

KAFKA, AVRO SERIALIZATION AND THE SCHEMA REGISTRY

May 9, 2017



Share

Tweet

Like 49

Share

Kafka Tutorial: Kafka, Avro Serialization and the Schema Registry

Confluent Schema Registry stores Avro (<http://cloudurable.com/blog/avro/index.html>) Schemas for Kafka producers and consumers. The Schema Registry and provides RESTful interface for managing Avro schemas It allows the storage of a history of schemas which are versioned. the *Confluent Schema Registry* supports checking schema compatibility for Kafka.

Cloudurable provides Kafka training (<http://cloudurable.com/kafka-training/index.html>), Kafka consulting (<http://cloudurable.com/kafka-aws-consulting/index.html>), Kafka support (http://cloudurable.com/subscription_support/index.html) and helps setting up Kafka clusters in AWS (<http://cloudurable.com/services/index.html>).

You can configure compatibility setting which supports the evolution of schemas using Avro. Kafka Avro serialization project provides serializers. Kafka Producers and Consumers that use Kafka Avro serialization handle schema management and serialization of records using Avro and the Schema Registry. When using the *Confluent Schema Registry*, Producers don't have to send schema just the schema id which is unique. The consumer uses the schema id to look up the full schema from the *Confluent Schema Registry* if not already cached. Since you don't have to send the schema with each set of records, this saves time. Not sending the schema with each record or batch of records, speeds up the serialization as only the id of the schema is sent.

If you have never used Avro before, please read Avro Introduction for Big Data and Data Streams (<http://cloudurable.com/blog/avro/index.html>).

This article is going to cover what is the Schema Registry and cover why you want to use it with Kafka. We drill down into understanding Avro *schema evolution* and setting up and using Schema Registry with Kafka Avro Serializers. We show how to manage Avro Schemas with REST interface of the Schema Registry and then how to write Avro Serializer based Producers and Avro Deserializer based Consumers for Kafka.

The Kafka Producer creates a record/message, which is an Avro record. The record contains a schema id and data. With Kafka Avro Serializer, the schema is registered if needed and then it serializes the data and schema id. The Kafka Avro Serializer keeps a cache of registered schemas from Schema Registry their schema ids.

Consumers receive payloads and deserialize them with Kafka Avro Deserializers which use the Confluent Schema Registry. Deserializer looks up the full schema from cache or Schema Registry based on id.

Why Schema Registry?

Consumer has its schema which could be different than the producers. The consumer schema is the schema the consumer is expecting the record/message to conform to. With the Schema Registry a compatibility check is performed and if the two schemas don't match but are compatible, then the payload transformation happens via Avro Schema Evolution. Kafka records can have a `key` and a `value` and both can have a schema.

Schema Registry Operations

The Schema Registry can store schemas for keys and values of Kafka records. It can also list schemas by subject. It can list all versions of a subject (schema). It can retrieve a schema by version or id. It can get the latest version of a schema. Importantly, the Schema Registry can check to see if schema is compatible with a certain version. There is a compatibility level (BACKWARDS, FORWARDS, FULL, NONE) setting for the Schema Registry and an individual subject. You can manage schemas via a REST API with the Schema registry.

Schema Registry Schema Compatibility Settings

Backward compatibility means data written with older schema is readable with a newer schema. *Forward compatibility* means data written with newer schema is readable with old schemas. *Full compatibility* means a new version of a schema is backward and forward compatible. *None* disables schema validation and it not recommended. If you set the level to none then Schema Registry just stores the schema and Schema will not be validated for compatibility at all.

Cloudurable provides Kafka training (<http://cloudurable.com/kafka-training/index.html>), Kafka consulting (<http://cloudurable.com/kafka-aws-consulting/index.html>), Kafka support (http://cloudurable.com/subscription_support/index.html) and helps setting up Kafka clusters in AWS (<http://cloudurable.com/services/index.html>).

Schema Registry Config

The Schema compatibility checks can be configured globally or per subject.

The compatibility checks value is one of the following:

- NONE - don't check for schema compatibility
- FORWARD - check to make sure last schema version is forward compatible with new schemas
- BACKWARDS (default) - make sure new schema is backwards compatible with latest
- FULL - make sure new schema is forwards and backwards compatible from latest to new and from new to latest

Schema Evolution

If an Avro schema is changed after data has been written to store using an older version of that schema, then Avro might do a Schema Evolution when you try to read that data.

From Kafka perspective, Schema evolution happens only during deserialization at Consumer (read). If Consumer's schema is different from Producer's schema, then value or key is automatically modified during deserialization to conform to consumers reader schema if possible.

Avro *schema evolution* is an automatic transformation of Avro schema between the consumer schema version and what the schema the producer put into the Kafka log. When Consumer schema is not identical to the Producer schema used to serialize the Kafka Record, then a data transformation is performed on the Kafka record's key or value. If the schemas match then no need to do a transformation

Allowed Modification During Schema Evolution

You can add a field with a default to a schema. You can remove a field that had a default value. You can change a field's order attribute. You can change a field's default value to another value or add a default value to a field that did not have one. You can remove or add a field alias (keep in mind that this could break some consumers that depend on the alias). You can change a type to a union that contains original type. If you do any of the above, then your schema can use Avro's schema evolution when reading with an old schema.

Rules of the Road for modifying Schema

If you want to make your schema evolvable, then follow these guidelines. Provide a default value for fields in your schema as this allows you to delete the field later. Never change a field's data type. When adding a new field to your schema, you have to provide a default value for the field. Don't rename an existing field (use aliases instead). You can add an alias

Let's use an example to talk about this. The following example is from our Avro tutorial (<http://cloudurable.com/blog/avro/index.html>).

Employee example Avro Schema

```
{
  "namespace": "com.cloudurable.phonebook",
  "type": "record",
  "name": "Employee",
  "doc" : "Represents an Employee at a company",
  "fields": [
    { "name": "firstName", "type": "string", "doc": "The persons given name"},
    { "name": "nickName", "type": ["null", "string"], "default" : null},
    { "name": "lastName", "type": "string"},
    { "name": "age", "type": "int", "default": -1},
    { "name": "emails", "default": [], "type": { "type": "array", "items": "string" }},
    { "name": "phoneNumber", "type":
      [ "null",
        { "type": "record", "name": "PhoneNumber",
          "fields": [
            { "name": "areaCode", "type": "string"},
            { "name": "countryCode", "type": "string", "default" : ""},
            { "name": "prefix", "type": "string"},
            { "name": "number", "type": "string"}
          ]
        }
      ]
    },
    { "name": "status", "default" : "SALARY", "type": { "type": "enum", "name": "Status",
      "symbols" : ["RETIRED", "SALARY", "HOURLY", "PART_TIME"]}
    }
  ]
}
```

Avro Schema Evolution Scenario

Let's say our `Employee` record did not have an `age` in version 1 of the schema and then later we decided to add an `age` field with a default value of -1. Now let's say we have a Producer using version 2 of the schema with `age`, and a Consumer using version 1 with no `age`.

The Producer uses version 2 of the `Employee` schema and creates a `com.cloudurable.Employee` record, and sets `age` field to 42, then sends it to Kafka topic `new-employees`. The Consumer consumes records from `new-employees` using version 1 of the `Employee` Schema. Since Consumer is using version 1 of the schema, the `age` field gets removed during deserialization.

The same consumer modifies some records and then writes the record to a NoSQL store. When the Consumer does this, the `age` field is missing from the record that it writes to the NoSQL store. Another client using version 2 of the schema which has the `age`, reads the record from the NoSQL store. The `age` field is missing from the record because the Consumer wrote it with version 1, thus the client reads the record and the `age` is set to default value of

-1 .

If you added the `age` and it was not optional, i.e., the `age` field did not have a default, then the Schema Registry could reject the schema, and the Producer could never add it to the Kafka log.

Using REST Schema Registry REST API

Recall that the Schema Registry allows you to manage schemas using the following operations:

- store schemas for keys and values of Kafka records
- List schemas by subject.
- list all versions of a subject (schema).
- Retrieves a schema by version
- Retrieves a schema by id
- Retrieve the latest version of a schema
- Perform compatibility checks
- Set compatibility level globally
- Set compatibility level globally

Recall that all of this is available via a REST API with the Schema Registry.

To post a new schema you could do the following:

Posting a new schema

```
curl -X POST -H "Content-Type:
application/vnd.schemaregistry.v1+json" \
  --data '{"schema": "{\"type\": ...}'}' \
  http://localhost:8081/subjects/Employee/versions
```

To list all of the schemas

```
curl -X GET http://localhost:8081/subjects
```

If you have a good HTTP client, you can basically perform all of the above operations via the REST interface for the Schema Registry. I wrote a little example to do this so I could understand the Schema registry a little better using the OkHttp client from Square (`com.squareup.okhttp3:okhttp:3.7.0+`) as follows:

Using REST endpoints to try out all of the Schema Registry options

```
package com.cloudurable.kafka.schema;

import okhttp3.*;

import java.io.IOException;

public class SchemaMain {

    private final static MediaType SCHEMA_CONTENT =
        MediaType.parse("application/vnd.schemaregistry.v1+json");

    private final static String EMPLOYEE_SCHEMA = "{\n" +
        "  \"schema\": \"\" +
        "  {\n" +
        "    \"namespace\": \"com.cloudurable.phonebook\", \" +
        "    \"type\": \"record\", \" +
        "    \"name\": \"Employee\", \" +
        "    \"fields\": [\n" +
        "      {\n" +
        "        \"name\": \"fName\", \"type\": \"string\", \" +
        "        \"name\": \"lName\", \"type\": \"string\", \" +
        "        \"name\": \"age\", \"type\": \"int\", \" +
        "        \"name\": \"phoneNumber\", \"type\": \"string\" \" +
        "      ]\n" +
        "    }\n" +
        "  }\"";

    public static void main(String... args) throws IOException {

        System.out.println(EMPLOYEE_SCHEMA);

        final OkHttpClient client = new OkHttpClient();

        //POST A NEW SCHEMA
        Request request = new Request.Builder()
            .post(RequestBody.create(SCHEMA_CONTENT, EMPLOYEE_SCHEMA))
            .url("http://localhost:8081/subjects/Employee/versions")
            .build();

        String output = client.newCall(request).execute().body().string();
        System.out.println(output);
    }
}
```

```
//LIST ALL SCHEMAS
request = new Request.Builder()
    .url("http://localhost:8081/subjects")
    .build();

output = client.newCall(request).execute().body().string();
System.out.println(output);

//SHOW ALL VERSIONS OF EMPLOYEE
request = new Request.Builder()
    .url("http://localhost:8081/subjects/Employee/versions/")
    .build();

output = client.newCall(request).execute().body().string();
System.out.println(output);

//SHOW VERSION 2 OF EMPLOYEE
request = new Request.Builder()
    .url("http://localhost:8081/subjects/Employee/versions/2")
    .build();

output = client.newCall(request).execute().body().string();
System.out.println(output);

//SHOW THE SCHEMA WITH ID 3
request = new Request.Builder()
    .url("http://localhost:8081/schemas/ids/3")
    .build();

output = client.newCall(request).execute().body().string();
System.out.println(output);

//SHOW THE LATEST VERSION OF EMPLOYEE 2
request = new Request.Builder()
    .url("http://localhost:8081/subjects/Employee/versions/latest")
    .build();

output = client.newCall(request).execute().body().string();
```

```
System.out.println(output);

//CHECK IF SCHEMA IS REGISTERED
request = new Request.Builder()
    .post(RequestBody.create(SCHEMA_CONTENT, EMPLOYEE_SCHEMA))
    .url("http://localhost:8081/subjects/Employee")
    .build();

output = client.newCall(request).execute().body().string();
System.out.println(output);

//TEST COMPATIBILITY
request = new Request.Builder()
    .post(RequestBody.create(SCHEMA_CONTENT, EMPLOYEE_SCHEMA))
    .url("http://localhost:8081/compatibility/subjects/Employee/versions/latest")
    .build();

output = client.newCall(request).execute().body().string();
System.out.println(output);

// TOP LEVEL CONFIG
request = new Request.Builder()
    .url("http://localhost:8081/config")
    .build();

output = client.newCall(request).execute().body().string();
System.out.println(output);

// SET TOP LEVEL CONFIG
// VALUES are none, backward, forward and full
request = new Request.Builder()
    .put(RequestBody.create(SCHEMA_CONTENT, "{ \"compatibility\": \"none\" }"))
    .url("http://localhost:8081/config")
    .build();

output = client.newCall(request).execute().body().string();
System.out.println(output);
```



```
// SET CONFIG FOR EMPLOYEE
// VALUES are none, backward, forward and full
request = new Request.Builder()
    .put(RequestBody.create(SCHEMA_CONTENT, "{\\"compatibility\\": \\"backward\\"}"))
    .url("http://localhost:8081/config/Employee")
    .build();

output = client.newCall(request).execute().body().string();
System.out.println(output);

}
}
```

I suggest running the example and trying to force incompatible schemas to the Schema Registry and note the behavior for the various compatibility settings.

Running Schema Registry

```
$ cat ~/tools/confluent-3.2.1/etc/schema-registry/schema-registry.properties
```

```
listeners=http://0.0.0.0:8081
kafkastore.connection.url=localhost:2181
kafkastore.topic=_schemas
debug=false
```

```
~/tools/confluent-3.2.1/bin/schema-registry-start ~/tools/confluent-3.2.1/etc/schema-registry/schema-registry.properties
```

Writing Consumers and Producers that use Kafka Avro Serializers and the Schema Registry

Now let's cover writing consumers and producers that use Kafka Avro Serializers which in turn use the Schema Registry and Avro.

We will need to start up the Schema Registry server pointing to our Zookeeper cluster. Then we will need to import the Kafka Avro Serializer and Avro Jars into our gradle project. You will then need to configure the Producer to use Schema Registry and the `KafkaAvroSerializer`. To write the consumer, you will need to configure it to use Schema Registry and to use the `KafkaAvroDeserializer`.

Here is our build file which shows the Avro jar files and such that we need.

Gradle build file for Kafka Avro Serializer examples

```
plugins {  
    id "com.commercehub.gradle.plugin.avro" version "0.9.0"  
}  
  
group 'cloudurable'  
version '1.0-SNAPSHOT'  
apply plugin: 'java'  
sourceCompatibility = 1.8  
  
dependencies {  
    compile "org.apache.avro:avro:1.8.1"  
    compile 'com.squareup.okhttp3:okhttp:3.7.0'  
    testCompile 'junit:junit:4.11'  
    compile 'org.apache.kafka:kafka-clients:0.10.2.0'  
    compile 'io.confluent:kafka-avro-serializer:3.2.1'  
}  
repositories {  
    jcenter()  
    mavenCentral()  
    maven {  
        url "http://packages.confluent.io/maven/"  
    }  
}  
avro {  
    createSetters = false  
    fieldVisibility = "PRIVATE"  
}
```

Notice that we include the Kafka Avro Serializer lib (`io.confluent:kafka-avro-serializer:3.2.1`) and the Avro lib (`org.apache.avro:avro:1.8.1`).

To learn more about the Gradle Avro plugin, please read this article on using Avro (<http://cloudurable.com/blog/avro/index.html>).

Writing a Producer

Next, let's write the Producer as follows.

Producer that uses Kafka Avro Serialization and Kafka Registry

```
package com.cloudurable.kafka.schema;

import com.cloudurable.phonebook.Employee;
import com.cloudurable.phonebook.PhoneNumber;
import io.confluent.kafka.serializers.KafkaAvroSerializerConfig;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.common.serialization.LongSerializer;
import io.confluent.kafka.serializers.KafkaAvroSerializer;

import java.util.Properties;
import java.util.stream.IntStream;

public class AvroProducer {

    private static Producer<Long, Employee> createProducer() {
        Properties props = new Properties();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
        props.put(ProducerConfig.CLIENT_ID_CONFIG, "AvroProducer");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
            LongSerializer.class.getName());

        // Configure the KafkaAvroSerializer.
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
            KafkaAvroSerializer.class.getName());

        // Schema Registry location.
        props.put(KafkaAvroSerializerConfig.SCHEMA_REGISTRY_URL_CONFIG,
            "http://localhost:8081");

        return new KafkaProducer<>(props);
    }

    private final static String TOPIC = "new-employees";

    public static void main(String... args) {

        Producer<Long, Employee> producer = createProducer();
```

```
Employee bob = Employee.newBuilder().setAge(35)
    .setFirstName("Bob")
    .setLastName("Jones")
    .setPhoneNumber(
        PhoneNumber.newBuilder()
            .setAreaCode("301")
            .setCountryCode("1")
            .setPrefix("555")
            .setNumber("1234")
            .build())
    .build();

IntStream.range(1, 100).forEach(index->{
    producer.send(new ProducerRecord<>(TOPIC, 1L * index, bob));

});

producer.flush();
producer.close();
}

}
```

Notice that we configure the schema registry and the `KafkaAvroSerializer` as part of the Producer setup.

```
// Configure the KafkaAvroSerializer.
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
    KafkaAvroSerializer.class.getName());

// Schema Registry location.      props.put(KafkaAvroSerializerConfig.SCHEMA_REGISTRY_URL_CONFIG,
    "http://localhost:8081");
```

Then we use the Producer as expected.

Writing a Consumer

Next we have to write the Consumer.

Consumer that uses Kafka Avro Serialization and Schema Registry

```
package com.cloudurable.kafka.schema;

import com.cloudurable.phonebook.Employee;
import io.confluent.kafka.serializers.KafkaAvroDeserializer;
import io.confluent.kafka.serializers.KafkaAvroDeserializerConfig;
import org.apache.kafka.clients.consumer.Consumer;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.kafka.common.serialization.LongDeserializer;

import java.util.Collections;
import java.util.Properties;
import java.util.stream.IntStream;

public class AvroConsumer {

    private final static String BOOTSTRAP_SERVERS = "localhost:9092";
    private final static String TOPIC = "new-employees";

    private static Consumer<Long, Employee> createConsumer() {
        Properties props = new Properties();
        props.put(ConsumerConfig.BootstrapServersConfig, BOOTSTRAP_SERVERS);
        props.put(ConsumerConfig.GroupIdConfig, "KafkaExampleAvroConsumer");
        props.put(ConsumerConfig.KeyDeserializerClassConfig,
            LongDeserializer.class.getName());

        //Use Kafka Avro Deserializer.
        props.put(ConsumerConfig.ValueDeserializerClassConfig,
            KafkaAvroDeserializer.class.getName()); //<-----

        //Use Specific Record or else you get Avro GenericRecord.
        props.put(KafkaAvroDeserializerConfig.SpecificAvroReaderConfig, "true");

        //Schema registry location.
        props.put(KafkaAvroDeserializerConfig.SchemaRegistryUrlConfig,
            "http://localhost:8081"); //<----- Run Schema Registry on 8081
```

```
        return new KafkaConsumer<>(props);
    }

    public static void main(String... args) {

        final Consumer<Long, Employee> consumer = createConsumer();
        consumer.subscribe(Collections.singletonList(TOPIC));

        IntStream.range(1, 100).forEach(index -> {

            final ConsumerRecords<Long, Employee> records =
                consumer.poll(100);

            if (records.count() == 0) {
                System.out.println("None found");
            } else records.forEach(record -> {

                Employee employeeRecord = record.value();

                System.out.printf("%s %d %d %s \n", record.topic(),
                    record.partition(), record.offset(), employeeRecord);
            });
        });
    }
}
```

Notice just like the producer we have to tell the consumer where to find the Registry, and we have to configure the Kafka Avro Deserializer.

Configuring Schema Registry for Consumer


```
//Use Kafka Avro Deserializer.

props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
          KafkaAvroDeserializer.class.getName());

//Use Specific Record or else you get Avro GenericRecord.
props.put(KafkaAvroDeserializerConfig.SPECIFIC_AVRO_READER_CONFIG, "true");

//Schema registry location.          props.put(KafkaAvroDeserializerConfig.SCHEMA_REGISTRY_URL_CONFIG,
          "http://localhost:8081"); //<----- Run Schema Registry on 8081
```

An additional step is we have to tell it to use the generated version of the `Employee` object. If we did not, then it would use `Avro GenericRecord` instead of our generated `Employee` object, which is a `SpecificRecord`. To learn more about using `GenericRecord` and generating code from Avro read the Avro Kafka tutorial (<http://cloudurable.com/blog/avro/index.html>) as it has examples of both.

To run the above example, you need to startup Kafka and Zookeeper. To learn how to do this if you have not done it before see Kafka Tutorial (<http://cloudurable.com/blog/kafka-tutorial-v1/index.html>). Essentially, there is a startup script for Kafka and ZooKeeper like there was with the Schema Registry and there is default configuration, you pass the default configuration to the startup scripts, and Kafka is running locally on your machine.

Running Zookeeper and Kafka

```
kafka/bin/zookeeper-server-start.sh kafka/config/zookeeper.properties &

kafka/bin/kafka-server-start.sh kafka/config/server.properties
```

Conclusion

Confluent provides Schema Registry to manage Avro Schemas for Kafka Consumers and Producers. Avro provides Schema Migration which is necessary for streaming and big data architectures.

Confluent uses Schema compatibility checks to see if the Producer's schema and Consumer's schemas are compatible and to do Schema evolution if needed. You use `KafkaAvroSerializer` from the Producer and point to the Schema Registry. You use `KafkaAvroDeserializer` from Consumer and point to the Schema Registry.

Related slide deck



Cassandra / Kafka Support in EC2/AWS. [Kafka Training](#), [Kafka Consulting](#)



Avro

Kafka & Avro: Confluent Schema Registry

Managing Record Schema i
Kafka

1 of 29

Kafka and Avro with Confluent Schema Registry ([//www.slideshare.net/JeanPaulAzar1/kafka-and-avro-with-confluent-schema-registry](http://www.slideshare.net/JeanPaulAzar1/kafka-and-avro-with-confluent-schema-registry)) from **Jean-Paul Azar** (<https://www.slideshare.net/JeanPaulAzar1>)

About Cloudurable

Cloudurable™: streamline DevOps for Cassandra and Kafka running on AWS (<http://cloudurable.com/>) provides AMIs, CloudWatch Monitoring, CloudFormation templates and monitoring tools to support Cassandra and Kafka in production running in EC2 (<http://cloudurable.com/services/index.html>).

Please take some time to read the Advantage of using Cloudurable™ (<http://cloudurable.com/advantages/index.html>).

Related content

- What is Kafka? (<http://cloudurable.com/blog/what-is-kafka/index.html>)
- Kafka Architecture (<http://cloudurable.com/blog/kafka-architecture/index.html>)
- Kafka Topic Architecture (<http://cloudurable.com/blog/kafka-architecture-topics/index.html>)
- Kafka Consumer Architecture (<http://cloudurable.com/blog/kafka-architecture-consumers/index.html>)
- Kafka Producer Architecture (<http://cloudurable.com/blog/kafka-architecture-producers/index.html>)
- Kafka Architecture and low level design (<http://cloudurable.com/blog/kafka-architecture-low-level/index.html>)
- Kafka and Schema Registry (<http://cloudurable.com/blog/kafka-avro-schema-registry/index.html>)
- Kafka and Avro (<http://cloudurable.com/blog/avro/index.html>)
- Kafka Ecosystem (<http://cloudurable.com/blog/kafka-ecosystem/index.html>)
- Kafka vs. JMS (<http://cloudurable.com/blog/kafka-vs-jms/index.html>)
- Kafka versus Kinesis (<http://cloudurable.com/blog/kinesis-vs-kafka/index.html>)
- Kafka Tutorial: Using Kafka from the command line (<http://cloudurable.com/blog/kafka-tutorial-kafka-from-command-line/index.html>)
- Kafka Tutorial: Kafka Broker Failover and Consumer Failover (<http://cloudurable.com/blog/kafka-tutorial-kafka-failover-kafka-cluster/index.html>)
- Kafka Tutorial (<http://cloudurable.com/ppt/kafka-tutorial-cloudurable-v2.pdf>)
- Kafka Tutorial: Writing a Kafka Producer example in Java (<http://cloudurable.com/blog/kafka-tutorial-kafka-producer/index.html>)
- Kafka Tutorial: Writing a Kafka Consumer example in Java (<http://cloudurable.com/blog/kafka-tutorial-kafka-consumer/index.html>)
- Kafka Architecture: Log Compaction (<http://cloudurable.com/blog/kafka-architecture-log-compaction/index.html>)
- Kafka Architecture: Low-Level PDF Slides (<http://cloudurable.com/ppt/4-kafka-detailed-architecture.pdf>)

About Cloudurable

We hope you enjoyed this article. Please provide feedback (<http://cloudurable.com/contact/index.html>). Cloudurable provides Kafka training (<http://cloudurable.com/kafka-training/index.html>), Kafka consulting (<http://cloudurable.com/kafka-aws-consulting/index.html>), Kafka support (http://cloudurable.com/subscription_support/index.html) and helps setting up Kafka clusters in AWS (<http://cloudurable.com/services/index.html>).

Cloudurable provides:

- Kafka training (<http://cloudurable.com/kafka-training/index.html>)
- Kafka consulting (<http://cloudurable.com/kafka-aws-consulting/index.html>)
- Cassandra training (<http://cloudurable.com/cassandra-course/index.html>)
- Cassandra consulting (<http://cloudurable.com/kafka-aws-consulting/index.html>)
- Spark training (<http://cloudurable.com/spark-aws-emr-training/index.html>)
- Spark consulting (<http://cloudurable.com/spark-aws-emr-consulting/index.html>)

Check out our new GoLang course. We provide onsite Go Lang training which is instructor led (<http://cloudurable.com/golang-onsite-instructor-led-training/index.html>).



Share

Tweet

Like 49

Share

SEARCH

Search



SHARE

Tweet



Share

facebook

49

Like

Share

FOLLOW

Follow @cloudurable



Follow

64

facebook

CATEGORIES

amazon-ebs (1) (<http://cloudurable.com/categories/amazon-ebs/index.html>)

amazon-ec2 (1) (<http://cloudurable.com/categories/amazon-ec2/index.html>)

amazon-vpc (1) (<http://cloudurable.com/categories/amazon-vpc/index.html>)

ansible (4) (<http://cloudurable.com/categories/ansible/index.html>)

avro (2) (<http://cloudurable.com/categories/avro/index.html>)

aws (4) (<http://cloudurable.com/categories/aws/index.html>)

aws-cassandra (6) (<http://cloudurable.com/categories/aws-cassandra/index.html>)

aws-command-line (1) (<http://cloudurable.com/categories/aws-command-line/index.html>)

cassandra (12) (<http://cloudurable.com/categories/cassandra/index.html>)

cassandra-aws (3) (<http://cloudurable.com/categories/cassandra-aws/index.html>)

cassandra-cluster (1) (<http://cloudurable.com/categories/cassandra-cluster/index.html>)

cassandra-database (2) (<http://cloudurable.com/categories/cassandra-database/index.html>)

cassandra-training (5) (<http://cloudurable.com/categories/cassandra-training/index.html>)

cassandra-tutorial (5) (<http://cloudurable.com/categories/cassandra-tutorial/index.html>)

cloud (4) (<http://cloudurable.com/categories/cloud/index.html>)

cloudformation (1) (<http://cloudurable.com/categories/cloudformation/index.html>)

cloudurable (15) (<http://cloudurable.com/categories/cloudurable/index.html>)

cluster (1) (<http://cloudurable.com/categories/cluster/index.html>)

devops (16) (<http://cloudurable.com/categories/devops/index.html>)

ebs (3) (<http://cloudurable.com/categories/ebs/index.html>)

ec2 (1) (<http://cloudurable.com/categories/ec2/index.html>)

kafka (13) (<http://cloudurable.com/categories/kafka/index.html>)

kafka-advanced-consumers (1) (<http://cloudurable.com/categories/kafka-advanced-consumers/index.html>)

kafka-architecture (13) (<http://cloudurable.com/categories/kafka-architecture/index.html>)

kafka-avro-serialization (1) (<http://cloudurable.com/categories/kafka-avro-serialization/index.html>)

kafka-consulting (2) (<http://cloudurable.com/categories/kafka-consulting/index.html>)

kafka-consumer (1) (<http://cloudurable.com/categories/kafka-consumer/index.html>)

kafka-consumers (1) (<http://cloudurable.com/categories/kafka-consumers/index.html>)

kafka-ecosystem (1) (<http://cloudurable.com/categories/kafka-ecosystem/index.html>)

kafka-schema-registry (1) (<http://cloudurable.com/categories/kafka-schema-registry/index.html>)

kafka-training (23) (<http://cloudurable.com/categories/kafka-training/index.html>)

kafka-tutorial (20) (<http://cloudurable.com/categories/kafka-tutorial/index.html>)

kaka-replication (1) (<http://cloudurable.com/categories/kaka-replication/index.html>)

kinesis (1) (<http://cloudurable.com/categories/kinesis/index.html>)

kinesis-consulting (1) (<http://cloudurable.com/categories/kinesis-consulting/index.html>)

linux (1) (<http://cloudurable.com/categories/linux/index.html>)

metricsd (1) (<http://cloudurable.com/categories/metricsd/index.html>)

microservices (3) (<http://cloudurable.com/categories/microservices/index.html>)

nodetool (1) (<http://cloudurable.com/categories/nodetool/index.html>)

schema-registry (1) (<http://cloudurable.com/categories/schema-registry/index.html>)

smack (3) (<http://cloudurable.com/categories/smack/index.html>)

spark (3) (<http://cloudurable.com/categories/spark/index.html>)

spark--cassandra (1) (<http://cloudurable.com/categories/spark--cassandra/index.html>)

spark--kafka (1) (<http://cloudurable.com/categories/spark--kafka/index.html>)

spark-training (3) (<http://cloudurable.com/categories/spark-training/index.html>)

spark-tutorial (3) (<http://cloudurable.com/categories/spark-tutorial/index.html>)

ssh (1) (<http://cloudurable.com/categories/ssh/index.html>)

ssh-config (1) (<http://cloudurable.com/categories/ssh-config/index.html>)

ssl (1) (<http://cloudurable.com/categories/ssl/index.html>)

systemd (1) (<http://cloudurable.com/categories/systemd/index.html>)

tls (1) (<http://cloudurable.com/categories/tls/index.html>)

vagrant (5) (<http://cloudurable.com/categories/vagrant/index.html>)

TAGS



🔗 [AKKA \(HTTP://CLOUDURABLE.COM/TAGS/AKKA/INDEX.HTML\)](http://cloudurable.com/tags/akka/index.html)

🔗 [AKKA-CONSULTING \(HTTP://CLOUDURABLE.COM/TAGS/AKKA-CONSULTING/INDEX.HTML\)](http://cloudurable.com/tags/akka-consulting/index.html)

🔗 [AMAZON-EBS \(HTTP://CLOUDURABLE.COM/TAGS/AMAZON-EBS/INDEX.HTML\)](http://cloudurable.com/tags/amazon-ebs/index.html)

🔗 [AMAZON-EC2 \(HTTP://CLOUDURABLE.COM/TAGS/AMAZON-EC2/INDEX.HTML\)](http://cloudurable.com/tags/amazon-ec2/index.html)

🔗 [AMI \(HTTP://CLOUDURABLE.COM/TAGS/AMI/INDEX.HTML\)](http://cloudurable.com/tags/ami/index.html)

🔗 [ANSIBLE \(HTTP://CLOUDURABLE.COM/TAGS/ANSIBLE/INDEX.HTML\)](http://cloudurable.com/tags/ansible/index.html)

🔗 [AVRO \(HTTP://CLOUDURABLE.COM/TAGS/AVRO/INDEX.HTML\)](http://cloudurable.com/tags/avro/index.html)

🔗 [AVRO-KAFKA \(HTTP://CLOUDURABLE.COM/TAGS/AVRO-KAFKA/INDEX.HTML\)](http://cloudurable.com/tags/avro-kafka/index.html)

🔗 [AWS \(HTTP://CLOUDURABLE.COM/TAGS/AWS/INDEX.HTML\)](http://cloudurable.com/tags/aws/index.html)

🔗 [AWS-CASSANDRA \(HTTP://CLOUDURABLE.COM/TAGS/AWS-CASSANDRA/INDEX.HTML\)](http://cloudurable.com/tags/aws-cassandra/index.html)

🔗 [AWS-COMMAND-LINE \(HTTP://CLOUDURABLE.COM/TAGS/AWS-COMMAND-LINE/INDEX.HTML\)](http://cloudurable.com/tags/aws-command-line/index.html)

🔗 [CASSANDRA \(HTTP://CLOUDURABLE.COM/TAGS/CASSANDRA/INDEX.HTML\)](http://cloudurable.com/tags/cassandra/index.html)

🔗 [CASSANDRA-ARCHITECTURE \(HTTP://CLOUDURABLE.COM/TAGS/CASSANDRA-ARCHITECTURE/INDEX.HTML\)](http://cloudurable.com/tags/cassandra-architecture/index.html)

🔗 [CASSANDRA-AWS \(HTTP://CLOUDURABLE.COM/TAGS/CASSANDRA-AWS/INDEX.HTML\)](http://cloudurable.com/tags/cassandra-aws/index.html)

🔗 [CASSANDRA-CLOUD \(HTTP://CLOUDURABLE.COM/TAGS/CASSANDRA-CLOUD/INDEX.HTML\)](http://cloudurable.com/tags/cassandra-cloud/index.html)

🔗 [CASSANDRA-CLUSTER \(HTTP://CLOUDURABLE.COM/TAGS/CASSANDRA-CLUSTER/INDEX.HTML\)](http://cloudurable.com/tags/cassandra-cluster/index.html)

🔗 [CASSANDRA-DATABASE \(HTTP://CLOUDURABLE.COM/TAGS/CASSANDRA-DATABASE/INDEX.HTML\)](http://cloudurable.com/tags/cassandra-database/index.html)

🔗 [CASSANDRA-DBA \(HTTP://CLOUDURABLE.COM/TAGS/CASSANDRA-DBA/INDEX.HTML\)](http://cloudurable.com/tags/cassandra-dba/index.html)

🔗 [CASSANDRA-DEVOPS \(HTTP://CLOUDURABLE.COM/TAGS/CASSANDRA-DEVOPS/INDEX.HTML\)](http://cloudurable.com/tags/cassandra-devops/index.html)

🔗 [CASSANDRA-OS-SYSTEM-MEMORY \(HTTP://CLOUDURABLE.COM/TAGS/CASSANDRA-OS-SYSTEM-MEMORY/INDEX.HTML\)](http://cloudurable.com/tags/cassandra-os-system-memory/index.html)

🔗 [CASSANDRA-TRAINING \(HTTP://CLOUDURABLE.COM/TAGS/CASSANDRA-TRAINING/INDEX.HTML\)](http://cloudurable.com/tags/cassandra-training/index.html)

🔗 [CASSANDRA-TUTORIAL \(HTTP://CLOUDURABLE.COM/TAGS/CASSANDRA-TUTORIAL/INDEX.HTML\)](http://cloudurable.com/tags/cassandra-tutorial/index.html)

🔗 [CLOUD \(HTTP://CLOUDURABLE.COM/TAGS/CLOUD/INDEX.HTML\)](http://cloudurable.com/tags/cloud/index.html)

🔗 [CLOUDFORMATION \(HTTP://CLOUDURABLE.COM/TAGS/CLOUDFORMATION/INDEX.HTML\)](http://cloudurable.com/tags/cloudformation/index.html)

🔗 [CLOUDFORMATION-TUTORIAL \(HTTP://CLOUDURABLE.COM/TAGS/CLOUDFORMATION-TUTORIAL/INDEX.HTML\)](http://cloudurable.com/tags/cloudformation-tutorial/index.html)

🔗 [CLOUDURABLE \(HTTP://CLOUDURABLE.COM/TAGS/CLOUDURABLE/INDEX.HTML\)](http://cloudurable.com/tags/cloudurable/index.html)

🔗 [CLUSTER \(HTTP://CLOUDURABLE.COM/TAGS/CLUSTER/INDEX.HTML\)](http://cloudurable.com/tags/cluster/index.html)

🔗 [COMPUTE \(HTTP://CLOUDURABLE.COM/TAGS/COMPUTE/INDEX.HTML\)](http://cloudurable.com/tags/compute/index.html)

🔗 [CONSUMERS \(HTTP://CLOUDURABLE.COM/TAGS/CONSUMERS/INDEX.HTML\)](http://cloudurable.com/tags/consumers/index.html)

🔗 [DBA \(HTTP://CLOUDURABLE.COM/TAGS/DBA/INDEX.HTML\)](http://cloudurable.com/tags/dba/index.html)

🔗 [DEVOPS \(HTTP://CLOUDURABLE.COM/TAGS/DEVOPS/INDEX.HTML\)](http://cloudurable.com/tags/devops/index.html)

🔗 [EBS \(HTTP://CLOUDURABLE.COM/TAGS/EBS/INDEX.HTML\)](http://cloudurable.com/tags/ebs/index.html)

🔗 [EC2 \(HTTP://CLOUDURABLE.COM/TAGS/EC2/INDEX.HTML\)](http://cloudurable.com/tags/ec2/index.html)

🔗 [EC2-INSTANCE-STORE \(HTTP://CLOUDURABLE.COM/TAGS/EC2-INSTANCE-STORE/INDEX.HTML\)](http://cloudurable.com/tags/ec2-instance-store/index.html)

🔗 [ECU \(HTTP://CLOUDURABLE.COM/TAGS/ECU/INDEX.HTML\)](http://cloudurable.com/tags/ecu/index.html)

🔗 [FAILOVER \(HTTP://CLOUDURABLE.COM/TAGS/FAILOVER/INDEX.HTML\)](http://cloudurable.com/tags/failover/index.html)

🔗 [IMAGES \(HTTP://CLOUDURABLE.COM/TAGS/IMAGES/INDEX.HTML\)](http://cloudurable.com/tags/images/index.html)

🔗 [INSTANCES \(HTTP://CLOUDURABLE.COM/TAGS/INSTANCES/INDEX.HTML\)](http://cloudurable.com/tags/instances/index.html)

🔗 [JMS \(HTTP://CLOUDURABLE.COM/TAGS/JMS/INDEX.HTML\)](http://cloudurable.com/tags/jms/index.html)

🔗 [KAFKA \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA/INDEX.HTML\)](http://cloudurable.com/tags/kafka/index.html)

🔗 [KAFKA-ADVANCED-CONSUMERS \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-ADVANCED-CONSUMERS/INDEX.HTML\)](http://cloudurable.com/tags/kafka-advanced-consumers/index.html)

🔗 [KAFKA-ADVANCED-PRODUCERS \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-ADVANCED-PRODUCERS/INDEX.HTML\)](http://cloudurable.com/tags/kafka-advanced-producers/index.html)

🔗 [KAFKA-ARCHITECTURE \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-ARCHITECTURE/INDEX.HTML\)](http://cloudurable.com/tags/kafka-architecture/index.html)

🔗 [KAFKA-AVRO \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-AVRO/INDEX.HTML\)](http://cloudurable.com/tags/kafka-avro/index.html)

🔗 [KAFKA-CONNECT \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-CONNECT/INDEX.HTML\)](http://cloudurable.com/tags/kafka-connect/index.html)

🔗 [KAFKA-CONSULTING \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-CONSULTING/INDEX.HTML\)](http://cloudurable.com/tags/kafka-consulting/index.html)

🔗 [KAFKA-CONSUMERS \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-CONSUMERS/INDEX.HTML\)](http://cloudurable.com/tags/kafka-consumers/index.html)

🔗 [KAFKA-CONSUMERS-ADVANCED \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-CONSUMERS-ADVANCED/INDEX.HTML\)](http://cloudurable.com/tags/kafka-consumers-advanced/index.html)

🔗 [KAFKA-ECOSYSTEM \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-ECOSYSTEM/INDEX.HTML\)](http://cloudurable.com/tags/kafka-ecosystem/index.html)

🔗 [KAFKA-LOG-COMPACTION \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-LOG-COMPACTION/INDEX.HTML\)](http://cloudurable.com/tags/kafka-log-compaction/index.html)

🔗 [KAFKA-REST-PROXY \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-REST-PROXY/INDEX.HTML\)](http://cloudurable.com/tags/kafka-rest-proxy/index.html)

🔗 [KAFKA-STREAMS \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-STREAMS/INDEX.HTML\)](http://cloudurable.com/tags/kafka-streams/index.html)

🔗 [KAFKA-TRAINING \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-TRAINING/INDEX.HTML\)](http://cloudurable.com/tags/kafka-training/index.html)

🔗 [KAFKA-TUTORIAL \(HTTP://CLOUDURABLE.COM/TAGS/KAFKA-TUTORIAL/INDEX.HTML\)](http://cloudurable.com/tags/kafka-tutorial/index.html)

🔗 [KINESIS \(HTTP://CLOUDURABLE.COM/TAGS/KINESIS/INDEX.HTML\)](http://cloudurable.com/tags/kinesis/index.html)

🔗 [KINESIS-CONSULTING \(HTTP://CLOUDURABLE.COM/TAGS/KINESIS-CONSULTING/INDEX.HTML\)](http://cloudurable.com/tags/kinesis-consulting/index.html)

🔗 [LINUX \(HTTP://CLOUDURABLE.COM/TAGS/LINUX/INDEX.HTML\)](http://cloudurable.com/tags/linux/index.html)

🔗 [METRICSD \(HTTP://CLOUDURABLE.COM/TAGS/METRICSD/INDEX.HTML\)](http://cloudurable.com/tags/metricSD/index.html)

🔗 [MICROSERVICES \(HTTP://CLOUDURABLE.COM/TAGS/MICROSERVICES/INDEX.HTML\)](http://cloudurable.com/tags/microservices/index.html)

🔗 [MICROSERVICES-ARCHITECTURE \(HTTP://CLOUDURABLE.COM/TAGS/MICROSERVICES-ARCHITECTURE/INDEX.HTML\)](http://cloudurable.com/tags/microservices-architecture/index.html)

[NAT \(HTTP://CLOUDURABLE.COM/TAGS/NAT/INDEX.HTML\)](http://cloudurable.com/tags/nat/index.html)[NODETOOL \(HTTP://CLOUDURABLE.COM/TAGS/NODETOOL/INDEX.HTML\)](http://cloudurable.com/tags/node-tool/index.html)[NUMA \(HTTP://CLOUDURABLE.COM/TAGS/NUMA/INDEX.HTML\)](http://cloudurable.com/tags/numa/index.html)[PRODUCERS \(HTTP://CLOUDURABLE.COM/TAGS/PRODUCERS/INDEX.HTML\)](http://cloudurable.com/tags/producers/index.html)[QBIT \(HTTP://CLOUDURABLE.COM/TAGS/QBIT/INDEX.HTML\)](http://cloudurable.com/tags/qbit/index.html)[RAM \(HTTP://CLOUDURABLE.COM/TAGS/RAM/INDEX.HTML\)](http://cloudurable.com/tags/ram/index.html)[REAKT \(HTTP://CLOUDURABLE.COM/TAGS/REAKT/INDEX.HTML\)](http://cloudurable.com/tags/reakt/index.html)[REPLICATION \(HTTP://CLOUDURABLE.COM/TAGS/REPLICATION/INDEX.HTML\)](http://cloudurable.com/tags/replication/index.html)[SCHEMA-REGISTRY \(HTTP://CLOUDURABLE.COM/TAGS/SCHEMA-REGISTRY/INDEX.HTML\)](http://cloudurable.com/tags/schema-registry/index.html)[SMACK \(HTTP://CLOUDURABLE.COM/TAGS/SMACK/INDEX.HTML\)](http://cloudurable.com/tags/smack/index.html)[SPARK \(HTTP://CLOUDURABLE.COM/TAGS/SPARK/INDEX.HTML\)](http://cloudurable.com/tags/spark/index.html)[SPARK--CASSANDRA \(HTTP://CLOUDURABLE.COM/TAGS/SPARK--CASSANDRA/INDEX.HTML\)](http://cloudurable.com/tags/spark--cassandra/index.html)[SPARK-TRAINING \(HTTP://CLOUDURABLE.COM/TAGS/SPARK-TRAINING/INDEX.HTML\)](http://cloudurable.com/tags/spark-training/index.html)[SPARK-TUTORIAL \(HTTP://CLOUDURABLE.COM/TAGS/SPARK-TUTORIAL/INDEX.HTML\)](http://cloudurable.com/tags/spark-tutorial/index.html)[SSH \(HTTP://CLOUDURABLE.COM/TAGS/SSH/INDEX.HTML\)](http://cloudurable.com/tags/ssh/index.html)[SSH-CONFIG \(HTTP://CLOUDURABLE.COM/TAGS/SSH-CONFIG/INDEX.HTML\)](http://cloudurable.com/tags/ssh-config/index.html)[SSL \(HTTP://CLOUDURABLE.COM/TAGS/SSL/INDEX.HTML\)](http://cloudurable.com/tags/ssl/index.html)[SYSTEMD \(HTTP://CLOUDURABLE.COM/TAGS/SYSTEMD/INDEX.HTML\)](http://cloudurable.com/tags/systemd/index.html)[TLS \(HTTP://CLOUDURABLE.COM/TAGS/TLS/INDEX.HTML\)](http://cloudurable.com/tags/tls/index.html)[VAGRANT \(HTTP://CLOUDURABLE.COM/TAGS/VAGRANT/INDEX.HTML\)](http://cloudurable.com/tags/vagrant/index.html)[VCPU \(HTTP://CLOUDURABLE.COM/TAGS/VCPU/INDEX.HTML\)](http://cloudurable.com/tags/vcpu/index.html)[VPC \(HTTP://CLOUDURABLE.COM/TAGS/VPC/INDEX.HTML\)](http://cloudurable.com/tags/vpc/index.html)[WHAT-IS-KAFKA \(HTTP://CLOUDURABLE.COM/TAGS/WHAT-IS-KAFKA/INDEX.HTML\)](http://cloudurable.com/tags/what-is-kafka/index.html)

Apache Spark Training (<http://cloudurable.com/spark-training/index.html>)

Kafka Tutorial (<http://cloudurable.com/blog/kafka-tutorial/index.html>)

Akka Consulting (<http://cloudurable.com/akka-consulting/index.html>)

Cassandra Training (<http://cloudurable.com/cassandra-course/index.html>)
AWS Cassandra Database Support (http://cloudurable.com/subscription_support_benefits_cassandra/index.html)
Kafka Support Pricing (http://cloudurable.com/subscription_support/index.html?q=kafka)
Cassandra Database Support Pricing (http://cloudurable.com/subscription_support/index.html?q=cassandra)
Non-stop Cassandra (<http://cloudurable.com/cloudurable-cassandra-watchdog/index.html?q=cassandra>)
Watchdog (<http://cloudurable.com/cloudurable-cassandra-watchdog/index.html?q=watchdog>)
Advantages of using Cloudurable™ (<http://cloudurable.com/advantages/index.html>)
Cassandra Consulting (<http://cloudurable.com/service-quick-start-mentoring-cassandra-or-kafka-aws-ec2/index.html>)
Cloudurable™| Guide to AWS Cassandra Deploy (<http://cloudurable.com/ppt/amazon-cassandra.pdf>)
Cloudurable™| AWS Cassandra Guidelines and Notes (<http://cloudurable.com/ppt/amazon-cassandra-notes.pdf>)
Free guide to deploying Cassandra on AWS (<http://cloudurable.com/cassandra-aws-consulting/index.html>)
Kafka Training (<http://cloudurable.com/kafka-training/index.html>)
Kafka Consulting (<http://cloudurable.com/kafka-aws-consulting/index.html>)
DynamoDB Training (<http://cloudurable.com/dynamodb-training/index.html>)
DynamoDB Consulting (<http://cloudurable.com/dynamodb-consulting/index.html>)
Kinesis Training (<http://cloudurable.com/kinesis-training/index.html>)
Kinesis Consulting (<http://cloudurable.com/kinesis-consulting/index.html>)
Kafka Tutorial PDF (<http://cloudurable.com/blog/kafka-tutorial-v1/index.html>)
Redis Consulting (<http://cloudurable.com/redis-consulting/index.html>)
Redis Training (<http://cloudurable.com/redis-onsite-instructor-led-training/index.html>)
ElasticSearch / ELK Consulting (<http://cloudurable.com/elk-consulting/index.html>)
ElasticSearch Training (<http://cloudurable.com/elasticsearch-onsite-instructor-led-training/index.html>)
InfluxDB/TICK Training (<http://cloudurable.com/influxdb-onsite-instructor-led-training/index.html>) TICK Consulting (<http://cloudurable.com/tick-consulting/index.html>)

ABOUT US

Cloudurable™: Leader in AWS cloud computing for Kafka™, Cassandra™ Database, Apache Spark, AWS CloudFormation™ DevOps. We do **Cassandra training, Apache Spark, Kafka training, Kafka consulting** and **cassandra consulting** with a focus on AWS and data engineering. (FAQ (<http://cloudurable.com/faq/index.html>))

FOLLOW CLOUDURABLE™




facebook page (<https://www.facebook.com/cloudurable>)

google plus (<https://plus.google.com/116648719730180908239>)

twitter (<https://twitter.com/cloudurable>)
linkedin (<https://www.linkedin.com/company/17964258/>)

Why Cloudurable™?
Advantage of using Cloudurable™ (<http://cloudurable.com/advantages/index.html>)
About Cloudurable™?
About Cloudurable™ (<http://cloudurable.com/faq/index.html>)
What are the benefits of using subscription support?
Benefits of Subscription Cassandra Support (http://cloudurable.com/subscription_support_benefits_cassandra/index.html)

RECENT POSTS

	KAFKA CONSUMER: ADVANCED CONSUMERS (HTTP://CLOUDURABLE.COM/BLOG/KAFKA-ADVANCED-CONSUMER-1/INDEX.HTML)
	KAFKA TUTORIAL (HTTP://CLOUDURABLE.COM/BLOG/KAFKA-TUTORIAL/INDEX.HTML)
	n/blog/kafka- KAFKA TUTORIAL: CREATING ADVANCED KAFKA PRODUCERS IN JAVA (HTTP://CLOUDURABLE.COM/BLOG/KAFKA-TUTORIAL-KAFKA-PRODUCER-ADVANCED-JAVA-EXAMPLES/INDEX.HTML)
	n/blog/kafka- KAFKA CONSULTING (HTTP://CLOUDURABLE.COM/BLOG/KAFKA-CONSULTING/INDEX.HTML)
	n/blog/kafka-

CONTACT

<http://cloudurable.com/blog/kafka-examples/index.html>
101 California Street
San Francisco
CA 94111
USA
America
(415) 758-1113 (tel:14157581113)

GO TO CONTACT PAGE ([/CONTACT/INDEX.HTML](/contact/index.html))

Copyright © 2015 - 2018, Cloudurable™, all rights reserved. Streamline your Cassandra Database, Apache Spark and Kafka DevOps in AWS. SMACK/Lambda architecture consulting! Spark, Mesos, Akka, Cassandra and Kafka in AWS.

Apache Spark Training (<http://cloudurable.com/spark-training/index.html>), Akka Consulting (<http://cloudurable.com/akka-consulting/index.html>), AWS Cassandra Support (http://cloudurable.com/subscription_support_benefits_cassandra/index.html), Cassandra Training (<http://cloudurable.com/cassandra-course/index.html>), Kafka Training (<http://cloudurable.com/kafka-training/index.html>), Cassandra Consulting (<http://cloudurable.com/service-architecture-analysis-cassandra-or-kafka-aws-ec2/index.html>), Kafka Consulting (<http://cloudurable.com/kafka-aws-consulting/index.html>), Spark Training (<http://cloudurable.com/spark-aws-emr-training/index.html>), Spark Consulting (<http://cloudurable.com/spark-aws-emr-consulting/index.html>), Kafka Tutorial (<http://cloudurable.com/blog/kafka-tutorial-v1/index.html>)

Template by DevCows (<https://github.com/devcows/hugo-universal-theme>)