WEBINAR

Monitoring Microservices in Production with Lightbend Platform

Duncan Devore (@ironfish) & Henrik Engström (@h3nk3)

Agenda

- What is Monitoring?
- Traditional Monitoring
- Monitoring Reactive Applications
- Lightbend Monitoring
  - Overview
  - Under the hood
  - Reporters

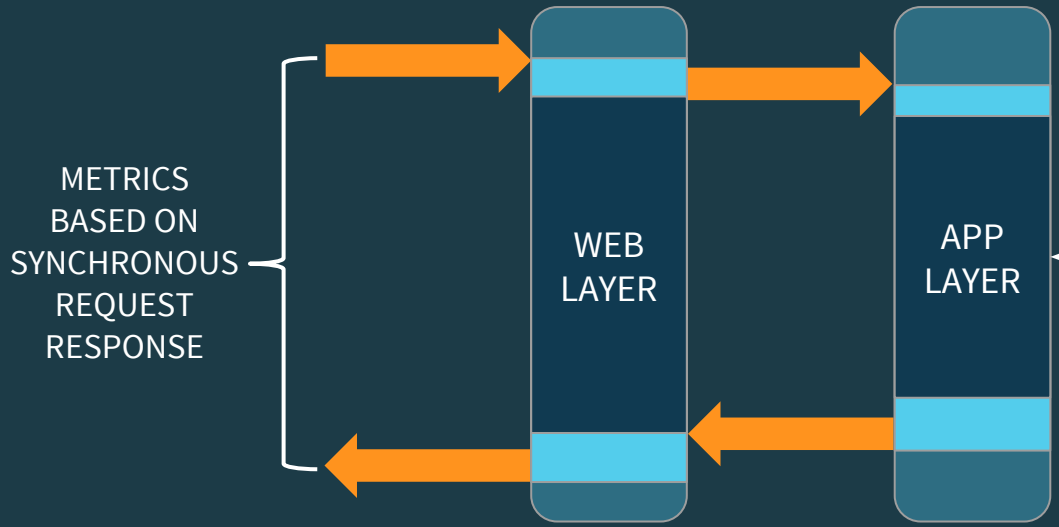
What is Monitoring? - Categories

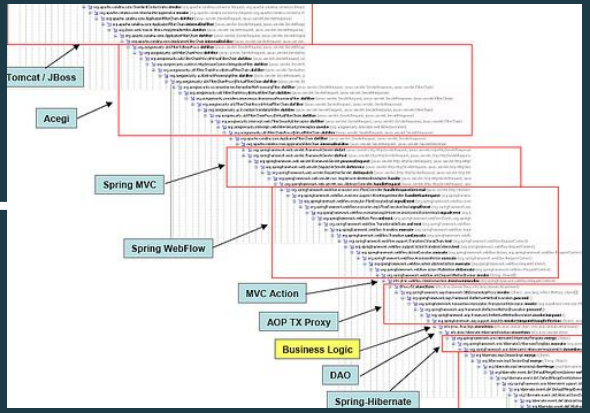
- **Business Process Monitoring**
  - Focus on business process level
- **Functional Monitoring**
  - Focus on use-case level
- **Technical Monitoring**
  - Focus on individual pieces

What is Monitoring? - Types

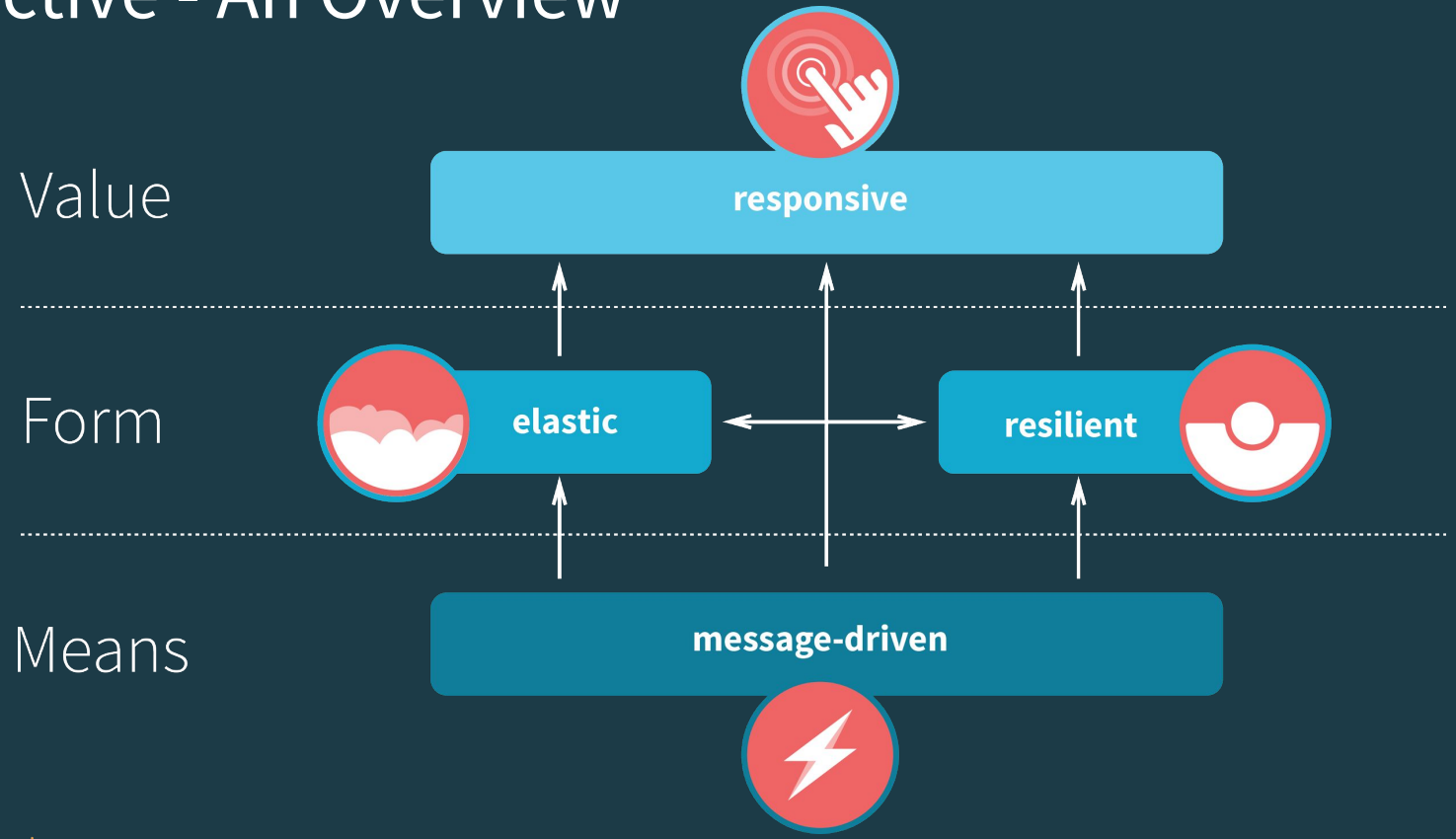
- System Monitoring
- Tracing/Span/Transaction Monitoring
- Performance Monitoring
- Integration Monitoring
- Application Monitoring

Traditional Monitoring: Metrics and Debugging





Reactive - An Overview



Reactive - An Overview

- **Responsive**: reacting in a desired or positive way; quick to react or respond
- **Elastic**: return to original shape/size after being stretched, squeezed, etc.;
- **Resilient**: become strong, or successful again after something bad happens
- **Message**: recorded communication sent to a recipient who cannot be contacted
- **Driven**: operated, moved, or controlled by a specified person or source of power

-- Merriam Webster Dictionary

Architecture Shift : Freedom for all!

Finally, asynchronous, parallel, distributed computing that works for the common person

Distributed Computing Disclaimer: The availability and accuracy of the information delivered by these services and information they portray are contingent on a distributed computing environment comprised of several unique asynchronous, parallel and non-deterministic applications within the said companies multi-clustered system. As a result, the availability, accuracy and access to these services and the data they provide, may or may not be available at any given time. Additionally, the availability, precision and access to dynamic data provided on these multi-clustered services may or may not be dependent on other systems being available at the time of use. The user is cautioned that these services: may or may not be available at any time; may become unavailable without notice; for any reason and for any length of time. Why? Well, because the stack traces produced are pretty much useless, and the supporting development staff must fall back on print-line statements to isoloate the cause of said error.

Monitoring Reactive Applications: The Challenge

Asynchronous and Distributed => traditional ways of monitoring no longer valid...

```
[info]  at cinnamon.sample.failure.B$$anonfun$receive$2.applyOrElse(FailureDemo.scala:102)
[info]  at akka.actor.Actor$class.aroundReceive(Actor.scala:467)
[info]  at cinnamon.sample.failure.B.aroundReceive(FailureDemo.scala:86)
[info]  at akka.actor.ActorCell.receiveMessage(ActorCell.scala:516)
[info]  at akka.actor.ActorCell.invoke(ActorCell.scala)
[info]  at akka.dispatch.Mailbox.processMailbox(Mailbox.scala:238)
[info]  at akka.dispatch.Mailbox.run$$original(Mailbox.scala:220)
[info]  at akka.dispatch.Mailbox.run(Mailbox.scala:29)
[info]  at akka.dispatch.ForkJoinExecutorConfigurator$AkkaForkJoinTask.exec(AbstractDispatcher.scala:397)
[info]  at scala.concurrent.forkjoin.ForkJoinTask.doExec(ForkJoinTask.java:260)
[info]  at scala.concurrent.forkjoin.ForkJoinPool$WorkQueue.runTask(ForkJoinPool.java:1339)
[info]  at scala.concurrent.forkjoin.ForkJoinPool.runWorker(ForkJoinPool.java:1979)
[info]  at scala.concurrent.forkjoin.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:107)
```

Monitoring Reactive Applications: The Challenge

Asynchronous and Distributed means:

- The path to where you are is no longer available
- You don’t know where your code will execute
- The epitome of non-determinism

What to do??

Monitoring Reactive Applications: Tracing

- Traces follow asynchronous message flows
- Create a unique entry point
- Flows through multiple actors via message sends
- Create a unique ending point
- The result is a trace span or “hot path”
- From this we can:
  - Gather metrics on time-to-complete
  - Derive latency metrics
  - Establish a base for flow visualization

Monitoring Reactive Applications: Tracing

```
import akka.actor.{ Actor, Props }
import com.lightbend.cinnamon.akka.Tracer

class ActorA extends Actor {
  val actorB = context.actorOf(Props[ActorB], "b")
  def receive = {
    case "begin" =>
      Tracer(context.system).start("myTraceName") {
        actorB forward "doWork"
      }
  }
}
```

Lightbend

Monitoring Reactive Applications: Tracing

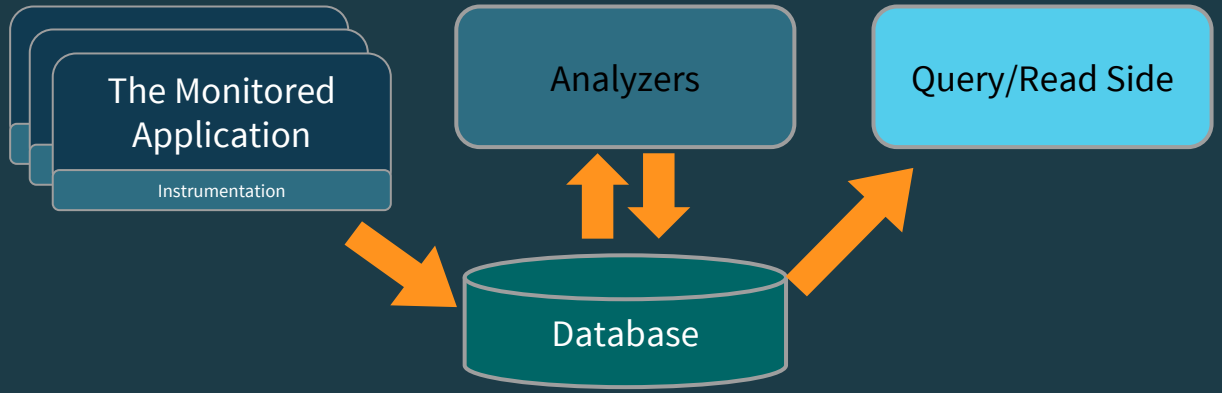
```
class ActorB extends Actor {
  def receive = {
    case "doWork" =>
      doSomeWork()
      Tracer(context.system).end("myTraceName")
  }

  def doSomeWork(): Unit = ???
}
```

Output:

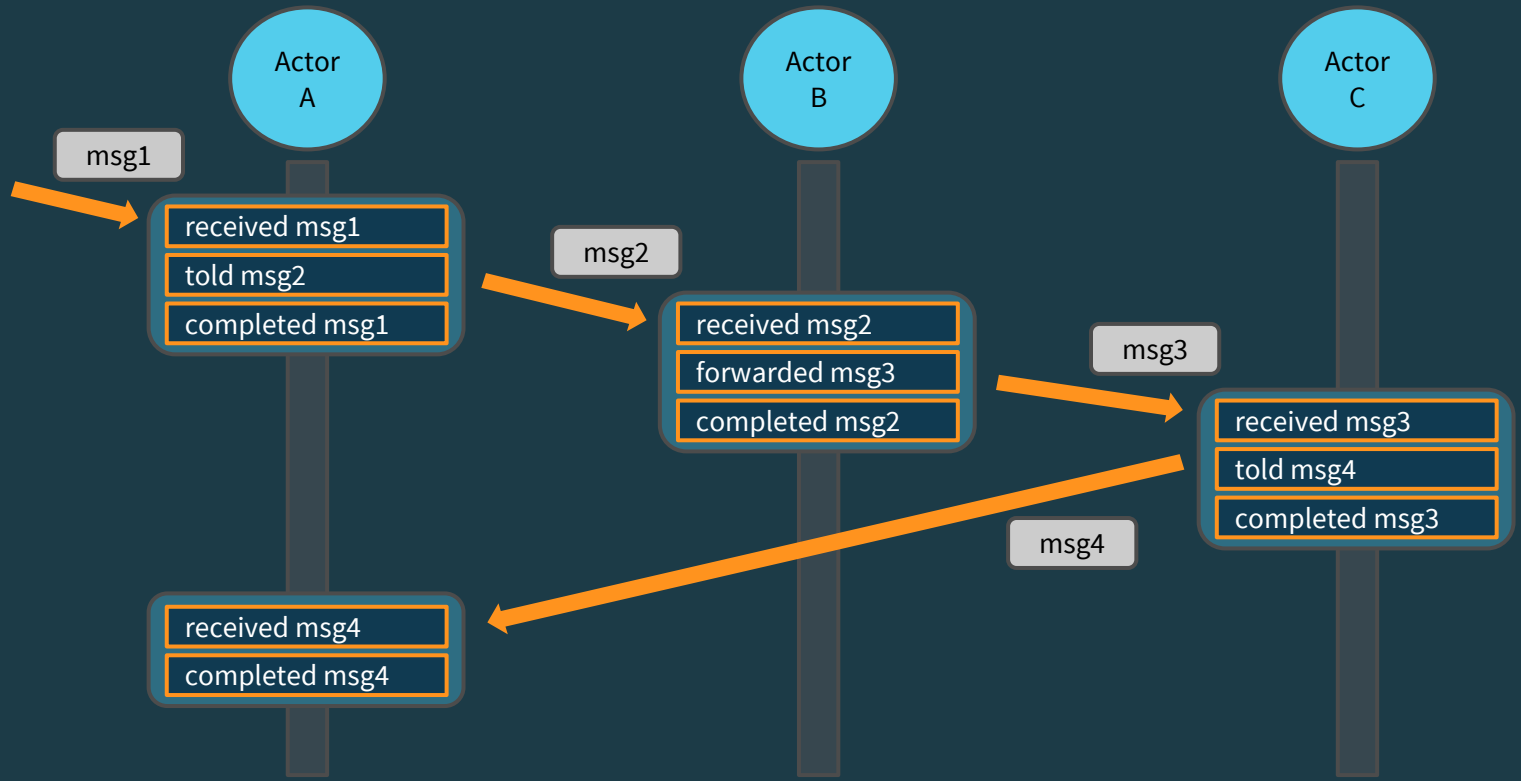
```
[metrics-logger-reporter-1-thread-1] INFO  output-sink - type=HISTOGRAM, name=traces.myTraceName.
trace-histogram, count=109, min=266598801, max=4344502206, mean=2.132377393674398E9, stddev=1.
3304223746691678E9, median=1.84076406E9, p75=3.410501492E9, p95=4.157100469E9, p98=4.276913089E9,
p99=4.283057143E9, p999=4.344502206E9
```

Monitoring Reactive Applications: Dapper

- Dapper paper from Google in 2010 - Highly Influential
- Typesafe's implementation of the Dapper paper was called Typesafe Console (2012 - 2013)



Monitoring Reactive Applications: Trace Info.



Monitoring Reactive Applications: Trace Events

```
"id" : "4RHh8WFt3T6JkhKx9rj2qQ==",
"trace" : "4RHh8ZFx2z6JkhKx9rj2qQ==",
"parent" : "4RHh8UAf3T6JkhKx9rj2qQ==",
"host" : "192.168.99.100",
"nanoTime" : "13462485522467120000",
"annotation" : {
    "type" : "received",
    "actor" : "akka://SomeActorSystem/user/ActorA"
}
```

Monitoring Reactive Applications: Associated Cost

- Which Akka application is more costly to monitor?
  - Application with few actors with lots of business logic
  - Application with several temporary, short-lived, actors
- Cost associated with each “event” in the application
- Each application has its own set of characteristics
- Performance overhead is therefore **very difficult** to predict!

Monitoring Reactive Applications: Cost Mitigation

- Flexible Configuration
- Sampling Capabilities
- Delta Approach

Each cost mitigation has its drawbacks: less insight or higher complexity. When it comes to monitoring; ***lunch != free***

Lightbend Monitoring

- Project started 1.5 years ago
- Support for Akka actors and Lagom circuit breakers
- Generates metrics, events and traces
- Support for various backends
  - StatsD, Takipi, MDC, etc.
  - Miscellaneous Coda Hale reporters
- Monitoring requires subscription with Lightbend
  - *Free to try out during development!*

Lightbend Monitoring - v2.0 (released in August)

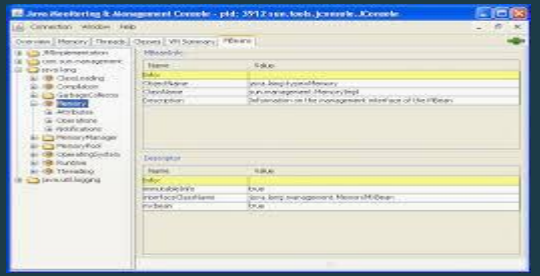
- Agent based - enables instrumentation of more libraries
  - Used to be pre-instrumented JAR files
  - Supports OS Akka 2.3.x and 2.4.x out of the box
- ***Lagom*** *circuit breaker metrics*
- More reporters - e.g. ElasticSearch integration
- Simplified setup for sbt, Maven and Gradle
- Improved configuration and documentation

Lightbend Monitoring - Overview

The Monitored Application — Instr. & Metrics Events Traces

The Monitored Application — Instr. & Metrics Events Traces

The Monitored Application — Instr. & Metrics Events Traces



Lightbend Monitoring: 2.x (x > 0)

- Monitoring of Dispatchers and Thread Pools
- “Flows” a.k.a. Futures and Streams
- Improved monitoring of Lagom
- **ConductR** based “packaged solution” for easy exploration
  - Provided UI dashboards out of the box
  - Uses Elasticsearch, Grafana and Kibana

Lightbend Monitoring - Under the hood

- Uses a Java agent
  - Start with “java -javaagent:cinnamon-agent.jar ...”
    - Documentation describes how to set this up for sbt, Maven and Gradle
- Agent instruments code during load-time
- Instrumented code calls a SPI

Lightbend Monitoring - Instrumented Code

```
// Inside the Akka ActorCell class...
protected def create(failure: Option[ActorInitializationException]):Unit = {
  //...
  try {
    val created = newActor()
    //...
    system.instrumentation.actorStarted(self)
  } catch {
   //...
  }
}
```

Lightbend Monitoring - SPI

```
public abstract class ActorInstrumentation {

    public abstract void actorCreated(ActorRef actorRef, MessageDispatcher
            dispatcher);

    public abstract void actorStarted(ActorRef actorRef);

    public abstract void actorStopped(ActorRef actorRef);

    //...
}
```

Lightbend Monitoring - Under the hood

Backend Plugins

- CodaHale Metrics
- Takipi

Lightbend Monitoring - Under the hood

Metrics Abstractions

- MetricsBackend & Factory
  - Counter
  - Guage
  - Rate
  - Recorder

```
/**
 * {@code Recorder} metric for measuring changing
 *      values (like durations).
 */
public interface Recorder extends Metric {
    /**
     * Record a new value.
     */
    void record(long value);

    /**
     * Empty no-op {@code Recorder}.
     */
    Recorder NONE = new Recorder() {
        @Override
        public void record(long value) {

        }

        @Override
        public void destroy() {

        }
    };
}
```

Lightbend Monitoring - Configuration

MetricsFactory by Config

```
# Automatically set up the Coda Hale Metrics backend
cinnamon.backends += chmetrics

cinnamon {
  chmetrics {
    backend-class = "cinnamon.chmetrics.CodaHaleBackend"

    ... more config
  }
}
```

Lightbend Monitoring - Reporters

CodaHale Metrics Reporters

- JMX
- Console
- ElasticSearch
- SLF4J
- HTTP
- StatsD
- Memory usage
- Garbage Collection
- Class loading

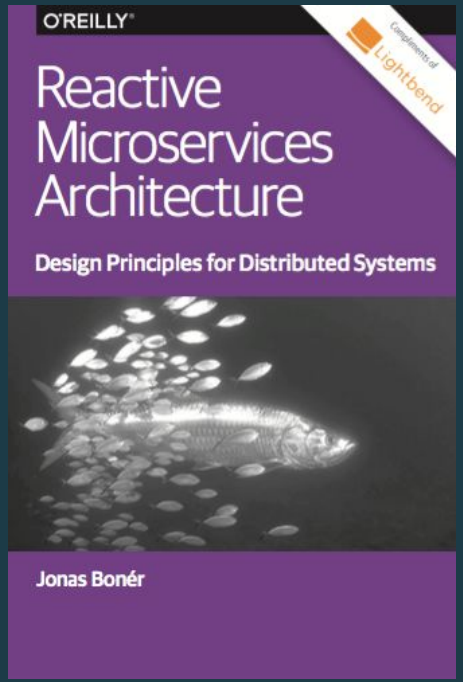
```
cinnamon {
  chmetrics {
    show-config-info = true
    show-loaded-config-settings = true

    reporters += "console-reporter"
  }

  akka.actors {
    "sample.bottleneck.*" = {
      report-by = class
    }
  }

  trace.thresholds {
    "hotpath" = 60s
  }
}
```

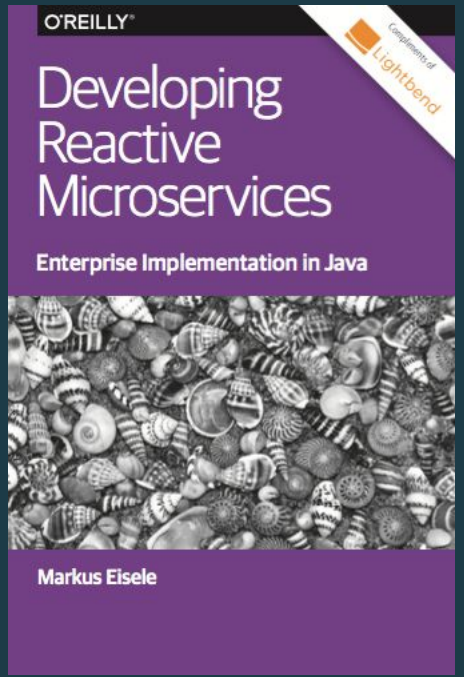
Lightbend Monitoring - Information

- “Sounds like this is cool, how can I try it out?”
  - Commercial, but free use in development
  - https://www.lightbend.com/products/monitoring
- Checkout the docs for more info
  - http://monitoring.lightbend.com/docs/latest/home.html
- Checkout our online demo
  - http://demo.lightbend.com

Upgrade your grey matter

Two free O'Reilly eBooks by Lightbend





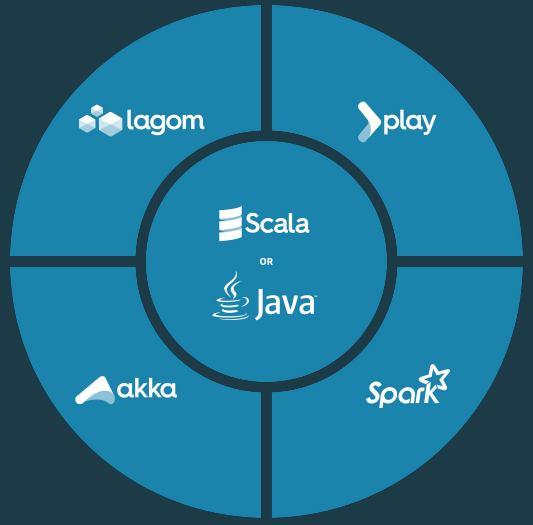
http://bit.ly/ReactiveMicroservice

http://bit.ly/DevelopReactiveMicroservice

Proof of Value Service

Accelerate Project Success

lightbend.com/pov



Reactive Roundtable

World Tour by Lightbend

lightbend.com/reactive-roundtable

Thanks for listening!