opentext™

Kafka Reference Guide

Overview

Apache Kafka is a distributed publish-subscribe messaging system that enables passing of messages from one system to another, while handling large volumes of data. Kafka can be enabled on a single Linux or Appliance Primary Server or can be run as a cluster on one or more Linux or Appliance servers that can span multiple data centers. Each server in the cluster is called a broker. Kafka is run as a cluster to ensure high availability of its services by replicating Kafka topics or messages to multiple Kafka brokers. Kafka requires ZooKeeper to co-ordinate between the servers within the Kafka cluster. For more information on ZooKeeper, see Managing ZooKeeper in the *ZENworks Primary Server and Satellite Reference*

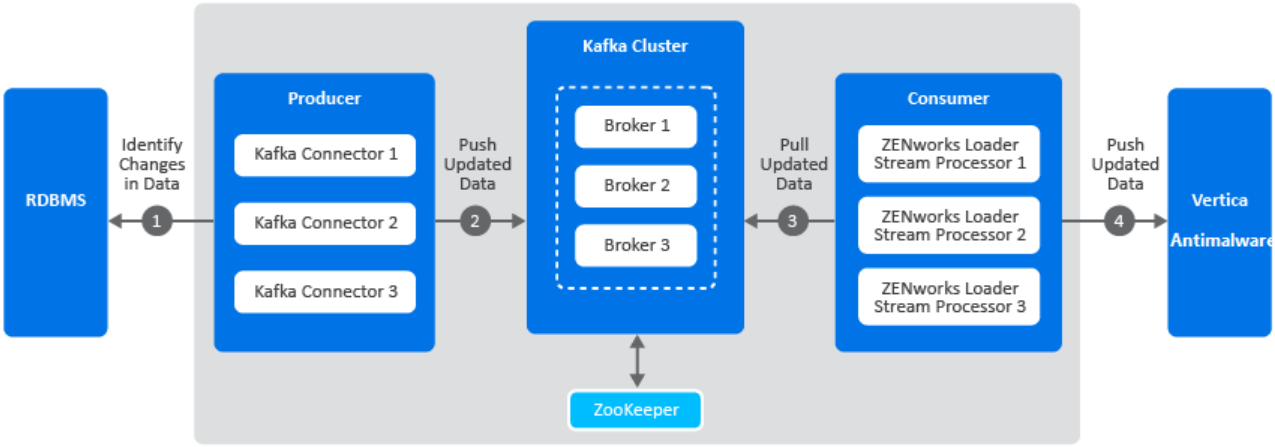
In ZENworks, Apache Kafka is required for the following components:

- **Vertica:** For more information, see ZENworks Vertica Guide
- **Antimalware:** For more information, see ZENworks Endpoint Security Antimalware Reference

You can enable Apache Kafka either through the ZENworks Control Center or by using the ZMAN utility.

Kafka Change Data Capture Workflow

The following diagram is a graphical representation of the Kafka workflow:



A description of each component in this architecture is as follows:

- **Kafka Cluster:** A group of servers or nodes that are connected to each other to achieve a common objective is a cluster. Each of the servers or nodes in the cluster, will have one instance of Kafka broker running.
- **Kafka Broker:** One or more servers that are added in a Kafka cluster are called brokers.
- **Apache ZooKeeper:** Kafka uses ZooKeeper to manage and co-ordinate between brokers. It notifies Kafka Producers and Consumers of any broker failures.
- **Kafka Producers:** The processes that publish messages to Kafka brokers. In this case, Kafka connectors are created as soon as Kafka is enabled. These connectors are created for each table in the RDBMS database and is responsible for identifying changes in these tables and publishing them to Kafka.
- **Kafka Consumers:** A service that reads data from Kafka brokers. In this case, the ZENworks loader stream processors subscribes to data from Kafka and passes this information to the Vertica or the Antimalware database.

The Kafka workflow is as follows:

1. The Kafka connectors (Kafka producer) identifies changes in the respective RDBMS database tables.
2. These changes are published to a topic within a Kafka broker. Kafka maintains messages as categories and these categories are called topics. To modify the interval at which the Kafka connector identifies changes in the RDBMS tables and publishes this data to Kafka, you can update the connector-configs.xml file (available at `etc/opt/microfocus/zenworks`) and run the `zman srkccn` command to re-configure the connectors. For more information on this command, see Maintaining Kafka Connect.

   The topics are further broken down into partitions for speed and scalability. These partitions are replicated across multiple brokers, which is used for fault tolerance. The replicas are created based on the replication count specified while enabling the Kafka cluster. Each message within a partition is maintained in a sequential order, which is identified as an offset, also known as a position.
3. The ZENworks loader stream processors (consumers) subscribe to a topic, Kafka offers the current offset of the topic to the consumer and saves the offset in the ZooKeeper cluster.
4. The ZENworks loader stream processor receives the message and processes it to the Vertica or the Antimalware database and the Kafka broker receives an acknowledgment of the processed message.

Kafka also uses a Schema Registry, which is a separate entity that producers and consumers talk to, for sending and retrieving schemas that describe the data models for the messages.

For more information on the Kafka Architecture, see the Confluent docs.

Enabling Kafka on a Single Server

This section explains the procedure to enable Kafka in the zone and to configure a single server with the Kafka service. You can enable Kafka on a single server in either of the following ways:

- Using ZENworks Control Center
- Using the ZMAN Utility

Prerequisites

Before enabling Kafka in the zone, you need to make a note of the following requirement:

- As Kafka requires client authentication to be enabled in the certificate, if you are using an external CA certificate, ensure that client authentication is enabled.
- Ensure that you have already added a Linux or an Appliance server to the zone.
- You need to first configure the Kafka cluster and then add a Kafka broker to the zone. Before adding a Kafka broker, ensure that the following prerequisites are met:
  - The Kafka cluster should already be configured.
  - The broker or server that is to be added to the cluster should not already be added to the cluster.
  - The server should be accessible and should have a specific host name. If a server has multiple host names, then the command to add the Kafka broker might fail.
  - As Kafka requires Client Authentication to be enabled in the certificate, if you are using an external CA certificate, ensure that Client Authentication is enabled in the Extended Key Usage (EKU) parameter of the server certificate.

Using ZENworks Control Center

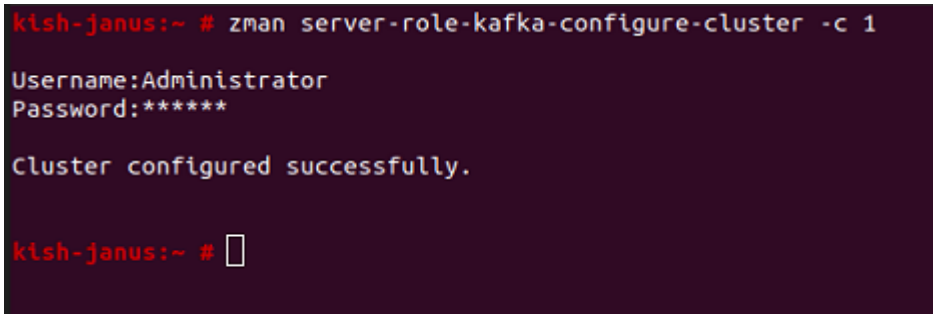
1. In ZENworks Control Center (ZCC), navigate to **Configuration > Performance Upgrade**.
2. Click **Enable Kafka**.
3. Select a Linux Primary Server or an Appliance server in which Kafka should be enabled.

The Kafka service will be enabled and the first broker will be added to the Kafka cluster. To add more brokers to the cluster, see Adding Servers to the Kafka Cluster.

Using the ZMAN Utility

1. **Configure the Kafka Cluster:** Execute the following command:

   ```
   zman server-role-kafka-configure-cluster (zman srkcc)
   ```

   While running this command, you need to specify the replication count. If the values for the remaining parameters are not specified, then the default values are considered. The parameters are:

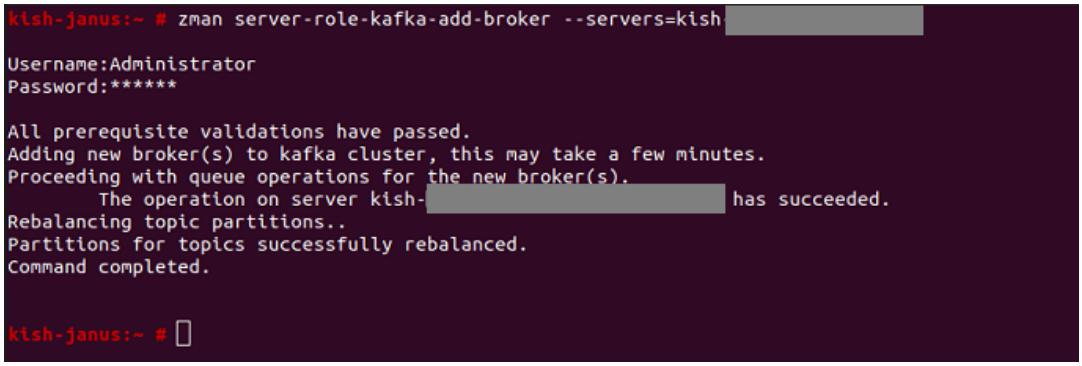
   - **-c --replication count:** This ensures that messages remain available when a server in the cluster fails.Specify the number of copies to be maintained for each topic. For a single node cluster, this count can be set as 1.

- **-l --logRetentionBytes:** Specify the maximum permissible size of the log, beyond which, the existing data is overwritten with the new data. By default the log size is unlimited.
- **-t --zkSessionTimeout:** Specify the ZooKeeper session timeout (in milliseconds), which is the maximum time that the server waits to establish a connection to ZooKeeper. If the server fails to signal a heartbeat to ZooKeeper within this specified time period, then the server is considered to be dead. A heartbeat request helps identify if the server is still connected to the Kafka cluster. The default value is 30000 milliseconds.
- **-r --retainDetectedLogsDuration:** Specify the maximum time to retain deleted logs. The default value is 86400000 milliseconds (1 day).
- **-p --logCleanupPolicy:** Specify the default cleanup policy for segments that exceed the maximum permissible retention window. The possible values are Delete and Compact. The default value is Delete. The Delete policy will remove old segments when the retention time or size limit has reached. The Compact policy will enable log compaction on the topic, which ensures that Kafka will always retain at least the last known value for each message key within the log of data for a single topic partition.
- **-s --schemaregistryport:** Specify the port on which the Schema Registry is running. The default value is 8081.
- **-k --kafkaport:** Specify the port on which Kafka listens. The default value is 9093.
- **-x --connectport:** Specify the port on which Kafka connect listens. The default value is 8083.

For example, `zman server-role-kafka-configure-cluster -c=1`

2 **Add Kafka brokers:** Before adding a broker, ensure that you have read the Prerequisites. Execute the following command:

`zman server-role-kafka-add-broker (zman srkab)`



This command adds brokers or servers to the configured Kafka cluster.

NOTE: If you are using an external CA certificate, for Kafka to work as expected, ensure that Client Authentication is enabled in the Extended Key Usage (EKU) parameter of the server certificate or else an error message is displayed. However, if you want to continue to use the existing certificate, execute the command again, using the i=true parameter. Kafka might work with the existing certificate.

The parameters to be specified are:

- **--servers:** Specify the appliance server on which Kafka should be enabled. Specify the DNS, GUID or path of the server object (server, server folder or server group) relative to /Devices/Servers.

- **-i --ignorewarning message (optional):** As Kafka requires client authentication to be enabled in the certificate, if you are using an external CA certificate, ensure that client authentication is enabled. However, if you want to ignore the error message and continue with the existing certificate, then execute the command again with the option `i` set as true.

For example: `zman server-role-kafka-add-broker --servers=server1.microfocus.com`

When this command is executed for a server, the Kafka, Kafka connect and Schema Registry services are enabled and started on the specified server.

For more information on debugging Kafka configuration issues, see Debugging Issues and Log Locations.

You can view the status of the Kafka Cluster configuration by navigating to **Configuration > Performance Upgrade**. For more information on monitoring the Kafka cluster, see Monitoring the Status of the Kafka Cluster.

Adding Servers to the Kafka Cluster

Prerequisites

The prerequisites to be followed before adding servers to the Kafka cluster, are:

- Ensure that the hardware and software configuration of the servers that are to be added are the same as the configuration of the existing server within that cluster.
- Although there is no limit to the number of servers to be added in a cluster, ZENworks recommends a cluster size of 3 servers for Kafka.
- For Kafka, before adding servers, the Kafka cluster should already be enabled. For more information, see .
- The broker or server that is to be added to the Kafka cluster should not already be added to the cluster.
- The server should be accessible and should have a specific host name.

Procedure

1 **Add Brokers:** Execute the following command to add brokers or servers to the configured Kafka cluster.

```
zman server-role-kafka-add-broker (zman srkab)
```

The parameters to be specified are:

- **--servers:** Specify a comma separated list of appliance servers on which Kafka should be enabled. You can specify the DNS, GUID or path of the server object (server, server folder or server group) relative to /Devices/Servers.
- **-i --ignorewarning message (optional):** As Kafka requires client authentication to be enabled in the certificate, if you are using an external CA certificate, ensure that client authentication is enabled and execute the command again with the option `i` set as true. Option `i` enables you to ignore the warning message and proceed with the execution of the command.

  For example: `zman server-role-kafka-add-broker --servers=server1.microfocus.com,server2.microfocus.com`

IMPORTANT: When a broker is being added to the zone, tasks to remove or re-configure servers cannot be performed on other servers.

When this command is executed for a server, the Kafka, Kafka connect and Schema Registry services are enabled and started on the specified server. The server is added to the cluster. When a broker is added in the cluster, ZENworks re-configures other existing brokers in the cluster. When re-configuration fails in one these brokers, you need to run the zman command (`zman srkrcb`) to re-configure the broker. For more information on this command, see Managing the Kafka Cluster.

2 **Update the Kafka Cluster:** Execute the following command to modify the existing cluster parameters such as the replication count.

```
zman server-role-kafka-update-cluster (zman srkuc)
```

The parameters are:

- **-c --replication count:** This ensures that messages remain available when a server in the cluster fails.Specify the number of copies to be maintained for each topic. For a three node cluster, this count can be set as 3, which indicates that a replica of the topic will be created on each server.
- **-l --logRetentionBytes:** Specify the maximum permissible size of the log, beyond which, the existing data is overwritten with the new data. By default the log size is unlimited.
- **-t --zkSessionTimeout:** Specify the ZooKeeper session timeout (in milliseconds), which is the maximum time that the server waits to establish a connection to ZooKeeper. If the server fails to signal a heartbeat to ZooKeeper within this specified time period, then the server is considered to be dead. A heartbeat request helps identify if the server is still connected to the Kafka cluster. The default value is 30000 milliseconds.
- **-r --retainDetectedLogsDuration:** Specify the maximum time to retain deleted logs. The default value is 86400000 milliseconds (1 day).
- **-p --logCleanupPolicy:** Specify the default cleanup policy for segments that exceed the maximum permissible retention window. The possible values are Delete and Compact. The default value is Delete. The Delete policy will remove old segments when the retention time or size limit has reached. The Compact policy will enable log compaction on the topic, which ensures that Kafka will always retain at least the last known value for each message key within the log of data for a single topic partition.
- **-s --schemaregistryport:** Specify the port on which the schema registry is running. The default value is 8081.
- **-k --kafkaport:** Specify the port on which Kafka listens. The default value is 9093.
- **-x --connectport:** Specify the port on which Kafka connect listens. The default value is 8083.

  For example: `zman server-role-kafka-update-cluster -c=3`

To remove a broker from the cluster or to re-configure a broker, see Managing the Kafka Cluster.

Managing the Kafka Cluster

The topics covered in this section are:

Maintaining Kafka Brokers

- **Removing brokers from the Kafka cluster:** Execute the following command to remove a broker from the Kafka cluster:

`zman server-role-kafka-remove-broker (zman srkrb)`

This command lets you remove Kafka brokers from an existing cluster. Specify the DNS, GUID or path of the server object (server, server folder or server group) relative to /Devices/Servers.

For example: `zman server-role-kafka-remove-broker --servers=server1.microfocus.com`

- **Re-configuring Kafka broker:** Execute the following command to re-configure the Kafka broker:

  `zman server-role-kafka-reconfig-broker (zman srkrcb)`

  Specify the DNS, GUID or path of the server object (server, server folder or server group) relative to /Devices/Servers, while executing this command. When a broker is added to Kafka cluster, the remaining servers are re-configured by ZENworks. However, if re-configuration fails for a specific server, then run this command for that server. This command is also used for Disaster Recovery scenarios. For more information, see ZENworks Disaster Recovery Reference

  For example: `zman server-role-kafka-reconfig-broker --servers=server1.microfocus.com`

- **Viewing list of Kafka brokers:** Execute the following command to view the list of Kafka brokers.

  `zman server-role-kafka-list-cluster (zman srklc)`

Maintaining Kafka Connect

When Kafka is enabled, Kafka connectors are created that is associated with each table in the RDBMS database. If any of these connectors are not running, then you might have to restart these connectors or re-configure the connectors.

To view the status of these connectors, navigate to the Diagnostic page in ZCC and view the status in the Data Sync Status section.

- **Restart Kafka Connectors:** Execute the following command:

  `zman server-role-kafka-restart-connectors (zman srkrcn)`

  Specify a comma separated list of connectors. This command can be used when any of the Kafka connectors displayed in the Diagnostics page in ZCC, is not running.

  For example: `zman server-role-kafka-restart-connectors -c=zenconnectorzBundlefolderrights,zenconnectorzbundlerights`

- **Reconfigure Kafka Connectors:** Execute the following command to reconfigure connectors based on the properties mentioned in the connector-configs.xml. This file is available at `etc/opt/microfocus/zenworks`.

  `zman server-role-kafka-reconfigure-connectors (zman srkccn)`

  The connector-configs.xml file lets you modify properties such as the interval at which the Kafka connector identifies changes in the RDBMS tables or the interval at which updated data is published to Kafka. Specify a comma separated list of connectors.

  For example: `zman server-role-kafka-reconfigure-connectors -c=zenconnectorzBundlefolderrights,zenconnectorzbundlerights`

- **Re-create Kafka Connectors:** Execute the following command to re-create connectors.

  `zman server-role-kafka-recreate-connectors (zman srkrcc)`

You can execute this command if you have migrated your primary database to another RDBMS database.

For example: `zman server-role-kafka-recreate-connectors -f`

- **Retrieve Kafka configuration details:** Execute the following command to retrieve the Kafka configuration details:

  `zman server-role-kafka-get-connector-config (zman srkgcc)`

  For example: `zman server-role-kafka-get-connector-config -c=zenconnectorzBundlefolderrights`

- **Retrieve list of Kafka connectors:** Execute the following command to retrieve a list of Kafka connectors:

  `zman server-role-kafka-list-connectors (zman srklcn)`

Monitoring the Status of the Kafka Cluster

You can view the overall configuration status of Kafka in the **Performance Upgrade** page in ZCC. For more information, see Viewing the Configuration Status. However, if you want to continuously monitor the status of the Kafka Cluster, see Monitoring Diagnostics.

Viewing the Configuration Status

To view the configuration status of Kafka, you can navigate to the Getting Started page in ZCC. To navigate to this page, click **Configuration** > **Performance Upgrade**.

Each task on this page, includes an icon that indicates the following status:

- : the component is successfully configured.
- : the component is ready to be configured.
- : an error has occurred while configuring the component.

Monitoring Diagnostics

To monitor the overall health of the Kafka clusters, you need to navigate to the Diagnostics page in ZCC. To navigate to the diagnostics page in ZCC, click **Diagnostics** in the left pane of ZCC.

The statuses displayed are as follows:

Up: indicates that all servers are up.

Down: if at least one server is down in the zone, then based on the replication count the relevant status is displayed. For example, the status is displayed as down if at least one server is down in a three node cluster with replication count being 1.

Warning: if at least one server is down in the zone, then based on the replication factor the relevant status is displayed. For example, the status is displayed as warning if at least one server is down in a three node cluster with replication factor being 2.

For the Kafka Cluster, the status of the Kafka Brokers, Kafka Connect and the Schema registry are also displayed.

For debugging issues related to the Kafka cluster, see Debugging Issues and Log Locations

---

NOTE: If client authentication is not enabled for external CA certificates, then the Data Sync Status and Kafka Cluster status will display correct values only if the Diagnostics page is accessed from a server in which the Kafka role is enabled.

Debugging Issues and Log Locations

Configuration Issues

For any issue related to adding a Kafka broker, you can refer to either one of the logs on the server on which the Kafka role is to be enabled:

- `/var/opt/microfocus/log/zenworks/zeus.log`
- `/var/opt/microfocus/log/zenworks/zman.log`

General Debugging

Kafka Broker Issues: If the status of **Kafka broker** in the Diagnostics page is **Not Running**, then check whether the port 9093 of the server is not being used by any other component. You can also check the Kafka logs within the `server.logs` file available at `/var/opt/microfocus/log/zenworks/kafka-logs`.

Schema Registry Issues: If the status of **Schema Registry** in the Diagnostics page is **Not Running,** then check `schema-registry.log` available at `/var/opt/microfocus/log/zenworks/kafka-logs`.

Kafka Connect Issues: If the status of **Kafka Connect** in the Diagnostics page is **Not Running**, then check whether the port 8083 of the server is not being used by any other component. You can also check the `kafka-connect.log` available at `/var/opt/microfocus/log/zenworks/kafka-logs`.

If due to some reason any of the Kafka connectors displayed in the **Data Sync** section of the **Diagnostics** page, have failed, then you need to reconfigure the Kafka connector and restart the Kafka-connect service. To reconfigure the connector run the following command:

```
zman server-role-kafka-reconfigure-connectors -c <name of the connector>
```

You can also retrieve the names of all the connectors in the command line utility by running the `zman server-role-kafka-list-connectors` command. After reconfiguring the Kafka connector, restart the Kafka-connect service by running the following command:

```
systemctl restart microfocus-zenkafka-connect.service
```

Also, if the RDBMS is down for more than an hour, the Kafka connectors will not be able to sync data between Vertica and the RDBMS. You need to restart the Kafka.connect service by running the command:

```
systemctl restart microfocus-zenkafka-connect.service
```

Disaster Recovery and Preparedness

In the event of a failure of the Primary Server in which Kafka is enabled, you need to replace the device hosting the Primary Server with a new device that has the same identity, that is the same host name and IP address as that of the old device. For more information, see Replacing an Existing Primary Server with a New Primary Server in the ZENworks Disaster Recovery Reference.

If the replication count is 1, then before decommissioning the last server that has a Kafka role enabled in it, run the `zman server-role-kafka-add-broker (zman srkab)` to add a new broker and then proceed with the decommissioning of the existing server.

However, if you have already decommissioned the server, ensure that you bring up a server with the same hostname by backing up and restoring the Kafka data. For more information, see Replacing an Existing Primary Server with a New Primary Server in the ZENworks Disaster Recovery Reference.

Changing the Kafka role from the first Primary Server to the second Primary Server

Perform the following steps only if the first Primary Server is up and running. In case of a failure of the first Primar Server, see Replacing an Existing Primary Server with a New Primary Server in the ZENworks Disaster Recovery Reference.

1. **Add the second Primary Server to the Kafka cluster:** Run the command `zman srkab --servers=<GUID, path of the server object relative to /Devices/Servers, or the DNS of the second Primary Server>` in the command line utility.
2. **Remove the first Primary Server:** Run the command `zman srkrb --servers= <GUID, path of the server object relative to /Devices/Servers, or the DNS of the first Primary Server>` in the command line utility.

Troubleshooting

Addition of another server to the Kafka cluster fails after the only server that was part of the Kafka cluster is decommissioned

Explanation: Consider a scenario, where the replication count is 1 and the only server (broker) that was added to the Kafka cluster is decommissioned. On executing the command `zman server-role-kafka-add-broker (zman srkab)` to add another server (broker) in the zone, the operation to add the broker to the Kafka cluster fails. The error messages **Schema registry started – false** and **Some of the processes have failed to start. Marking queue as failed** are displayed in the `zenworks loader-messages.log`.

Action: Refer to the following steps to clean up both the ZooKeeper and Kafka data directory:

For ZooKeeper:

1. Stop all the ZENworks services on the Primary Server on which the ZooKeeper role is enabled. For more information, see Stopping the ZENworks Services.
2. Remove the `zookeeper.properties` file from the path `/etc/opt/microfocus/ZENworks/conf` (on Linux).

3. Remove the ZooKeeper data directory from the path `/var/opt/microfocus/ZENworks/zookeeper` (on Linux).
4. Restart all the ZENworks services. For more information, see Starting the ZENworks Services.

For Kafka:

1. Stop all the ZENworks services on the server on which the zman command to add the Kafka broker was executed but failed. For more information, see Stopping the ZENworks Services.
2. Remove the Kafka data directory from the path `/var/opt/microfocus/zenworks/kafka-data`.
3. Restart all the ZENworks services. For more information, see Starting the ZENworks Services.

NOTE: If the replication count is 1, then before decommissioning the last server that has a Kafka role enabled in it, run the `zman server-role-kafka-add-broker (zman srkab)` to add a new broker and then proceed with the decommissioning of the existing server.

However, if you have already decommissioned the server, ensure that you bring up a server with the same hostname by backing up and restoring the Kafka data. For more information, see Replacing an Existing Primary Server with a New Primary Server in the ZENworks Disaster Recovery Reference.

An error is displayed when adding a Kafka node

Explanation: When you try to add a Kafka node, an error was displayed. This issue occurs because the node's containers from the previous instance still exists.

Action: Run the following command to remove all the containers:

```
docker container --prune
```

At least one Kafka connector has failed

Explanation: If for any reason any of the Kafka connectors have failed, which is displayed in the **Connector Status** panel of the Diagnostics page in ZCC, then the failed connectors will not be able to sync data from their associated RDBMS tables to Vertica.

Action: You should enable debug logging for the zone if not already enabled. In ZCC, navigate to **Configuration** > **Device Management** > **Local Device Logging**. Select the **Errors, Warnings, Info, Debug** option from the **Log message to a local file if severity** drop-down list and click **Apply**.

Navigate to the ZENworks **Diagnostics** page and expand the **Connector Status** panel. Next, open the `services-messages.log` file of the server where you have logged in and search for the *Schema being registered is incompatible with an earlier schema* error. The error message is available in the *Trace* section under *Status*. Here is an example:

"ZENconnector-ZVDEVICE":{

```
"status":{
  "name":"ZENconnector-ZVDEVICE",
  "connector":{
    "state":"RUNNING",
    "worker_id":"zoneoraclapp3ps.epm.blr.novell.com:8083"
  },
  "tasks":"(("{
    "id":0,
    "state":"FAILED",
    "worker_id":"zoneoraclapp3ps.epm.blr.novell.com:8083",
    "trace":"...  Schema being registered is incompatible with an earlier schema ...",
    "type":"source"
  }
}
```


If the error is logged in the `services-messages.log` file, then run the command `zman server-role-kafka-reconfigure-connectors -c <name of the connector>` to reconfigure the failed Kafka connector.

Specify the `<name of the connector>` in the command. You can obtain the name of the failed connector from the `services-messages.log`file.

If the *Schema being registered is incompatible with an earlier schema error* message is not logged in the `services-messages.log` file, you can ignore the connector status as the application will try to reconfigure the failed connector during the next recovery attempt.

After reconfiguring the Kafka connectors, you need to restart the Kafka-connect service by running the following command:

```
systemctl restart microfocus-zenkafka-connect.service
```

Glossary

This section provides a description of some of the terminologies that you will come across in this document.

- **Kafka Cluster:** A group of servers or nodes that are connected to each other to achieve a common objective is a cluster. Each of the servers or nodes in the cluster, will have one instance of Kafka broker running.
- **Kafka Broker:** One or more servers that are added in a Kafka cluster are called Brokers. These are typically message brokers that act as mediators between two systems, ensuring that messages are delivered to the correct systems.
- **Kafka Topics:** Kafka stores streams of records in categories called Topics. These topics are further broken down into partitions that can be stored across multiple servers and disks. Topics in Kafka are always multi-subscriber; that is, a topic can have zero, one, or many consumers that subscribe to the data written to it.

- **Kafka Partitions:** Kafka topics are further broken down into partitions that enables topics to be replicated across multiple brokers for fault tolerance.
- **Kafka Producers:** The processes that publish messages to one or more Kafka topics are called Producers. In this case, the producer is the RDBMS system.
- **Kafka Consumers:** The processes that consumes messages from the Kafka topics are called consumers. In this case, the consumer is Vertica.
- **Kafka Replicas:** A replica or a copy of a partition is essentially used to prevent loss of data.
- **Kafka Leader:** A node that handles all read and write requests for a partition is called a leader.
- **Kafka Follower:** A node that replicates the leader are called followers. If the leader fails, one of the followers will automatically become a leader.
- **Kafka Connect:** Kafka Connect is a tool included with Kafka that imports and exports data to Kafka. It is an extensible tool that runs connectors, which implement the custom logic for interacting with an external system. For example, a connector to an RDBMS system captures every change in data made in a table.
- **Kafka Schema Registry:** Kafka producers write data to Kafka topics and Kafka consumers read data from Kafka topics. Schema Registry helps ensure that producers write data with a schema that can be read by consumers, even as producers and consumers evolve their schemas.
- **Heartbeat Requests:** ZooKeeper sends heartbeat requests to determine whether the Kafka broker is alive.
- **Offset:** A Kafka topic receives messages across a set of partitions and each partition maintains these messages in a sequential order, which is identified as an offset, also known as a position.

Legal Notice