Schema Evolution

Confluent’s Schema registry tool

2/9/2017

Shalaj Shukla

Table of Contents

[Schema Evolution 2](#_Toc474425230)

[Schema Registry 2](#_Toc474425231)

[Confluent installation on RHEL7 3](#_Toc474425232)

[Develop Producer and Consumer for AVRO Schema 16](#_Toc474425233)

[Click Record Version1 17](#_Toc474425234)

[Click Record Version2 20](#_Toc474425235)

[Producing Record with new Producer and Consuming with old Consumer 23](#_Toc474425236)

[Producing Record with old Producer and Consuming with new Consumer 24](#_Toc474425237)

[Producing Record with new Producer and Consuming with new Consumer 25](#_Toc474425238)

# Schema Evolution

Generally we pass plain string or number data type on kafka topic but If we want to pass object instead of plain string or number data type, we need to write custom serializer and deserializer for converting java object to byte array at producer end and again convert byte array to object at consumer end but if our java object changed in between we need to again write custom serializer and deserializer but new components will not work with old objects or schema and old components will not work with new schema in this case we need to stop the kafka broker and do the changes and again restart the kafka server.

We want to support both old and new schemas simultaneously

We need to have a combination of old and new producer as well as combination of old and new consumer, Kafka should be able to store both type of messages on the same topic. Consumer should be read both types of messages without any error

The Industry solution to handle schema evolution is to include schema information with the data when someone is writing, data should write both schema and data and when someone wants to read the data they first read schema and then read data based on the schema

AVRO is one of them, it is most popular serialization system for Hadoop and its eco system, Kafka follows the same approach and uses Avro to handle the schema evolution problem

There are prebuild and reusable serialization system to help us and simplify the whole process of translation messages according to schema and embedding schema information in the message records

Here we can use confluent Platform that provides us KafkaAvroSerializer and KafkaAvroDeserializer classes for serialization and deserialization process respectively. It also provide Schema Registry tool where we can register Avro Schema.

# Schema Registry

The Confluent Schema Registry enables safe, zero downtime evolution of schemas by centralizing the management of schemas written for the [Avro](http://avro.apache.org/) serialization system. It tracks all versions of schemas used for every topic in Kafka and only allows evolution of schemas according to user-defined compatibility settings. This gives developers confidence that they can safely modify schemas as necessary without worrying that doing so will break a different service they may not even be aware of.

Schema Registry provides a serving layer for your metadata. It provides a RESTful interface for storing and retrieving Avro schemas. It stores a versioned history of all schemas, provides multiple compatibility settings and allows evolution of schemas according to the configured compatibility setting

The Schema Registry also includes plugins for Kafka clients that handle schema storage and retrieval for Kafka messages that are sent in the Avro format. This integration is seamless – if you are already using Kafka with Avro data, using the Schema Registry only requires including the serializers with your application and changing one setting.

# Confluent installation on RHEL7

First install Confluent’s public key, which is used to sign packages in the yum repository.

First install Confluent’s public key, which is used to sign packages in the yum repository.

|  |
| --- |
| $ sudo rpm --import http://packages.confluent.io/rpm/3.1/archive.key |

Add the repository to your **/etc/yum.repos.d/** directory in a file named **confluent.repo**

If you are using RHEL/Centos/Oracle 7

|  |
| --- |
| **[Confluent.dist]**  name**=**Confluent repository (dist)  baseurl**=**http://packages.confluent.io/rpm/3.1/7  gpgcheck**=**1  gpgkey**=**http://packages.confluent.io/rpm/3.1/archive.key  enabled**=**1  **[Confluent]**  name**=**Confluent repository  baseurl**=**http://packages.confluent.io/rpm/3.1  gpgcheck**=**1  gpgkey**=**http://packages.confluent.io/rpm/3.1/archive.key  enabled**=**1 |

It is recommended to clear the yum caches before proceeding:

|  |
| --- |
| $ sudo yum clean all |

The repository is now ready for use.

You can install Confluent Open Source with:

|  |
| --- |
| $ sudo yum install confluent-platform-oss-2.11 |

The number at the end of the package name specifies the Scala version. Currently supported versions are 2.11 (recommended) and 2.10.

Here 2.11 is Scala version

|  |
| --- |
| [root@mac92 yum.repos.d]# sudo yum install confluent-platform-oss-2.11  Loaded plugins: fastestmirror, langpacks  Repository HDP-UTILS-1.1.0.20 is listed more than once in the configuration  Confluent | 2.9 kB 00:00:00  Confluent.dist | 2.9 kB 00:00:00  HDP-2.3 | 2.9 kB 00:00:00  HDP-2.3.4.0 | 2.9 kB 00:00:00  HDP-UTILS-1.1.0.20 | 2.9 kB 00:00:00  Updates-ambari-2.2.0.0 | 2.9 kB 00:00:00  base | 3.6 kB 00:00:00  cloudera-cdh5 | 951 B 00:00:00  cloudera-manager | 2.9 kB 00:00:00  epel/x86\_64/metalink | 6.1 kB 00:00:00  epel | 4.3 kB 00:00:00  extras | 3.4 kB 00:00:00  mesosphere | 2.9 kB 00:00:00  mesosphere-noarch | 2.9 kB 00:00:00  mysql-connectors-community | 2.5 kB 00:00:00  mysql-tools-community | 2.5 kB 00:00:00  mysql56-community | 2.5 kB 00:00:00  updates | 3.4 kB 00:00:00  (1/19): HDP-2.3.4.0/primary\_db | 61 kB 00:00:00  (2/19): HDP-2.3/primary\_db | 61 kB 00:00:00  (3/19): Updates-ambari-2.2.0.0/primary\_db | 5.8 kB 00:00:00  (4/19): HDP-UTILS-1.1.0.20/primary\_db | 28 kB 00:00:00  (5/19): cloudera-manager/primary\_db | 11 kB 00:00:00  (6/19): Confluent/primary\_db | 17 kB 00:00:00  (7/19): Confluent.dist/primary\_db | 8.5 kB 00:00:00  (8/19): base/7/x86\_64/group\_gz | 155 kB 00:00:00  (9/19): extras/7/x86\_64/primary\_db | 121 kB 00:00:00  (10/19): mesosphere/x86\_64/primary\_db | 35 kB 00:00:00  (11/19): mesosphere-noarch/primary\_db | 2.5 kB 00:00:00  (12/19): mysql-connectors-community/x86\_64/primary\_db | 13 kB 00:00:00  (13/19): mysql-tools-community/x86\_64/primary\_db | 32 kB 00:00:00  (14/19): mysql56-community/x86\_64/primary\_db | 159 kB 00:00:00  (15/19): base/7/x86\_64/primary\_db | 5.6 MB 00:00:01  (16/19): epel/x86\_64/group\_gz | 170 kB 00:00:01  (17/19): updates/7/x86\_64/primary\_db | 2.2 MB 00:00:01  (18/19): epel/x86\_64/updateinfo | 732 kB 00:00:03  (19/19): epel/x86\_64/primary\_db | 4.5 MB 00:00:10  cloudera-cdh5/primary | 43 kB 00:00:00  Determining fastest mirrors  \* base: centos.excellmedia.net  \* epel: epel.mirror.net.in  \* extras: centos.excellmedia.net  \* updates: centos.excellmedia.net  cloudera-cdh5 146/146  Resolving Dependencies  --> Running transaction check  ---> Package confluent-platform-oss-2.11.noarch 0:3.1.2-1 will be installed  --> Processing Dependency: confluent-kafka-connect-elasticsearch >= 3.1.2 for package: confluent-platform-oss-2.11-3.1.2-1.noarch  --> Processing Dependency: confluent-kafka-connect-hdfs >= 3.1.2 for package: confluent-platform-oss-2.11-3.1.2-1.noarch  --> Processing Dependency: confluent-camus >= 3.1.2 for package: confluent-platform-oss-2.11-3.1.2-1.noarch  --> Processing Dependency: confluent-kafka-connect-jdbc >= 3.1.2 for package: confluent-platform-oss-2.11-3.1.2-1.noarch  --> Processing Dependency: confluent-schema-registry >= 3.1.2 for package: confluent-platform-oss-2.11-3.1.2-1.noarch  --> Processing Dependency: confluent-kafka-2.11 >= 0.10.1.1 for package: confluent-platform-oss-2.11-3.1.2-1.noarch  --> Processing Dependency: confluent-kafka-rest >= 3.1.2 for package: confluent-platform-oss-2.11-3.1.2-1.noarch  --> Running transaction check  ---> Package confluent-camus.noarch 0:3.1.2-1 will be installed  ---> Package confluent-kafka-2.11.noarch 0:0.10.1.1-1 will be installed  ---> Package confluent-kafka-connect-elasticsearch.noarch 0:3.1.2-1 will be installed  --> Processing Dependency: confluent-common for package: confluent-kafka-connect-elasticsearch-3.1.2-1.noarch  ---> Package confluent-kafka-connect-hdfs.noarch 0:3.1.2-1 will be installed  ---> Package confluent-kafka-connect-jdbc.noarch 0:3.1.2-1 will be installed  ---> Package confluent-kafka-rest.noarch 0:3.1.2-1 will be installed  --> Processing Dependency: confluent-rest-utils for package: confluent-kafka-rest-3.1.2-1.noarch  ---> Package confluent-schema-registry.noarch 0:3.1.2-1 will be installed  --> Running transaction check  ---> Package confluent-common.noarch 0:3.1.2-1 will be installed  ---> Package confluent-rest-utils.noarch 0:3.1.2-1 will be installed  --> Finished Dependency Resolution  Dependencies Resolved  =====================================================================================  Package Arch Version Repository Size  =====================================================================================  Installing:  confluent-platform-oss-2.11 noarch 3.1.2-1 Confluent 6.6 k  Installing for dependencies:  confluent-camus noarch 3.1.2-1 Confluent 19 M  confluent-common noarch 3.1.2-1 Confluent 2.0 M  confluent-kafka-2.11 noarch 0.10.1.1-1 Confluent 36 M  confluent-kafka-connect-elasticsearch noarch 3.1.2-1 Confluent 4.3 M  confluent-kafka-connect-hdfs noarch 3.1.2-1 Confluent 86 M  confluent-kafka-connect-jdbc noarch 3.1.2-1 Confluent 6.0 M  confluent-kafka-rest noarch 3.1.2-1 Confluent 16 M  confluent-rest-utils noarch 3.1.2-1 Confluent 7.0 M  confluent-schema-registry noarch 3.1.2-1 Confluent 26 M  Transaction Summary  ====================================================================================================================================  Install 1 Package (+9 Dependent packages)  Total download size: 203 M  Installed size: 229 M  Is this ok [y/d/N]: y  Downloading packages:  (1/10): confluent-common-3.1.2-1.noarch.rpm | 2.0 MB 00:00:04  (2/10): confluent-kafka-2.11-0.10.1.1-1.noarch.rpm | 36 MB 00:00:24  (3/10): confluent-camus-3.1.2-1.noarch.rpm | 19 MB 00:00:33  (4/10): confluent-kafka-connect-elasticsearch-3.1.2-1.noarch.rpm | 4.3 MB 00:00:07  (5/10): confluent-kafka-connect-jdbc-3.1.2-1.noarch.rpm | 6.0 MB 00:00:07  (6/10): confluent-kafka-rest-3.1.2-1.noarch.rpm | 16 MB 00:00:09  (7/10): confluent-platform-oss-2.11-3.1.2-1.noarch.rpm | 6.6 kB 00:00:00  (8/10): confluent-rest-utils-3.1.2-1.noarch.rpm | 7.0 MB 00:00:12  (9/10): confluent-schema-registry-3.1.2-1.noarch.rpm | 26 MB 00:00:25  (10/10): confluent-kafka-connect-hdfs-3.1.2-1.noarch.rpm | 86 MB 00:01:00  ------------------------------------------------------------------------------------------------------------------------------------  Total 2.2 MB/s | 203 MB 00:01:33  Running transaction check  Running transaction test  Transaction test succeeded  Running transaction  Installing : confluent-common-3.1.2-1.noarch 1/10  Installing : confluent-rest-utils-3.1.2-1.noarch 2/10  Installing : confluent-schema-registry-3.1.2-1.noarch 3/10  Installing : confluent-kafka-rest-3.1.2-1.noarch 4/10  Installing : confluent-kafka-connect-jdbc-3.1.2-1.noarch 5/10  Installing : confluent-kafka-connect-elasticsearch-3.1.2-1.noarch 6/10  Installing : confluent-kafka-connect-hdfs-3.1.2-1.noarch 7/10  Installing : confluent-camus-3.1.2-1.noarch 8/10  Installing : confluent-kafka-2.11-0.10.1.1-1.noarch 9/10  Installing : confluent-platform-oss-2.11-3.1.2-1.noarch 10/10  Verifying : confluent-kafka-2.11-0.10.1.1-1.noarch 1/10  Verifying : confluent-platform-oss-2.11-3.1.2-1.noarch 2/10  Verifying : confluent-schema-registry-3.1.2-1.noarch 3/10  Verifying : confluent-kafka-rest-3.1.2-1.noarch 4/10  Verifying : confluent-kafka-connect-jdbc-3.1.2-1.noarch 5/10  Verifying : confluent-kafka-connect-elasticsearch-3.1.2-1.noarch 6/10  Verifying : confluent-camus-3.1.2-1.noarch 7/10  Verifying : confluent-rest-utils-3.1.2-1.noarch 8/10  Verifying : confluent-kafka-connect-hdfs-3.1.2-1.noarch 9/10  Verifying : confluent-common-3.1.2-1.noarch 10/10  Installed:  confluent-platform-oss-2.11.noarch 0:3.1.2-1  Dependency Installed:  confluent-camus.noarch 0:3.1.2-1 confluent-common.noarch 0:3.1.2-1  confluent-kafka-2.11.noarch 0:0.10.1.1-1 confluent-kafka-connect-elasticsearch.noarch 0:3.1.2-1  confluent-kafka-connect-hdfs.noarch 0:3.1.2-1 confluent-kafka-connect-jdbc.noarch 0:3.1.2-1  confluent-kafka-rest.noarch 0:3.1.2-1 confluent-rest-utils.noarch 0:3.1.2-1  confluent-schema-registry.noarch 0:3.1.2-1  Complete!  [root@mac92 yum.repos.d]# |

Components are installed under **/etc/kafka**, **/etc/kafka-rest**, **/etc/schema-registry** and **/etc/camus**.

Start zookeeper, Kafka and schema registry

|  |
| --- |
| $ zookeeper-server-start /etc/kafka/zookeeper.properties |
| $kafka-server-start /etc/kafka/server.properties |
| $schema-registry-start /etc/schema-registry/schema-registry.properties |

Schema registry default listen on 8081 check schema-registry.properties file

|  |
| --- |
| [root@mac92 schema-registry]# schema-registry-start /etc/schema-registry/schema-registry.properties  [2017-02-08 09:17:49,139] INFO SchemaRegistryConfig values:  metric.reporters = []  kafkastore.sasl.kerberos.kinit.cmd = /usr/bin/kinit  response.mediatype.default = application/vnd.schemaregistry.v1+json  kafkastore.ssl.trustmanager.algorithm = PKIX  authentication.realm =  ssl.keystore.type = JKS  kafkastore.topic = \_schemas  metrics.jmx.prefix = kafka.schema.registry  kafkastore.ssl.enabled.protocols = TLSv1.2,TLSv1.1,TLSv1  kafkastore.topic.replication.factor = 3  ssl.truststore.password =  kafkastore.timeout.ms = 500  host.name = mac92.cybage.com  kafkastore.bootstrap.servers = []  schema.registry.zk.namespace = schema\_registry  kafkastore.sasl.kerberos.ticket.renew.window.factor = 0.8  kafkastore.sasl.kerberos.service.name =  ssl.endpoint.identification.algorithm =  compression.enable = false  kafkastore.ssl.truststore.type = JKS  avro.compatibility.level = backward  kafkastore.ssl.protocol = TLS  kafkastore.ssl.provider =  kafkastore.ssl.truststore.location =  response.mediatype.preferred = [application/vnd.schemaregistry.v1+json, application/vnd.schemaregistry+json, application/json]  kafkastore.ssl.keystore.type = JKS  ssl.truststore.type = JKS  kafkastore.ssl.truststore.password =  access.control.allow.origin =  ssl.truststore.location =  ssl.keystore.password =  port = 8081  kafkastore.ssl.keystore.location =  master.eligibility = true  ssl.client.auth = false  kafkastore.ssl.keystore.password =  kafkastore.security.protocol = PLAINTEXT  ssl.trustmanager.algorithm =  authentication.method = NONE  request.logger.name = io.confluent.rest-utils.requests  ssl.key.password =  kafkastore.zk.session.timeout.ms = 30000  kafkastore.sasl.mechanism = GSSAPI  kafkastore.sasl.kerberos.ticket.renew.jitter = 0.05  kafkastore.ssl.key.password =  zookeeper.set.acl = false  authentication.roles = [\*]  metrics.num.samples = 2  ssl.protocol = TLS  kafkastore.ssl.keymanager.algorithm = SunX509  kafkastore.connection.url = localhost:2181  debug = false  listeners = [http://0.0.0.0:8081]  ssl.provider =  ssl.enabled.protocols = []  shutdown.graceful.ms = 1000  ssl.keystore.location =  ssl.cipher.suites = []  kafkastore.ssl.endpoint.identification.algorithm =  kafkastore.ssl.cipher.suites =  access.control.allow.methods =  kafkastore.sasl.kerberos.min.time.before.relogin = 60000  ssl.keymanager.algorithm =  metrics.sample.window.ms = 30000  kafkastore.init.timeout.ms = 60000  (io.confluent.kafka.schemaregistry.rest.SchemaRegistryConfig:169)  [2017-02-08 09:17:50,800] INFO Initializing KafkaStore with broker endpoints: PLAINTEXT://mac92.cybage.com:9092 (io.confluent.kafka.schemaregistry.storage.KafkaStore:127)  [2017-02-08 09:17:50,815] WARN Creating the schema topic \_schemas using a replication factor of 1, which is less than the desired one of 3. If this is a production environment, it's crucial to add more brokers and increase the replication factor of the topic. (io.confluent.kafka.schemaregistry.storage.KafkaStore:258)  [2017-02-08 09:17:51,663] INFO Initialized last consumed offset to -1 (io.confluent.kafka.schemaregistry.storage.KafkaStoreReaderThread:122)  [2017-02-08 09:17:51,664] INFO [kafka-store-reader-thread-\_schemas], Starting (io.confluent.kafka.schemaregistry.storage.KafkaStoreReaderThread:70)  [2017-02-08 09:17:51,881] INFO Wait to catch up until the offset of the last message at 0 (io.confluent.kafka.schemaregistry.storage.KafkaStore:343)  [2017-02-08 09:17:52,113] INFO Created schema registry namespace localhost:2181/schema\_registry (io.confluent.kafka.schemaregistry.storage.KafkaSchemaRegistry:238)  [2017-02-08 09:17:52,179] INFO Successfully elected the new master: {"host":"mac92.cybage.com","port":8081,"master\_eligibility":true,"version":1} (io.confluent.kafka.schemaregistry.zookeeper.ZookeeperMasterElector:83)  [2017-02-08 09:17:52,229] INFO Successfully elected the new master: {"host":"mac92.cybage.com","port":8081,"master\_eligibility":true,"version":1} (io.confluent.kafka.schemaregistry.zookeeper.ZookeeperMasterElector:83)  [2017-02-08 09:17:52,339] INFO Logging initialized @4316ms (org.eclipse.jetty.util.log:186)  [2017-02-08 09:17:52,409] INFO Adding listener: http://0.0.0.0:8081 (io.confluent.rest.Application:174)  [2017-02-08 09:17:52,470] INFO jetty-9.2.12.v20150709 (org.eclipse.jetty.server.Server:327)  [2017-02-08 09:17:53,395] INFO HV000001: Hibernate Validator 5.1.2.Final (org.hibernate.validator.internal.util.Version:27)  [2017-02-08 09:17:53,625] INFO Started o.e.j.s.ServletContextHandler@5c153b9e{/,null,AVAILABLE} (org.eclipse.jetty.server.handler.ContextHandler:744)  [2017-02-08 09:17:53,666] INFO Started NetworkTrafficServerConnector@784c3487{HTTP/1.1}{0.0.0.0:8081} (org.eclipse.jetty.server.NetworkTrafficServerConnector:266)  [2017-02-08 09:17:53,668] INFO Started @5645ms (org.eclipse.jetty.server.Server:379)  [2017-02-08 09:17:53,669] INFO Server started, listening for requests... (io.confluent.kafka.schemaregistry.rest.SchemaRegistryMain:45) |

To test it we can use console command first

|  |
| --- |
| kafka-avro-console-producer \  --broker-list localhost:9092 --topic test \  --property value.schema**=**'{"type":"record","name":"myrecord","fields":[{"name":"f1","type":"string"}]}' |

Once started, the process will wait for you to enter messages, one per line, and will send them immediately when you hit the Enter key. Try entering a couple of messages:

**{**"f1": "value1"**}**

**{**"f1": "value2"**}**

**{**"f1": "value3"**}**

|  |
| --- |
| [root@mac92 ~]# kafka-avro-console-producer --broker-list localhost:9092 --topic test --property value.schema='{"type":"record","name":"myrecord","fields":[{"name":"f1","type":"string"}]}'  SLF4J: Class path contains multiple SLF4J bindings.  SLF4J: Found binding in [jar:file:/usr/share/java/kafka-serde-tools/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/share/java/schema-registry/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.  SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  {"f1": "value1"}  {"f1": "value2"}  {"f1": "value3"} |

Now start consumer in separate window

|  |
| --- |
| [root@mac92 ~]# kafka-avro-console-consumer --topic test \  > --zookeeper localhost:2181 \  > --from-beginning  SLF4J: Class path contains multiple SLF4J bindings.  SLF4J: Found binding in [jar:file:/usr/share/java/kafka-serde-tools/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/share/java/schema-registry/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.  SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] instead of [zookeeper].  {"f1":"value1"}  {"f1":"value1"}  {"f1":"value2"} |

You should see all the messages you created in the previous step written to the console in the same format.

The consumer does not exit after reading all the messages so it can listen for and process new messages as they are published. Try keeping the consumer running and repeating step 5 – you will see messages delivered to the consumer immediately after you hit Enter for each message in the producer.

When you’re done, shut down the consumer with Ctrl+C.

Now let’s try to produce data to the same topic using an incompatible schema. We’ll run the producer with nearly the same command, but change the schema to expect plain integers.

|  |
| --- |
| kafka-avro-console-producer \  --broker-list localhost:9092 --topic test \  --property value.schema**=**'{"type":"int"}' |

Now if you enter an integer and hit enter, you should see the following (expected) exception:

|  |
| --- |
| org.apache.kafka.common.errors.SerializationException: Error registering Avro schema: "int"  Caused by: io.confluent.kafka.schemaregistry.client.rest.exceptions.RestClientException: Schema being registered is incompatible with the latest schema; error code: 409  at io.confluent.kafka.schemaregistry.client.rest.utils.RestUtils.httpRequest**(**RestUtils.java:146**)**  at io.confluent.kafka.schemaregistry.client.rest.utils.RestUtils.registerSchema**(**RestUtils.java:174**)**  at io.confluent.kafka.schemaregistry.client.CachedSchemaRegistryClient.registerAndGetId**(**CachedSchemaRegistryClient.java:51**)**  at io.confluent.kafka.schemaregistry.client.CachedSchemaRegistryClient.register**(**CachedSchemaRegistryClient.java:89**)**  at io.confluent.kafka.serializers.AbstractKafkaAvroSerializer.serializeImpl**(**AbstractKafkaAvroSerializer.java:49**)**  at io.confluent.kafka.formatter.AvroMessageReader.readMessage**(**AvroMessageReader.java:155**)**  at kafka.tools.ConsoleProducer$.main**(**ConsoleProducer.scala:94**)**  at kafka.tools.ConsoleProducer.main**(**ConsoleProducer.scala**)** |

It is because we have change the registered schema with topic

|  |
| --- |
| kafka-avro-console-producer \  > --broker-list localhost:9092 --topic test \  > --property value.schema='{"type":"int"}'  SLF4J: Class path contains multiple SLF4J bindings.  SLF4J: Found binding in [jar:file:/usr/share/java/kafka-serde-tools/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/share/java/schema-registry/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.  SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  123  org.apache.kafka.common.errors.SerializationException: Error registering Avro schema: "int"  Caused by: io.confluent.kafka.schemaregistry.client.rest.exceptions.RestClientException: Schema being registered is incompatible with the latest schema; error code: 409  at io.confluent.kafka.schemaregistry.client.rest.RestService.sendHttpRequest(RestService.java:170)  at io.confluent.kafka.schemaregistry.client.rest.RestService.httpRequest(RestService.java:187)  at io.confluent.kafka.schemaregistry.client.rest.RestService.registerSchema(RestService.java:238)  at io.confluent.kafka.schemaregistry.client.rest.RestService.registerSchema(RestService.java:230)  at io.confluent.kafka.schemaregistry.client.rest.RestService.registerSchema(RestService.java:225)  at io.confluent.kafka.schemaregistry.client.CachedSchemaRegistryClient.registerAndGetId(CachedSchemaRegistryClient.java:59)  at io.confluent.kafka.schemaregistry.client.CachedSchemaRegistryClient.register(CachedSchemaRegistryClient.java:91)  at io.confluent.kafka.serializers.AbstractKafkaAvroSerializer.serializeImpl(AbstractKafkaAvroSerializer.java:72)  at io.confluent.kafka.formatter.AvroMessageReader.readMessage(AvroMessageReader.java:158)  at kafka.tools.ConsoleProducer$.main(ConsoleProducer.scala:55)  at kafka.tools.ConsoleProducer.main(ConsoleProducer.scala)  [root@mac92 ~]# |

Please follow below link for install confluent platform

<http://docs.confluent.io/3.1.2/installation.html>

# Develop Producer and Consumer for AVRO Schema

First create a maven project and use below pom.xml file

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>org.cybage.kafka</groupId>  <artifactId>KafkaProject</artifactId>  <version>0.0.1-SNAPSHOT</version>  <repositories>  <repository>  <id>confluent</id>  <url>http://packages.confluent.io/maven/</url>  </repository>  </repositories>  <dependencies>  <dependency>  <groupId>io.confluent</groupId>  <artifactId>kafka-avro-serializer</artifactId>  <version>3.1.2</version>  </dependency>  <dependency>  <groupId>org.apache.kafka</groupId>  <artifactId>kafka\_2.11</artifactId>  <version>0.10.1.1</version>  </dependency>  <dependency>  <groupId>org.apache.avro</groupId>  <artifactId>avro</artifactId>  <version>1.8.1</version>  </dependency>  <dependency>  <groupId>org.apache.avro</groupId>  <artifactId>avro-tools</artifactId>  <version>1.8.1</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-shade-plugin</artifactId>  <configuration>  <createDependencyReducedPom>true</createDependencyReducedPom>  </configuration>  <executions>  <execution>  <phase>package</phase>  <goals>  <goal>shade</goal>  </goals>  <configuration>  <transformers>  <transformer  implementation=*"org.apache.maven.plugins.shade.resource.ServicesResourceTransformer"* />  <transformer  implementation=*"org.apache.maven.plugins.shade.resource.ManifestResourceTransformer"*>  <mainClass>org.cybage.esplus.ingestion.topology.Extract</mainClass>  </transformer>  </transformers>  </configuration>  </execution>  </executions>  </plugin>  </plugins>  </build>  </project> |

## Click Record Version1

Create an Avro (JSON) Schema for Click Record

|  |
| --- |
| { "type": "record",  "name": "ClickRecord",  "fields": [  {"name": "session\_id", "type": "string"},  {"name": "browser", "type": ["string", "null"]},  {"name": "campaign", "type": ["string", "null"]},  {"name": "channel", "type": "string"},  {"name": "referrer", "type": ["string", "null"], "default": "None"},  {"name": "ip", "type": ["string", "null"]}  ]  } |

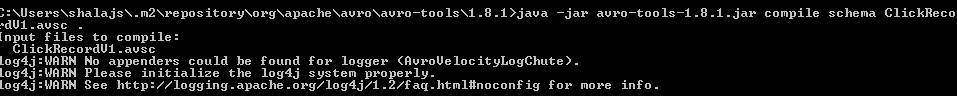
This is version 1 schema for click record

|  |  |
| --- | --- |
| Field | Description |
| session\_id | An identifier for Session |
| browser | An identifier for the browser |
| campaign | A custom identifier for running campaign |
| channel | A custom identifier for section of the site |
| referrer | A first hit referrer (ex. Facebook.com) (default is None) |
| IP | An IP Address from your ISP |

Now run below command to create a java file using this schema

|  |
| --- |
| java -jar /path/to/avro-tools-1.8.1.jar compile schema <schema file> <destination> |

|  |
| --- |
| java -jar avro-tools-1.8.1.jar compile schema ClickRecordV1.avsc . |



It will create ClickRecord.java file in current location

Just get this file and add this in your java folder (under default package) now create AvroProducer and AvroConsumer for version 1

|  |
| --- |
| **import** java.util.\*;  **import** org.apache.kafka.clients.producer.\*;  **public** **class** AvroProducer {  **public** **static** **void** main(String[] args) **throws** Exception{  String topicName = "AvroClicks";    Properties props = **new** Properties();  props.put("bootstrap.servers", "localhost:9092");  props.put("key.serializer","org.apache.kafka.common.serialization.StringSerializer");  props.put("value.serializer", "io.confluent.kafka.serializers.KafkaAvroSerializer");  //  props.put("schema.registry.url", "http://localhost:8081");  Producer<String, ClickRecord> producer = **new** KafkaProducer <>(props);  ClickRecord cr = **new** ClickRecord();  **try**{  cr.setSessionId("10001");  cr.setChannel("HomePage");  cr.setIp("192.168.0.1");  producer.send(**new** ProducerRecord<String, ClickRecord>(topicName,cr.getSessionId().toString(),cr)).get();  System.***out***.println("Complete");  }  **catch**(Exception ex){  ex.printStackTrace(System.***out***);  }  **finally**{  producer.close();  }  }  } |

|  |
| --- |
| **import** java.util.\*;  **import** org.apache.kafka.clients.consumer.\*;  **public** **class** AvroConsumer{    **public** **static** **void** main(String[] args) **throws** Exception{  String topicName = "AvroClicks";    String groupName = "RG";  Properties props = **new** Properties();  props.put("bootstrap.servers", "localhost:9092");  props.put("group.id", groupName);  props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");  props.put("value.deserializer", "io.confluent.kafka.serializers.KafkaAvroDeserializer");  props.put("schema.registry.url", "http://localhost:8081");  //this is mandatory for avro read  props.put("specific.avro.reader", "true");    KafkaConsumer<String, ClickRecord> consumer = **new** KafkaConsumer<>(props);  consumer.subscribe(Arrays.*asList*(topicName));  **try**{  **while** (**true**){  ConsumerRecords<String, ClickRecord> records = consumer.poll(100);  **for** (ConsumerRecord<String, ClickRecord> record : records){  System.***out***.println("Session id="+ record.value().getSessionId()  + " Channel=" + record.value().getChannel()  + " Referrer=" + record.value().getReferrer());  }  }  }**catch**(Exception ex){  ex.printStackTrace();  }  **finally**{  consumer.close();  }  }    } |

All components for Click Record version1 are attached herewith



Now create a jar file click on Run As >> Maven Install, you will find jar file under target folder

Now Start Producer first to send one Avro record on AvroClicks Topic

|  |
| --- |
| [root@mac92 kakfa\_test]# java -classpath KafkaProject-0.0.1-SNAPSHOT.jar AvroProducer  log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.producer.ProducerConfig).  log4j:WARN Please initialize the log4j system properly.  log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  Complete  [root@mac92 kakfa\_test]# |

Now Strat Consumer

|  |
| --- |
| [root@mac92 kakfa\_test]# java -classpath KafkaProject-0.0.1-SNAPSHOT.jar AvroConsumer  log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.consumer.ConsumerConfig).  log4j:WARN Please initialize the log4j system properly.  log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  Session id=10001 Channel=HomePage Referrer=null |

Here consumer consumed the data without issue

## Click Record Version2

Now assume that if Click Record schema changed as below

|  |
| --- |
| {"type": "record",  "name": "ClickRecord",  "fields": [  {"name": "session\_id", "type": "string"},  {"name": "browser", "type": ["string", "null"]},  {"name": "campaign", "type": ["string", "null"]},  {"name": "channel", "type": "string"},  {"name": "entry\_url", "type": ["string", "null"], "default": "None"},  {"name": "ip", "type": ["string", "null"]},  {"name": "language", "type": ["string", "null"], "default": "None"},  {"name": "os", "type": ["string", "null"],"default": "None"}  ]  } |

This is version2 schema for click record

|  |  |
| --- | --- |
| Field | Description |
| session\_id | An identifier for Session |
| browser | An identifier for the browser |
| campaign | A custom identifier for running campaign |
| channel | A custom identifier for section of the site |
| ~~referrer~~ | ~~A first hit referrer (ex. Facebook.com) (default is None)~~ |
| entry\_url | A first hit referrer URL (default is None) |
| ip | An IP Address from your ISP |
| language | An identifier for the language |
| os | An identifier for the Operating System |

Here we added some new field and removed one field

Now again build ClickRecord.java file using avro build tool for new schema

|  |
| --- |
| java -jar avro-tools-1.8.1.jar compile schema ClickRecordV2.avsc . |

Now write new Producer and Consumer files

|  |
| --- |
| **import** java.util.\*;  **import** org.apache.kafka.clients.consumer.\*;  **public** **class** ClickRecordConsumerV2{    **public** **static** **void** main(String[] args) **throws** Exception{  String topicName = "AvroClicks";    String groupName = "RG";  Properties props = **new** Properties();  props.put("bootstrap.servers", "localhost:9092,localhost:9093");  props.put("group.id", groupName);  props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");  props.put("value.deserializer", "io.confluent.kafka.serializers.KafkaAvroDeserializer");  props.put("schema.registry.url", "http://localhost:8081");  props.put("specific.avro.reader", "true");    KafkaConsumer<String, ClickRecord> consumer = **new** KafkaConsumer<>(props);  consumer.subscribe(Arrays.*asList*(topicName));  **try**{  **while** (**true**){  ConsumerRecords<String, ClickRecord> records = consumer.poll(100);  **for** (ConsumerRecord<String, ClickRecord> record : records){  System.***out***.println("Session id="+ record.value().getSessionId()  + " Channel=" + record.value().getChannel()  + " Entry URL=" + record.value().getEntryUrl()  + " Language=" + record.value().getLanguage());  }  }  }**catch**(Exception ex){  ex.printStackTrace();  }  **finally**{  consumer.close();  }  }    } |

|  |
| --- |
| **import** java.util.\*;  **import** org.apache.kafka.clients.producer.\*;  **public** **class** ClickRecordProducerV2 {  **public** **static** **void** main(String[] args) **throws** Exception{  String topicName = "AvroClicks";    Properties props = **new** Properties();  props.put("bootstrap.servers", "localhost:9092,localhost:9093");  props.put("key.serializer","org.apache.kafka.common.serialization.StringSerializer");  props.put("value.serializer", "io.confluent.kafka.serializers.KafkaAvroSerializer");  props.put("schema.registry.url", "http://localhost:8081");  Producer<String, ClickRecord> producer = **new** KafkaProducer <>(props);  ClickRecord cr = **new** ClickRecord();  **try**{  cr.setSessionId("10001");  cr.setChannel("HomePage");  cr.setIp("192.168.0.1");  cr.setLanguage("Spanish");  cr.setOs("Mac");  cr.setEntryUrl("http://facebook.com/myadd");  producer.send(**new** ProducerRecord<String, ClickRecord>(topicName,cr.getSessionId().toString(),cr)).get();  System.***out***.println("Complete");  }  **catch**(Exception ex){  ex.printStackTrace(System.***out***);  }  **finally**{  producer.close();  }  }  } |

Again build new jar with some different name so that we test it using both old and new jars

## Producing Record with new Producer and Consuming with old Consumer

First start old Consumer

|  |
| --- |
| java -classpath KafkaProject-0.0.1-SNAPSHOT.jar AvroConsumer |

Now produce record with new Producer

|  |
| --- |
| [root@mac92 kakfa\_test]# java -classpath KafkaProject-0.0.1-SNAPSHOT\_V2.jar ClickRecordProducerV2  log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.producer.ProducerConfig).  log4j:WARN Please initialize the log4j system properly.  log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  Complete |

Now Check Consumer if it consumed the record using new schema

|  |
| --- |
| [root@mac92 kakfa\_test]# java -classpath KafkaProject-0.0.1-SNAPSHOT.jar AvroConsumer  log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.consumer.ConsumerConfig).  log4j:WARN Please initialize the log4j system properly.  log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  Session id=10001 Channel=HomePage Referrer=None |

We can see old Consumer consumed the data without any issue

## Producing Record with old Producer and Consuming with new Consumer

Run New Consumer

|  |
| --- |
| java -classpath KafkaProject-0.0.1-SNAPSHOT\_V2.jar ClickRecordConsumerV2 |

Produce record with old Producer

|  |
| --- |
| [root@mac92 kakfa\_test]# java -classpath KafkaProject-0.0.1-SNAPSHOT.jar AvroProducer  log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.producer.ProducerConfig).  log4j:WARN Please initialize the log4j system properly.  log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  Complete |

Check new Consumer

|  |
| --- |
| [root@mac92 kakfa\_test]# java -classpath KafkaProject-0.0.1-SNAPSHOT\_V2.jar ClickRecordConsumerV2  log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.consumer.ConsumerConfig).  log4j:WARN Please initialize the log4j system properly.  log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  Session id=10001 Channel=HomePage Entry URL=None Language=None |

Here you can see it is consuming values without any issue

## Producing Record with new Producer and Consuming with new Consumer

Run New Consumer

|  |
| --- |
| java -classpath KafkaProject-0.0.1-SNAPSHOT\_V2.jar ClickRecordConsumerV2 |

Produce record with new Producer

|  |
| --- |
| [root@mac92 kakfa\_test]# java -classpath KafkaProject-0.0.1-SNAPSHOT\_V2.jar ClickRecordProducerV2  log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.producer.ProducerConfig).  log4j:WARN Please initialize the log4j system properly.  log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  Complete |

Check new Consumer

|  |
| --- |
| [root@mac92 kakfa\_test]# java -classpath KafkaProject-0.0.1-SNAPSHOT\_V2.jar ClickRecordConsumerV2  log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.consumer.ConsumerConfig).  log4j:WARN Please initialize the log4j system properly.  log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  Session id=10001 Channel=HomePage Entry URL=http://facebook.com/myadd Language=Spanish |

Here you can see it is accepting new values

That is how we can use both old and new schema simultaneously using schema registry tool with zero downtime