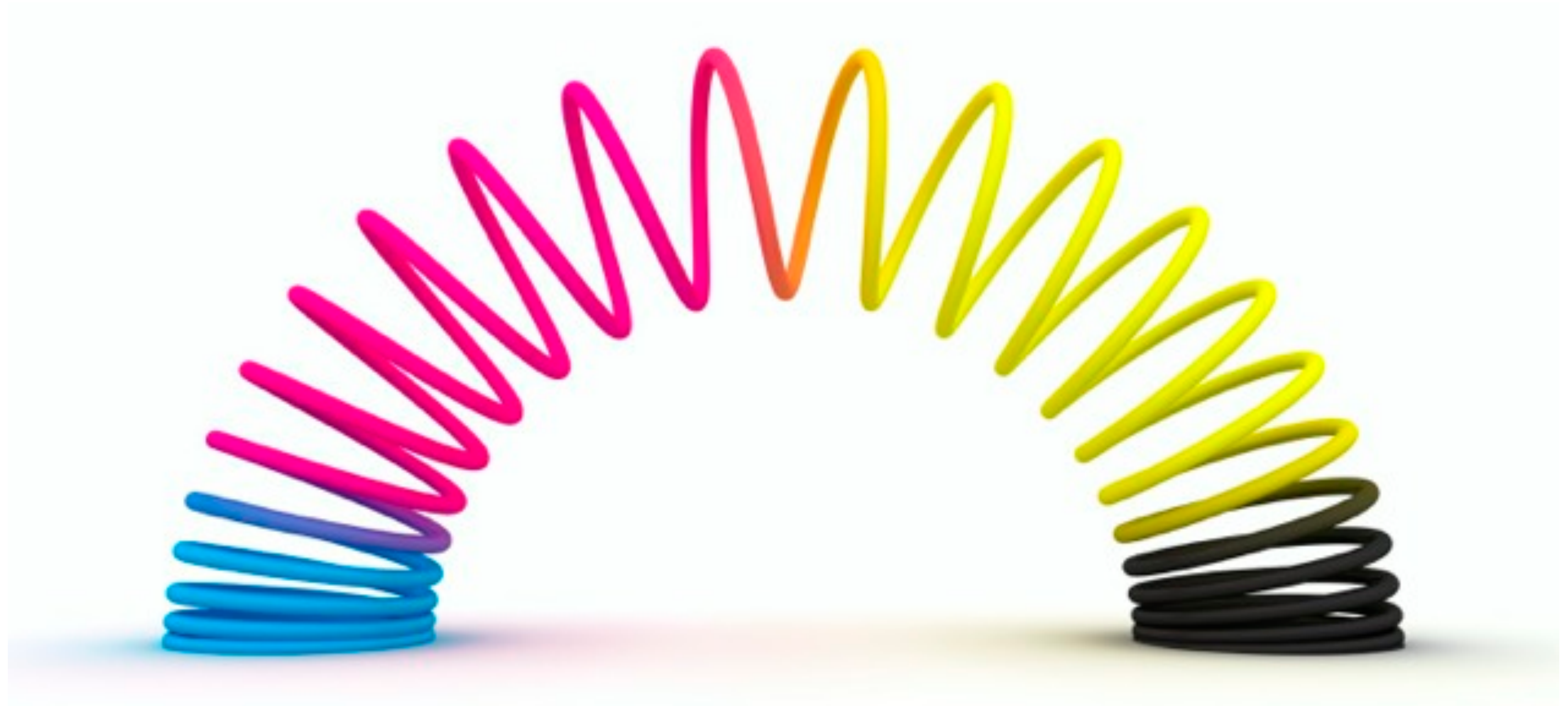
ElasticMQ
a fully async, Akka-based Amazon SQS server

Adam Warski
SoftwareMill



What is Amazon SQS?

- MQ-as-a-service
- Send, Receive, Delete
- At-least-once delivery

How to test SQS apps?

1. Don't?

How to test SQS apps?

2. Just use SQS?

How to test SQS apps?

3. Use a local SQS server

ElasticMQ

- (relevant) subset of SQS
- In-memory
- Lightweight

Stand-alone

```
$ java -jar elasticmq-server-0.7.1.jar

[main] INFO  org.elasticmq.server.Main$ - Starting ElasticMQ
server (0.7.1) ...

[main] INFO  o.e.rest.sqs.TheSQSRestServerBuilder - Started SQS
rest server, bind address 0.0.0.0:9324, visible server address
http://localhost:9324

[main] INFO  org.elasticmq.server.Main$ - === ElasticMQ server
(0.7.1) started in 1444 ms ===
```

Using ElasticMQ

```
import com.amazonaws.auth.BasicAWSCredentials
import com.amazonaws.services.sqs.AmazonSQSClient

client = new AmazonSQSClient(new BasicAWSCredentials("x", "x"))
client.setEndpoint("http://localhost:9324")

val queueUrl = client.createQueue(
  new CreateQueueRequest("testQueue1"))

client.sendMessage(new SendMessageRequest(queueUrl, "Hello!"))
```

Embedded

```
<dependency>
   <groupId>org.elasticmq</groupId>
   <artifactId>elasticmq-rest-sqs_2.10</artifactId>
   <version>0.7.1</version>
</dependency>

val server = SQSRestServerBuilder
   .withPort(9325)
   .withInterface("localhost")
   .start()

// ... use ...

server.stopAndWait()
```

That's all!
Thanks!

Technologies

- Scala
- Akka
- Spray

➡ reactive

Asynchronous: why?

- Traditional model could work well?
- **Long polling**

Receive message walk-through

```
import spray.routing.SimpleRoutingApp

val routes = sendMessage ~ receiveMessage ~ createQueue ~ ...

val app = new SimpleRoutingApp {}
app.startServer(interface, port, "...") {
   routes
}
```

Receive message walk-through

```
val receiveMessage =
  action("ReceiveMessage") {              // path("") and param()
    param("VisibilityTimeout".as[Int]?,
          "WaitTimeSeconds".as[Long]?) {
      (visibilityTimeout, waitTimeSeconds) =>

      respondWithMediaType(MediaTypes.`text/xml`) {
        // inner route: RequestContext => Unit
      }
    }
  }
```

Receive message walk-through

```
// inner route: RequestContext => Unit
ctx: RequestContext =>
  val actorMsg = ReceiveMessages(visibilityTimeout,
                                 waitTimeSeconds)

  val msgs: Future[List[Message]] = queueActor ? actorMsg

  msgs.map { msgs =>
    ctx.complete(
      <ReceiveMessageResponse>
        ...
      </ReceiveMessageResponse>
    ) }
```

Receive message walk-through

```
import akka.actor.{Actor, ActorRef}

class QueueActor extends Actor {
  val messageQueue = mutable.PriorityQueue[InternalMessage]()
  val awaiting = mutable.PriorityQueue[ActorRef]()

  def receive = {
    case ReceiveMessages(...) => {
      // if there are messages, reply
      // otherwise put the sender aside
      // schedule a timeout in 20 seconds
  } } }
```

Dataflow

- Write async code as if it was sync!
- Many Futures, if-s => trouble
- Alternative: Scala Async

Dataflow

```
val result: Future[ActorRef] = flow {
 (queueManager ? Lookup(name)).apply() match {
   case Some(queueActor) => queueActor
   case None =>
     val createFuture = queueManager ? Create(name)
     createFuture.apply()
 }
}
```

Links

- http://github.com/adamw/elasticmq
- http://akka.io/
- http://spray.io/
- http://warski.org

There's more!

- "The ideal module system and the harsh reality"
- Today, 17:50, Room 9

Thank you; Come & get a sticker

http://codebrag.com/devoxx/