Integration Tests with Docker, Scala, Akka & ScalaTest

WoQ
WAY OF QUALITY

Andreas Gies
Lead Architect

- Technical Consulting
- Conference speaking slots

- Technical Trainings
- Management Seminars
- Team Breakouts

Coding by the sea

Yoga & learning

Team Building

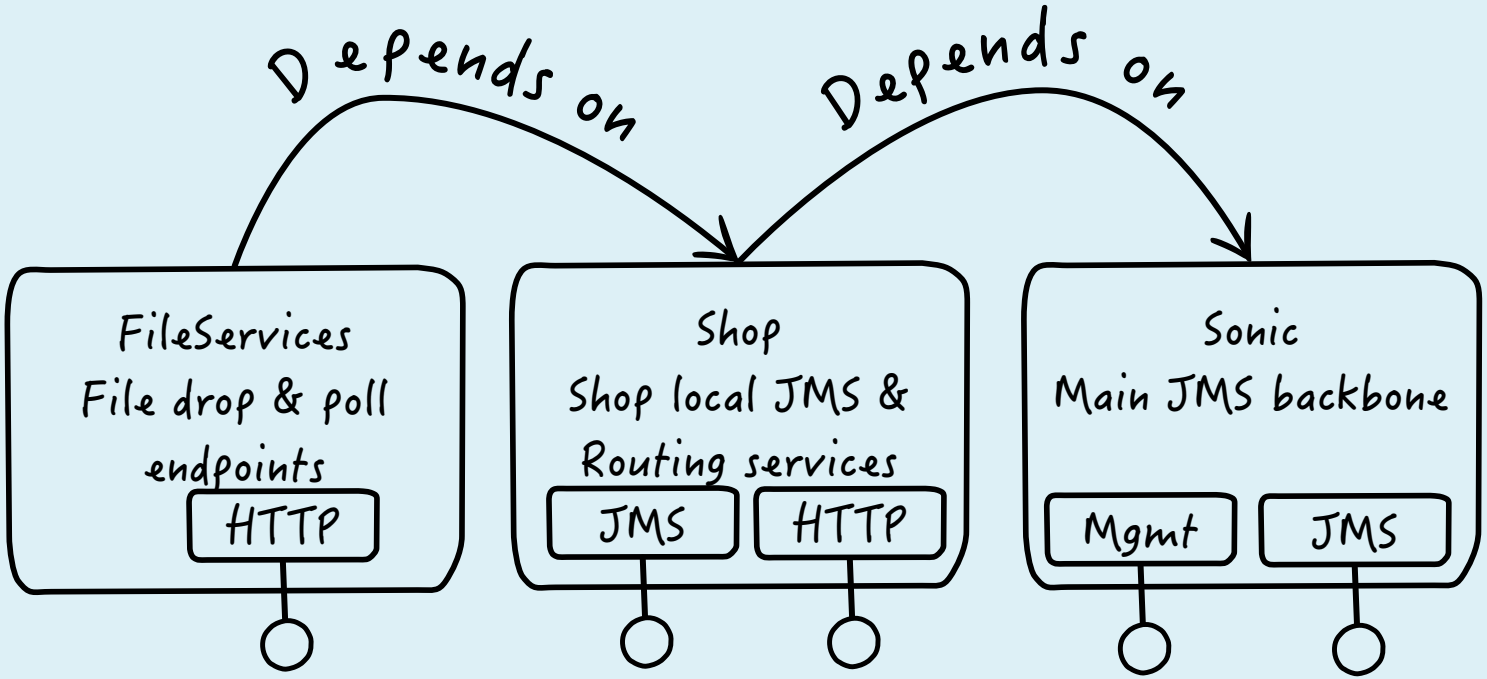
Enjoy Andalucia

Project context

- Java based, distributed retail application
  - Overall functionality as Messaging Backbone
  - Different container types in collaboration
- ~7 years in development, OSGi based
- Migration to OSGi/Scala/Akka since 2011
- Inherited with weak Specs and only manual Tests

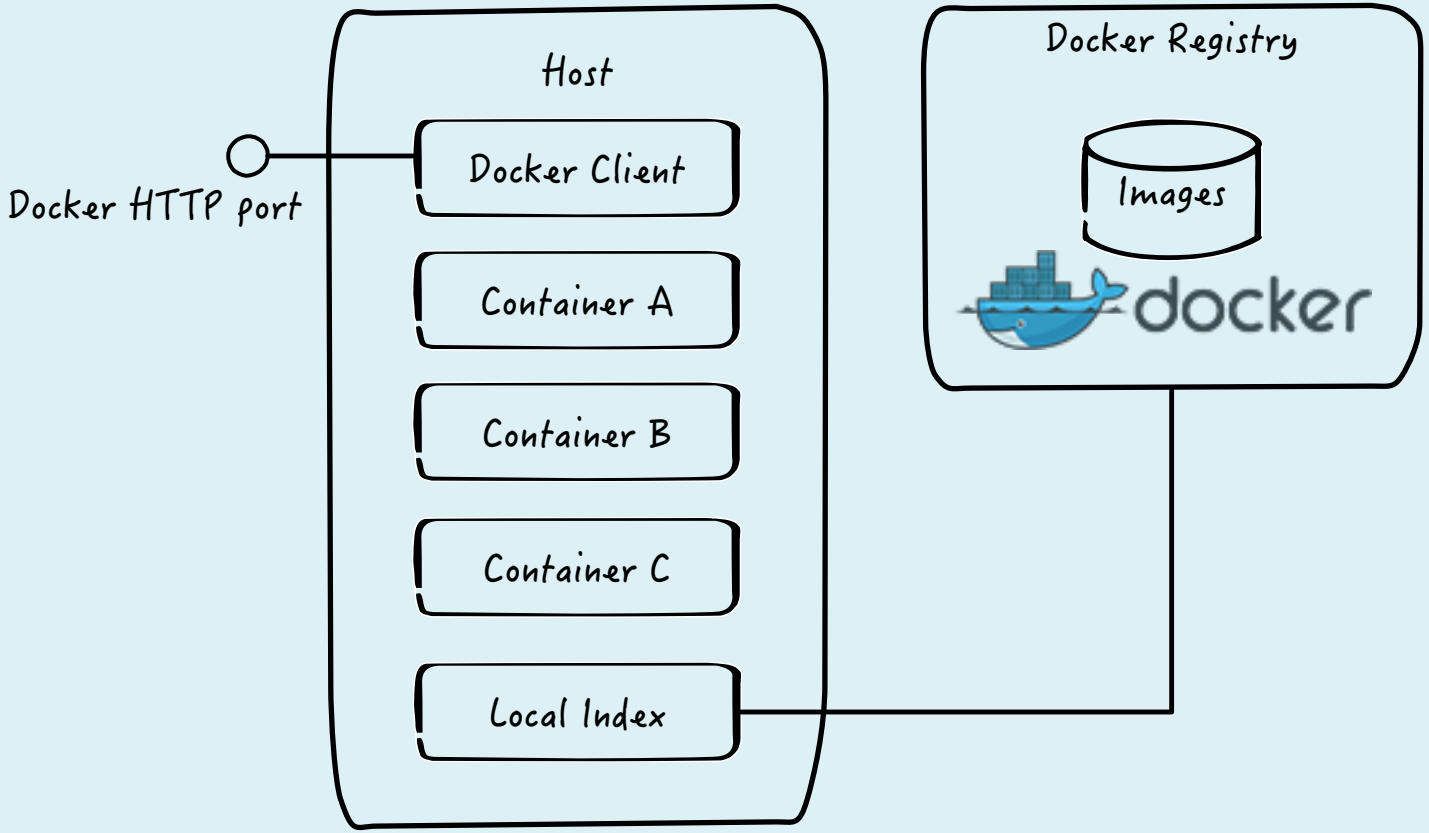
SAMPLE TEST

A sample test scenario



CREATING DOCKER IMAGES

Docker in a nutshell



Layering Docker images

Application
Distributables & Configuration

Customization
i.e. Additional libs, tools, services etc

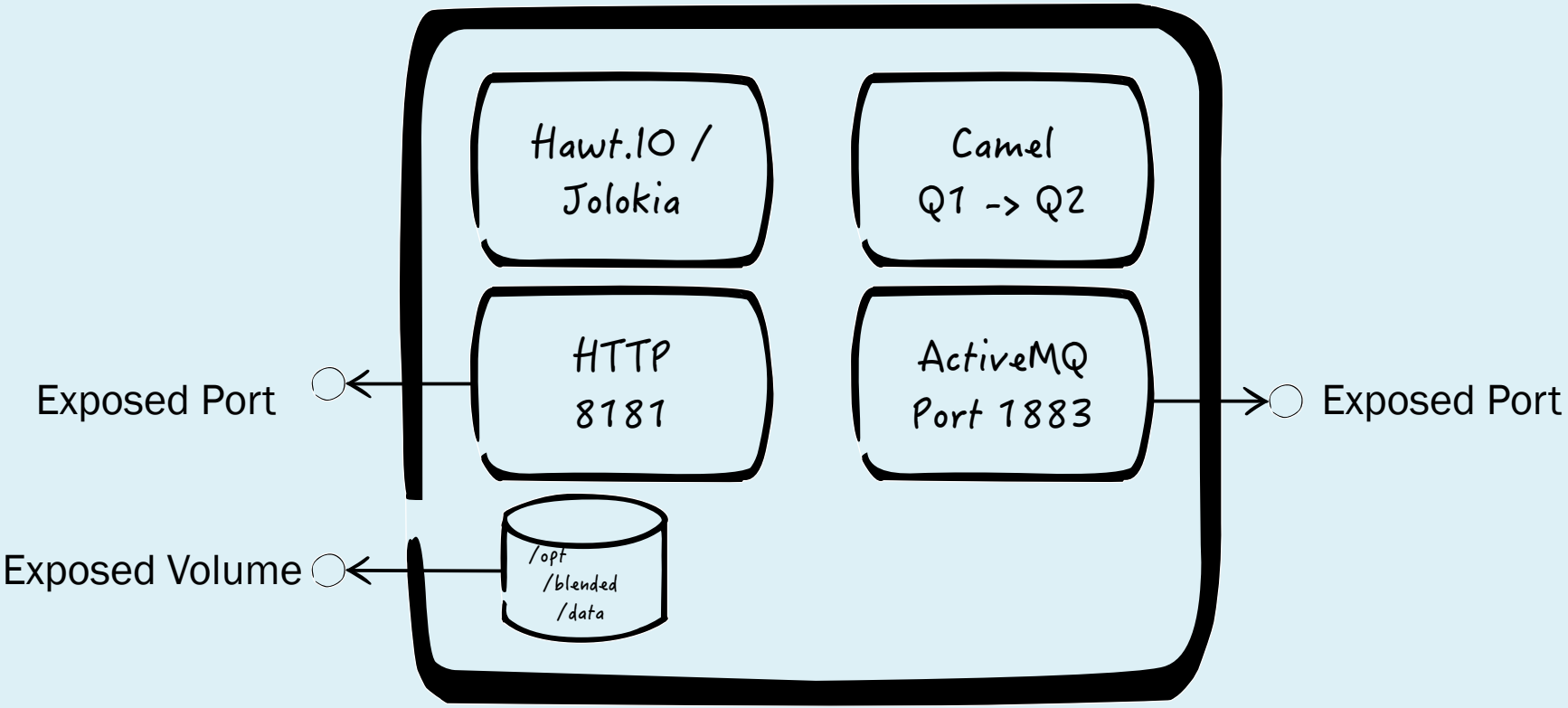
Fitting base image from Registry

Docker container configuration

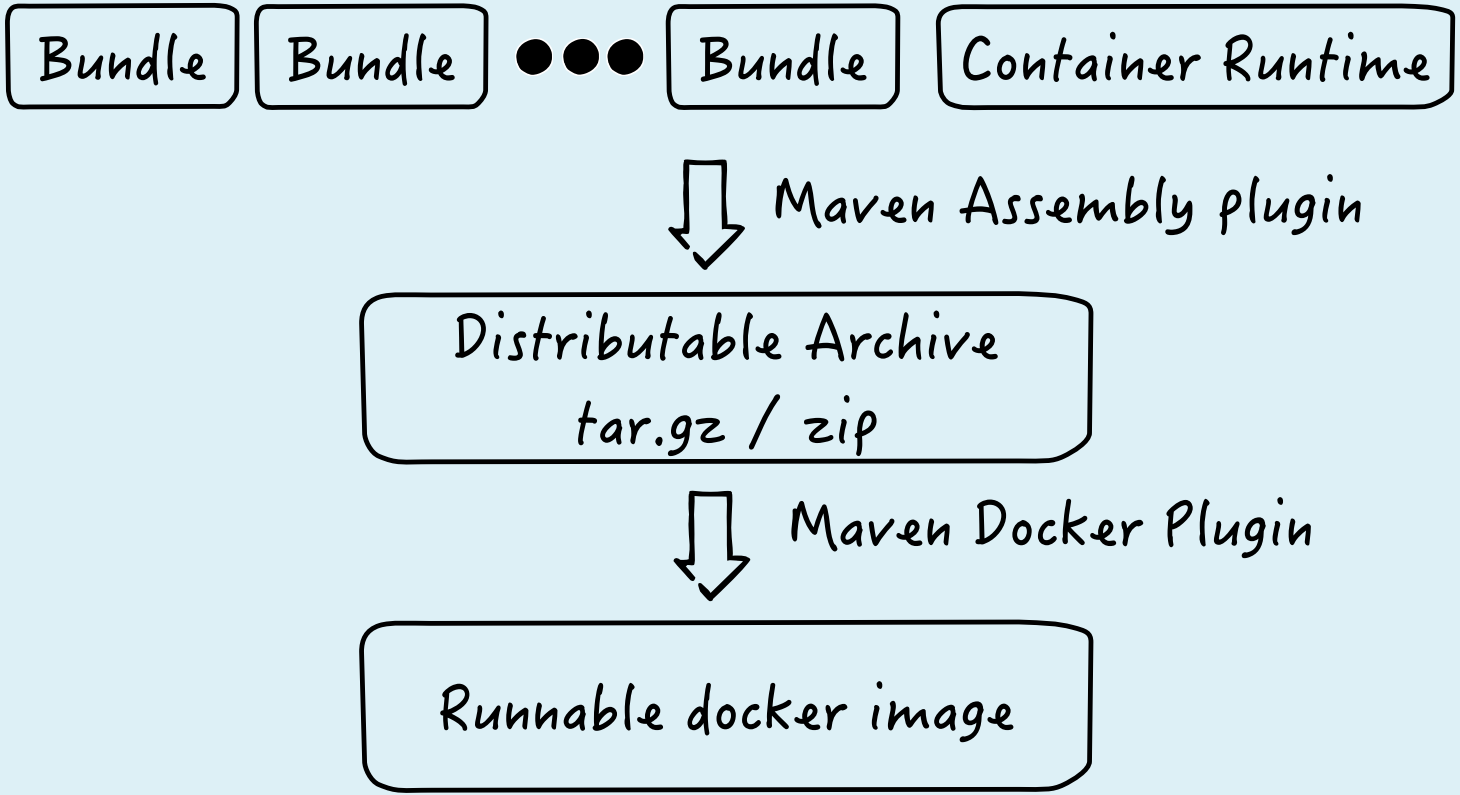
```
FROM atooni/blended-base:latest
ADD blended-karaf-demo-${project.version}-nojre.tar.gz /opt
RUN ln -s /opt/blended-karaf-demo-${project.version} /opt/blended
RUN mkdir -p /opt/blended/data
RUN chown -R blended.blended /opt/blended*
USER blended
ENV JAVA_HOME /opt/java
ENV PATH ${PATH}:${JAVA_HOME}/bin
ENTRYPOINT ["/opt/blended/bin/karaf"]
VOLUME /opt/blended/data
EXPOSE 1883
EXPOSE 8181
EXPOSE 1099
```

https://github.com/woq-blended/blended/blob/master/blended-docker/blended-docker-demo/src/main/docker/demo/Dockerfile

A container under test



Test packaging



Defining the CuT

```
{
    name:   "gs4"
    image:  "^store_sibfile:latest"
    links:  [ "shop_0:shop" ]
    ports: [
      { name: "http", value: "8181" }
    ]
    volumes: [
      {
        host: "data"
        container: "/opt/sibfile/data"
      },
      {
        host: "tmp"
        container: "/tmp"
      }
    ]
  }
```

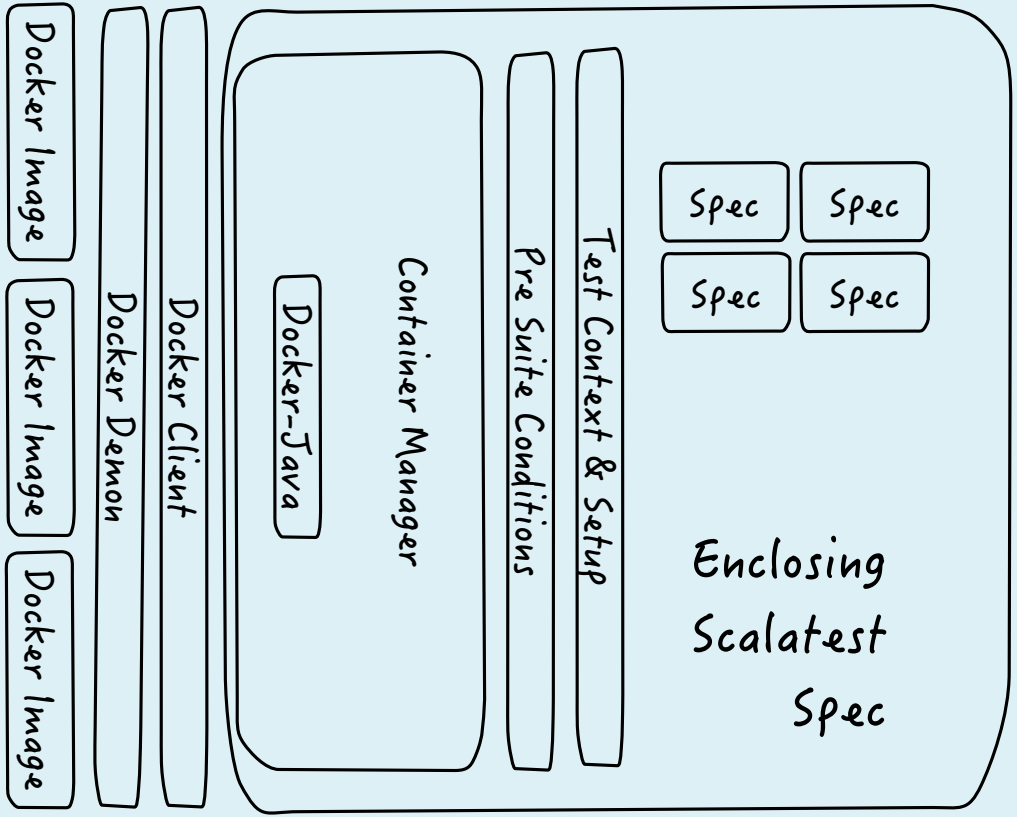
```
op"
tore_shop:latest"
nic_0:sonic" ]

ata"
r : "/opt/shop/data"

tp",   value: "4747" },
ttp",  value: "8777" },
ms",   value: "1883" }
```

```
e:   "sonic"
ge:  "^store_sonic:latest"
s : [
me: "data", value: "2620" },
me: "mgmt", value: "2608" }
```

https://github.com/woq-blended/blended/blob/master/blended-akka-itest/src/test/resources/application.conf

INTEGRATION TEST ELEMENTS

High Level Test Architecture



General tasks to write tests

- Define the containers under test
- Start the CuT and wait until they are ready
- Provide a blackbox test context
- Execute the test suite

The enclosing Spec

```
class BlendedDemoIntegrationSpec extends TestActorSys
  with SpecLike
  with BlendedIntegrationTestSupport {

  beforeSuite()

  override def nestedSuites = IndexedSeq(new BlendedDemoSpec)

  override def preCondition = {
    val t = 60.seconds

    SequentialComposedCondition(
      JMSAvailableCondition(amqConnectionFactory, Some(t)),
      JolokiaAvailableCondition(jmxRest, Some(t), Some("blended"), Some("blended")),
      CamelContextExistsCondition(
        jmxRest, Some("blended"), Some("blended"), "BlendedSample", Some(t)
      )
    )
  }
```

https://github.com/woq-blended/blended/blob/master/blended-akka-itest/src/test/scala/de/woq/blended/akka/itest/BlendedDemoIntegrationSpec.scala#L44

CamelTestSupport

- Create & send test messages
- Wire mock Endpoints
- Frequently used message assertions

https://github.com/woq-blended/blended/blob/master/blended-itestsupport/src/main/scala/de/woq/blended/itestsupport/camel/CamelTestSupport.scala

The example test context

Provide a Camel based test context with all required components for testing

```
private def testContext = {
  val result = new TestCamelContext()
  result.withComponent(
    "jms", JmsComponent.jmsComponent(
      BlendedTestContext(
        BlendedDemoIntegrationSpec.amqConnectionFactory)
          .asInstanceOf[ConnectionFactory]
    )
  )
  result
}
```

https://github.com/woq-blended/blended/blob/master/blended-akka-itest/src/test/scala/de/woq/blended/akka/itest/BlendedDemoSpec.scala#L42

Providing the JMS connection

- We know the JMS port is 1883 and Active MQ is the JMS provider
- Keep information relevant across Specs in an instance of BlendedTestContext

```
private lazy val amqConnectionFactory = {
  val ctInfo  = Await.result(containerInfo("blended_demo_0"), 3.seconds)
  val address = ctInfo.getNetworkSettings.getIpAddress

  val brokerUrl = s"tcp://${address}:1883"

  system.log.info(s"Using AMQ connection url [${brokerUrl}]")

  BlendedTestContext.set(
    BlendedDemoIntegrationSpec.amqConnectionFactory,
    new ActiveMQConnectionFactory(brokerUrl)
  ).asInstanceOf[ConnectionFactory]
}
```

https://github.com/woq-blended/blended/blob/master/blended-akka-itest/src/test/scala/de/woq/blended/akka/itest/BlendedDemoIntegrationSpec.scala#L66

Waiting for the container

Wait until Jolokia, JMS and the Camel Route under test are available

```
override def preCondition = {
  val t = 60.seconds

  SequentialComposedCondition(
    ParallelComposedCondition(
      JolokiaAvailableCondition(jmxRest, Some(t), Some("blended"), Some("blended")),
      JMSAvailableCondition(amqConnectionFactory, Some(t))
    ),
    CamelContextExistsCondition(
      jmxRest, Some("blended"), Some("blended"), "BlendedSample", Some(t)
    )
  )
}
```

https://github.com/woq-blended/blended/blob/master/blended-akka-itest/src/test/scala/de/woq/blended/akka/itest/BlendedDemoIntegrationSpec.scala#L47

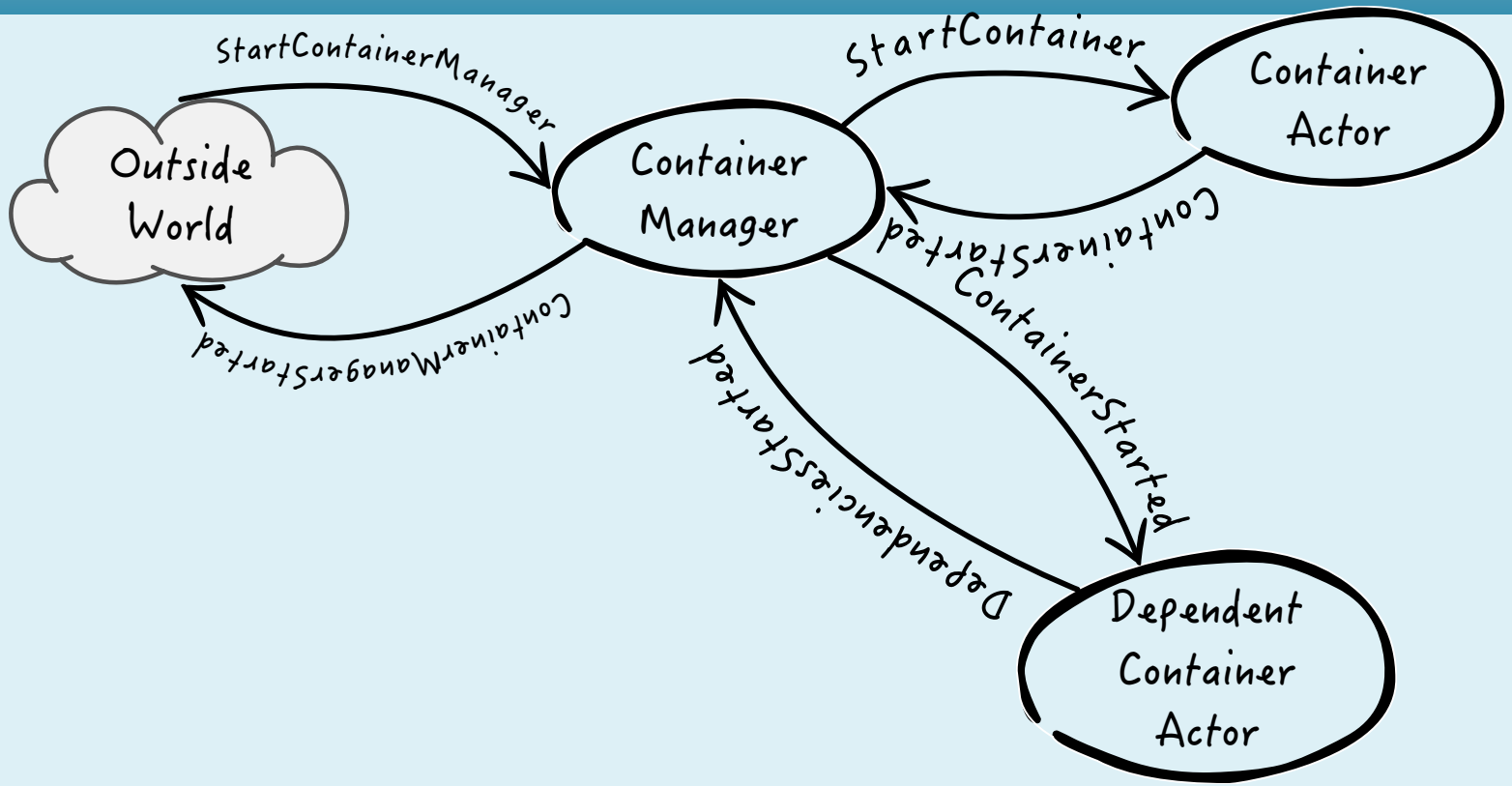
Finally, the spec

```
"The demo container" should {

  "Define the sample Camel Route from SampleIn to SampleOut" in {

    implicit val camelContext = testContext

    withTestContext { ctxt =>
      ctxt.withMock("sampleOut", "jms:queue:SampleOut")
      ctxt.start()

      val mock = ctxt.mockEndpoint("sampleOut")
      mock.setExpectedMessageCount(1)
      mock.expectedBodyReceived().constant("Hello Blended!")

      ctxt.sendTestMessage("Hello Blended!", "jms:queue:SampleIn")
      mock.assertIsSatisfied(2000l)

      None
    }
  }
}
```

https://github.com/woq-blended/blended/blob/master/blended-akka-itest/src/test/scala/de/woq/blended/akka/itest/BlendedDemoSpec.scala#L54

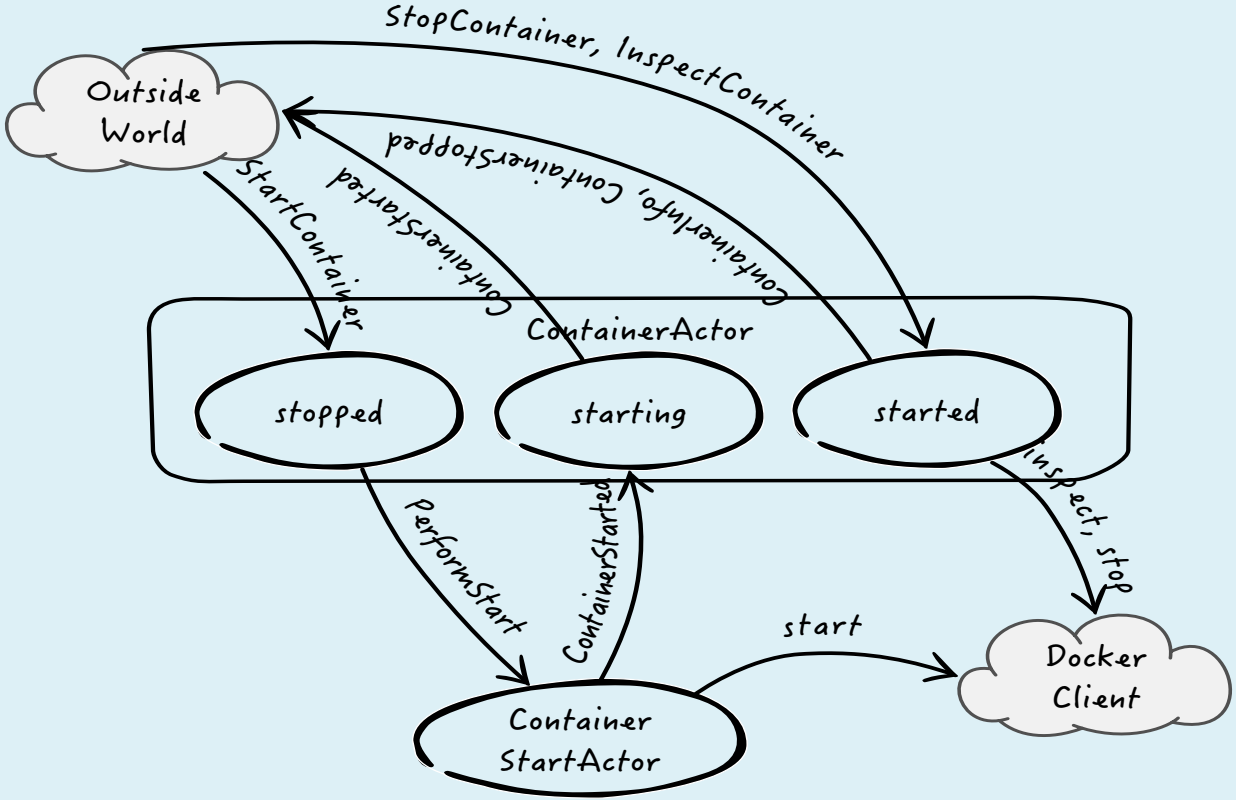
MAIN TEST FRAMEWORK ELEMENTS

An Akka based Container Mgr



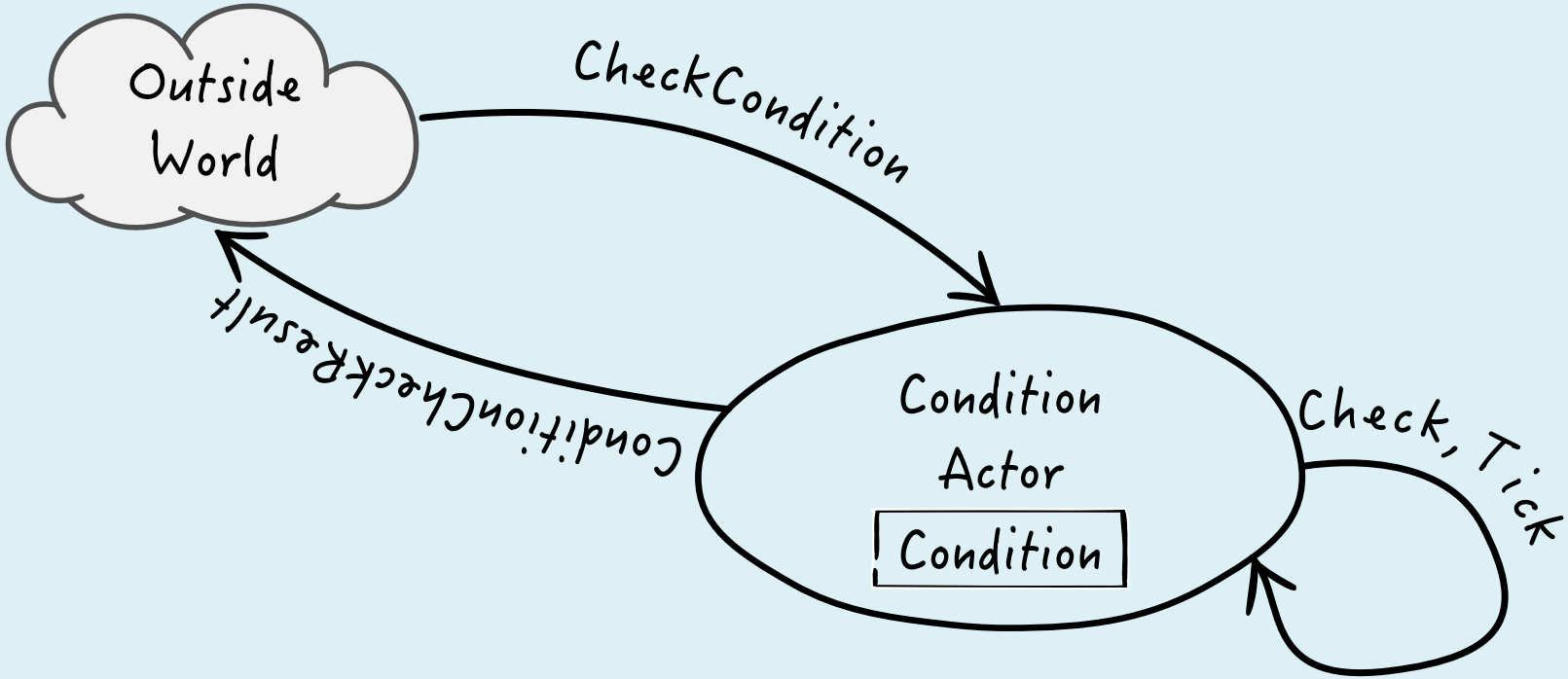
https://github.com/woq-blended/blended/blob/master/blended-itestsupport/src/main/scala/de/woq/blended/itestsupport/docker/ContainerManager.scala

Container Actor



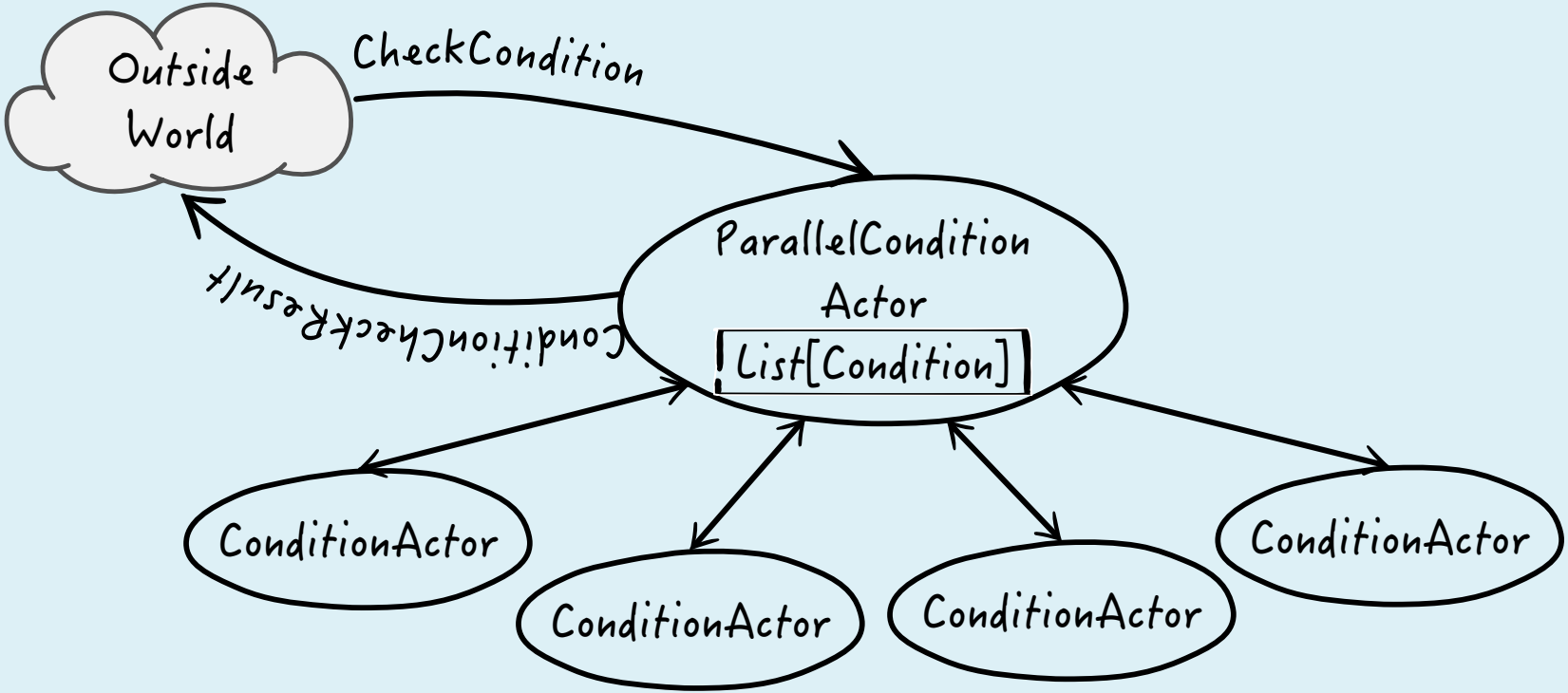
https://github.com/woq-blended/blended/blob/master/blended-itestsupport/src/main/scala/de/woq/blended/itestsupport/docker/ContainerActor.scala#L17

Managing Conditions



https://github.com/woq-blended/blended/blob/master/blended-itestsupport/src/main/scala/de/woq/blended/itestsupport/condition/ConditionActor.scala#L30

Composed Conditions



https://github.com/woq-blended/blended/blob/master/blended-itestsupport/src/main/scala/de/woq/blended/itestsupport/condition/ParallelConditionActor.scala

https://github.com/woq-blended/blended/blob/master/blended-itestsupport/src/main/scala/de/woq/blended/itestsupport/condition/SequentialConditionActor.scala

More test support

- Akka based JMS producer/consumer for throughput measurements (Early stage)
- Akka based Jolokia client to Query the container's JMX resources via REST

https://github.com/woq-blended/blended/blob/master/blended-itestsupport/src/main/scala/de/woq/blended/itestsupport/camel/CamelTestSupport.scala

ONLINE EXAMPLE

Example sources / build

- Container packaging
- Docker definition
- Integration test framework
- Jolokia support
- Example Test
- Jenkins build of Example Test

INTEGRATION TEST GUIDELINES

Integration Test Guidelines

- Concentrate on Blackbox Tests
  - Or Don't modify the CuT to test it …
- Identify the Communication endpoints
  - Aim for readable Specs and discuss with end users
- Specify Green and Red Path Tests
- Try to capture load scenarios
  - Hard to define due to machine dependencies
- Test fixtures
  - Try to abstract test fixtures into abstract classes

CONCLUSION

More to come…

Working

- Container Management
- Test Framework with easy hooks to CuT
- Implicit Test Context for Blackbox testing
- Syntactic sugar and support for Camel based black-box testing

Wishlist / Issues

- More syntactic sugar for Conditions
- Kill Containers throughout test for failover testing
- Leverage a hosting env. like Openshift to execute Containers under test
- Publish Test Framework as independent project

Resources

- GitHub Project Blended
  - Travis Build – Waffle Board – Codacy Code metrics - Project Page
- Way of Quality Homepage
- Scala – Akka – Docker – ScalaTest
- Apache Camel - for writing test routes
- Apache Karaf - The containers under test in the samples
- Apache ActiveMQ - The JMS layer used in the samples
- Hawtio – The management console of the test containers
- Jolokia – A JMX to REST bridge

Stay in contact !

- andreas@wayofquality.de
- http://de.linkedin.com/pub/andreas-gies/0/594/aa5
- https://www.facebook.com/andreas.gies.5
- @andreasgies, #WOQ-Blended on Twitter

- Enjoy a course or a Yoga & coding breakout by the sea

http://www.castillosanrafael.com