



Red Hat AMQ Streams 1.0-Beta

Using AMQ Streams on Red Hat Enterprise Linux (RHEL)

Red Hat AMQ Streams 1.0-Beta Using AMQ Streams on Red Hat Enterprise Linux (RHEL)

Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This guide describes how to install, configure, and manage Red Hat AMQ Streams to build a large-scale messaging network.

Table of Contents

CHAPTER 1. OVERVIEW OF AMQ STREAMS	4
1.1. KEY FEATURES	4
1.2. SUPPORTED CONFIGURATIONS	5
1.3. DOCUMENT CONVENTIONS	5
CHAPTER 2. GETTING STARTED	6
2.1. AMQ STREAMS DISTRIBUTION	6
2.2. DOWNLOADING AN AMQ STREAMS ARCHIVE	6
2.3. INSTALLING AMQ STREAMS	6
2.4. RUNNING SINGLE NODE AMQ STREAMS CLUSTER	7
2.5. USING THE CLUSTER	8
2.6. STOPPING THE AMQ STREAMS SERVICES	9
2.7. CONFIGURING AMQ STREAMS	9
CHAPTER 3. CONFIGURING ZOOKEEPER	11
3.1. BASIC CONFIGURATION	11
3.2. ZOOKEEPER CLUSTER CONFIGURATION	11
3.3. RUNNING MULTI-NODE ZOOKEEPER CLUSTER	13
3.4. AUTHENTICATION	14
3.4.1. Authentication with SASL	14
3.4.2. Enabling Server-to-server authentication using DIGEST-MD5	16
3.4.3. Enabling Client-to-server authentication using DIGEST-MD5	17
3.5. AUTHORIZATION	19
3.6. TLS	19
3.7. ADDITIONAL CONFIGURATION OPTIONS	19
3.8. LOGGING	19
CHAPTER 4. CONFIGURING KAFKA	21
4.1. ZOOKEEPER	21
4.2. LISTENERS	21
4.3. COMMIT LOGS	22
4.4. BROKER ID	22
4.5. RUNNING A MULTI-NODE KAFKA CLUSTER	23
4.6. ZOOKEEPER AUTHENTICATION	24
4.6.1. JAAS Configuration	24
4.6.2. Enabling Zookeeper authentication	24
4.7. ZOOKEEPER AUTHORIZATION	25
4.7.1. ACL Configuration	25
4.7.2. Enabling Zookeeper ACLs for a new Kafka cluster	26
4.7.3. Enabling Zookeeper ACLs in an existing Kafka cluster	26
4.8. ENCRYPTION AND AUTHENTICATION	27
4.8.1. Listener configuration	27
4.8.2. TLS Encryption	28
4.8.3. Enabling TLS encryption	29
4.8.4. Authentication	30
4.8.4.1. TLS client authentication	30
4.8.4.2. SASL authentication	30
4.8.5. Enabling TLS client authentication	34
4.8.6. Enabling SASL PLAIN authentication	34
4.8.7. Enabling SASL SCRAM authentication	35
4.8.8. Adding SASL SCRAM users	36
4.8.9. Deleting SASL SCRAM users	37

4.9. LOGGING	38
CHAPTER 5. TOPICS	39
5.1. PARTITIONS AND REPLICAS	39
5.2. MESSAGE RETENTION	39
5.3. TOPIC AUTO-CREATION	40
5.4. TOPIC DELETION	40
5.5. TOPIC CONFIGURATION	40
5.6. INTERNAL TOPICS	41
5.7. CREATING A TOPIC	42
5.8. LISTING AND DESCRIBING TOPICS	43
5.9. MODIFYING A TOPIC CONFIGURATION	43
5.10. DELETING A TOPIC	45
CHAPTER 6. SCALING CLUSTERS	47
6.1. SCALING KAFKA CLUSTERS	47
6.1.1. Adding brokers to a cluster	47
6.1.2. Removing brokers from the cluster	47
6.2. REASSIGNMENT OF PARTITIONS	47
6.2.1. Reassignment JSON file	48
6.2.2. Generating reassignment JSON files	48
6.2.3. Creating reassignment JSON files manually	49
6.3. REASSIGNMENT THROTTLES	49
6.4. SCALING UP A KAFKA CLUSTER	49
6.5. SCALING DOWN A KAFKA CLUSTER	51
CHAPTER 7. KAFKA CONNECT	53
7.1. KAFKA CONNECT IN STANDALONE MODE	53
7.1.1. Configuring Kafka Connect in standalone mode	53
7.1.2. Configuring connectors in Kafka Connect in standalone mode	54
7.1.3. Running Kafka Connect in standalone mode	54
7.2. KAFKA CONNECT IN DISTRIBUTED MODE	55
7.2.1. Configuring Kafka Connect in distributed mode	55
7.2.2. Configuring connectors in distributed Kafka Connect	56
7.2.3. Running distributed Kafka Connect	57
7.2.4. Creating connectors	58
7.2.5. Deleting connectors	59
7.3. CONNECTOR PLUG-INS	59
7.4. ADDING CONNECTOR PLUGINS	60
APPENDIX A. BROKER CONFIGURATION PARAMETERS	62
APPENDIX B. TOPIC CONFIGURATION PARAMETERS	121
APPENDIX C. CONSUMER CONFIGURATION PARAMETERS	140
APPENDIX D. PRODUCER CONFIGURATION PARAMETERS	170
APPENDIX E. ADMIN CLIENT CONFIGURATION PARAMETERS	202
APPENDIX F. KAFKA CONNECT CONFIGURATION PARAMETERS	220
APPENDIX G. KAFKA STREAMS CONFIGURATION PARAMETERS	253

CHAPTER 1. OVERVIEW OF AMQ STREAMS

Red Hat AMQ Streams is a massively-scalable, distributed, and high-performance data streaming platform based on the Apache Zookeeper and Apache Kafka projects. It consists of the following main components:

Zookeeper

Service for highly reliable distributed coordination.

Kafka Broker

Messaging broker responsible for delivering records from producing clients to consuming clients.

Kafka Connect

A toolkit for streaming data between Kafka brokers and other systems using *Connector* plugins.

Kafka Consumer and Producer APIs

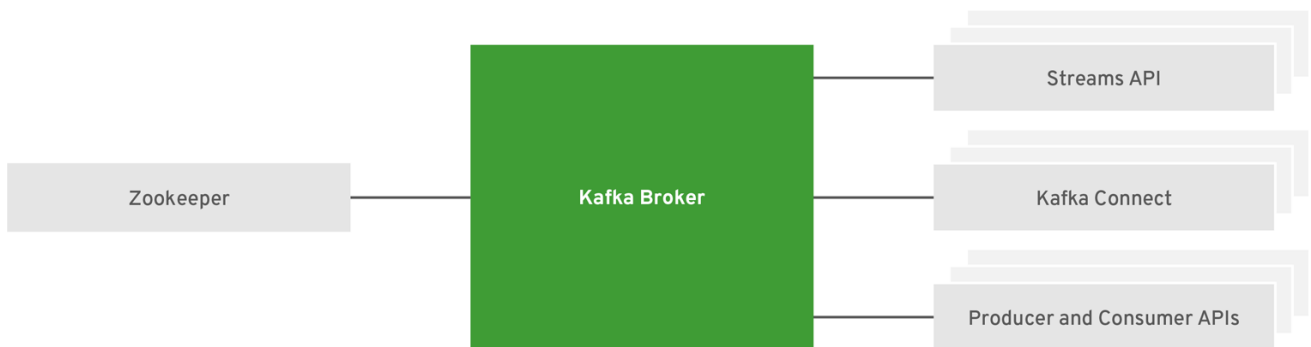
Java based APIs for producing and consuming messages to / from Kafka brokers.

Kafka Streams API

API for writing *stream processor* applications.

Kafka Broker is the central point connecting all these components. The broker uses Apache Zookeeper for storing configuration data and for cluster coordination. Before running Apache Kafka, an Apache Zookeeper cluster has to be ready.

Figure 1.1. Example Architecture diagram of AMQ Streams



AMQ_478987_0918

1.1. KEY FEATURES

- Scalability and performance
 - Designed for horizontal scalability
- Message ordering guarantee
 - At partition level
- Message rewind/replay
 - "Long term" storage
 - Allows to reconstruct application state by replaying the messages
 - Combined with compacted topics allows to use Kafka as key-value store

1.2. SUPPORTED CONFIGURATIONS

In order to be running in a supported configuration, AMQ Streams must be running in one of the following JVM versions and on one of the supported operating systems.

Table 1.1. List of supported Java Virtual Machines

Java Virtual Machine	Version
OpenJDK	1.8
OracleJDK	1.8
IBM JDK	1.8

Table 1.2. List of supported Operating Systems

Operating System	Architecture	Version
Red Hat Enterprise Linux	x86_64	7.x

1.3. DOCUMENT CONVENTIONS

Replaceables

In this document, replaceable text is styled in monospace and surrounded by angle brackets.

For example, in the following code, you will want to replace **<bootstrap-address>** and **<topic-name>** with your own address and topic name:

```
bin/kafka-console-consumer.sh --bootstrap-server <bootstrap-address> --  
topic <topic-name> --from-beginning
```

CHAPTER 2. GETTING STARTED

2.1. AMQ STREAMS DISTRIBUTION

AMQ Streams is distributed as single ZIP file. This ZIP file contains all AMQ Streams components:

- Apache Zookeeper
- Apache Kafka
- Apache Kafka Connect
- Apache Kafka Mirror Maker

2.2. DOWNLOADING AN AMQ STREAMS ARCHIVE

An archived distribution of AMQ Streams is available for download from the Red Hat website. You can download a copy of the distribution by following the steps below.

Procedure

- Download the AMQ Streams archive from the [Customer Portal](#).

2.3. INSTALLING AMQ STREAMS

Prerequisites

- Download the [installation archive](#).
- Review the [Section 1.2, “Supported Configurations”](#)

Procedure

1. Add new **kafka** user and group.

```
sudo groupadd kafka
sudo useradd -g kafka kafka
sudo passwd kafka
```

2. Create directory **/opt/kafka**.

```
sudo mkdir /opt/kafka
```

3. Create a temporary directory and extract the contents of the AMQ Streams ZIP file.

```
mkdir /tmp/kafka
unzip kafka_y.y-x.x.x.zip -d /tmp/kafka
```

4. Move the extracted contents into **/opt/kafka** directory and delete the temporary directory.

```
sudo mv /tmp/kafka/kafka_y.y-x.x.x/* /opt/kafka/
rm -r /tmp/kafka
```

5. Change the ownership of the **/opt/kafka** directory to the **kafka** user.

```
sudo chown -R kafka:kafka /opt/kafka
```

6. Create directory **/var/lib/zookeeper** for storing Zookeeper data and set its ownership to the **kafka** user.

```
sudo mkdir /var/lib/zookeeper
sudo chown -R kafka:kafka /var/lib/zookeeper
```

7. Create directory **/var/lib/kafka** for storing Kafka data and set its ownership to the **kafka** user.

```
sudo mkdir /var/lib/kafka
sudo chown -R kafka:kafka /var/lib/kafka
```

2.4. RUNNING SINGLE NODE AMQ STREAMS CLUSTER

This procedure will show you how to run a basic AMQ Streams cluster consisting of single Zookeeper and single Apache Kafka node both running on the same host. It is using the default configuration files for both Zookeeper and Kafka.



WARNING

Single node AMQ Streams cluster does not provide realibility and high availability and is suitable only for development purposes.

Prerequisites

- AMQ Streams is installed on the host

Running the cluster

1. Edit the Zookeeper configuration file **/opt/kafka/config/zookeeper.properties**. Set the **dataDir** option to **/var/lib/zookeeper/**.

```
dataDir=/var/lib/zookeeper/
```

2. Start Zookeeper.

```
su - kafka
/opt/kafka/bin/zookeeper-server-start.sh -daemon
/opt/kafka/config/zookeeper.properties
```

3. Make sure that Apache Zookeeper is running.

```
jcmd | grep zookeeper
```

-
- 4. Edit the Kafka configuration file `/opt/kafka/config/server.properties`. Set the `log.dirs` option to `/var/lib/kafka/`.

```
log.dirs=/var/lib/kafka/
```

- 5. Start Kafka.

```
su - kafka
/opt/kafka/bin/kafka-server-start.sh -daemon
/opt/kafka/config/server.properties
```

- 6. Make sure that Kafka is running.

```
jcmd | grep kafka
```

Additional resources

- For more information about installing AMQ Streams, see [Section 2.3, “Installing AMQ Streams”](#).
- For more information about configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).

2.5. USING THE CLUSTER

Prerequisites

- AMQ Streams is installed on the host
- Zookeeper and Kafka are up and running

Procedure

- 1. Start the Kafka console producer.

```
bin/kafka-console-producer.sh --broker-list <bootstrap-address> --
topic <topic-name>
```

For example:

```
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic
my-topic
```

- 2. Type your message into the console where the producer is running.
- 3. Press Enter to send.
- 4. Press Ctrl+C to exit the Kafka console producer.
- 5. Start the message receiver.

```
bin/kafka-console-consumer.sh --bootstrap-server <bootstrap-address>
--topic <topic-name> --from-beginning
```

■

For example:

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --
topic my-topic --from-beginning
```

6. Confirm that you see the incoming messages in the consumer console.
7. Press Ctrl+C to exit the Kafka console consumer.

2.6. STOPPING THE AMQ STREAMS SERVICES

You can stop the Kafka and Zookeeper services by running a script. All connections to the Kafka and Zookeeper services will be terminated.

Prerequisites

- AMQ Streams is installed on the host
- Zookeeper and Kafka are up and running

Procedure

1. Stop the Kafka broker.

```
su - kafka
/opt/kafka/bin/kafka-server-stop.sh
```

2. Confirm that the Kafka broker is stopped.

```
jcmd | grep kafka
```

3. Stop Zookeeper.

```
su - kafka
/opt/kafka/bin/zookeeper-server-stop.sh
```

2.7. CONFIGURING AMQ STREAMS

Prerequisites

- AMQ Streams is downloaded and installed on the host

Procedure

1. Open Zookeeper and Kafka broker configuration files in a text editor. The configuration files are located at :

Zookeeper

`/opt/kafka/config/zookeeper.properties`

Kafka

/opt/kafka/config/server.properties

2. Edit the configuration options. The configuration files are in the Java properties format. Every configuration option should be on separate line in the following format:

```
<option> = <value>
```

Lines starting with # or ! will be treated as comments and will be ignored by AMQ Streams components.

```
# This is a comment
```

Values can be split into multiple lines by using \ directly before the newline / carriage return.

```
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \  
    username="bob" \  
    password="bobs-password";
```

3. Save the changes
4. Restart the Zookeeper or Kafka broker
5. Repeat this procedure on all the nodes of the cluster.

CHAPTER 3. CONFIGURING ZOOKEEPER

Kafka uses Zookeeper to store configuration data and for cluster coordination. It is strongly recommended to run a cluster of replicated Zookeeper instances.

3.1. BASIC CONFIGURATION

The most important Zookeeper configuration options are:

tickTime

Zookeeper's basic time unit in milliseconds. It is used for heartbeats and session timeouts. For example, minimum session timeout will be two ticks.

dataDir

The directory where Zookeeper stores its transaction log and snapshots of its in-memory database. This should be set to the `/var/lib/zookeeper/` directory created during installation.

clientPort

Port number where clients can connect. Defaults to **2181**.

An example zookeeper configuration file `config/zookeeper.properties` is located in the AMQ Streams installation directory. It is recommended to place the `dataDir` directory on a separate disk device to minimize the latency in Zookeeper.

Zookeeper configuration file should be located in `/opt/kafka/config/zookeeper.properties`. A basic example of the configuration file can be found below. The configuration file has to be readable by the `kafka` user.

```
timeTick=2000
dataDir=/var/lib/zookeeper/
clientPort=2181
```

3.2. ZOOKEEPER CLUSTER CONFIGURATION

For reliable ZooKeeper service, you should deploy ZooKeeper in a cluster. Hence, for production use cases, you must run a cluster of replicated Zookeeper instances. Zookeeper clusters are also referred as ensembles.

Zookeeper clusters usually consist of an odd number of nodes. Zookeeper requires a majority of the nodes in the cluster should be up and running. For example, a cluster with three nodes, at least two of them must be up and running. This means it can tolerate one node being down. A cluster consisting of five nodes, at least three nodes must be available. This means it can tolerate two nodes being down. A cluster consisting of seven nodes, at least four nodes must be available. This means it can tolerate three nodes being down. Having more nodes in the Zookeeper cluster delivers better resiliency and reliability of the whole cluster.

Zookeeper can run in clusters with an even number of nodes. The additional node, however, does not increase the resiliency of the cluster. A cluster with four nodes requires at least three nodes to be available and can tolerate only one node being down. Therefore it has exactly the same resiliency as a cluster with only three nodes.

The different Zookeeper nodes should be ideally placed into different data centers or network segments. Increasing the number of Zookeeper nodes increases the workload spent on cluster synchronization. For most Kafka use cases Zookeeper cluster with 3, 5 or 7 nodes should be fully sufficient.

**WARNING**

Zookeeper cluster with 3 nodes can tolerate only 1 unavailable node. This means that when a cluster node crashes while you are doing maintenance on another node your Zookeeper cluster will be unavailable.

Replicated Zookeeper configuration supports all configuration options supported by the standalone configuration. Additional options are added for the clustering configuration:

initLimit

Amount of time to allow followers to connect and sync to the cluster leader. The time is specified as a number of ticks (see the [timeTick option](#) for more details).

syncLimit

Amount of time for which followers can be behind the leader. The time is specified as a number of ticks (see the [timeTick option](#) for more details).

In addition to the options above, every configuration file should contain a list of servers which should be members of the Zookeeper cluster. The server records should be specified in the format

server.id=hostname:port1:port2, where:

id

The ID of the Zookeeper cluster node.

hostname

The hostname or IP address where the node listens for connections.

port1

The port number used for intra-cluster communication.

port2

The port number used for leader election.

The following is an example configuration file of a Zookeeper cluster with three nodes:

```
timeTick=2000
dataDir=/var/lib/zookeeper/
clientPort=2181
initLimit=5
syncLimit=2

server.1=172.17.0.1:2888:3888
server.2=172.17.0.2:2888:3888
server.3=172.17.0.3:2888:3888
```

Each node in the Zookeeper cluster has to be assigned with a unique **ID**. Each node's **ID** has to be configured in **myid** file and stored in the **dataDir** folder like **/var/lib/zookeeper/**. The **myid** files should contain only a single line with the written **ID** as text. The **ID** can be any integer from 1 to 255. You must manually create this file on each cluster node. Using this file, each Zookeeper instance will use the configuration from the corresponding **server .** line in the configuration file to configure its listeners. It will also use all other **server .** lines to identify other cluster members.

In the above example, there are three nodes, so each one will have a different **myid** with values **1**, **2**, and **3** respectively.

3.3. RUNNING MULTI-NODE ZOOKEEPER CLUSTER

This procedure will show you how to configure and run Zookeeper as a multi-node cluster.

Prerequisites

- AMQ Streams is installed on all hosts which will be used as Zookeeper cluster nodes.

Running the cluster

1. Create the **myid** file in **/var/lib/zookeeper/**. Enter ID **1** for the first Zookeeper node, **2** for the second Zookeeper node, and so on.

```
su - kafka
echo "<NodeID>" > /var/lib/zookeeper/myid
```

For example:

```
su - kafka
echo "1" > /var/lib/zookeeper/myid
```

2. Edit the Zookeeper **/opt/kafka/config/zookeeper.properties** configuration file for the following:
 - Set the option **dataDir** to **/var/lib/zookeeper/**. + Configure the **initLimit** and **syncLimit** options.
 - Add a list of all Zookeeper nodes. The list should include also the current node.

Example configuration for a node of Zookeeper cluster with five members

```
timeTick=2000
dataDir=/var/lib/zookeeper/
clientPort=2181
initLimit=5
syncLimit=2

server.1=172.17.0.1:2888:3888
server.2=172.17.0.2:2888:3888
server.3=172.17.0.3:2888:3888
server.4=172.17.0.4:2888:3888
server.5=172.17.0.5:2888:3888
```

3. Start Zookeeper with the default configuration file.

```
su - kafka
/opt/kafka/bin/zookeeper-server-start.sh -daemon
/opt/kafka/config/zookeeper.properties
```

4. Verify that the Zookeeper is running.

```
jcmb | grep zookeeper
```

5. Repeat this procedure on all the nodes of the cluster.
6. Once all nodes of the clusters are up and running, verify that all nodes are members of the cluster by sending a **stat** command to each of the nodes using **ncat** utility.

Use ncat stat to check the node status

```
echo stat | ncat localhost 2181
```

In the output you should see information that the node is either **leader** or **follower**.

Example output from the ncat command

```
Zookeeper version: 3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03,
built on 06/29/2018 00:39 GMT
Clients:
 /0:0:0:0:0:0:0:1:59726[0](queued=0,recved=1,sent=0)

Latency min/avg/max: 0/0/0
Received: 2
Sent: 1
Connections: 1
Outstanding: 0
Zxid: 0x2000000000
Mode: follower
Node count: 4
```

Additional resources

- For more information about installing AMQ Streams, see [Section 2.3, “Installing AMQ Streams”](#).
- For more information about configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).

3.4. AUTHENTICATION

By default, Zookeeper does not use any form of authentication and allows anonymous connections. However, it supports Java Authentication and Authorization Service (JAAS) which can be used to set up authentication using Simple Authentication and Security Layer (SASL). Zookeeper supports authentication using the DIGEST-MD5 SASL mechanism with locally stored credentials.

3.4.1. Authentication with SASL

JAAS is configured using a separate configuration file. It is recommended to place the JAAS configuration file in the same directory as the Zookeeper configuration (**/opt/kafka/config/**). The recommended file name is **zookeeper-jaa.conf**. When using a Zookeeper cluster with multiple nodes, the JAAS configuration file has to be created on all cluster nodes.

JAAS is configured using contexts. Separate parts such as the server and client are always configured with a separate *context*. The context is a *configuration* option and has the following format:

```
ContextName {
    param1
    param2;
};
```

SASL Authentication is configured separately for server-to-server communication (communication between Zookeeper instances) and client-to-server communication (communication between Kafka and Zookeeper). Server-to-server authentication is relevant only for Zookeeper clusters with multiple nodes.

Server-to-Server authentication

For server-to-server authentication, the JAAS configuration file contains two parts:

- The server configuration
- The client configuration

When using DIGEST-MD5 SASL mechanism, the **QuorumServer** context is used to configure the authentication server. It must contain all the usernames to be allowed to connect together with their passwords in an unencrypted form. The second context, **QuorumLearner**, has to be configured for the client which is built into Zookeeper. It also contains the password in an unencrypted form. An example of the JAAS configuration file for DIGEST-MD5 mechanism can be found below:

```
QuorumServer {
    org.apache.zookeeper.server.auth.DigestLoginModule required
    user_zookeeper="123456";
};

QuorumLearner {
    org.apache.zookeeper.server.auth.DigestLoginModule required
    username="zookeeper"
    password="123456";
};
```

In addition to the JAAS configuration file, you must enable the server-to-server authentication in the regular Zookeeper configuration file by specifying the following options:

```
quorum.auth.enableSasl=true
quorum.auth.learnerRequireSasl=true
quorum.auth.serverRequireSasl=true
quorum.auth.learner.loginContext=QuorumLearner
quorum.auth.server.loginContext=QuorumServer
quorum.cnxn.threads.size=20
```

Use the **EXTRA_ARGS** environment variable to pass the JAAS configuration file to the Zookeeper server as a Java property:

```
su - kafka
export EXTRA_ARGS="-
Djava.security.auth.login.config=/opt/kafka/config/zookeeper-jaas.conf";
/opt/kafka/bin/zookeeper-server-start.sh -daemon
/opt/kafka/config/zookeeper.properties
```

For more information about server-to-server authentication, see [Zookeeper wiki](#).

Client-to-Server authentication

Client-to-server authentication is configured in the same JAAS file as the server-to-server authentication. However, unlike the server-to-server authentication, it contains only the server configuration. The client part of the configuration has to be done in the client. For information on how to configure a Kafka broker to connect to Zookeeper using authentication, see the [Kafka installation](#) section.

Add the Server context to the JAAS configuration file to configure client-to-server authentication. For DIGEST-MD5 mechanism it configures all usernames and passwords:

```
Server {
    org.apache.zookeeper.server.auth.DigestLoginModule required
    user_super="123456"
    user_kafka="123456"
    user_someoneelse="123456";
};
```

After configuring the JAAS context, enable the client-to-server authentication in the Zookeeper configuration file by adding the following line:

```
requireClientAuthScheme=sasl
authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
authProvider.2=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
authProvider.3=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
```

You must add the **authProvider.<ID>** property for every server that is part of the Zookeeper cluster.

Use the **EXTRA_ARGS** environment variable to pass the JAAS configuration file to the Zookeeper server as a Java property:

```
su - kafka
export EXTRA_ARGS="-
Djava.security.auth.login.config=/opt/kafka/config/zookeeper-jaas.conf";
/opt/kafka/bin/zookeeper-server-start.sh -daemon
/opt/kafka/config/zookeeper.properties
```

For more information about configuring Zookeeper authentication in Kafka brokers, see [Section 4.6, “Zookeeper authentication”](#).

3.4.2. Enabling Server-to-server authentication using DIGEST-MD5

This procedure describes how to enable authentication using the SASL DIGEST-MD5 mechanism between the nodes of the Zookeeper cluster.

Prerequisites

- AMQ Streams is installed on the host
- Zookeeper cluster is [configured](#) with multiple nodes.

Enabling SASL DIGEST-MD5 authentication

1. On all Zookeeper nodes, create or edit the `/opt/kafka/config/zookeeper-jaas.conf` JAAS configuration file and add the following contexts:

■

```

QuorumServer {
    org.apache.zookeeper.server.auth.DigestLoginModule required
    user__<Username>_="<Password>";
};

QuorumLearner {
    org.apache.zookeeper.server.auth.DigestLoginModule required
    username="<Username>"
    password="<Password>";
};

```

The username and password must be the same in both JAAS contexts. For example:

```

QuorumServer {
    org.apache.zookeeper.server.auth.DigestLoginModule required
    user_zookeeper="123456";
};

QuorumLearner {
    org.apache.zookeeper.server.auth.DigestLoginModule required
    username="zookeeper"
    password="123456";
};

```

2. On all Zookeeper nodes, edit the `/opt/kafka/config/zookeeper.properties` Zookeeper configuration file and set the following options:

```

quorum.auth.enableSasl=true
quorum.auth.learnerRequireSasl=true
quorum.auth.serverRequireSasl=true
quorum.auth.learner.loginContext=QuorumLearner
quorum.auth.server.loginContext=QuorumServer
quorum.cnxn.threads.size=20

```

3. Restart all Zookeeper nodes one by one. To pass the JAAS configuration to Zookeeper, use the **EXTRA_ARGS** environment variable.

```

su - kafka
export EXTRA_ARGS="-
Djava.security.auth.login.config=/opt/kafka/config/zookeeper-
jaas.conf"; /opt/kafka/bin/zookeeper-server-start.sh -daemon
/opt/kafka/config/zookeeper.properties

```

Additional resources

- For more information about installing AMQ Streams, see [Section 2.3, “Installing AMQ Streams”](#).
- For more information about configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).
- For more information about running a Zookeeper cluster, see [Section 3.3, “Running multi-node Zookeeper cluster”](#).

3.4.3. Enabling Client-to-server authentication using DIGEST-MD5

This procedure describes how to enable authentication using the SASL DIGEST-MD5 mechanism between Zookeeper clients and Zookeeper.

Prerequisites

- AMQ Streams is installed on the host
- Zookeeper cluster is [configured and running](#).

Enabling SASL DIGEST-MD5 authentication

1. On all Zookeeper nodes, create or edit the `/opt/kafka/config/zookeeper-jaa.conf` JAAS configuration file and add the following context:

```
Server {
    org.apache.zookeeper.server.auth.DigestLoginModule required
    user_super="<SuperUserPassword>"
    user<Username1>_="<Password1>" user<Username2>_="<Password2>";
};
```

The **super** will have automatically administrator privileges. The file can contain multiple users, but only one additional user is required by the Kafka brokers. The recommended name for the Kafka user is **kafka**.

The following example shows the **Server** context for client-to-server authentication:

```
Server {
    org.apache.zookeeper.server.auth.DigestLoginModule required
    user_super="123456"
    user_kafka="123456";
};
```

2. On all Zookeeper nodes, edit the `/opt/kafka/config/zookeeper.properties` Zookeeper configuration file and set the following options:

```
requireClientAuthScheme=sasl
authProvider.<IdOfBroker1>=org.apache.zookeeper.server.auth.SASLAuth
enticationProvider
authProvider.<IdOfBroker2>=org.apache.zookeeper.server.auth.SASLAuth
enticationProvider
authProvider.<IdOfBroker3>=org.apache.zookeeper.server.auth.SASLAuth
enticationProvider
```

The **authProvider.<ID>** property has to be added for every node which is part of the Zookeeper cluster. An example three-node Zookeeper cluster configuration must look like the following:

```
requireClientAuthScheme=sasl
authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationPr
ovider
authProvider.2=org.apache.zookeeper.server.auth.SASLAuthenticationPr
ovider
authProvider.3=org.apache.zookeeper.server.auth.SASLAuthenticationPr
ovider
```

- Restart all Zookeeper nodes one by one. To pass the JAAS configuration to Zookeeper, use the **EXTRA_ARGS** environment variable.

```
su - kafka
export EXTRA_ARGS="-
Djava.security.auth.login.config=/opt/kafka/config/zookeeper-
jaas.conf"; /opt/kafka/bin/zookeeper-server-start.sh -daemon
/opt/kafka/config/zookeeper.properties
```

Additional resources

- For more information about installing AMQ Streams, see [Section 2.3, “Installing AMQ Streams”](#).
- For more information about configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).
- For more information about running a Zookeeper cluster, see [Section 3.3, “Running multi-node Zookeeper cluster”](#).

3.5. AUTHORIZATION

Zookeeper supports access control lists (ACLs) to protect data stored inside it. Kafka brokers can automatically configure the ACL rights for all Zookeeper records they create so no other Zookeeper user can modify them.

For more information about enabling Zookeeper ACLs in Kafka brokers, see [Section 4.7, “Zookeeper authorization”](#).

3.6. TLS

The version of Zookeeper which is part of AMQ Streams currently does not support TLS for encryption or authentication.

3.7. ADDITIONAL CONFIGURATION OPTIONS

You can set the following options based on your use case:

maxClientCnxns

The maximum number of concurrent client connections to a single member of the ZooKeeper cluster.

autopurge.snapRetainCount

Number of snapshots of Zookeeper’s in-memory database which will be retained. Default value is **3**.

autopurge.purgeInterval

The time interval in hours for purging snapshots. The default value is **0** and this option is disabled.

All available configuration options can be found in [Zookeeper documentation](#).

3.8. LOGGING

Zookeeper is using *log4j* as their logging infrastructure. Logging configuration is by default read from the **log4j.properties** configuration file which should be placed either in the **/opt/kafka/config/** directory or in the classpath. The location and name of the configuration file can be changed using the

Java property **log4j.configuration** which can be passed to Zookeeper using the **KAFKA_LOG4J_OPTS** environment variable:

```
su - kafka
export KAFKA_LOG4J_OPTS="-
Dlog4j.configuration=file:/my/path/to/log4j.properties";
/opt/kafka/bin/zookeeper-server-start.sh -daemon
/opt/kafka/config/zookeeper.properties
```

For more information about Log4j configurations, see [Log4j documentation](#).

CHAPTER 4. CONFIGURING KAFKA

Kafka uses a properties file to store static configuration. The recommended location for the configuration file is `/opt/kafka/config/server.properties`. The configuration file has to be readable by the `kafka` user.

AMQ Streams ships an example configuration file that highlights various basic and advanced features of the product. It can be found under `config/server.properties` in the AMQ Streams installation directory.

This chapter explains the most important configuration options. For a complete list of supported Kafka broker configuration options, see [Appendix A, Broker configuration parameters](#).

4.1. ZOOKEEPER

Kafka brokers need Zookeeper to store some parts of their configuration as well as to coordinate the cluster (for example to decide which node is a leader for which partition). Connection details for the Zookeeper cluster are stored in the configuration file. The field `zookeeper.connect` contains a comma-separated list of hostnames and ports of members of the zookeeper cluster.

For example:

```
zookeeper.connect=zoo1.my-domain.com:2181,zoo2.my-domain.com:2181,zoo3.my-domain.com:2181
```

Kafka will use these addresses to connect to the Zookeeper cluster. With this configuration, all Kafka **znodes** will be created directly in the root of Zookeeper database. Therefore, such a Zookeeper cluster could be used only for a single Kafka cluster. To configure multiple Kafka clusters to use single Zookeeper cluster, specify a base (prefix) path at the end of the Zookeeper connection string in the Kafka configuration file:

```
zookeeper.connect=zoo1.my-domain.com:2181,zoo2.my-domain.com:2181,zoo3.my-domain.com:2181/my-cluster-1
```

4.2. LISTENERS

Kafka brokers can be configured to use multiple listeners. Each listener can be used to listen on a different port or network interface and can have different configuration. Listeners are configured in the `listeners` property in the configuration file. The `listeners` property contains a list of listeners with each listener configured as `<listenerName>://<hostname>:<port>`. When the hostname value is empty, Kafka will use `java.net.InetAddress.getCanonicalHostName()` as hostname. The following example shows how multiple listeners might be configured:

```
listeners=INT1://:9092,INT2://:9093,REPLICATION://:9094
```

When a Kafka client wants to connect to a Kafka cluster, it first connects to a *bootstrap server*. The *bootstrap server* is one of the cluster nodes. It will provide the client with a list of all other brokers which are part of the cluster and the client will connect to them individually. By default the *bootstrap server* will provide the client with a list of nodes based on the `listeners` field.

Advertised listeners

It is possible to give the client a different set of addresses than given in the `listeners` property. It is useful

in situations when additional network infrastructure, such as a proxy, is between the client and the broker, or when an external DNS name should be used instead of an IP address. Here, the broker allows defining the advertised addresses of the listeners in the `advertised.listeners` configuration property. This property has the same format as the `listeners` property. The following example shows how to configure advertised listeners:

```
listeners=INT1://:9092,INT2://:9093
advertised.listeners=INT1://my-broker-1.my-domain.com:1234,INT2://my-broker-1.my-domain.com:1234:9093
```



NOTE

The names of the listeners have to match the names of the listeners from the **`listeners`** property.

Inter-broker listeners

When the cluster has replicated topics, the brokers responsible for such topics need to communicate with each other in order to replicate the messages in those topics. When multiple listeners are configured, the configuration field **`inter.broker.listener.name`** can be used to specify the name of the listener which should be used for replication between brokers. For example:

```
inter.broker.listener.name=REPLICATION
```

4.3. COMMIT LOGS

Apache Kafka stores all records it receives from producers in commit logs. The commit logs contain the actual data, in the form of records, that Kafka needs to deliver. These are not the application log files which record what the broker is doing.

Log directories

You can configure log directories using the **`log.dirs`** property filed to store commit logs in one or multiple log directories. It should be set to **`/var/lib/zookeeper`** directory created during installation:

```
log.dirs=/var/lib/zookeeper
```

For performance reasons, you can configure `log.dirs` to multiple directories and place each of them on a different physical device to improve disk I/O performance. For example:

```
log.dirs=/var/lib/zookeeper1,/var/lib/zookeeper2,/var/lib/zookeeper3
```

4.4. BROKER ID

Broker ID is a unique identifier for each broker in the cluster. You can assign an integer greater than or equal to 0 as broker ID. The broker ID is used to identify the brokers after restarts or crashes and it is therefore important that the id is stable and does not change over time. The broker ID is configured in the broker properties file:

```
broker.id=1
```

4.5. RUNNING A MULTI-NODE KAFKA CLUSTER

This procedure will show you how to configure and run Kafka as a multi-node cluster.

Prerequisites

- AMQ Streams is [installed](#) on all hosts which will be used as Kafka brokers.
- Zookeeper cluster is [configured and running](#).

Running the cluster

1. Edit the `/opt/kafka/config/server.properties` Kafka configuration file for the following:

- Set the `broker.id` field to `0` for the first broker, `1` for the second broker, and so on. + Configure the details for connecting to Zookeeper in the `zookeeper.connect` option. + Configure the Kafka listeners. + Set the directories where the commit logs should be stored in the `logs.dir` directory.

The following example shows the configuration for a Kafka broker:

```
broker.id=0
zookeeper.connect=zoo1.my-domain.com:2181,zoo2.my-
domain.com:2181,zoo3.my-domain.com:2181
listeners=REPLICATION://:9091,PLAINTEXT://:9092
inter.broker.listener.name=REPLICATION
log.dirs=/var/lib/zookeeper
```

In a typical installation where each Kafka broker is running on identical hardware, only the `broker.id` configuration property will differ between each broker config.

2. Start the Kafka broker with the default configuration file.

```
su - kafka
/opt/kafka/bin/kafka-server-start.sh -daemon
/opt/kafka/config/server.properties
```

3. Verify that the Kafka broker is running.

```
jcmd | grep Kafka
```

4. Repeat this procedure on all the nodes of the Kafka cluster.
5. Once all nodes of the clusters are up and running, verify that all nodes are members of the Kafka cluster by sending a `dump` command to one of the Zookeeper nodes using the `ncat` utility. The command will print all Kafka brokers registered in Zookeeper.

Use ncat stat to check the node status

```
echo dump | ncat zoo1.my-domain.com 2181
```

The output should contain all Kafka brokers you just configured and started.

Example output from the ncat command for Kafka cluster with 3 nodes

-

```

SessionTracker dump:
org.apache.zookeeper.server.quorum.LearnerSessionTracker@28848ab9
ephemeral nodes dump:
Sessions with Ephemerals (3):
0x20000015dd00000:
    /brokers/ids/1
0x10000015dc70000:
    /controller
    /brokers/ids/0
0x10000015dc70001:
    /brokers/ids/2

```

Additional resources

- For more information about installing AMQ Streams, see [Section 2.3, “Installing AMQ Streams”](#).
- For more information about configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).
- For more information about running a Zookeeper cluster, see [Section 3.3, “Running multi-node Zookeeper cluster”](#).
- For a complete list of supported Kafka broker configuration options, see [Appendix A, *Broker configuration parameters*](#).

4.6. ZOOKEEPER AUTHENTICATION

By default, connections between Zookeeper and Kafka are not authenticated. However, Kafka and Zookeeper support Java Authentication and Authorization Service (JAAS) which can be used to set up authentication using Simple Authentication and Security Layer (SASL). Zookeeper supports authentication using the DIGEST-MD5 SASL mechanism with locally stored credentials.

4.6.1. JAAS Configuration

SASL authentication for Zookeeper connections has to be configured in the JAAS configuration file. By default, Kafka will use the JAAS context named **Client** for connecting to Zookeeper. The **Client** context should be configured in the `/opt/kafka/config/jass.conf` file. The context has to enable the **PLAIN** SASL authentication, as in the following example:

```

Client {
    org.apache.kafka.common.security.plain.PlainLoginModule required
        username="kafka"
        password="123456";
};

```

4.6.2. Enabling Zookeeper authentication

This procedure describes how to enable authentication using the SASL DIGEST-MD5 mechanism when connecting to Zookeeper.

Prerequisites

- Client-to-server authentication is [enabled](#) in Zookeeper

Enabling SASL DIGEST-MD5 authentication

1. On all Kafka broker nodes, create or edit the `/opt/kafka/config/jaas.conf` JAAS configuration file and add the following context:

```
Client {
    org.apache.kafka.common.security.plain.PlainLoginModule required
        username="<Username>"
        password="<Password>";
};
```

The username and password should be the same as configured in Zookeeper.

Following example shows the **Client** context:

```
Client {
    org.apache.kafka.common.security.plain.PlainLoginModule required
        username="kafka"
        password="123456";
};
```

2. Restart all Kafka broker nodes one by one. To pass the JAAS configuration to Kafka brokers, use the **KAFKA_OPTS** environment variable.

```
su - kafka
export KAFKA_OPTS="-
Djava.security.auth.login.config=/opt/kafka/config/jaas.conf";
/opt/kafka/bin/kafka-server-start.sh -daemon
/opt/kafka/config/server.properties
```

Additional resources

- For more information about configuring client-to-server authentication in Zookeeper, see [Section 3.4, “Authentication”](#).

4.7. ZOOKEEPER AUTHORIZATION

When authentication is enabled between Kafka and Zookeeper, Kafka can be configured to automatically protect all its records with Access Control List (ACL) rules which will allow only the Kafka user to change the data. All other users will have read-only access.

4.7.1. ACL Configuration

Enforcement of ACL rules is controlled by the **zookeeper.set.acl** property in the **config/server.properties** Kafka configuration file and is disabled by default. To enable the ACL protection set **zookeeper.set.acl** to **true**:

```
zookeeper.set.acl=true
```

Kafka will set the ACL rules only for newly created Zookeeper **znodes**. When the ACLs are only enabled after the first start of the cluster, the tool **zookeeper-security-migration.sh** can be used to set ACLs on all existing **znodes**.

The data stored in Zookeeper includes information such as topic names and their configuration. The Zookeeper database also contains the salted and hashed user credentials when SASL SCRAM authentication is used. But it does not include any records sent and received using Kafka. Kafka, in general, considers the data stored in Zookeeper as non-confidential. In case these data are considered confidential (for example because topic names contain customer identification) the only way how to protect them is by isolating Zookeeper on the network level and allowing access only to Kafka brokers.

4.7.2. Enabling Zookeeper ACLs for a new Kafka cluster

This procedure describes how to enable Zookeeper ACLs in Kafka configuration for a new Kafka cluster. Use this procedure only before the first start of the Kafka cluster. For enabling Zookeeper ACLs in already running cluster, see [Section 4.7.3, “Enabling Zookeeper ACLs in an existing Kafka cluster”](#).

Prerequisites

- AMQ Streams is [installed](#) on all hosts which will be used as Kafka brokers.
- Zookeeper cluster is [configured and running](#).
- Client-to-server authentication is [enabled](#) in Zookeeper.
- Zookeeper authentication is [enabled](#) in the Kafka brokers.
- Kafka broker have not yet been started.

Procedure

1. Edit the `/opt/kafka/config/server.properties` Kafka configuration file to set the `zookeeper.set.acl` field to `true` on all cluster nodes.

```
zookeeper.set.acl=true
```

2. Start the Kafka brokers.

Additional resources

- For more information about installing AMQ Streams, see [Section 2.3, “Installing AMQ Streams”](#).
- For more information about configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).
- For more information about running a Zookeeper cluster, see [Section 3.3, “Running multi-node Zookeeper cluster”](#).
- For more information about running a Kafka cluster, see [Section 4.5, “Running a multi-node Kafka cluster”](#).

4.7.3. Enabling Zookeeper ACLs in an existing Kafka cluster

The `zookeeper-security-migration.sh` tool can to be used to set Zookeeper ACLs on all existing `znodes`. The `zookeeper-security-migration.sh` is available as part of AMQ Streams and can be found in the `bin` directory.

Prerequisites

- Kafka cluster is [configured and running](#).

Enabling the Zookeeper ACLs

1. Edit the `/opt/kafka/config/server.properties` Kafka configuration file to set the `zookeeper.set.acl` field to `true` on all cluster nodes.

```
zookeeper.set.acl=true
```

2. Restart all Kafka brokers one by one
3. Set the ACLs on all existing Zookeeper **znodes** using the `zookeeper-security-migration.sh` tool.

```
su - kafka
cd /opt/kafka
KAFKA_OPTS="-Djava.security.auth.login.config=./config/jaas.conf";
./bin/zookeeper-security-migration.sh --zookeeper.acl=secure --
zookeeper.connect=_<ZookeeperURL>_
exit
```

For example:

```
su - kafka
cd /opt/kafka
KAFKA_OPTS="-Djava.security.auth.login.config=./config/jaas.conf";
./bin/zookeeper-security-migration.sh --zookeeper.acl=secure --
zookeeper.connect=zoo1.my-domain.com:2181
exit
```

4.8. ENCRYPTION AND AUTHENTICATION

Kafka supports TLS for encrypting the communication with Kafka clients. Additionally, it supports two types of authentication:

- TLS client authentication based on X.509 certificates
- SASL Authentication based on a username and password

4.8.1. Listener configuration

Encryption and authentication in Kafka brokers is configured per listener. For more information about Kafka listener configuration, see [Section 4.2, “Listeners”](#).

Each listener in the Kafka broker is configured with its own security protocol. The configuration property `listener.security.protocol.map` defines which listener uses which security protocol. It maps each listener name to its security protocol. Supported security protocols are:

PLAINTEXT

Listener without any encryption or authentication.

SSL

Listener using TLS encryption and, optionally, authentication using TLS client certificates.

SASL_PLAINTEXT

Listener without encryption but with SASL-based authentication.

SASL_SSL

Listener with TLS-based encryption and SASL-based authentication.

Given the following **listeners** configuration:

```
listeners=INT1://:9092,INT2://:9093,REPLICATION://:9094
```

the **listener.security.protocol.map** might look like this:

```
listener.security.protocol.map=INT1:SASL_PLAINTEXT,INT2:SASL_SSL,REPLICATION:SSL
```

This would configure the listener **INT1** to use unencrypted connections with SASL authentication, the listener **INT2** to use encrypted connections with SASL authentication and the **REPLICATION** interface to use TLS encryption (possibly with TLS client authentication). The same security protocol can be used multiple times. The following example is also a valid configuration:

```
listener.security.protocol.map=INT1:SSL,INT2:SSL,REPLICATION:SSL
```

Such a configuration would use TLS encryption and TLS authentication for all interfaces. The following chapters will explain in more detail how to configure TLS and SASL.

4.8.2. TLS Encryption

In order to use TLS encryption and server authentication, a keystore containing private and public keys has to be provided. This is usually done using a file in the Java Keystore (JKS) format. A path to this file is set in the **ssl.keystore.location** property. The **ssl.keystore.password** property should be used to set the password protecting the keystore. For example:

```
ssl.keystore.location=/path/to/keystore/server-1.jks
ssl.keystore.password=123456
```

In some cases, an additional password is used to protect the private key. Any such password can be set using the **ssl.key.password** property.

Kafka is able to use keys signed by certification authorities as well as self-signed keys. Using keys signed by certification authorities should always be the preferred method. In order to allow clients to verify the identity of the Kafka broker they are connecting to, the certificate should always contain the advertised hostname(s) as its Common Name (CN) or in the Subject Alternative Names (SAN).

It is possible to use different SSL configurations for different listeners. All options starting with **ssl.** can be prefixed with **listener.name.<NameOfTheListener>.**, where the name of the listener has to be always in lower case. This will override the default SSL configuration for that specific listener. The following example shows how to use different SSL configurations for different listeners:

```
listeners=INT1://:9092,INT2://:9093,REPLICATION://:9094
listener.security.protocol.map=INT1:SSL,INT2:SSL,REPLICATION:SSL

# Default configuration - will be used for listeners INT1 and INT2
ssl.keystore.location=/path/to/keystore/server-1.jks
```



```
ssl.keystore.password=123456

# Different configuration for listener REPLICATION
listener.name.replication.ssl.keystore.location=/path/to/keystore/server-1.jks
listener.name.replication.ssl.keystore.password=123456
```

Additional TLS configuration options

In addition to the main TLS configuration options described above, Kafka supports many options for fine-tuning the TLS configuration. For example, to enable or disable TLS / SSL protocols or cipher suites:

`ssl.cipher.suites`

List of enabled cipher suites. Each cipher suite is a combination of authentication, encryption, MAC and key exchange algorithms used for the TLS connection. By default, all available cipher suites are enabled.

`ssl.enabled.protocols`

List of enabled TLS / SSL protocols. Defaults to **TLSv1.2**, **TLSv1.1**, **TLSv1**.

For a complete list of supported Kafka broker configuration options, see [Appendix A, Broker configuration parameters](#).

4.8.3. Enabling TLS encryption

This procedure describes how to enable encryption in Kafka brokers.

Prerequisites

- AMQ Streams is [installed](#) on all hosts which will be used as Kafka brokers.

Procedure

1. Generate TLS certificates for all Kafka brokers in your cluster. The certificates should have their advertised and bootstrap addresses in their Common Name or Subject Alternative Name.
2. Edit the `/opt/kafka/config/server.properties` Kafka configuration file on all cluster nodes for the following:
 - Change the `listener.security.protocol.map` field to specify the **SSL** protocol for the listener where you want to use TLS encryption.
 - Set the `ssl.keystore.location` option to the path to the JKS keystore with the broker certificate.
 - Set the `ssl.keystore.password` option to the password you used to protect the keystore.

For example:

```
listeners=UNENCRYPTED://:9092,ENCRYPTED://:9093,REPLICATION://:9094
listener.security.protocol.map=UNENCRYPTED:PLAINTEXT,ENCRYPTED:SSL,REPLICATION:PLAINTEXT
ssl.keystore.location=/path/to/keystore/server-1.jks
ssl.keystore.password=123456
```

3. (Re)start the Kafka brokers

Additional resources

- For more information about configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).
- For more information about running a Kafka cluster, see [Section 4.5, “Running a multi-node Kafka cluster”](#).
- For more information about configuring TLS encryption in clients, see [Appendix D, *Producer configuration parameters*](#) and [Appendix C, *Consumer configuration parameters*](#).

4.8.4. Authentication

Kafka supports two methods of authentication. On all connections, authentication using one of the supported SASL (Simple Authentication and Security Layer) mechanisms can be used. On encrypted connections, TLS client authentication based on X.509 certificates can be used.

4.8.4.1. TLS client authentication

TLS client authentication can be used only on connections which are already using TLS encryption. To use TLS client authentication, a truststore with public keys can be provided to the broker. These keys can be used to authenticate clients connecting to the broker. The truststore should be provided in Java Keystore (JKS) format and should contain public keys of the certification authorities. All clients with public and private keys signed by one of the certification authorities included in the truststore will be authenticated. The location of the truststore is set using field `ssl.truststore.location`. In case the truststore is password protected, the password should be set in the `ssl.truststore.password` property. For example:

```
ssl.truststore.location=/path/to/keystore/server-1.jks
ssl.truststore.password=123456
```

Once the truststore is configured, TLS client authentication has to be enabled using the `ssl.client.auth` property. This property can be set to one of three different values:

none

TLS client authentication is switched off. (Default value)

requested

TLS client authentication is optional. Clients will be asked to authenticate using TLS client certificate but they can choose not to.

required

Clients are required to authenticate using TLS client certificate.

When a client authenticates using TLS client authentication, the authenticated principal name is the distinguished name from the authenticated client certificate. For example, a user with a certificate which has a distinguished name **CN=someuser** will be authenticated with the following principal **CN=someuser, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown**. When TLS client authentication is not used and SASL is disabled, the principal name will be **ANONYMOUS**.

4.8.4.2. SASL authentication

SASL authentication is configured using Java Authentication and Authorization Service (JAAS). JAAS is also used for authentication of connections between Kafka and Zookeeper. JAAS uses its own configuration file. The recommended location for this file is `/opt/kafka/config/jaas.conf`. The file has to be readable by the `kafka` user. When running Kafka, the location of this file is specified using Java system property `java.security.auth.login.config`. This property has to be passed to Kafka when starting the broker nodes:

```
KAFKA_OPTS="-Djava.security.auth.login.config=/path/to/my/jaas.config";
bin/kafka-server-start.sh
```

SASL authentication is supported both through plain unencrypted connections as well as through TLS connections. SASL can be enabled individually for each listener. To enable it, the security protocol in `listener.security.protocol.map` has to be either `SASL_PLAINTEXT` or `SASL_SSL`.

SASL authentication in Kafka supports several different mechanisms:

PLAIN

Implements authentication based on username and passwords. Usernames and passwords are stored locally in Kafka configuration.

SCRAM-SHA-256 and SCRAM-SHA-512

Implements authentication using Salted Challenge Response Authentication Mechanism (SCRAM). SCRAM credentials are stored centrally in Zookeeper. SCRAM can be used in situations where Zookeeper cluster nodes are running isolated in a private network.

GSSAPI

Implements authentication against a Kerberos server.



WARNING

The **PLAIN** mechanism sends the username and password over the network in an unencrypted format. It should be therefore only be used in combination with TLS encryption.

The SASL mechanisms are configured via the JAAS configuration file. Kafka uses the JAAS context named `KafkaServer`. After they are configured in JAAS, the SASL mechanisms have to be enabled in the Kafka configuration. This is done using the `sasl.enabled.mechanisms` property. This property contains a comma-separated list of enabled mechanisms:

```
sasl.enabled.mechanisms=PLAIN, SCRAM-SHA-256, SCRAM-SHA-512
```

In case the listener used for inter-broker communication is using SASL, the property `sasl.mechanism.inter.broker.protocol` has to be used to specify the SASL mechanism which it should use. For example:

```
sasl.mechanism.inter.broker.protocol=PLAIN
```

The username and password which will be used for the inter-broker communication has to be specified in the `KafkaServer` JAAS context using the field `username` and `password`.

SASL PLAIN

To use the PLAIN mechanism, the usernames and password which are allowed to connect are specified directly in the JAAS context. The following example shows the context configured for SASL PLAIN authentication. The example configures three different users:

- **admin**
- **user1**
- **user2**

```
KafkaServer {
    org.apache.kafka.common.security.plain.PlainLoginModule required
        user_admin="123456"
        user_user1="123456"
        user_user2="123456";
};
```

The JAAS configuration file with the user database should be kept in sync on all Kafka brokers.

When SASL PLAIN is also used for inter-broker authentication, the **username** and **password** properties should be included in the JAAS context:

```
KafkaServer {
    org.apache.kafka.common.security.plain.PlainLoginModule required
        username="admin"
        password="123456"
        user_admin="123456"
        user_user1="123456"
        user_user2="123456";
};
```

SASL SCRAM

SCRAM authentication in Kafka consists of two mechanisms: **SCRAM-SHA-256** and **SCRAM-SHA-512**. These mechanisms differ only in the hashing algorithm used - SHA-256 versus stronger SHA-512. To enable SCRAM authentication, the JAAS configuration file has to include the following configuration:

```
KafkaServer {
    org.apache.kafka.common.security.scram.ScramLoginModule required;
};
```

When enabling SASL authentication in the Kafka configuration file, both SCRAM mechanisms can be listed. However, only one of them can be chosen for the inter-broker communication. For example:

```
sasl.enabled.mechanisms=SCRAM-SHA-256,SCRAM-SHA-512
sasl.mechanism.inter.broker.protocol=SCRAM-SHA-512
```

User credentials for the SCRAM mechanism are stored in Zookeeper. The **kafka-configs.sh** tool can be used to manage them. For example, run the following command to add user **user1** with password **123456**:

```
bin/kafka-configs.sh --zookeeper zoo1.my-domain.com:2181 --alter --add-
config 'SCRAM-SHA-256=[password=123456],SCRAM-SHA-512=[password=123456]' -
-entity-type users --entity-name user1
```

To delete a user credential use:

```
bin/kafka-configs.sh --zookeeper zoo1.my-domain.com:2181 --alter --delete-
config 'SCRAM-SHA-512' --entity-type users --entity-name user1
```

SASL GSSAPI

The SASL mechanism used for authentication using Kerberos is called **GSSAPI**. To configure Kerberos SASL authentication, the following configuration should be added to the JAAS configuration file:

```
KafkaServer {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    keyTab="/etc/security/keytabs/kafka_server.keytab"
    principal="kafka/kafka1.hostname.com@EXAMPLE.COM";
};
```

The domain name in the Kerberos principal has to be always in upper case.

In addition to the JAAS configuration, the Kerberos service name needs to be specified in the **sasl.kerberos.service.name** property in the Kafka configuration:

```
sasl.enabled.mechanisms=GSSAPI
sasl.mechanism.inter.broker.protocol=GSSAPI
sasl.kerberos.service.name=kafka
```

Multiple SASL mechanisms

Kafka can use multiple SASL mechanisms at the same time. The different JAAS configurations can be all added to the same context:

```
KafkaServer {
    org.apache.kafka.common.security.plain.PlainLoginModule required
    user_admin="123456"
    user_user1="123456"
    user_user2="123456";

    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    keyTab="/etc/security/keytabs/kafka_server.keytab"
    principal="kafka/kafka1.hostname.com@EXAMPLE.COM";

    org.apache.kafka.common.security.scram.ScramLoginModule required;
};
```

When multiple mechanisms are enabled, clients will be able to choose the mechanism which they want to use.

4.8.5. Enabling TLS client authentication

This procedure describes how to enable TLS client authentication in Kafka brokers.

Prerequisites

- AMQ Streams is [installed](#) on all hosts which will be used as Kafka brokers.
- TLS encryption is [enabled](#).

Procedure

1. Prepare a JKS truststore containing the public key of the certification authority used to sign the user certificates.
2. Edit the `/opt/kafka/config/server.properties` Kafka configuration file on all cluster nodes for the following:
 - Set the `ssl.truststore.location` option to the path to the JKS truststore with the certification authority of the user certificates.
 - Set the `ssl.truststore.password` option to the password you used to protect the truststore.
 - Set the `ssl.client.auth` option to **required**.
For example:

```
ssl.truststore.location=/path/to/truststore.jks
ssl.truststore.password=123456
ssl.client.auth=required
```

3. (Re)start the Kafka brokers

Additional resources

- For more information about configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).
- For more information about running a Kafka cluster, see [Section 4.5, “Running a multi-node Kafka cluster”](#).
- For more information about configuring TLS client authentication in clients, see [Appendix D, *Producer configuration parameters*](#) and [Appendix C, *Consumer configuration parameters*](#).

4.8.6. Enabling SASL PLAIN authentication

This procedure describes how to enable SASL PLAIN authentication in Kafka brokers.

Prerequisites

- AMQ Streams is [installed](#) on all hosts which will be used as Kafka brokers.

Procedure

1. Edit or create the `/opt/kafka/config/jaas.conf` JAAS configuration file. This file should

contain all your users and their passwords. Make sure this file is the same on all Kafka brokers. For example:

```
KafkaServer {
    org.apache.kafka.common.security.plain.PlainLoginModule required
    user_admin="123456"
    user_user1="123456"
    user_user2="123456";
};
```

2. Edit the `/opt/kafka/config/server.properties` Kafka configuration file on all cluster nodes for the following:

- Change the `listener.security.protocol.map` field to specify the **SASL_PLAINTEXT** or **SASL_SSL** protocol for the listener where you want to use SASL PLAIN authentication.
- Set the `sasl.enabled.mechanisms` option to **PLAIN**.
For example:

```
listeners=INSECURE://:9092,AUTHENTICATED://:9093,REPLICATION://:9094
listener.security.protocol.map=INSECURE:PLAINTEXT,AUTHENTICATED:SASL_PLAINTEXT,REPLICATION:PLAINTEXT
sasl.enabled.mechanisms=PLAIN
```

3. (Re)start the Kafka brokers using the `KAFKA_OPTS` environment variable to pass the JAAS configuration to Kafka brokers.

```
su - kafka
export KAFKA_OPTS="-Djava.security.auth.login.config=/opt/kafka/config/jaas.conf";
/opt/kafka/bin/kafka-server-start.sh -daemon
/opt/kafka/config/server.properties
```

Additional resources

- For more information about configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).
- For more information about running a Kafka cluster, see [Section 4.5, “Running a multi-node Kafka cluster”](#).
- For more information about configuring SASL PLAIN authentication in clients, see [Appendix D, *Producer configuration parameters*](#) and [Appendix C, *Consumer configuration parameters*](#).

4.8.7. Enabling SASL SCRAM authentication

This procedure describes how to enable SASL SCRAM authentication in Kafka brokers.

Prerequisites

- AMQ Streams is [installed](#) on all hosts which will be used as Kafka brokers.

Procedure

1. Edit or create the `/opt/kafka/config/jaas.conf` JAAS configuration file. Enable the **ScramLoginModule** for the **KafkaServer** context. Make sure this file is the same on all Kafka brokers.

For example:

```
KafkaServer {
    org.apache.kafka.common.security.scram.ScramLoginModule
    required;
};
```

2. Edit the `/opt/kafka/config/server.properties` Kafka configuration file on all cluster nodes for the following:

- Change the `listener.security.protocol.map` field to specify the **SASL_PLAINTEXT** or **SASL_SSL** protocol for the listener where you want to use SASL SCRAM authentication.
- Set the `sasl.enabled.mechanisms` option to **SCRAM-SHA-256** or **SCRAM-SHA-512**.

For example:

```
listeners=INSECURE://:9092,AUTHENTICATED://:9093,REPLICATION://:9094
listener.security.protocol.map=INSECURE:PLAINTEXT,AUTHENTICATED:SASL_PLAINTEXT,REPLICATION:PLAINTEXT
sasl.enabled.mechanisms=SCRAM-SHA-512
```

3. (Re)start the Kafka brokers using the `KAFKA_OPTS` environment variable to pass the JAAS configuration to Kafka brokers.

```
su - kafka
export KAFKA_OPTS="-Djava.security.auth.login.config=/opt/kafka/config/jaas.conf";
/opt/kafka/bin/kafka-server-start.sh -daemon
/opt/kafka/config/server.properties
```

Additional resources

- For more information about configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).
- For more information about running a Kafka cluster, see [Section 4.5, “Running a multi-node Kafka cluster”](#).
- For more information about adding SASL SCRAM users, see [Section 4.8.8, “Adding SASL SCRAM users”](#).
- For more information about deleting SASL SCRAM users, see [Section 4.8.9, “Deleting SASL SCRAM users”](#).
- For more information about configuring SASL SCRAM authentication in clients, see [Appendix D, *Producer configuration parameters*](#) and [Appendix C, *Consumer configuration parameters*](#).

4.8.8. Adding SASL SCRAM users

This procedure describes how to add new users for authentication using SASL SCRAM.

Prerequisites

- AMQ Streams is [installed](#) on all hosts which will be used as Kafka brokers.
- SASL SCRAM authentication is [enabled](#).

Procedure

- Use the **kafka-configs.sh** tool to add new SASL SCRAM users.

```
bin/kafka-configs.sh --zookeeper <ZookeeperAddress> --alter --add-
config 'SCRAM-SHA-512=[password=<Password>]' --entity-type users --
entity-name <Username>
```

For example:

```
bin/kafka-configs.sh --zookeeper zoo1.my-domain.com:2181 --alter --
add-config 'SCRAM-SHA-512=[password=123456]' --entity-type users --
entity-name user1
```

Additional resources

- For more information about configuring SASL SCRAM authentication in clients, see [Appendix D, *Producer configuration parameters*](#) and [Appendix C, *Consumer configuration parameters*](#).

4.8.9. Deleting SASL SCRAM users

This procedure describes how to remove users when using SASL SCRAM authentication.

Prerequisites

- AMQ Streams is [installed](#) on all hosts which will be used as Kafka brokers.
- SASL SCRAM authentication is [enabled](#).

Procedure

- Use the **kafka-configs.sh** tool to delete SASL SCRAM users.

```
bin/kafka-configs.sh --zookeeper <ZookeeperAddress> --alter --
delete-config 'SCRAM-SHA-512' --entity-type users --entity-name
<Username>
```

For example:

```
bin/kafka-configs.sh --zookeeper zoo1.my-domain.com:2181 --alter --
delete-config 'SCRAM-SHA-512' --entity-type users --entity-name
user1
```

Additional resources

- For more information about configuring SASL SCRAM authentication in clients, see [Appendix D, *Producer configuration parameters*](#) and [Appendix C, *Consumer configuration parameters*](#).

4.9. LOGGING

Kafka brokers use Log4j as their logging infrastructure. Logging configuration is by default read from the **log4j.properties** configuration file which should be placed either in the **/opt/kafka/config/** directory or on the classpath. The location and name of the configuration file can be changed using the Java property **log4j.configuration** which can be passed to Kafka using the **KAFKA_LOG4J_OPTS** environment variable:

```
su - kafka
export KAFKA_LOG4J_OPTS="-
Dlog4j.configuration=file:/my/path/to/log4j.config"; /opt/kafka/bin/kafka-
server-start.sh /opt/kafka/config/server.properties
```

For more information about Log4j configurations, see [Log4j manual](#).

CHAPTER 5. TOPICS

Messages in Kafka are always sent to or received from a topic. This chapter describes how to configure and manage Kafka topics.

5.1. PARTITIONS AND REPLICAS

Messages in Kafka are always sent to or received from a topic. A topic is always split into one or more partitions. Partitions act as shards. That means that every message sent by a producer is always written only into a single partition. Thanks to the sharding of messages into different partitions, topics are easy to scale horizontally.

Each partition can have one or more replicas, which will be stored on different brokers in the cluster. When creating a topic you can configure the number of replicas using the *replication factor*. *Replication factor* defines the number of copies which will be held within the cluster. One of the replicas for given partition will be elected as a leader. The leader replica will be used by the producers to send new messages and by the consumers to consume messages. The other replicas will be follower replicas. The followers replicate the leader.

If the leader fails, one of the followers will automatically become the new leader. Each server acts as a leader for some of its partitions and a follower for others so the load is well balanced within the cluster.

NOTE: The replication factor determines the number of replicas including the leader and the followers. For example, if you set the replication factor to **3**, then there will one leader and two follower replicas.

5.2. MESSAGE RETENTION

The message retention policy defines how long the messages will be stored on the Kafka brokers. It can be defined based on time, partition size or both.

For example, you can define that the messages should be kept:

- For 7 days
- Until the partition has 1GB of messages. Once the limit is reached, the oldest messages will be removed.
- For 7 days or until the 1GB limit has been reached. Whatever limit comes first will be used.



WARNING

Kafka brokers store messages in log segments. The messages which are past their retention policy will be deleted only when a new log segment is created. New log segments are created when the previous log segment exceeds the configured log segment size. Additionally, users can request new segments to be created periodically.

Additionally, Kafka brokers support a compacting policy.

For a topic with the compacted policy, the broker will always keep only the last message for each key.

The older messages with the same key will be removed from the partition. Because compacting is a periodically executed action, it does not happen immediately when the new message with the same key are sent to the partition. Instead it might take some time until the older messages are removed.

For more information about the message retention configuration options, see [Section 5.5, “Topic configuration”](#).

5.3. TOPIC AUTO-CREATION

When a producer or consumer tries to sent to or received from from a topic which does not exist, Kafka will, by default, automatically create that topic. This behavior is controlled by the **auto.create.topics.enable** configuration property which is set to **true** by default.

To disable it, set **auto.create.topics.enable** to **false** in the Kafka broker configuration file:

```
auto.create.topics.enable=false
```

5.4. TOPIC DELETION

Kafka offers the possibility to disable deletion of topics. This is configured through the **delete.topic.enable** property, which is set to **true** by default (that is, deleting topics is possible). When this property is set to **false** it will be not possible to delete topics and all attempts to delete topic will return success but the topic will not be deleted.

```
delete.topic.enable=false
```

5.5. TOPIC CONFIGURATION

Auto-created topics will use the default topic configuration which can be specified in the broker properties file. However, when creating topics manually, their configuration can be specified at creation time. It is also possible to change a topic’s configuration after it has been created. The main topic configuration options for manually created topics are:

cleanup.policy

Configures the retention policy to **delete** or **compact**. The **delete** policy will delete old records. The **compact** policy will enable log compaction. The default value is **delete**. For more information about log compaction, see [Kafka website](#).

compression.type

Specifies the compression which is used for stored messages. Valid values are **gzip**, **snappy**, **lz4**, **uncompressed** (no compression) and **producer** (retain the compression codec used by the producer). The default value is **producer**.

max.message.bytes

The maximum size of a batch of messages allowed by the Kafka broker, in bytes. The default value is **1000012**.

min.insync.replicas

The minimum number of replicas which must be in sync for a write to be considered successful. The default value is **1**.

retention.ms

Maximum number of milliseconds for which log segments will be retained. Log segments older than this value will be deleted. The default value is **604800000** (7 days).

retention.bytes

The maximum number of bytes a partition will retain. Once the partition size grows over this limit, the oldest log segments will be deleted. Value of **-1** indicates no limit. The default value is **-1**.

segment.bytes

The maximum file size of a single commit log segment file in bytes. When the segment reaches its size, a new segment will be started. The default value is **1073741824** bytes (1 gibibyte).

For list of all supported topic configuration options, see [Appendix B, Topic configuration parameters](#).

The defaults for auto-created topics can be specified in the Kafka broker configuration using similar options:

log.cleanup.policy

See **cleanup.policy** above.

compression.type

See **compression.type** above.

message.max.bytes

See **max.message.bytes** above.

min.insync.replicas

See **min.insync.replicas** above.

log.retention.ms

See **retention.ms** above.

log.retention.bytes

See **retention.bytes** above.

log.segment.bytes

See **segment.bytes** above.

default.replication.factor

Default replication factor for automatically created topics. Default value is **1**.

num.partitions

Default number of partitions for automatically created topics. Default value is **1**.

For list of all supported Kafka broker configuration options, see [Appendix A, Broker configuration parameters](#).

5.6. INTERNAL TOPICS

Internal topics are created and used internally by the Kafka brokers and clients. Kafka has several internal topics. These are used to store consumer offsets (**__consumer_offsets**) or transaction state (**__transaction_state**). These topics can be configured using dedicated Kafka broker configuration options starting with prefix **offsets.topic.** and **transaction.state.log..** The most important configuration options are:

offsets.topic.replication.factor

Number of replicas for **__consumer_offsets** topic. The default value is **3**.

offsets.topic.num.partitions

Number of partitions for `__consumer_offsets` topic. The default value is **50**.

transaction.state.log.replication.factor

Number of replicas for `__transaction_state` topic. The default value is **3**.

transaction.state.log.num.partitions

Number of partitions for `__transaction_state` topic. The default value is **50**.

transaction.state.log.min.isr

Minimum number of replicas that must acknowledge a write to `__transaction_state` topic to be considered successful. If this minimum cannot be met, then the producer will fail with an exception. The default value is **2**.

5.7. CREATING A TOPIC

The **kafka-topics.sh** tool can be used to manage topics. **kafka-topics.sh** is part of the AMQ Streams distribution and can be found in the **bin** directory.

Prerequisites

- AMQ Streams cluster is installed and running

Creating a topic

1. Create a topic using the **kafka-topics.sh** utility and specify the following: Zookeeper URL in the `--zookeeper` option. The new topic to be created in the `--create` option. Topic name in the `--topic` option. The number of partitions in the `--partitions` option. Replication factor in the `--replication-factor` option.

You can also override some of the default topic configuration options using the option `--config`. This option can be used multiple times to override different options.

```
bin/kafka-topics.sh --zookeeper <ZookeeperAddress> --create --topic
<TopicName> --partitions <NumberOfPartitions> --replication-factor
<ReplicationFactor> --config <Option1>=<Value1> --config
<Option2>=<Value2>
```

Example of the command to create a topic named mytopic

```
bin/kafka-topics.sh --zookeeper zoo1.my-domain.com:2181 --create --
topic mytopic --partitions 50 --replication-factor 3 --config
cleanup.policy=compact --config min.insync.replicas=2
```

2. Verify that the topic exists using **kafka-topics.sh**.

```
bin/kafka-topics.sh --zookeeper <ZookeeperAddress> --describe --
topic <TopicName>
```

Example of the command to describe a topic named mytopic

```
bin/kafka-topics.sh --zookeeper zoo1.my-domain.com:2181 --describe -
-topic mytopic
```

Additional resources

- For more information about topic configuration, see [Section 5.5, “Topic configuration”](#).
- For list of all supported topic configuration options, see [Appendix B, *Topic configuration parameters*](#).

5.8. LISTING AND DESCRIBING TOPICS

The **kafka-topics.sh** tool can be used to list and describe topics. **kafka-topics.sh** is part of the AMQ Streams distribution and can be found in the **bin** directory.

Prerequisites

- AMQ Streams cluster is installed and running
- Topic **mytopic** exists

Describing a topic

1. Describe a topic using the **kafka-topics.sh** utility.
 - Specify the Zookeeper URL in the **--zookeeper** option.
 - Use **--describe** option to specify that you want to describe a topic.
 - Topic name has to be specified in the **--topic** option.
 - When the **--topic** option is omitted, it will describe all available topics.

```
bin/kafka-topics.sh --zookeeper <ZookeeperAddress> --describe --
topic <TopicName>
```

Example of the command to describe a topic named mytopic

```
bin/kafka-topics.sh --zookeeper zoo1.my-domain.com:2181 --
describe --topic mytopic
```

The describe command will list all partitions and replicas which belong to this topic. It will also list all topic configuration options.

Additional resources

- For more information about topic configuration, see [Section 5.5, “Topic configuration”](#).
- For more information about creating topics, see [Section 5.7, “Creating a topic”](#).

5.9. MODIFYING A TOPIC CONFIGURATION

The **kafka-configs.sh** tool can be used to modify topic configurations. **kafka-configs.sh** is part of the AMQ Streams distribution and can be found in the **bin** directory.

Prerequisites

- AMQ Streams cluster is installed and running
- Topic **mytopic** exists

Modify topic configuration

1. Use the **kafka-configs.sh** tool to get the current configuration.

- Specify the Zookeeper URL in the **--zookeeper** option.
- Set the **--entity-type** as **topic** and **--entity-name** to the name of your topic.
- Use **--describe** option to get the current configuration.

```
bin/kafka-configs.sh --zookeeper <ZookeeperAddress> --entity-type
topics --entity-name <TopicName> --describe
```

Example of the command to get configuration of a topic named **mytopic**

```
bin/kafka-configs.sh --zookeeper zoo1.my-domain.com:2181 --
entity-type topics --entity-name mytopic --describe
```

2. Use the **kafka-configs.sh** tool to change the configuration.

- Specify the Zookeeper URL in the **--zookeeper** options.
- Set the **--entity-type** as **topic** and **--entity-name** to the name of your topic.
- Use **--alter** option to modify the current configuration.
- Specify the options you want to add or change in the option **--add-config**.

```
bin/kafka-configs.sh --zookeeper <ZookeeperAddress> --entity-type
topics --entity-name <TopicName> --alter --add-config
<Option>=<Value>
```

Example of the command to change configuration of a topic named **mytopic**

```
bin/kafka-configs.sh --zookeeper zoo1.my-domain.com:2181 --
entity-type topics --entity-name mytopic --alter --add-config
min.insync.replicas=1
```

3. Use the **kafka-configs.sh** tool to delete an existing configuration option.

- Specify the Zookeeper URL in the **--zookeeper** options.
- Set the **--entity-type** as **topic** and **--entity-name** to the name of your topic.
- Use **--delete-config** option to remove existing configuration option.
- Specify the options you want to remove in the option **--remove-config**.

```
bin/kafka-configs.sh --zookeeper <ZookeeperAddress> --entity-type
topics --entity-name <TopicName> --alter --delete-config <Option>
```


Example of the command to change configuration of a topic named `mytopic`

```
bin/kafka-configs.sh --zookeeper zoo1.my-domain.com:2181 --
entity-type topics --entity-name mytopic --alter --delete-config
min.insync.replicas
```

Additional resources

- For more information about topic configuration, see [Section 5.5, “Topic configuration”](#).
- For more information about creating topics, see [Section 5.7, “Creating a topic”](#).
- For list of all supported topic configuration options, see [Appendix B, *Topic configuration parameters*](#).

5.10. DELETING A TOPIC

The `kafka-topics.sh` tool can be used to manage topics. `kafka-topics.sh` is part of the AMQ Streams distribution and can be found in the `bin` directory.

Prerequisites

- AMQ Streams cluster is installed and running
- Topic `mytopic` exists

Deleting a topic

1. Delete a topic using the `kafka-topics.sh` utility.
 - Specify the Zookeeper URL in the `--zookeeper` option.
 - Use `--delete` option to specify that an existing topic should be deleted.
 - Topic name has to be specified in the `--topic` option.

```
bin/kafka-topics.sh --zookeeper <ZookeeperAddress> --delete --
topic <TopicName>
```

Example of the command to create a topic named `mytopic`

```
bin/kafka-topics.sh --zookeeper zoo1.my-domain.com:2181 --delete
--topic mytopic
```

2. Verify that the topic was deleted using `kafka-topics.sh`.

```
bin/kafka-topics.sh --zookeeper <ZookeeperAddress> --list
```

Example of the command to list all topics

```
bin/kafka-topics.sh --zookeeper zoo1.my-domain.com:2181 --list
```

Additional resources

- For more information about creating topics, see [Section 5.7, “Creating a topic”](#).

CHAPTER 6. SCALING CLUSTERS

6.1. SCALING KAFKA CLUSTERS

6.1.1. Adding brokers to a cluster

The primary way of increasing throughput for a topic is to increase the number of partitions for that topic. That works because the partitions allow the load for that topic to be shared between the brokers in the cluster. When the brokers are all constrained by some resource (typically I/O), then using more partitions will not yield an increase in throughput. Instead, you must add brokers to the cluster.

When you add an extra broker to the cluster, AMQ Streams does not assign any partitions to it automatically. You have to decide which partitions to move from the existing brokers to the new broker.

Once the partitions have been redistributed between all brokers, each broker should have a lower resource utilization.

6.1.2. Removing brokers from the cluster

Before you remove a broker from a cluster, you must ensure that it is not assigned to any partitions. You should decide which remaining brokers will be responsible for each of the partitions on the broker being decommissioned. Once the broker has no assigned partitions, you can stop it.

6.2. REASSIGNMENT OF PARTITIONS

The `kafka-reassign-partitions.sh` utility is used to reassign partitions to different brokers.

It has three different modes:

--generate

Takes a set of topics and brokers and generates a *reassignment JSON file* which will result in the partitions of those topics being assigned to those brokers. It is an easy way to generate a *reassignment JSON file*, but it operates on whole topics, so its use is not always appropriate.

--execute

Takes a *reassignment JSON file* and applies it to the partitions and brokers in the cluster. Brokers which are gaining partitions will become followers of the partition leader. For a given partition, once the new broker has caught up and joined the ISR the old broker will stop being a follower and will delete its replica.

--verify

Using the same *reassignment JSON file* as the **--execute** step, **--verify** checks whether all of the partitions in the file have been moved to their intended brokers. If the reassignment is complete it will also remove any [throttles](#) which are in effect. Unless removed, throttles will continue to affect the cluster even after the reassignment has finished.

It is only possible to have one reassignment running in the cluster at any given time, and it is not possible to cancel a running reassignment. If you need to cancel a reassignment you have to wait for it to complete and then perform another reassignment to revert the effects of the first one. The `kafka-reassign-partitions.sh` will print the reassignment JSON for this reversion as part of its output. Very large reassignments should be broken down into a number of smaller reassignments in case there is a need to stop in-progress reassignment.

6.2.1. Reassignment JSON file

The *reassignment JSON file* has a specific structure:

```
{
  "version": 1,
  "partitions": [
    <PartitionObjects>
  ]
}
```

Where *<PartitionObjects>* is a comma-separated list of objects like:

```
{
  "topic": <TopicName>,
  "partition": <Partition>,
  "replicas": [ <AssignedBrokerIds> ],
  "log_dirs": [<LogDirs>]
}
```

The **"log_dirs"** property is optional and is used to move the partition to a specific log directory.

The following is an example reassignment JSON file that assigns topic **topic-a**, partition **4** to brokers **2, 4** and **7**, and topic **topic-b** partition **2** to brokers **1, 5** and **7**:

```
{
  "version": 1,
  "partitions": [
    {
      "topic": "topic-a",
      "partition": 4,
      "replicas": [2,4,7]
    },
    {
      "topic": "topic-b",
      "partition": 2,
      "replicas": [1,5,7]
    }
  ]
}
```

Partitions not included in the JSON are not changed.

6.2.2. Generating reassignment JSON files

The easiest way to assign all the partitions for a given set of topics to a given set of brokers is to generate a reassignment JSON file using `kafka-reassign-partitions.sh --generate`, command.

```
bin/kafka-reassign-partitions.sh --zookeeper <Zookeeper> --topics-to-move-
json-file <TopicsFile> --broker-list <BrokerList> --generate
```

The **<TopicsFile>** is a JSON file which lists the topics to move. It has the following structure:

```
{
```

```
{
  "version": 1,
  "topics": [
    <TopicObjects>
  ]
}
```

where *<TopicObjects>* is a comma-separated list of objects like:

```
{
  "topic": <TopicName>
}
```

For example to move all the partitions of **topic-a** and **topic-b** to brokers **4** and **7**

```
bin/kafka-reassign-partitions.sh --zookeeper localhost:2181 --topics-to-move-json-file topics-to-be-moved.json --broker-list 4,7 --generate
```

where **topics-to-be-moved.json** has contents:

```
{
  "version": 1,
  "topics": [
    { "topic": "topic-a"},
    { "topic": "topic-b"}
  ]
}
```

6.2.3. Creating reassignment JSON files manually

You can manually create the reassignment JSON file if you want to move specific partitions.

6.3. REASSIGNMENT THROTTLES

Reassigning partitions can be a slow process because it can require moving lots of data between brokers. To avoid this having a detrimental impact on clients it is possible to *throttle* the reassignment. Using a throttle can mean the reassignment takes longer. If the throttle is too low then the newly assigned brokers will not be able to keep up with records being published and the reassignment will never complete. If the throttle is too high then clients will be impacted. For example, for producers, this could manifest as higher than normal latency waiting for acknowledgement. For consumers, this could manifest as a drop in throughput caused by higher latency between polls.

6.4. SCALING UP A KAFKA CLUSTER

This procedure describes how to increase the number of brokers in a Kafka cluster.

Prerequisites

- An existing Kafka cluster.
- A new machine with the AMQ broker [installed](#).
- A *reassignment JSON file* of how partitions should be reassigned to brokers in the enlarged cluster.

Procedure

1. Create a configuration file for the new broker using the same settings as for the other brokers in your cluster, except for **broker.id** which should be a number that is not already used by any of the other brokers.
2. Start the new Kafka broker passing the configuration file you created in the previous step as the argument to the **kafka-server-start.sh** script:

```
su - kafka
/opt/kafka/bin/kafka-server-start.sh -daemon
/opt/kafka/config/server.properties
```

3. Verify that the Kafka broker is running.

```
jcmd | grep Kafka
```

4. Repeat the above steps for each new broker.
5. Execute the partition reassignment using the **kafka-reassign-partitions.sh** command line tool.

```
kafka-reassign-partitions.sh --zookeeper <ZookeeperHostAndPort> --
reassignment-json-file <ReassignmentJsonFile> --execute
```

If you are going to throttle replication you can also pass the **--throttle** option with an inter-broker throttled rate in bytes per second. For example:

```
kafka-reassign-partitions.sh --zookeeper zookeeper1:2181 --
reassignment-json-file reassignment.json --throttle 5000000 --
execute
```

This command will print out two reassignment JSON objects. The first records the current assignment for the partitions being moved. You should save this to a file in case you need to revert the reassignment later on. The second JSON object is the target reassignment you have passed in your reassignment JSON file.

6. If you need to change the throttle during reassignment you can use the same command line with a different throttled rate. For example:

```
kafka-reassign-partitions.sh --zookeeper zookeeper1:2181 --
reassignment-json-file reassignment.json --throttle 10000000 --
execute
```

7. Periodically verify whether the reassignment has completed using the **kafka-reassign-partitions.sh** command line tool. This is the same command as the previous step but with the **--verify** option instead of the **--execute** option.

```
kafka-reassign-partitions.sh --zookeeper <ZookeeperHostAndPort> --
reassignment-json-file <ReassignmentJsonFile> --verify
```

For example:

```
kafka-reassign-partitions.sh --zookeeper zookeeper1:2181 --
```

```
reassignment-json-file reassignment.json --verify
```

8. The reassignment has finished when the **--verify** command reports each of the partitions being moved as completed successfully. This final **--verify** will also have the effect of removing any reassignment throttles. You can now delete the revert file if you saved the JSON for reverting the assignment to their original brokers.

6.5. SCALING DOWN A KAFKA CLUSTER

Additional resources

This procedure describes how to decrease the number of brokers in a Kafka cluster.

Prerequisites

- An existing Kafka cluster.
- A *reassignment JSON file* of how partitions should be reassigned to brokers in the cluster once the broker(s) have been removed.

Procedure

1. Execute the partition reassignment using the **kafka-reassign-partitions.sh** command line tool.

```
kafka-reassign-partitions.sh --zookeeper <ZookeeperHostAndPort> --
reassignment-json-file <ReassignmentJsonFile> --execute
```

If you are going to throttle replication you can also pass the **--throttle** option with an inter-broker throttled rate in bytes per second. For example:

```
kafka-reassign-partitions.sh --zookeeper zookeeper1:2181 --
reassignment-json-file reassignment.json --throttle 5000000 --
execute
```

This command will print out two reassignment JSON objects. The first records the current assignment for the partitions being moved. You should save this to a file in case you need to revert the reassignment later on. The second JSON object is the target reassignment you have passed in your reassignment JSON file.

2. If you need to change the throttle during reassignment you can use the same command line with a different throttled rate. For example:

```
kafka-reassign-partitions.sh --zookeeper zookeeper1:2181 --
reassignment-json-file reassignment.json --throttle 10000000 --
execute
```

3. Periodically verify whether the reassignment has completed using the **kafka-reassign-partitions.sh** command line tool. This is the same command as the previous step but with the **--verify** option instead of the **--execute** option.

```
kafka-reassign-partitions.sh --zookeeper <ZookeeperHostAndPort> --
reassignment-json-file <ReassignmentJsonFile> --verify
```

For example:

```
kafka-reassign-partitions.sh --zookeeper zookeeper1:2181 --
reassignment-json-file reassignment.json --verify
```

4. The reassignment has finished when the **--verify** command reports each of the partitions being moved as completed successfully. This final **--verify** will also have the effect of removing any reassignment throttles. You can now delete the revert file if you saved the JSON for reverting the assignment to their original brokers.
5. Once all the partition reassignments have finished, the broker being removed should not have responsibility for any of the partitions in the cluster. You can verify this by checking each of the directories given in the broker's **log.dirs** configuration parameters. If any of the log directories on the broker contains a directory that does not match the extended regular expression **\.[a-z0-9]-delete\$** then the broker still has live partitions and it should not be stopped. You can check this by executing the command:

```
ls -l <LogDir> | grep -E '^d' | grep -vE '[a-zA-Z0-9.-]+\.[a-z0-9]+-
delete$'
```

If the above command prints any output then the broker still has live partitions. In this case, either the reassignment has not finished, or the reassignment JSON file was incorrect.

6. Once you have confirmed that the broker has no live partitions you can stop it.

```
su - kafka
/opt/kafka/bin/kafka-server-stop.sh
```

7. Confirm that the Kafka broker is stopped.

```
jcmd | grep kafka
```


CHAPTER 7. KAFKA CONNECT

Additional resources

Kafka Connect is a tool for streaming data between Apache Kafka and external systems. It provides a framework for moving large amounts of data while maintaining scalability and reliability. Kafka Connect is typically used to integrate Kafka with database, storage, and messaging systems that are external to your Kafka cluster.

Kafka Connect uses connector plug-ins that implement connectivity for different types of external systems. There are two types of connector plug-ins: sink and source. Sink connectors stream data from Kafka to external systems. Source connectors stream data from external systems into Kafka.

Kafka Connect can run in standalone or distributed modes.

Standalone mode

In standalone mode, Kafka Connect runs on a single node with user-defined configuration read from a properties file.

Distributed mode

In distributed mode, Kafka Connect runs across one or more worker nodes and the workloads are distributed among them. You manage connectors and their configuration using an HTTP REST interface.

7.1. KAFKA CONNECT IN STANDALONE MODE

In standalone mode, Kafka Connect runs as a single process, on a single node. You manage the configuration of standalone mode using properties files.

7.1.1. Configuring Kafka Connect in standalone mode

To configure Kafka Connect in standalone mode, edit the **config/connect-standalone.properties** configuration file. The following options are the most important.

bootstrap.servers

A list of Kafka broker addresses used as bootstrap connections to Kafka. For example, **kafka0.my-domain.com:9092,kafka1.my-domain.com:9092,kafka2.my-domain.com:9092**.

key.converter

The class used to convert message keys to and from Kafka format. For example, **org.apache.kafka.connect.json.JsonConverter**.

value.converter

The class used to convert message payloads to and from Kafka format. For example, **org.apache.kafka.connect.json.JsonConverter**.

offset.storage.file.filename

Specifies the file in which the offset data is stored.

An example configuration file is provided in the installation directory at **config/connect-standalone.properties**. For a complete list of all supported Kafka Connect configuration options, see [kafka-connect-configuration-parameters-str].

Connector plug-ins open client connections to the Kafka brokers using the bootstrap address. To configure these connections, use the standard Kafka producer and consumer configuration options prefixed by **producer.** or **consumer.**

For more information on configuring Kafka producers and consumers, see [Appendix D, *Producer configuration parameters*](#) and [Appendix C, *Consumer configuration parameters*](#).

7.1.2. Configuring connectors in Kafka Connect in standalone mode

You can configure connector plug-ins for Kafka Connect in standalone mode using properties files. Most configuration options are specific to each connector. The following options apply to all connectors:

name

The name of the connector, which must be unique within the current Kafka Connect instance.

connector.class

The class of the connector plug-in. For example,
org.apache.kafka.connect.file.FileStreamSinkConnector.

tasks.max

The maximum number of tasks that the specified connector can use. Tasks enable the connector to perform work in parallel. The connector might create fewer tasks than specified.

key.converter

The class used to convert message keys to and from Kafka format. This overrides the default value set by the Kafka Connect configuration. For example,
org.apache.kafka.connect.json.JsonConverter.

value.converter

The class used to convert message payloads to and from Kafka format. This overrides the default value set by the Kafka Connect configuration. For example,
org.apache.kafka.connect.json.JsonConverter.

Additionally, you must set at least one of the following options for sink connectors:

topics

A comma-separated list of topics used as input.

topics.regex

A Java regular expression of topics used as input.

For all other options, see the documentation for the available connectors.

AMQ Streams includes example connector configuration files – see **config/connect-file-sink.properties** and **config/connect-file-source.properties** in the AMQ Streams installation directory.

7.1.3. Running Kafka Connect in standalone mode

This procedure describes how to configure and run Kafka Connect in standalone mode.

Prerequisites

- An installed and running AMQ Streams} cluster.

Procedure

1. Edit the `/opt/kafka/config/connect-standalone.properties` Kafka Connect configuration file and set **`bootstrap.server`** to point to your Kafka brokers. For example:

```
bootstrap.servers=kafka0.my-domain.com:9092,kafka1.my-  
domain.com:9092,kafka2.my-domain.com:9092
```

2. Start Kafka Connect with the configuration file and specify one or more connector configurations.

```
su - kafka  
/opt/kafka/bin/connect-standalone.sh /opt/kafka/config/connect-  
standalone.properties connector1.properties  
[connector2.properties ...]
```

3. Verify that Kafka Connect is running.

```
jcmd | grep ConnectStandalone
```

Additional resources

- For more information on installing AMQ Streams, see [Section 2.3, “Installing AMQ Streams”](#).
- For more information on configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).
- For a complete list of supported Kafka Connect configuration options, see [Appendix F, *Kafka Connect configuration parameters*](#).

7.2. KAFKA CONNECT IN DISTRIBUTED MODE

In distributed mode, Kafka Connect runs across one or more worker nodes and the workloads are distributed among them. You manage connector plug-ins and their configuration using the HTTP REST interface.

7.2.1. Configuring Kafka Connect in distributed mode

To configure Kafka Connect in distributed mode, edit the **`config/connect-distributed.properties`** configuration file. The following options are the most important.

`bootstrap.servers`

A list of Kafka broker addresses used as bootstrap connections to Kafka. For example, **`kafka0.my-domain.com:9092,kafka1.my-domain.com:9092,kafka2.my-domain.com:9092`**.

`key.converter`

The class used to convert message keys to and from Kafka format. For example, **`org.apache.kafka.connect.json.JsonConverter`**.

`value.converter`

The class used to convert message payloads to and from Kafka format. For example, **`org.apache.kafka.connect.json.JsonConverter`**.

`group.id`

The name of the distributed Kafka Connect cluster. This must be unique and must not conflict with another consumer group ID. The default value is **connect-cluster**.

config.storage.topic

The Kafka topic used to store connector configurations. The default value is **connect-configs**.

offset.storage.topic

The Kafka topic used to store offsets. The default value is **connect-offset**.

status.storage.topic

The Kafka topic used for worker node statuses. The default value is **connect-status**.

AMQ Streams includes an example configuration file for Kafka Connect in distributed mode – see **config/connect-distributed.properties** in the AMQ Streams installation directory.

For a complete list of all supported Kafka Connect configuration options, see [Appendix F, Kafka Connect configuration parameters](#).

Connector plug-ins open client connections to the Kafka brokers using the bootstrap address. To configure these connections, use the standard Kafka producer and consumer configuration options prefixed by **producer.** or **consumer.**

For more information on configuring Kafka producers and consumers, see [Appendix D, Producer configuration parameters](#) and [Appendix C, Consumer configuration parameters](#).

7.2.2. Configuring connectors in distributed Kafka Connect

HTTP REST Interface

Connectors for distributed Kafka Connect are configured using HTTP REST interface. The REST interface listens on port 8083 by default. It supports following endpoints:

GET /connectors

Return a list of existing connectors.

POST /connectors

Create a connector. The request body has to be a JSON object with the connector configuration.

GET /connectors/<name>

Get information about a specific connector.

GET /connectors/<name>/config

Get configuration of a specific connector.

PUT /connectors/<name>/config

Update the configuration of a specific connector.

GET /connectors/<name>/status

Get the status of a specific connector.

PUT /connectors/<name>/pause

Pause the connector and all its tasks. The connector will stop processing any messages.

PUT /connectors/<name>/resume

Resume a paused connector.

POST /connectors/<name>/restart

Restart a connector in case it has failed.

DELETE /connectors/<name>

Delete a connector.

GET /connector-plugins

Get a list of all supported connector plugins.

Connector configuration

Most configuration options are connector specific and included in the documentation for the connectors. The following fields are common for all connectors.

name

Name of the connector. Must be unique within a given Kafka Connect instance.

connector.class

Class of the connector plugin. For example

org.apache.kafka.connect.file.FileStreamSinkConnector.

tasks.max

The maximum number of tasks used by this connector. Tasks are used by the connector to parallelise its work. Connectors may create fewer tasks than specified.

key.converter

Class used to convert message keys to and from Kafka format. This overrides the default value set by the Kafka Connect configuration. For example,

org.apache.kafka.connect.json.JsonConverter.

value.converter

Class used to convert message payloads to and from Kafka format. This overrides the default value set by the Kafka Connect configuration. For example,

org.apache.kafka.connect.json.JsonConverter.

Additionally, one of the following options must be set for sink connectors:

topics

A comma-separated list of topics used as input.

topics.regex

A Java regular expression of topics used as input.

For all other options, see the documentation for the specific connector.

AMQ Streams includes example connector configuration files. They can be found in **config/connect-file-sink.properties** and **config/connect-file-source.properties** in the AMQ Streams installation directory.

7.2.3. Running distributed Kafka Connect

This procedure describes how to configure and run Kafka Connect in distributed mode.

Prerequisites

- An installed and running AMQ Streams cluster.

Running the cluster

1. Edit the `/opt/kafka/config/connect-distributed.properties` Kafka Connect configuration file on all Kafka Connect worker nodes.

- Set the `bootstrap.server` option to point to your Kafka brokers.
- Set the `group.id` option.
- Set the `config.storage.topic` option.
- Set the `offset.storage.topic` option.
- Set the `status.storage.topic` option.
For example:

```
bootstrap.servers=kafka0.my-domain.com:9092,kafka1.my-
domain.com:9092,kafka2.my-domain.com:9092
group.id=my-group-id
config.storage.topic=my-group-id-configs
offset.storage.topic=my-group-id-offsets
status.storage.topic=my-group-id-status
```

2. Start the Kafka Connect workers with the `/opt/kafka/config/connect-distributed.properties` configuration file on all Kafka Connect nodes.

```
su - kafka
/opt/kafka/bin/connect-distributed.sh /opt/kafka/config/connect-
distributed.properties
```

3. Verify that Kafka Connect is running.

```
jcmd | grep ConnectDistributed
```

Additional resources

- For more information about installing AMQ Streams, see [Section 2.3, “Installing AMQ Streams”](#).
- For more information about configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).
- For a complete list of supported Kafka Connect configuration options, see [Appendix F, *Kafka Connect configuration parameters*](#).

7.2.4. Creating connectors

This procedure describes how to use the Kafka Connect REST API to create a connector plug-in for use with Kafka Connect in distributed mode.

Prerequisites

- A Kafka Connect installation running in distributed mode.

Procedure

1. Prepare a JSON payload with the connector configuration. For example:

```
{
  "name": "my-connector",
  "config": {
    "connector.class":
    "org.apache.kafka.connect.file.FileStreamSinkConnector",
    "tasks.max": "1",
    "topics": "my-topic-1,my-topic-2",
    "file": "/tmp/output-file.txt"
  }
}
```

2. Send a POST request to **<KafkaConnectAddress>:8083/connectors** to create the connector. The following example uses **curl**:

```
curl -X POST -H "Content-Type: application/json" --data @sink-
connector.json http://connect0.my-domain.com:8083/connectors
```

3. Verify that the connector was deployed by sending a GET request to **<KafkaConnectAddress>:8083/connectors**. The following example uses **curl**:

```
curl http://connect0.my-domain.com:8083/connectors
```

7.2.5. Deleting connectors

This procedure describes how to use the Kafka Connect REST API to delete a connector plug-in from Kafka Connect in distributed mode.

Prerequisites

- A Kafka Connect installation running in distributed mode.

Deleting connectors

1. Verify that the connector exists by sending a **GET** request to **<KafkaConnectAddress>:8083/connectors/<ConnectorName>**. The following example uses **curl**:

```
curl http://connect0.my-domain.com:8083/connectors
```

2. To delete the connector, send a **DELETE** request to **<KafkaConnectAddress>:8083/connectors**. The following example uses **curl**:

```
curl -X DELETE http://connect0.my-domain.com:8083/connectors/my-
connector
```

3. Verify that the connector was deleted by sending a GET request to **<KafkaConnectAddress>:8083/connectors**. The following example uses **curl**:

```
curl http://connect0.my-domain.com:8083/connectors
```

7.3. CONNECTOR PLUG-INS

The following connector plug-ins are included with AMQ Streams.

FileStreamSink Reads data from Kafka topics and writes the data to a file.

FileStreamSource Reads data from a file and sends the data to Kafka topics.

You can add more connector plug-ins if needed. Kafka Connect searches for and runs additional connector plug-ins at startup. To define the path that kafka Connect searches for plug-ins, set the **plugin.path configuration** option:

```
plugin.path=/opt/kafka/connector-plugins,/opt/connectors
```

The **plugin.path** configuration option can contain a comma-separated list of paths.

When running Kafka Connect in distributed mode, plug-ins must be made available on all worker nodes.

7.4. ADDING CONNECTOR PLUGINS

This procedure shows you how to add additional connector plug-ins.

Prerequisites

- An installed and running AMQ Streams cluster.

Procedure

1. Create the **/opt/kafka/connector-plugins** directory.

```
su - kafka
mkdir /opt/kafka/connector-plugins
```

2. Edit the **/opt/kafka/config/connect-standalone.properties** or **/opt/kafka/config/connect-distributed.properties** Kafka Connect configuration file, and set the **plugin.path** option to **/opt/kafka/connector-plugins**. For example:

```
plugin.path=/opt/kafka/connector-plugins
```

3. Copy your connector plug-ins to **/opt/kafka/connector-plugins**.
4. Start or restart the Kafka Connect workers.

Additional resources

- For more information on installing AMQ Streams, see [Section 2.3, “Installing AMQ Streams”](#).
- For more information on configuring AMQ Streams, see [Section 2.7, “Configuring AMQ Streams”](#).
- For more information on running Kafka Connect in standalone mode, see [Section 7.1.3, “Running Kafka Connect in standalone mode”](#).
- For more information on running Kafka Connect in distributed mode, see [Section 7.2.3, “Running distributed Kafka Connect”](#).

- For a complete list of supported Kafka Connect configuration options, see [Appendix F, *Kafka Connect configuration parameters*](#).

APPENDIX A. BROKER CONFIGURATION PARAMETERS

Name	Description	Type	Default	Valid Values	Importance
zookeeper.connect	Zookeeper host string	string			high
advertised.host.name	<p>DEPRECATED : only used when advertised.listeners or listeners are not set. Use advertised.listeners instead.</p> <p>Hostname to publish to ZooKeeper for clients to use. In IaaS environments, this may need to be different from the interface to which the broker binds. If this is not set, it will use the value for host.name if configured. Otherwise it will use the value returned from <code>java.net.InetAddress.getCanonicalHostName()</code>.</p>	string	null		high

Name	Description	Type	Default	Valid Values	Importance
advertised.listeners	Listeners to publish to ZooKeeper for clients to use, if different than the listeners config property. In IaaS environments, this may need to be different from the interface to which the broker binds. If this is not set, the value for listeners will be used. Unlike listeners it is not valid to advertise the 0.0.0.0 meta-address.	string	null		high

Name	Description	Type	Default	Valid Values	Importance
advertised.port	DEPRECATED : only used when advertised.listeners or listeners are not set. Use advertised.listeners instead. The port to publish to ZooKeeper for clients to use. In IaaS environments, this may need to be different from the port to which the broker binds. If this is not set, it will publish the same port that the broker binds to.	int	null		high
auto.create.topics.enable	Enable auto creation of topic on the server	boolean	true		high
auto.leader.rebalance.enable	Enables auto leader balancing. A background thread checks and triggers leader balance if required at regular intervals	boolean	true		high
background.threads	The number of threads to use for various background processing tasks	int	10	[1,...]	high

Name	Description	Type	Default	Valid Values	Importance
broker.id	The broker id for this server. If unset, a unique broker id will be generated. To avoid conflicts between zookeeper generated broker id's and user configured broker id's, generated broker ids start from reserved.broker.max.id + 1.	int	-1		high
compression.type	Specify the final compression type for a given topic. This configuration accepts the standard compression codecs ('gzip', 'snappy', 'lz4'). It additionally accepts 'uncompressed' which is equivalent to no compression; and 'producer' which means retain the original compression codec set by the producer.	string	producer		high

Name	Description	Type	Default	Valid Values	Importance
delete.topic.enable	Enables delete topic. Delete topic through the admin tool will have no effect if this config is turned off	boolean	true		high
host.name	DEPRECATED : only used when listeners is not set. Use listeners instead. hostname of broker. If this is set, it will only bind to this address. If this is not set, it will bind to all interfaces	string	""		high
leader.imbalance.check.interval.seconds	The frequency with which the partition rebalance check is triggered by the controller	long	300		high
leader.imbalance.per.broker.percentage	The ratio of leader imbalance allowed per broker. The controller would trigger a leader balance if it goes above this value per broker. The value is specified in percentage.	int	10		high

Name	Description	Type	Default	Valid Values	Importance
listeners	Listener List - Comma-separated list of URIs we will listen on and the listener names. If the listener name is not a security protocol, listener.security.protocol.map must also be set. Specify hostname as 0.0.0.0 to bind to all interfaces. Leave hostname empty to bind to default interface. Examples of legal listener lists: PLAINTEXT://myhost:9092,S SL://:9091 CLIENT://0.0.0.0:9092,REPLICATION://localhost:9093	string	null		high
log.dir	The directory in which the log data is kept (supplemental for log.dirs property)	string	/tmp/kafka-logs		high
log.dirs	The directories in which the log data is kept. If not set, the value in log.dir is used	string	null		high

Name	Description	Type	Default	Valid Values	Importance
log.flush.interval.messages	The number of messages accumulated on a log partition before messages are flushed to disk	long	9223372036854775807	[1,...]	high
log.flush.interval.ms	The maximum time in ms that a message in any topic is kept in memory before flushed to disk. If not set, the value in log.flush.scheduler.interval.ms is used	long	null		high
log.flush.offset.checkpoint.interval.ms	The frequency with which we update the persistent record of the last flush which acts as the log recovery point	int	60000	[0,...]	high
log.flush.scheduler.interval.ms	The frequency in ms that the log flusher checks whether any log needs to be flushed to disk	long	9223372036854775807		high
log.flush.start.offset.checkpoint.interval.ms	The frequency with which we update the persistent record of log start offset	int	60000	[0,...]	high
log.retention.bytes	The maximum size of the log before deleting it	long	-1		high

Name	Description	Type	Default	Valid Values	Importance
log.retention.hours	The number of hours to keep a log file before deleting it (in hours), tertiary to log.retention.ms property	int	168		high
log.retention.minutes	The number of minutes to keep a log file before deleting it (in minutes), secondary to log.retention.ms property. If not set, the value in log.retention.hours is used	int	null		high
log.retention.ms	The number of milliseconds to keep a log file before deleting it (in milliseconds), If not set, the value in log.retention.minutes is used	long	null		high
log.roll.hours	The maximum time before a new log segment is rolled out (in hours), secondary to log.roll.ms property	int	168	[1,...]	high

Name	Description	Type	Default	Valid Values	Importance
log.roll.jitter.hours	The maximum jitter to subtract from logRollTimeMill is (in hours), secondary to log.roll.jitter.ms property	int	0	[0,...]	high
log.roll.jitter.ms	The maximum jitter to subtract from logRollTimeMill is (in milliseconds). If not set, the value in log.roll.jitter.hours is used	long	null		high
log.roll.ms	The maximum time before a new log segment is rolled out (in milliseconds). If not set, the value in log.roll.hours is used	long	null		high
log.segment.bytes	The maximum size of a single log file	int	1073741824	[14,...]	high
log.segment.delete.delay.ms	The amount of time to wait before deleting a file from the filesystem	long	60000	[0,...]	high

Name	Description	Type	Default	Valid Values	Importance
message.max.bytes	<p>The largest record batch size allowed by Kafka. If this is increased and there are consumers older than 0.10.2, the consumers' fetch size must also be increased so that they can fetch record batches this large.</p> <p>In the latest message format version, records are always grouped into batches for efficiency. In previous message format versions, uncompressed records are not grouped into batches and this limit only applies to a single record in that case.</p> <p>This can be set per topic with the topic level max.message.bytes config.</p>	int	1000012	[0,...]	high

Name	Description	Type	Default	Valid Values	Importance
min.insync.replicas	<p>When a producer sets acks to "all" (or "-1"), min.insync.replicas specifies the minimum number of replicas that must acknowledge a write for the write to be considered successful. If this minimum cannot be met, then the producer will raise an exception (either NotEnoughReplicas or NotEnoughReplicasAfterAppend). When used together, min.insync.replicas and acks allow you to enforce greater durability guarantees. A typical scenario would be to create a topic with a replication factor of 3, set min.insync.replicas to 2, and produce with acks of "all". This will ensure that the producer raises an exception if a majority of replicas do not receive a write.</p>	int	1	[1,...]	high

Name	Description	Type	Default	Valid Values	Importance
num.io.threads	The number of threads that the server uses for processing requests, which may include disk I/O	int	8	[1,...]	high
num.network.threads	The number of threads that the server uses for receiving requests from the network and sending responses to the network	int	3	[1,...]	high
num.recovery.threads.per.data.dir	The number of threads per data directory to be used for log recovery at startup and flushing at shutdown	int	1	[1,...]	high
num.replica.alter.log.dirs.threads	The number of threads that can move replicas between log directories, which may include disk I/O	int	null		high

Name	Description	Type	Default	Valid Values	Importance
num.replic a.fetchers	Number of fetcher threads used to replicate messages from a source broker. Increasing this value can increase the degree of I/O parallelism in the follower broker.	int	1		high
offset.met adata.max. bytes	The maximum size for a metadata entry associated with an offset commit	int	4096		high
offsets.co mmit.requi red.acks	The required acks before the commit can be accepted. In general, the default (-1) should not be overridden	short	-1		high
offsets.co mmit.timeo ut.ms	Offset commit will be delayed until all replicas for the offsets topic receive the commit or this timeout is reached. This is similar to the producer request timeout.	int	5000	[1,...]	high
offsets.lo ad.buffer. size	Batch size for reading from the offsets segments when loading offsets into the cache.	int	5242880	[1,...]	high

Name	Description	Type	Default	Valid Values	Importance
offsets.retention.check.interval.ms	Frequency at which to check for stale offsets	long	600000	[1,...]	high
offsets.retention.minutes	Offsets older than this retention period will be discarded	int	10080	[1,...]	high
offsets.topic.compression.codec	Compression codec for the offsets topic - compression may be used to achieve "atomic" commits	int	0		high
offsets.topic.num.partitions	The number of partitions for the offset commit topic (should not change after deployment)	int	50	[1,...]	high
offsets.topic.replication.factor	The replication factor for the offsets topic (set higher to ensure availability). Internal topic creation will fail until the cluster size meets this replication factor requirement.	short	3	[1,...]	high

Name	Description	Type	Default	Valid Values	Importance
offsets.topic.segment.bytes	The offsets topic segment bytes should be kept relatively small in order to facilitate faster log compaction and cache loads	int	104857600	[1,...]	high
port	DEPRECATED : only used when listeners is not set. Use listeners instead. the port to listen and accept connections on	int	9092		high
queued.max.requests	The number of queued requests allowed before blocking the network threads	int	500	[1,...]	high
quota.consumer.default	DEPRECATED : Used only when dynamic default quotas are not configured for <user, <client-id> or <user, client-id> in Zookeeper. Any consumer distinguished by clientId/consumer group will get throttled if it fetches more bytes than this value per-second	long	9223372036854775807	[1,...]	high

Name	Description	Type	Default	Valid Values	Importance
quota.producer.default	DEPRECATED : Used only when dynamic default quotas are not configured for <user>, <client-id> or <user, client-id> in Zookeeper. Any producer distinguished by clientId will get throttled if it produces more bytes than this value per-second	long	9223372036854775807	[1,...]	high
replica.fetch.min.bytes	Minimum bytes expected for each fetch response. If not enough bytes, wait up to replicaMaxWaitTimeMs	int	1		high
replica.fetch.wait.max.ms	max wait time for each fetcher request issued by follower replicas. This value should always be less than the replica.lag.time.max.ms at all times to prevent frequent shrinking of ISR for low throughput topics	int	500		high

Name	Description	Type	Default	Valid Values	Importance
replica.high.watermark.checkpoint.interval.ms	The frequency with which the high watermark is saved out to disk	long	5000		high
replica.lag.time.max.ms	If a follower hasn't sent any fetch requests or hasn't consumed up to the leaders log end offset for at least this time, the leader will remove the follower from isr	long	10000		high
replica.socket.receive.buffer.bytes	The socket receive buffer for network requests	int	65536		high
replica.socket.timeout.ms	The socket timeout for network requests. Its value should be at least replica.fetch.wait.max.ms	int	30000		high

Name	Description	Type	Default	Valid Values	Importance
request.timeout.ms	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.	int	30000		high
socket.receive.buffer.bytes	The SO_RCVBUF buffer of the socket server sockets. If the value is -1, the OS default will be used.	int	102400		high
socket.request.max.bytes	The maximum number of bytes in a socket request	int	104857600	[1,...]	high
socket.send.buffer.bytes	The SO_SNDBUF buffer of the socket server sockets. If the value is -1, the OS default will be used.	int	102400		high

Name	Description	Type	Default	Valid Values	Importance
transaction.max.timeout.ms	The maximum allowed timeout for transactions. If a client's requested transaction time exceed this, then the broker will return an error in InitProducerId Request. This prevents a client from too large of a timeout, which can stall consumers reading from topics included in the transaction.	int	900000	[1,...]	high
transaction.state.log.load.buffer.size	Batch size for reading from the transaction log segments when loading producer ids and transactions into the cache.	int	5242880	[1,...]	high
transaction.state.log.min.isr	Overridden min.insync.replicas config for the transaction topic.	int	2	[1,...]	high
transaction.state.log.num.partitions	The number of partitions for the transaction topic (should not change after deployment).	int	50	[1,...]	high

Name	Description	Type	Default	Valid Values	Importance
transaction.state.log.replication.factor	The replication factor for the transaction topic (set higher to ensure availability). Internal topic creation will fail until the cluster size meets this replication factor requirement.	short	3	[1,...]	high
transaction.state.log.segment.bytes	The transaction topic segment bytes should be kept relatively small in order to facilitate faster log compaction and cache loads	int	104857600	[1,...]	high
transactional.id.expiration.ms	The maximum amount of time in ms that the transaction coordinator will wait before proactively expire a producer's transactional id without receiving any transaction status updates from it.	int	604800000	[1,...]	high

Name	Description	Type	Default	Valid Values	Importance
unclean.leader.election.enable	Indicates whether to enable replicas not in the ISR set to be elected as leader as a last resort, even though doing so may result in data loss	boolean	false		high
zookeeper.connection.timeout.ms	The max time that the client waits to establish a connection to zookeeper. If not set, the value in <code>zookeeper.session.timeout.ms</code> is used	int	null		high
zookeeper.max.in.flight.requests	The maximum number of unacknowledged requests the client will send to Zookeeper before blocking.	int	10	[1,...]	high
zookeeper.session.timeout.ms	Zookeeper session timeout	int	6000		high
zookeeper.set.acl	Set client to use secure ACLs	boolean	false		high

Name	Description	Type	Default	Valid Values	Importance
broker.id.generation.enable	Enable automatic broker id generation on the server. When enabled the value configured for reserved.broker.max.id should be reviewed.	boolean	true		medium
broker.rack	Rack of the broker. This will be used in rack aware replication assignment for fault tolerance. Examples: RACK1, us-east-1d	string	null		medium
connections.max.idle.ms	Idle connections timeout: the server socket processor threads close the connections that idle more than this	long	600000		medium
controlled.shutdown.enable	Enable controlled shutdown of the server	boolean	true		medium
controlled.shutdown.max.retries	Controlled shutdown can fail for multiple reasons. This determines the number of retries when such failure happens	int	3		medium

Name	Description	Type	Default	Valid Values	Importance
controlled.shutdown.retry.backoff.ms	Before each retry, the system needs time to recover from the state that caused the previous failure (Controller fail over, replica lag etc). This config determines the amount of time to wait before retrying.	long	5000		medium
controller.socket.timeout.ms	The socket timeout for controller-to-broker channels	int	30000		medium
default.replication.factor	default replication factors for automatically created topics	int	1		medium
delegation.token.expiry.time.ms	The token validity time in seconds before the token needs to be renewed. Default value 1 day.	long	86400000	[1,...]	medium

Name	Description	Type	Default	Valid Values	Importance
delegation.token.master.key	Master/secret key to generate and verify delegation tokens. Same key must be configured across all the brokers. If the key is not set or set to empty string, brokers will disable the delegation token support.	password	null		medium
delegation.token.max.lifetime.ms	The token has a maximum lifetime beyond which it cannot be renewed anymore. Default value 7 days.	long	604800000	[1,...]	medium
delete.records.purgatory.purge.interval.requests	The purge interval (in number of requests) of the delete records request purgatory	int	1		medium
fetch.purgatory.purge.interval.requests	The purge interval (in number of requests) of the fetch request purgatory	int	1000		medium

Name	Description	Type	Default	Valid Values	Importance
group.initial.rebalance.delay.ms	The amount of time the group coordinator will wait for more consumers to join a new group before performing the first rebalance. A longer delay means potentially fewer rebalances, but increases the time until processing begins.	int	3000		medium
group.max.session.timeout.ms	The maximum allowed session timeout for registered consumers. Longer timeouts give consumers more time to process messages in between heartbeats at the cost of a longer time to detect failures.	int	300000		medium

Name	Description	Type	Default	Valid Values	Importance
group.min.session.timeout.ms	The minimum allowed session timeout for registered consumers. Shorter timeouts result in quicker failure detection at the cost of more frequent consumer heartbeating, which can overwhelm broker resources.	int	6000		medium
inter.broker.listener.name	Name of listener used for communication between brokers. If this is unset, the listener name is defined by security.inter.broker.protocol. It is an error to set this and security.inter.broker.protocol properties at the same time.	string	null		medium

Name	Description	Type	Default	Valid Values	Importance
inter.broker.protocol.version	Specify which version of the inter-broker protocol will be used. This is typically bumped after all brokers were upgraded to a new version. Example of some valid values are: 0.8.0, 0.8.1, 0.8.1.1, 0.8.2, 0.8.2.0, 0.8.2.1, 0.9.0.0, 0.9.0.1 Check ApiVersion for the full list.	string	2.0-IV1		medium
log.cleaner.backoff.ms	The amount of time to sleep when there are no logs to clean	long	15000	[0,...]	medium
log.cleaner.dedupe.buffer.size	The total memory used for log deduplication across all cleaner threads	long	134217728		medium
log.cleaner.delete.retention.ms	How long are delete records retained?	long	86400000		medium

Name	Description	Type	Default	Valid Values	Importance
log.cleaner.enable	Enable the log cleaner process to run on the server. Should be enabled if using any topics with a cleanup.policy=compact including the internal offsets topic. If disabled those topics will not be compacted and continually grow in size.	boolean	true		medium
log.cleaner.io.buffer.load.factor	Log cleaner dedupe buffer load factor. The percentage full the dedupe buffer can become. A higher value will allow more log to be cleaned at once but will lead to more hash collisions	double	0.9		medium
log.cleaner.io.buffer.size	The total memory used for log cleaner I/O buffers across all cleaner threads	int	524288	[0,...]	medium
log.cleaner.io.max.bytes.per.second	The log cleaner will be throttled so that the sum of its read and write i/o will be less than this value on average	double	1.7976931348623157E308		medium

Name	Description	Type	Default	Valid Values	Importance
log.cleaner.min.cleannable.ratio	The minimum ratio of dirty log to total log for a log to eligible for cleaning	double	0.5		medium
log.cleaner.min.compaction.lag.ms	The minimum time a message will remain uncompactd in the log. Only applicable for logs that are being compacted.	long	0		medium
log.cleaner.threads	The number of background threads to use for log cleaning	int	1	[0,...]	medium
log.cleanup.policy	The default cleanup policy for segments beyond the retention window. A comma separated list of valid policies. Valid policies are: "delete" and "compact"	list	delete	[compact, delete]	medium
log.index.interval.bytes	The interval with which we add an entry to the offset index	int	4096	[0,...]	medium
log.index.size.max.bytes	The maximum size in bytes of the offset index	int	10485760	[4,...]	medium

Name	Description	Type	Default	Valid Values	Importance
log.message.format.version	Specify the message format version the broker will use to append messages to the logs. The value should be a valid ApiVersion. Some examples are: 0.8.2, 0.9.0.0, 0.10.0, check ApiVersion for more details. By setting a particular message format version, the user is certifying that all the existing messages on disk are smaller or equal than the specified version. Setting this value incorrectly will cause consumers with older versions to break as they will receive messages with a format that they don't understand.	string	2.0-IV1		medium

Name	Description	Type	Default	Valid Values	Importance
log.message.timestamp.difference.max.ms	The maximum difference allowed between the timestamp when a broker receives a message and the timestamp specified in the message. If log.message.timestamp.type = CreateTime, a message will be rejected if the difference in timestamp exceeds this threshold. This configuration is ignored if log.message.timestamp.type = LogAppendTime. The maximum timestamp difference allowed should be no greater than log.retention.ms to avoid unnecessarily frequent log rolling.	long	9223372036854775807		medium
log.message.timestamp.type	Define whether the timestamp in the message is message create time or log append time. The value should be either CreateTime or LogAppendTime	string	CreateTime	[CreateTime, LogAppendTime]	medium

Name	Description	Type	Default	Valid Values	Importance
log.preallocate	Should pre allocate file when create new segment? If you are using Kafka on Windows, you probably need to set it to true.	boolean	false		medium
log.retention.check.interval.ms	The frequency in milliseconds that the log cleaner checks whether any log is eligible for deletion	long	300000	[1,...]	medium
max.connections.per.ip	The maximum number of connections we allow from each ip address. This can be set to 0 if there are overrides configured using max.connections.per.ip.overrides property	int	2147483647	[0,...]	medium
max.connections.per.ip.overrides	A comma-separated list of per-ip or hostname overrides to the default maximum number of connections. An example value is "hostName:100,127.0.0.1:200"	string	""		medium

Name	Description	Type	Default	Valid Values	Importance
max.incremental.fetch.session.cache.slots	The maximum number of incremental fetch sessions that we will maintain.	int	1000	[0,...]	medium
num.partitions	The default number of log partitions per topic	int	1	[1,...]	medium
password.encoder.old.secret	The old secret that was used for encoding dynamically configured passwords. This is required only when the secret is updated. If specified, all dynamically encoded passwords are decoded using this old secret and re-encoded using password.encoder.secret when broker starts up.	password	null		medium
password.encoder.secret	The secret used for encoding dynamically configured passwords for this broker.	password	null		medium
principal.builder.class	The fully qualified name of a class that implements the KafkaPrincipal Builder interface, which is used	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
	<p>to build the KafkaPrincipal object used during authorization. This config also supports the deprecated PrincipalBuilder interface which was previously used for client authentication over SSL. If no principal builder is defined, the default behavior depends on the security protocol in use. For SSL authentication, the principal name will be the distinguished name from the client certificate if one is provided; otherwise, if client authentication is not required, the principal name will be ANONYMOUS. For SASL authentication, the principal will be derived using the rules defined by sasl.kerberos.principal.to.local.rules if GSSAPI is in use, and the SASL authentication ID for other mechanisms.</p>				

Name	Description	Type	Default	Valid Values	Importance
	For PLAINTEXT, the principal will be ANONYMOUS.				
producer.purgatory.purge.interval.requests	The purge interval (in number of requests) of the producer request purgatory	int	1000		medium
queued.max.request.bytes	The number of queued bytes allowed before no more requests are read	long	-1		medium
replica.fetch.backoff.ms	The amount of time to sleep when fetch partition error occurs.	int	1000	[0,...]	medium

Name	Description	Type	Default	Valid Values	Importance
replica.fetch.max.bytes	The number of bytes of messages to attempt to fetch for each partition. This is not an absolute maximum, if the first record batch in the first non-empty partition of the fetch is larger than this value, the record batch will still be returned to ensure that progress can be made. The maximum record batch size accepted by the broker is defined via message.max.bytes (broker config) or max.message.bytes (topic config).	int	1048576	[0,...]	medium

Name	Description	Type	Default	Valid Values	Importance
replica.fetch.response.max.bytes	Maximum bytes expected for the entire fetch response. Records are fetched in batches, and if the first record batch in the first non-empty partition of the fetch is larger than this value, the record batch will still be returned to ensure that progress can be made. As such, this is not an absolute maximum. The maximum record batch size accepted by the broker is defined via message.max.bytes (broker config) or max.message.bytes (topic config).	int	10485760	[0,...]	medium
reserved.broker.max.id	Max number that can be used for a broker.id	int	1000	[0,...]	medium
sasl.client.callback.handler.class	The fully qualified name of a SASL client callback handler class that implements the AuthenticateCallbackHandler interface.	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.enabled.mechanisms	The list of SASL mechanisms enabled in the Kafka server. The list may contain any mechanism for which a security provider is available. Only GSSAPI is enabled by default.	list	GSSAPI		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.jaas.config	JAAS login context parameters for SASL connections in the format used by JAAS configuration files. JAAS configuration file format is described here . The format for the value is: 'loginModuleClass controlFlag (optionName=optionValue)*;'. For brokers, the config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScrumLoginModule required;	password	null		medium
sasl.kerberos.kinit.cmd	Kerberos kinit command path.	string	/usr/bin/kinit		medium
sasl.kerberos.min.time.before.relogin	Login thread sleep time between refresh attempts.	long	60000		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.kerberos.principal.to.local.rules	<p>A list of rules for mapping from principal names to short names (typically operating system usernames). The rules are evaluated in order and the first rule that matches a principal name is used to map it to a short name. Any later rules in the list are ignored. By default, principal names of the form {username}/{hostname}@{REALM} are mapped to {username}. For more details on the format please see <code>#security_authz[security authorization and acls]</code>. Note that this configuration is ignored if an extension of <code>KafkaPrincipalBuilder</code> is provided by the principal.builder.class configuration.</p>	list	DEFAULT		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.kerberos.service.name	The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.	string	null		medium
sasl.kerberos.ticket.renew.jitter	Percentage of random jitter added to the renewal time.	double	0.05		medium
sasl.kerberos.ticket.renew.window.factor	Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.	double	0.8		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.callback.handler.class	The fully qualified name of a SASL login callback handler class that implements the AuthenticateCallbackHandler interface. For brokers, login callback handler config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.class	The fully qualified name of a class that implements the Login interface. For brokers, login config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.buffer.seconds	The amount of buffer time before credential expiration to maintain when refreshing a credential, in seconds. If a refresh would otherwise occur closer to expiration than the number of buffer seconds then the refresh will be moved up to maintain as much of the buffer time as possible. Legal values are between 0 and 3600 (1 hour); a default value of 300 (5 minutes) is used if no value is specified. This value and sasl.login.refresh.min.period.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.	short	300		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.min.period.seconds	The desired minimum time for the login refresh thread to wait before refreshing a credential, in seconds. Legal values are between 0 and 900 (15 minutes); a default value of 60 (1 minute) is used if no value is specified. This value and sasl.login.refresh.buffer.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.	short	60		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.window.factor	Login refresh thread will sleep until the specified window factor relative to the credential's lifetime has been reached, at which time it will try to refresh the credential. Legal values are between 0.5 (50%) and 1.0 (100%) inclusive; a default value of 0.8 (80%) is used if no value is specified. Currently applies only to OAUTHBEARER.	double	0.8		medium
sasl.login.refresh.window.jitter	The maximum amount of random jitter relative to the credential's lifetime that is added to the login refresh thread's sleep time. Legal values are between 0 and 0.25 (25%) inclusive; a default value of 0.05 (5%) is used if no value is specified. Currently applies only to OAUTHBEARER.	double	0.05		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.mechanism.inter.broker.protocol	SASL mechanism used for inter-broker communication. Default is GSSAPI.	string	GSSAPI		medium
sasl.server.callback.handler.class	The fully qualified name of a SASL server callback handler class that implements the AuthenticateCallbackHandler interface. Server callback handlers must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.plain.sasl.server.callback.handler.class=com.example.CustomPlainCallbackHandler.	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
security.inter.broker.protocol	Security protocol used to communicate between brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL. It is an error to set this and <code>inter.broker.listener.name</code> properties at the same time.	string	PLAINTEXT		medium
ssl.cipher.suites	A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol. By default all the available cipher suites are supported.	list	""		medium
ssl.client.auth	Configures kafka broker to request client authentication. The following settings are common: <ul style="list-style-type: none"> ssl.client.auth= 	string	none	[required, requested, none]	medium

Name	Description	Type	Default	Valid Values	Importance
	<p>If set to required client authentication is required.</p> <ul style="list-style-type: none">• ssl.client.auth=requested This means client authentication is optional. Unlike requested, if this option is set client can choose not to provide authentication information about itself• ssl.client.auth=none This means client authentication is				

Name	Description	Type	Default	Valid Values	Importance
ssl.enabled.protocols	The list of protocols enabled for SSL connections.	list	TLSv1.2,TLSv1.1,TLSv1		medium
ssl.key.password	The password of the private key in the key store file. This is optional for client.	password	null		medium
ssl.keymanager.algorithm	The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.	string	SunX509		medium
ssl.keystore.location	The location of the key store file. This is optional for client and can be used for two-way authentication for client.	string	null		medium
ssl.keystore.password	The store password for the key store file. This is optional for client and only needed if ssl.keystore.location is configured.	password	null		medium

Name	Description	Type	Default	Valid Values	Importance
ssl.keystore.type	The file format of the key store file. This is optional for client.	string	JKS		medium
ssl.protocol	The SSL protocol used to generate the SSLContext. Default setting is TLS, which is fine for most cases. Allowed values in recent JVMs are TLS, TLSv1.1 and TLSv1.2. SSL, SSLv2 and SSLv3 may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities.	string	TLS		medium
ssl.provider	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.	string	null		medium

Name	Description	Type	Default	Valid Values	Importance
ssl.trustmanager.algorithm	The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.	string	PKIX		medium
ssl.truststore.location	The location of the trust store file.	string	null		medium
ssl.truststore.password	The password for the trust store file. If a password is not set access to the truststore is still available, but integrity checking is disabled.	password	null		medium
ssl.truststore.type	The file format of the trust store file.	string	JKS		medium
alter.config.policy.class.name	The alter configs policy class that should be used for validation. The class should implement the org.apache.kafka.server.policy.AlterConfigPolicy interface.	class	null		low

Name	Description	Type	Default	Valid Values	Importance
alter.log.dirs.replication.quota.window.num	The number of samples to retain in memory for alter log dirs replication quotas	int	11	[1,...]	low
alter.log.dirs.replication.quota.window.size.seconds	The time span of each sample for alter log dirs replication quotas	int	1	[1,...]	low
authorizer.class.name	The authorizer class that should be used for authorization	string	""		low
client.quota.callback.class	The fully qualified name of a class that implements the ClientQuotaCallback interface, which is used to determine quota limits applied to client requests. By default, <user, client-id>, <user> or <client-id> quotas stored in ZooKeeper are applied. For any given request, the most specific quota that matches the user principal of the session and the client-id of the request is applied.	class	null		low

Name	Description	Type	Default	Valid Values	Importance
create.topic.policy.class.name	The create topic policy class that should be used for validation. The class should implement the org.apache.kafka.server.policy.CreateTopicPolicy interface.	class	null		low
delegation.token.expiry.check.interval.ms	Scan interval to remove expired delegation tokens.	long	3600000	[1,...]	low
listener.security.protocol.map	Map between listener names and security protocols. This must be defined for the same security protocol to be usable in more than one port or IP. For example, internal and external traffic can be separated even if SSL is required for both. Concretely, the user could define listeners with names INTERNAL and EXTERNAL and this property as: INTERNAL:SSL,EXTERNAL:SSL . As shown, key	string	PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL		low

Name	Description	Type	Default	Valid Values	Importance
	<p>and value are separated by a colon and map entries are separated by commas. Each listener name should only appear once in the map. Different security (SSL and SASL) settings can be configured for each listener by adding a normalised prefix (the listener name is lowercased) to the config name. For example, to set a different keystore for the INTERNAL listener, a config with name listener.name.internal.ssl.keystore.location would be set. If the config for the listener name is not set, the config will fallback to the generic config (i.e. ssl.keystore.location).</p>				

Name	Description	Type	Default	Valid Values	Importance
log.message.downconversion.enable	<p>This configuration controls whether down-conversion of message formats is enabled to satisfy consume requests. When set to false, broker will not perform down-conversion for consumers expecting an older message format. The broker responds with UNSUPPORTED_VERSION error for consume requests from such older clients. This configuration does not apply to any message format conversion that might be required for replication to followers.</p>	boolean	true		low

Name	Description	Type	Default	Valid Values	Importance
metric.reporters	A list of classes to use as metrics reporters. Implementing the org.apache.kafka.common.metrics.MetricsReporter interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.	list	""		low
metrics.num.samples	The number of samples maintained to compute metrics.	int	2	[1,...]	low
metrics.recording.level	The highest recording level for metrics.	string	INFO		low
metrics.sample.window.ms	The window of time a metrics sample is computed over.	long	30000	[1,...]	low
password.encoder.cipher.algorithm	The Cipher algorithm used for encoding dynamically configured passwords.	string	AES/CBC/PKCS5Padding		low

Name	Description	Type	Default	Valid Values	Importance
password.encoder.iterations	The iteration count used for encoding dynamically configured passwords.	int	4096	[1024,...]	low
password.encoder.key.length	The key length used for encoding dynamically configured passwords.	int	128	[8,...]	low
password.encoder.key.factory.algorithm	The SecretKeyFactory algorithm used for encoding dynamically configured passwords. Default is PBKDF2WithHmacSHA512 if available and PBKDF2WithHmacSHA1 otherwise.	string	null		low
quota.window.num	The number of samples to retain in memory for client quotas	int	11	[1,...]	low
quota.window.size.seconds	The time span of each sample for client quotas	int	1	[1,...]	low
replication.quota.window.num	The number of samples to retain in memory for replication quotas	int	11	[1,...]	low

Name	Description	Type	Default	Valid Values	Importance
replication.quota.window.size.seconds	The time span of each sample for replication quotas	int	1	[1,...]	low
ssl.endpoint.identification.algorithm	The endpoint identification algorithm to validate server hostname using server certificate.	string	https		low
ssl.secure.random.implementation	The SecureRandom PRNG implementation to use for SSL cryptography operations.	string	null		low
transaction.abort.timed.out.transaction.cleanup.interval.ms	The interval at which to rollback transactions that have timed out	int	60000	[1,...]	low
transaction.remove.expired.transaction.cleanup.interval.ms	The interval at which to remove transactions that have expired due to transactional.id.expiration.ms passing	int	3600000	[1,...]	low
zookeeper.sync.time.ms	How far a ZK follower can be behind a ZK leader	int	2000		low

APPENDIX B. TOPIC CONFIGURATION PARAMETERS

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
cleanup.policy	A string that is either "delete" or "compact". This string designates the retention policy to use on old log segments. The default policy ("delete") will discard old segments when their retention time or size limit has been reached. The "compact" setting will enable <code>#compaction[log compaction]</code> on the topic.	list	delete	[compact, delete]	log.cleanup.policy	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
compression.type	Specify the final compression type for a given topic. This configuration accepts the standard compression codecs ('gzip', 'snappy', 'lz4'). It additionally accepts 'uncompressed' which is equivalent to no compression; and 'producer' which means retain the original compression codec set by the producer.	string	producer	[uncompressed, snappy, lz4, gzip, producer]	compression.type	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
delete.retention.ms	The amount of time to retain delete tombstone markers for #compaction[log compacted] topics. This setting also gives a bound on the time in which a consumer must complete a read if they begin from offset 0 to ensure that they get a valid snapshot of the final stage (otherwise delete tombstones may be collected before they complete their scan).	long	86400000	[0,...]	log.cleaner.delete.retention.ms	medium
file.delete.delay.ms	The time to wait before deleting a file from the filesystem	long	60000	[0,...]	log.segment.delete.delay.ms	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
flush.messages	This setting allows specifying an interval at which we will force an fsync of data written to the log. For example if this was set to 1 we would fsync after every message; if it were 5 we would fsync after every five messages. In general we recommend you not set this and use replication for durability and allow the operating system's background flush capabilities as it is more efficient. This setting can be overridden on a per-topic basis (see <code>#topicconfigs</code> [the per-topic configuration section]).	long	9223372036854775807	[0,...]	log.flush.interval.messages	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
flush.ms	This setting allows specifying a time interval at which we will force an fsync of data written to the log. For example if this was set to 1000 we would fsync after 1000 ms had passed. In general we recommend you not set this and use replication for durability and allow the operating system's background flush capabilities as it is more efficient.	long	9223372036854775807	[0,...]	log.flush.interval.ms	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
follower.replication.throttled.replicas	A list of replicas for which log replication should be throttled on the follower side. The list should describe a set of replicas in the form [PartitionId]:[BrokerId], [PartitionId]:[BrokerId]:... or alternatively the wildcard '*' can be used to throttle all replicas for this topic.	list	""	[partitionId], [brokerId]: [partitionId], [brokerId]:...	follower.replication.throttled.replicas	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
index.interval.bytes	This setting controls how frequently Kafka adds an index entry to its offset index. The default setting ensures that we index a message roughly every 4096 bytes. More indexing allows reads to jump closer to the exact position in the log but makes the index larger. You probably don't need to change this.	int	4096	[0,...]	log.index.interval.bytes	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
leader.replication.throttled.replicas	A list of replicas for which log replication should be throttled on the leader side. The list should describe a set of replicas in the form [PartitionId]:[BrokerId], [PartitionId]:[BrokerId]:... or alternatively the wildcard '*' can be used to throttle all replicas for this topic.	list	""	[partitionId], [brokerId]: [partitionId], [brokerId]:...	leader.replication.throttled.replicas	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
max.message.bytes	<p>The largest record batch size allowed by Kafka. If this is increased and there are consumers older than 0.10.2, the consumers' fetch size must also be increased so that they can fetch record batches this large.</p> <p>In the latest message format version, records are always grouped into batches for efficiency. In previous message format versions, uncompressed records are not grouped into batches and this limit only applies to a single record in that case.</p>	int	1000012	[0,...]	message.max.bytes	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
message.format.version	Specify the message format version the broker will use to append messages to the logs. The value should be a valid ApiVersion. Some examples are: 0.8.2, 0.9.0.0, 0.10.0, check ApiVersion for more details. By setting a particular message format version, the user is certifying that all the existing messages on disk are smaller or equal than the specified version. Setting this value incorrectly will cause consumers with older versions to break as they will receive messages with a	string	2.0-IV1		log.message.format.version	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
message.timestamp.difference.max.ms	<p>format that they don't understand. The maximum difference allowed between the timestamp when a broker receives a message and the timestamp specified in the message. If message.timestamp.type=CreateTime, a message will be rejected if the difference in timestamp exceeds this threshold. This configuration is ignored if message.timestamp.type=LogAppendTime.</p>	long	9223372036854775807	[0,...]	log.message.timestamp.difference.max.ms	medium
message.timestamp.type	<p>Define whether the timestamp in the message is message create time or log append time. The value should be either CreateTime or LogAppendTime</p>	string	CreateTime	[CreateTime, LogAppendTime]	log.message.timestamp.type	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
min.cleanable.dirty.ratio	<p>This configuration controls how frequently the log compactor will attempt to clean the log (assuming <code>#compaction[log compaction]</code> is enabled). By default we will avoid cleaning a log where more than 50% of the log has been compacted. This ratio bounds the maximum space wasted in the log by duplicates (at 50% at most 50% of the log could be duplicates). A higher ratio will mean fewer, more efficient cleanings but will mean more wasted space in the log.</p>	double	0.5	[0,...,1]	log.cleaner.min.cleanable.ratio	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
min.compaction.lag.ms	The minimum time a message will remain uncompacted in the log. Only applicable for logs that are being compacted.	long	0	[0,...]	log.cleaner.min.compaction.lag.ms	medium
min.insync.replicas	When a producer sets acks to "all" (or "-1"), this configuration specifies the minimum number of replicas that must acknowledge a write for the write to be considered successful. If this minimum cannot be met, then the producer will raise an exception (either NotEnoughReplicas or NotEnoughReplicasAfterAppend). When used together, min.insync.replicas and acks allow you to enforce	int	1	[1,...]	min.insync.replicas	medium

Name	greater Description durability guarantees. A typical scenario	Type	Default	Valid Values	Server Default Property	Importance
	would be to create a topic with a replication factor of 3, set min.insync.r eplicas to 2, and produce with acks of "all". This will ensure that the producer raises an exception if a majority of replicas do not receive a write.					
preallocate	True if we should preallocate the file on disk when creating a new log segment.	boolean	false		log.prealloc ate	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
retention.bytes	This configuration controls the maximum size a partition (which consists of log segments) can grow to before we will discard old log segments to free up space if we are using the "delete" retention policy. By default there is no size limit only a time limit. Since this limit is enforced at the partition level, multiply it by the number of partitions to compute the topic retention in bytes.	long	-1		log.retention.bytes	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
retention.ms	This configuration controls the maximum time we will retain a log before we will discard old log segments to free up space if we are using the "delete" retention policy. This represents an SLA on how soon consumers must read their data. If set to -1, no time limit is applied.	long	604800000		log.retention.ms	medium
segment.bytes	This configuration controls the segment file size for the log. Retention and cleaning is always done a file at a time so a larger segment size means fewer files but less granular control over retention.	int	1073741824	[14,...]	log.segment.bytes	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
segment.index.bytes	This configuration controls the size of the index that maps offsets to file positions. We preallocate this index file and shrink it only after log rolls. You generally should not need to change this setting.	int	10485760	[0,...]	log.index.size.max.bytes	medium
segment.jitter.ms	The maximum random jitter subtracted from the scheduled segment roll time to avoid thundering herds of segment rolling	long	0	[0,...]	log.roll.jitter.ms	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
segment.ms	This configuration controls the period of time after which Kafka will force the log to roll even if the segment file isn't full to ensure that retention can delete or compact old data.	long	604800000	[1,...]	log.roll.ms	medium
unclean.leader.election.enable	Indicates whether to enable replicas not in the ISR set to be elected as leader as a last resort, even though doing so may result in data loss.	boolean	false		unclean.leader.election.enable	medium

Name	Description	Type	Default	Valid Values	Server Default Property	Importance
message.downconversion.enable	<p>This configuration controls whether down-conversion of message formats is enabled to satisfy consume requests. When set to false, broker will not perform down-conversion for consumers expecting an older message format. The broker responds with UNSUPPORTED_VERSION error for consume requests from such older clients. This configuration does not apply to any message format conversion that might be required for replication to followers.</p>	boolean	true		log.message.downconversion.enable	low

APPENDIX C. CONSUMER CONFIGURATION PARAMETERS

Name	Description	Type	Default	Valid Values	Importance
key.deserializer	Deserializer class for key that implements the org.apache.kafka.common.serialization.Deserializer interface.	class			high
value.deserializer	Deserializer class for value that implements the org.apache.kafka.common.serialization.Deserializer interface.	class			high

Name	Description	Type	Default	Valid Values	Importance
bootstrap.servers	<p>A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping—this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form host1:port1,host2:port2,.... Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).</p>	list	""	non-null value	high

Name	Description	Type	Default	Valid Values	Importance
fetch.min.bytes	The minimum amount of data the server should return for a fetch request. If insufficient data is available the request will wait for that much data to accumulate before answering the request. The default setting of 1 byte means that fetch requests are answered as soon as a single byte of data is available or the fetch request times out waiting for data to arrive. Setting this to something greater than 1 will cause the server to wait for larger amounts of data to accumulate which can improve server throughput a bit at the cost of some additional latency.	int	1	[0,...]	high

Name	Description	Type	Default	Valid Values	Importance
group.id	A unique string that identifies the consumer group this consumer belongs to. This property is required if the consumer uses either the group management functionality by using subscribe(topic) or the Kafka-based offset management strategy.	string	""		high

Name	Description	Type	Default	Valid Values	Importance
heartbeat.interval.ms	The expected time between heartbeats to the consumer coordinator when using Kafka's group management facilities. Heartbeats are used to ensure that the consumer's session stays active and to facilitate rebalancing when new consumers join or leave the group. The value must be set lower than session.timeout.ms , but typically should be set no higher than 1/3 of that value. It can be adjusted even lower to control the expected time for normal rebalances.	int	3000		high

Name	Description	Type	Default	Valid Values	Importance
max.partition.fetch.bytes	<p>The maximum amount of data per-partition the server will return. Records are fetched in batches by the consumer. If the first record batch in the first non-empty partition of the fetch is larger than this limit, the batch will still be returned to ensure that the consumer can make progress. The maximum record batch size accepted by the broker is defined via message.max.bytes (broker config) or max.message.bytes (topic config). See <code>fetch.max.bytes</code> for limiting the consumer request size.</p>	int	1048576	[0,...]	high

Name	Description	Type	Default	Valid Values	Importance
session.timeout.ms	The timeout used to detect consumer failures when using Kafka's group management facility. The consumer sends periodic heartbeats to indicate its liveness to the broker. If no heartbeats are received by the broker before the expiration of this session timeout, then the broker will remove this consumer from the group and initiate a rebalance. Note that the value must be in the allowable range as configured in the broker configuration by group.min.session.timeout.ms and group.max.session.timeout.ms .	int	10000		high
ssl.key.password	The password of the private key in the key store file. This is optional for client.	password	null		high

Name	Description	Type	Default	Valid Values	Importance
ssl.keystore.location	The location of the key store file. This is optional for client and can be used for two-way authentication for client.	string	null		high
ssl.keystore.password	The store password for the key store file. This is optional for client and only needed if ssl.keystore.location is configured.	password	null		high
ssl.truststore.location	The location of the trust store file.	string	null		high
ssl.truststore.password	The password for the trust store file. If a password is not set access to the truststore is still available, but integrity checking is disabled.	password	null		high
auto.offset.reset	What to do when there is no initial offset in Kafka or if the current offset does not exist any more on the server (e.g. because that data has been deleted):	string	latest	[latest, earliest, none]	medium

Name	Description	Type	Default	Valid Values	Importance
	<ul style="list-style-type: none"> • earliest: automatically reset the offset to the earliest offset • latest: automatically reset the offset to the latest offset • none: throw exception to the consumer if no previous offset is found for the consumer's group • anything else: throw exception to the consumer. 				
connections.max.idle.ms	Close idle connections after the number of milliseconds specified by this config.	long	540000		medium

Name	Description	Type	Default	Valid Values	Importance
default.api.timeout.ms	Specifies the timeout (in milliseconds) for consumer APIs that could block. This configuration is used as the default timeout for all consumer operations that do not explicitly accept a timeout parameter.	int	60000	[0,...]	medium
enable.auto.commit	If true the consumer's offset will be periodically committed in the background.	boolean	true		medium
exclude.internal.topics	Whether records from internal topics (such as offsets) should be exposed to the consumer. If set to true the only way to receive records from an internal topic is subscribing to it.	boolean	true		medium

Name	Description	Type	Default	Valid Values	Importance
fetch.max.bytes	<p>The maximum amount of data the server should return for a fetch request. Records are fetched in batches by the consumer, and if the first record batch in the first non-empty partition of the fetch is larger than this value, the record batch will still be returned to ensure that the consumer can make progress. As such, this is not a absolute maximum. The maximum record batch size accepted by the broker is defined via message.max.bytes (broker config) or max.message.bytes (topic config). Note that the consumer performs multiple fetches in parallel.</p>	int	52428800	[0,...]	medium
isolation.level	<p>Controls how to read messages written transactionally. If set to read_commi</p>	string	read_uncommi tted	[read_committ ed, read_uncommi tted]	medium

Name	Description	Type	Default	Valid Values	Importance
	<p>consumer.poll() will only return transactional messages which have been committed. If set to 'read_uncommitted' (the default), consumer.poll() will return all messages, even transactional messages which have been aborted. Non-transactional messages will be returned unconditionally in either mode.</p> <p>Messages will always be returned in offset order. Hence, in read_committed mode, consumer.poll() will only return messages up to the last stable offset (LSO), which is the one less than the offset of the first open transaction. In particular any messages appearing after messages belonging to ongoing transactions will be withheld until the</p>				

Name	Description	Type	Default	Valid Values	Importance
	<p>relevant transaction has been completed. As a result, read_committed consumers will not be able to read up to the high watermark when there are in flight transactions.</p> <p>Further, when in <code>read_committed</code> mode > the <code>seekToEnd</code> method will return the <code>LSO</code></p>				

Name	Description	Type	Default	Valid Values	Importance
max.poll.interval.ms	The maximum delay between invocations of poll() when using consumer group management. This places an upper bound on the amount of time that the consumer can be idle before fetching more records. If poll() is not called before expiration of this timeout, then the consumer is considered failed and the group will rebalance in order to reassign the partitions to another member.	int	300000	[1,...]	medium
max.poll.records	The maximum number of records returned in a single call to poll().	int	500	[1,...]	medium

Name	Description	Type	Default	Valid Values	Importance
partition.assignment.strategy	The class name of the partition assignment strategy that the client will use to distribute partition ownership amongst consumer instances when group management is used	list	class org.apache.kafka.clients.consumer.RangeAssignor	non-null value	medium
receive.buffer.bytes	The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.	int	65536	[-1,...]	medium
request.timeout.ms	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.	int	30000	[0,...]	medium

Name	Description	Type	Default	Valid Values	Importance
sasl.client.callback.handler.class	The fully qualified name of a SASL client callback handler class that implements the <code>AuthenticateCallbackHandler</code> interface.	class	null		medium
sasl.jaas.config	JAAS login context parameters for SASL connections in the format used by JAAS configuration files. JAAS configuration file format is described here . The format for the value is: 'loginModuleClass controlFlag (optionName=optionValue)*;'. For brokers, the config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, <code>listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScamLoginModule required;</code>	password	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.kerberos.service.name	The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.	string	null		medium
sasl.login.callback.handler.class	The fully qualified name of a SASL login callback handler class that implements the AuthenticateCallbackHandler interface. For brokers, login callback handler config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.class	The fully qualified name of a class that implements the Login interface. For brokers, login config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin	class	null		medium
sasl.mechanism	SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.	string	GSSAPI		medium
security.protocol	Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.	string	PLAINTEXT		medium

Name	Description	Type	Default	Valid Values	Importance
send.buffer.bytes	The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.	int	131072	[-1,...]	medium
ssl.enabled.protocols	The list of protocols enabled for SSL connections.	list	TLSv1.2,TLSv1.1,TLSv1		medium
ssl.keystore.type	The file format of the key store file. This is optional for client.	string	JKS		medium
ssl.protocol	The SSL protocol used to generate the SSLContext. Default setting is TLS, which is fine for most cases. Allowed values in recent JVMs are TLS, TLSv1.1 and TLSv1.2. SSL, SSLv2 and SSLv3 may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities.	string	TLS		medium

Name	Description	Type	Default	Valid Values	Importance
ssl.provider	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.	string	null		medium
ssl.truststore.type	The file format of the trust store file.	string	JKS		medium
auto.commit.interval.ms	The frequency in milliseconds that the consumer offsets are auto-committed to Kafka if enable.auto.commit is set to true .	int	5000	[0,...]	low
check.crcs	Automatically check the CRC32 of the records consumed. This ensures no on-the-wire or on-disk corruption to the messages occurred. This check adds some overhead, so it may be disabled in cases seeking extreme performance.	boolean	true		low

Name	Description	Type	Default	Valid Values	Importance
client.id	An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.	string	""		low
fetch.max.wait.ms	The maximum amount of time the server will block before answering the fetch request if there isn't sufficient data to immediately satisfy the requirement given by fetch.min.bytes.	int	500	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
interceptor.classes	A list of classes to use as interceptors. Implementing the org.apache.kafka.clients.consumer.ConsumerInterceptor interface allows you to intercept (and possibly mutate) records received by the consumer. By default, there are no interceptors.	list	""	non-null value	low
metadata.max.age.ms	The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.	long	300000	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
metric.reporters	A list of classes to use as metrics reporters. Implementing the org.apache.kafka.common.metrics.MetricsReporter interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.	list	""	non-null value	low
metrics.num.samples	The number of samples maintained to compute metrics.	int	2	[1,...]	low
metrics.recording.level	The highest recording level for metrics.	string	INFO	[INFO, DEBUG]	low
metrics.sample.window.ms	The window of time a metrics sample is computed over.	long	30000	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
reconnect.backoff.max.ms	The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.	long	1000	[0,...]	low
reconnect.backoff.ms	The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.	long	50	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
retry.backoff.ms	The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.	long	100	[0,...]	low
sasl.kerberos.kinit.cmd	Kerberos kinit command path.	string	/usr/bin/kinit		low
sasl.kerberos.min.time.before.relogin	Login thread sleep time between refresh attempts.	long	60000		low
sasl.kerberos.ticket.renew.jitter	Percentage of random jitter added to the renewal time.	double	0.05		low
sasl.kerberos.ticket.renew.window.factor	Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.	double	0.8		low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.buffer.seconds	The amount of buffer time before credential expiration to maintain when refreshing a credential, in seconds. If a refresh would otherwise occur closer to expiration than the number of buffer seconds then the refresh will be moved up to maintain as much of the buffer time as possible. Legal values are between 0 and 3600 (1 hour); a default value of 300 (5 minutes) is used if no value is specified. This value and sasl.login.refresh.min.period.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.	short	300	[0,...,3600]	low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.min.period.seconds	The desired minimum time for the login refresh thread to wait before refreshing a credential, in seconds. Legal values are between 0 and 900 (15 minutes); a default value of 60 (1 minute) is used if no value is specified. This value and sasl.login.refresh.buffer.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.	short	60	[0,...,900]	low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.window.factor	Login refresh thread will sleep until the specified window factor relative to the credential's lifetime has been reached, at which time it will try to refresh the credential. Legal values are between 0.5 (50%) and 1.0 (100%) inclusive; a default value of 0.8 (80%) is used if no value is specified. Currently applies only to OAUTHBEARER.	double	0.8	[0.5,...,1.0]	low
sasl.login.refresh.window.jitter	The maximum amount of random jitter relative to the credential's lifetime that is added to the login refresh thread's sleep time. Legal values are between 0 and 0.25 (25%) inclusive; a default value of 0.05 (5%) is used if no value is specified. Currently applies only to OAUTHBEARER.	double	0.05	[0.0,...,0.25]	low

Name	Description	Type	Default	Valid Values	Importance
ssl.cipher.suites	A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol. By default all the available cipher suites are supported.	list	null		low
ssl.endpoint.identification.algorithm	The endpoint identification algorithm to validate server hostname using server certificate.	string	https		low
ssl.keymanager.algorithm	The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.	string	SunX509		low

Name	Description	Type	Default	Valid Values	Importance
ssl.secure.random.implementation	The SecureRandom PRNG implementation to use for SSL cryptography operations.	string	null		low
ssl.trustmanager.algorithm	The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.	string	PKIX		low

APPENDIX D. PRODUCER CONFIGURATION PARAMETERS

Name	Description	Type	Default	Valid Values	Importance
key.serializer	Serializer class for key that implements the org.apache.kafka.common.serialization.Serializer interface.	class			high
value.serializer	Serializer class for value that implements the org.apache.kafka.common.serialization.Serializer interface.	class			high
acks	<p>The number of acknowledgments the producer requires the leader to have received before considering a request complete. This controls the durability of records that are sent. The following settings are allowed:</p> <ul style="list-style-type: none"> • acks=0 If set to zero then the producer will not wait for any 	string	1	[all, -1, 0, 1]	high

Name	Description	Type	Default	Valid Values	Importance
	<p>acknowledgment from the server at all. The record will be immediately added to the socket buffer and considered sent. No guarantee can be made that the server has received the record in this case, and the retries configuration will not take effect (as the client won't generally know of any failures). The offset given back for each record will</p>				

Name	Description	Type	Default	Valid Values	Importance
	<p>always set to -1.</p> <ul style="list-style-type: none">• acks=1 This will mean the leader will write the record to its local log but will respond without awaiting full acknowledgment from all followers. In this case should the leader fail immediately after acknowledging the record but before the followers have replicated it then the record will be lost.• acks=all This mean				

Name	Description	Type	Default	Valid Values	Importance
	<p>s the order will wait for the full set of in- sync replic as to ackno wledg e the record . This guara ntees that the record will not be lost as long as at least one in- sync replic a remai ns alive. This is the strong est availa ble guara ntee. This is equiv alent to the acks= -1 settin g.</p>				

Name	Description	Type	Default	Valid Values	Importance
bootstrap.servers	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping —this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form host1:port1,host2:port2,... . Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).	list	""	non-null value	high

Name	Description	Type	Default	Valid Values	Importance
buffer.memory	The total bytes of memory the producer can use to buffer records waiting to be sent to the server. If records are sent faster than they can be delivered to the server the producer will block for max.block.ms after which it will throw an exception. This setting should correspond roughly to the total memory the producer will use, but is not a hard bound since not all memory the producer uses is used for buffering. Some additional memory will be used for compression (if compression is enabled) as well as for maintaining in-flight requests.	long	33554432	[0,...]	high

Name	Description	Type	Default	Valid Values	Importance
compression.type	<p>The compression type for all data generated by the producer. The default is none (i.e. no compression). Valid values are none, gzip, snappy, or lz4. Compression is of full batches of data, so the efficacy of batching will also impact the compression ratio (more batching means better compression).</p>	string	none		high

Name	Description	Type	Default	Valid Values	Importance
retries	Setting a value greater than zero will cause the client to resend any record whose send fails with a potentially transient error. Note that this retry is no different than if the client resent the record upon receiving the error. Allowing retries without setting max.in.flight.requests.per.connection to 1 will potentially change the ordering of records because if two batches are sent to a single partition, and the first fails and is retried but the second succeeds, then the records in the second batch may appear first.	int	0	[0,...,2147483647]	high
ssl.key.password	The password of the private key in the key store file. This is optional for client.	password	null		high

Name	Description	Type	Default	Valid Values	Importance
ssl.keystore.location	The location of the key store file. This is optional for client and can be used for two-way authentication for client.	string	null		high
ssl.keystore.password	The store password for the key store file. This is optional for client and only needed if ssl.keystore.location is configured.	password	null		high
ssl.truststore.location	The location of the trust store file.	string	null		high
ssl.truststore.password	The password for the trust store file. If a password is not set access to the truststore is still available, but integrity checking is disabled.	password	null		high
batch.size	The producer will attempt to batch records together into fewer requests whenever multiple records are being sent to the same partition. This helps performance on both the client and the	int	16384	[0,...]	medium

Name	Description	Type	Default	Valid Values	Importance
	controls the default batch size in bytes. No attempt will be made to batch records larger than this size. Requests sent to brokers will contain multiple batches, one for each partition with data available to be sent. A small batch size will make batching less common and may reduce throughput (a batch size of zero will disable batching entirely). A very large batch size may use memory a bit more wastefully as we will always allocate a buffer of the specified batch size in anticipation of additional records.				

Name	Description	Type	Default	Valid Values	Importance
client.id	An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.	string	""		medium
connections.max.idle.ms	Close idle connections after the number of milliseconds specified by this config.	long	540000		medium
linger.ms	The producer groups together any records that arrive in between request transmissions into a single batched request. Normally this occurs only under load when records arrive faster than they can be sent out. However in some circumstances the client may want to reduce the number of	long	0	[0,...]	medium

Name	Description	Type	Default	Valid Values	Importance
	<p>moderate load. This setting accomplishes this by adding a small amount of artificial delay—that is, rather than immediately sending out a record the producer will wait for up to the given delay to allow other records to be sent so that the sends can be batched together. This can be thought of as analogous to Nagle's algorithm in TCP. This setting gives the upper bound on the delay for batching: once we get batch.size worth of records for a partition it will be sent immediately regardless of this setting, however if we have fewer than this many bytes accumulated for this partition we will 'linger' for the specified time waiting for more records to show up. This setting defaults to 0</p>				

Name	Description	Type	Default	Valid Values	Importance
	(i.e. no delay). linger.ms=5 , for example, would have the effect of reducing the number of requests sent but would add up to 5ms of latency to records sent in the absence of load.				
max.block.ms	The configuration controls how long KafkaProducer.send() and KafkaProducer.partitionsFor() will block. These methods can be blocked either because the buffer is full or metadata unavailable. Blocking in the user-supplied serializers or partitioner will not be counted against this timeout.	long	60000	[0,...]	medium

Name	Description	Type	Default	Valid Values	Importance
max.request.size	The maximum size of a request in bytes. This setting will limit the number of record batches the producer will send in a single request to avoid sending huge requests. This is also effectively a cap on the maximum record batch size. Note that the server has its own cap on record batch size which may be different from this.	int	1048576	[0,...]	medium
partitioner.class	Partitioner class that implements the org.apache.kafka.clients.producer.Partitioner interface.	class	org.apache.kafka.clients.producer.internals.DefaultPartitioner		medium
receive.buffer.bytes	The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.	int	32768	[-1,...]	medium

Name	Description	Type	Default	Valid Values	Importance
request.timeout.ms	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted. This should be larger than <code>replica.lag.time.max.ms</code> (a broker configuration) to reduce the possibility of message duplication due to unnecessary producer retries.	int	30000	[0,...]	medium
sasl.client.callback.handler.class	The fully qualified name of a SASL client callback handler class that implements the <code>AuthenticateCallbackHandler</code> interface.	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.jaas.config	JAAS login context parameters for SASL connections in the format used by JAAS configuration files. JAAS configuration file format is described here . The format for the value is: 'loginModuleClass controlFlag (optionName=optionValue)*;'. For brokers, the config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScamLoginModule required;	password	null		medium
sasl.kerberos.service.name	The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.	string	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.callback.handler.class	The fully qualified name of a SASL login callback handler class that implements the AuthenticateCallbackHandler interface. For brokers, login callback handler config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.class	The fully qualified name of a class that implements the Login interface. For brokers, login config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin	class	null		medium
sasl.mechanism	SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.	string	GSSAPI		medium
security.protocol	Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.	string	PLAINTEXT		medium

Name	Description	Type	Default	Valid Values	Importance
send.buffer.bytes	The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.	int	131072	[-1,...]	medium
ssl.enabled.protocols	The list of protocols enabled for SSL connections.	list	TLSv1.2,TLSv1.1,TLSv1		medium
ssl.keystore.type	The file format of the key store file. This is optional for client.	string	JKS		medium
ssl.protocol	The SSL protocol used to generate the SSLContext. Default setting is TLS, which is fine for most cases. Allowed values in recent JVMs are TLS, TLSv1.1 and TLSv1.2. SSL, SSLv2 and SSLv3 may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities.	string	TLS		medium

Name	Description	Type	Default	Valid Values	Importance
ssl.provider	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.	string	null		medium
ssl.truststore.type	The file format of the trust store file.	string	JKS		medium

Name	Description	Type	Default	Valid Values	Importance
enable.idempotence	<p>When set to 'true', the producer will ensure that exactly one copy of each message is written in the stream. If 'false', producer retries due to broker failures, etc., may write duplicates of the retried message in the stream. Note that enabling idempotence requires max.in.flight.requests.per.connection to be less than or equal to 5, retries to be greater than 0 and acks must be 'all'. If these values are not explicitly set by the user, suitable values will be chosen. If incompatible values are set, a <code>ConfigException</code> will be thrown.</p>	boolean	false		low

Name	Description	Type	Default	Valid Values	Importance
interceptor.classes	A list of classes to use as interceptors. Implementing the org.apache.kafka.clients.producer.ProducerInterceptor interface allows you to intercept (and possibly mutate) the records received by the producer before they are published to the Kafka cluster. By default, there are no interceptors.	list	""	non-null value	low
max.in.flight.requests.per.connection	The maximum number of unacknowledged requests the client will send on a single connection before blocking. Note that if this setting is set to be greater than 1 and there are failed sends, there is a risk of message re-ordering due to retries (i.e., if retries are enabled).	int	5	[1,...]	low

Name	Description	Type	Default	Valid Values	Importance
metadata.max.age.ms	The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.	long	300000	[0,...]	low
metric.reporters	A list of classes to use as metrics reporters. Implementing the org.apache.kafka.common.metrics.MetricsReporter interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.	list	""	non-null value	low
metrics.num.samples	The number of samples maintained to compute metrics.	int	2	[1,...]	low
metrics.recording.level	The highest recording level for metrics.	string	INFO	[INFO, DEBUG]	low

Name	Description	Type	Default	Valid Values	Importance
metrics.sample.window.ms	The window of time a metrics sample is computed over.	long	30000	[0,...]	low
reconnect.backoff.max.ms	The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.	long	1000	[0,...]	low
reconnect.backoff.ms	The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.	long	50	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
retry.backoff.ms	The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.	long	100	[0,...]	low
sasl.kerberos.kinit.cmd	Kerberos kinit command path.	string	/usr/bin/kinit		low
sasl.kerberos.min.time.before.relogin	Login thread sleep time between refresh attempts.	long	60000		low
sasl.kerberos.ticket.renew.jitter	Percentage of random jitter added to the renewal time.	double	0.05		low
sasl.kerberos.ticket.renew.window.factor	Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.	double	0.8		low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.buffer.seconds	The amount of buffer time before credential expiration to maintain when refreshing a credential, in seconds. If a refresh would otherwise occur closer to expiration than the number of buffer seconds then the refresh will be moved up to maintain as much of the buffer time as possible. Legal values are between 0 and 3600 (1 hour); a default value of 300 (5 minutes) is used if no value is specified. This value and sasl.login.refresh.min.period.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.	short	300	[0,...,3600]	low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.min.period.seconds	The desired minimum time for the login refresh thread to wait before refreshing a credential, in seconds. Legal values are between 0 and 900 (15 minutes); a default value of 60 (1 minute) is used if no value is specified. This value and sasl.login.refresh.buffer.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.	short	60	[0,...,900]	low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.window.factor	Login refresh thread will sleep until the specified window factor relative to the credential's lifetime has been reached, at which time it will try to refresh the credential. Legal values are between 0.5 (50%) and 1.0 (100%) inclusive; a default value of 0.8 (80%) is used if no value is specified. Currently applies only to OAUTHBEARER.	double	0.8	[0.5,...,1.0]	low
sasl.login.refresh.window.jitter	The maximum amount of random jitter relative to the credential's lifetime that is added to the login refresh thread's sleep time. Legal values are between 0 and 0.25 (25%) inclusive; a default value of 0.05 (5%) is used if no value is specified. Currently applies only to OAUTHBEARER.	double	0.05	[0.0,...,0.25]	low

Name	Description	Type	Default	Valid Values	Importance
ssl.cipher.suites	A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol. By default all the available cipher suites are supported.	list	null		low
ssl.endpoint.identification.algorithm	The endpoint identification algorithm to validate server hostname using server certificate.	string	https		low
ssl.keymanager.algorithm	The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.	string	SunX509		low

Name	Description	Type	Default	Valid Values	Importance
ssl.secure.random.implementation	The SecureRandom PRNG implementation to use for SSL cryptography operations.	string	null		low
ssl.trustmanager.algorithm	The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.	string	PKIX		low
transaction.timeout.ms	The maximum amount of time in ms that the transaction coordinator will wait for a transaction status update from the producer before proactively aborting the ongoing transaction. If this value is larger than the transaction.max.timeout.ms setting in the broker, the request will fail with a InvalidTransactionTimeout error.	int	60000		low
	The	string	null		low

transactional.id	TransactionalId Description	Type	Default	non-empty Valid Values	Importance
	transactional delivery. This enables reliability semantics which span multiple producer sessions since it allows the client to guarantee that transactions using the same TransactionalId have been completed prior to starting any new transactions. If no TransactionalId is provided, then the producer is limited to idempotent delivery. Note that enable.idempotence must be enabled if a TransactionalId is configured. The default is null , which means transactions cannot be used. Note that transactions requires a cluster of at least three brokers by default what is the recommended setting for production; for development you can change this, by adjusting				

Name	broker setting Description	Type	Default	Valid Values	Importance
	transaction.state.log.replication.factor				

APPENDIX E. ADMIN CLIENT CONFIGURATION PARAMETERS

Name	Description	Type	Default	Valid Values	Importance
bootstrap.servers	<p>A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping—this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form host1:port1, host2:port2, Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).</p>	list			high

Name	Description	Type	Default	Valid Values	Importance
ssl.key.password	The password of the private key in the key store file. This is optional for client.	password	null		high
ssl.keystore.location	The location of the key store file. This is optional for client and can be used for two-way authentication for client.	string	null		high
ssl.keystore.password	The store password for the key store file. This is optional for client and only needed if ssl.keystore.location is configured.	password	null		high
ssl.truststore.location	The location of the trust store file.	string	null		high
ssl.truststore.password	The password for the trust store file. If a password is not set access to the truststore is still available, but integrity checking is disabled.	password	null		high

Name	Description	Type	Default	Valid Values	Importance
client.id	An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.	string	""		medium
connections.max.idle.ms	Close idle connections after the number of milliseconds specified by this config.	long	300000		medium
receive.buffer.bytes	The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.	int	65536	[-1,...]	medium

Name	Description	Type	Default	Valid Values	Importance
request.timeout.ms	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.	int	120000	[0,...]	medium
sasl.client.callback.handler.class	The fully qualified name of a SASL client callback handler class that implements the AuthenticateCallbackHandler interface.	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.jaas.config	JAAS login context parameters for SASL connections in the format used by JAAS configuration files. JAAS configuration file format is described here . The format for the value is: 'loginModuleClass controlFlag (optionName=optionValue)*;'. For brokers, the config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScamLoginModule required;	password	null		medium
sasl.kerberos.service.name	The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.	string	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.callback.handler.class	The fully qualified name of a SASL login callback handler class that implements the AuthenticateCallbackHandler interface. For brokers, login callback handler config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.class	The fully qualified name of a class that implements the Login interface. For brokers, login config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin	class	null		medium
sasl.mechanism	SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.	string	GSSAPI		medium
security.protocol	Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.	string	PLAINTEXT		medium

Name	Description	Type	Default	Valid Values	Importance
send.buffer.bytes	The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.	int	131072	[-1,...]	medium
ssl.enabled.protocols	The list of protocols enabled for SSL connections.	list	TLSv1.2,TLSv1.1,TLSv1		medium
ssl.keystore.type	The file format of the key store file. This is optional for client.	string	JKS		medium
ssl.protocol	The SSL protocol used to generate the SSLContext. Default setting is TLS, which is fine for most cases. Allowed values in recent JVMs are TLS, TLSv1.1 and TLSv1.2. SSL, SSLv2 and SSLv3 may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities.	string	TLS		medium

Name	Description	Type	Default	Valid Values	Importance
ssl.provider	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.	string	null		medium
ssl.truststore.type	The file format of the trust store file.	string	JKS		medium
metadata.max.age.ms	The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.	long	300000	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
metric.reporters	A list of classes to use as metrics reporters. Implementing the org.apache.kafka.common.metrics.MetricsReporter interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.	list	""		low
metrics.num.samples	The number of samples maintained to compute metrics.	int	2	[1,...]	low
metrics.recording.level	The highest recording level for metrics.	string	INFO	[INFO, DEBUG]	low
metrics.sample.window.ms	The window of time a metrics sample is computed over.	long	30000	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
reconnect.backoff.max.ms	The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.	long	1000	[0,...]	low
reconnect.backoff.ms	The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.	long	50	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
retries	Setting a value greater than zero will cause the client to resend any request that fails with a potentially transient error.	int	5	[0,...]	low
retry.backoff.ms	The amount of time to wait before attempting to retry a failed request. This avoids repeatedly sending requests in a tight loop under some failure scenarios.	long	100	[0,...]	low
sasl.kerberos.kinit.cmd	Kerberos kinit command path.	string	/usr/bin/kinit		low
sasl.kerberos.min.time.before.relogin	Login thread sleep time between refresh attempts.	long	60000		low
sasl.kerberos.ticket.renew.jitter	Percentage of random jitter added to the renewal time.	double	0.05		low

Name	Description	Type	Default	Valid Values	Importance
sasl.kerberos.ticket.renew.window.factor	Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.	double	0.8		low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.buffer.seconds	The amount of buffer time before credential expiration to maintain when refreshing a credential, in seconds. If a refresh would otherwise occur closer to expiration than the number of buffer seconds then the refresh will be moved up to maintain as much of the buffer time as possible. Legal values are between 0 and 3600 (1 hour); a default value of 300 (5 minutes) is used if no value is specified. This value and sasl.login.refresh.min.period.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.	short	300	[0,...,3600]	low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.min.period.seconds	The desired minimum time for the login refresh thread to wait before refreshing a credential, in seconds. Legal values are between 0 and 900 (15 minutes); a default value of 60 (1 minute) is used if no value is specified. This value and sasl.login.refresh.buffer.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.	short	60	[0,...,900]	low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.window.factor	Login refresh thread will sleep until the specified window factor relative to the credential's lifetime has been reached, at which time it will try to refresh the credential. Legal values are between 0.5 (50%) and 1.0 (100%) inclusive; a default value of 0.8 (80%) is used if no value is specified. Currently applies only to OAUTHBEARER.	double	0.8	[0.5,...,1.0]	low
sasl.login.refresh.window.jitter	The maximum amount of random jitter relative to the credential's lifetime that is added to the login refresh thread's sleep time. Legal values are between 0 and 0.25 (25%) inclusive; a default value of 0.05 (5%) is used if no value is specified. Currently applies only to OAUTHBEARER.	double	0.05	[0.0,...,0.25]	low

Name	Description	Type	Default	Valid Values	Importance
ssl.cipher.suites	A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol. By default all the available cipher suites are supported.	list	null		low
ssl.endpoint.identification.algorithm	The endpoint identification algorithm to validate server hostname using server certificate.	string	https		low
ssl.keymanager.algorithm	The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.	string	SunX509		low

Name	Description	Type	Default	Valid Values	Importance
ssl.secure.random.implementation	The SecureRandom PRNG implementation to use for SSL cryptography operations.	string	null		low
ssl.trustmanager.algorithm	The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.	string	PKIX		low

APPENDIX F. KAFKA CONNECT CONFIGURATION PARAMETERS

Name	Description	Type	Default	Valid Values	Importance
config.storage.topic	The name of the Kafka topic where connector configurations are stored	string			high
group.id	A unique string that identifies the Connect cluster group this worker belongs to.	string			high
key.converter	Converter class used to convert between Kafka Connect format and the serialized form that is written to Kafka. This controls the format of the keys in messages written to or read from Kafka, and since this is independent of connectors it allows any connector to work with any serialization format. Examples of common formats include JSON and Avro.	class			high

Name	Description	Type	Default	Valid Values	Importance
offset.storage.topic	The name of the Kafka topic where connector offsets are stored	string			high
status.storage.topic	The name of the Kafka topic where connector and task status are stored	string			high
value.converter	Converter class used to convert between Kafka Connect format and the serialized form that is written to Kafka. This controls the format of the values in messages written to or read from Kafka, and since this is independent of connectors it allows any connector to work with any serialization format. Examples of common formats include JSON and Avro.	class			high

Name	Description	Type	Default	Valid Values	Importance
bootstrap.servers	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping —this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form host1:port 1, host2:port 2, Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).	list	localhost:9092		high

Name	Description	Type	Default	Valid Values	Importance
heartbeat.interval.ms	The expected time between heartbeats to the group coordinator when using Kafka's group management facilities. Heartbeats are used to ensure that the worker's session stays active and to facilitate rebalancing when new members join or leave the group. The value must be set lower than session.timeout.ms , but typically should be set no higher than 1/3 of that value. It can be adjusted even lower to control the expected time for normal rebalances.	int	3000		high

Name	Description	Type	Default	Valid Values	Importance
rebalance.timeout.ms	The maximum allowed time for each worker to join the group once a rebalance has begun. This is basically a limit on the amount of time needed for all tasks to flush any pending data and commit offsets. If the timeout is exceeded, then the worker will be removed from the group, which will cause offset commit failures.	int	60000		high

Name	Description	Type	Default	Valid Values	Importance
session.timeout.ms	<p>The timeout used to detect worker failures. The worker sends periodic heartbeats to indicate its liveness to the broker. If no heartbeats are received by the broker before the expiration of this session timeout, then the broker will remove the worker from the group and initiate a rebalance. Note that the value must be in the allowable range as configured in the broker configuration by group.min.session.timeout.ms and group.max.session.timeout.ms.</p>	int	10000		high
ssl.key.password	<p>The password of the private key in the key store file. This is optional for client.</p>	password	null		high

Name	Description	Type	Default	Valid Values	Importance
ssl.keystore.location	The location of the key store file. This is optional for client and can be used for two-way authentication for client.	string	null		high
ssl.keystore.password	The store password for the key store file. This is optional for client and only needed if ssl.keystore.location is configured.	password	null		high
ssl.truststore.location	The location of the trust store file.	string	null		high
ssl.truststore.password	The password for the trust store file. If a password is not set access to the truststore is still available, but integrity checking is disabled.	password	null		high
connections.max.idle.ms	Close idle connections after the number of milliseconds specified by this config.	long	540000		medium

Name	Description	Type	Default	Valid Values	Importance
receive.buffer.bytes	The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.	int	32768	[0,...]	medium
request.timeout.ms	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.	int	40000	[0,...]	medium
sasl.client.callback.handler.class	The fully qualified name of a SASL client callback handler class that implements the AuthenticateCallbackHandler interface.	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.jaas.config	JAAS login context parameters for SASL connections in the format used by JAAS configuration files. JAAS configuration file format is described here . The format for the value is: 'loginModuleClass controlFlag (optionName=optionValue)*;'. For brokers, the config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScrumLoginModule required;	password	null		medium
sasl.kerberos.service.name	The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.	string	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.callback.handler.class	The fully qualified name of a SASL login callback handler class that implements the AuthenticateCallbackHandler interface. For brokers, login callback handler config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler	class	null		medium

Name	Description	Type	Default	Valid Values	Importance
sasl.login.class	The fully qualified name of a class that implements the Login interface. For brokers, login config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin	class	null		medium
sasl.mechanism	SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.	string	GSSAPI		medium
security.protocol	Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.	string	PLAINTEXT		medium

Name	Description	Type	Default	Valid Values	Importance
send.buffer.bytes	The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.	int	131072	[0,...]	medium
ssl.enabled.protocols	The list of protocols enabled for SSL connections.	list	TLSv1.2,TLSv1.1,TLSv1		medium
ssl.keystore.type	The file format of the key store file. This is optional for client.	string	JKS		medium
ssl.protocol	The SSL protocol used to generate the SSLContext. Default setting is TLS, which is fine for most cases. Allowed values in recent JVMs are TLS, TLSv1.1 and TLSv1.2. SSL, SSLv2 and SSLv3 may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities.	string	TLS		medium

Name	Description	Type	Default	Valid Values	Importance
ssl.provider	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.	string	null		medium
ssl.truststore.type	The file format of the trust store file.	string	JKS		medium
worker.sync.timeout.ms	When the worker is out of sync with other workers and needs to resynchronize configurations, wait up to this amount of time before giving up, leaving the group, and waiting a backoff period before rejoining.	int	3000		medium
worker.uncsync.backoff.ms	When the worker is out of sync with other workers and fails to catch up within <code>worker.sync.timeout.ms</code> , leave the Connect cluster for this long before rejoining.	int	300000		medium

Name	Description	Type	Default	Valid Values	Importance
access.control.allow.methods	Sets the methods supported for cross origin requests by setting the Access-Control-Allow-Methods header. The default value of the Access-Control-Allow-Methods header allows cross origin requests for GET, POST and HEAD.	string	""		low
access.control.allow.origin	Value to set the Access-Control-Allow-Origin header to for REST API requests. To enable cross origin access, set this to the domain of the application that should be permitted to access the API, or '*' to allow access from any domain. The default value only allows access from the domain of the REST API.	string	""		low

Name	Description	Type	Default	Valid Values	Importance
client.id	An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.	string	""		low
config.providers	Comma-separated names of ConfigProvider classes, loaded and used in the order specified. Implementing the interface ConfigProvider allows you to replace variable references in connector configurations, such as for externalized secrets.	list	""		low
config.storage.replication.factor	Replication factor used when creating the configuration storage topic	short	3	[1,...]	low

Name	Description	Type	Default	Valid Values	Importance
header.converter	HeaderConverter class used to convert between Kafka Connect format and the serialized form that is written to Kafka. This controls the format of the header values in messages written to or read from Kafka, and since this is independent of connectors it allows any connector to work with any serialization format. Examples of common formats include JSON and Avro. By default, the SimpleHeaderConverter is used to serialize header values to strings and deserialize them by inferring the schemas.	class	org.apache.kafka.connect.storage.SimpleHeaderConverter		low

Name	Description	Type	Default	Valid Values	Importance
internal.key.converter	<p>Converter class used to convert between Kafka Connect format and the serialized form that is written to Kafka. This controls the format of the keys in messages written to or read from Kafka, and since this is independent of connectors it allows any connector to work with any serialization format. Examples of common formats include JSON and Avro. This setting controls the format used for internal bookkeeping data used by the framework, such as configs and offsets, so users can typically use any functioning Converter implementation. Deprecated; will be removed in an upcoming version.</p>	class	org.apache.kafka.connect.json.JsonConverter		low

Name	Description	Type	Default	Valid Values	Importance
internal.value.converter	<p>Converter class used to convert between Kafka Connect format and the serialized form that is written to Kafka. This controls the format of the values in messages written to or read from Kafka, and since this is independent of connectors it allows any connector to work with any serialization format. Examples of common formats include JSON and Avro. This setting controls the format used for internal bookkeeping data used by the framework, such as configs and offsets, so users can typically use any functioning Converter implementation . Deprecated; will be removed in an upcoming version.</p>	class	org.apache.kafka.connect.json.JsonConverter		low

Name	Description	Type	Default	Valid Values	Importance
listeners	List of comma-separated URIs the REST API will listen on. The supported protocols are HTTP and HTTPS. Specify hostname as 0.0.0.0 to bind to all interfaces. Leave hostname empty to bind to default interface. Examples of legal listener lists: HTTP://myhost:8083,HTTPS://myhost:8084	list	null		low
metadata.max.age.ms	The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.	long	300000	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
metric.reporters	A list of classes to use as metrics reporters. Implementing the org.apache.kafka.common.metrics.MetricsReporter interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.	list	""		low
metrics.num.samples	The number of samples maintained to compute metrics.	int	2	[1,...]	low
metrics.recording.level	The highest recording level for metrics.	string	INFO	[INFO, DEBUG]	low
metrics.sample.window.ms	The window of time a metrics sample is computed over.	long	30000	[0,...]	low
offset.flush.interval.ms	Interval at which to try committing offsets for tasks.	long	60000		low

Name	Description	Type	Default	Valid Values	Importance
offset.flush.timeout.ms	Maximum number of milliseconds to wait for records to flush and partition offset data to be committed to offset storage before cancelling the process and restoring the offset data to be committed in a future attempt.	long	5000		low
offset.storage.partitions	The number of partitions used when creating the offset storage topic	int	25	[1,...]	low
offset.storage.replication.factor	Replication factor used when creating the offset storage topic	short	3	[1,...]	low

Name	Description	Type	Default	Valid Values	Importance
plugin.path	<p>List of paths separated by commas (,) that contain plugins (connectors, converters, transformations). The list should consist of top level directories that include any combination of:</p> <ul style="list-style-type: none"> a) directories immediately containing jars with plugins and their dependencies b) uber-jars with plugins and their dependencies c) directories immediately containing the package directory structure of classes of plugins and their dependencies <p>Note: symlinks will be followed to discover dependencies or plugins.</p> <p>Examples: plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/connectors</p>	list	null		low

Name	Description	Type	Default	Valid Values	Importance
reconnect.backoff.max.ms	The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.	long	1000	[0,...]	low
reconnect.backoff.ms	The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.	long	50	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
rest.advertised.host.name	If this is set, this is the hostname that will be given out to other workers to connect to.	string	null		low
rest.advertised.listener	Sets the advertised listener (HTTP or HTTPS) which will be given to other workers to use.	string	null		low
rest.advertised.port	If this is set, this is the port that will be given out to other workers to connect to.	int	null		low
rest.extension.classes	Comma-separated names of ConnectRestExtension classes, loaded and called in the order specified. Implementing the interface ConnectRestExtension allows you to inject into Connect's REST API user defined resources like filters. Typically used to add custom capability like logging, security, etc.	list	""		low

Name	Description	Type	Default	Valid Values	Importance
rest.host.name	Hostname for the REST API. If this is set, it will only bind to this interface.	string	null		low
rest.port	Port for the REST API to listen on.	int	8083		low
retry.backoff.ms	The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.	long	100	[0,...]	low
sasl.kerberos.kinit.cmd	Kerberos kinit command path.	string	/usr/bin/kinit		low
sasl.kerberos.min.time.before.relogin	Login thread sleep time between refresh attempts.	long	60000		low
sasl.kerberos.ticket.renew.jitter	Percentage of random jitter added to the renewal time.	double	0.05		low

Name	Description	Type	Default	Valid Values	Importance
sasl.kerberos.ticket.renew.window.factor	Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.	double	0.8		low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.buffer.seconds	The amount of buffer time before credential expiration to maintain when refreshing a credential, in seconds. If a refresh would otherwise occur closer to expiration than the number of buffer seconds then the refresh will be moved up to maintain as much of the buffer time as possible. Legal values are between 0 and 3600 (1 hour); a default value of 300 (5 minutes) is used if no value is specified. This value and sasl.login.refresh.min.period.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.	short	300	[0,...,3600]	low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.min.period.seconds	The desired minimum time for the login refresh thread to wait before refreshing a credential, in seconds. Legal values are between 0 and 900 (15 minutes); a default value of 60 (1 minute) is used if no value is specified. This value and sasl.login.refresh.buffer.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.	short	60	[0,...,900]	low

Name	Description	Type	Default	Valid Values	Importance
sasl.login.refresh.window.factor	Login refresh thread will sleep until the specified window factor relative to the credential's lifetime has been reached, at which time it will try to refresh the credential. Legal values are between 0.5 (50%) and 1.0 (100%) inclusive; a default value of 0.8 (80%) is used if no value is specified. Currently applies only to OAUTHBEARER.	double	0.8	[0.5,...,1.0]	low
sasl.login.refresh.window.jitter	The maximum amount of random jitter relative to the credential's lifetime that is added to the login refresh thread's sleep time. Legal values are between 0 and 0.25 (25%) inclusive; a default value of 0.05 (5%) is used if no value is specified. Currently applies only to OAUTHBEARER.	double	0.05	[0.0,...,0.25]	low

Name	Description	Type	Default	Valid Values	Importance
ssl.cipher.suites	A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol. By default all the available cipher suites are supported.	list	null		low
ssl.client.auth	<p>Configures kafka broker to request client authentication. The following settings are common:</p> <ul style="list-style-type: none"> • ssl.client.auth=required If set to required client authentication is required. • ssl.client.auth=requested This mean 	string	none		low

Name	Description	Type	Default	Valid Values	Importance
	<p>SSL client authentication is optional. Unlike requested, if this option is set client can choose not to provide authentication information about itself</p> <ul style="list-style-type: none"> • ssl.client.auth=none This means client authentication is not needed. 				
ssl.endpoint.identification.algorithm	The endpoint identification algorithm to validate server hostname using server certificate.	string	https		low

Name	Description	Type	Default	Valid Values	Importance
ssl.keymanager.algorithm	The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.	string	SunX509		low
ssl.secure.random.implementation	The SecureRandom PRNG implementation to use for SSL cryptography operations.	string	null		low
ssl.trustmanager.algorithm	The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.	string	PKIX		low
status.storage.partitions	The number of partitions used when creating the status storage topic	int	5	[1,...]	low
status.storage.replication.factor	Replication factor used when creating the status storage topic	short	3	[1,...]	low

Name	Description	Type	Default	Valid Values	Importance
task.shutdown.graceful.timeout.ms	Amount of time to wait for tasks to shutdown gracefully. This is the total amount of time, not per task. All task have shutdown triggered, then they are waited on sequentially.	long	5000		low

APPENDIX G. KAFKA STREAMS CONFIGURATION PARAMETERS

Name	Description	Type	Default	Valid Values	Importance
application.id	An identifier for the stream processing application. Must be unique within the Kafka cluster. It is used as 1) the default client-id prefix, 2) the group-id for membership management, 3) the changelog topic prefix.	string			high

Name	Description	Type	Default	Valid Values	Importance
bootstrap.servers	<p>A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping —this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form host1:port 1, host2:port 2, Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).</p>	list			high

Name	Description	Type	Default	Valid Values	Importance
replication.factor	The replication factor for change log topics and repartition topics created by the stream processing application.	int	1		high
state.dir	Directory location for state store.	string	/tmp/kafka-streams		high
cache.max.bytes.buffering	Maximum number of memory bytes to be used for buffering across all threads	long	10485760	[0,...]	medium
client.id	An ID prefix string used for the client IDs of internal consumer, producer and restore-consumer, with pattern '<client.id>-StreamThread-<threadSequenceNumber>-<consumer producer restore-consumer>'.	string	""		medium
default.deserialization.exception.handler	Exception handling class that implements the org.apache.kafka.streams.errors.DeserializationExceptionHandler interface.	class	org.apache.kafka.streams.errors.LogAndFailExceptionHandler		medium

Name	Description	Type	Default	Valid Values	Importance
default.key.serde	Default serializer / deserializer class for key that implements the org.apache.kafka.common.serialization.Serde interface. Note when windowed serde class is used, one needs to set the inner serde class that implements the org.apache.kafka.common.serialization.Serde interface via 'default.windowed.key.serde.inner' or 'default.windowed.value.serde.inner' as well	class	org.apache.kafka.common.serialization.Serde\$ByteArraySerde		medium
default.production.exception.handler	Exception handling class that implements the org.apache.kafka.streams.errors.ProductionsExceptionHandler interface.	class	org.apache.kafka.streams.errors.DefaultProductionExceptionHandler		medium

Name	Description	Type	Default	Valid Values	Importance
default.timestamp.extractor	Default timestamp extractor class that implements the org.apache.kafka.streams.processor.TimestampExtractor interface.	class	org.apache.kafka.streams.processor.FailOnInvalidTimestamp		medium
default.value.serde	Default serializer / deserializer class for value that implements the org.apache.kafka.common.serialization.Serde interface. Note when windowed serde class is used, one needs to set the inner serde class that implements the org.apache.kafka.common.serialization.Serde interface via 'default.windowed.key.serde.inner' or 'default.windowed.value.serde.inner' as well	class	org.apache.kafka.common.serialization.Serdes\$ByteArraySerde		medium
num.standby.replicas	The number of standby replicas for each task.	int	0		medium

Name	Description	Type	Default	Valid Values	Importance
num.stream.threads	The number of threads to execute stream processing.	int	1		medium
processing.guarantee	The processing guarantee that should be used. Possible values are at_least_once (default) and exactly_once . Note that exactly-once processing requires a cluster of at least three brokers by default what is the recommended setting for production; for development you can change this, by adjusting broker setting transaction.state.log.replication.factor .	string	at_least_once	[at_least_once, exactly_once]	medium
security.protocol	Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.	string	PLAINTEXT		medium

Name	Description	Type	Default	Valid Values	Importance
topology.optimization	A configuration telling Kafka Streams if it should optimize the topology, disabled by default	string	none	[none, all]	medium
application.server	A host:port pair pointing to an embedded user defined endpoint that can be used for discovering the locations of state stores within a single KafkaStreams application	string	""		low
buffered.records.per.partition	The maximum number of records to buffer per partition.	int	1000		low
commit.interval.ms	The frequency with which to save the position of the processor. (Note, if 'processing.guarantee' is set to 'exactly_once', the default value is 100, otherwise the default value is 30000.	long	30000		low

Name	Description	Type	Default	Valid Values	Importance
connections.max.idle.ms	Close idle connections after the number of milliseconds specified by this config.	long	540000		low
metadata.max.age.ms	The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.	long	300000	[0,...]	low
metric.reporters	A list of classes to use as metrics reporters. Implementing the org.apache.kafka.common.metrics.MetricsReporter interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.	list	""		low

Name	Description	Type	Default	Valid Values	Importance
metrics.num.samples	The number of samples maintained to compute metrics.	int	2	[1,...]	low
metrics.recording.level	The highest recording level for metrics.	string	INFO	[INFO, DEBUG]	low
metrics.sample.window.ms	The window of time a metrics sample is computed over.	long	30000	[0,...]	low
partition.grouper	Partition grouper class that implements the org.apache.kafka.streams.processor.PartitionGrouper interface.	class	org.apache.kafka.streams.processor.DefaultPartitionGrouper		low
poll.ms	The amount of time in milliseconds to block waiting for input.	long	100		low
receive.buffer.bytes	The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.	int	32768	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
reconnect.backoff.max.ms	The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.	long	1000	[0,...]	low
reconnect.backoff.ms	The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.	long	50	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
request.timeout.ms	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.	int	40000	[0,...]	low
retries	Setting a value greater than zero will cause the client to resend any request that fails with a potentially transient error.	int	0	[0,...,2147483647]	low
retry.backoff.ms	The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.	long	100	[0,...]	low

Name	Description	Type	Default	Valid Values	Importance
rocksdb.config.setter	A Rocks DB config setter class or class name that implements the org.apache.kafka.streams.state.RocksDBConfigSetter interface	class	null		low
send.buffer.bytes	The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.	int	131072	[0,...]	low
state.cleanup.delay.ms	The amount of time in milliseconds to wait before deleting state when a partition has migrated. Only state directories that have not been modified for at least state.cleanup.delay.ms will be removed	long	600000		low

Name	Description	Type	Default	Valid Values	Importance
upgrade.from	Allows upgrading from versions 0.10.0/0.10.1/0.10.2/0.11.0/1.0/1.1 to version 1.2 (or newer) in a backward compatible way. When upgrading from 1.2 to a newer version it is not required to specify this config. Default is null. Accepted values are "0.10.0", "0.10.1", "0.10.2", "0.11.0", "1.0", "1.1" (for upgrading from the corresponding old version).	string	null	[null, 0.10.0, 0.10.1, 0.10.2, 0.11.0, 1.0, 1.1]	low
windowstore.changelog.addition.al.retention.ms	Added to a windows maintainMs to ensure data is not deleted from the log prematurely. Allows for clock drift. Default is 1 day	long	86400000		low