AkkaNetWithKafka

> Akka Kafka
>
> AkkaDotModule HeadLessService .
>
> Git .
>
> git : https://github.com/psmon/AkkaNetWithKafka

APP

netcoreapp3.1 Test, .

```
    <PackageReference Include="AkkaDotModule.Webnori" Version="1.1.2" />
    <PackageReference Include="Microsoft.Extensions.Hosting" Version="3.1.14" />
    <PackageReference Include="Microsoft.Extensions.Hosting.Abstractions" Version="3.1.14" />
```

AkkaDotModule.Webnori

AkkaDotModule AKKA+KAFKA

.

```
    <PackageReference Include="Akka.Streams" Version="1.4.12" />
    <PackageReference Include="Akka.Streams.Kafka" Version="1.1.0" />
    <PackageReference Include="Confluent.Kafka" Version="1.5.2" />
```

Docker Kafka

bitnami StandAlone Kafka

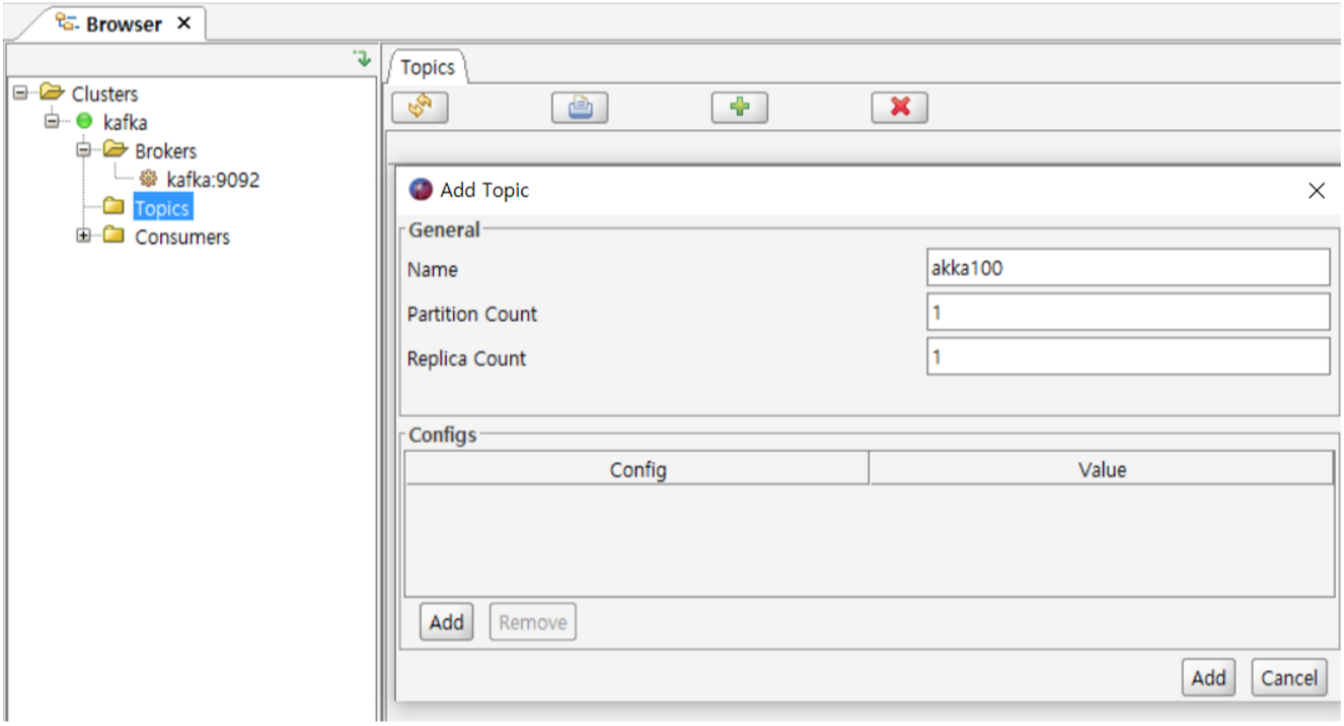
docker-compose up -d .

```
version: '3.4'

services:
  zookeeper:
    image: 'bitnami/zookeeper:latest'
    ports:
    - '2181:2181'
    environment:
    - ALLOW_ANONYMOUS_LOGIN=yes

  kafka:
    hostname: kafka
    image: 'bitnami/kafka:latest'
    ports:
    - '9092:9092'
    environment:
    - KAFKA_ADVERTISED_HOST_NAME=kafka
    - KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181
    - ALLOW_PLAINTEXT_LISTENER=yes
```

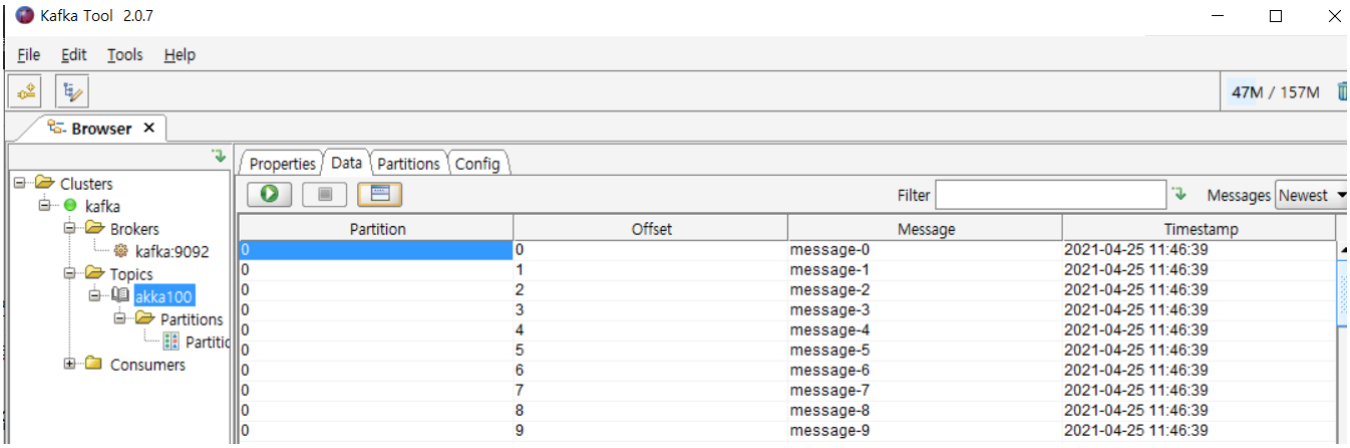
Topic



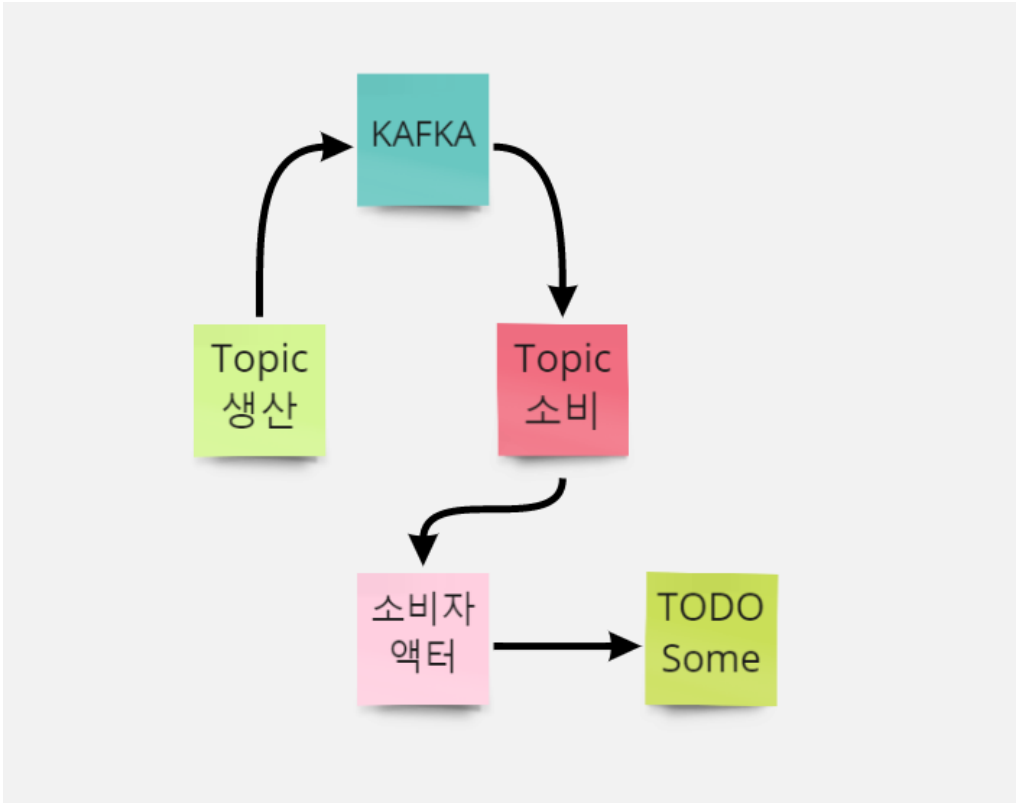
Kafka ， topic . (kafka tool .)

topic : akka100

Topic



, 1 10 Kafka-Topic .



Pub/Sub , Kafka

.

StandAlone HeadlessService

```
using AkkaNetWithKafka.Services;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System.Threading.Tasks;

namespace AkkaNetWithKafka
{
    internal class Program
    {
        private static async Task Main(string[] args)
        {
            var host = new HostBuilder()
                .ConfigureServices((hostContext, services) =>
                {
                    services.AddLogging();

                // register our host service
                services.AddHostedService<KafkaService>();

                })
                .ConfigureLogging((hostContext, configLogging) =>
                {
                    configLogging.AddConsole();

                })
                .UseConsoleLifetime()
                .Build();

            await host.RunAsync();
        }
    }
}
```

Console Base,IHostedService KafkaService .

Kafka .

: https://docs.microsoft.com/ko-kr/dotnet/architecture/microservices/multi-container-microservice-net-applications/background-tasks-with-ihostedservice

KafkaService

, .

, Kafka .

```
using Akka.Actor;
using AkkaDotModule.Kafka;
using AkkaNetWithKafka.Actors;
using Microsoft.Extensions.Hosting;
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using ConsumerSystem = AkkaNetWithKafka.Modules.ConsumerSystem;
using ProducerSystem = AkkaNetWithKafka.Modules.ProducerSystem;

namespace AkkaNetWithKafka.Services
{
    public sealed class KafkaService : IHostedService
    {
        private ActorSystem AkkaSystem;

        private ConsumerSystem KafkaConsumerSystem;

        private ProducerSystem KafkaProducerSystem;

        public Task StartAsync(CancellationToken cancellationToken)
        {
```

```
            AkkaSystem = ActorSystem.Create("KafkaService");

            KafkaConsumerSystem = new ConsumerSystem();

            KafkaProducerSystem = new ProducerSystem();

            var consumerActor = AkkaSystem.ActorOf(Props.Create(() => new ConsumerActor()),
                    "consumerActor" /*AKKA  Path*/);

            // :
            KafkaConsumerSystem.Start(new ConsumerAkkaOption()
            {
                KafkaGroupId = "testGroup",
                BootstrapServers = "kafka:9092",
                RelayActor = consumerActor,              //
                Topics = "akka100",
            });

            // :
            KafkaProducerSystem.Start(new ProducerAkkaOption()
            {
                BootstrapServers = "kafka:9092",
                ProducerName = "producer1",
            });


            List<string> messages = new List<string>();
            for (int i = 0; i < 10; i++)
            {
                messages.Add($"message-{i}");
            }

            // :  TPS   .
            int tps = 10;
            KafkaProducerSystem.SinkMessage("producer1", "akka100", messages, tps);


            return Task.CompletedTask;
        }

        public async Task StopAsync(CancellationToken cancellationToken)
        {
            // strictly speaking this may not be necessary - terminating the ActorSystem would also work
            // but this call guarantees that the shutdown of the cluster is graceful regardless
            await CoordinatedShutdown.Get(AkkaSystem).Run(CoordinatedShutdown.ClrExitReason.Instance);
        }
    }
}
```

```
            int tps = 10;
            KafkaProducerSystem.SinkMessage("producer1", "akka100", messages, tps);
```

, . Akka Stream .

, .

ConsumerActor

Kafka , .

```
using Akka.Actor;
using Akka.Event;

namespace AkkaNetWithKafka.Actors
{

    public class ConsumerActor : ReceiveActor
    {
        private readonly ILoggingAdapter logger = Context.GetLogger();

        public ConsumerActor()
        {
            ReceiveAsync<string>(async strData =>
            {
                logger.Info("ConsumerActor IncomeMessage:" + strData);
                //TODO : Somthing
            });
        }
    }
}
```

Kafka / , Kafka

Akka . Reactive Stream .

- MailBox
- Routers
- Streams Quickstart Guide

- https://github.com/confluentinc/confluent-kafka-dotnet
- https://github.com/akkadotnet/Akka.Streams.Kafka
- https://github.com/reactive-streams/reactive-streams-dotnet