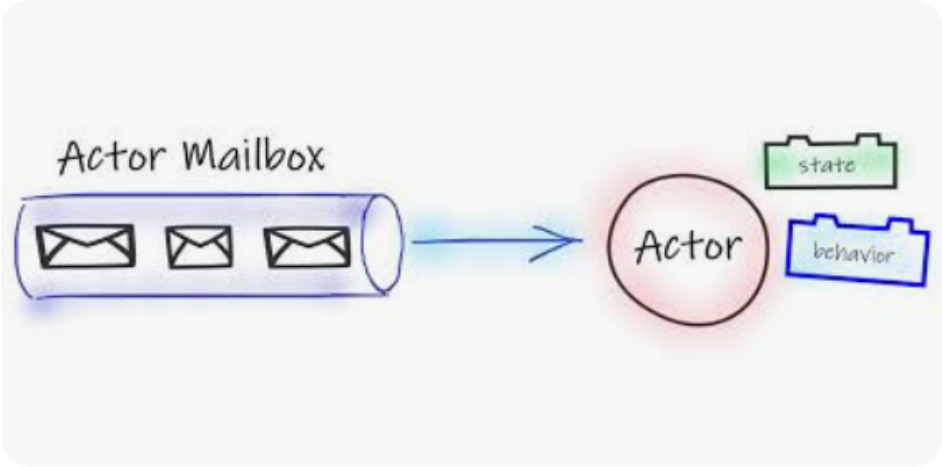
[Labs] AKKA

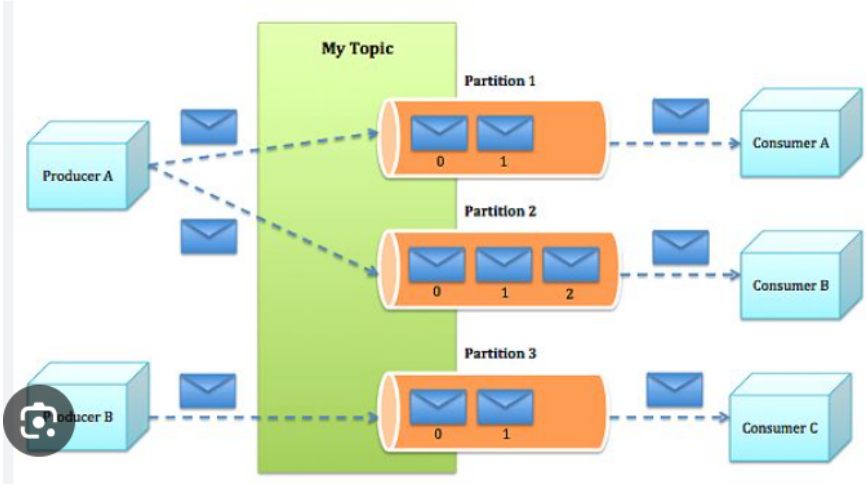
AKKA , , , .

() .

- Spring BOOT With AKKA - https://www.baeldung.com/akka-with-spring
- ASP .NetCore with AKKA - https://getakka.net/articles/deployment/aspnet-core.html
- AKKA MS - https://learn.microsoft.com/ko-kr/dotnet/orleans/
    - AKKA Oeleans .
    - AKKA 80% Orleans .

.





MailBox Kafka MailBox.

.

. .

1986

AKKA .

Erlang "" . Erlang ( ) , . .

Erlang :

1. **:** . , .
2. **:** Erlang . , , , .
3. **:** . .
4. **:** . , , .
5. **:** Erlang . , .

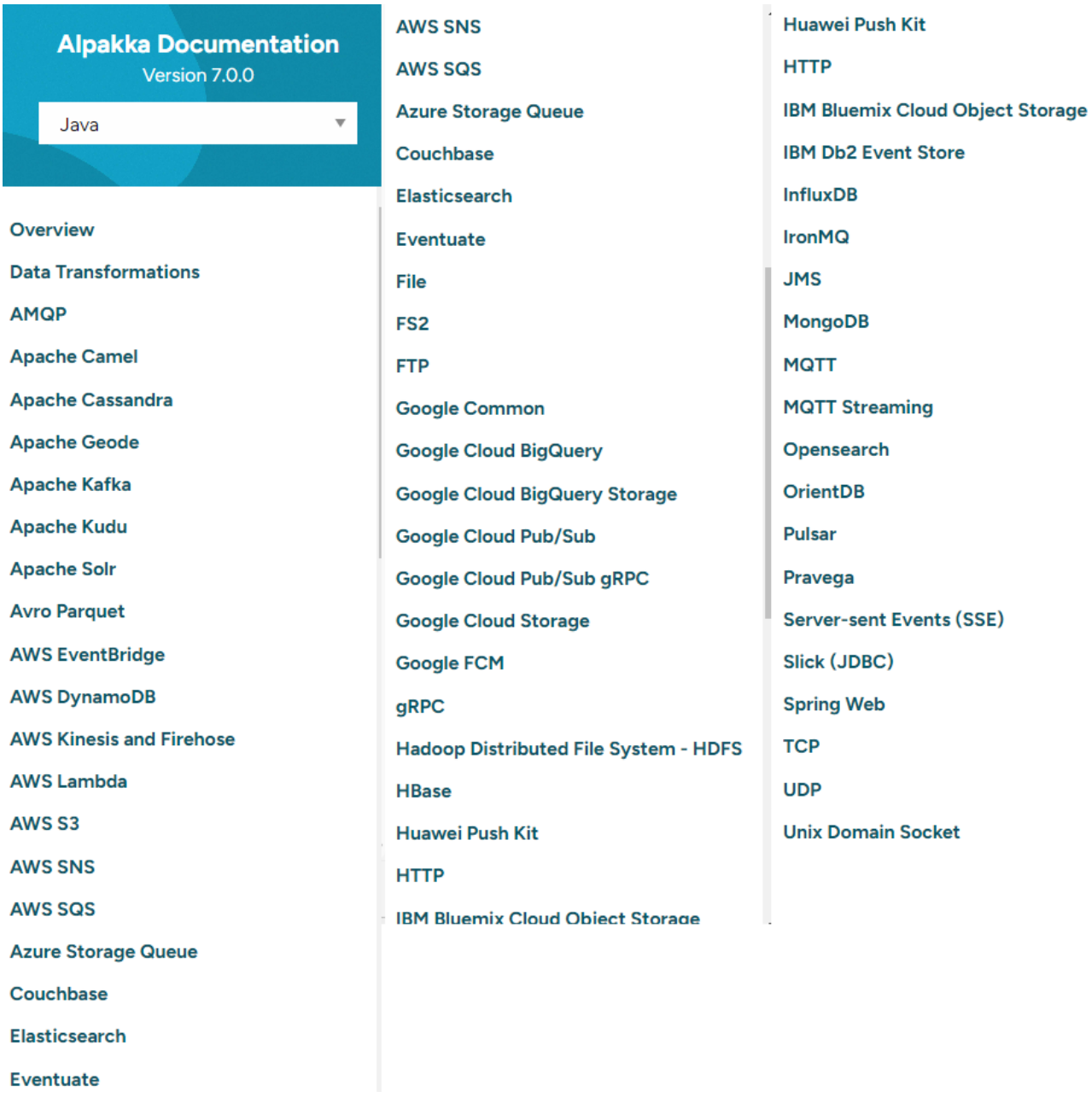
Erlang , , . Erlang , , .

. .

AKKA Alpakka(https://doc.akka.io/docs/alpakka/current/index.html) Kafka .

Reactive Stream .

AKKA OpenStack



Akka Kafka Kafka ,

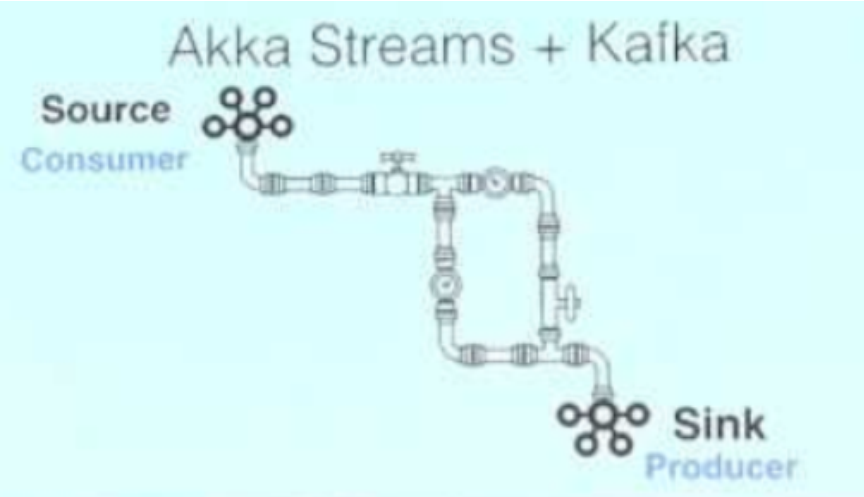
Akka Kafka .

Akka Kafka

Alpakka Kafka Documentation

Version 5.0.0

Java



.

() .

DB .

- https://getakka.net/articles/actors/finite-state-machine.html -

/ AtLeastOnce Delivery / Transaction .

? .

- https://doc.akka.io/docs/akka/current/general/message-delivery-reliability.html

.

AKKA .

VS

, .

- () : . ( )
- : .
- () : ~ .

AKKA .

.

- https://akka.io/blog/article/2016/07/06/threading-and-concurrency-in-akka-streams-explained :
  - ,, 3 .

.NET .

1. **Future CompletableFuture:** `Future` . `Future` .
   8 `CompletableFuture` . `CompletableFuture` , , .
2. `:` RxJava Project Reactor . .

.NET

1. **async/await :** .NET `async` `await` .
   .
2. **Task (TAP):** .NET `Task` `Task<TResult>` , `async/await` .
   .

- `:` `CompletableFuture` , .NET `async` `await` .
- `:` .NET `async/await` , `CompletableFuture` .
- `:` ,
  .NET TPL (Task Parallel Library) Reactive Extensions (Rx.NET) .

AKKA /

- https://doc.akka.io/docs/akka/2.5/futures.html - JAVA Scala .
- https://petabridge.com/blog/async-await-vs-pipeto/ - Await/TPL .

AKKA Scala Future/Promice `asynce/await` .

`CompletableFuture` 7 Scala(Akka) .

Java9 Akka Stream StreamAPI .

AKKA Stream API .

stream api linq .

- https://www.baeldung.com/java-9-stream-api
- https://www.iodigital.com/en/history/foreach/java-and-net-comparing-streams-linq

/ Stream API AkksStream .

AKKA . **(.)**

~ AKKA .

`Rx.net/Rx.net/WebPlux` .

`AKKA 1:1 , AkkaStream API .`

AKKA .

`AKKA(Akka.net)`

.

AKKA .

AKKA .

.

.

.

Dispatcher

.

```
default-fork-join-dispatcher {
  type = ForkJoinDispatcher
  throughput = 30
  dedicated-thread-pool {
      thread-count = 3
      deadlock-timeout = 3s
      threadtype = background
  }
}

akka.remote.default-remote-dispatcher {
    type = Dispatcher
    executor = channel-executor
    fork-join-executor {
      parallelism-min = 2
      parallelism-factor = 0.5
      parallelism-max = 16
    }
}
```

- TPL(https://learn.microsoft.com/ko-kr/dotnet/standard/parallel-programming/task-parallel-library-tpl)
  .
- ForkJoin ThreadPool .
  - .
    - AKKA Netty TCP / JSON, .

AKKA .

,

.

() .

AKKA .

.

- https://velog.io/@maketheworldwise/ForkJoin%EC%9D%84-%EC%95%8C%EC%95%84%EB%B3%B4%EC%9E%90 - java
- https://learn.microsoft.com/ko-kr/dotnet/standard/parallel-programming/task-parallel-library-tpl - c#
- https://learn.microsoft.com/ko-kr/dotnet/standard/threading/threading-objects-and-features - c#

Node.js .

.

.

? ? .

AKKA /Stream/dispacher . ?

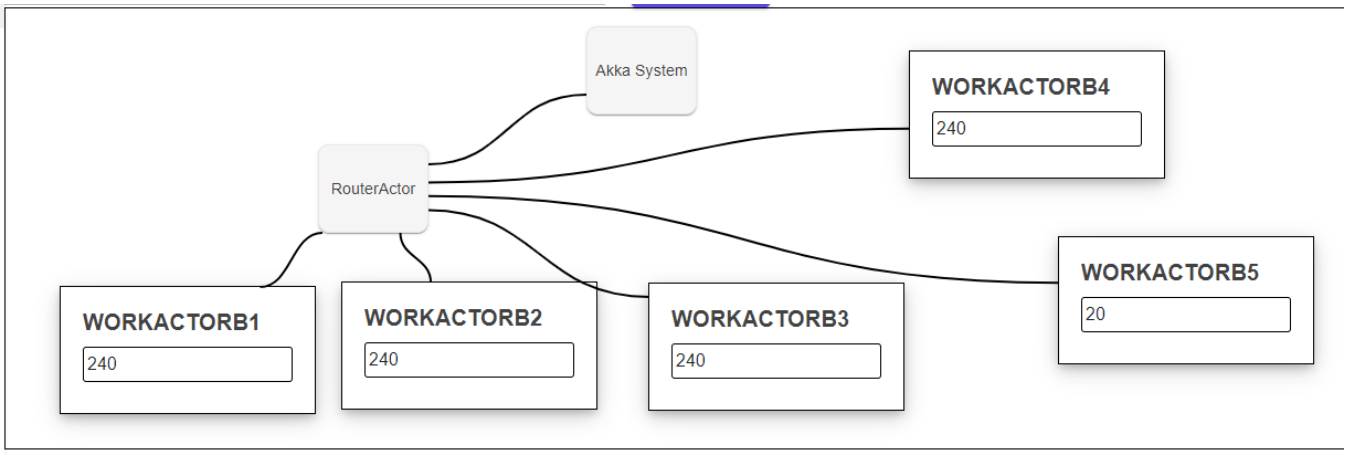
.

. . .

1. **Twitter**: Twitter . , Twitter Scala Akka . Akka , . Twitter .
2. **LinkedIn**: LinkedIn . LinkedIn Kafka , .
3. **WhatsApp**: WhatsApp Erlang . Erlang , WhatsApp .

, . , .

.



SampleCode

```
// Create ActorSystem
var actorSystem = akkaService.CreateActorSystem();
// Create RoundRobin Router
var broadcast = actorSystem.ActorOf(Props.Create<BasicActor>().WithRouter(new BroadcastPool(0)));
// Create Worker and Add Routee
var workActor = actorSystem.ActorOf(Props.Create<BasicActor>(), NodeName);
var routee = Routee.FromActorRef(workActor);
broadcast.Tell(new AddRoutee(routee));
// Say Hello
broadcast.Tell("Hello");
```

: http://code.webnori.com/actor/broadcast

.

(.)

Node Thread .

.

- .
- .
- Router AkkaStream TPS .
- /// .

. .

.

JAVA

```
public class HelloWorldActor extends AbstractActor {
    @Override
    public Receive createReceive() {
        return receiveBuilder()
            .matchEquals("sayHello", s -> {
                log.info("Hello World");
            })
            .build();
    }
}
```

C#

```
public class HelloWorldActor : ReceiveActor
{
    public HelloWorldActor()
    {
        Receive<string>(message =>
        {
            if (message == "sayHello")
            {
                Console.WriteLine("Hello World");
            }
        });
    }
}
```

C# - Ms Orleans

```
using Orleans;

public class HelloGrain : Grain, IHelloGrain
{
    public Task<string> SayHello(string greeting)
    {
        return Task.FromResult($"You said: '{greeting}', I say: Hello World!");
    }
}
```

.

. .

. AKKA .

- ,
- ?
- ?
- ?
- ?( )

.

.

- TCP
-
-

JAVA 7 AKKA AKKA . .

.

.

? C++ ?

.

OOP ~ .

(//) .

MS Orlean

, **PASS .**

(Active Object) . . , .

:

1. : , . , .
2. : , . .
3. : , .
4. : . .
5. : , . , .
6. **(Future)**: . .

, , . , , .

.

() . .

AKKA .

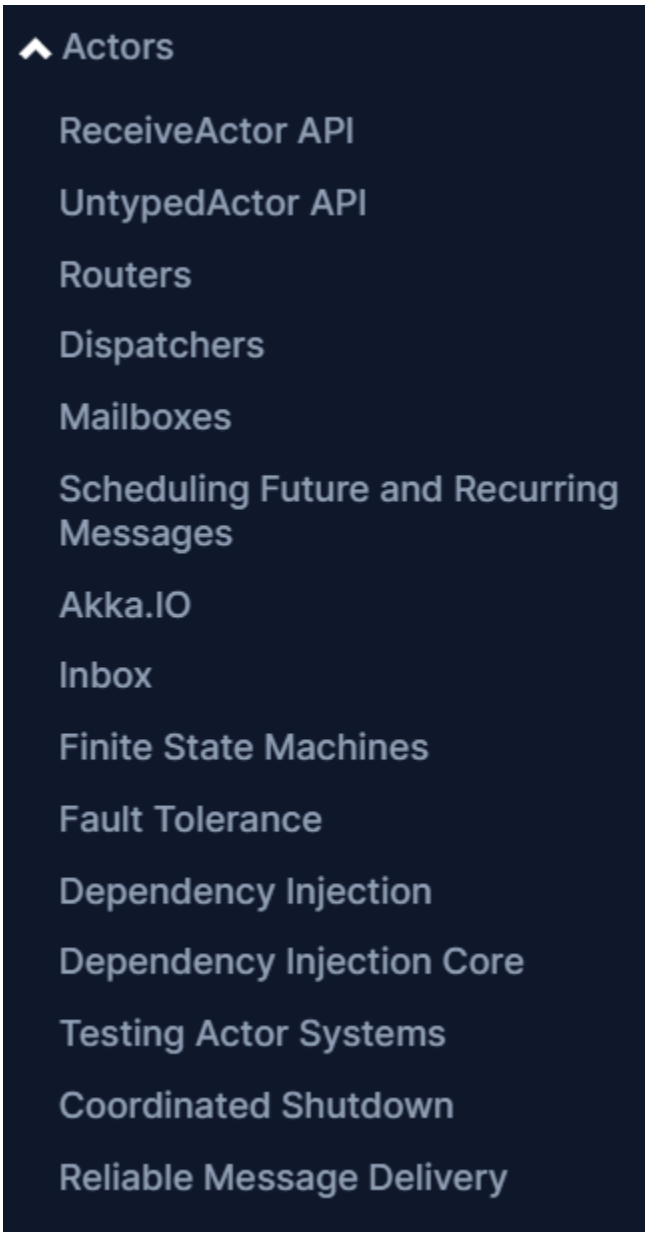
.

.

AKKA GPT .

AKKA . OOP

AKKA AKKA .



- MailBox : MailBox .
- Routers : .
- Dispatchers : .
- Fine State Machines : .
- Relible Message Deivery : ? .
- Coordinated Shutdown : GraceFul Down .

```
public class ReActor : ReceiveActor
{
  private ILoggingAdapter log = Context.GetLogger();

  public ReActor()
  {
      string myPath = Self.Path.ToString();

      Receive<string>(message => {
          Handle(message);
      });

      Receive<DelayReply>(message => {
          Handle(message);
      });
  }

  public void Handle(string str)      //InMessage
  {
      Task.Run(async () =>
      {
          await Task.Delay(1000); //     .
          DelayReply reply = new DelayReply();
          reply.message = str;
          return reply;
      }).PipeTo(Self);
  }

  public void Handle(DelayReply data) //Out
  {
      string logtrace = string.Format("I'am {0} RE:{1}", Self.Path, data.message);
      log.Info(data.message);
      Sender.Tell(data);
  }

}
```

/ . .

 .

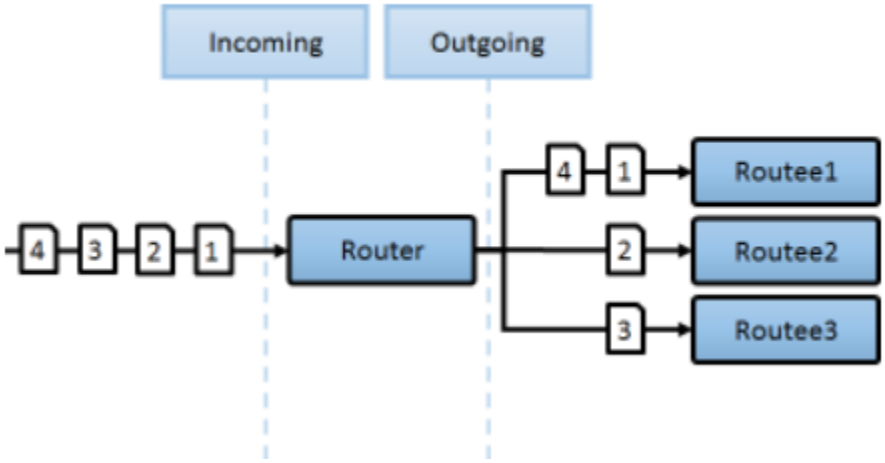
Router

~ .

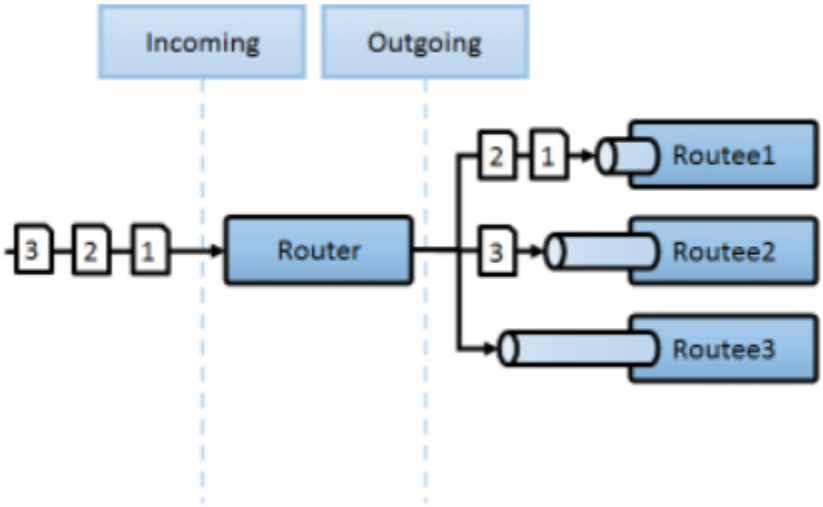
 .

AKKA .

- .

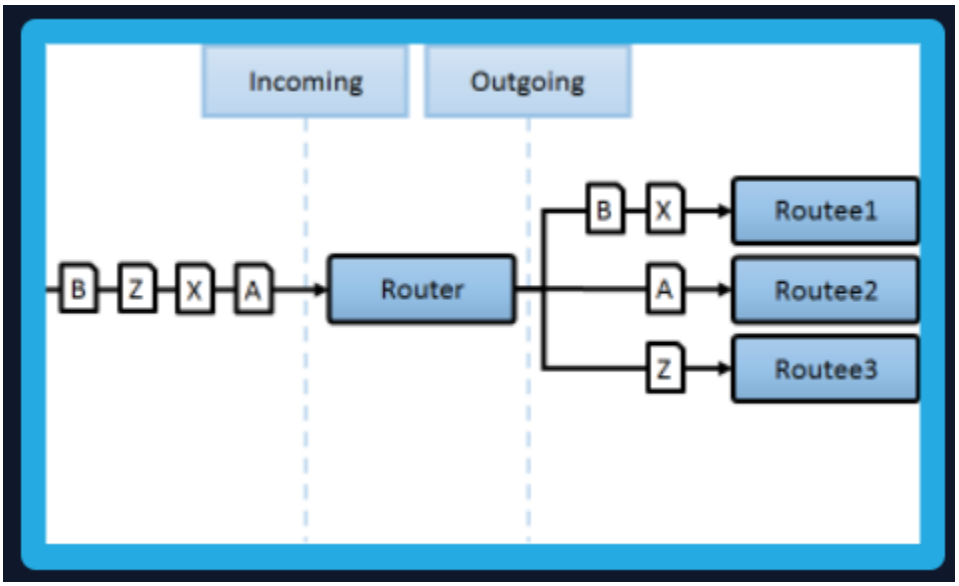

Incoming
Outgoing
4
3
2
1
Router
4
1
Routee1
2
Routee2
3
Routee3

- , .


Incoming
Outgoing
3
2
1
Router
2
1
Routee1
3
Routee2
Routee3

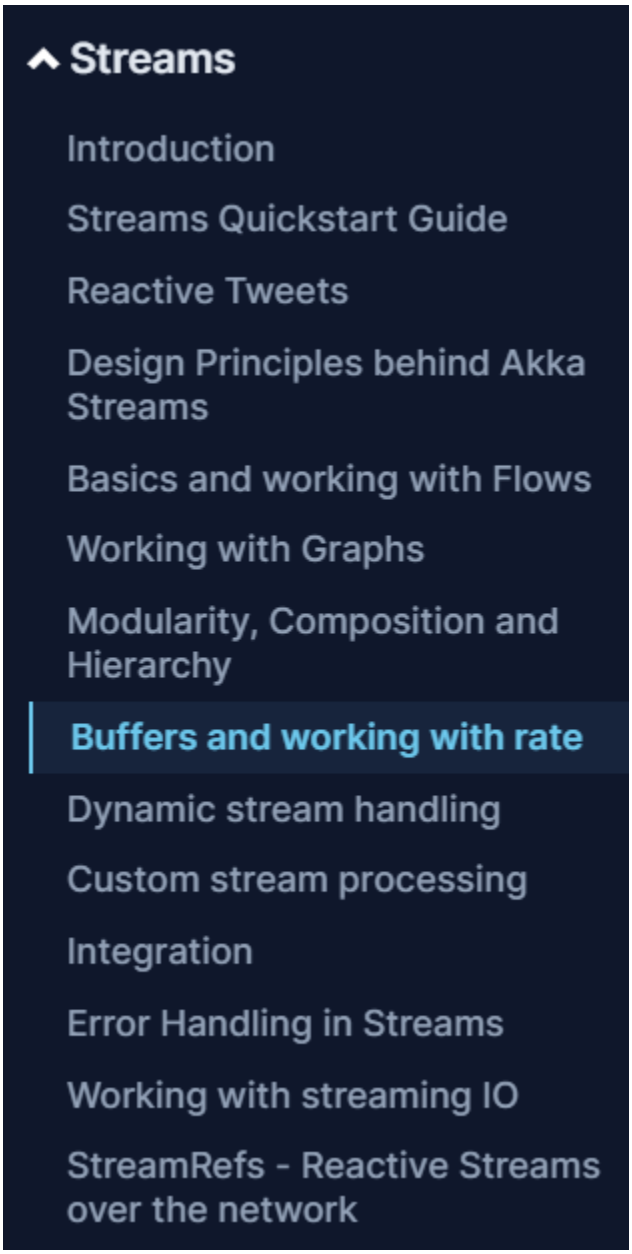
Hash - .( )

() .



: 05.RouterActor

Stream



Stream .

LIghtbend **Reactive Stream**

Rx.NET Reactive Stream .

Akka Stream Reactive Stream 9 StreamAPI , Pivotal Webplux .

- https://www.youtube.com/@ReactiveSummit/videos - Reactive Stream Summit

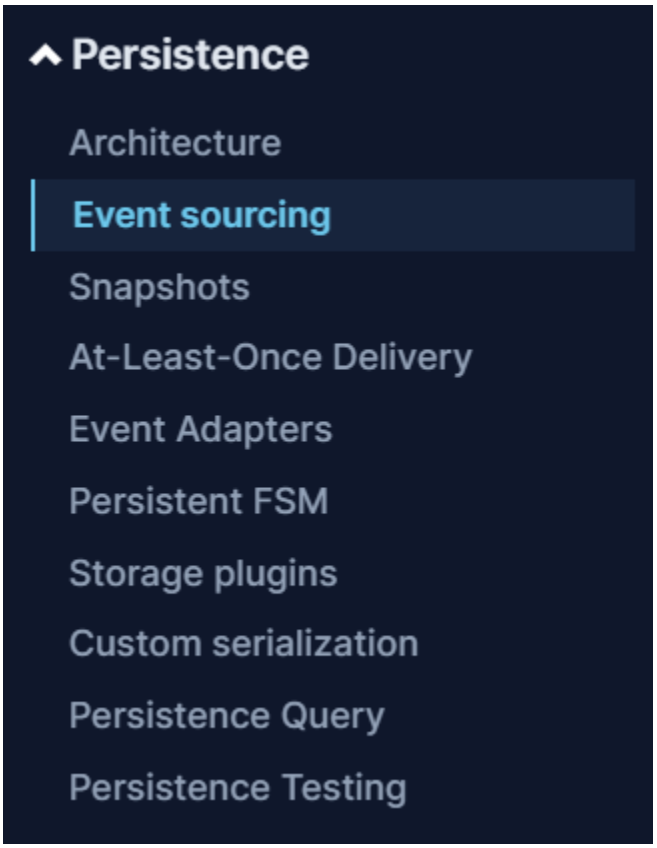
TPS10 Throttle

```
        using (var system = ActorSystem.Create("MySystem"))
        {
            using (var materializer = system.Materializer())
            {
                //   :
                var source = Source.From(Enumerable.Range(1, 100));

                //   :
                var sink = Sink.ForEach<int>(i => Console.WriteLine($"Processed {i}"));

                // Throttle   10
                var throttledFlow = Flow.Create<int>()
                                        .Throttle(10, TimeSpan.FromSeconds(1), 1, ThrottleMode.Shaping);

                //
                await source.Via(throttledFlow).RunWith(sink, materializer);
            }
        }
```

/ .

CQRS . PASS

.



- Event Sourcing : CQRS ,
- At-Least-Once Delivery : Kafka .
- Persitent FSM / Storage : .

AKKA.net

```
public class ShoppingCartActor : ReceivePersistentActor
{
    private readonly string _cartId;
    private readonly List<string> _items = new List<string>();

    public ShoppingCartActor(string cartId)
    {
        _cartId = cartId;

        Recover<ItemAdded>(evt => _items.Add(evt.ItemId));
        Recover<ItemRemoved>(evt => _items.Remove(evt.ItemId));

        Command<AddItem>(cmd =>
        {
            Persist(new ItemAdded(cmd.ItemId), evt =>
            {
                _items.Add(evt.ItemId);
            });
        });

        Command<RemoveItem>(cmd =>
        {
            Persist(new ItemRemoved(cmd.ItemId), evt =>
            {
                _items.Remove(evt.ItemId);
            });
        });
    }

    public override string PersistenceId => _cartId;
}
```

Cluster

AKKA .

(/DB/KAFKA) ? .

Akka Split Brain Resolver ( )

.

.

AKKA .

- https://getakka.net/articles/clustering/split-brain-resolver.html
- https://doc.akka.io/docs/akka/current/split-brain-resolver.html

(Static Quorum)

Sprit Brain AKKA . .

Spilt Brain

```
Akka.NET Split Brain Resolver (SBR)      . S
plit Brain          ,
  .   ,   ,        .

Akka.NET     Split Brain Resolver    . SBR
       . Akka.NET    :

1. Static Quorum
:         .
:       .

2. Keep Majority
:      .
:        .

3. Keep Oldest
:       .
:       .

4. Lease Majority
:  (: Etcd, Consul) ''      .
:      .


AKKA      .
akka.cluster.split-brain-resolver {
  active-strategy = keep-majority
}


   :
1.
  ,        .
   (:  , ,  )  .

2.
Akka.NET    '(Up)'           .
 Akka.NET     ,      .

3.
     ,       .
 ,  ' (Joining)'  '(Up)'   ,   .

4.
         ,     .
  '(Up)'           .

5.
          .
        .


 :              .
             .

:    , Akka.NET           .
Akka.NET            .      ,
     .

Akka      Split Brain       .        .

   :
  :   ,            .
            .
```

: .
 , .

: , , .

:
: . .

: .Akka Split Brain Resolver .

 , .

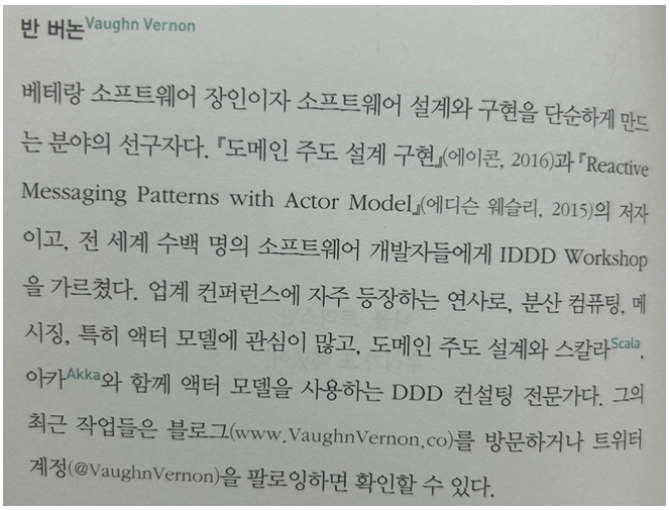
AKKA . .

.

.

A B TCP 0.1 1 .

- NetCoreCluster -
- https://github.com/psmon/springcloud - spirng boot akka cluseter



반 버논Vaughn Vernon

베테랑 소프트웨어 장인이자 소프트웨어 설계와 구현을 단순하게 만드는 분야의 선구자다. 『도메인 주도 설계 구현』(에이콘, 2016)과 『Reactive Messaging Patterns with Actor Model』(에디슨 웨슬리, 2015)의 저자이고, 전 세계 수백 명의 소프트웨어 개발자들에게 IDDD Workshop을 가르쳤다. 업계 컨퍼런스에 자주 등장하는 연사로, 분산 컴퓨팅, 메시징, 특히 액터 모델에 관심이 많고, 도메인 주도 설계와 스칼라Scala, 아카Akka와 함께 액터 모델을 사용하는 DDD 컨설팅 전문가다. 그의 최근 작업들은 블로그(www.VaughnVernon.co)를 방문하거나 트위터 계정(@VaughnVernon)을 팔로잉하면 확인할 수 있다.

AKKA CQRS

DDD 3 AKKA DDD .
(DDD ~)

OOP

.

.

async/await CompletableFuture . rx.net webplux .

/

.

~

Kafka ?

,

AKKA JVM/CLR

AKKA AKKA , .

- akkalabs -
- DDD - DDD
- https://doc.akka.io/docs/akka/current/typed/persistence.html - DDD CQRS
- https://www.slideshare.net/Lightbend/lightbend-fast-data-platform - DataLake AKKA FastData

NEXT

- API - AKKA