Welcome to the Upgrade module of the On-premises Management, Security, and Upgrade course of Google Cloud's Apigee API Platform.

In this module, you learn how to upgrade the Apigee platform for private cloud.

Agenda

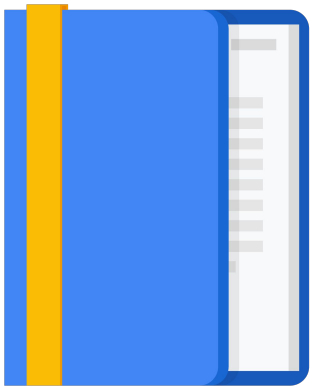
Upgrade planning

Prerequisites and dependencies

Upgrade process

Lab

Rollback process



We will start by discussing the planning and preparation process and the prerequisites and dependencies required to perform the upgrade.

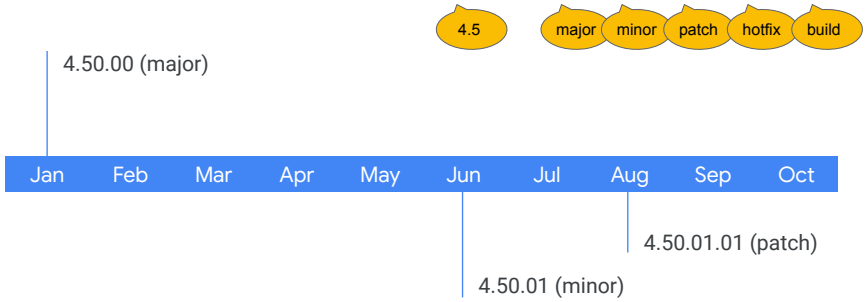
Next, you will learn how to perform the upgrade and do a lab to upgrade Apigee Edge for private cloud on Google Compute Engine.

Finally, we will look at post upgrade validation and the rollback process.

Let's get started.

Software releases

- Up to 2 major and minor software releases per year (as needed)
- Patches per software version released once per month (for priority issues)
- Each version of the software supported for up to one year
  - Includes actively releasing patches for the version
- Releases numbering convention: <generation>N.<XX>.<YY>.<ZZ>.<BBBB>



Starting with 4.50.00, Apigee Edge for Private Cloud releases will use the version numbering scheme as shown.

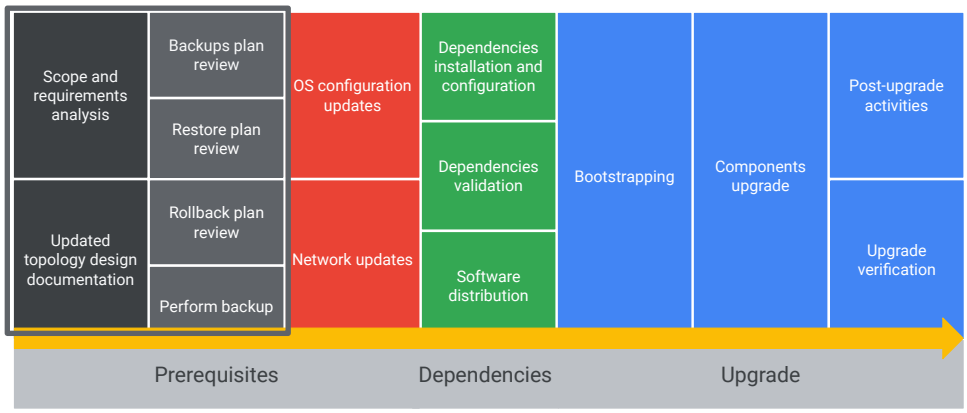
Major and minor versions will be released as needed up to twice a year. These releases may include cumulative bug fixes, configuration changes, installer changes, documentation changes, and feature enhancements.

Patches will be released once per month if there are priority issues to be resolved for customers.

Hot fixes will be released as soon as possible when needed. These releases may include critical bug fixes and fixes to mitigate security or vulnerability issues.

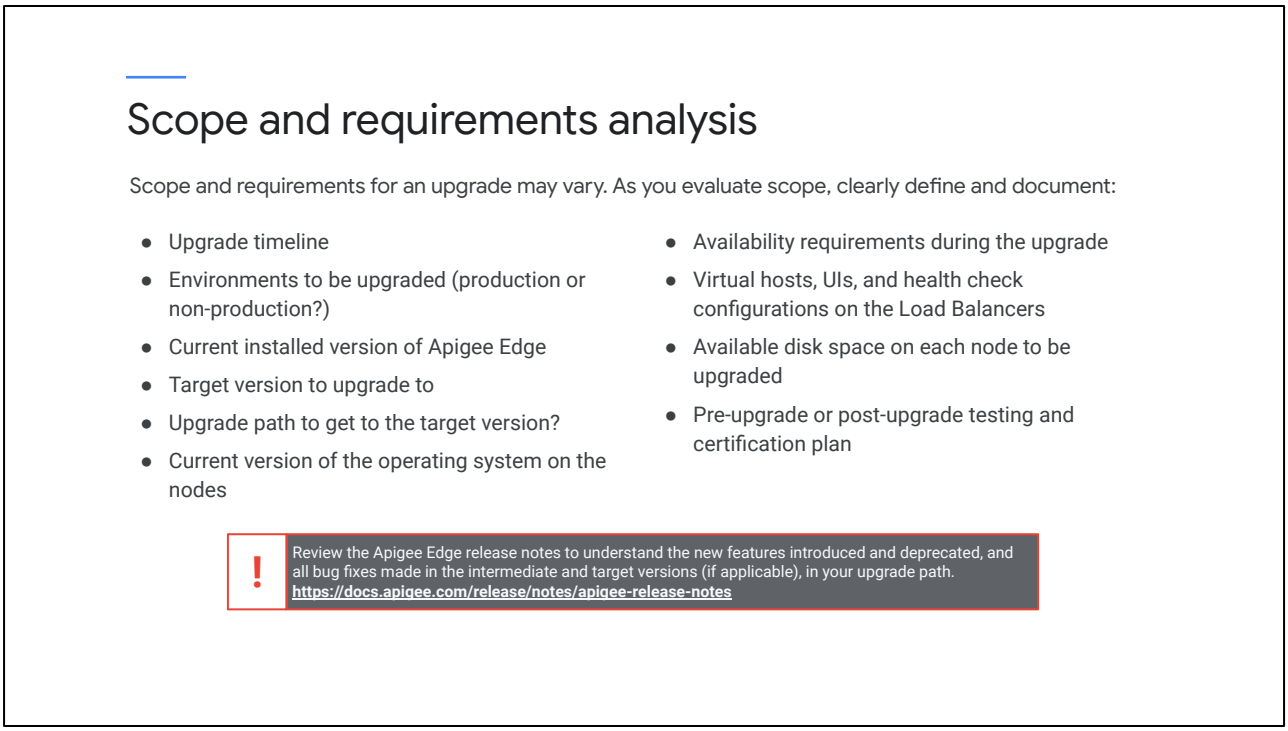
Upgrade planning

Key activities for upgrade planning



Here are some of the key activities to consider as you plan to upgrade your installation of Apigee for private cloud.

Let’s discuss the prerequisites that you need to consider as part of your upgrade planning process.



It is important to have a clear definition and set of expectations of the work to be completed for an upgrade to ensure alignment between stakeholders.

During the upgrade planning process you will want to ask several questions, such as what is the timeline for your upgrade?

Which environments will be upgraded, and in which order? What is the current installed and planned version of Apigee Edge? And what is the upgrade path to go from one to the other?

What is the current version of the operating system, and is it supported by the new release? What are the availability requirements during the upgrade process?

How are virtual hosts and Apigee UIs configured on load balancers? And how are health checks performed?

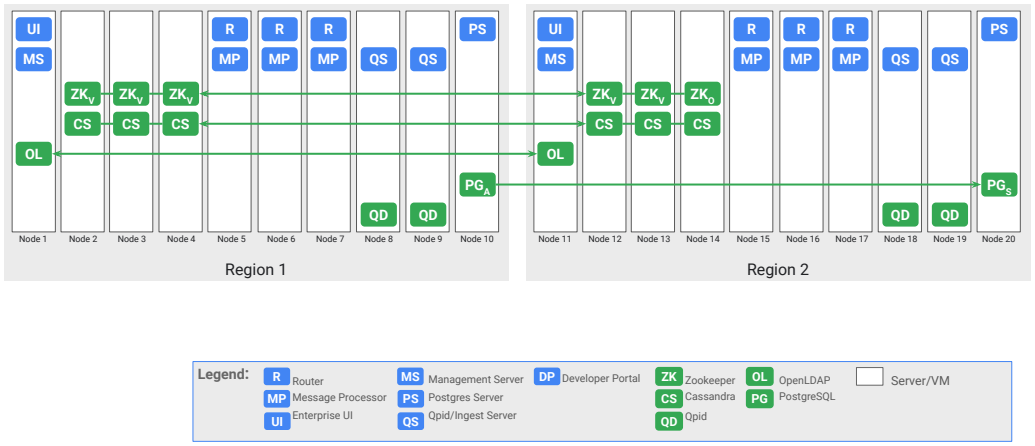
How much disk space is available on the hosts that will be upgraded?

What are the pre- and post-upgrade testing plans for API proxies, and for the installation as a whole?

When planning your upgrade, it is important to carefully review the release notes and upgrade procedures for the new release.

Apigee documents the differences and upgrade steps based on your current version and notes any known issues that you'll need to be aware of.

Updated topology design documentation



Similar to the installation process, the upgrade process defines steps that are completed on all nodes and steps that are specific to a node.

The upgrade process uses the same profile-based mechanism as used with the installation process.

In order to facilitate planning for the upgrade, it is a good practice to create documentation in the form of runbooks, the set of upgrade activities (manual and automated), and an updated topology diagram.

Review backup, restore, and rollback plans

- Ensuring business continuity should be part of your upgrade plan.
- Maintain backup, restore, and rollback plans in place.
- Plans must document all relevant procedures before you begin any upgrade activity.
- Include the plans in your Apigee Edge upgrade process runbook.

What if?

Although upgrade failures are rare, they can happen. When you are planning your upgrade, you will want to review your backup, restore, and rollback plans so that you can quickly recover if needed.

Most components can simply be rolled back to their previous state, so a full restoration shouldn't be required.

You'll also want to perform a backup immediately before the upgrade takes place using the built-in backup scripts, VM snapshots, or another method.

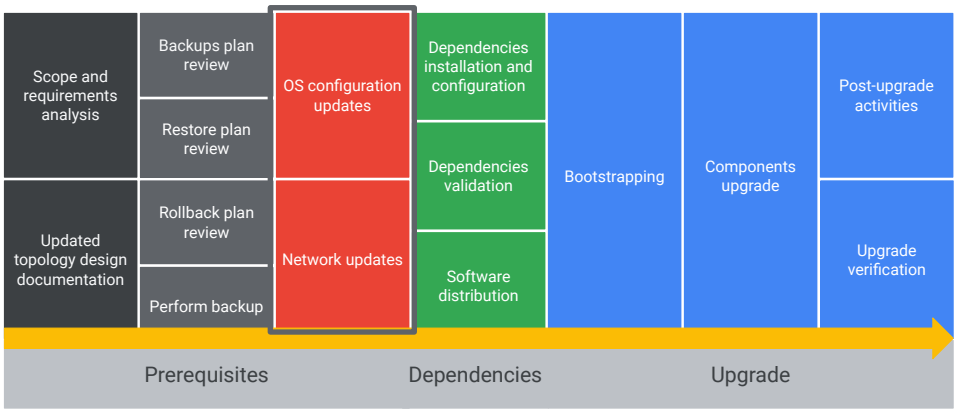
This is critical to ensuring business continuity and should be part of any upgrade plan.

A good practice is to include these plans in the runbook for upgrading Apigee Edge, which should include relevant references to backup, restore and rollback procedures.

The plans should also provide clear roles and responsibilities for those and other tasks that are undertaken as part of the upgrade process.

Upgrade preparation

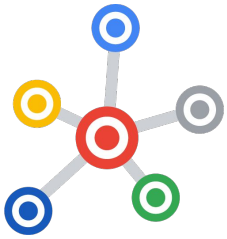
OS and network preparation



Let’s now talk about the operating system configuration and network updates that you may need to consider as part of your upgrade planning process.

OS configuration and network updates

- New releases may require an operating system update.
  - There may be new OS level dependencies and/or configuration changes.
- Network configuration changes may be required.
- The supported version of Java may change in a major release.
- Evaluate changing prerequisites and dependencies during the planning process.



New releases occasionally require operating system configuration changes, such as adding a new software repository or configuration file changes, such as adding a new variable.

Port requirements for Apigee services should generally stay constant over time. The release notes will clearly call out any changes to network configuration if they are needed.

Occasionally the supported version of Java will change as part of a major release. In those cases you can upgrade the JDK on your system manually before the upgrade or allow the upgrade program to do the JDK upgrade for you.

Review the release note for each release carefully to determine whether any changes are needed before you upgrade the Edge software.

Agenda

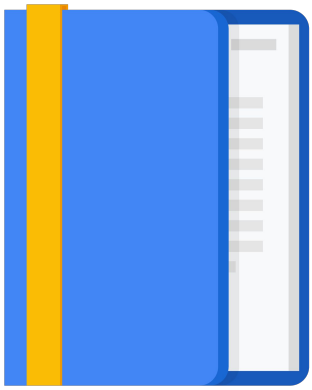
Upgrade planning

Prerequisites and dependencies

Upgrade process

Lab

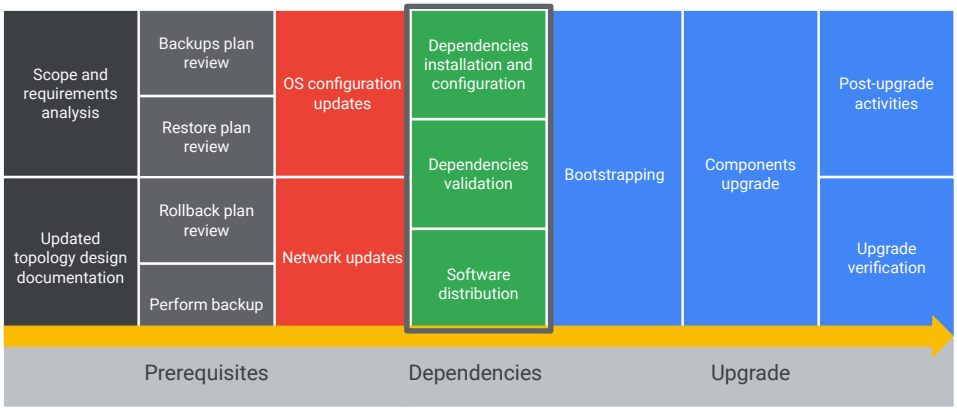
Rollback process



Let’s now discuss the prerequisites and dependencies that are required to update the Apigee private cloud platform.

Upgrade preparation

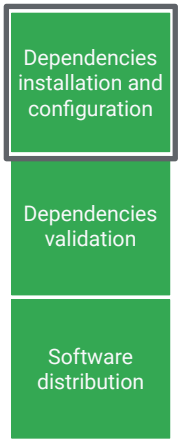
Dependency validation



We first discuss any software dependencies that you may need to consider as part of your upgrade planning process.

Software dependencies

- Software dependencies may vary between Apigee Edge software release versions.
- Major software releases may introduce new versions of open source components (Cassandra, PostgreSQL, etc).
- Refer to https://docs.apigee.com/release/supported-software for the version of open source components used.
- Software distribution for installation and upgrade uses the same mechanism.
  - Software is packaged as RPMs and distributed via centralized RPM repository.
- If you use an offline repository, the mirror must be updated with the appropriate software version.

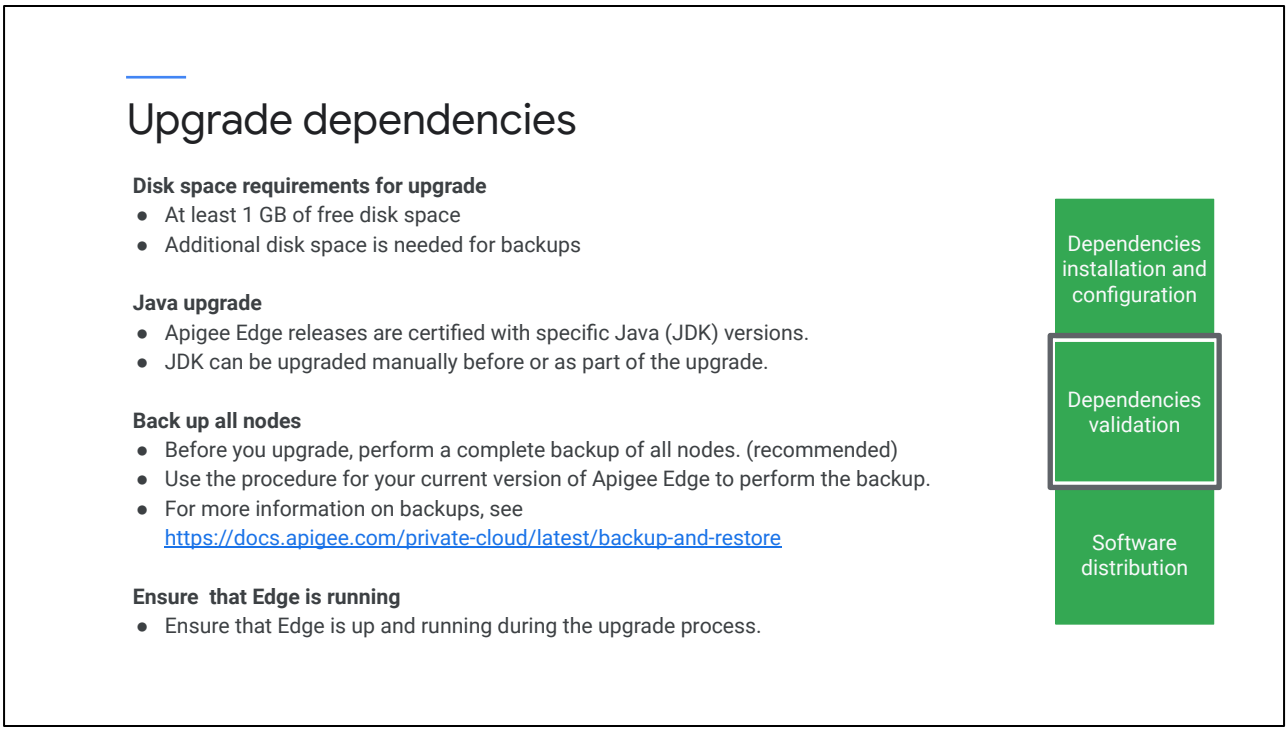


Major releases of Apigee software can contain feature additions or deprecations, new versions of open source components, and bug fixes.

The software distribution process for an upgrade is the same as the one used for an initial installation.

Software is packaged in RPM format and distributed from a network repository or tarball. If you are not using the internet repository at software.apigee.com, you will need to update your existing mirror or create a new one with the newer version of the Apigee software.

The upgrade process upgrades the binary and data related to the open source components used in Apigee Edge.



You should ensure that you have at least 1GB of free disk space before you perform the upgrade. You will need additional space for backups.

Apigee Edge releases are certified to use specific versions of Java (JDK). If required, the JDK can be updated manually before you perform the upgrade, or it can be updated as part of the upgrade process.

It is recommended to perform a complete backup of all nodes before upgrading the Apigee Edge software.

You can use the documented procedures for your current version of Apigee Edge to perform the backup.

It is also important to ensure that Apigee Edge is up and running during the upgrade process. You can check the status of Apigee services by running the apigee-all status command on each node used in the installation.

Upgrade dependencies: EPEL repository

Enable EPEL repo

You must enable Extra Packages for Enterprise Linux (or EPEL) to update Edge or to create a local repository.

The command you use depends on your version of RedHat/CentOS/Oracle Linux:

- For RedHat/CentOS/Oracle 7.x:
  ```
  sudo yum install
  https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
  ```
- For RedHat/CentOS/Oracle 6.x:
  ```
  sudo yum install
  https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm
  ```

https://fedoraproject.org/wiki/EPEL

The Apigee software depends on packages from your distributors' repositories and the EPEL repository.

You can provide access to your distributors' repositories through the internet, a proxy, or a local mirror.

EPEL can also be accessed remotely or through a local mirror.

More information on accessing EPEL is provided at the link provided.

Agenda

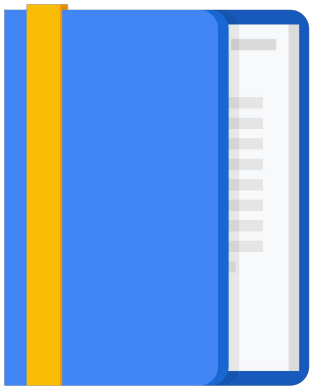
Upgrade planning

Prerequisites and dependencies

Upgrade process

Lab

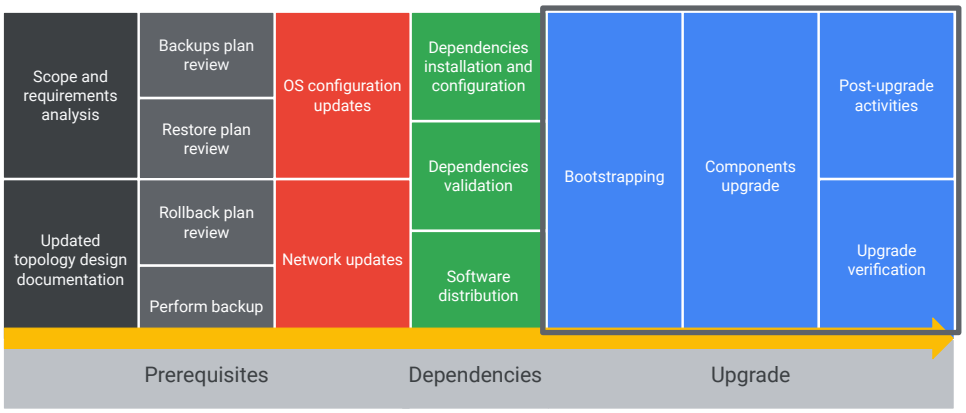
Rollback process



In this lesson, we discuss the process you use to upgrade the Apigee Edge private cloud platform.

Upgrade process

Creating an upgrade runbook



The upgrade process involves bootstrapping, upgrading the Apigee private cloud components, performing any post-upgrade activities, and verifying the upgrade.

The process upgrades the components to their latest versions. As the Apigee Edge software is distributed via RPM packages, the upgrade must be executed by a user with root or the required level of access privileges.

When an upgrade takes place, any changes you have made to the /opt/apigee/customer directory are retained. This ensures that any local configuration updates that you made are not overwritten.

Upgrade prerequisites

Before performing the upgrade:

- To ensure that all Apigee private cloud services are running, execute the following command on all nodes:

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

- Stop the Cassandra repair job (disable cron job).
- Review the official documentation and release notes for specific considerations related to the upgrade path and target version.

  https://docs.apigee.com/release/notes/apigee-release-notes

Note that all Apigee services must be up and running during the upgrade. If any services are down or unavailable, do not start the upgrade until they are back online.

If you are performing an upgrade during the window in which your weekly scheduled Cassandra node tool repair maintenance job runs, be sure to disable that job temporarily while you perform the upgrade.

Also, make sure to review the release notes for any specific instructions related to the upgrade path and the target version you are upgrading to.

Upgrade bootstrapping

As with the installation process, a bootstrap script is used to begin the upgrade.

Download the bootstrap script:

```
curl https://software.apigee.com/bootstrap_<version>.sh -o
/tmp/apigee/bootstrap_<version>.sh
```

Execute the bootstrap script:

```
sudo bash /tmp/apigee/bootstrap_<version>.sh apigeeuser=<uName>
```

With the planning and preparation for the upgrade completed, it's time to start the upgrade process.

We begin with the bootstrapping process. The bootstrapping process for upgrading Apigee for private cloud is identical to the process we used during the initial installation.

Bootstrapping configures a host to point to a specific release of Apigee. To point to a new version, you will simply bootstrap using the correct script for your desired release.

In the bootstrap version, you can specify only the major and minor version of Apigee Edge and not a specific patch. You will always receive the latest patch release for the version.

The bootstrap script will configure the Apigee software repository, so if you're using a local repository mirror or tarball, be sure to provide the same arguments you used at installation time.

You start by downloading the bootstrap script from the desired repository. In the example shown, the repository hosted by Google at software.apigee.com is used.

In your environment, you may be downloading the script from your local network repository mirror. If you are using the tarball software distribution method, the bootstrap script will already be available on your local file system at the location to which you decompressed the tarball.

The bootstrap script is downloaded to the /tmp/apigee directory using the curl command.

Note the version string embedded in the file name. Each release has its own bootstrap script, so be sure to download the correct script for your desired release.

The next step is to execute the bootstrap script. You can either make the script executable or prefix the script path with bash as shown here. Arguments to the script vary, but if you're using the Apigee hosted repository at software.apigee.com, you'll need to provide at least the Apigee user argument.

You can also provide the Apigee password argument on the command line for non-interactive use, but in the example, the script will prompt us for the correct password.

Upgrade installation utilities

The following commands must be run on all nodes and can be executed concurrently:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-setup update
```

Additionally, on the management server nodes, update the following utilities:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-provision
update
```

```
/opt/apigee/apigee-service/bin/apigee-service apigee-validate
update
```

When you performed the initial installation, you finished the bootstrapping process by installing the Apigee setup utility. When the host is bootstrapped again during the upgrade process, that utility will need to be upgraded before proceeding with the rest of the upgrade.

To upgrade it, run apigee-service apigee-setup update.

Finally, the organization-provisioning and post-installation validation utilities we installed during the installation process need to be upgraded.

These utilities will generally only exist on one management server, but they should be upgraded anywhere they're installed.

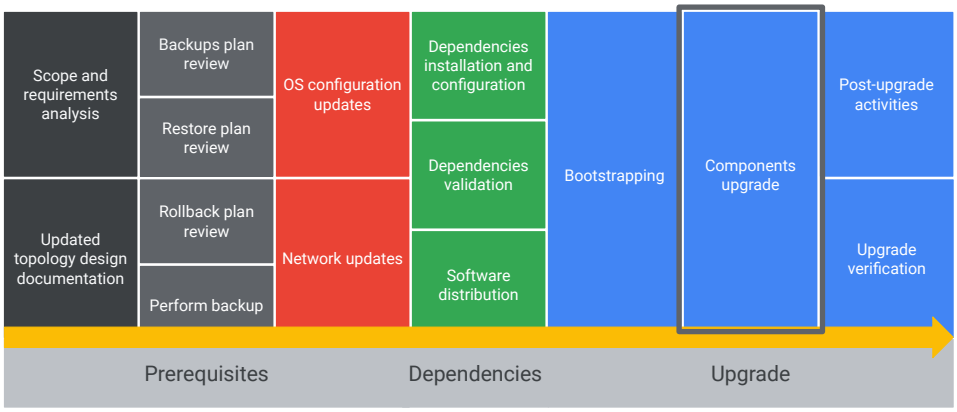
To upgrade the organization-provisioning utility, run apigee-service apigee-provision update.

To upgrade the validation utility, run apigee-service apigee-validate update.

At this point, you have successfully bootstrapped the nodes with the new version of the Apigee software.

Upgrade process

Key activities for the upgrade process



Now that the bootstrapping process for the upgrade is completed, you are ready to begin upgrading the components of the system.

Upgrade components

Upgrade profiles are component-based.

Use the following command to upgrade components:

```
/opt/apigee/apigee-setup/bin/update.sh
-c component -f configFile
```

Component	Description
ldap	Openldap
cs	Cassandra
zk	ZooKeeper
qpid	qpidd
ps	postgresql
edge	All components except Edge UI: Management Server, Message Processor, Router, QPID Server, Postgres Server
ui	Edge UI
sso	Edge SSO
ue	New Edge Experience
dp	Developer portal

The installation and upgrade processes are similar in some ways. For example, both use scripts included with the apigee-setup utility and use the same configuration file.

An important difference between the two processes is that the installation process operates on profiles, while the upgrade process operates on components.

Remember that a profile such as DS, or data store, or RMP for routers and message processors, can bundle together multiple components.

Because components from more than one profile can be installed together on a single node, the upgrade process enforces ordering by targeting specific components, based on their dependencies.

For example if you upgrade a node which has the management server, UI and Openldap components, the order of upgrade based on the dependency of the components will be Openldap first, then the management server, and finally the UI.

The table shows a list of the components you can upgrade. Note that each of the open source components has a dedicated component name. These components are normally upgraded first in the correct upgrade order.

There is also an edge component which includes any Apigee Edge component other than the UI. The UI component targets just the UI, and the DP component is for the

developer portal.

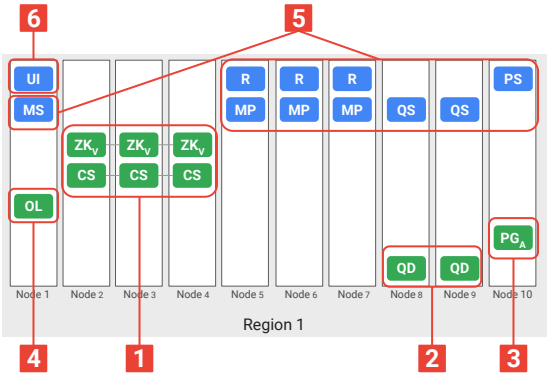
The update script will upgrade the RPMs to the newer software release, update service configurations and update or migrate data if necessary, and restart services.

In some cases, the only action will be to upgrade the RPMs and restart services. For components such as Cassandra or PostgreSQL, a data update or migration may occasionally be necessary, which is automatically carried out.

Upgrade order: single region

Component upgrade order:

1. Cassandra (CS) and ZooKeeper (ZK)
2. Qpid (QD)
3. PostgreSQL (PG)
4. Openldap (OL)
5. Edge components in the following order:
   - Qpid server (QS)
   - Postgres server (PS)
   - Management Server (MS)
   - Router and Message Processor (R, MP)
6. Enterprise UI (UI)



The ordering of component upgrades can vary from release to release.

If underlying database components such as Cassandra or PostgreSQL are upgraded to a new upstream release, the ordering may change slightly, so be sure to check the release notes and documented upgrade procedure for details.

The typical ordering is to upgrade Cassandra and ZooKeeper first, Qpid second, PostgreSQL third, and Openldap fourth.

With the open source components upgraded first, the Edge profile can be applied next in order on the Qpid servers, Postgres servers, management server, routers and message processors, and finally the enterprise UI.

Sample configuration file: single region

- The upgrade process reuses the same configuration files used during installation.
- No changes are required if the topology, placement of components, and password remain the same as the original installation.
- Any changes are explicitly described as part of official documentation.

```
IP1=<node 1>
IP2=<node 2>
IP3=<node 3>
IP4=<node 4>
IP5=<node 5>

HOSTIP="$(hostname -i)"
MSIP="$IP1"
ADMIN_EMAIL="<email@example.com>"
APIGEE_ADMINPW="<password>"
LICENSE_FILE="/tmp/apigee/license.txt"
USE_LDAP_REMOTE_HOST="n"
LDAP_TYPE="1"
APIGEE_LDAPPW="<password>"
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 $IP3:1,1"
PG_MASTER="$IP4"
PG_STANDBY="$IP5"
SKIP_SMTP="y"
SMTPHOST="smtp.example.com"
SMTPPORT="25"
SMTPUSER="smtp@example.com"
SMTPPASSWORD="<password>"
SMTPSSL="n"
BIND_ON_ALL_INTERFACES="y"
```

The configuration file used for upgrading must be identical to the one used for installation.

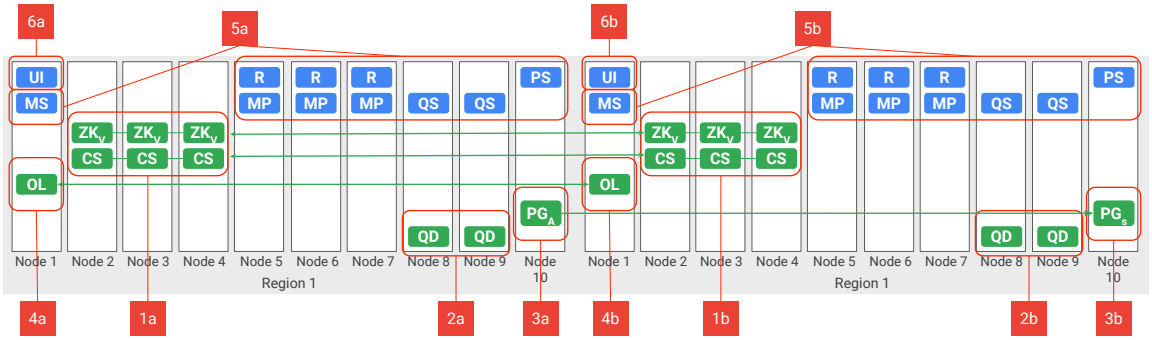
Any differences in the configuration file will overwrite the installation variable values and may cause an outage on the platform.

You should keep the configuration file in a secure place that you can reuse for future upgrades. It is considered a best practice to version-control the configuration files and keep them in a private repository. Because passwords are sensitive, they should be stored separately and inserted into the configuration file before the upgrade activities.

On occasion, a new variable may be added to the configuration file.

The release notes for each release will clearly identify any changes to the configuration file so that you can update it if needed.

Upgrade order: multi-region



Component upgrade order:

When upgrading two or more regions, the upgrade order remains the same by component type.

In a multi-region installation, the order in which component types are upgraded remains the same as in a single-region installation.

Each component should be upgraded in order across all regions before continuing to the next component.

The diagram shows that all Cassandra and ZooKeeper components in both regions are upgraded first (in steps 1a and 1b), followed by all Qpid components in both regions, and so on.

For steps 5a and 5b, use the following order: Qpid servers, Postgres Server, Management Server and finally the Routers and Message Processors.

Sample configuration files: multi-region

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

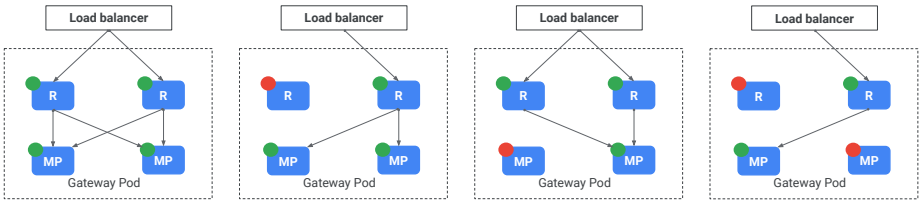
```
HOSTIP="$(hostname -i)"
MSIP="$DC2IP1"
ADMIN_EMAIL="<email@example.com>"
APIGEE_ADMINPW="<password>"
LICENSE_FILE="/tmp/apigee/license.txt"
USE_LDAP_REMOTE_HOST="n"
LDAP_TYPE="2"
LDAP_SID="2"
LDAP_PEER="$DC1IP1"
APIGEE_LDAPPW="<password>"
MP_POD="gateway"
REGION="dc-2"
ZK_HOSTS="$DC1IP2 $DC1IP3 $DC1IP4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

Here is an example of a sample configuration file used in a multi-region installation of Apigee Edge for private cloud.

The entries shown in red are region-specific name-value pairs.

Zero downtime upgrade considerations

Router and Message Processor accessibility



All R and MPs are accessible.

R access is disabled. Load Balancer should stop sending traffic to the unreachable R. Load Balancer configuration is done by the customer.

MP access is disabled. Both Rs will automatically stop sending traffic to the unreachable MP.

R and MP access is disabled. Only a single path exists, but the traffic still flows.

Note that the upgrade process causes each component to be unavailable for a brief period of time.

With access to one of the routers disabled, the load balancer will direct traffic to the available routers on the platform.

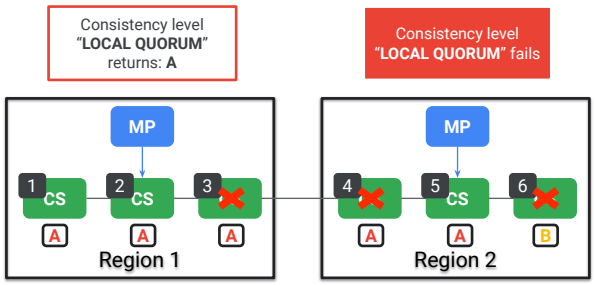
With a message processor disabled, all available routers forward API traffic to the available message processors.

In order to minimize the downtime for runtime traffic, you should disable access to the router and message processor components before starting the upgrade process on those components.

With one pair of router and message processors still available, runtime API traffic can still flow through the platform.

Zero downtime upgrade considerations

Cassandra availability



Cassandra availability is based on the number of available replicas given the consistency level of the query operation being executed.

- Local Quorum = replication_factor / 2 + 1
- Local Quorum = 3/2 + 1 = 2 required

As discussed in the Architecture module, Cassandra availability is based on the number of replicas (data copies) and required consistency level of the query operation.

Apigee uses LOCAL_QUORUM as the highest consistency level because Apigee regions are designed to be self-sufficient.

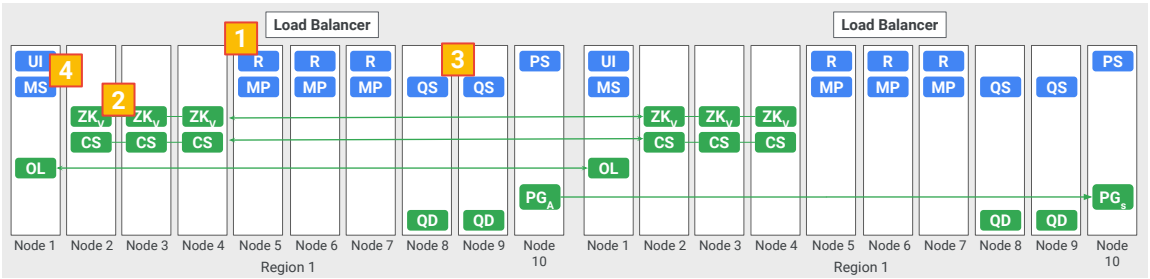
When the consistency level is LOCAL_QUORUM, more than 50% of the data copies in one data center should be available for the transaction to succeed.

To execute a zero downtime upgrade, you should ensure that the LOCAL_QUORUM is maintained during the entire upgrade process.

Zero downtime upgrade considerations

Capacity requirements

1. Adequate capacity and availability to handle runtime API traffic during R and MP restart
2. Adequate capacity to handle runtime traffic generated by API calls using Cassandra-related policies
3. Adequate availability to handle API analytics data during Qpid restart
4. Adequate capacity to handle management API and UI calls during UI, MS, and OL restart



Another important aspect of a zero downtime upgrade is to consider capacity.

As we stop the components one at a time, we have to make sure that the rest of the components of the same type can handle the load of the current runtime API traffic.

Can 2 out of the 3 message processors handle the current amount of traffic in the first data center?
If not, we should consider rescheduling the upgrade activity for a later time when a reduced amount of traffic is flowing through the platform, or redirect the traffic to another region.

The same consideration must be made for the Cassandra nodes, because they can be part of the runtime API traffic.

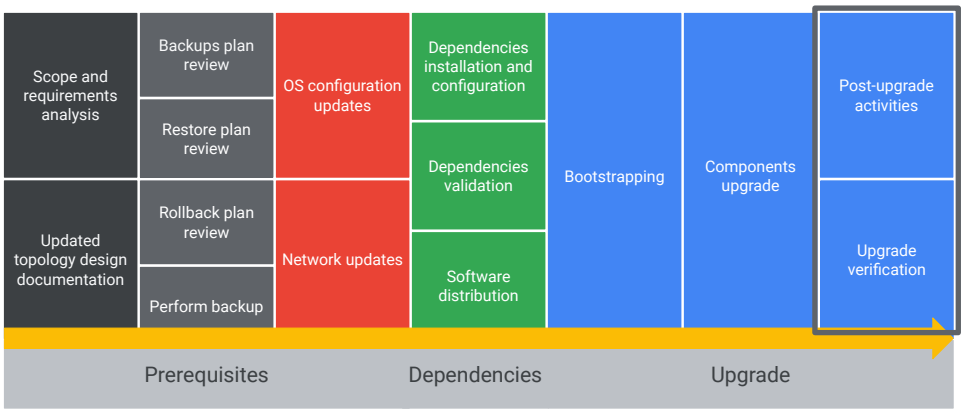
Although analytics and management services are not considered on the critical API runtime path, they can play an important role in reporting functions or SDLC, so we should also consider their capacity during upgrades.

A full list of upgrade considerations is on the Apigee documentation website.

Link:
https://docs.apigee.com/private-cloud/v4.50.00/update-apigee-edge-4190x-45000#zerodowntimeupdate

Upgrade planning

Key activities for the upgrade process



Let’s now discuss some of the post-upgrade activities, including upgrade verification.

Post-upgrade

After the upgrade:

- To ensure that Apigee Edge is up and running, execute the command:

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

- Start the Cassandra repair job (enable cron job).
- Perform upgrade verification.

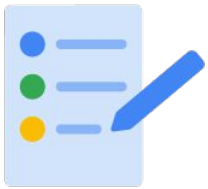
Following the upgrade, you should run the apigee-all status script to ensure that all services are up and running on each node in the cluster.

You will also need to re-enable the Cassandra node tool repair maintenance job if you temporarily disabled it for the upgrade.

Finally, you should perform post-upgrade validation and testing as described in the next section.

Upgrade verification

- Run traffic during the upgrade.
- Monitoring should show a clean system status after the upgrade is completed.
- Run apigee-all, apigee-version, and apigee-validate.
- Execute API tests.



With the upgrade complete, the final step is to verify that the upgrade was successful.

Assuming that all went well and the upgrade was completed without error, the next step is to validate that the platform was correctly upgraded and is functioning normally.

The Apigee platform is designed to be upgraded without experiencing downtime. During the upgrade, all services should be up, and you should run real or test traffic continuously.

You should have monitoring in place to notify you when your infrastructure or the Apigee platform is unhealthy. You will probably receive some alerts during the upgrade process as services are restarted, but following the upgrade, you should check your monitoring platform to ensure that all alarms have been resolved.

It is a good idea to run apigee-all status after you upgrade each host to verify that services are up.

You can also run apigee-all version to see the version numbers of each installed apigee software package. In many cases, the release version, such as 4.50.00, will be directly shown in the package version.

For open source components, you may need to check the supported software versions referenced in the Apigee private cloud release documentation to confirm that the correct software version is present.

You can optionally run apigee-validate to ensure that the management API and underlying services are functioning normally. For details on how to run apigee-validate, review the installation module in the on-premises installation and fundamentals course.

Finally, you should run your API test suites after the installation is complete, both to test the health of the platform and to catch any regressions or incompatibilities in API proxy code between versions.

This is particularly important in non-production environments because you will want to identify and fix any issues before upgrading production.

Agenda

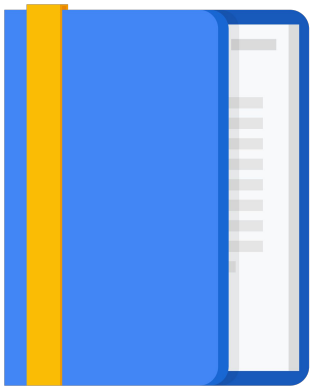
Upgrade planning

Prerequisites and dependencies

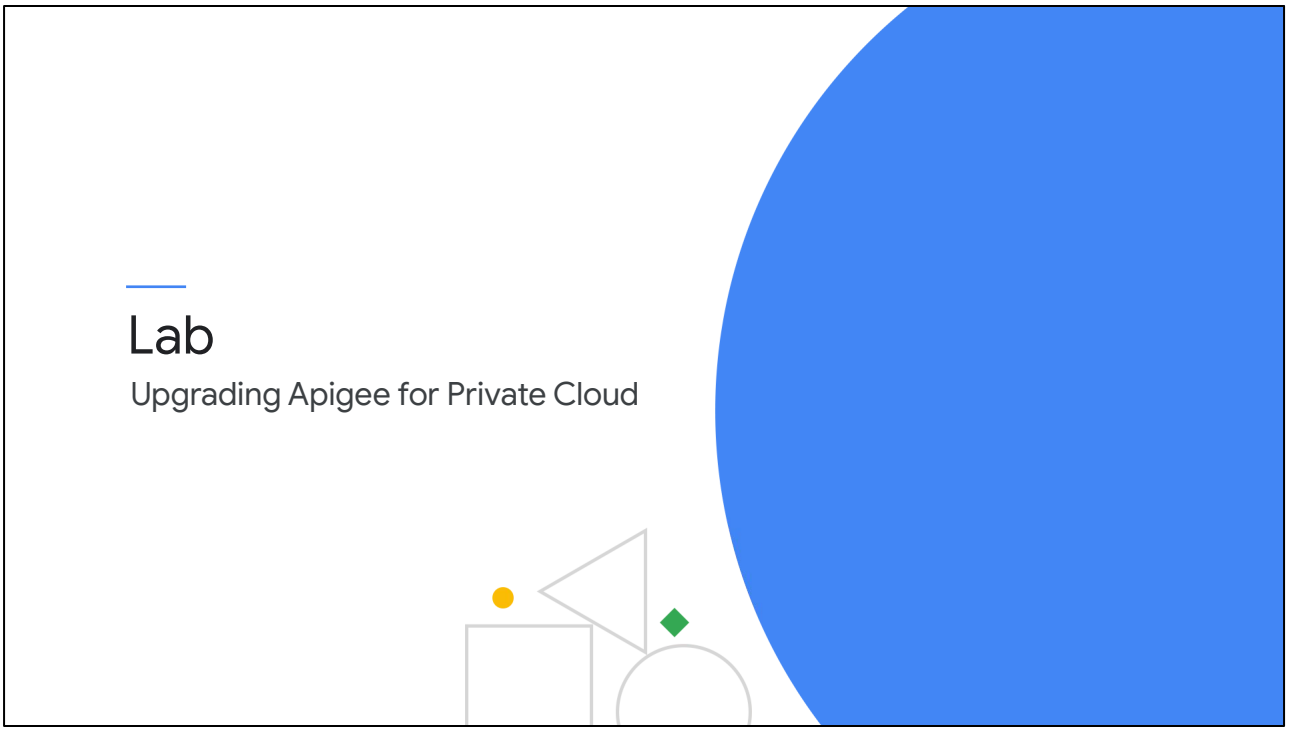
Upgrade process

Lab

Rollback process



We will now complete a lab on upgrading Apigee Edge for private cloud.



In this lab, you upgrade the Apigee Edge for private cloud API platform using a 5-node topology in a single region on Google Compute Engine.

You first bootstrap the upgrade by downloading and upgrading the utility scripts required to upgrade the platform.

You then upgrade the open source components, followed by the Apigee Edge components.

Finally, you verify that the upgrade was successful by checking the status and version of all the components.

You also validate the platform by using the apigee-validate utility and verify that a test API proxy functions as expected.

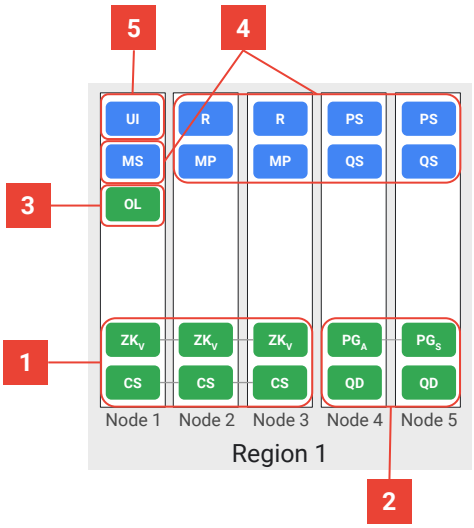
Edge upgrade lab: Scope

Scope:

- Single region
- 5-node Apigee Edge Install
- VMs instances running CentOS
- Upgrade from 4.19.06 to 4.50.00

Steps overview:

1. Bootstrapping
2. Component upgrade
3. Upgrade verification



In this lab you use a 5-node installation of Apigee Edge for private cloud in a single region.

The high-level steps include Bootstrapping the upgrade, upgrading the Apigee components, and verifying the upgrade.

The bootstrap process can be completed on all nodes at the same time, but the components are upgraded one node at a time.

Bootstrapping

On all nodes, download and execute the Edge bootstrap.sh:

```
curl -s https://software.apigee.com/bootstrap_4.50.00.sh
-o /tmp/apigee/bootstrap_4.50.00.sh
```

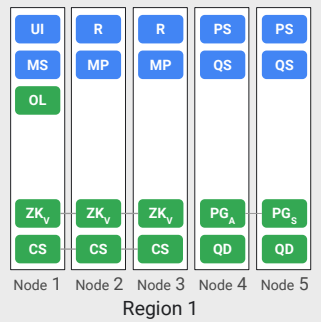
Verification:

```
cat /tmp/apigee/bootstrap_4.50.00.sh
```

```
sudo bash /tmp/apigee/bootstrap_4.50.00.sh
apigeeuser=<username> apigeepassword=<password>
```

Verification:

```
sudo cat /etc/yum.repos.d/apigee.repo
sudo yum -v repolist 'apigee*'
sudo rpm -qa | egrep "(apigee-|edge-)"
```



The first step in the upgrade process is to download and run the bootstrap script for the new Apigee private cloud version on all nodes in the cluster.

Each Apigee Edge version has its own bootstrap script.

In our lab we use software.apigee.com as the Apigee private cloud repository source for downloading the script..

Access to the repository is not public, so you will need credentials, which will be provided as part of the lab. The credentials are on a VM in a text file in the /tmp/apigee/ directory.

After successful execution of the bootstrap script, you should have a configured YUM repository and an upgraded apigee-service utility.

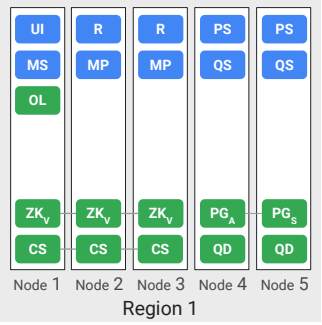
Bootstrapping

On all nodes, upgrade the apigee-setup and apigee-adminapi utilities:

```
/opt/apigee/apigee-service/bin/apigee-service apigee-setup update
/opt/apigee/apigee-service/bin/apigee-service apigee-adminapi update
```

Verification:

```
ls /opt/apigee
```



The next step is to upgrade the apigee-setup and apigee-adminapi utilities on all nodes in the cluster.

This is achieved with the help of the apigee-service tool.

In the lab, you also upgrade the apigee-provision and apigee-validate utilities at this time.

Components upgrade

1. On nodes 1, 2, and 3, upgrade Cassandra and ZooKeeper:

   ```
   sudo /opt/apigee/apigee-setup/bin/update.sh -f
   /tmp/apigee/edge-configuration.txt -c cs,zk
   ```

2. On nodes 4 and 5, upgrade qpidd and postgreSQL:

   ```
   sudo /opt/apigee/apigee-setup/bin/update.sh -f
   /tmp/apigee/edge-configuration.txt -c qpid,ps
   ```

3. On node 1, upgrade Openldap:

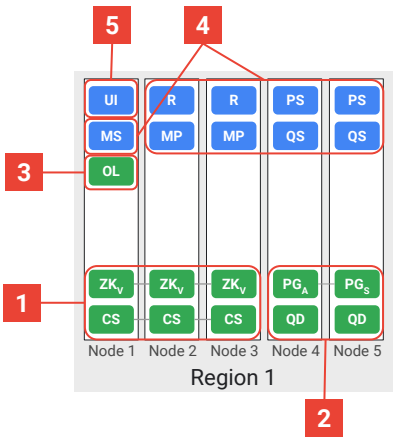
   ```
   sudo /opt/apigee/apigee-setup/bin/update.sh -f
   /tmp/apigee/edge-configuration.txt -c ldap
   ```

4. On all nodes (in order: 4, 5, 1, 2, 3), upgrade Edge components:

   ```
   sudo /opt/apigee/apigee-setup/bin/update.sh -f
   /tmp/apigee/edge-configuration.txt -c edge
   ```

5. On node 1, upgrade the UI:

   ```
   sudo /opt/apigee/apigee-setup/bin/update.sh -f
   /tmp/apigee/edge-configuration.txt -c ui
   ```



The next step is to upgrade the Apigee private cloud components in the following order:

First, upgrade Cassandra (CS) and ZooKeeper (ZK) components on nodes 1, 2 and 3.

Second, upgrade the Apache Qpid (QD) and PostgreSQL (PG) components on nodes 4 and 5.

Third, upgrade the Openldap (OL) component on node 1.

Next, you upgrade the Apigee Edge components in the following order:

You first upgrade the Qpid and Postgres server components on nodes 4 and 5.

Then upgrade the Management Server on node 1, followed by the Router and Message Processor components on nodes 2 and 3.

Finally, you upgrade the Enterprise UI component on node 1.

Upgrade verification

Verifying the upgrade involves the following practices:

- To confirm that Apigee Edge is up, execute the following command on all nodes:
  `/opt/apigee/apigee-service/bin/apigee-all status`
- To verify the software release version of components, execute the following command on all nodes:
  `/opt/apigee/apigee-service/bin/apigee-all version`

  Open source component versions are listed on the supported software page:
  https://docs.apigee.com/release/supported-software
- Perform functional validation:
  - Confirm that API traffic remains active.
  - Confirm that analytics data is being captured.

After upgrading the components, check the status of the processes by using the apigee-service or apigee-all utilities.

You should also verify the version of the components to confirm that they have been upgraded.

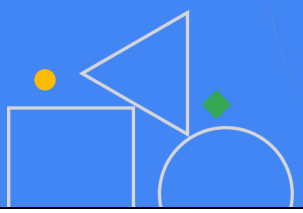
All Apigee Edge components follow the <major>.<minor>.<patch> release version string, but the open source components follow their own versioning schemes.

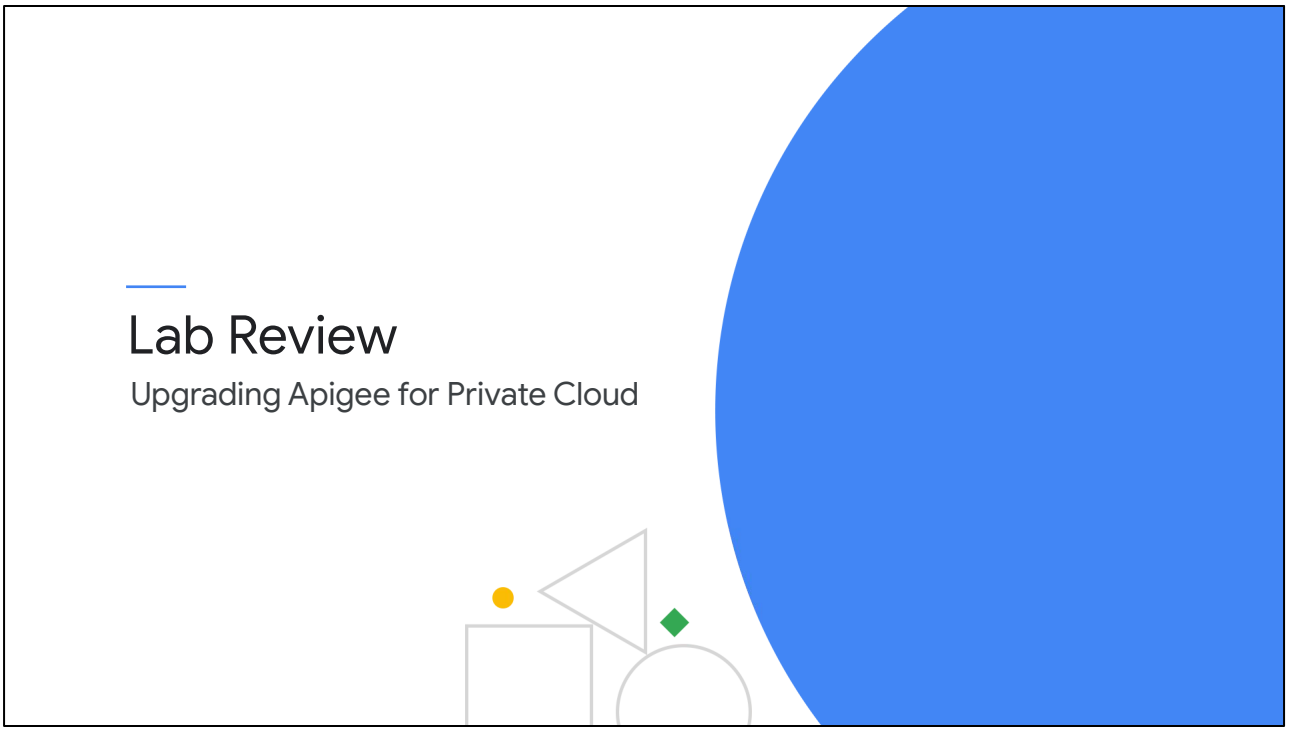
The correct version of the open source components is on the Apigee documentation site at the link provided.

It is also a good practice to check that API traffic continues to work as expected and analytics data is available after the upgrade.

Lab

Upgrading Apigee for Private Cloud





In this lab, you upgraded the Apigee API platform for private cloud.

You first upgraded the utility scripts that are required to upgrade the platform.

You then upgraded the open source components, followed by the Apigee Edge components.

You verified that the upgrade was successful by checking the status and version of all the components.

You also validated the platform by using the apigee-validate utility and verified that a test API proxy functions as expected.

Agenda

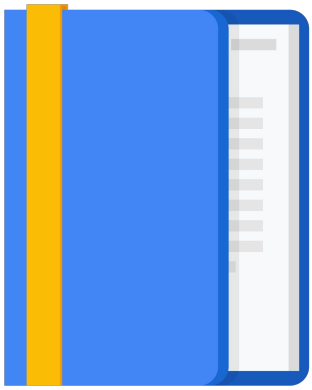
Upgrade planning

Prerequisites and dependencies

Upgrade process

Lab

Rollback process



In this lesson, we discuss the rollback process used in Apigee for private cloud.

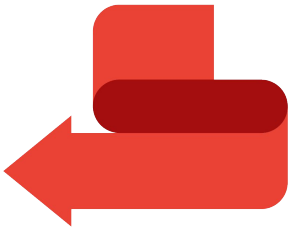
Rollback process overview

There are two scenarios where you might want to perform a rollback:

- Roll back to an older release.
- Roll back to an older version of the same release.

Components that can be rolled back:

- Management Server
- Router
- Message Processor
- Qpid Server
- Postgres Server



In rare situations, you may need to roll back components if the upgrade fails.

In the event of an error during an update to an Edge component, you can roll back the component that caused the error and then try the update again.

This is true for most upgrades between most versions. Please consult the documentation about your specific upgrade path, because some exceptions may exist.

There are two scenarios in which you may want to roll back: rolling back to an older release, and rolling back to an older version of the same release.

In both cases, the rollback process is similar.

It is recommend that you roll back only the Google built stateless components and leave the stateful databases on their current versions.

Rollback steps

Follow these steps to roll back a component:

1. Stop the service component:

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

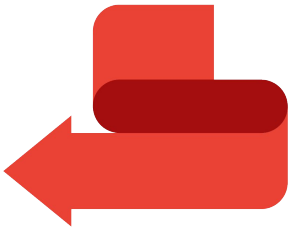
2. Uninstall the component:

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

3. Uninstall the apigee-setup utility:

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

4. Download and re-bootstrap to the previous release version.
5. Re-install the previous release version of apigee-setup.
6. Re-install and set up the previous version of the component.
7. Verify the rollback.



The steps to roll back a component to a previous release version are:

Stop and uninstall the existing component by using the apigee-service utility.

Uninstall the apigee-setup utility, re-bootstrap to the previous release, and then reinstall apigee-setup.

Finally, set up the previous version of the component by using the apigee-setup utility and the component profile as discussed in the installation module.

When the component is installed and running, verify that the rollback was successful as described in the previous section.

Official documentation

- The Apigee for private cloud official documentation or release notes **may describe additional steps** needed to upgrade between specific versions as a prerequisite for a new version.

Upgrade Apigee Edge for Private Cloud:

https://docs.apigee.com/private-cloud/latest/upgrade-paths

Apigee release notes:

https://docs.apigee.com/release/notes/apigee-release-notes

The Apigee for private cloud official documentation or release notes **may describe additional steps** needed to upgrade between specific versions as a prerequisite for a new version.

We **strongly recommend that you review this documentation** as part of upgrade planning, and incorporate any required items when creating runbooks for this task.

In this module, we discussed the planning and preparation process and the prerequisites and dependencies required to perform an upgrade of the Apigee private cloud platform.

You learned how to perform the upgrade and completed a lab to upgrade Apigee Edge for private cloud on Google Compute Engine.

We also discussed post upgrade validation and the rollback process.


Google Cloud