**Content:**

* **Kafka Producer config**
* **Bootstrap Server**
* **Client ID**
* **Kay Serializer**
* **Value Serializer**
* **connections.max.idle.ms**
* **Acks**

1. **Bootstrap Server:**
   * It is used to connect to Kafka server.
   * A list of host:port pairs to us for establishing the initial connection to Kafka cluster.
   * Client will make use of all the servers irrespective of which servers are specified here for bootstrapping.
   * This list only impacts the initial host used to discover the full set of servers. Means the server which we have mentioned in the command for connection is used to connect to kafka cluster and client will use all the servers or brokers.
   * This list will be in form of host1:port1, host2:port2, host3:port3,…….., hostn:portn.
   * If we are giving a list of bootstrap servers to connect, then it will use first host:port pair to connect and if not able to connect to the kafka cluster using first pair, it will try second pair and so on.
   * For e.g. if we have 10 servers and we need to connect using bootstrap server we can give one or more host:port pairs to connect.
   * The parameters are as shown below:

* **bootstrap.servers**
  + - **Type: list**
    - **Default: “”**
    - **Valid value: Not-null string**
    - **Importance: high**
  + **Below we have a diagram of 3 host:port pairs which are used to connect producer to kafka cluster.**

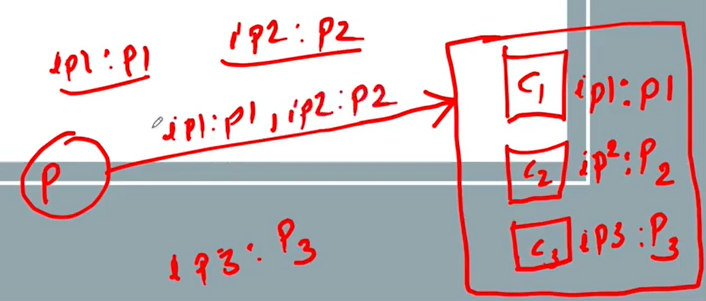


Figure Host:port pairs for producer to kafka cluster connection

1. **Client ID:**
   * Used to trace the requests.
   * Mainly used for debugging.
   * An id string to pass to the server when making requests. The purpose of this is to able to track the source of requests beyond just ip/port by allowing the logical application name to be included in server-side request logging.
   * The parameters are as shown below:

* **client.id**
  + - **Type: string**
    - **Default: “”**
    - **Valid value:**
    - **Importance: medium**

1. **Key serializer:**
   * Serialization is a process of converting an object into stream of bytes. So that we can transmit it over network.
   * Whenever producer publish a message, it sends with key and value.
   * It serializes the key objects.
   * Why serialization?
     + To persist the object.
     + To send an object across a network.
   * The parameters are as shown below:

* **key.serializer:**

Serializer class for a key that implements the **org.apache.kafka.common.serialization.Serializer** interface.

* + - **Type: class**
    - **Default: “byte array Serializer”**
    - **Valid value:**
    - **Importance: high**

1. **Value serializer:**
   * Serialization is a process of converting an object into stream of bytes. So that we can transmit it over network.
   * Whenever producer publish a message, it sends with key and value.
   * It serializes the value objects.
   * Why serialization?
     + To persist the object.
     + To send an object across a network.
   * The parameters are as shown below:

* **value.serializer:**

Serializer class for a value that implements the **org.apache.kafka.common.serialization.Serializer** interface.

* + - **Type: class**
    - **Default: “byte array Serializer”**
    - **Valid value:**
    - **Importance: high**

1. **Connections.max.idle.ms:**

* When a producer connects to kafka cluster it establishes a connection, and **connection.max.idle.ms** decides if a producer is not sending any message to kafka cluster or idle for specified time interval then connection between producer and kafka cluster will be closed
* The parameters are as shown below:
* **connections.max.idle.ms:**

Close idle connections after the number of milliseconds specified by the config.

* + - **Type: long**
    - **Default: 54000**
    - **Valid value:**
    - **Importance: medium**

1. **ACKS:**

* Acks stands for acknowledgement.
* When producer sends records, it will wait for the acknowledgement from the cluster.
* To know if producer should wait for the acknowledgement or not after sending the message to the cluster, it is set by Acks.
* Below we have diagram shows producer and Kafka cluster:

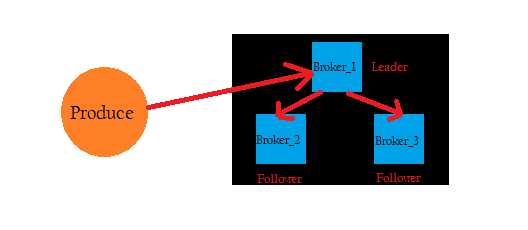


Figure Kafka cluster

* There are three settings possible:

1. **Acks=0;**

If set to zero, then producer will not wait for the acknowledgement from cluster at all.

No guarantee has made that the serer has received the record in this case.

1. **Acks=1; (DEFAULT)**

This will mean the leader will write the record in its local log but respond immediately without waiting for full acknowledgement from all followers.

In this case if leader fails immediately after acknowledging the record, but before the follower have replicated it then the records will be lost.

1. **Acks=-1; OR Acks=All;**

This means that the leader will wait for the full set of in-sync replicas to acknowledge the message.

This guarantees that the record will not be lost until and unless at least one in-sync replica remains alive.

This is **strongest available guarantee**.

This is equivalent to **Acks=All.**

But it will increase the latency of acknowledgement. Means it is slow than other two settings.

* The parameters are as shown below:
* **Type: string**
* **Default: all**
* **Valid Values: [all, -1, 0, 1]**
* **Importance: low**