Alpakka Kafka Test -JAVA

JAVA KAFKA ~ Alpakka Kafka UnitTest .

Kafka

```yaml
version: '3.5'

services:
  zookeeper-1:
    image: confluentinc/cp-zookeeper:5.5.1
    ports:
      - '32181:32181'
    environment:
      ZOOKEEPER_CLIENT_PORT: 32181
      ZOOKEEPER_TICK_TIME: 2000


  kafka-1:
    image: confluentinc/cp-kafka:5.5.1
    ports:
      - '9092:9092'
    depends_on:
      - zookeeper-1
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper-1:32181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:PLAINTEXT,EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
      KAFKA_ADVERTISED_LISTENERS: INTERNAL://kafka-1:29092,EXTERNAL://localhost:9092
      KAFKA_DEFAULT_REPLICATION_FACTOR: 3
      KAFKA_NUM_PARTITIONS: 3


  kafka-2:
    image: confluentinc/cp-kafka:5.5.1
    ports:
      - '9093:9093'
    depends_on:
      - zookeeper-1
    environment:
      KAFKA_BROKER_ID: 2
      KAFKA_ZOOKEEPER_CONNECT: zookeeper-1:32181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:PLAINTEXT,EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
      KAFKA_ADVERTISED_LISTENERS: INTERNAL://kafka-2:29093,EXTERNAL://localhost:9093
      KAFKA_DEFAULT_REPLICATION_FACTOR: 3
      KAFKA_NUM_PARTITIONS: 3


  kafka-3:
    image: confluentinc/cp-kafka:5.5.1
    ports:
      - '9094:9094'
    depends_on:
      - zookeeper-1
    environment:
      KAFKA_BROKER_ID: 3
      KAFKA_ZOOKEEPER_CONNECT: zookeeper-1:32181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:PLAINTEXT,EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
      KAFKA_ADVERTISED_LISTENERS: INTERNAL://kafka-3:29094,EXTERNAL://localhost:9094
      KAFKA_DEFAULT_REPLICATION_FACTOR: 3
      KAFKA_NUM_PARTITIONS: 3

  kafka-ui:
    image: provectuslabs/kafka-ui
    container_name: kafka-ui
    ports:
      - "8989:8080"
    restart: always
    environment:
      - KAFKA_CLUSTERS_0_NAME=local
      - KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=kafka-1:29092,kafka-2:29093,kafka-3:29094
      - KAFKA_CLUSTERS_0_ZOOKEEPER=zookeeper-1:22181
```

kafka ui

docker-compose up -d kafka .

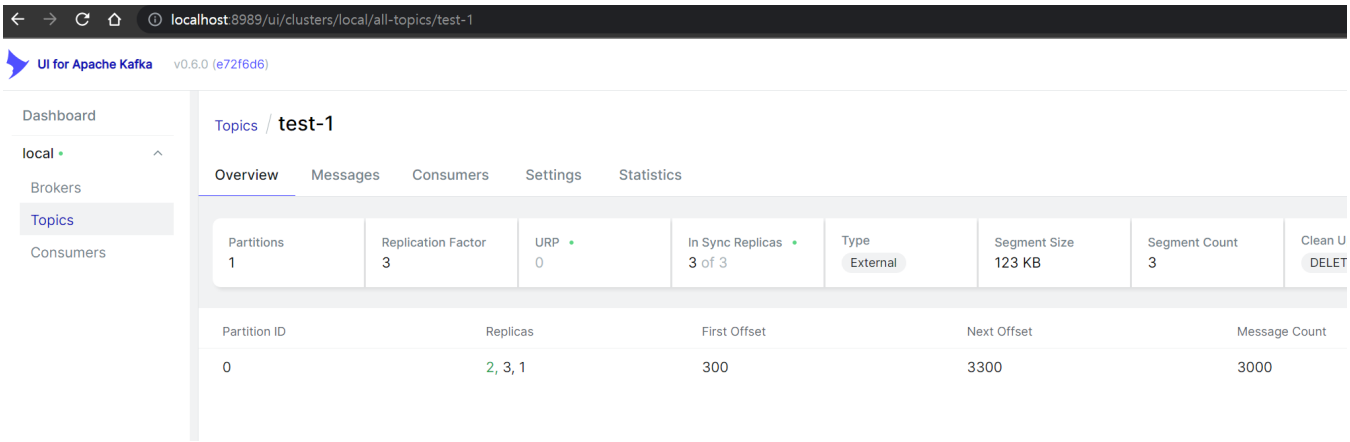
Kafka 100

```
AkkaKafkaTests (com.webn 4 sec 972 ms
    TestKafkaProduce 962 ms
    TestKafkaProduceAndCo 4 sec 10 ms

    ssl.truststore.certificates = null
    ssl.truststore.location = null
    ssl.truststore.password = null
    ssl.truststore.type = JKS
    transaction.timeout.ms = 60000
    transactional.id = null
    value.serializer = class org.apache.kafka.common.serialization.StringSerializer
]
[INFO ] [2023-03-19 19:55:55,123] [akka-spring-demo-akka.actor.default-dispatcher-5] [[Producer clientId=producer-1] I
[INFO ] [2023-03-19 19:55:55,177] [akka-spring-demo-akka.actor.default-dispatcher-5] [Kafka version: 3.1.2]
[INFO ] [2023-03-19 19:55:55,178] [akka-spring-demo-akka.actor.default-dispatcher-5] [Kafka commitId: f8c67dc3ae0a3265
[INFO ] [2023-03-19 19:55:55,178] [akka-spring-demo-akka.actor.default-dispatcher-5] [Kafka startTimeMs: 1679223355175
[INFO ] [2023-03-19 19:55:55,474] [kafka-producer-network-thread | producer-1] [[Producer clientId=producer-1] Cluster
[INFO ] [2023-03-19 19:55:55,476] [kafka-producer-network-thread | producer-1] [[Producer clientId=producer-1] Produce
[INFO ] [2023-03-19 19:55:55,540] [akka-spring-demo-akka.kafka.default-dispatcher-10] [[Producer clientId=producer-1]
[INFO ] [2023-03-19 19:55:55,549] [akka-spring-demo-akka.kafka.default-dispatcher-10] [Metrics scheduler closed]
[INFO ] [2023-03-19 19:55:55,549] [akka-spring-demo-akka.kafka.default-dispatcher-10] [Closing reporter org.apache.kaf
[INFO ] [2023-03-19 19:55:55,549] [akka-spring-demo-akka.kafka.default-dispatcher-10] [Metrics reporters closed]
[INFO ] [2023-03-19 19:55:55,550] [akka-spring-demo-akka.kafka.default-dispatcher-10] [App info kafka.producer for pro
[INFO ] [2023-03-19 19:55:55,563] [akka-spring-demo-akka.actor.default-dispatcher-23] [Received String message: hello]
```

Junit Base + Akka UnitTest , .

```java
@Test
@DisplayName("KafkaProduce - 100  ")
public void TestIt() {
    new TestKit(system) {
        {
            final TestKit probe = new TestKit(system);
            final ActorRef greetActor = AkkaManager.getInstance().getGreetActor();

            greetActor.tell(probe.getRef(), getRef());
            expectMsg(Duration.ofSeconds(1), "done");

            final Config config = system.settings().config().getConfig("akka.kafka.producer");
            final ProducerSettings<String, String> producerSettings =
                    ProducerSettings.create(config, new StringSerializer(), new StringSerializer())
                            .withBootstrapServers("localhost:9092");

            String topic = "test-1";

            CompletionStage<Done> done =
                    Source.range(1, 100)
                            .map(number -> number.toString())
                            .map(value -> new ProducerRecord<String, String>(topic, value))
                            .runWith(Producer.plainSink(producerSettings), system);

            Source<Done, NotUsed> source = Source.completionStage(done);

            within(
                    Duration.ofSeconds(10),
                    () -> {

                        Sink<Done, CompletionStage<Done>> sink = Sink.foreach(i ->
                                greetActor.tell("hello", getRef())
                        );
                        source.runWith(sink, system); // 10

                        // check that the probe we injected earlier got the msg
                        probe.expectMsg(Duration.ofSeconds(5), "world");

                        // Will wait for the rest of the 3 seconds
                        expectNoMessage();
                        return null;
                    });
        }
    };
}
```

Kafka Message



Kafka .

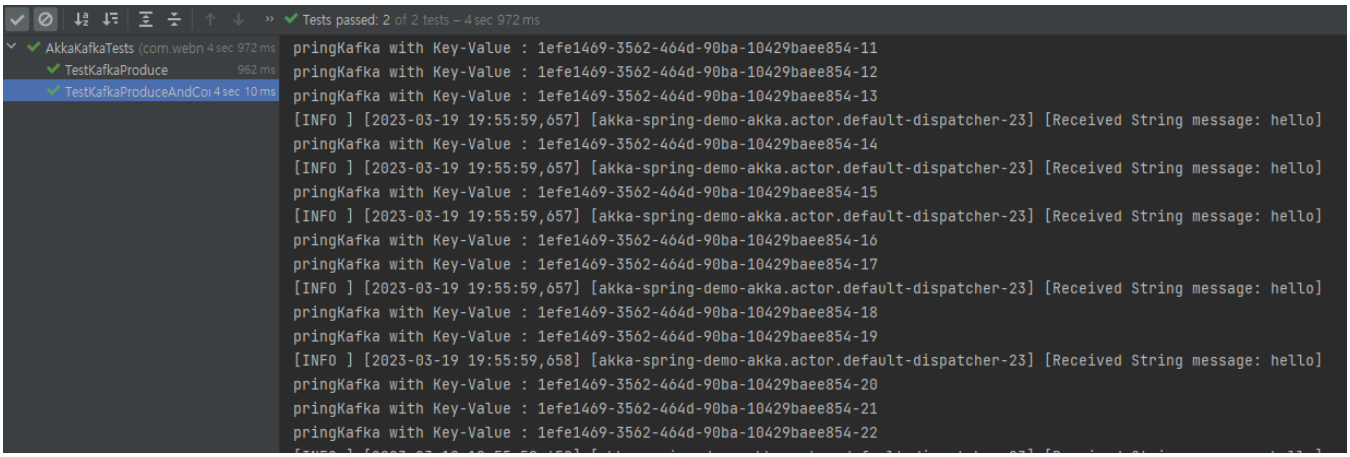
, .

Alpakka Kafka AtLeastOnce .

- https://doc.akka.io/docs/alpakka-kafka/current/atleastonce.html
    - AtMostOnce -
    - AtLeastOnce -
    - EXACTLY-ONCE DELIVERY - , .

```
    void debugKafkaMsg(String key, String value, ActorRef greet, String testKey) {
        System.out.printf("pringKafka with Key-Value : %s-%s%n", key, value);

        //   ..( Kafka )
        if(testKey.equals(key)) greet.tell("hello", null);

    }

    @Test
    @DisplayName("KafkaProduce - 100  100  ")
    public void TestKafkaProduceAndConsume() {
        new TestKit(system) {
            {
                final TestKit probe = new TestKit(system);
                final ActorRef greetActor = AkkaManager.getInstance().getGreetActor();
                final int testCount = 100;
                final String testKey = java.util.UUID.randomUUID().toString();
                final String testKafkaServer = "localhost:9092";
                final String testGroup = "group1";

                greetActor.tell(probe.getRef(), getRef());
                expectMsg(Duration.ofSeconds(1), "done");

                final Config producerConfig = system.settings().config().getConfig("akka.kafka.producer");
                final ProducerSettings<String, String> producerSettings =
                        ProducerSettings.create(producerConfig, new StringSerializer(), new StringSerializer())
                                .withBootstrapServers(testKafkaServer);


                final Config conSumeConfig = system.settings().config().getConfig("akka.kafka.consumer");
                final ConsumerSettings<String, String> consumerSettings =
                        ConsumerSettings.create(conSumeConfig, new StringDeserializer(), new
StringDeserializer())
                                .withBootstrapServers(testKafkaServer)
                                .withGroupId(testGroup)
                                .withProperty(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "true")
                                .withProperty(ConsumerConfig.AUTO_COMMIT_INTERVAL_MS_CONFIG, "3000")
                                .withProperty(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");


                String topic = "test-1";

                //Consumer Setup
                Consumer
                        .plainSource(
                                consumerSettings,
                                Subscriptions.topics(topic))
                        .to(Sink.foreach(msg ->
                                debugKafkaMsg(msg.key(), msg.value(), greetActor, testKey))
                        )
                        .run(system);

                //Producer Setup
```

```
                CompletionStage<Done> done =
                        Source.range(1, testCount)
                                .map(number -> number.toString())
                                .map(value -> new ProducerRecord<String, String>(topic, testKey, value))
                                .runWith(Producer.plainSink(producerSettings), system);

                //Producer Task Setup
                Source<Done, NotUsed> source = Source.completionStage(done);

                within(
                        Duration.ofSeconds(10),
                        () -> {

                            Sink<Done, CompletionStage<Done>> sink = Sink.foreach(i ->
                                    System.out.println("")
                            );

                            //For Clean Test - 3
                            expectNoMessage(Duration.ofSeconds(3));

                            // Kafka
                            source.runWith(sink, system);

                            // Kafka   (100)
                            for (int i = 0; i < testCount; i++) {
                                probe.expectMsg(Duration.ofSeconds(5), "world");
                            }

                            return null;
                        });
            }
        };
    }
```



AkkaStream , Kafka .

Kafka Kafka Kafka .

Dashboard

local

Brokers

Topics

Consumers

Topics / test-1

Overview | Messages | Consumers | Settings | Statistics

Topic Name *

test-1

Cleanup policy

Delete

Min In Sync Replicas

1

Time to retain data (in ms) 1d

86400000

12 hours | 1 day | 2 days | 7 days | 4 weeks

Max size on disk in GB

Not Set

Maximum message size in bytes

1048588

Custom parameters

+Add Custom Parameter

Cancel | Update topic

Danger Zone

Change these parameters only if you are absolutely sure what you are doing.

Number of partitions *

2

Submit

Replication Factor *

3

Submit

```
//Consumer Setup
var control =
        Consumer.plainSource(
                        consumerSettings,
                        Subscriptions.assignment(new TopicPartition(topic,  partition: 0))) Source<ConsumerRecord<String, String>, Control>
                .to(Sink.foreach(msg ->
                        debugKafkaMsg(msg.key(), msg.value(), greetActor, testKey,  consumerId: "consumer1"))
                ) RunnableGraph<Control>
                .run(system);

var control2 =
        Consumer.plainSource(
                        consumerSettings,
                        Subscriptions.assignment(new TopicPartition(topic,  partition: 1))) Source<ConsumerRecord<String, String>, Control>
                .to(Sink.foreach(msg ->
                        debugKafkaMsg(msg.key(), msg.value(), greetActor, testKey,  consumerId: "consumer2"))
                ) RunnableGraph<Control>
                .run(system);
```

```
                                // Kafka
                                source.runWith(sink, system);   // 0
                                source2.runWith(sink, system);  // 1

                                // Kafka    -
                                for (int i = 0; i < testCount * partitionCount; i++) {
                                    probe.expectMsg(Duration.ofSeconds(5), "world");
                                }
```

N N .

N .

Consumer Type : https://doc.akka.io/docs/alpakka-kafka/current/consumer.html

```
# Properties for akka.kafka.ProducerSettings can be
# defined in this section or a configuration section with
# the same layout.
akka.kafka.producer {
  # Config path of Akka Discovery method
  # "akka.discovery" to use the Akka Discovery method configured for the ActorSystem
  discovery-method = akka.discovery

  # Set a service name for use with Akka Discovery
  # https://doc.akka.io/docs/alpakka-kafka/current/discovery.html
  service-name = ""

  # Timeout for getting a reply from the discovery-method lookup
  resolve-timeout = 3 seconds

  # Tuning parameter of how many sends that can run in parallel.
  # In 2.0.0: changed the default from 100 to 10000
  parallelism = 10000

  # Duration to wait for `KafkaProducer.close` to finish.
  close-timeout = 60s

  # Call `KafkaProducer.close` when the stream is shutdown. This is important to override to false
  # when the producer instance is shared across multiple producer stages.
  close-on-producer-stop = true

  # Fully qualified config path which holds the dispatcher configuration
  # to be used by the producer stages. Some blocking may occur.
  # When this value is empty, the dispatcher configured for the stream
  # will be used.
  use-dispatcher = "akka.kafka.default-dispatcher"

  # The time interval to commit a transaction when using the `Transactional.sink` or `Transactional.flow`
  # for exactly-once-semantics processing.
  eos-commit-interval = 100ms

  # Properties defined by org.apache.kafka.clients.producer.ProducerConfig
  # can be defined in this configuration section.
  kafka-clients {
  }
}

akka.kafka.consumer {

  enable.auto.commit = true

  kafka-clients {
```

```
    bootstrap.servers = "localhost:9092"
  }
}

akka.kafka.committer {

  # Maximum number of messages in a single commit batch
  max-batch = 1000

  # Maximum interval between commits
  max-interval = 3s

  # Parallelsim for async committing
  parallelism = 100

  # API may change.
  # Delivery of commits to the internal actor
  # WaitForAck: Expect replies for commits, and backpressure the stream if replies do not arrive.
  # SendAndForget: Send off commits to the internal actor without expecting replies (experimental feature since
1.1)
  delivery = WaitForAck

  # API may change.
  # Controls when a `Committable` message is queued to be committed.
  # OffsetFirstObserved: When the offset of a message has been successfully produced.
  # NextOffsetObserved: When the next offset is observed.
  when = OffsetFirstObserved
}
```

```
                final Config conSumeConfig = system.settings().config().getConfig("akka.kafka.consumer");
                final ConsumerSettings<String, String> consumerSettings =
                        ConsumerSettings.create(conSumeConfig, new StringDeserializer(), new
StringDeserializer())
                                .withBootstrapServers(testKafkaServer)
                                .withGroupId(testGroup)
                                .withProperty(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "true")
                                .withProperty(ConsumerConfig.AUTO_COMMIT_INTERVAL_MS_CONFIG, "3000")
                                .withProperty(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");
```

(//) .

- ChatGPT

ChatGPT .

> *Q>*
>
> *ChatGPT> Akka-Streams Akka-Kafka Kafka .*
>
> *"TestKafkaProduce" Kafka 1 100*
>
> *Kafka Kafka . "hello" Akka .*
>
> *"TestKafkaProduceAndConsume" Kafka 1 100 Kafka*
>
> *. Kafka Kafka . "hello" Akka .*

Kafka

Akka JAVA Scala JAVA SCALA JAVA .

AKKA SCALA

: https://getakka.net/community/contributing/porting-guide.html

AKKA Reactive JAVA SPRING KAFKA Webflux .

- https://velog.io/@deogicorgi/Spring-Webflux-Reactive-Kafka-2-Consumer

:

- https://github.com/psmon/java-labs/tree/master/springweb/src/test/java/com/webnori/springweb/akka - AKKA Unit Test
- https://doc.akka.io/docs/akka/current/stream/stream-flows-and-basics.html - Akka Stream
- https://doc.akka.io/docs/alpakka/current/index.html - alpakka
- https://alpakka.getakka.net/ - alpakka
- kafka and akka.net - ( )

: alpakka