

## Integrating with Apache Kafka

Date published: 2019-04-15

Date modified: 2024-07-23



# Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

# Contents

<b>Integration with Apache Kafka.....</b>	<b>4</b>
Configuring Kafka for integration with EFM.....	4
Configuring the EFM properties.....	4

## Integration with Apache Kafka

You can integrate Edge Flow Manager (EFM) with Apache Kafka and forward agent heartbeats to defined Kafka topics. Learn how to perform the integration with Apache Kafka.

To integrate EFM with Kafka, you need to configure Kafka and EFM properties.

EFM supports the forwarding of agent heartbeats and acknowledges messages exchanged on the C2 protocol between the EFM server and MiNiFi agents. You can also enable two-way TLS or authentication using SASL\_SSL for secure communication between EFM and Kafka brokers.

The integration is not on critical path so, if there is any communication issue between the server and the Kafka broker(s), it results in error logs but not in core functionality degradation. When connection between EFM and Kafka is restored, the accumulated messages are forwarded. You can fine tune the buffering related settings.

## Configuring Kafka for integration with EFM

Configuring Apache Kafka to work seamlessly with Edge Flow Manager (EFM) involves creating Kafka topics that receive the heartbeats and acknowledgments from MiNiFi agents.

### Before you begin

Ensure that you are using Apache Kafka version 2.5 or above. For more information about Kafka and stream processing setup, see the *Stream Processing documentation*.

### Procedure

1. Create a Kafka topic for receiving heartbeats.

To receive heartbeats from MiNiFi agents, you need to create a Kafka topic dedicated to this purpose. For example, you can create a Kafka topic named heartbeat.

2. Create a Kafka topic for receiving acknowledgements.

Similar to heartbeats, acknowledgments from MiNiFi agents require a separate Kafka topic. For example, you can create a Kafka topic called ack.

3. If you wish to enhance the security of your Kafka deployment by enabling two-way TLS or authentication using SASL\_SSL, configure the specific Kafka server properties accordingly.

For more information about enabling TLS, see the [Confluent Security Tutorial](#).

### Related Information

[Stream Processing documentation](#)

## Configuring the EFM properties

To integrate EFM with Kafka, you need to configure the EFM properties in the `efm.properties` file.

### Before you begin

You have created Kafka topics to receive heartbeats and acknowledgements from MiNiFi agents.

## Procedure

1. Set the following property to true:

```
efm.heartbeat.kafka.publishEnabled=true
```

If you do not set it to true, none of the configurations are considered.

2. Set the following property to provide comma separated addresses for brokers if Kafka runs in a cluster:

```
efm.heartbeat.kafka.brokerAddress=localhost:9093
```

3. Set the ID that shows up as client on Kafka side logs.

```
efm.heartbeat.kafka.clientId=efm
```

4. Provide topic names where the heartbeat and acknowledgement messages are sent to respectively.

```
efm.heartbeat.kafka.heartbeatTopicName=heartbeat
efm.heartbeat.kafka.ackTopicName=ack
```

The topic names must match the topics previously created on Kafka side.

5. Optional. Set the following Kafka parameters:

```
efm.heartbeat.kafka.retryBackoff=1000
efm.heartbeat.kafka.reconnectBackoff=1000
efm.heartbeat.kafka.reconnectBackoffMax=60000
efm.heartbeat.kafka.requestTimeout=30000
efm.heartbeat.kafka.bufferMemory=33554432
efm.heartbeat.kafka.batchSize=16384
efm.heartbeat.kafka.compressionType=gzip
efm.heartbeat.kafka.deliveryTimeout=120000
efm.heartbeat.kafka.maxRequestSize=1048576
efm.heartbeat.kafka.acks=1
```

For more details on Kafka parameters, see <https://kafka.apache.org/documentation/>.

6. Optional. Set the following properties if two-way TLS is enabled:

```
efm.heartbeat.kafka.ssl.enabled=true
efm.heartbeat.kafka.ssl.keystoreLocation=
efm.heartbeat.kafka.ssl.keystorePassword=
efm.heartbeat.kafka.ssl.keyPassword=
efm.heartbeat.kafka.ssl.truststoreLocation=
efm.heartbeat.kafka.ssl.truststorePassword=
efm.heartbeat.kafka.ssl.securityProtocol=SSL
efm.heartbeat.kafka.ssl.enabledProtocols=
```

7. Optional. Set the following properties if SASL\_SSL authentication is enabled:

```
efm.heartbeat.kafka.ssl.enabled=true
efm.heartbeat.kafka.ssl.truststoreLocation=
efm.heartbeat.kafka.ssl.truststorePassword=
efm.heartbeat.kafka.ssl.securityProtocol=SASL_SSL
efm.heartbeat.kafka.ssl.saslSslUsername=client
efm.heartbeat.kafka.ssl.saslSslPassword=client-secret
```