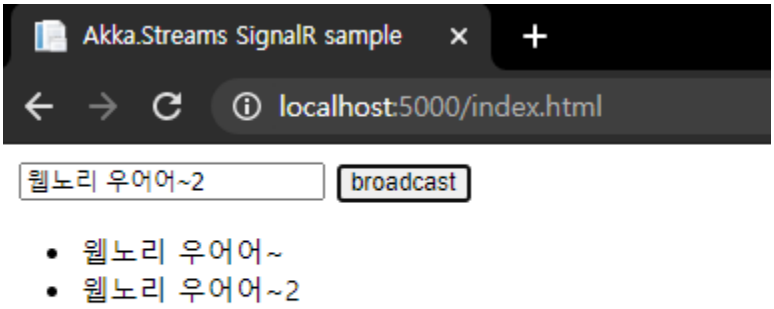
SignalR with AKKA Stream

> SignalR
>
> AKKA STREAM .
>
> Akka.net Alpakka .

STEP 1 - WS CLIENT



, ,

Echo Client.

: https://github.com/psmon/AkkaDotModule/tree/master/AkkaDotBootApi/wwwroot

```
<PackageReference Include="AkkaDotModule.Webnori" Version="1.1.1" />

<PackageReference Include="Akka.Streams.SignalR.AspNetCore" Version="1.0.0-beta3" />
```

SignalR AkkaStream Alpakka .

Alpakka , AkkaStream

Reactive Stream .

Build high-performance, streaming applications **quickly** and with **less code**.

Alpakka makes it easy to build high-performance streaming applications using common techologies like RabbitMQ, Kafka, Azure Service Bus in combination with Akka.NET and Akka.Streams.

This project is currently in beta, but is officially supported by Petabridge and the Akka.NET Project.

- .net : https://alpakka.getakka.net/
- jvm : https://doc.akka.io/docs/alpakka/current/index.html
- reactive stream : https://github.com/reactive-streams

jvm Alpakka .

.

AkkaStream .

STEP 2 - Echo Server

```
using Akka.Streams.Dsl;
using Akka.Streams.SignalR.AspNetCore;
using Akka.Streams.SignalR.AspNetCore.Internals;
using AkkaDotModule.Config;
using Microsoft.AspNetCore.SignalR;

// Sample : https://github.com/akkadotnet/Alpakka/tree/dev/src/SignalR.AspNetCore
namespace AkkaDotBootApi.SignalR
{
    public class EchoHub : StreamHub<EchoStream>
    {
        public EchoHub(IStreamDispatcher dispatcher)
            : base(dispatcher)
        { }
    }

    // AKKA STREAM : https://getakka.net/articles/streams/cookbook.html
    public class EchoStream : StreamConnector
    {
        public EchoStream(IHubClients clients, ConnectionSourceSettings sourceSettings = null,
ConnectionSinkSettings sinkSettings = null)
            : base(clients, sourceSettings, sinkSettings)
        {
            Source
                .Collect(x => x as Received)
                // Tell everyone
                .Select(x => Signals.Broadcast(x.Data))
                // Or tell everyone else, except one-self
                // .Select(x => Signals.Broadcast(x.Data, new[] { x.Request.ConnectionId }))
                // Or just send it back to one-self
                // .Select(x => Signals.Send(x.Request.ConnectionId, x.Data.Payload))
                .To(Sink)
                .Run(AkkaLoad.Materializer);
        }
    }
}
```

STEP3 - SignalR AkkaStream

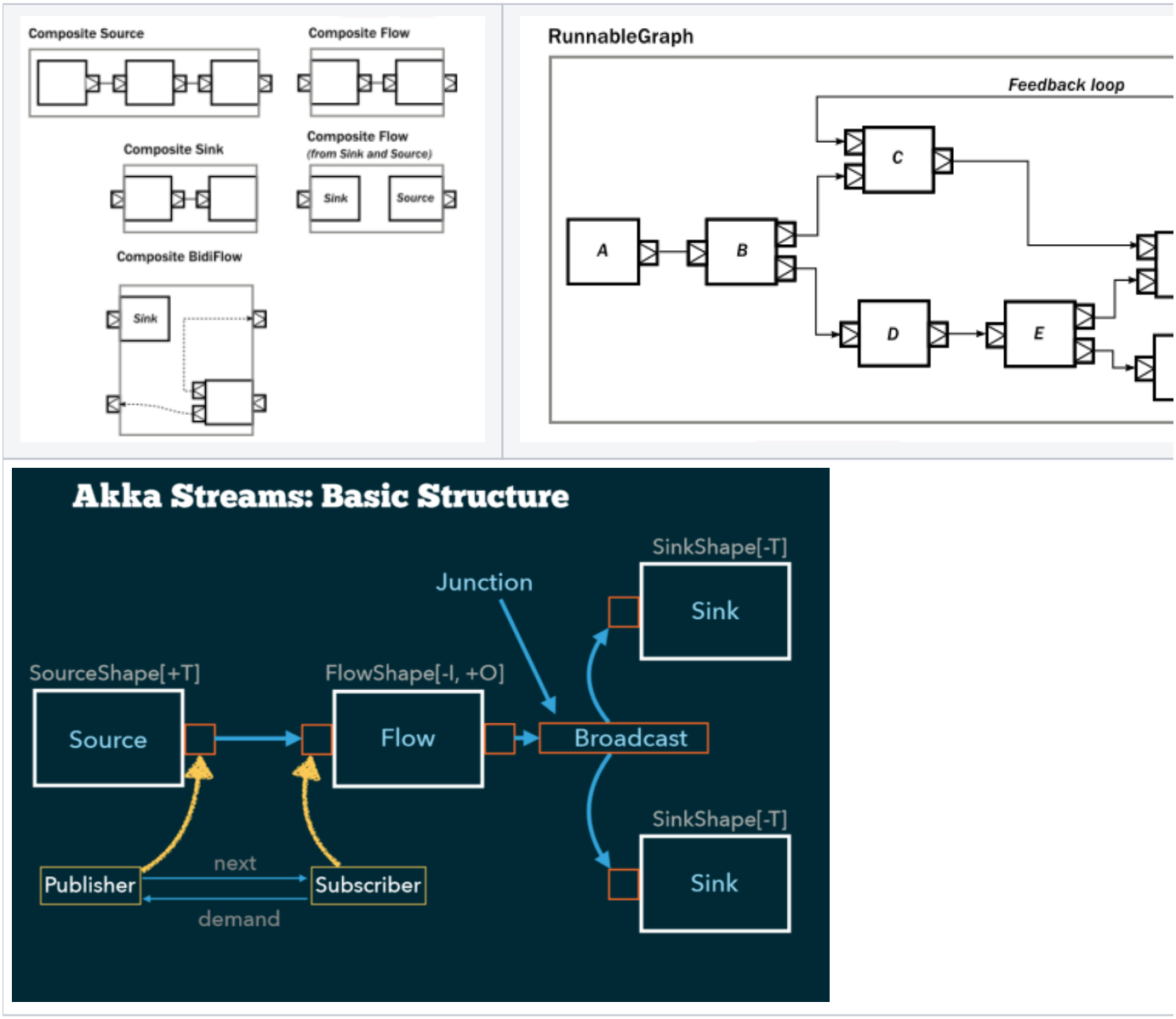
```
public void ConfigureServices(..............
            // Akka
            var envName = Environment.GetEnvironmentVariable("ASPNETCORE_ENVIRONMENT");
            var akkaConfig = AkkaLoad.Load(envName, Configuration);
            actorSystem = ActorSystem.Create("AkkaDotBootSystem", akkaConfig);
            services.AddAkka(actorSystem);

            // Signal R
            services
            .AddSingleton(new ConnectionSourceSettings(102400, OverflowStrategy.DropBuffer))
            .AddSignalRAkkaStream()
            .AddSignalR();

public void Configure(..............
            app.UseStaticFiles()
                .UseRouting()
                .UseEndpoints(endpoints =>
                {
                    endpoints.MapHub<EchoHub>("/echo");
                });
```

SignalR AkkaStream , Endpoint ws endpoint .

AkkaStream



EchoStream Code

```
            Source
                .Collect(x => x as Received)
                // Tell everyone
                .Select(x => Signals.Broadcast(x.Data))
                // Or tell everyone else, except one-self
                // .Select(x => Signals.Broadcast(x.Data, new[] { x.Request.ConnectionId }))
                // Or just send it back to one-self
                // .Select(x => Signals.Send(x.Request.ConnectionId, x.Data.Payload))
                .To(Sink)
                .Run(AkkaLoad.Materializer);
```

AkkaStream .

Flow , .

AkkaStream Flow ,SignalR Stream .

:

- NET AKKA.Stream : https://getakka.net/articles/streams/modularitycomposition.html
- AKKA : https://sslee05.github.io/blog/2018/05/13/akka-stream-01/
- AKKA : https://engineering.linecorp.com/ko/blog/reactive-streams-with-armeria-1/