Actor Clustering with Akka.NET in F#

BY RICCARDO TERRELL - @TRIKACE

Actor Clustering with Akka.NET in F#

BY RICCARDO TERRELL - @TRIKACE

Agenda

- Actor Remoting quick review
- Actor Clustering
- Tabasco & Code Quotations

What is an Actor?

- **Share Nothing**
- **Message are passed by value**
- Light weight processes/threads communicating through messaging
- Communication only by messages
- Lightweight object
- Processing
- Storage – State
- Running on it’s own thread.
- Messages are buffered in a “mailbox”

Akka.NET

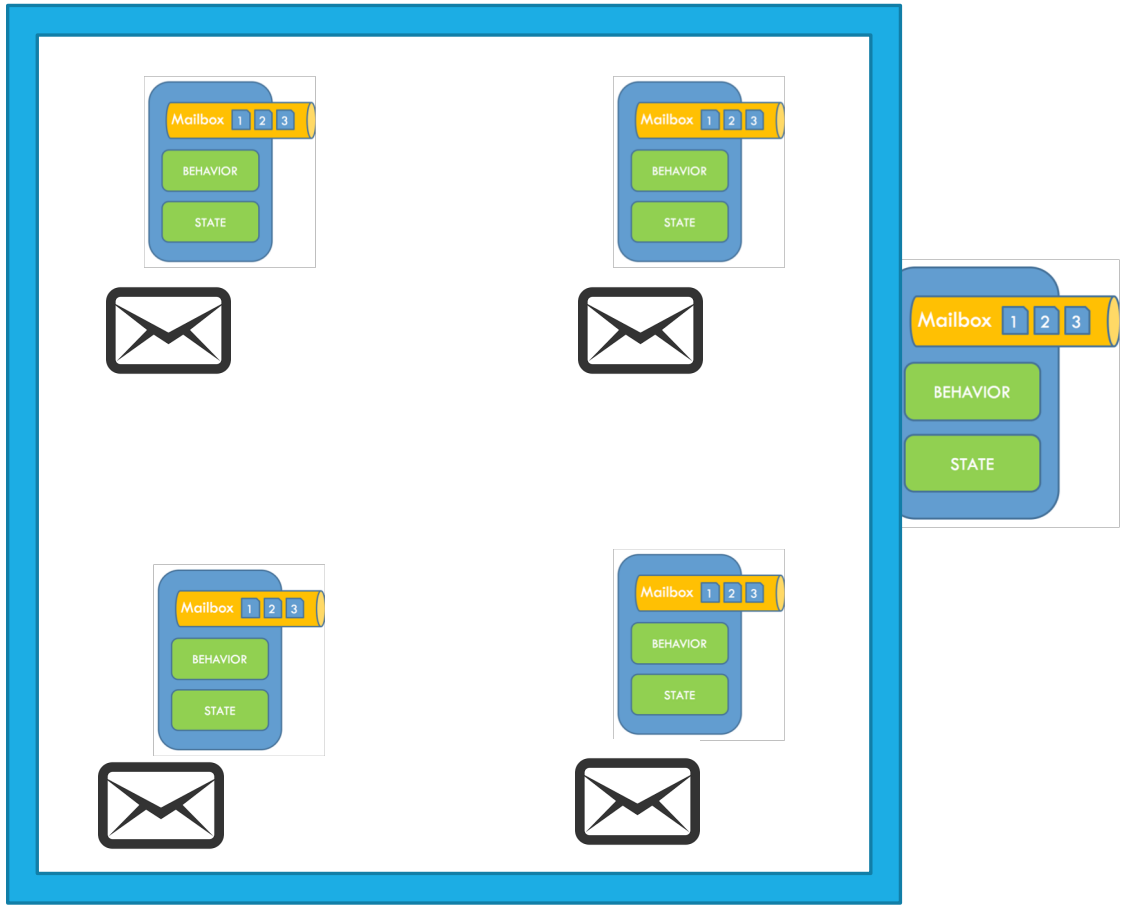
- Akka.Remote
- Akka.Cluster
- Akka.Persistence
- Akka.Streams
- Supervision
- Routing

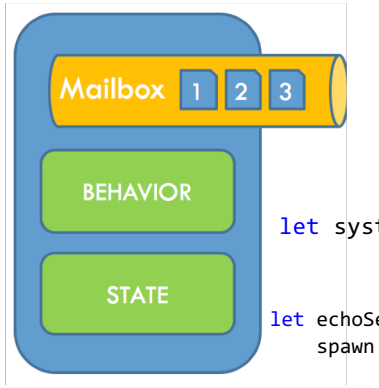
** Concurrent*
** Resilient*
** Distributed*
** Scalable*

Clustering

- Load balancing
- Fault-tolerant
- Scalability (Elastic)
- Zero-Downtime
- Blue-Green deployment

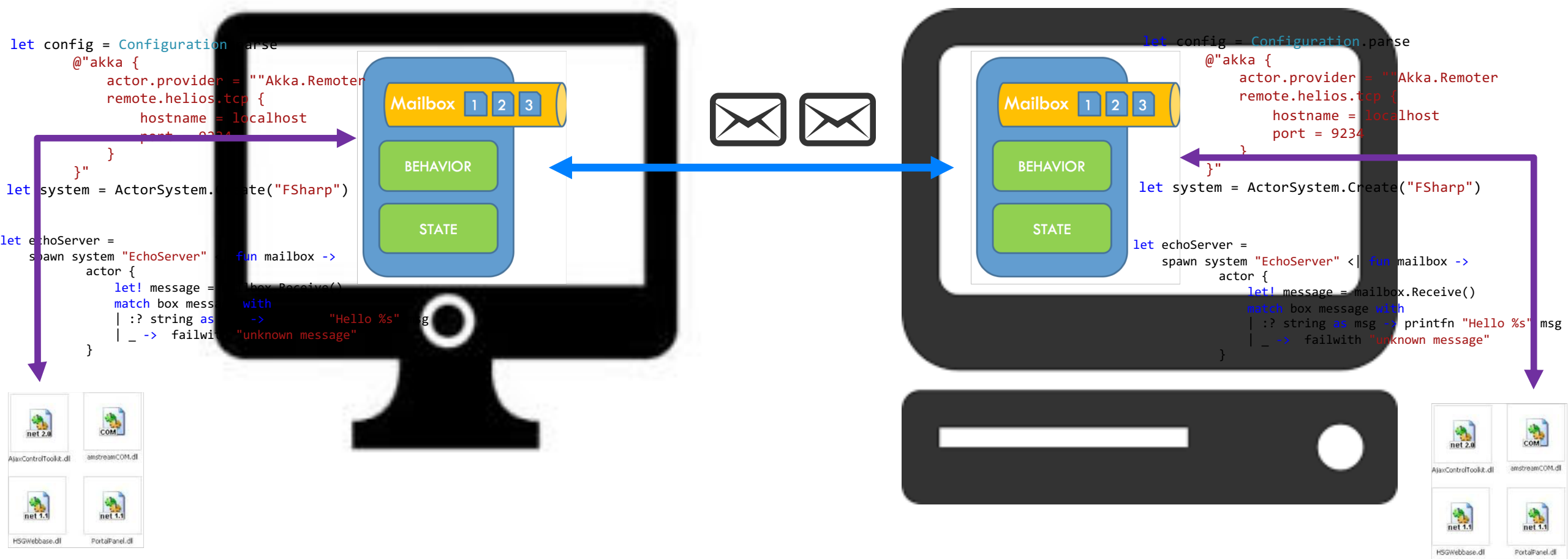
Clustering



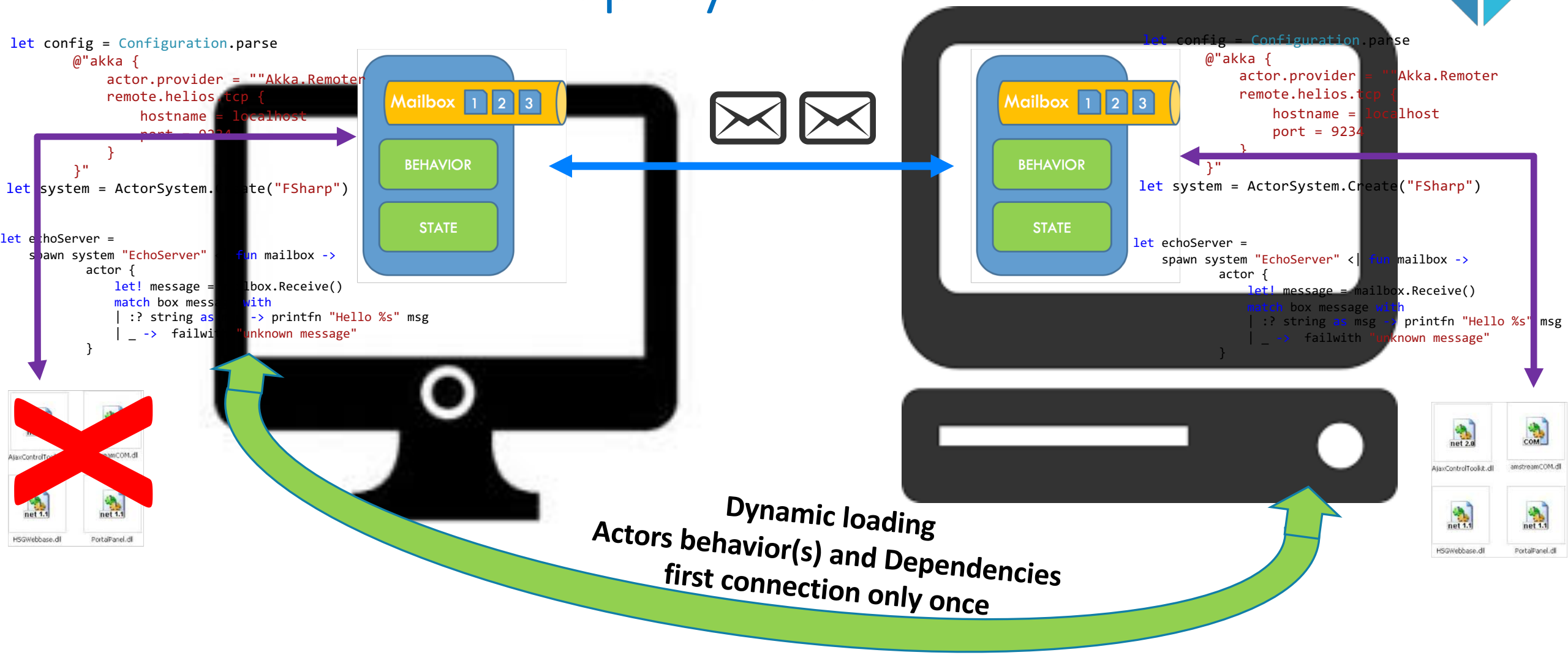



```
let config = Configuration.parse
    @"akka {
        actor.provider = ""Akka.Remoter
        remote.helios.tcp {
            hostname = localhost
            port = 9234
        }
    }"
let system = ActorSystem.Create("FSharp")


let echoServer =
    spawn system "EchoServer" <| fun mailbox ->
        actor {
            let! message = mailbox.Receive()
            match box message with
            | :? string as msg -> printfn "Hello %s" msg
            | _ ->  failwith "unknown message"
        }
```

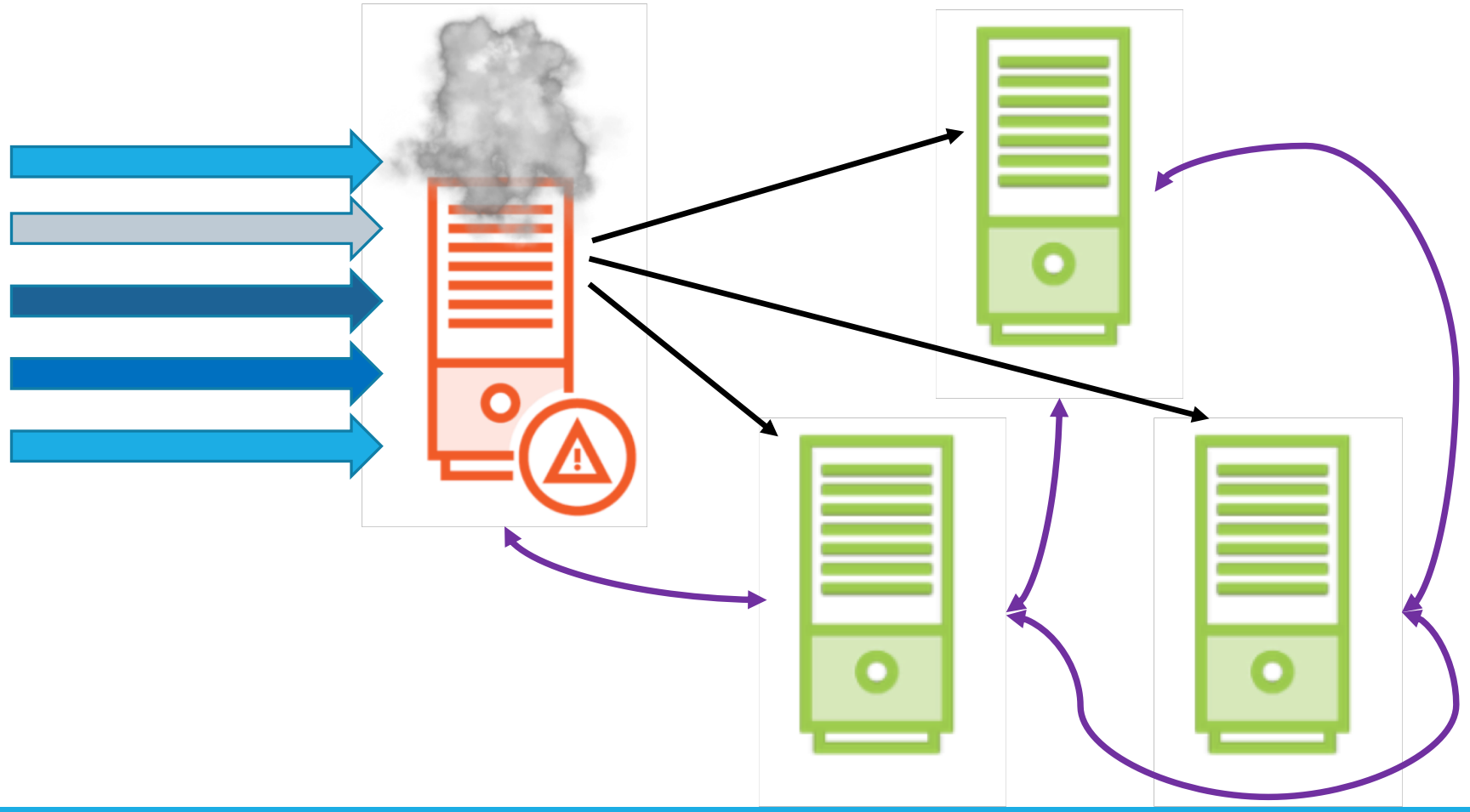
Remoting



Remote full deployment

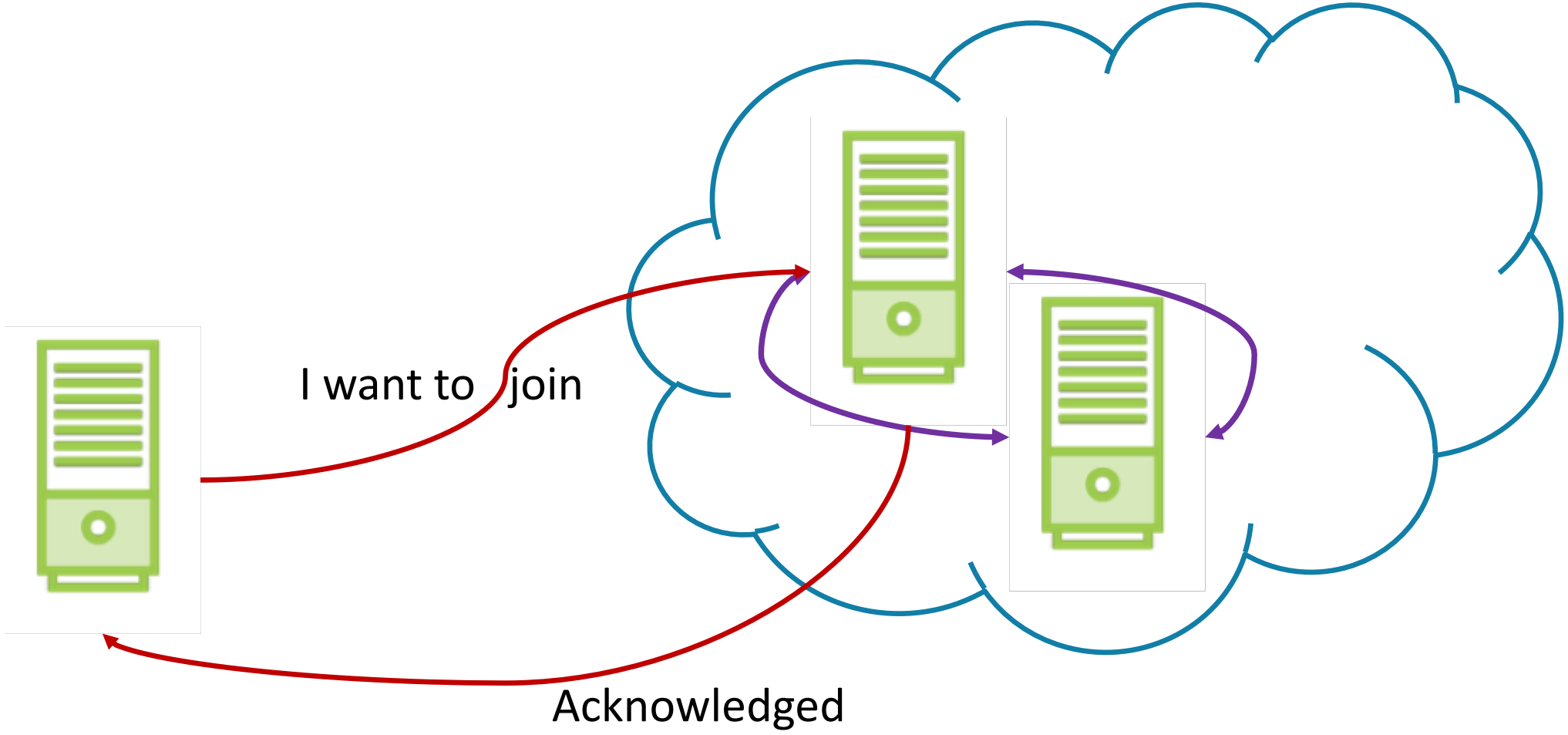


Actor Clustering

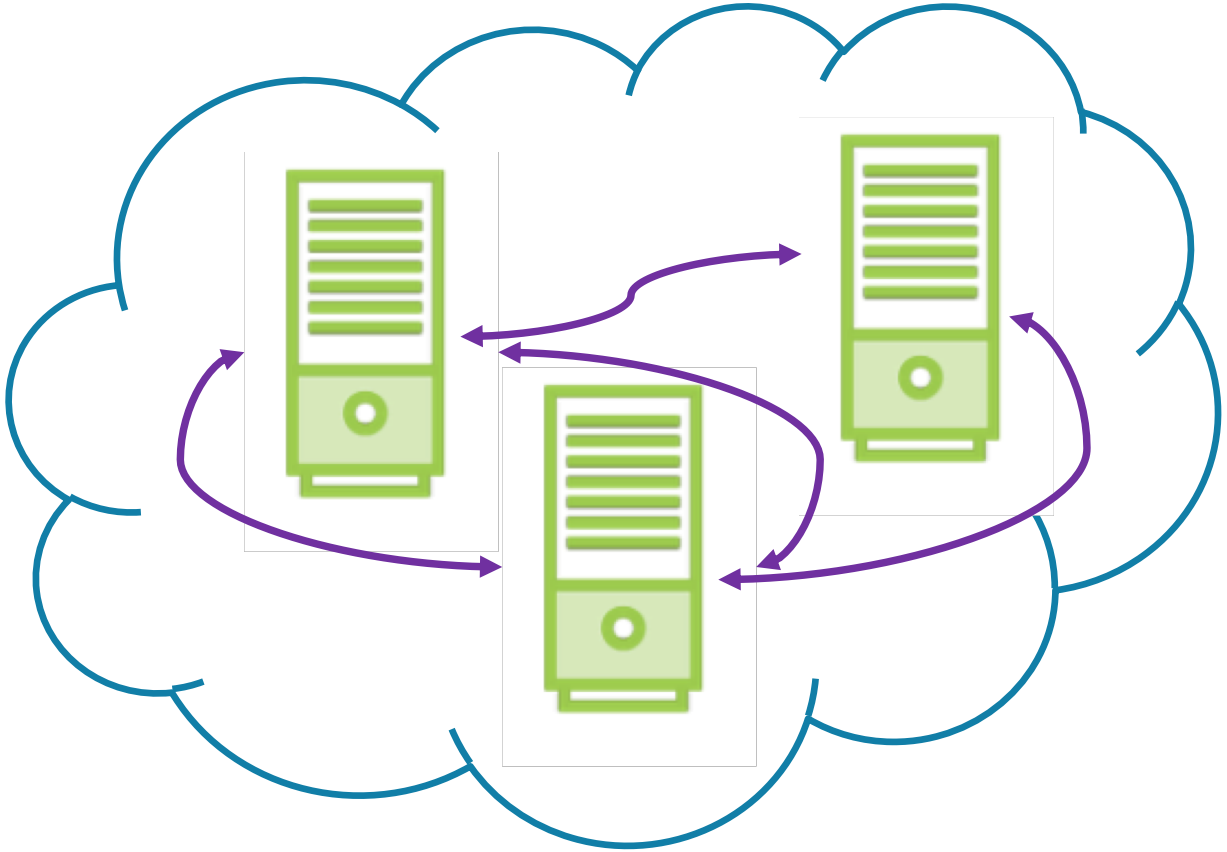


Gossip

Joining the Actor Cluster

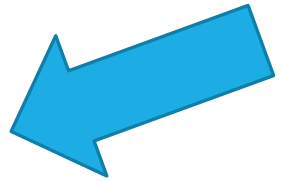


Joining the Actor Cluster



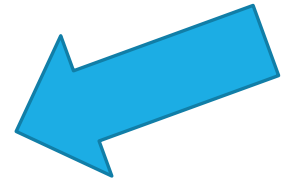
Clustering – Just a configuration matter

```
let workerSystem = System.create "cluster" <| Configuration.parse """
akka {
    actor {
        provider = "Akka.Cluster.ClusterActorRefProvider, Akka.Cluster"
        }
    }
    remote {
        helios.tcp {
            hostname = "localhost"
            port = 0
        }
    }
    cluster {
        seed-nodes = [ "akka.tcp://cluster@localhost:8091/" ]
    }
}
```



Clustering – Just a configuration matter

```
let masterSystem = System.create "cluster" <| Configuration.parse """
akka {
    actor {
        provider = "Akka.Cluster.ClusterActorRefProvider, Akka.Cluster"
        }
    }
    remote {
        helios.tcp {
            hostname = "localhost"
            port = 8091
        }
    }
    cluster {
        seed-nodes = [ "akka.tcp://cluster@localhost:8091/" ]
    }
}
```



<@ Code Quotations @>

<@ F# Code Quotations @>

```
// A typed code quotation.
let sum5 : Expr<int> = <@ 2 + 3 @>
```

```
val sum5 : Quotations.Expr<int> =
  Call (None, op_Addition, [Value (2), Value (3)])
```

<@ F# Code Quotations @>

```
let rec print expr =
    match expr with
    | Call(exprOpt, methodInfo, exprList) ->
        match exprOpt with
        | Some expr -> print expr
        | None -> printf "%s" methodInfo.DeclaringType.Name
        for expr in exprList.Tail do  print expr
    | Int32(n) ->printf "%d" n
    | Value(value, typ) -> printf ”%d” value
```

<@ F# Code Quotations @>

```
let fsum42 = fun x -> x + 42
```

<@ F# Code Quotations @>

```
let fsum42 = fun x -> x + 42
let serializer:BinarySerializer = FsPickler.CreateBinarySerializer()

let sum42:byte[] = serializer.Pickle<@  fsum42  @>
```

<@ F# Code Quotations @>

```
let fsum42 = fun x -> x + 42
let serializer:BinarySerializer = FsPickler.CreateBinarySerializer()

let sum42:byte[] = serializer.Pickle<@  fsum42  @>
let sum42 = serializer.UnPickle<Quotations.Expr<int -> int>> sum42
```

<@ F# Code Quotations @>

```
let fsum42 = fun x -> x + 42
let serializer:BinarySerializer = FsPickler.CreateBinarySerializer()

let sum42:byte[] = serializer.Pickle<@  fsum42  @>
let sum42 = serializer.UnPickle<Quotations.Expr<int -> int>> sum42

let sumfun = QuotationEvaluator.Evaluate sum42

printfn "%d" (sumfun  4) // print 46
```

<@ F# Code Quotations @>

```
let remoter =
        spawne system "remote"
                  fun mailbox ->
                  let rec loop(): Cont<int * string, unit> =
                      actor {
                          let! msg = mailbox.Receive()
                          match msg with
                          | (REQ, m) ->
                              printfn "Remote actor received: %A" m
                              mailbox.Sender() <! (RES, "ECHO " + m)
                          | _ -> logErrorf "Received unexpected message: %A" msg
                          return! loop()
                      }
                  loop()
                  [ SpawnOption.Deploy(remoteDeploy remoteSystemAddress) ]
```

<@ F# Code Quotations @>

```
let remoter =
        spawne system "remote"
              <@  fun mailbox ->
                  let rec loop(): Cont<int * string, unit> =
                      actor {
                          let! msg = mailbox.Receive()
                          match msg with
                          | (REQ, m) ->
                              printfn "Remote actor received: %A" m
                              mailbox.Sender() <! (RES, "ECHO " + m)
                          | _ -> logErrorf "Received unexpected message: %A" msg
                          return! loop()
                      }
                  loop()
               @> [ SpawnOption.Deploy(remoteDeploy remoteSystemAddress) ]
```

Akka.Net + F# API

Hotswap is the ability to change the behavior of an Actor at runtime without shutting down the system

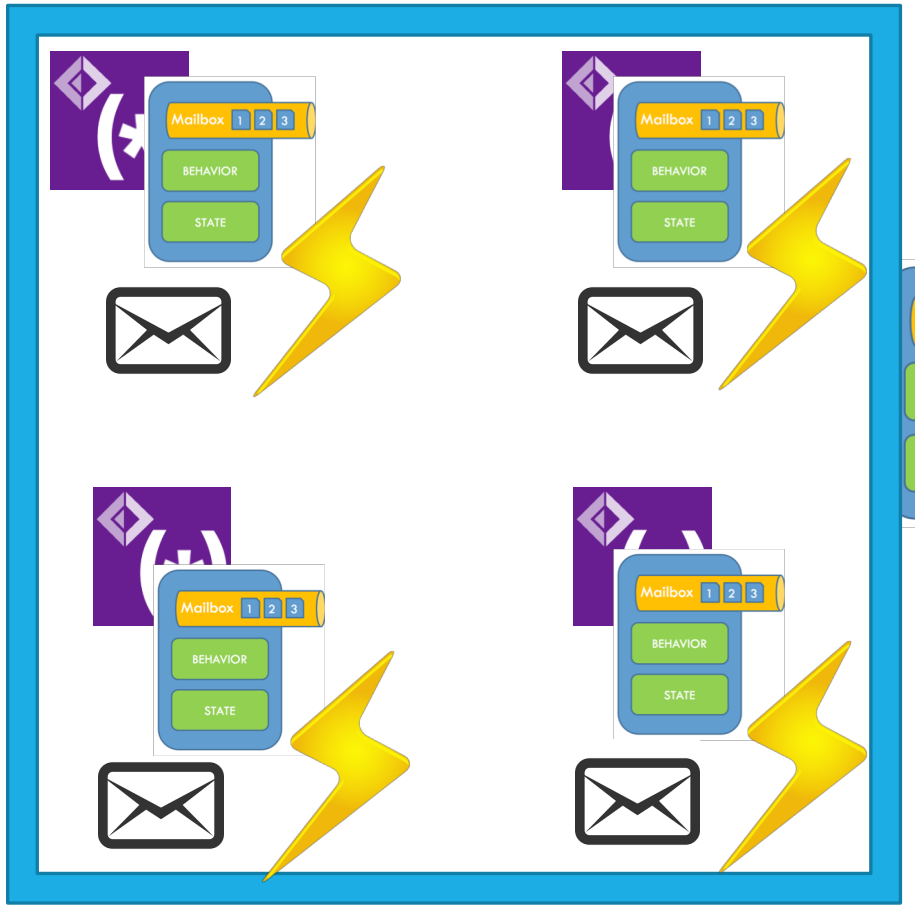
<@ Code Quotations @> are serializable

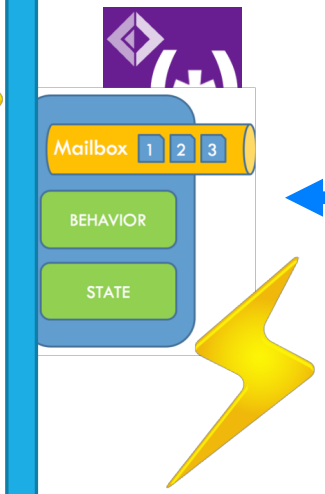
We can deploy arbitrary behavior to a remote Actor!!

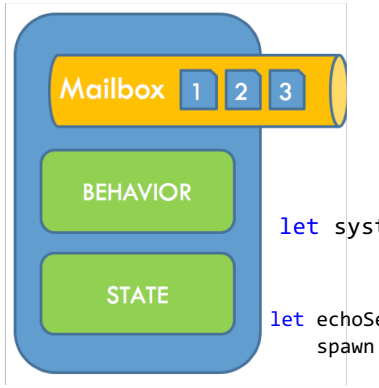
Tabasco

Dynamically load modules
types and dependencies
between different actor systems

Clustering

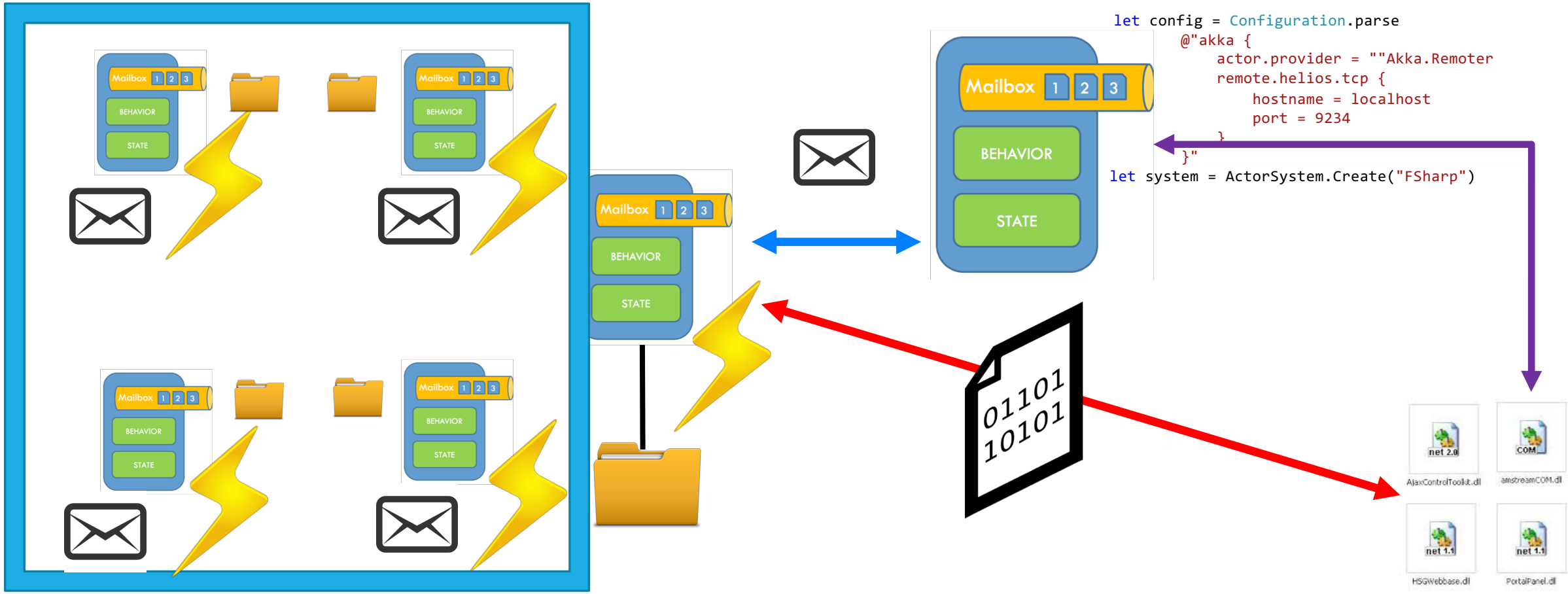







```
let config = Configuration.parse
    @"akka {
        actor.provider = ""Akka.Remoter
        remote.helios.tcp {
            hostname = localhost
            port = 9234
        }
    }"
let system = ActorSystem.Create("FSharp")


let echoServer =
    spawn system "EchoServer" <| fun mailbox ->
        actor {
            let! message = mailbox.Receive()
            match box message with
            | :? string as msg -> printfn "Hello %s" msg
            | _ ->  failwith "unknown message"
        }
```

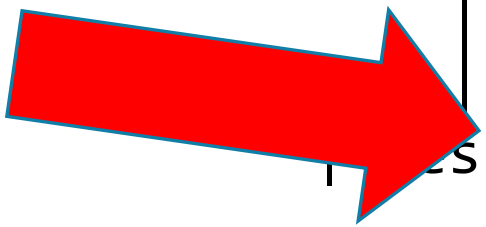
Clustering




```
let config = Configuration.parse
    @"akka {
        actor.provider = ""Akka.Remoter
        remote.helios.tcp {
            hostname = localhost
            port = 9234
        }
    }"
let system = ActorSystem.Create("FSharp")
```

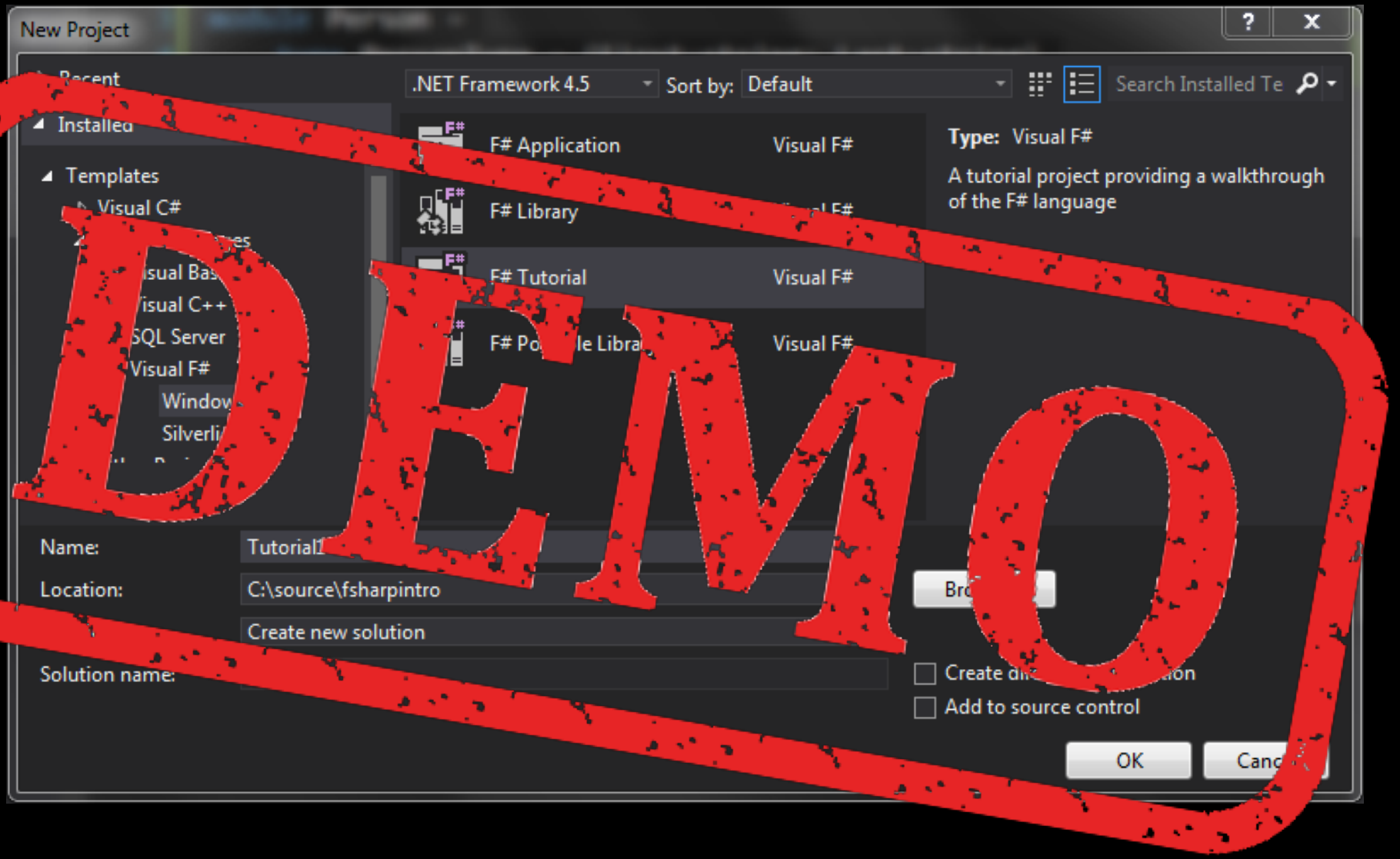
Tabasco - Dynamic Actor

```
type DynamicActor(receive: Receive, zero: obj) as this =
    inherit UntypedActor()

    let untypedContext = UntypedActor.Context :> IActorContext
    let ctx = { Context = untypedContext}
    let mutable currentState: obj = zero
    let mutable currentReceive: Receive = receive

    override this.OnReceive(msg:obj) =
        match msg with
        | :? ActorMessage as message ->
            match message with
            | GetState -> untypedContext.Sender.Tell currentState
            | Stop -> untypedContext.Stop(untypedContext.Self)
            | Load(newReceive) -> currentReceive <- newReceive
        | message -> currentState <- currentReceive ctx
                                     currentState message
```



Tabasco – Assemblies Resolver

```
type internal AssemblyReceiver (vm: ResolverManager, serverRef: IActorRef) =
    member __.ServerRef = serverRef
    interface IRemoteAssemblyReceiver with

        member __.GetLoadedAssemblyInfo (ids:AssemblyId[]) = async {
            let! reply = serverRef <? GetAssemblyInfo ids
            return downcast reply
        }
        member __.PushAssemblies vas = async {
            let! reply = serverRef <? LoadAssemblies (vm.CreateRawAssemblies vas)
            return downcast reply
        }
```


New Project
.NET Framework 4.5
Sort by:
Default
Search Installed Te
Installed
Templates
Visual C#
Visual C++
SQL Server
Visual F#
Silverli
F# Application
Visual F#
F# Library
F# Tutorial
Visual F#
Visual F#
Type: Visual F#
A tutorial project providing a walkthrough of the F# language
Name:
Tutorial1
Location:
C:\source\fsharpintro
Create new solution
Solution name:
Add to source control
OK
DEMO

The tools we use have a profound (and devious!) influence on our thinking habits, and, therefore, on our thinking abilities.

-- Edsger Dijkstra