.

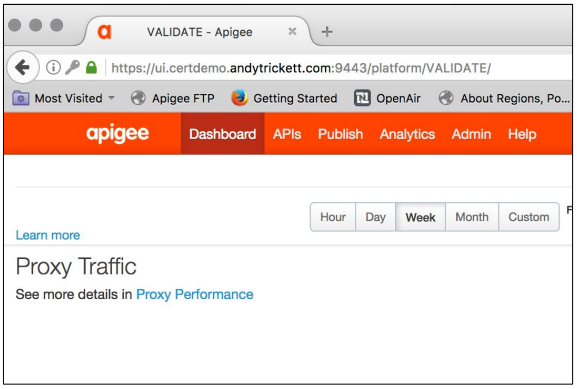
Finally, you enter and confirm the keystore password. At this point, configuration is complete.

To implement the changes, restart the Edge UI using the Apigee service utility.

Configuring TLS on the Enterprise UI

Test

- Open the UI in the browser.



Once the service has started, confirm that the configured port is open and the service is listening.

Open the UI in your browser using the configured port to verify that TLS was enabled successfully.

You should now be able to sign in to the Edge UI over HTTPS.

Configuring TLS on the Enterprise UI

Configure the UI to communicate with the Management Server over HTTPS

Confirm that the management API is accessible and configured to use HTTPS:

```
curl -u <admin@email> https://<MS_IP>:<SSL_PORT>/v1/o
```

On the server running the UI, append the following property to the file:

`/opt/apigee/customer/application/ui.properties`

Property:

`conf_apigee_apigee.mgmt.baseurl="https://<MS_IP>:8443/v1"`

Restart the Edge UI.

It is possible to configure the Edge UI to make management calls exclusively over HTTPS (default is HTTP).

First, confirm that the management server is available on HTTPS by either using a port check or invoke the /organizations API as shown.

Next, on the server running the Edge UI, edit or create a file called UI.properties in the customer application directory and add the baseurl property.

This property sets the base URL for all management API calls.

To complete the configuration, restart the Edge UI service.

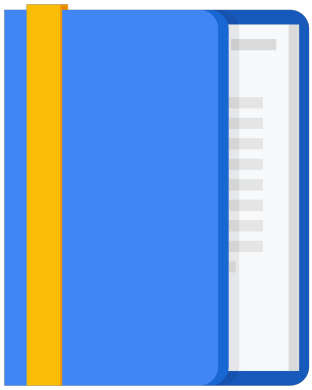
Agenda

TLS overview

Configuring TLS

Apigee mTLS

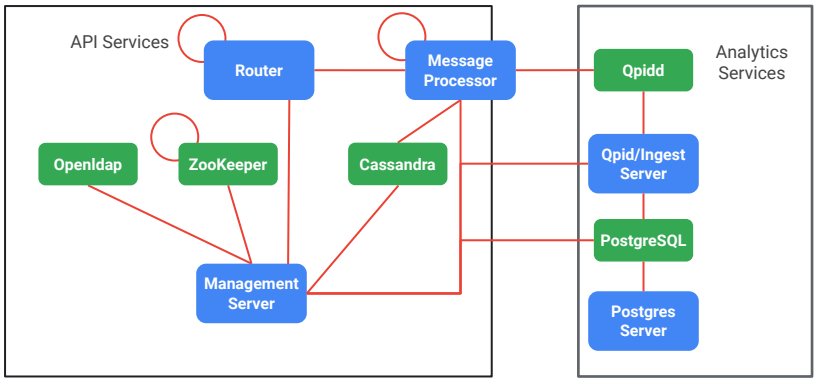
SSO integration



In this lesson, we discuss the installation and configuration of the mTLS feature in Apigee Edge for private cloud.

Architecture

Overview



The Apigee mTLS feature adds security to communications between components in your Apigee Edge private cloud installation.

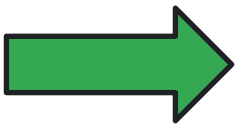
Apigee mTLS uses a service mesh that establishes secure, mutually authenticated TLS connections between components.

Apigee mTLS has the following limitations:

- It does not encrypt inter-node Cassandra communications on port 7000.
- The configuration and setup is not idempotent. This means that if you make a mTLS configuration change on any Apigee node, you must make the same change on all nodes. The system does not apply that change for you on any other node.

Architecture

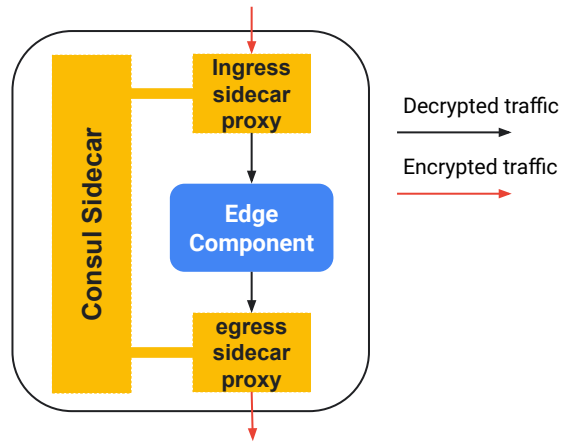
mTLS connections between components

Management Server		Router, MP, QPID, LDAP, Postgres, ZooKeeper, and Cassandra nodes
Router		Loopback; Qpid, ZooKeeper, and Cassandra
Message Processor		Loopback; Qpid, ZooKeeper, and Cassandra
ZooKeeper		Other ZooKeeper nodes
Edge UI		SMTP (for external IDP only)
Postgres		Other Postgres, ZooKeeper, and Cassandra

Apigee mTLS can be implemented on connections between most components in the cluster.

Architecture

Message encryption/decryption



The Apigee mTLS service mesh consists of Consul servers that run on each ZooKeeper node and a Consul sidecar on every Apigee node in the cluster:

There are two sidecar proxies as part of the Consul Sidecar process:

- An **egress sidecar proxy** which transparently intercepts and encrypts outgoing messages before sending them to their destination.
- An **ingress sidecar proxy** that intercepts incoming messages to the host node. This service decrypts incoming messages before sending them to the component for processing.

Additionally, Apigee mTLS uses the iptables utility, which is a Linux firewall service that manages traffic redirection.

Traffic rules use the following patterns:

- component_name-port-IP-ingress, and
- component_name-port-IP-egress.

Requirements

Version, platform, and topology requirements

- Apigee mTLS is supported on Apigee for Private Cloud version 4.19.01 or later.
- All single-region and multi-region topologies with at least three ZooKeeper nodes are supported.
- Supported operating systems are CentOS, RHEL, and Oracle Linux.

Operating system	Private Cloud version		
	v4.19.01	v4.19.06	v4.50.00
CentOS RedHat Enterprise Linux (RHEL) Oracle Linux	7.4, 7.5, 7.6, 7.7	7.5, 7.6, 7.7	7.5, 7.6, 7.7, 7.8

Apigee mTLS has specific private cloud and operating system version requirements.

It also requires that you use a private cloud topology that includes at least three ZooKeeper nodes.

Note that Apigee mTLS does not necessarily support all the operating systems that are currently supported by the underlying version of Apigee for Private Cloud.

Requirements

Utilities/packages

Utility/package	Description
base64	Verifies data within the installation scripts.
gnu-bash gnu-sed gnu-grep	Used by the installation script and other common tools.
iptables	Replaces the default firewall, firewalld.
iptables-services	Provides functionality to the iptables utility.
lsof	Used by the installation script.
nc	Verifies iptables routes.
openssl	Signs certificates locally during the initial bootstrapping process.

Apigee mTLS requires that you have a set of packages installed and enabled on each machine in your cluster, including your administration machine, before you begin the installation process.

These packages provide various capabilities that are used by the installation script.

During installation, you also install the Consul package on the administration machine so that you can generate credentials and the encryption key.

Consul is an open source tool that provides service discovery, mTLS enforcement and other capabilities.

The apigee-mtls package installs and configures the Consul servers including the ingress and egress proxies on the ZooKeeper nodes in the cluster.

Requirements

User account permissions

- Start, stop, restart, and initialize Apigee components.
- Set firewall rules.
- Create a new OS/system user account.
- Enable, disable, start, stop, and mask services with systemctl.

The account that executes the Apigee mTLS installation on each node in the Apigee private cloud cluster must be able to perform various functions including starting and stopping Apigee services, setting firewall rules and creating new system user accounts.

You can create a new user account or ensure that you have access to one that has elevated privileges that allows you to perform these functions.

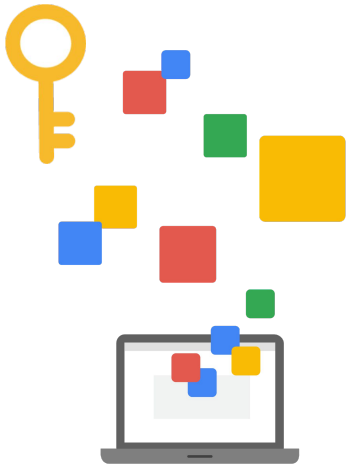
Requirements

Administration server

We recommend that you use a dedicated admin server for Apigee mTLS installation and management.

You use the administration server to:

- Install Consul
- Generate and distribute a certificate/key pair and gossip encryption key.
- Update and distribute the configuration file.



Apigee recommends that you have a node within the cluster on which you can perform various administrative tasks.

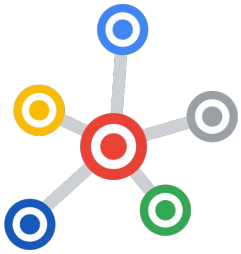
When setting up the administration machine:

- Ensure that you have root access to it.
- Download and install the apigee-service and apigee-setup utilities on it.
- Verify you can use scp/ssh to access all nodes in the cluster.

Requirements

Port usage: All nodes running apigee-mtls

All nodes in the cluster that use the apigee-mtls service must allow connections from services on the localhost.



Consul proxies use localhost to communicate with the other services as they process incoming and outgoing messages.

All nodes in the cluster that use the apigee-mtls service must allow connections from services on the localhost.

Requirements

Port usage: Consul server nodes (nodes running ZooKeeper)

Port	Description	Protocol	External access
8300	Connects all Consul servers in the cluster.	RPC	Yes
8301	Handles membership and broadcast messages within the cluster.	UDP/TCP	Yes
8302	WAN port that handles membership and broadcast messages in a multi-data center configuration.	UDP/TCP	Yes
8500	Handles HTTP connections to the Consul Server APIs from processes on the same node.	HTTP	No
8502	Handles gRPC+HTTPS connections to the Consul Server APIs from other nodes in the cluster.	gRPC+HTTPS	Yes
8503	Handles HTTPS connections to the Consul Server APIs from other nodes in the cluster.	HTTPS	Yes
8600	Handles the Consul server's DNS.	UDP/TCP	No

The ZooKeeper nodes running the consul-server component must open ports 8300, 8301, 8302, 8502, and 8503 to all members of the cluster that are running the apigee-mtls service, even across data centers.

Nodes that are not running the consul-server do not need to open these ports.

Requirements

Port assignments for all Consul nodes

Apigee Component	Port range	Number of ports required per node
Apigee mTLS	10700 to 10799	1
Cassandra	10100 to 10199	2
Message Processor	10500 to 10599	2
OpenLDAP	10200 to 10299	1
Postgres, PostgreSQL	10300 to 10399	3
Qpidd, Qpid server	10400 to 10499	2
Router	10600 to 10699	2
ZooKeeper	10001 to 10099	3

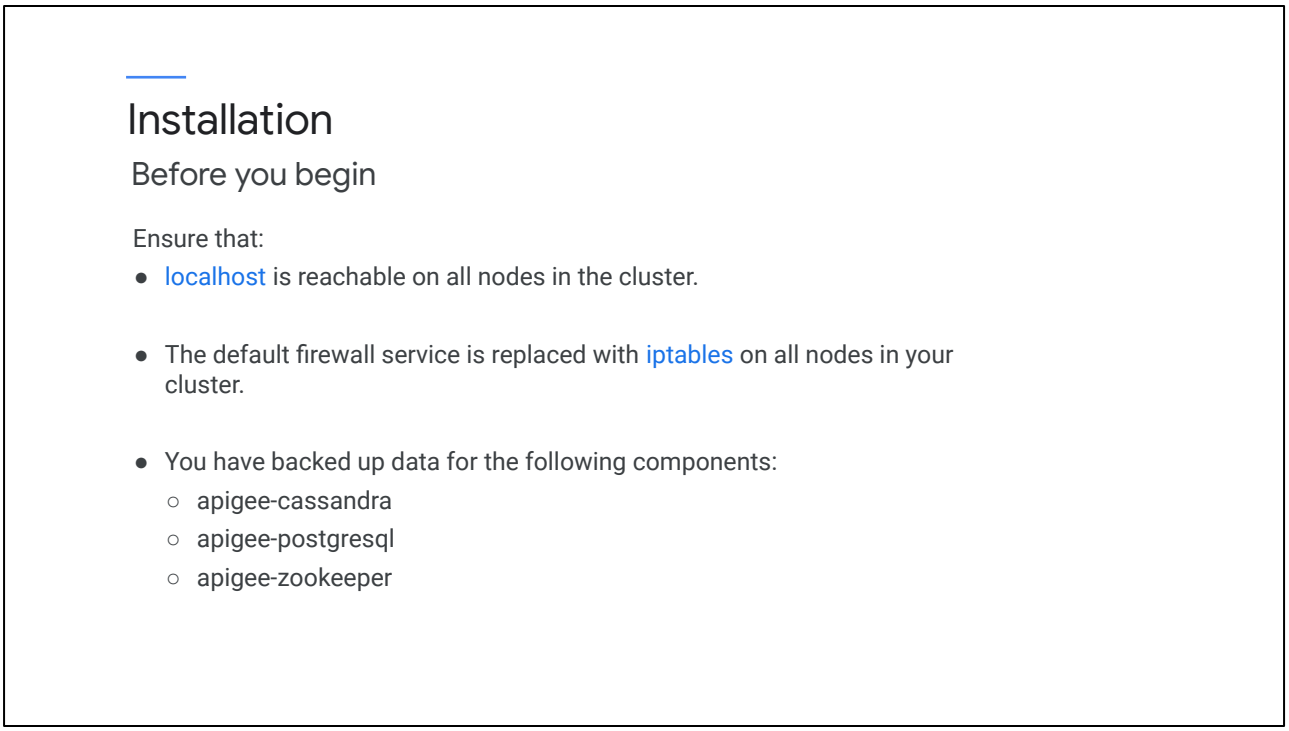
To support Consul communications, nodes running Apigee components must allow external connections to ports within the specific ranges.

Consul assigns ports linearly. For example, if your cluster has two Postgres nodes, the first node uses three ports, so Consul assigns ports 10300 through 10302 to it.

The second node also uses three ports, so Consul assigns 10303 through 10305 to that node. This applies to all component types.

Note that:

- Consul proxies cannot listen on the same ports as Apigee services.
- Consul has only one port address space. Consul proxy port assignments must be unique across the cluster, which includes data centers. This means that if proxy A on one node listens on port 15000, then proxy B on another node cannot listen on port 15000.
- The number of ports used varies based on the topology.



Apigee mTLS requires that the localhost loopback address is enabled. The IP address 127.0.0.1 must be routable and it must resolve to localhost on every node in the cluster.

The default firewall on CentOS and RedHat Enterprise Linux is firewalld. Apigee mTLS requires that you use iptables as your firewall instead. You must uninstall **firewalld** and install iptables on all nodes.

During the installation and configuration of Apigee mTLS, you will reinstall Cassandra and Postgres on their respective nodes. ZooKeeper will not be reinstalled but it is recommended to backup all three services.

Note that the Apigee mTLS installation process may cause some downtime for your Apigee cluster, so it is recommended to perform the installation during a maintenance window.

Installation

Install apigee-mtls

1. Stop all apigee components:
   ```
   /opt/apigee/apigee-service/bin/apigee-all stop
   ```

2. Confirm that all components are stopped:
   ```
   /opt/apigee/apigee-service/bin/apigee-all status
   ```

3. Install Apigee mTLS
   ```
   /opt/apigee/apigee-service/bin/apigee-service apigee-mtls install
   ```

To install apigee mtls, you must first stop all services on all Apigee nodes.

Once all the services have stopped, use the apigee service utility to install apigee mtls.

The utility installs two rpm packages - apigee-mtls and apigee-mtls-consul on the node.

Repeat the above steps on each node in your Apigee private cloud cluster.

You will continue with configuring mTLS after it is installed on all Apigee nodes.

Note that once Apigee mTLS is installed on your private cloud cluster, you must start it before starting any other component on that node.

Configure Apigee mTLS

Overview

1. Install Consul and generate credentials.
2. Update your configuration file.
3. Distribute the credentials and configuration file.
4. Initialize apigee-mtls.

Multi-region installation of Apigee mTLS:
https://docs.apigee.com/private-cloud/v4.50.00/mtls-multiple

After you have installed Apigee mTLS on all nodes in your cluster, you must configure and initialize the apigee-mtls component.

This section describes how to configure Apigee mTLS after the initial installation in a single region.

For information on how to install Apigee mTLS on a multi-region installation of Apigee, follow the documentation at the link provided.

Configure Apigee mTLS

Install Consul and generate credentials

Installation steps:

1. Log in to your administration machine.
2. Download Consul.
3. Move the executable to a permanent location.
4. Create a new certificate authority:

```
/opt/consul/consul tls ca create
```

5. Create an encryption key for Consul:

```
/opt/consul/consul keygen
```

It is recommended to use a dedicated machine for Apigee mTLS administration.

Login to your admin machine and download the Consul package from the HashiCorp website. Extract the contents from the archive to a permanent location, for example /opt/consul.

Create a certificate and key pair for a new certificate authority or CA using the consul executable and supplying the tls ca create command options.

The certificate and key pair files will be distributed to all Apigee nodes later. By default, the certificate and key files are X509v3 encoded.

Next, create an encryption key using the consul executable with the keygen option. Save the output of this command for use in the next step.

Configure Apigee mTLS

Update your configuration file

```
.....
ALL_IP="$IP1 $IP2 $IP3 $IP4 $IP5"
LDAP_MTLS_HOSTS="$IP3"
ZK_MTLS_HOSTS="$IP3 $IP4 $IP5"
CASS_MTLS_HOSTS="$IP3 $IP4 $IP5"
PG_MTLS_HOSTS="$IP2 $IP1"
RT_MTLS_HOSTS="$IP4 $IP5"
MS_MTLS_HOSTS="$IP3"
MP_MTLS_HOSTS="$IP4 $IP5"
QP_MTLS_HOSTS="$IP2 $IP1"
ENABLE_SIDECAR_PROXY="y"
ENCRYPT_DATA="QbhgD+EXAMPLE+Y9u0742X/IqX3X429/x1cIQ+JsQvY="
PATH_TO_CA_CERT="/opt/apigee/customer/consul-agent-ca.pem"
PATH_TO_CA_KEY="/opt/apigee/customer/consul-agent-ca-key.pem"
.....
```

Configuration properties: https://docs.apigee.com/private-cloud/v4.50.00/mtls-configure#updateconfig

To configure Apigee mTLS, you add a set of mTLS configuration properties to your existing private cloud installation configuration file.

The ENABLE_SIDECAR_PROXY property must be set to "y".

Ensure that the correct private IP address of each node is provided for the corresponding properties.

Set the value of the PATH_TO_CA_CERT and PATH_TO_CA_KEY properties to the location to which you will copy the certificate and key files on each node in the cluster. These files were generated in the previous step.

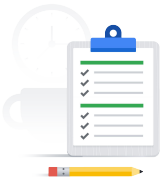
Set the value of the ENCRYPT_DATA property to the value of the encryption key that was generated in the previous step.

All configuration options are described on the documentation page at the link provided.

Configure Apigee mTLS

Distribute the configuration file and credentials

- Distribute the following files to all Apigee nodes:
  - configuration file
  - consul-agent-ca.pem
  - consul-agent-ca-key.pem



Once the configuration file is updated, copy the configuration and credential files to all Apigee nodes in the cluster.

The location of the credential files must match the values set for the configuration properties PATH_TO_CA_CERT and PATH_TO_CA_KEY in the previous step.

All files should be owned by the Apigee user.

Configure Apigee mTLS

Initialize apigee-mtls

1. Configure apigee-mtls:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-mtls setup -f config_file
```

2. Verify the setup:

```
/opt/apigee/apigee-mtls/lib/actions/iptables.sh validate
```

3. Start Apigee mTLS:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-mtls start
```

Log in to each Apigee node and initialize Apigee mTLS.

You start by configuring the mTLS component. It is important to use the full path to the configuration file in all commands.

Verify that the configuration was successful before you start the mTLS component.

Configure Apigee mTLS

Initialize apigee-mtls: Cassandra nodes

Cassandra-specific commands:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-cassandra setup -f config_file
/opt/apigee/apigee-service/bin/apigee-service apigee-cassandra configure
/opt/apigee/apigee-service/bin/apigee-service apigee-cassandra restart
```

Cassandra requires additional arguments to work within the security mesh.

You must execute the commands on each Cassandra node.

These commands configure and restart the Cassandra node.

Configure Apigee mTLS

Initialize apigee-mtls: Postgres nodes

Postgres active node:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-postgresql setup -f config_file
/opt/apigee/apigee-service/bin/apigee-service apigee-postgresql configure
/opt/apigee/apigee-service/bin/apigee-service apigee-postgresql restart
```

Postgres standby node:

```
rm -rf /opt/apigee/data/apigee-postgresql/pgdata
/opt/apigee/apigee-service/bin/apigee-service apigee-postgresql setup -f config_file
/opt/apigee/apigee-service/bin/apigee-service apigee-postgresql configure
/opt/apigee/apigee-service/bin/apigee-service apigee-postgresql restart
```

You have to reconfigure your PostgreSQL nodes to work properly in the service mesh.

Note that when run on the standby node, the commands remove all data and re-configure the active/standby replication process.

It is a good practice to backup your existing PostgreSQL data before running these commands.

The commands configure the nodes for mTLS and restarts the services.

Configure Apigee mTLS

Initialize apigee-mtls: Other Apigee nodes

Start the component:

```
/opt/apigee/apigee-service/bin/apigee-service component_name start
```

Finally, you must start the components on the remaining nodes used in the Apigee private cloud installation.

Repeat the apigee-service start command for each node in the cluster.

Configure Apigee mTLS

Change existing mTLS configuration

```
# DO THIS:
/opt/apigee/apigee-service/bin/apigee-service apigee-mtls uninstall

# BEFORE YOU DO THIS:
/opt/apigee/apigee-service/bin/apigee-service apigee-mtls setup -f config_file

# AND THIS:
/opt/apigee/apigee-service/bin/apigee-service apigee-mtls configure
```

The Apigee mTLS setup and configuration is not idempotent. If you make a change on one node, you must make that same change on all nodes.

If you do not make the same change on all nodes, the service mesh may fail since the apigee-mtls configuration entries must be identical on all nodes.

Repeatedly executing the apigee-mtls setup or configuration process does not have the same intended result as other Apigee services, and will not necessarily generate an updated configuration.

You must first uninstall Apigee mTLS when you change the configuration, and then re-run the setup and configure commands on all nodes in the cluster.

Verify Apigee mTLS installation

Validate the iptables configuration

- Log in to an Apigee node.
- Stop all components:

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

- Execute validate command:

```
/opt/apigee/apigee-mtls/lib/actions/iptables.sh validate
```

- Start mTLS:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-mtls start
```

- Start the other Apigee components:

```
/opt/apigee/apigee-service/bin/apigee-service component_name start
```

- Repeat the steps on all nodes in the cluster.

You can validate that the apigee-mtls installation was successful by checking that the iptable routes are working and that the rules are valid.

To do this, log in to a node and stop all Apigee components that are running on that node.

Execute the validate command. This command uses iptables to send messages to every port that is used by Consul or local Apigee services.

The script returns an error if it encounters an invalid or failed route.

Note that the script will fail if any Apigee services or Consul servers are running on the node.

After validation, start the apigee-mtls component first and then start all the other Apigee private cloud components on the node.

Repeat these steps on all the nodes in the cluster.

Verify Apigee mTLS installation

Verify the remote proxy status

- Log in to a ZooKeeper node.
- Execute the command:

```
systemctl status consul_server
```

You can use the Consul server on a ZooKeeper node to check if the ingress and egress sidecar proxy services on all nodes are alive, healthy and have joined the service mesh.

Execute the command shown to check the status of the egress sidecar proxy.

Verify Apigee mTLS installation

Verify the Consul quorum status

- Log in to a ZooKeeper node.
- Check Consul servers quorum status:

```
/opt/apigee/apigee-mtls-consul/bin/consul operator raft list-peers
```

Example output:

```
Node             ID                    Address            State     Voter  RaftProtocol
prc-test-0-1619  b59c1f44-6eb0-81d4-42  10.126.0.98:8300   leader    true   3
prc-test-1-1619  a4372a6e-8044-e587-43  10.126.0.146:8300  follower  true   3
prc-test-2-1619  71eb181f-4242-5353-44  10.126.0.100:8300  follower  true   3
```

- Get additional information about Consul service mesh health:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-mtls status
```

Consul needs at least three servers to form a quorum.

To check the availability of all Consul servers, login to a ZooKeeper node and use the consul operator command.

The command displays a list of the Consul instances and their status.

In addition, you can get information on the health of the service mesh using the apigee-service utility's apigee-mtls status command.

Upgrade Apigee mTLS

1. Backup Cassandra, Postgres, and ZooKeeper data.
2. Stop all components:
   ```
   /opt/apigee/apigee-service/bin/apigee-all stop
   ```
3. Upgrade Apigee mTLS:
   ```
   /opt/apigee/apigee-service/bin/apigee-service apigee-mtls update
   ```
4. Start the mTLS component:
   ```
   /opt/apigee/apigee-service/bin/apigee-service apigee-mtls start
   ```
5. Start the other Apigee components.

Before upgrading Apigee mTLS, it is recommended to backup the Cassandra, Postgres, and ZooKeeper data in your private cloud installation.

To upgrade Apigee mTLS, first stop all components on the node.

Next, upgrade the mTLS component using the apigee-mtls update command of the apigee-service utility.

Start the mTLS component using the apigee-service apigee-mtls start command.

Finally, start the rest of the components on the node.

Note that upgrading Apigee mTLS may cause downtime for your Apigee cluster. It is recommended to perform the upgrade in a maintenance window.

Uninstall Apigee mTLS

Uninstall process

1. Log in to any Apigee node.
2. Stop all components on the node:
   `/opt/apigee/apigee-service/bin/apigee-all stop`
3. Uninstall mTLS:
   `/opt/apigee/apigee-service/bin/apigee-service apigee-mtls uninstall`
4. Start all components:
   `/opt/apigee/apigee-service/bin/apigee-service component_name start`

   Component start order:
   https://docs.apigee.com/private-cloud/v4.50.00/starting-stopping-and-restarting-apigee-edge#start-order

If you need to uninstall Apigee mTLS, you can do so at any time.

To uninstall Apigee mTLS, log in to any node in the cluster.

Stop all components on the node and execute the apigee-service apigee-mtls uninstall command.

Once the uninstall command completes, start all the components on the node.

Follow the start order of the components as documented in the link provided.

Repeat this process for each node in the cluster.

Note that the Apigee mTLS uninstall process may cause downtime for your Apigee cluster, so it is recommended to uninstall Apigee mTLS in a maintenance window.

Uninstall Apigee mTLS

Verify uninstallation

- On each ZooKeeper node, confirm:
  - The files consul_egress.service and consul_server.service are not in the /usr/lib/systemd/system directory.
  - The following directories and symlinks do not exist in the Apigee root directory (/opt/apigee):
    ```
    apigee-mtls-version
    apigee-mtls-consul-version
    apigee-mtls
    apigee-mtls-consul
    ```
  - The Consul binary was removed and Consul sidecar process is not running.
- Check whether all iptables rules were removed:
  ```
  iptables -t nat -L OUTPUT
  ```
- Use OS package manager to check whether Apigee mTLS packages were removed.

To verify that the uninstall was successful, you confirm that all systemd service files, the Consul binary and apigee-mtls folders and symlinks have been removed from the ZooKeeper nodes.

You should also confirm that the iptables rules and rpm packages have been removed.

Additional readings

- mTLS customization:
  - Customize proxy port ranges: https://docs.apigee.com/private-cloud/v4.50.00/mtls-ports
  - Configure apigee-monit with mTLS: https://docs.apigee.com/private-cloud/v4.50.00/mtls-monit
  - Configure multiple data centers for Apigee mTLS: https://docs.apigee.com/private-cloud/v4.50.00/mtls-multiple
- Using a custom certificate: https://docs.apigee.com/private-cloud/v4.50.00/mtls-certs
- Troubleshooting Apigee mTLS: https://docs.apigee.com/private-cloud/v4.50.00/mtls-troubleshoot

For additional information on customizing mTLS, using custom certificates or troubleshooting Apigee mTLS please review the documentation at the links provided.

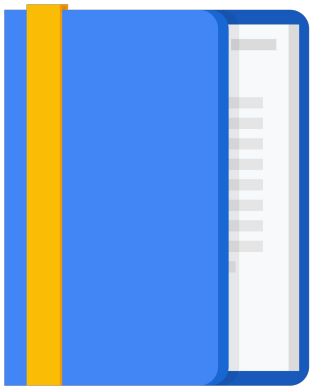
Agenda

TLS overview

Configuring TLS

Apigee mTLS

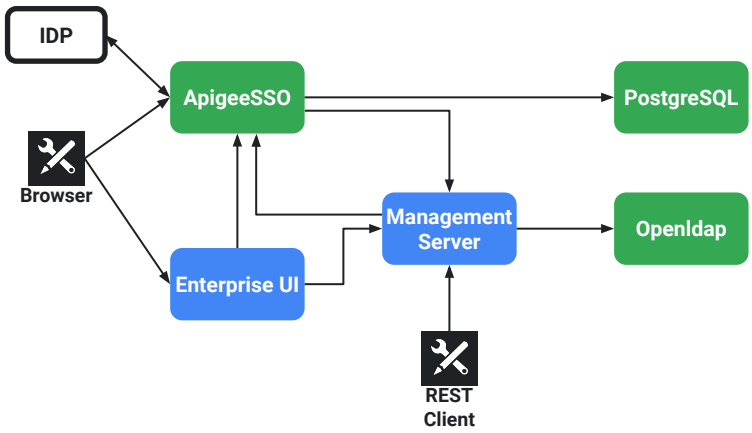
SSO integration



In this lesson we discuss how to configure Apigee Edge for private cloud to use an identity provider for authentication and single sign on.

Architecture

Overview



By default, Apigee Edge for private cloud uses Apache Openldap to authenticate users who log in to the UI or send requests to the management API.

By installing the Apigee SSO module, you can use an external identity provider for authentication.

By using an external IDP with Edge, you can support single sign-on for the Edge UI and API in addition to any other services that you provide and that also support your external IDP.

To integrate with external identity providers, Apigee supports the following authentication standards:

- Security Assertion Markup Language (SAML) 2.0 can be used to generate OAuth access tokens from SAML assertions that are returned by a SAML identity provider, and
- Lightweight Directory Access Protocol (LDAP) that can be used with LDAP binding methods for authentication to generate OAuth access tokens.

To support SAML or LDAP on Apigee Edge, you must install the Apigee SSO module.

Note that both SAML and LDAP are used as authentication mechanisms and support a single sign-on environment, but are not supported for authorization.

You will still use the Apigee Openldap database to maintain authorization information.

Architecture

About LDAP

The Lightweight Directory Access Protocol (LDAP) is an open, industry-standard application protocol for accessing and maintaining distributed directory information services.

LDAP authentication within Apigee SSO uses the Spring Security LDAP module.

The authentication methods and configuration options for Apigee SSO's LDAP support are directly correlated to those found in Spring Security LDAP.

LDAP with Apigee for Private Cloud supports the following authentication methods against an LDAP-compatible server:

- Search and Bind (indirect binding)
- Simple Bind (direct binding)

Architecture

About SAML

Security Assertion Markup Language (SAML) is an open standard that allows identity providers (IDP) to pass authorization credentials to service providers (SP).

With SAML enabled, access to the Apigee Edge UI and management API uses OAuth2 access tokens.

These tokens are generated by the Apigee SSO module which accepts SAML assertions returned by your identity provider.

Once generated from a SAML assertion, the OAuth token is valid for 30 minutes and the refresh token is valid for 24 hours.

Installation and configuration

Configure your IDP

These links provide more information on how to configure your external identity provider (IDP):

- SAML IDP:
  https://docs.apigee.com/private-cloud/v4.50.00/configure-your-saml-idp
- LDAP IDP:
  https://docs.apigee.com/private-cloud/v4.50.00/configure-ldap-idp

The process to install and configure external IDP support on Apigee Edge for Private Cloud requires that you perform some tasks on your IDP and some on Apigee Edge.

The links provided contain documentation on how to configure your external identity provider using the SAML or LDAP authentication mechanisms.

After completing the tasks to configure your IDP, you can then continue with the next step in the process to install and configure the Apigee SSO module.

Installation and configuration

Create the TLS keys for verification and signing

1. Create the private key:
   ```
   openssl genrsa -out privkey.pem 2048
   ```
2. Create the public key:
   ```
   openssl rsa -pubout -in privkey.pem -out pubkey.pem
   ```
3. Place them in a folder owned by the Apigee user:
   `/opt/apigee/customer/application/apigee-sso/jwt-keys`
4. Verify that all files and folders are owned by the Apigee user.

To install and configure the Apigee SSO module with an external IDP, you must first create TLS keys and certificates.

The Apigee SSO module uses TLS to secure the transmission of information with external entities.

Use the openssl tool on Linux to create the private and public keys. The key files should be owned and accessible by the Apigee user.

Installation and configuration

Create the key and certificate for communicating with the IDP

1. Generate your private key:
   ```
   openssl genrsa -aes256 -out server.key 1024
   ```
2. (Optional) To remove the private key passphrase for test environments:
   ```
   openssl rsa -in server.key -out server.key
   ```
3. Generate a certificate signing request:
   ```
   sudo openssl req -x509 -sha256 -new -key server.key -out server.csr
   ```
4. Generate a self-signed certificate valid for 365 days:
   ```
   sudo openssl x509 -sha256 -days 365 -in server.csr -signkey server.key -out selfsigned.crt
   ```
5. Place the key and certificate in a folder accessible by the Apigee user:
   ```
   /opt/apigee/customer/application/apigee-sso/idp/
   ```

For testing you can use self-signed certificates and private key without passphrase. In production environments you have to use certificates signed by an internal (company trusted) or public certificate authority.

If you use a passphrase for your private key , make sure to provide that passphrase in the apigee SSO configuration file as property SSO_SAML_SERVICE_PROVIDER_PASSWORD.

Use the openssl tool on Linux to create the private key and certificate. The key and certificate files should be owned and accessible by the Apigee user.

Installation and configuration

Apigee SSO configuration file: Management Server properties

```
## Servers
IP1=hostname_or_IP_of_management_server
IP2=hostname_or_IP_of_UI_and_apigge_SSO

## Management Server configuration.
MSIP=$IP1
MGMT_PORT=8080
ADMIN_EMAIL=opdk@google.com
APIGEE_ADMINPW=Secret123
MS_SCHEME=http
```

Before you can install the Apigee SSO module, you must create a configuration file.

You pass this configuration file to the installer when you install the Apigee SSO module.

We will review the configuration file by section.

We start with the section that contains the server IP addresses and management server properties.

This section configures access to the management API that is hosted by the management server.

Installation and configuration

Apigee SSO configuration file: PostgreSQL properties

```
## Postgres configuration.
PG_HOST=$IP1
PG_PORT=5432
PG_USER=apigee
PG_PWD=postgres
```

ApigeeSSO stores data in a PostgreSQL database. You could use a dedicated PostgreSQL server or use the PostgreSQL servers that are installed with Apigee for private cloud.

During installation, ApigeeSSO creates a dedicated database on the PostgreSQL server that you configure.

In this section, you provide the IP address and port of the PostgreSQL server and access credentials.

Installation and configuration

Apigee SSO configuration file: SSO properties

```
## Apigee SSO module configuration.
SSO_PROFILE="[saml|ldap]"
SSO_PUBLIC_URL_HOSTNAME=$IP2
SSO_PG_DB_NAME=database_name_for_sso
SSO_PUBLIC_URL_PORT=9099
SSO_TOMCAT_PORT=9099
SSO_TOMCAT_PROFILE=DEFAULT
SSO_PUBLIC_URL_SCHEME=http
SSO_ADMIN_NAME=ssoadmin
SSO_ADMIN_SECRET=Secret123
SSO_SAML_SIGN_REQUEST=y
SSO_JWT_SIGNING_KEY_FILEPATH=/opt/apigee/customer/application/apigee-sso/jwt-keys/priv
key.pem
SSO_JWT_VERIFICATION_KEY_FILEPATH=/opt/apigee/customer/application/apigee-sso/jwt-keys
/pubkey.pem
```

In the SSO section of the configuration file, you configure several key properties that include:

The type of the external IDP that will be used: SAML or LDAP.

The externally accessible IP address or DNS name, port and protocol of the apigee-sso service.

The SSO administrator username and password, SSO PostgreSQL database name, and the location of signing and verification keys that were created earlier.

Installation and configuration

Apigee SSO configuration file: SMTP properties

```
# Configure an SMTP server so that the Apigee SSO
module can send emails to users
SKIP_SMTP=n
SMTPHOST=smtp.example.com
SMTPUSER=smtp@example.com
SMTPPASSWORD=smtppwd
SMTPSSL=n
SMTPPORT=25
SMTPMAILFROM="My Company <myco@company.com>"
```

ApigeeSSO needs an SMTP server to be configured in order to send emails to users. This configuration is done in the SMTP section of the configuration file.

Installation and configuration

Apigee SSO configuration file: IDP

Use one of the following configuration blocks to define your IDP settings:

- SAML configuration
- External LDAP Direct Binding configuration
- External LDAP Indirect Binding configuration
- Indirect binding to internal Apigee Openldap IDP

IDP configuration settings:
https://docs.apigee.com/private-cloud/v4.50.00/install-and-configure-edge-sso#saml

The last section of the configuration file includes IDP configuration.

You choose one block with the correct configuration properties based on your type of IDP.

The configuration settings are specific to the type of IDP and are documented at the link provided.

Installation and configuration

Apigee SSO configuration file: External IDP SAML configuration

```
## SAML Configuration Properties
SSO_SAML_IDP_NAME=[okta|adfs]
SSO_SAML_IDP_LOGIN_TEXT="Please log in to your IDP"
SSO_SAML_IDP_METADATA_URL=https://dev-343434.oktapreview.com/app/exkar20cl/sso/saml/me
tadata
SSO_SAML_IDPMETAURL_SKIPSSLVALIDATION=n
SSO_SAML_SERVICE_PROVIDER_KEY=/opt/apigee/customer/application/apigee-sso/idp/server.k
ey
SSO_SAML_SERVICE_PROVIDER_CERTIFICATE=/opt/apigee/customer/application/apigee-sso/idp/
selfsigned.crt
# SSO_SAML_SERVICE_PROVIDER_PASSWORD=samlSP123
SSO_SAML_SIGNED_ASSERTIONS=y
```

Use this block of configuration properties if your external IDP uses SAML.

These properties contain information about the SAML IDP like the name, login text and metadata url.

You also provide the location of the TLS key and certificate used to securely communicate with the IDP, that were created earlier.

Installation and configuration

Apigee SSO configuration file: External IDP LDAP Direct Binding

```
## LDAP Direct Binding configuration
SSO_LDAP_PROFILE=direct
SSO_LDAP_BASE_URL=ldap://hostname_or_IP:port
SSO_LDAP_MAIL_ATTRIBUTE=LDAP_email_attribute
SSO_LDAP_USER_DN_PATTERN=LDAP_DN_example_pattern_=cn={0},ou=people,dc=example,dc=org
```

Use this block of configuration properties in your configuration file if you are using LDAP direct binding with your IDP.

These properties configure the LDAP profile and the LDAP server base url to which Apigee SSO connects.

You also configure the attribute name used by the LDAP server to refer to the user's email address and the pattern of the user's distinguished name or DN.

Installation and configuration

Apigee SSO configuration file: External IDP LDAP Indirect binding

```
## LDAP Indirect Binding configuration
SSO_LDAP_PROFILE=indirect
SSO_LDAP_BASE_URL=ldap://hostname_or_IP:port
SSO_LDAP_ADMIN_USER_DN=LDAP_admin_DN
SSO_LDAP_ADMIN_PWD=LDAP_admin_password
SSO_LDAP_SEARCH_BASE=LDAP_example_search_base_dc=example,dc=org
SSO_LDAP_SEARCH_FILTER=LDAP_example_search_filter_cn={0}
```

If you use LDAP indirect binding with your IDP, the following block of properties must be configured in your configuration file:

The type of LDAP profile set to ‘indirect’.

The base LDAP server URL to which the Apigee SSO service connects to.

The distinguished name (DN) and password of the LDAP administrator.

The LDAP search base DN and search filter.

Installation and configuration

Apigee SSO configuration file: Internal Apigee LDAP Indirect binding

```
## internal Apigee LDAP Indirect Binding configuration
SSO_LDAP_PROFILE=indirect
SSO_LDAP_BASE_URL=ldap://apigee-openldap_ip:10389
SSO_LDAP_ADMIN_USER_DN=uid=admin,ou=users,ou=global,dc=apigee,dc=com
SSO_LDAP_ADMIN_PWD=Secret123
SSO_LDAP_SEARCH_BASE=dc=apigee,dc=com
SSO_LDAP_SEARCH_FILTER=mail={0}
SSO_LDAP_MAIL_ATTRIBUTE=mail
```

To use the internal Apigee LDAP server for authentication with Apigee SSO, you use the properties shown here.

These properties are tuned for use with the Apigee Openldap server.

The SSO_LDAP_ADMIN_USER_DN points to the sysadmin user that was used to install Apigee Edge for private cloud.

The value for the SSO_LDAP_ADMIN_PWD property must be the sysadmin password.

Installation and configuration

Apigee SSO module installation

1. Log in to the server.
2. Install and configure apigee-sso:
   ```
   /opt/apigee/apigee-setup/bin/setup.sh -p sso -f configFile
   ```
3. Verify apigee-sso is running:
   ```
   /opt/apigee/apigee-service/bin/apigee-service apigee-sso status
   ```
4. Install the apigee-ssoadminapi.sh utility:
   ```
   /opt/apigee/apigee-service/bin/apigee-service apigee-ssoadminapi install
   ```

Once the configuration file has been created, you are ready to install Apigee SSO.

Choose a server to install the ApigeeSSO component. You could install it on the server that hosts the Management Server or on any other server.

If you pick a different server, you must first install the apigee-service utility as described in the installation module of the Apigee private cloud on premises Installation and fundamentals course.

Also ensure that the ApigeeSSO configuration file is accessible and owned by the Apigee user.

You begin by first installing and configuring the sso component using the setup utility and providing the location of the configuration file that was created earlier.

After the setup utility completes, verify that apigee-sso is running using the apigee service utility with the apigee-sso status command options.

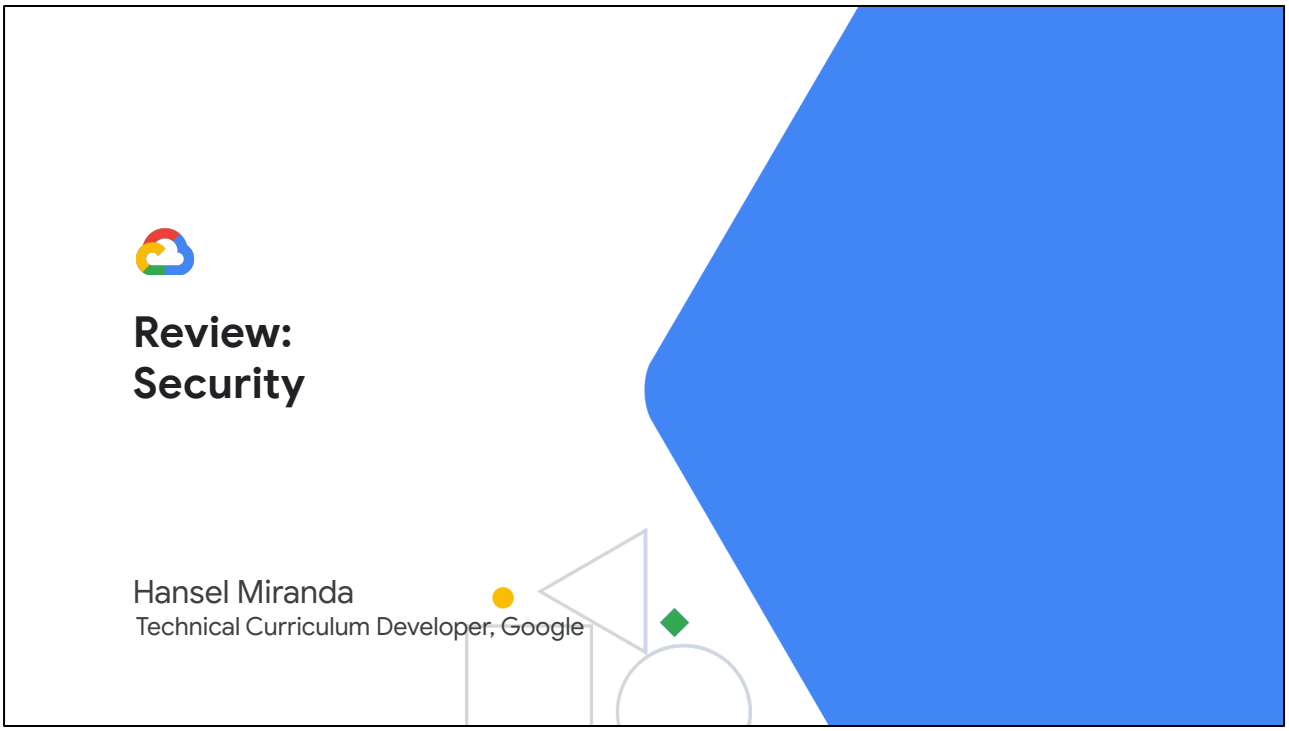
You then install the apigee-ssoadminapi module using the apigee-service utility. This module is used to manage admin and machine users for the apigee-sso service.

Additional information

Please visit this documentation page for additional information:

https://docs.apigee.com/private-cloud/v4.50.00/idp-auth

Additional information on installing and configuring single sign-on, and integrating Apigee private cloud with identity providers is available at the link provided.



Security is one of the key features of the Apigee API platform for private cloud.

In this module, we discussed how to configure TLS or transport layer security in Apigee Edge for private cloud.

You learned the process to configure TLS on the router and message processor components, and the process to secure the management server and enterprise UI.

You also learned how to configure mutual TLS on the Apigee private cloud platform, and how to configure Apigee Edge to use an external identity provider for authentication and single sign on.


Google Cloud