Kafka with Stream

AkkaStream .

Stream Kafka (,) Reactive Stream .

, .

, .

```
// Kafka Reactive Stream
// https://github.com/akkadotnet/Akka.Streams.Kafka
// https://github.com/akka/alpakka

namespace AkkaNetCoreTest.Adapters
{
    //   : Kafka   ,Akka Stream  Reactive Stream  .
    public class AlpakkaTest : TestKitXunit
    {
        protected TestProbe probe;

        protected ProducerSettings<Null,string> producerSettings;

        protected ConsumerSettings<Null, string> consumerSettings;

        protected string testTopic;

        protected IAutoSubscription subscription;

        protected ActorMaterializer materializer_producer;
        protected ActorMaterializer materializer_consumer;

        public AlpakkaTest(ITestOutputHelper output) : base(output)
        {
            Setup();
        }

        protected void Setup()
        {
            testTopic = "akka100";

            subscription = Subscriptions.Topics(testTopic);

            probe = this.CreateTestProbe();

            var config = HoconConfigurationFactory.FromFile("akka.test.conf");

            var system_producer = ActorSystem.Create("TestKafka", config);
            materializer_producer = system_producer.Materializer();

            var system_consumer = ActorSystem.Create("TestKafka", config);
            materializer_consumer = system_producer.Materializer();

            this.Sys.Settings.Config.WithFallback(config);

            producerSettings = ProducerSettings<Null, string>.Create(system_producer, null, null)
                .WithBootstrapServers("kafka:9092");

            consumerSettings = ConsumerSettings<Null, string>.Create(system_consumer, null, null)
                .WithBootstrapServers("kafka:9092")
                .WithGroupId("group1");

        }
```

```
        [Theory]
        [InlineData(20,10)] //20   ,, 10 ( )
        public void __(int limit, int cutoff)
        {
            string lastSignal = Guid.NewGuid().ToString();
            int readyTimeForConsume = 3;
            int recCnt = 0;

            KafkaConsumer.CommittableSource(consumerSettings, subscription)
            .RunForeach(result =>
            {
                Console.WriteLine($"Consumer: {result.Record.Topic}/{result.Record.Partition} {result.Record.
Offset}: {result.Record.Value}");
                if (lastSignal == result.Record.Value)
                    probe.Tell("");

                result.CommitableOffset.Commit();

            }, materializer_consumer);

            Source<int, NotUsed> source = Source.From(Enumerable.Range(1, limit));

            source
                .Throttle(2, TimeSpan.FromSeconds(1), 1, ThrottleMode.Shaping)      // : 2- .(Akka Stream)
                .Select(c =>
                {
                    var result = $"No:{c.ToString()}";
                    if(c == limit)
                    {
                        result = lastSignal;
                    }
                    return result;
                })
                .Select(elem => ProducerMessage.Single(new ProducerRecord<Null, string>(testTopic, elem)))
                .Via(KafkaProducer.FlexiFlow<Null, string, NotUsed>(producerSettings))
                .Select(result =>
                {
                    var response = result as Result<Null, string, NotUsed>;
                    Console.WriteLine($"Producer: {response.Metadata.Topic}/{response.Metadata.Partition}
{response.Metadata.Offset}: {response.Metadata.Value}");
                    return result;
                })
                .RunWith(Sink.Ignore<IResults<Null, string, NotUsed>>(), materializer_producer);

            Within(TimeSpan.FromSeconds(cutoff), () =>
            {
                probe.ExpectMsg("", TimeSpan.FromSeconds(cutoff));
            });
        }
    }
}
```

:

- https://github.com/psmon/AkkaForNetCore/blob/master/InfraCompose/docker-compose.yml - (host : kafka 127.0.0.1)
- https://github.com/psmon/AkkaForNetCore/blob/master/AkkaNetCoreTest/Adapters/AlpakkaTest.cs -
- https://github.com/akkadotnet/Akka.Streams.Kafka -