Apache Kafka - Configuration Guide

Content

Related Content

- Apache Kafka - Configuration Guide

Created 24 Mar 2020

Updated 21 Oct 2020

Summary

This document will help users who wish to use Apache Kafka in conjunction with z/IRIS.

Purpose of this document

This document will assist users who want to:

- Configure their Kafka Cluster for z/IRIS.
- Configure and use recommended settings for the Kafka Producer (z/IRIS z/OS Client).
- Configure and use recommended settings for the Kafka Consumer (z/IRIS IronTap).

Kafka Cluster

Architecture

We suggest to use an odd number of Zookeeper Instances (at least 3), at least 3 Kafka Brokers with a Replication Factor of 3 which are deployed on (physically) separate machines to exploit the Replication Factor effectively and to ensure High Availability.

Topic

For the utilisation of z/IRIS a Kafka Topic is needed to enable producing and consuming of SMF data. For determining a suitable amount of Partitions for the Topic the following should be considered:

- The amount of Partitions should be a multiple of the number of Kafka Brokers to achieve proper Load Balancing between them.
- More Partitions increases parallelism, throughput, the possibility to leverage more Kafka Brokers in a Cluster and finally scale capabilities, since a Partition can only be assigned to one Consumer in a Consumer Group.
- Less Partitions means that less files are opened on the Kafka Cluster and Zookeeper has less work to manage it.

Taking these remarks into account we recommend to use around ~15 Partitions for the Topic. Considering all the mentioned aspects a suitable Topic named **smf** can be created with the help of the Kafka CLI by:

Example

```
kafka-topics.sh --bootstrap-server <kafka-ip>:<kafka-port> --topic smf --partitions 15 --replication-
factor 3 --create
```

In addition to optionally further reduce the risk of data loss we suggest to use 2 In-Sync Replicas which can be configured on topic-level after creating the Topic **smf** by:

Example

```
kafka-configs.sh --zookeeper <zookeeper-ip>:<zookeeper-port> --entity-type topics --entity-name smf --add-
config min.insync.replicas=2 --alter
```

Quotas

When z/IRIS is set up on a Kafka Cluster we highly recommend to use Apache Kafka's feature named Quotas to protect against the possibility of monopolizing resources and causing network saturation. In order to do that Kafka needs the capability to identify the z/IRIS IronTap (Kafka Consumer) in the Kafka Cluster. This can be done by setting an unique, logical application name (e.g. smf-consumer) in the file **application.conf** through the configuration property **client.id**. Afterwards Apache Kafka's Quotas can be set dynamically via the Kafka CLI. We recommend to limit the network bandwidth to around ~50 MB/s for z/IRIS IronTap:

Example

```
kafka-configs.sh --zookeeper <zookeeper-ip>:<zookeeper-port> --entity-type clients --entity-name <client-
id> --add-config 'consumer_byte_rate=52428800' --alter
```

Kafka Producer (z/IRIS z/OS Client)

Kafka Mode Requirements

To stream SMF record via a z/IRIS Kafka Producer on z/OS, the following properties must be configured in the file **$APPHOME/lib/ziris.comm.config**:

ⓘ For more information regarding the file **ziris.comm.config** and other required parameters, please refer to the Latest Administration and User Guide.

Required ziris.comm.config settings for Kafka streaming

```
mode=kafka
# kafkaTopic defines the Kafka Cluster Topic used to stream SMF records.
# Required if mode=kafka
# syntax: kafkaTopic=kafka_topic_name
kafkaTopic=ziris_prod_smf
# host refers to the host address destination for SMF record streaming.
# If mode=kafka, host is a list of Kafka Bootstrap Servers that will overwrite
# any bootstrap servers configured in the producer.properties file linked by kafkaConfig
# syntax: host=kafka_bootstrap_server:port
host=kafka_bootstrap_server:port[,kafka_bootstrap_server:port]
# kafkaConfig is an optional parameter that configures an alternate
# Kafka producer.properties file to the z/IRIS z/OS default.
# default: $APP_HOME/kafka/default-ziris-producer.properties
# syntax: kafkaConfig=/producer/properties/file
# kafkaConfig=/usr/lpp/ziris/ziris-producer.properties
```

The configuration file **$APP_HOME/kafka/default-ziris-producer.properties** does not configure the property bootstrap.servers. As a quick start, users can simply list Kafka Bootstrap Servers using the host parameter in **ziris.comm.config**.

Default Kafka Producer Properties

The default configuration file producer.properties for z/IRIS z/OS Clients can be located in the directory **$APP_HOME/kafka**. This file is used by the z/IRIS z/OS Clients and can be changed by specifying a file path and file name using the kafkaConfig parameter in the configuration file **$APP_HOME/lib/ziris.comm.config**.

default-ziris.producer.properties

```
####################################################################################
############################ DEFAULT PRODUCER CONFIGURATION ########################
############################# for Apache Kafka 2.3.0 ###############################
################################ z/IRIS z/OS Client ################################

# A list of host/port pairs to use for establishing the initial connection to the
# Kafka Cluster.
### NOTE: Kafka Cluster address needs to be adjusted.
#bootstrap.servers=localhost:9092

# Ensures that no duplicate messages will be introduced into Kafka due to network
# errors.
enable.idempotence=true

##################### Implied settings by enable.idempotence #####################
# Producer can choose between different required acknowledgements of data writes
# for data acquisition confirmation. When acks=0, the producer doesn't request any
# response from the Kafka Cluster. When acks=1, the producer requests an
# acknowledgement only from the leader. When acks=all, the producer requests
# acknowledgements from the leader and the corresponding replicas. The default
# setting is acks=1, however enable.idempotence requires acks=all.
acks=all

# Producer will automatically retry to send messages. The default and recommended
# setting is 2147483647 since Kafka 2.1.0. When enable.idempotence=true, it
# has to be a value > 0, otherwise an exception will be thrown.
retries=2147483647

# Restricts how many unacknowledged requests can be made by a producer parallel on
# a single connection before blocking. When enable.idempotence=true, then the
# default setting is 5, otherwise it can be a value between 1<=x<=5 where message
# ordering is ensured as well.
max.in.flight.requests.per.connection=5
```

```
##################################################################################

# In general enabled compression would reduce latency because of allowing faster
# data transfers, quicker replication across Kafka Brokers and better Kafka disk
# utilisation due to less data sizes at the expense of CPU cycles. It becomes more
# effective the bigger the batch of messages being sent to Kafka due to a higher
# compression ratio.
compression.type=none

# The total bytes of memory the producer can use to buffer records waiting to be
# sent to the server. The default setting is 33554432 bytes (32 MB).
buffer.memory=33554432

# When the producer produces faster than the Kafka Broker can process the data
# at the moment in time or is temporarily down, then data is going to be buffered
# in the buffer memory. If it fills completely up, then the producer will start to
# block by not producing new data while waiting. The producer will wait a certain
# amount of time until it throws a timeout exception. The default setting for this
# is 60000 ms (1 min).
max.block.ms=60000

# Determines the maximum number of bytes that can be included in a batch. Any
# message that is bigger than the batch size will be sent as soon as possible
# regardless of linger.ms and thus not be batched. The default setting is 16384
# (16 KB). Don't set the batch size too high, since a batch is allocated per
# partition which could possibly lead to wasting or even outrunning of memory.
batch.size=65536

# Describes the maximum size of a request in bytes. This setting will limit the
# number of message batches the producer will send in a single request to avoid
# sending huge requests. Thus it is also effectively a cap on the maximum record
# batch size. The default setting is 1048576 (1 MB).
max.request.size=1048576

# Producer will wait a small amount of time before sending. At the expense of
# introducing a small delay the chance of messages being sent together in a batch
# can be increased. The default setting is 0 ms.
linger.ms=15

# Close idle connections. The default setting is 540000 ms (9 mins).
connections.max.idle.ms=30000

# For a certain amount of time the producer will continue to retry to send a
# message. After that the producer will stop sending the message. When a message
# can't be acknowledged in this given time frame, then the message sending counts
# as failed. The value should be >= linger.ms + request.timeout.ms. The default
# setting is 120000 ms (2 mins).
delivery.timeout.ms=120000

# Controls the maximum amount of time the producer will wait for Kafka Leader's
# acknowledgement of a sent message. If the acknowledgement is not received in
# time, then the message will be sent again depending on retries and
# delivery.timeout.ms. The default setting is 30000 ms (30 s).
request.timeout.ms=30000

# Describes the amount of time to wait before attempting to retry a failed
# request to a given topic partition. The default setting is 100 ms.
retry.backoff.ms=100

# Client ID in form of a string to pass to the server when making requests. The
# purpose 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. The default setting is "".
client.id=""

# The size of the TCP receive buffer to use when reading data. If the value is
# -1, the OS default will be used. The default setting is 32768 (32 KB).
receive.buffer.bytes=32768

# The size of the TCP send buffer to use when sending data. If the value is
# -1, the OS default will be used. The default setting is 131072 (128 KB).
```

```
send.buffer.bytes=131072

# The period of time after which a refresh of metadata is forced even if there
# haven't been seen any partition leadership changes to proactively discover any
# new Kafka Brokers or Partitions. The default setting is 300000 ms (5 mins).
metadata.max.age.ms=300000

# Determines the maximum amount of time to wait when reconnection to a Kafka
# 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. The default setting is 1000 ms (1 s).
reconnect.backoff.max.ms=1000

# The base amount of time to wait before attempting to reconnect to a given host.
# The default setting is 50 ms.
reconnect.backoff.ms=50
```

Required z/IRIS Kafka Producer Properties

The following properties are required for all z/IRIS z/OS Clients where the Kafka Mode "mode=kafka" is set, thus activated and should be present in all alternate producer.properties configurations used by z/IRIS z/OS Clients. Failure to configure these properties may result in erroneous behaviour, reduced reliability and/or reduced performance:

Property	Value	Requirement description
`enable.idempotence`	`true`	The order in which SMF records are read and sent must be maintained to achieve correlation of data within SMF records. Enforcing the order of records sent via the Kafka Producer used by the z/IRIS z/OS Client is achieved by setting the Producer configuration `enable.idempotence=true`.

Kafka Producer (z/IRIS z/OS Client)

Kafka Mode Requirements

To stream SMF record via a z/IRIS Kafka Producer on z/OS, the following properties must be configured in the file **$APPHOME/lib/ziris.comm.config**:

For more information regarding the file **ziris.comm.config** and other required parameters, please refer to the z/IRIS - Latest Administration and User Guide.

Required ziris.comm.config settings for Kafka streaming

```
mode=kafka
# kafkaTopic defines the Kafka Cluster Topic used to stream SMF records.
# Required if mode=kafka
# syntax: kafkaTopic=kafka_topic_name
kafkaTopic=ziris_prod_smf
# host refers to the host address destination for SMF record streaming.
# If mode=kafka, host is a list of Kafka Bootstrap Servers that will overwrite
# any bootstrap servers configured in the producer.properties file linked by kafkaConfig
# syntax: host=kafka_bootstrap_server:port
host=kafka_bootstrap_server:port[,kafka_bootstrap_server:port]
# kafkaConfig is an optional parameter that configures an alternate
# Kafka producer.properties file to the z/IRIS z/OS default.
# default: $APP_HOME/kafka/default-ziris-producer.properties
# syntax: kafkaConfig=/producer/properties/file
# kafkaConfig=/usr/lpp/ziris/ziris-producer.properties
```

The configuration file **$APP_HOME/kafka/default-ziris-producer.properties** does not configure the property bootstrap.servers. As a quick start, users can simply list Kafka Bootstrap Servers using the host parameter in **ziris.comm.config**.

Default Kafka Producer Properties

The default configuration file producer.properties for z/IRIS z/OS Clients can be located in the directory **$APP_HOME/kafka**. This file is used by the z/IRIS z/OS Clients and can be changed by specifying a file path and file name using the kafkaConfig parameter in the configuration file **$APP_HOME/lib/ziris.comm.config**.

default-ziris.producer.properties

```
####################################################################################
############################ DEFAULT PRODUCER CONFIGURATION ########################
############################## for Apache Kafka 2.3.0 ###############################
################################ z/IRIS z/OS Client ################################

# A list of host/port pairs to use for establishing the initial connection to the
# Kafka Cluster.
### NOTE: Kafka Cluster address needs to be adjusted.
#bootstrap.servers=localhost:9092

# Ensures that no duplicate messages will be introduced into Kafka due to network
# errors.
enable.idempotence=true

##################### Implied settings by enable.idempotence #####################
# Producer can choose between different required acknowledgements of data writes
# for data acquisition confirmation. When acks=0, the producer doesn't request any
# response from the Kafka Cluster. When acks=1, the producer requests an
# acknowledgement only from the leader. When acks=all, the producer requests
# acknowledgements from the leader and the corresponding replicas. The default
# setting is acks=1, however enable.idempotence requires acks=all.
acks=all

# Producer will automatically retry to send messages. The default and recommended
# setting is 2147483647 since Kafka 2.1.0. When enable.idempotence=true, it
# has to be a value > 0, otherwise an exception will be thrown.
retries=2147483647

# Restricts how many unacknowledged requests can be made by a producer parallel on
# a single connection before blocking. When enable.idempotence=true, then the
# default setting is 5, otherwise it can be a value between 1<=x<=5 where message
# ordering is ensured as well.
max.in.flight.requests.per.connection=5
##################################################################################

# In general enabled compression would reduce latency because of allowing faster
# data transfers, quicker replication across Kafka Brokers and better Kafka disk
# utilisation due to less data sizes at the expense of CPU cycles. It becomes more
# effective the bigger the batch of messages being sent to Kafka due to a higher
# compression ratio.
compression.type=none

# The total bytes of memory the producer can use to buffer records waiting to be
# sent to the server. The default setting is 33554432 bytes (32 MB).
buffer.memory=33554432

# When the producer produces faster than the Kafka Broker can process the data
# at the moment in time or is temporarily down, then data is going to be buffered
# in the buffer memory. If it fills completely up, then the producer will start to
# block by not producing new data while waiting. The producer will wait a certain
# amount of time until it throws a timeout exception. The default setting for this
# is 60000 ms (1 min).
max.block.ms=60000

# Determines the maximum number of bytes that can be included in a batch. Any
# message that is bigger than the batch size will be sent as soon as possible
# regardless of linger.ms and thus not be batched. The default setting is 16384
# (16 KB). Don't set the batch size too high, since a batch is allocated per
# partition which could possibly lead to wasting or even outrunning of memory.
```

```
batch.size=65536

# Describes the maximum size of a request in bytes. This setting will limit the
# number of message batches the producer will send in a single request to avoid
# sending huge requests. Thus it is also effectively a cap on the maximum record
# batch size. The default setting is 1048576 (1 MB).
max.request.size=1048576

# Producer will wait a small amount of time before sending. At the expense of
# introducing a small delay the chance of messages being sent together in a batch
# can be increased. The default setting is 0 ms.
linger.ms=15

# Close idle connections. The default setting is 540000 ms (9 mins).
connections.max.idle.ms=30000

# For a certain amount of time the producer will continue to retry to send a
# message. After that the producer will stop sending the message. When a message
# can't be acknowledged in this given time frame, then the message sending counts
# as failed. The value should be >= linger.ms + request.timeout.ms. The default
# setting is 120000 ms (2 mins).
delivery.timeout.ms=120000

# Controls the maximum amount of time the producer will wait for Kafka Leader's
# acknowledgement of a sent message. If the acknowledgement is not received in
# time, then the message will be sent again depending on retries and
# delivery.timeout.ms. The default setting is 30000 ms (30 s).
request.timeout.ms=30000

# Describes the amount of time to wait before attempting to retry a failed
# request to a given topic partition. The default setting is 100 ms.
retry.backoff.ms=100

# Client ID in form of a string to pass to the server when making requests. The
# purpose 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. The default setting is "".
client.id=""

# The size of the TCP receive buffer to use when reading data. If the value is
# -1, the OS default will be used. The default setting is 32768 (32 KB).
receive.buffer.bytes=32768

# The size of the TCP send buffer to use when sending data. If the value is
# -1, the OS default will be used. The default setting is 131072 (128 KB).
send.buffer.bytes=131072

# The period of time after which a refresh of metadata is forced even if there
# haven't been seen any partition leadership changes to proactively discover any
# new Kafka Brokers or Partitions. The default setting is 300000 ms (5 mins).
metadata.max.age.ms=300000

# Determines the maximum amount of time to wait when reconnection to a Kafka
# 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. The default setting is 1000 ms (1 s).
reconnect.backoff.max.ms=1000

# The base amount of time to wait before attempting to reconnect to a given host.
# The default setting is 50 ms.
reconnect.backoff.ms=50
```

Required z/IRIS Kafka Producer Properties

The following properties are required for all z/IRIS z/OS Clients where the Kafka Mode "mode=kafka" is set, thus activated and should be present in all alternate producer.properties configurations used by z/IRIS z/OS Clients. Failure to configure these properties may result in erroneous behaviour, reduced reliability and/or reduced performance:

Property	Value	Requirement description
`enable.idempotence`	`true`	The order in which SMF records are read and sent must be maintained to achieve correlation of data within SMF records. Enforcing the order of records sent via the Kafka Producer used by the z/IRIS z/OS Client is achieved by setting the Producer configuration `enable.idempotence=true`.