API Gateway

Version 7.6.2

14 July 2020

Apache Cassandra Administrator Guide

This documentation describes the following Axway software:

Axway API Gateway 7.6.2

Contents

Preface

This guide explains how to configure and manage the Apache Cassandra database for API Gateway and API Manager.

Who should read this guide

The intended audience for this guide is API Gateway administrators.

How to use this guide

This guide should be used in conjunction with the other guides in the API Gateway documentation set.

Before you begin, review this guide thoroughly. The following is a brief description of the contents of each section:

- *Cassandra deployment architectures* on page 9 – Provides an overview of the Apache Cassandra deployment architectures supported by API Gateway.
- *Configure a highly available Cassandra cluster* on page 15 - Describes how to set up an Apache Cassandra database cluster for high availability (HA) of your API Gateway system.
- *Apache Cassandra best practices* on page 22 - Describes Apache Cassandra best practices to achieve a stable environment, and to prevent data integrity and performance issues.
- *Manage Apache Cassandra* on page 25 - Describes how to start or stop Apache Cassandra manually or as a service.
- *Apache Cassandra backup and restore* on page 28 - Explains how to use scripts to create and restore an Apache Cassandra snapshot backup and also describes what configuration to back up.
- *Perform essential Apache Cassandra operations* on page 37 - Describes the minimum essential operations that are required to maintain a healthy Apache Cassandra HA cluster.

Related documentation

The AMPLIFY API Management solution enables you to create, publish, promote, and manage Application Programming Interfaces (APIs) in a secure and scalable environment. For more information, see the *AMPLIFY API Management Getting Started Guide*.

The following reference documents are also available on the Axway Documentation portal at https://docs.axway.com:

- *Supported Platforms*

  Lists the different operating systems, databases, browsers, and thick client platforms supported by each Axway product.

- *Interoperability Matrix*

  Provides product version and interoperability information for Axway products.

Support services

The Axway Global Support team provides worldwide 24 x 7 support for customers with active support agreements.

Email support@axway.com or visit Axway Support at https://support.axway.com.

See "Get help with API Gateway" in the *API Gateway Administrator Guide* for the information that you should be prepared to provide when you contact Axway Support.

Training services

Axway offers training across the globe, including on-site instructor-led classes and self-paced online learning. For details, go to: http://www.axway.com/support-services/training

Accessibility

Axway strives to create accessible products and documentation for users.

This documentation provides the following accessibility features:

Screen reader support

- Alternative text is provided for images whenever necessary.
- The PDF documents are tagged to provide a logical reading order.

Support for high contrast and accessible use of colors

- The documentation can be used in high-contrast mode.
- There is sufficient contrast between the text and the background color.
- The graphics have the right level of contrast and take into account the way color-blind people perceive colors.

Cassandra deployment architectures 1

This topic describes the Cassandra deployment architectures supported by API Gateway. The following table provides a summary:

Deployment	Description	Use
Standalone	One API Gateway instance only. This is the default.	• Development environment • Production environment with one API Gateway instance
High availability with local storage	Multiple API Gateway instances in a group, each API Gateway instance on different hosts, with a Cassandra storage node local to each API Gateway host.	• Pre-production environment • Production environment
High availability with remote storage	Multiple API Gateway instances in a group, each API Gateway instance on different hosts, with Cassandra storage nodes on remote hosts.	• Pre-production environment • Production environment

Note You can use one Cassandra cluster to store data from one or multiple API Gateway groups and one or multiple API Gateway domains. Your Cassandra topology does not need to match your API Gateway topology.

Multiple API Gateway groups can also be deployed on the same host, each host running an API Gateway instance for each group. This applies to both local and remote storage.

Note Windows is supported only for a limited set of developer tools. API Gateway and API Manager do not support Windows.

You can continue using an existing remote Cassandra installation running on Windows that you have configured with your API Gateway installation.

For details on hosting Cassandra in datacenters, see "Configure API Management in multiple datacenters" in the *API Gateway Installation Guide*.

Cassandra in standalone mode

Cassandra is configured in non-HA standalone mode when deployed alongside a single API Gateway instance on the same host (for example, in a demonstration or development environment with one API Gateway instance). This is the default mode.

To configure Cassandra in standalone mode, perform the following steps:

1. Ensure Cassandra is installed on the local node (along with API Gateway). For details, see *Install an Apache Cassandra database* in the *API Gateway Installation Guide*.
2. Start Cassandra before starting API Gateway. For details, see *Manage Apache Cassandra* on page 25.
3. Start API Gateway. For details, see the *API Gateway Installation Guide*.

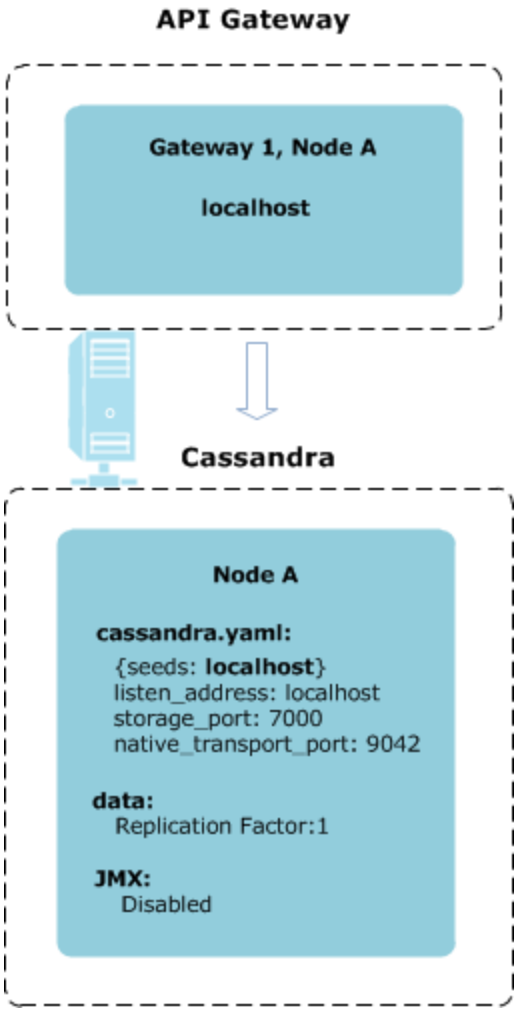
The next steps depend on your installation setup type:

- In a Standard or Complete setup that includes the QuickStart tutorial, the default configuration attempts to connect to Cassandra running on `localhost`.
- In a Custom setup without the QuickStart tutorial, you must configure the connection in Policy Studio. For more details, see *Connect to API Gateway for the first time* on page 26.

To use Cassandra with API Manager, see *Configure a highly available Cassandra cluster* on page 15. This is configured by default when API Manager is installed along with the QuickStart tutorial.

To use Cassandra with OAuth, see "Deploy OAuth configuration" in the *API Gateway OAuth User Guide*.

The following diagram shows Cassandra in standalone mode with a default Standard setup:



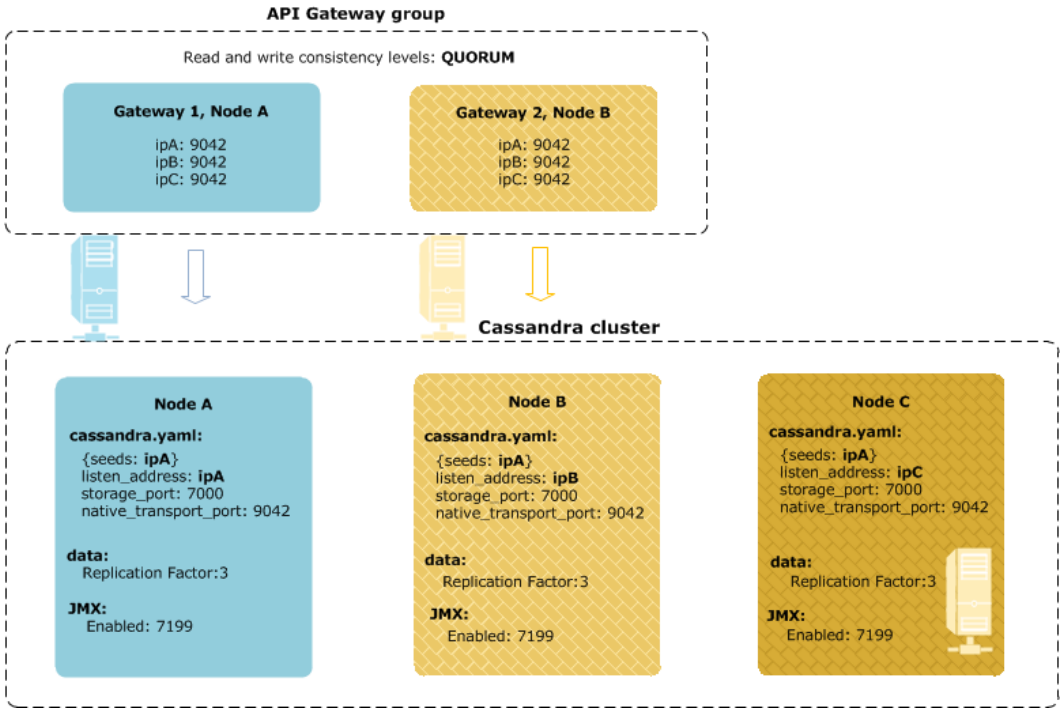
Cassandra in High Availability mode

For both local and remote Cassandra HA, Cassandra runs on multiple hosts. This section describes both scenarios.

HA with local storage

In local Cassandra HA, Cassandra runs alongside API Gateway on the same host. This means that you do not need to provision separate host machines for Cassandra and API Gateway, or open ports in your firewall. However, the data will be stored in the DMZ. Local Cassandra HA enables minimum cost of ownership.

The following diagram shows local Cassandra HA mode:



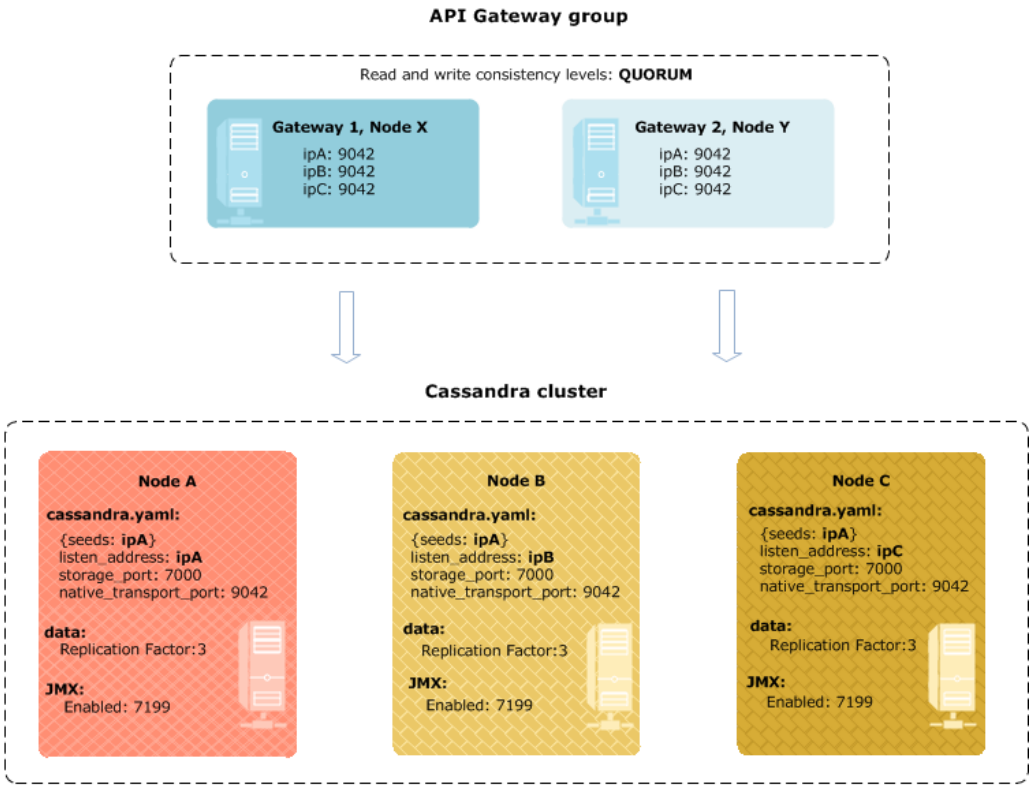
Note In this example, one of the Cassandra nodes runs without an API Gateway instance on the same host. This is because the minimum deployment architecture includes two API Gateway hosts and three Cassandra hosts. If you have three API Gateway instances, all Cassandra nodes are local.

HA with remote storage

In remote Cassandra HA, Cassandra runs on a different host from API Gateway. The main differences when installing and configuring remote Cassandra are:

- You must provision separate host machines for Cassandra and API Gateway. However, the data can be stored outside the DMZ, and there might be improved performance.
- You might need to open ports in the firewall to connect to Cassandra outside the DMZ. For more details, see *Configure a highly available Cassandra cluster* on page 15.
- You do not have to use the Cassandra component supplied by the API Gateway installer.
- You can configure the remote node using the `setup-cassandra` script supplied by the API Gateway installation. For more details, see *setup-cassandra script reference* on page 40. Alternatively, you can perform all necessary Cassandra configuration changes manually.
- You must update the API Gateway Cassandra client settings in Policy Studio to connect to the remote Cassandra host nodes.

The following diagram shows remote Cassandra HA mode:



Cassandra HA configuration

You can use a local or remote Cassandra HA configuration. The example Cassandra HA configuration in the diagrams consists of the following:

- Three Cassandra nodes in a single cluster.
- Three host machines with network addresses: `ipA` (seed node), `ipB`, and `ipC`.
- Replication factor of `3`. Each node holds 100% of the data.
- Default values in `cassandra.yaml` for:
    - Server-server communication:
      `listen_address` set to IP address
      `storage_port` set to `7000`
      `ssl_storage_port`: `7001` (when TLS v1.2 is configured)
    - Client-server communication:
      `native_transport_port` of `9042`
    - JMX monitoring on `localhost:7199`

Note

- `ipA`, `ipB`, and `ipC` are placeholders for real host machines on your network. You can specify IP addresses or host names.
- You must have at least one designated seed node. Seeds nodes are required at runtime when a node is added to the cluster. You can add more or change designation later.
- You can change the server-server port, but it must be the same across the cluster.
- For Cassandra administration, you must gain local access to the host machine (for example, using SSH) to perform administration tasks. This includes using `nodetool` to access the Cassandra cluster over JMX.

API Gateway configuration

API Gateway acts as a client of the Cassandra cluster as follows:

- Client failover:
  API Gateway can fail over to any of the Cassandra nodes for service. Each API Gateway is configured with the connection details of each Cassandra host.
- Strong consistency:
  Cassandra read and write consistency levels are both set to `QUORUM`. This, along with the replication factor of `3`, enables full availability in the event of one node loss.

Note You can have any number of API Gateway instances (all running either locally or remote to Cassandra). However, you must have at least two API Gateway instances for HA. This also applies to API Manager.

For more details on Cassandra HA configuration, see *Configure a highly available Cassandra cluster* on page 15.

Configure a highly available Cassandra cluster 2

This topic describes how to set up an Apache Cassandra database cluster for high availability (HA) of your API Gateway system.

Cassandra HA in a production environment

To tolerate the loss of one Cassandra node and to ensure 100% data consistency, API Gateway requires at a minimum the following Cassandra cluster configuration running in a production class environment:

- Three Cassandra nodes (with one seed node)
- `QUORUM` read/write consistency to ensure that you are reading from a quorum of Cassandra nodes (two) every time

Caution Eventual consistency is not supported in a production environment due to a risk of stale or missing data

- `Replication factor` setting set to `3`, so each node holds 100% of the data

If one Cassandra node fails or needs to be restarted, the cluster continues to operate with the remaining two nodes. This configuration applies to all supported use cases (for example, API Manager and API Gateway custom KPS, OAuth, and client registry data).

Preparing a Cassandra HA environment

The following general guidelines apply to configuring a Cassandra HA cluster:

- Decide on the number of Cassandra nodes, and install and configure the Cassandra database on each node.
- Decide on the number of API Gateway nodes, and install and configure them (local or remote to Cassandra).

Note It is recommended that you configure a Cassandra HA cluster with three Cassandra nodes, and at least two API Gateway instances (local or remote). For details, see *Cassandra deployment architectures* on page 9.

- Create a HA API Gateway environment.
- Configure the Cassandra client settings in Policy Studio for the API Gateway group.

For details on how to install an Apache Cassandra database or how to create a HA API Gateway environment, see *API Gateway Installation Guide*.

Example of a HA setup using three Cassandra nodes

This section shows how to configure a three nodes Cassandra HA cluster.

Configure the Cassandra seed node

Perform the following steps to configure a *seed* node:

1. Assign one of the nodes as the *seed* node, and connect to it via SSH.
2. Update the *CASSANDRA_HOME*/conf/cassandra.yaml file manually or using the setup-cassandra script.
   Manually:

```
seed_provider, parameters, seeds: IP_SEED_NODE
start_native_transport: true
start_rpc: false
native_transport_port: 9042
listen_address: IP_SEED_NODE
rpc_address: IP_SEED_NODE
authenticator: org.apache.cassandra.auth.PasswordAuthenticator
authorizer: org.apache.cassandra.auth.CassandraAuthorizer
```

   Using the setup-cassandra script:

```
setup-cassandra --seed --own-ip=<IP_SEED_NODE> --nodes=3 --cassandra-
config=CONFIG_FILE
```

3. Start Cassandra.
4. Verify the CASSANDRA_HOME/logs/system.log does not contain any errors or warnings.
5. Run the nodetool status command to verify that *one* node is present in UN status, and that the IP address is correct.

Configure the additional Cassandra nodes

The procedure to configure additional nodes uses the seed node that you have previously configured.

Note You must repeat these steps for each additional node.

1. Connect to the additional node via SSH, and update the *CASSANDRA_HOME*/conf/cassandra.yaml file.

Manually:

```
seed_provider, parameters, seeds: IP_SEED_NODE
 start_native_transport: true
start_rpc: false
native_transport_port: 9042
listen_address: IP_NODE_N
rpc_address: IP_NODE_N
authenticator: org.apache.cassandra.auth.PasswordAuthenticator
authorizer: org.apache.cassandra.auth.CassandraAuthorizer
```

Using the `setup-cassandra` script:

```
setup-cassandra --seed-ip=<IP_SEED_NODE> --own-ip=<IP_NODE_n> --
cassandra-config=<PATH_TO_CASSANDRA.YAML>
```

2. Start Cassandra.
3. Verify the `CASSANDRA_HOME/logs/system.log` does not contain any errors or warnings.
4. Run the `nodetool status` command to verify that the *new* node is present in `UN` status, and that the IP address is correct.

Create a new Cassandra database user

From the command line, execute the following commands to create a new user:

```
>./cqlsh <IP_NODE_1> -u cassandra -p cassandra
> ALTER KEYSPACE "system_auth" WITH REPLICATION = { 'class':
'SimpleStrategy', 'replication_factor': 3 };
> CREATE USER <USERNAME> WITH PASSWORD '<PASSWORD>' SUPERUSER;
> QUIT;
>./cqlsh <IP_NODE_1> -u <ADMIN_USERNAME> -p <PASSWORD>
> ALTER USER cassandra WITH PASSWORD '<PASSWORD>' NOSUPERUSER;
> QUIT
```

Configure the API Gateway Cassandra client connection

Note This section assumes that you have already set up a HA API Gateway environment. For details, see *API Gateway Administrator Guide*

In Policy Studio, open your API Gateway group configuration.

1. Select **Server Settings > Cassandra > Authentication**, and enter the cassandra database user name and password.
2. Select **Server Settings > Cassandra > Hosts**, and add a host for each Cassandra node in the cluster.
3. Select **Server Settings > Cassandra > Keyspace**, and set the **Initial replication factor** option to `3`.
4. Deploy the configuration to the group.

Configure the client settings for API Gateway or API Manager

Note You need at least two API Gateways in a group for HA.

This section describes the following options:

- *Configure API Gateway Cassandra client settings* on page 18
- *Configure API Manager Cassandra client settings* on page 20

Configure API Gateway Cassandra client settings

To update the Cassandra client configuration for API Gateway, perform the following steps:

Configure the API Gateway domain

1. Ensure API Gateway has been installed on the API Gateway 1 and API Gateway 2 nodes. For details, see the *API Gateway Installation Guide*.
2. Ensure an API Gateway domain has been created on the API Gateway 1 node using `managedomain`. For more details, see "Configure an API Gateway domain" in the *API Gateway Administrator Guide*.

Configure the API Gateway Cassandra client connection

1. In Policy Studio, open your API Gateway group configuration.
2. Select **Server Settings > Cassandra > Authentication**, and enter your Cassandra user name and password (both default to `cassandra`).
3. Select **Server Settings > Cassandra > Hosts**, and add an address for each Cassandra node in the cluster (`ipA`, `ipB` and `ipC` in this example).

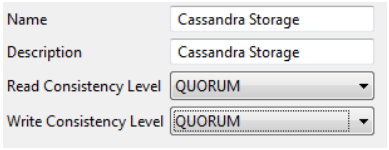
Tip You can automate these steps by running the `updateCassandraSettings.py` script against a deployment package (`.fed`). For more details, see *Configure a highly available Cassandra cluster* on page 15.

Configure the API Gateway Cassandra consistency levels

1. Ensure that the **API Server** KPS collection has been created under **Environment Configuration > Key Property Stores**. This is required to configure Cassandra consistency levels, and is created automatically if you installed the **Complete** setup type.

   If you installed the **Custom** or **Standard** setup, you must configure OAuth or API Manager settings in Policy Studio to create the required KPS collections. For more details, see:

   - "Deploy OAuth configuration" in the *API Gateway OAuth User Guide*
   - *Configure API Manager Cassandra client settings* on page 20
2. Select **Environment Configuration > Key Property Stores > API Server > Data Sources > Cassandra Storage**, and click **Edit**.
3. In the **Read Consistency Level** and **Write Consistency Level** fields, select **QUORUM**:



4. Repeat this step for each KPS collection using Cassandra (for example, **Key Property Stores > OAuth**, or **API Portal** for API Manager). This also applies to any custom KPS collections that you have created.
5. If you are using OAuth and Cassandra, you must also configure quorum consistency for all OAuth2 stores under **Libraries > OAuth2 Stores**:
   - **Access Token Stores > OAuth Access Token Store**
   - **Authorization Code Stores > Authz Code Store**
   - **Client Access Token Stores > OAuth Client Access Token Store**

Note By default, OAuth uses EhCache instead of Cassandra. For more details on OAuth, see the *API Gateway OAuth User Guide*.

Deploy the configuration

1. Click **Deploy** in the toolbar to deploy this configuration to the API Gateway group.
2. Restart each API Gateway in the group.

For details on any connection errors between API Gateway and Cassandra, see *Configure a highly available Cassandra cluster* on page 15.

Configure API Manager Cassandra client settings

To update the Cassandra client configuration for API Manager, perform the following steps:

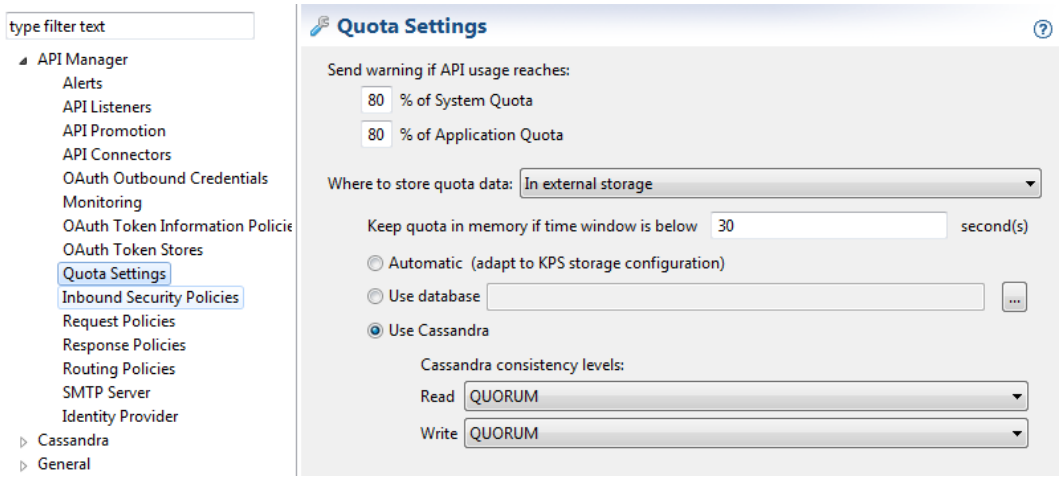
1. Ensure the API Gateway and API Manager components have been installed on the API Gateway 1 and API Gateway 2 nodes. These can be local or remote to Cassandra installations. For details, see the *API Gateway Installation Guide*.
2. Ensure an API Gateway domain, group, and instance have been created on the API Gateway 1 node using `managedomain`. For more details, see "Configure an API Gateway domain" in the *API Gateway Administrator Guide*.

Note This section assumes that you have already configured API Manager on the first node in non-HA standalone mode (for example, using Policy Studio or `setup-apimanager`). For more details, see the *API Manager User Guide*.

3. Start the first API Gateway instance in the group. For example:

```
startinstance -n "my_gw_server_1" -g "my_group"
```

4. Configure the Cassandra connection on the API Gateway 1 node. For details, see *Configure the API Gateway Cassandra client connection* on page 18.
5. Configure the Cassandra consistency levels for your KPS Collections. For details, see *Configure the API Gateway Cassandra consistency levels* on page 19.
6. In the Policy Studio tree, select **Server Settings** > **API Manager** > **Quota Settings**, and ensure that **Use Cassandra** is selected.
7. Under **Cassandra consistency levels**, in both the **Read** and **Write** fields, select `QUORUM`:



8. Add the API Gateway 2 host machine to the domain using `managedomain`.
9. Create the second API Gateway instance in the same group on the API Gateway 2 node.

Note Do not start this instance, and do not configure API Manager for this instance in Policy Studio.

10. Before starting the second API Manager-enabled instance, ensure that each instance has unique ports in the `envSettings.props` file. For example:

    i. Edit the `envSettings.props` file for the API Gateway instance in the following directory:

```
INSTALL_DIR/apigateway/groups/<group-n>/<instance-
m>/conf/envSettings.props
```

    ii. Add the API Manager ports. For example, the defaults are:

```
#API Manager Port
env.PORT.APIPORTAL=8075
#API Manager Traffic Port
env.PORT.PORTAL.TRAFFIC=8065
```

11. Start the second API Gateway instance. For example:

```
startinstance -n "my_gw_server_2" -g "my_group"
```

    On startup, this instance receives the API Manager configuration for the group. It now shares the same KPS and Cassandra configuration and data, and uses the ports specified in the `envSettings.props` file.

Apache Cassandra best practices 3

Follow the best practices in this section to achieve a stable Apache Cassandra environment, and to prevent data integrity and performance issues. You must complete all of these tasks before you start Apache Cassandra.

Clock synchronization and health check

The clocks of the system across all Cassandra cluster machines and the clocks of all client machines (API Gateway hosts) must be synchronized to one (1) millisecond precision.

Failing to synchronize the clocks will result in:

- Faults in data synchronization
- Failure to start or configure the machines correctly

The clock synchronization requires the use of a time service, such as NTP (Network Time Protocol), to ensure that the time is synchronized across all machines in the cluster.

You must also perform a health check of the clock drift between nodes on a regular basis.

User account

You must create a specific UNIX user account for the Cassandra database. This user account must own all Cassandra related files, and it must be used to run the Cassandra process.

This guide refers to this user account as `cassandra_user`.

Required system tuning

Cassandra executes basic performance and tuning checks at startup, and it writes warning messages to the console and to the system log file when issues are found.

These are examples of warnings for a misconfigured Cassandra host:

```
WARN Unable to lock JVM memory (ENOMEM). This can result in part of the JVM being
swapped out,
     especially with mmapped I/O enabled. Increase RLIMIT_MEMLOCK or run Cassandra
```

```
as root
WARN jemalloc shared library could not be preloaded to speed up memory allocations
WARN Cassandra server running in degraded mode. Is swap disabled? : true,
     Address space adequate? : true, nofile limit adequate? : false, nproc limit
adequate? : false
```

Perform the following procedures to ensure that each Cassandra machine meets the basic tuning requirements.

Note The following commands apply for Red Hat 7.x. If you are using another Linux distribution, consult your system administrator.

Install jemalloc

Ensure that `jemalloc` is installed.

1. Run the command `rpm -q jemalloc` to check if jemalloc is installed.
2. If jemalloc is not installed, run the following command to install it from `epel`:

```
sudo rpm -iv https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
sudo yum install -y jemalloc
```

Turn swap off

Ensure that you turn `swap` off.

Note Apache Cassandra recommends that `swap` is disabled. If your company policies or production environment requires `swap` to be `on` for other processes, you must ensure that the Cassandra process is not `swapped out` at any time.

1. The line `cat/proc/swaps` in the `/etc/fstab` file should show `NO entries`. If entries are present, execute the following command to disable all swap entries currently active:

```
sudo swapoff -a sad
```

2. Delete all swap entries in `/etc/fstab` to ensure that swap is not enabled again when the machine is restarted.

Set limits for user account

Set the minimum limits for the user account.

If running with a console or `init.d`, create a `/etc/security/limits.d/cassandra.conf` file, and add the following lines to it (replace `cassandra_user` with the relevant user account).

```
<cassandra_user> - memlock unlimited
<cassandra_user> - nofile 100000
<cassandra_user> - nproc 32768
<cassandra_user> - as unlimited
```

If running via a system service, ensure that the following lines are present in the `[SERVICE]` section of the Cassandra service file:

```
LimitMEMLOCK=infinity
LimitNOFILE=100000
LimitNPROC=32768
LimitAS=infinityd
```

Clean up Cassandra repair history

By default Apache Cassandra 2.2.x does *not* clean up `nodetool repair` trace history. This can cause the `system_distributed` keyspace to increase in size over time. The extent of the issue can be seen by running the following command to see how much space is being consumed by `system_distributed`:

```
du -md 1 <cassandra_root>/data/data/ | sort -n
```

To prevent this, it is recommended that you set a 7 day TTL on the repair history tables and remove any existing data. First, execute the following using `cqlsh` on one of the Cassandra nodes:

```
ALTER TABLE system_distributed.repair_history WITH default_time_to_live = 604800;
TRUNCATE system_distributed.repair_history;
ALTER TABLE system_distributed.parent_repair_history WITH default_time_to_live =
604800;
TRUNCATE system_distributed.parent_repair_history;
```

To reclaim the disk space, clean up the snapshots generated by the truncate by executing the following against all Cassandra nodes:

```
nodetool clearsnapshot system_distributed
```

Further information

See also *Perform essential Apache Cassandra operations* on page 37.

Manage Apache Cassandra 4

On Linux, when you select to install Cassandra using the API Gateway installer as part of a Standard or Complete setup, Cassandra starts automatically. To start or stop Cassandra manually or as a service, perform the steps described in this topic.

Note Before you start Cassandra, you must follow the best practices in Apache Cassandra Best Practices to achieve a stable Cassandra environment, and to prevent data integrity and performance issues.

Manage Apache Cassandra

This section explains how to start and stop Cassandra.

Start Apache Cassandra

To start Cassandra in the background:

1. Open a command prompt, and change to the following directory:

```
$ cd CASSANDRA_HOME/bin
```

2. Run the following command:

```
$ ./cassandra
```

To start Cassandra in the foreground, run the following command:

```
$ ./cassandra -f
```

For more details, see https://wiki.apache.org/cassandra/RunningCassandra.

Start Cassandra as a service

To install Cassandra as a service, you must install and configure appropriate startup script for your system. For example, see the following example startup scripts:

- **CentOS**:

  https://support.axway.com/kb/178063/language/en

- **Debian**:

  https://github.com/apache/cassandra/blob/cassandra-2.2/debian/init

When startup scripts are configured, you can then start Cassandra as a service.

Note You must have root or sudo permissions to start Cassandra as a service.

For example, typically the command to start Cassandra as a service is as follows:

```
$ sudo service cassandra start
```

Stop Cassandra

1. Find the Cassandra Java process ID (PID):

```
$ ps auwx | grep cassandra
```

2. Run the following command:

```
$ sudo kill pid
```

Stop Cassandra as a service

You must have `root` or `sudo` permissions to stop the Cassandra service as follows:

```
$ sudo service cassandra stop
```

Connect to API Gateway for the first time

Connecting to API Gateway depends on your operating system and installation setup type (Standard, Complete, or Custom).

Standard or Complete setup

If you installed a default Standard or Complete setup (both include the QuickStart tutorial), Cassandra is installed on the same host, and listens on `localhost` by default. API Gateway runs on the same host and connects to Cassandra by default.

Custom setup

If you installed a Custom setup and did not select the Quickstart tutorial, you must update the API Gateway Cassandra client configuration as follows:

1. Open the API Gateway group configuration in Policy Studio.
2. Select **Server Settings > Cassandra > Authentication**, and set the Cassandra username/password (default is `cassandra`/`cassandra`).
3. Select **Server Settings > Cassandra > Hosts**. Add an address for the Cassandra node. You can enter an IP address or host name.
4. Deploy the configuration to the API Gateway group.

For more details on configuring **Server Settings** in the Policy Studio client, see "Cassandra Settings" in the *API Gateway Administrator Guide*. For details on updating the Cassandra server configuration, see *Configure a highly available Cassandra cluster* on page 15.

Further details

For more details on Apache Cassandra, see the following:

- http://cassandra.apache.org/
- http://docs.datastax.com/en/cassandra/2.2/

Apache Cassandra backup and restore 5

In an Apache Cassandra database cluster, data is replicated between multiple nodes and potentially between multiple datacenters. Cassandra is highly fault-tolerant, and depending on the size of the cluster, it can survive the failure of one or more nodes without any interruption in service. However, backups are still needed to recover from the following scenarios:

- Any errors made in data by client applications
- Accidental deletions
- Catastrophic failure that requires a rebuild of the entire cluster
- Rollback of the cluster to a known good state in the event of data corruption

This topic describes which Apache Cassandra data keyspaces to back up, and explains how to use scripts to create and restore the data in those keyspaces. It also describes which Apache Cassandra configuration and API Gateway configuration to back up.

Before you begin

Caution You must read all of the following before you perform any of the instructions in this topic:

- These instructions apply only to an Apache Cassandra cluster where the replication factor is the same as the cluster size, which means that each node contains 100% of the data. This is the standard configuration for Axway API Management processes, and these instructions are intended to back up API Management keyspaces only.
- Because 100% of the data is stored on each node, you will run the backup procedure on a single node only, preferably on the seed node.
- The following instructions and scripts are intended as a starting point and must be customized and automated as needed to match your backup polices and environment.
- These instructions use the Cassandra snapshot functionality. A *snapshot* is a set of hard links for the current data files in a keyspace.

Note While the snapshot does not take up noticeable diskspace, it will cause stale data files not to be cleaned up. This is because a snapshot directory is created under each table directory and will contain hard links to all table data files.
When Cassandra cleans up the stale table data files, the snapshot files will remain. Therefore, it is critical to remove them promptly. The example backup script takes care of this by deleting the snapshot files at the end of the backup.

- For safety reasons, the backup location should *not* be on the same disk as the Cassandra data directory, and it also must have enough free space to contain the keyspace.

Which data keyspaces to back up?

These procedures apply to data in API Management and KPS keyspaces only. You must first obtain a list of the keyspace names to back up. API Management keyspaces may have a custom name defined, but are named in the format of `<UUID>_group_[n]` by default. For example:

```
x9fa003e2_d975_4a4a_a27e_280ab7fd8a5_group_2p_2
```

Note All Cassandra internal keyspaces begin with `system`, and should not be backed up using this process.

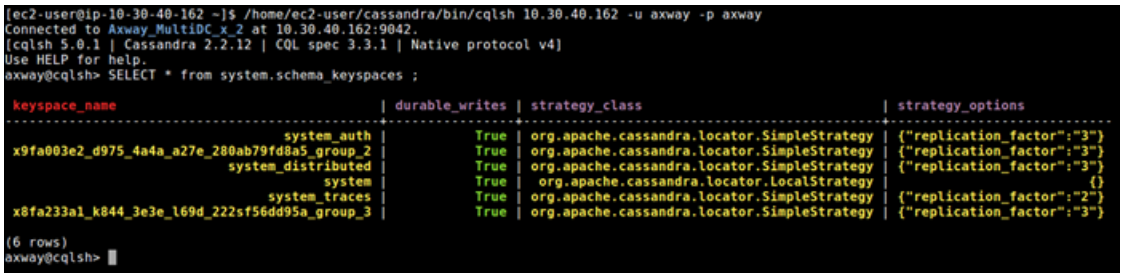
You can do this using `cqlsh` or `kpsadmin` commands:

Find keyspaces using cqlsh

Using `cqlsh`, execute the following command:

```
SELECT * from system.schema_keyspaces;
```

In the following example, `xxx_group_2` and `xxx_group_3` are API Management keyspaces:



Find keyspaces using kpsadmin

Using `kpsadmin`, choose: `option 30) Show Configuration`, and enter the API Gateway group and any instance to back up, or use the command line, as shown in the following example:

```
[ec2-user@ip-10-30-40-162 ~]$ ./Axway-7.5.3/apigateway/posix/bin/kpsadmin --username admin --password
changeme --group "g1" --name "i1" cassandra_configuration
Getting Topology...
========================
 Cassandra Configuration
========================
Getting Cassandra Configuration for all Gateways...
i1: OK

API Gateway: i1 [instance-1]

    Client config:
        hosts: 10.30.40.162:9042,10.30.50.104:9042
        keyspace: x9fa003e2_d975_4a4a_a27e_280ab79fd8a5_group_2


[ec2-user@ip-10-30-40-162 ~]$
```

Back up a keyspace

To back up a keyspace, you will use the `nodetool snapshot` command to create hard links, run a custom script to back up these links, and then archive that backup.

Note You must repeat these steps for each keyspace to back up.

1. Create a snapshot by running the following command on the seed node:

```
nodetool CONNECTION_PARMS snapshot -t SNAPSHOT_NAME-TIMESTAMP API_GW_KEYSPACE_NAME
```

For example:

```
[ec2-user@ip-10-30-40-162 ~]$ ./cassandra/bin/nodetool -u axway -pw axway -h 127.0.0.1 -p 7199 snapshot -t group_2-$(date +%Y%m%d_%H%M_%S)
 x9fa003e2_d975_4a4a_a27e_280ab79fd8a5_group_2
Requested creating snapshot(s) for [x9fa003e2_d975_4a4a_a27e_280ab79fd8a5_group_2] with snapshot name [group_2-20181128_1606_04]
Snapshot directory: group_2-20181128_1606_04
[ec2-user@ip-10-30-40-162 ~]$
```

2. Run the following script to copy the snapshot files to another location:

Note This script also clears the snapshot files from the Cassandra `data` directory.

```
#!/bin/bash
################################################################################
# Sample Cassandra snapshot backup script                                      #
# NOTE: This MUST be adapted for and validated in your environment before use! #
################################################################################

set -e
trap '[ "$?" -eq 0 ] || echo \*\*\* FATAL ERROR \*\*\*' EXIT $?

# Replace the xxx values below to match your environment
CASSANDRA_DATA_DIR="xxx"
KEYSPACE_NAME="xxx"
SNAPSHOT_NAME="xxx"
BACKUP_ROOT_DIR="xxx"

# Example:
#  CASSANDRA_DATA_DIR="/opt/cassandra/data/data"
#  KEYSPACE_NAME="x9fa003e2_d975_4a4a_a27e_280ab7fd8a5_group_2"
#  SNAPSHOT_NAME="Group2-20181127_2144_28"
```

```
#  BACKUP_ROOT_DIR="/backup/cassandra-snapshots"

##### Do NOT change anything below this line #####
backupdir="${BACKUP_ROOT_DIR}/${SNAPSHOT_NAME}"
if [ -d "${backupdir}" ]; then echo -e "\nERROR: Snapshot '${SNAPSHOT_NAME}' already
exists in the backup dir";exit 1; fi
mkdir -p ${backupdir}
keyspace_path="${CASSANDRA_DATA_DIR}/${KEYSPACE_NAME}"
if ! [ -d "${keyspace_path}" ]; then echo -e "\nERROR: Keyspace path '${keyspace_
path}' is not valid";exit 1; fi

snapshot_dirs=()

find "${keyspace_path}/" -maxdepth 1 -mindepth 1 -type d ! -name "$(printf "*\n*")"
> backup.tmp
while IFS= read -r table_dir
do
  str=${table_dir##*/}
  table_uuid=${str##*-}
  len=$((${#str} - ${#table_uuid} - 1))
  table_name=${str:0:${len}}
  echo "Copy table, $table_name"
  current_backup_dir="${backupdir}/${table_name}"
  mkdir -p "${current_backup_dir}"
  current_snapshot="${table_dir}/snapshots/${SNAPSHOT_NAME}"
  snapshot_dirs+=("${current_snapshot}")
  cp -r -a "${current_snapshot}"/* "${current_backup_dir}/"
done < backup.tmp
rm backup.tmp

echo -e "\nRemoving snapshot files from Cassandra data directory"
for dir in "${snapshot_dirs[@]}"
do
  rm -rf "${dir}"
done
```

When this script finishes, it creates the following backup directory structure:

```
<BACKUP_ROOT_DIR>
├── <SNAPSHOT_NAME>
│ ├── <TABLE_NAME>
│ │  ├── <SNAPSHOT_FILES>
│ ├── <TABLE_NAME>
│ │  ├── <SNAPSHOT FILES>
│ │...
```

3. Archive the `SNAPSHOT_NAME` directory using your company's archive method so you can restore it later if needed.

Note It is best to take a snapshot backup on a daily basis.

Restore a keyspace

This section explains how to restore API Management and KPS keyspaces and provides an example script to restore the files.

Before you begin

Note If you are restoring a keyspace to the same cluster that the backup was taken from, skip to *Steps to restore a keyspace* on page 32.

Before you restore a keyspace in a new Cassandra cluster, you must ensure the following:

- The Cassandra cluster must be created to the API Gateway HA specifications. For more details, see *Configure a highly available Cassandra cluster* on page 15.
- All API Gateway groups must have their schema created in the new cluster, and the replication factor must be the same as the cluster size (normally 3).
- If the keyspace name has changed in the new cluster, use the new name in the `KEYSPACE_NAME` variable in the restore script.

Steps to restore a keyspace

1. Shut down all API Gateway instances and any other clients in the Cassandra cluster.
2. Drain and shut down each Cassandra node in the cluster, one node at a time.

Note You must execute the following (and wait for it to complete) on each Cassandra node before shutting it down, otherwise data loss may occur:

```
nodetool CONNECTION_PARMS drain
```

3. On the Cassandra seed node, delete all files in the `commitlog` and `saved_caches` directory. For example:

```
rm -r /opt/cassandra/data/commitlog/* /opt/cassandra/data/saved_caches/*
```

Note Do not delete any folders in the keyspace folder on the node being restored. The restore script requires the table directories to be present in order to function correctly.

4. On the Cassandra seed node, run the Cassandra restore snapshot script to restore the snapshot files taken by the backup process and script:
5. On the other nodes in the cluster, perform the following:
    - Delete all files in the `commitlog` and `saved_caches` directory.
    - Delete all files in the `KEYSPACE` being restored under the `CASSANDRA_DATA_DIRECTORY`. For example:

```
rm -rf /opt/cassandra/data/data/x9fa003e2_d975_4a4a_a27e_280ab7fd8a5_group_2/*
```

6. If you have other keyspaces to restore, return to step 4 and repeat. Otherwise, continue to the next steps.
7. One at a time, start the Cassandra seed node, and then the other nodes, and wait for each to be in Up/Normal (`UN`) state in `nodetool status` before you proceed to the next node.
8. Perform a full repair of the cluster as follows on each node one at a time:

```
nodetool repair -pr --full
```

9. On the Cassandra seed node, run the Cassandra reload the indexes script.
10. Start the API Gateway instances.

Sample Cassandra restore snapshot script

```
#!/bin/bash
################################################################################
# Sample Cassandra restore snapshot script                                     #
# NOTE: This MUST be adapted for and validated in your environment before use! #
################################################################################

# Replace the xxx values below to match your environment
CASSANDRA_DATA_DIR="xxx"
KEYSPACE_NAME="xxx"
SNAPSHOT_NAME="xxx"
BACKUP_ROOT_DIR="xxx"

# Example:
#  CASSANDRA_DATA_DIR="/opt/cassandra/data/data"
#  KEYSPACE_NAME="x9fa003e2_d975_4a4a_a27e_280ab7fd8a5_group_2"
#  SNAPSHOT_NAME="Group2-20181127_2144_28"
#  BACKUP_ROOT_DIR="/backup/cassandra-snapshots"

##### Do NOT change anything below this line #####
backupdir="${BACKUP_ROOT_DIR}/${SNAPSHOT_NAME}"
keyspace_path="${CASSANDRA_DATA_DIR}/${KEYSPACE_NAME}"
echo -e "\n\tRestoring tables from directory, '${backupdir}', to directory,
'${keyspace_path}'"
echo -e "\tRestore snapshot, '${SNAPSHOT_NAME}', to keyspace, '${KEYSPACE_NAME}'"
read -n 1 -p "Continue (y/n)?" answer
echo -e "\n"
if [ "$answer" != "y" ] && [ "$answer" != "Y" ]; then
  exit 1
fi

set -e
trap '[ "$?" -eq 0 ] || echo \*\*\* FATAL ERROR \*\*\*' EXIT $?

if ! [ -d "${backupdir}" ]; then echo -e "\nERROR: Backup not found at
```

```
'${backupdir}'";exit 1; fi
if ! [ -d "${keyspace_path}" ]; then echo -e "\nERROR: Keyspace path '${keyspace_
path}' is not valid";exit 1; fi

keyspace_tables=$(mktemp)
find "${keyspace_path}/" -maxdepth 1 -mindepth 1 -type d -fprintf ${keyspace_tables}
"%f\n"
sort -o ${keyspace_tables} ${keyspace_tables}

backup_tablenames=$(mktemp)
find "${backupdir}/" -maxdepth 1 -mindepth 1 -type d -fprintf ${backup_tablenames}
"%f\n"
sort -o ${backup_tablenames} ${backup_tablenames}

keyspace_tablenames=$(mktemp)
table_names=()
table_uuids=()
while IFS= read -r table
do
  str=${table##*/}
  uuid=${str##*-}
  len=$((${#str} - ${#uuid} - 1))
  name=${str:0:${len}}
  echo "${name}" >> ${keyspace_tablenames}
  table_names+=(${name})
  table_uuids+=(${uuid})
done < ${keyspace_tables}

set +e
sort -o ${keyspace_tablenames} ${keyspace_tablenames}
diff -a -q ${keyspace_tablenames} ${backup_tablenames}
if [ $? -ne 0 ]; then
  echo -e "\nERROR: The tables on the keyspace at, '${keyspace_path}', are not the
same as the ones from the backup at,
'${backupdir}'"
  exit 1
fi

for ((i=0; i<${#table_names[*]}; i++));
do
  echo "Restoring table, '${table_names[i]}'"
  table_dir="${keyspace_path}/${table_names[i]}-${table_uuids[i]}"
  rm -r "${table_dir}"
  mkdir "${table_dir}"
  src="${backupdir}/${table_names[i]}"
  cp -r -a "${src}"/* "${table_dir}/"
done
```

Sample Cassandra reload the indexes script

```
#!/bin/bash
################################################################################
# Sample Cassandra reload indexes script                                       #
# NOTE: This MUST be adapted for and validated in your environment before use! #
################################################################################
# Replace the xxx values below to match your environment
CASSANDRA_DIR="xxx"
KEYSPACE_NAME="xxx"
# Example:
#  CASSANDRA_DATA_DIR="/opt/cassandra"
#  KEYSPACE_NAME="x9fa003e2_d975_4a4a_a27e_280ab7fd8a5_group_2"
##### Do NOT change anything below this line #####
cqlsh="${CASSANDRA_DIR}/bin/cqlsh"
keyspace_path="${CASSANDRA_DIR}/data/data/${KEYSPACE_NAME}"
echo -e "\n\tReloading indexes - all indexes will be dropped and reloaded again"
echo -e "\tReload indexes for keyspace ${KEYSPACE_NAME}?"
read -n 1 -p "Continue (y/n)?" answer
echo -e "\n"
if [ "$answer" != "y" ] && [ "$answer" != "Y" ]; then
  exit 1
fi
set -e
trap '[ "$?" -eq 0 ] || echo \*\*\* FATAL ERROR \*\*\*' EXIT $?
if ! [ -d "${CASSANDRA_DIR}" ]; then echo -e "\nERROR: Cassandra dir not found at
'${CASSANDRA_DIR}'";exit 1; fi
if ! [ -d "${keyspace_path}" ]; then echo -e "\nERROR: Keyspace path '${keyspace_
path}' is not valid";exit 1; fi
indexes=$(mktemp)
${cqlsh} -e "SELECT index_name FROM system.schema_columns WHERE keyspace_name =
'${KEYSPACE_NAME}'" > ${indexes}
sed -i -e '1,3d;/^$/d;/null/d;$d' ${indexes}
if [ $(wc -l < ${indexes}) -eq 0 ]; then
  echo -e "Error getting current indexes"
  exit 1
fi
while IFS= read -r line
do
  index="${line#"${line%%[![:space:]]*}"}"
  echo -e "Restoring index: '${index}'"
  create_index=$(${cqlsh} -e "DESCRIBE INDEX ${KEYSPACE_NAME}.\"${index}\";")
  if [ $(echo ${create_index} | wc -l) -ne 1 ]; then
    echo -e "Error getting create script for index: ${index}"
    exit 1
  fi
  echo -e "\tDropping index"
  ${cqlsh} -e "DROP INDEX ${KEYSPACE_NAME}.\"${index}\";"
  echo -e "\tCreating index again"
  ${cqlsh} -e "${create_index}"
```

```
  echo -e "\tChecking created index"
  ${cqlsh} -e "DESCRIBE INDEX ${KEYSPACE_NAME}.${index};" > /dev/null
  echo -e "\tDone"
done < "$indexes"
```

Which configuration to back up?

In addition to backing up your data in Apache Cassandra keyspaces, you must also back up your Apache Cassandra configuration and API Gateway configuration.

Apache Cassandra configuration

You must back up the `CASSANDRA_HOME/conf` directory on all nodes.

API Gateway group configuration

You must back up the API Gateway group configuration in the following directory:

```
API_GW_INSTALL_DIR/apigateway/groups/<group-name>/conf
```

This directory contains the API Gateway, API Manager, and KPS configuration data.

Perform essential Apache Cassandra operations 6

This topic describes the minimum essential operations that are required to maintain a healthy Apache Cassandra high availability (HA) cluster.

Repair inconsistent nodes

Apache Cassandra has different ways to maintain data consistency across the cluster during normal runtime processing. However, to resolve any data inconsistencies that could not be resolved by normal runtime processes, you must run an external repair process on a regular basis.

For standard HA configurations, it is best to run repair once per week (during a quiet period) on each node in the cluster.

Caution The repair must only be executed on one node at a time. You must therefore adjust the repair schedule for each node in the cluster to avoid overlap.

Schedule repair using crontab

When scheduling Cassandra repair events, it is best to use the `crontab` command of the user running the Cassandra process.

To edit the cron table of this user, execute `sudo crontab -u CASSANDRA_USER -e` and paste the matching node block from the following examples:

Example cron table entries for three-node cluster repairing one node per day (Saturday to Monday)

Node 1:

Run full repair at 1 a.m. every Saturday:

```
0 1 * * 6 PATH_TO_CASSANDRA/bin/nodetool CONNECTION_SECURITY_PARAMS
repair -pr --full > PATH_TO_CASSANDRA/logs/last_repair.log 2>&1
```

Node 2:

Run full repair at 1 a.m. every Sunday:

```
0 1 * * 0 PATH_TO_CASSANDRA/bin/nodetool CONNECTION_SECURITY_PARAMS
repair -pr --full > PATH_TO_CASSANDRA/logs/last_repair.log 2>&1
```

Node 3:

Run full repair at 1 a.m. every Monday:

```
0 1 * * 1 PATH_TO_CASSANDRA/bin/nodetool CONNECTION_SECURITY_PARAMS
repair -pr --full > PATH_TO_CASSANDRA/logs/last_repair.log 2>&1
```

See also *Clean up Cassandra repair history* on page 24.

Replace dead nodes

If a node is down for more than 10 days, it should be replaced. For details on replacing dead Cassandra nodes, see the Replacing a dead node or dead seed node documentation.

Reconfigure an existing Apache Cassandra installation from scratch

There is no need to reinstall Cassandra from scratch. Instead, you can move the Cassandra data files and restore the `cassandra.yaml` configuration file if necessary (if you updated Cassandra configuration). Perform the following steps:

1. Stop Cassandra. For details, see *Manage Apache Cassandra* on page 25.
2. Move `CASSANDRA_HOME/data` to `CASSANDRA_HOME/data/OLD-DATA-DATE`.
3. Restore `cassandra.yaml` in `CASSANDRA_HOME/conf` if necessary.

Enable Apache Cassandra debug logging

You can enable Cassandra debug logging using any of the following approaches:

- **logback.xml**

  You can specify a `logger` in the `cassandra/conf/logback.xml` configuration file as follows:

```
<logger name "org.apache.cassandra.transport" level=DEBUG/>
```

- **nodetool**

  You can use the `nodetool setlogginglevel` command as follows:

```
nodetool setlogginglevel org.apache.cassandra.transport DEBUG
```

- **JConsole**

  The JConsole tool enables you to configure Cassandra logging using JMX. For example, you can invoke `setLoggingLevel` and specify `org.apache.cassandra.db.StorageServiceMBean` and `DEBUG` as parameters. For more details, see the next section.

For more details on enabling logging, see the following, which also applies to Cassandra 2.2:

- https://docs.datastax.com/en/cassandra/2.1/cassandra/configuration/configLoggingLevels_r.html
- http://thelastpickle.com/blog/2016/02/10/locking-down-apache-cassandra-logging.html

Monitor a Cassandra cluster using JMX

You can use Java Management Extensions to monitor and manage performance in a Cassandra cluster. For details, see the Monitoring a Cassandra cluster documentation.

Upgrade your Cassandra version

For details on upgrading your Cassandra version, see "Upgrade Apache Cassandra" in the *API Gateway Upgrade Guide*.

setup-cassandra script reference 7

The `setup-cassandra` script provided by API Gateway enables you to configure a multi-node Cassandra HA cluster automatically. You can use this script when Cassandra is installed locally along with API Gateway, or installed remotely on a different node. For details on supported Cassandra deployment architectures and HA production environments, see *Configure a highly available Cassandra cluster* on page 15.

API Gateway provides the `setup-cassandra` script to help configure a Cassandra cluster by updating your Cassandra configuration files and providing instructions to finalize the configuration. This script also creates an automatic backup of the original `cassandra.yaml` file in the following format:

```
<timestamp>_cassandra.yaml.bak
```

This topic describes how to run the `setup-cassandra` script, and how to configure a multi-node Cassandra cluster with username/password authentication enabled. It also describes how to configure TLS v1.2 encryption for client-server and inter-node communications in the cluster.

Prerequisites

The following prerequisites apply for the `setup-cassandra` script:

Python

- Cassandra 2.2.5 requires Python 2.7.x *(up to 2.7.10)*
- Cassandra 2.2.8 requires Python 2.7.x *(up to the latest)*
- Cassandra 2.2.12 requires Python 2.7.x *(up to the latest)*

Note API Gateway 7.6.2 supports Cassandra 2.2.12. API Gateway 7.5.3 supports Cassandra versions 2.2.5 and 2.2.8.

User name and password authentication

User name and password authentication for clients connecting to the cluster is enabled by the `setup-cassandra` script by default. However, you must change the default user name and

password created by Cassandra on startup to further secure the installation. The script provides instructions describing how to change the default user name and password and replicate the `system_auth` keyspace using the `cqlsh` command.

For more details, see *Secure Cassandra HA configuration* on page 43.

Remote Cassandra HA nodes

`setup-cassandra` is available by default when Cassandra is installed locally on the same node as API Gateway. You can also configure remote Cassandra nodes to use the `setup-cassandra` script supplied by the API Gateway installation. Alternatively, you can perform all necessary Cassandra configuration changes manually. For details, see *Configure a highly available Cassandra cluster* on page 15.

To configure a remote Cassandra node to use the `setup-cassandra` script, perform the following steps:

1. Ensure that the `JAVA_HOME` environment variable is set on the remote Cassandra node.
2. Ensure that Python is installed and added to your `PATH` environment variable on the remote Cassandra node.
3. Change to the following directory on the local API Gateway node:

```
AXWAY_HOME/apigateway/system/lib/jython/
```

4. Copy the `setup-cassandra.py` file to your chosen directory on the remote Cassandra node.

Run the setup-cassandra script

The instructions for running the `setup-cassandra` script depend on whether the node is local or remote to API Gateway.

Run setup-cassandra on local API Gateway node

On Linux, the `setup-cassandra` script is bundled with the API Gateway installation and located in the `bin` directory. To run this script on a local node:

```
$ cd AXWAY_HOME/apigateway/posix/bin
$ ./setup-cassandra <options>
```

Note Windows is supported only for a limited set of developer tools. API Gateway and API Manager do not support Windows.

Run setup-cassandra on remote Cassandra node

On a remote Cassandra node, you first must ensure that the `setup-cassandra` script has been configured as described in *Remote Cassandra HA nodes* on page 41. To run this script on a remote node:

```
$ ./setup-cassandra.py <options>
```

Note The example commands in this topic assume that Cassandra is installed locally on an API Gateway node.

Configure the seed node

To configure Cassandra to run as the cluster seed node, run the `setup-cassandra` script with the following options:

```
setup-cassandra --seed --own-ip=<OWN_IP> --nodes=<NUMBER_OF_NODES> --
cassandra-config=<CONFIG_FILE>
```

These options are described as follows:

`OWN_IP`	IP address of this Cassandra host. Cassandra uses this IP address for communicating with other nodes in the cluster and for receiving client connections.
`NODES`	Total number of the nodes in the Cassandra cluster.
`CONFIG_FILE`	Full path to `cassandra.yaml` configuration file. Typically the path is `<CASSANDRA_INSTALL_DIR>/conf/cassandra.yaml`.

For example:

```
setup-cassandra --seed --own-ip=ipA --nodes=3 --cassandra-
config=/opt/cassandra/conf/cassandra.yaml
```

Configure additional nodes

To configure Cassandra on the remaining cluster nodes, run the `setup-cassandra` script with the following options:

```
setup-cassandra --seed-ip=<SEED_IP> --own-ip=<OWN_IP> --cassandra-
config=<CONFIG_FILE>
```

These options are described as follows:

SEED_IP	IP address of the Cassandra seed host (see *Configure the seed node* on page 42).
OWN_IP	IP address of this Cassandra host. Cassandra uses this IP address for communicating with other nodes in the cluster and for receiving client connections.
CONFIG_FILE	Full path to `cassandra.yaml` configuration file. Typically the path is `<CASSANDRA_INSTALL_DIR>/conf/cassandra.yaml`.

For example:

```
setup-cassandra --seed-ip=ipA --own-ip=ipB --cassandra-
config=/opt/cassandra/conf/cassandra.yaml
```

Secure Cassandra HA configuration

This section explains how to use the `setup-cassandra` script to secure your Cassandra HA configuration. The examples assume that Cassandra is installed locally on the same host as API Gateway. See also *Run setup-cassandra on remote Cassandra node* on page 42.

Reset your default user name and password

You can use the `setup-cassandra` script to reset the default user name and password (`cassandra`/`cassandra`). Run this command to see the instructions that you need to follow. For example, on the seed node the instructions are as follows:

```
./setup-cassandra --seed --own-ip=ipA --nodes=3 --cassandra
config=/opt/cassandra/conf/cassandra.yaml
Connect to Cassandra with cqlsh and run following commands to create an
alternative superuser account:
CREATE USER admin WITH PASSWORD 'amujsa26al2ns' SUPERUSER;QUIT
PLEASE MAKE A NOTE OF USERNAME AND PASSWORD FOR THE NEW SUPERUSER
ACCOUNT:
USERNAME: admin PASSWORD: amujsa26al2ns
Connect to Cassandra using newly created account to lock out the default
Cassandra superuser account and update "system_auth" keyspace
replication factor:
```

```
/opt/cassandra/bin/cqlsh -u admin -p amujsa26al2ns node1 ALTER USER
cassandra WITH PASSWORD
'g5q5h4h3bf1pnh2nsra9iucd82d7f1jams468vhaiimtibtuqpf' NOSUPERUSER;
ALTER KEYSPACE "system_auth" WITH REPLICATION = { 'class':
'SimpleStrategy', 'replication_factor': 3 }; QUIT
```

Note If you are setting up a Cassandra HA cluster, you must replicate the `system_auth` keyspace as shown in this example. This enables API Gateway to communicate with the cluster if a node goes down. For more details, see the Configuring authentication documentation.

Enable node-to-node traffic encryption

To configure TLS v1.2 encryption, you must generate a private key and certificate for every Cassandra node in the cluster. You must also obtain the certificate for the Certification Authority (CA) used to generate the node certificate. You can use Policy Studio to create the necessary certificates and keys or any other suitable method for generating certificates.

You must export and save the node certificate and private key as a PKCS12 file named `server.p12`, and save the CA certificate as a PEM file named (`server-ca.pem`). You must place these files into the same directory where you run the `setup-cassandra` script from.

To use the `setup-cassandra` script to configure node-to-node TLS/SSL encryption, add the `--enable-server-encryption` option. For example:

```
setup-cassandra --seed-ip=ipA --own-ip=ipB --cassandra-
config=/opt/cassandra/conf/cassandra.yaml --enable-server-encryption
```

After you run the `setup-cassandra` script, it provides instructions for converting the keys and certificates to a format required by Cassandra. After you perform these instructions, you can remove the `server.p12` and `server-ca.pem` files from the system.

Caution Anyone with a private key or certificate signed by `server-ca.pem` can connect to the cluster. You should limit the use of this CA to signing the node certificates only. In particular, do not use the same CA to sign client-to-node certificates.

Enable client-to-node traffic encryption

The requirements for setting up client-to-node TLS/SSL encryption are the same as the node-to-node TLS/SSL requirements. However, you must save the CA certificate/private key and node certificate in `client-ca.pem` and `client.p12` files respectively. For example, to instruct the `setup-cassandra` script to configure client-to-node encryption, add the `--enable-client-encryption` option to script arguments:

```
setup-cassandra --seed-ip=ipA --own-ip=ipB --cassandra-
config=/opt/cassandra/conf/cassandra.yaml --enable-client-encryption
```

After you run the `setup-cassandra` script, it provides instructions for converting the keys and certificates to a format required by Cassandra. After you perform these instructions, you can remove the `client.p12` and `client-ca.pem` files from the system.

Configure the cqlsh command for client-to-node traffic encryption

When the Cassandra cluster has been configured to use client-to-node TLS/SSL encryption, you must configure all clients connecting to the cluster (including `cqlsh`) to use TLS/SSL.

If client-to-node TLS/SSL encryption has been enabled, the `setup-cassandra` script creates a configuration file (`cqlshrc`) with the necessary configuration to enable TLS/SSL encryption. However, you must provide following files to configure `cqlsh` for TLS/SSL:

- `client-ca.pem`: PEM file containing CA certificate
- `cqlsh-cert.pem`: PEM file containing client certificate for `cqlsh`
- `cqlsh-key.pem`: PEM file containing private key of client certificate for `cqlsh`

Updated Cassandra configuration

This section shows the updates that the `setup-cassandra` script makes to the `cassandra.yaml` configuration file:

General settings

- `seed_provider`, `parameters`, `seeds`: <*IP address of seed node*>
- `start_native_transport`: `true`
- `native_transport_port`: `9042`
- `listen_address`: IP address of the node
- `rpc_address`: IP address of the node
- `authenticator`: `org.apache.cassandra.auth.PasswordAuthenticator`
- `authorizer`: `org.apache.cassandra.auth.CassandraAuthorizer`

Node-to-node traffic encryption

If node-to-node TLS/SSL traffic encryption is enabled:

- `server_encryption_options`: Specified options
- `internode_encryption`: all

- `keystore`: <*`server-keystore`*`.jks`>
- `keystore_password`: Randomly generated password
- `truststore`: <*`server-truststore`*`.jks`>
- `truststore_password`: Randomly generated password
- `protocol`: TLS
- `store_type`: JKS
- `algorithm`: SunX509
- `require_client_auth`: true

Client-to-node traffic encryption

If client-to-node TLS/SSL traffic encryption is enabled:

- `client_encryption_options`: Specified options
- `enabled`: true
- `optional`: false
- `keystore`: <*`client-keystore`*`.jks`>
- `keystore_password`: Randomly generated password
- `truststore`: <*`client-truststore`*`.jks`>
- `truststore_password`: Randomly generated password
- `protocol`: TLS
- `store_type`: JKS
- `algorithm`: SunX509
- `require_client_auth`: true