CamelOne 2013

June 10-11 2013

Boston, MA

Advanced integration testing with Fuse Fabric

Ioannis Canellos

about me

Principal Software Engineer @ Red Hat

Open source contributor

- Apache Karaf
- Apache Camel
- Apache ServiceMix
- Apache Jclouds
- Fuse Fabric

What are integration tests anyway?

"a phase in testing where individual modules are combined & tested as a group"

overview

Fabric is not:

"yet another testing framework"

Fabric is used for:

Creating
Provisioning
Managing "*Distributed Containers*"

Fabric focuses:

Karaf based containers
But can also support external clients

agenda

Why bother with integration testing?

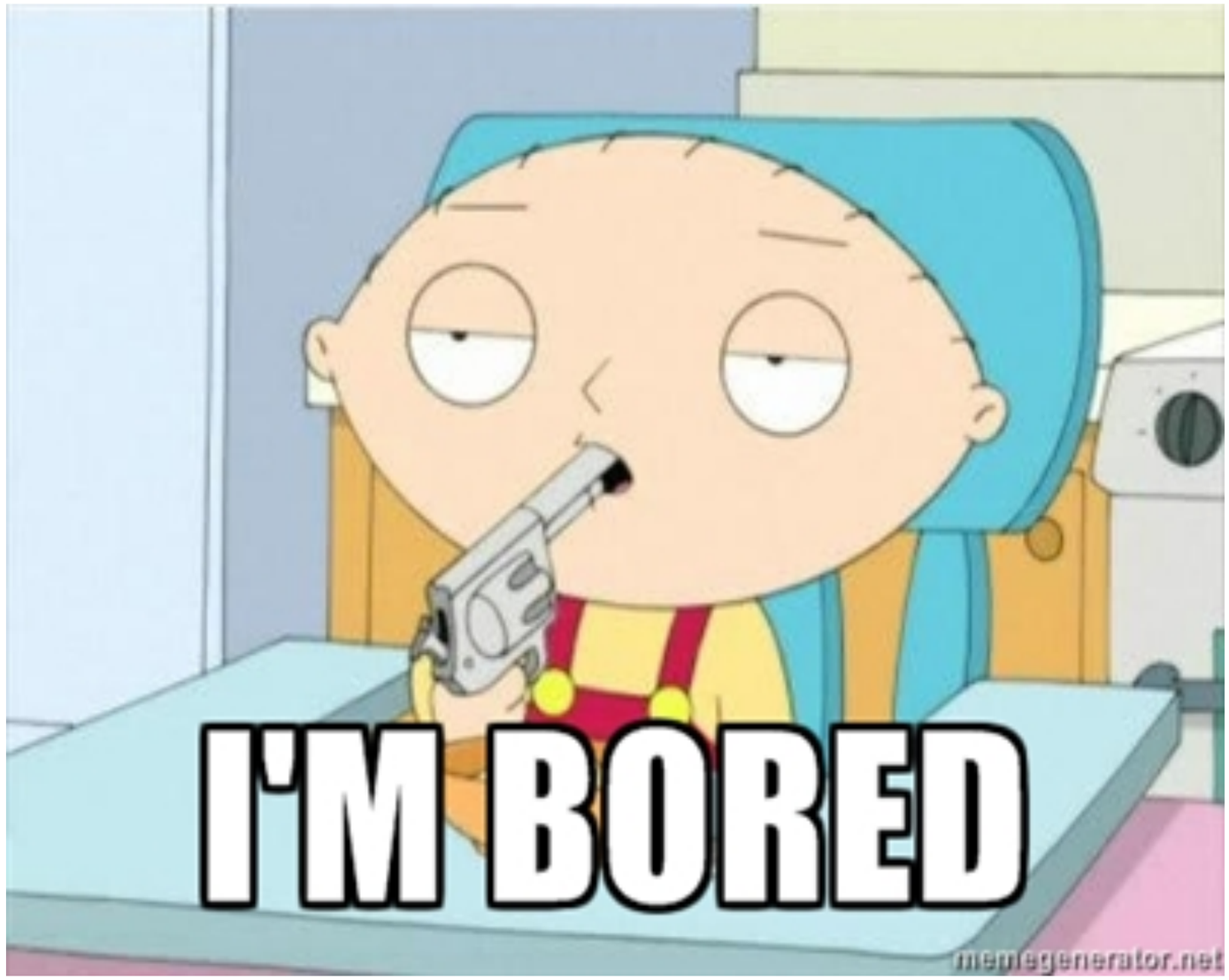
Tools of the trade

Pax-Exam

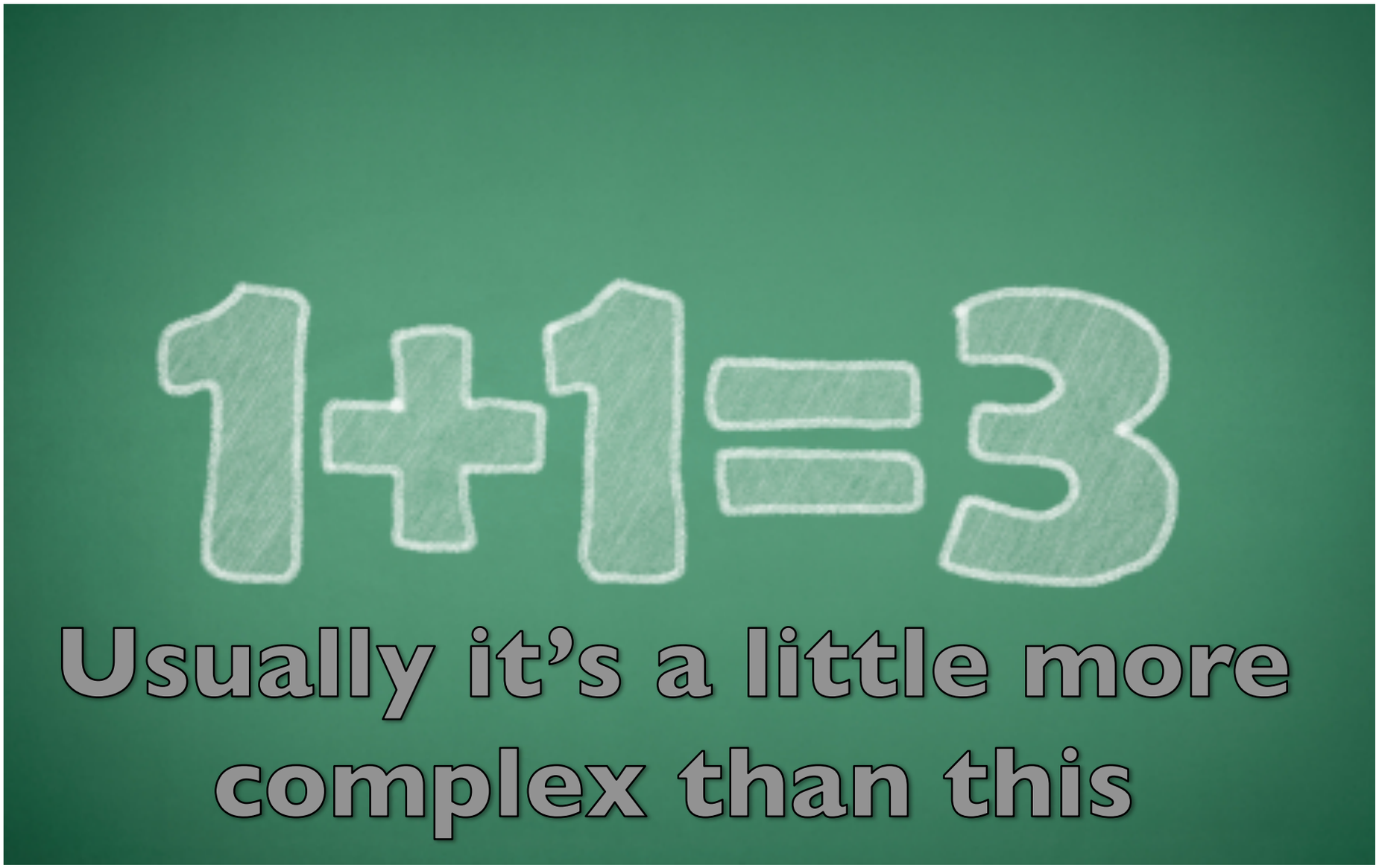
Using Fuse Fabric inside tests

CamelOne

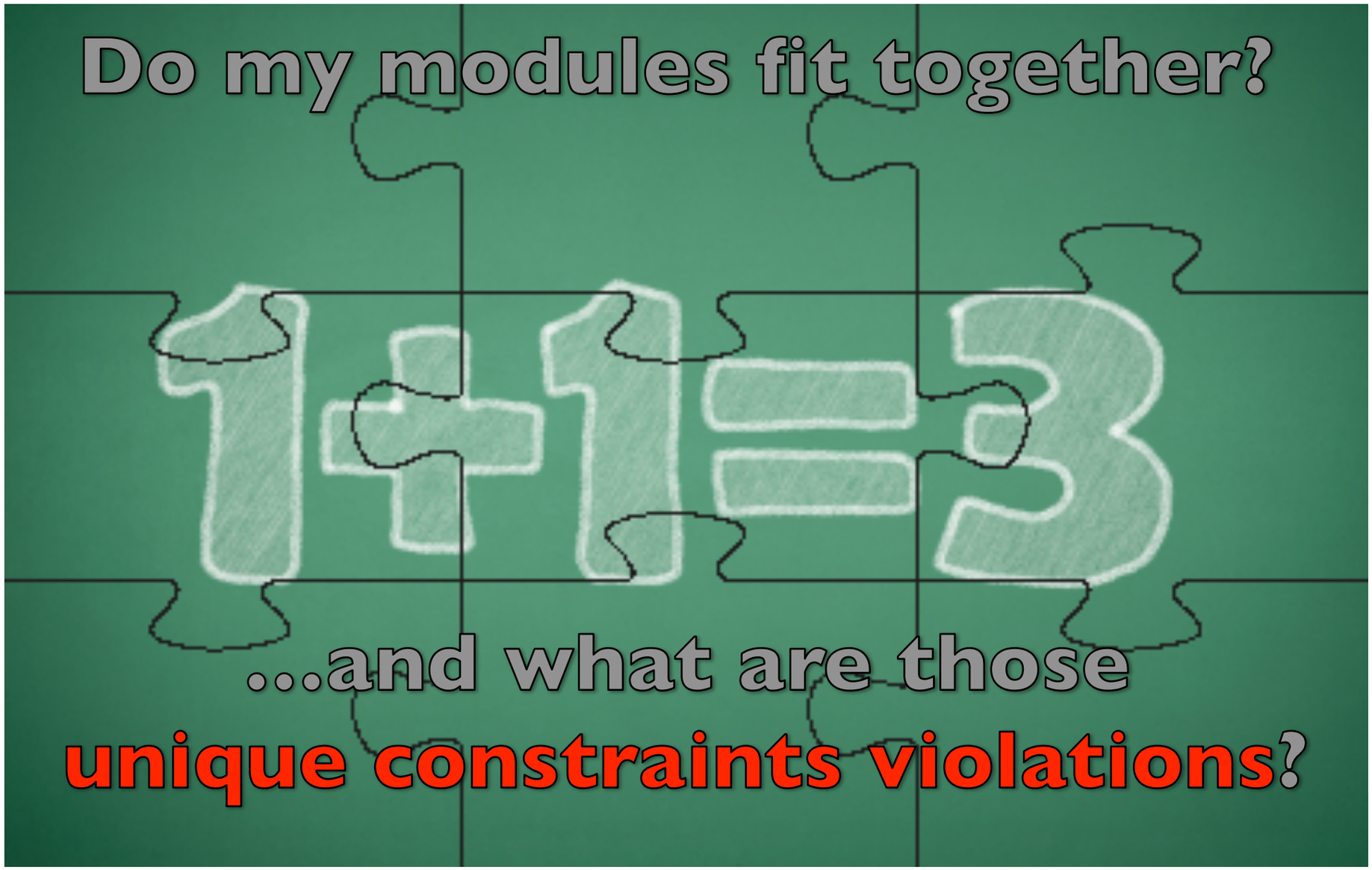
why bother?



why bother?



why bother?

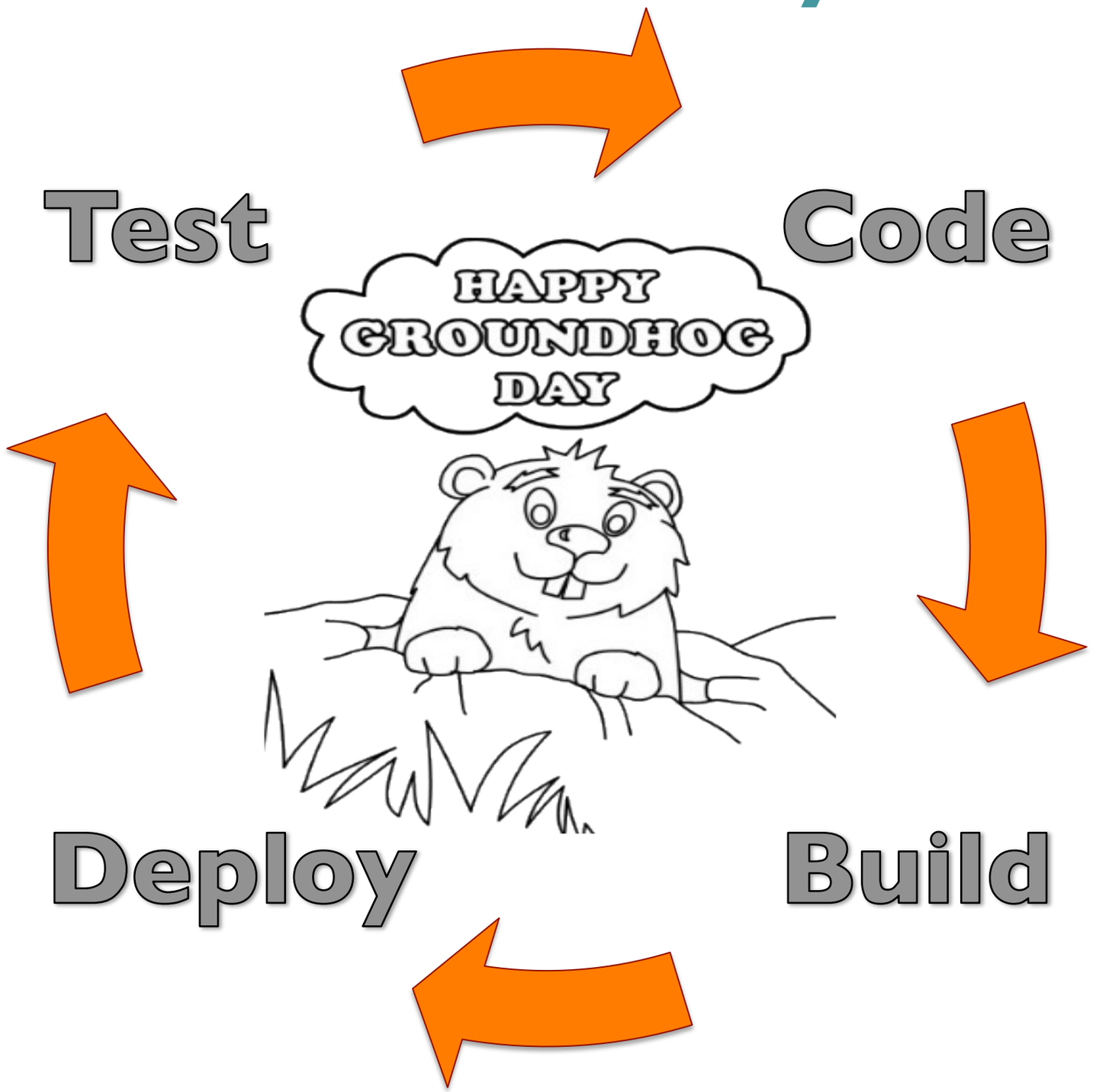

Do my modules fit together?
1+1=3
...and what are those
unique constraints violations?

why bother?

Distributed computing challenges

Services are not always collocated

why bother?



Now what ?

OSGi testing tools

Container agnostic

Rich Tooling

Mostly JEE

Container agnostic

Mostly OSGi

Excellent Karaf Support

Pax-Exam

An OPS4J project

ASL 2.0 license

Current version: 3.0.3

Focus on : 2.6.0

Pax-Exam: “Concepts”

The container

The Probe

Pax-Exam: “Container”

The application container

Location
Configuration
Deployments

Pax-Exam: “Container”

```
@Configuration
public Option[] config() {
    return new Option[]{
            karafDistributionConfiguration().frameworkUrl(
                    maven()
```

Location

```
                    .groupId("org.apache.karaf")
                    .artifactId("apache-karaf")
                    .version("2.3.1").type("zip"))
                    .karafVersion("2.3.1")
                    .name("Apache Karaf"),
```

Configuration

```
            logLevel(LogLevelOption.LogLevel.WARN)
```

Deployments

```
            mavenBundle("my.super.cool", "dependency", "1.0.0"),
            )
    };
}
```

Pax-Exam: “Probe”

The test artifact

Test Classes
Test Resources

Pax-Exam: “Probe”

```
@ProbeBuilder
public TestProbeBuilder builder(TestProbeBuilder probe) {
    String dynamicImports = "*,org.apache.felix.service.*;status=provisional";
    probe.setHeader(DYNAMICIMPORT_PACKAGE, dynamicImports);
    return probe;
}
```

Pax-Exam: “The test”

Typical junit test
@Inject support
Bundled inside the probe
Probe is deployed to the container

Pax-Exam: “The test”

```
@Inject
HelloWorldService helloWorld;

@Test
public void testService() {
    assertEquals("Hello World", helloWorld.greet());
}
```

Things to remember

Maven lifecycle

Install phase is after the test phase

Test dependencies

The "probe" is a bundle too

Ensure runtime dependencies are part of the container configuration

Isolation

The test methods runs in an other jvm

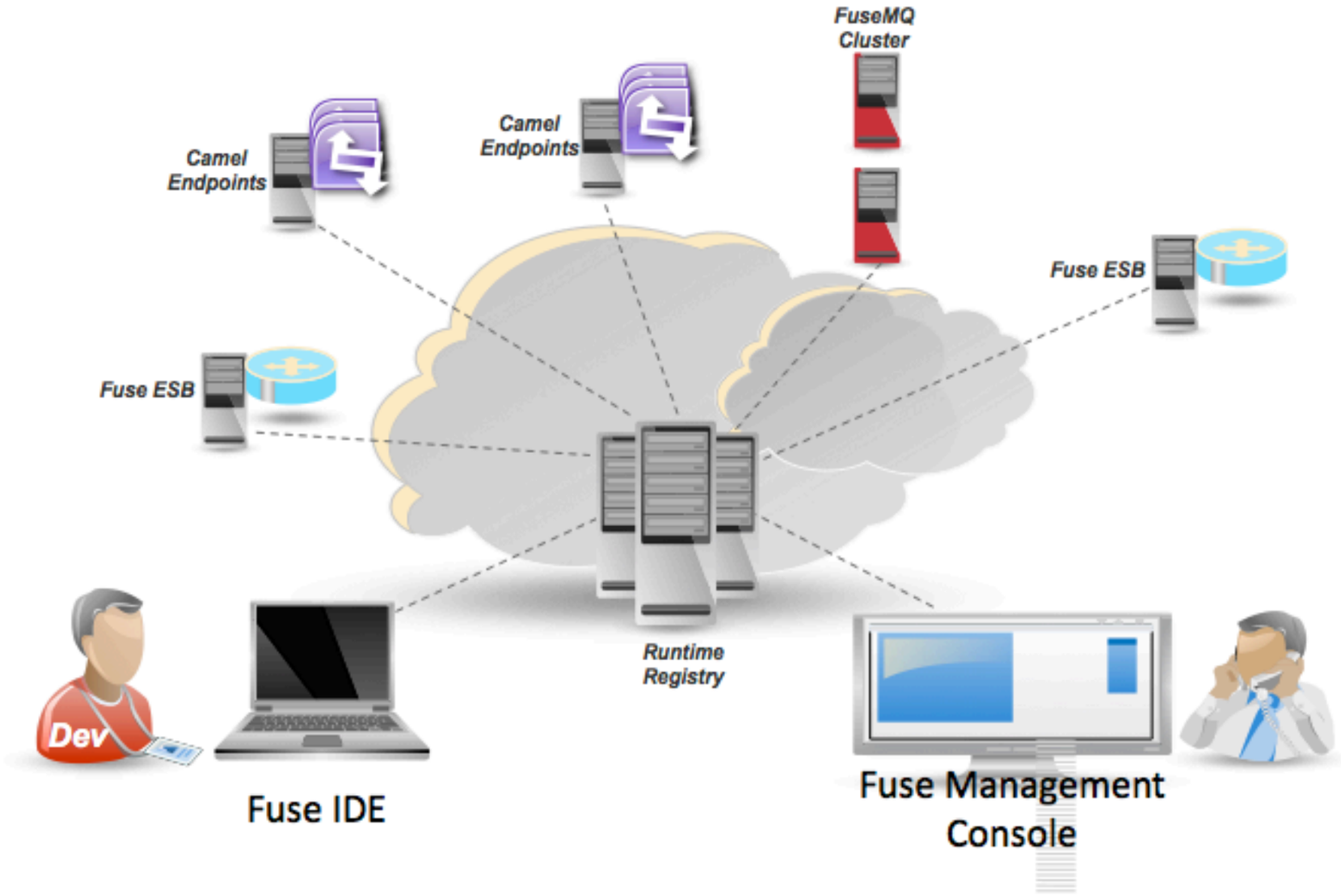
Apache Karaf shell

```
@Test
public void testService() {
   String response = executeCommand("camel:route-list");
   assertTrue(response.contains("awesome-route-1");
}
```

Caution: Ansi escaped response!

```
@Test
public void testService() {
   String response = new AnsiString(
      executeCommand("camel:route-list");
   ).getPlain().toString();
   assertTrue(response.contains("awesome-route-1");
}
```

Fuse Fabric


FuseMQ Cluster
Camel Endpoints
Camel Endpoints
Fuse ESB
Fuse ESB
Runtime Registry
Dev
Fuse IDE
Fuse Management Console

Fuse Fabric

A distributed service

- **configuration**
- **management**
- **provisioning**
- **coordination**

Mostly targets Karaf based Runtimes

Integrates with Camel, ActiveMQ, CXF

Fuse Fabric "registry"

Backed by Apache Zookeeper

- **configuration data**
- **runtime data**

Configuration

- **"provisioning information"**

Runtime

- **Network addresses & ports**
- **Clustered Services**
- **Running applications**

Fuse Fabric “profiles”

Fabric Profiles

- Provisioning information
- Hierarchical structure
- Versioned “rolling upgrades”

Using Fabric Profiles

- Assign profiles to containers
- Containers provision themselves according to the profile

Fuse Fabric "profiles"

Out of the box profiles

- Karaf
- Camel
- CXF
- MQ
- DOSGi

Example profiles for

- Camel
- CXF
- MQ

Fuse Fabric “agent”

A service that provisions containers

- **Reads the assigned profiles**
- **Resolves “dependencies”**
- **Downloads requirements**
- **Removes obsolete artifacts**
- **Installs new artifacts**

Fuse Fabric “weaving”

Fabric can weave itself

Fabric can install containers

- **locally *aka* “child containers”**
- **via SSH *aka* “ssh containers”**
- **via IAAS *aka* “cloud containers”**

Fuse Fabric

So what's the role of Fabric in testing?

Create multiple distributed containers

Provision remote services

Discover services

Coordinate

FabricTestSupport

A default "container" configuration
A set of convenience methods
Executing Shell commands
Looking up Fabric services
Interacting with containers
and more...

Creating a Fabric

```
@Test
public void testLocalFabricCluster() throws Exception {
    executeCommand("fabric:create -n --clean root");
    //Get all cluster containers.
    Container[] containers = getFabricService().getContainers();

    assertNotNull(containers);
    assertEquals("Expected to find 1 container",1, containers.length);
    assertEquals("Expected to find the root container","root", containers[0].getId());
}
```

Will create a fabric
and "assert" the cluster members

Creating a child container

```
@Test
public void testLocalChildCreation() throws Exception {
   executeCommand("fabric:create -n");
  ContainerBuilder.child(1)
                  .withName("child")
                  .assertProvisioningResult()
                  .build();
}
```

Will create a child container and "assert" the status

Provisioning containers

```
@Test
public void testLocalChildCreation() throws Exception {
    executeCommand("fabric:create -n");
    ContainerBuilder.child(1)
                    .withName("child")
                    .withProfiles("example-camel")
                    .assertProvisioningResult()
                    .build();
}
```

Sets a provisioning profile

Provisioning containers

```
@Test
public void testRemoteCamelRoutes() throws Exception {
    executeCommand("fabric:create -n");
    ContainerBuilder.child(1)
                    .withName("child")
                    .withProfiles("example-camel")
                    .assertProvisioningResult()
                    .build();
     StringBuilder sb = new StringBuilder();
    //Connect to remote container
    sb.append("fabric:connect");
     //Using the default credentials
    sb.append(" -u admin -p admin");
    //The name of the remote container
    sb.append(" child");
    //The remote command
    sb.append(" camel:route-list");
    String r = executeCommand(cmd);
    assertTrue(r.contains("super-cool-route");
}
```

Not only commands

Remote commands is one option ...

Each container registers:

- **JMX URL**
- **HTTP port**
- **All webapp URLs**

You can get info via JMX

You can get info via Rest

You can even run Selenium

Using multiple containers

Why don't I just use the root container?

Services are not always co-located

- **Message Brokers**
- **Camel Routes**
- **Web Services**
- **Web Applications**

Can be spread across multiple hosts!

And we need to test this !!!

Using discovery

Fabric provides discovery for:

ActiveMQ brokers
Camel endpoints
Distributed OSGi services

And you can use it in your tests !

Discovering a broker

Discovery is as simple as using the url:

"discovery:(fabirc:default)"

```
<bean id="jmsConnectionFactory"
      class="org.apache.activemq.ActiveMQConnectionFactory">
    <property name="brokerURL" value="discovery:(fabric:default)" />
    <property name="userName" value="${username}" />
    <property name="password" value="${password}" />
</bean>
```

Using it inside the test

```
@Test
public void testCamelWithBrokerDiscovery() throws Exception {
    executeCommand("fabric:create -n");
    //Create an MQ container
    ContainerBuilder.child(1)
                    .withName("broker")
                    .withProfiles("mq")
                    .assertProvisioningResult()
                    .build();

    //Create a Camel container
    ContainerBuilder.child(2)
                    .withName("camel")
                    .withProfiles("example-camel")
                    .assertProvisioningResult()
                    .build();
    //Add your assertions here....
}
```

ContainterBuilder

Not just “child” containers

Supports “remote” containers

Supports variable number & type

Supports environment configuration

ContainterBuilder via ssh

```
@Test
public void testCreate() throws Exception {
    System.err.println(executeCommand("fabric:create -n"));
    Set<Container> containers = ContainerBuilder.ssh(1)
                                        .withName("cnt")
                                        .assertProvisioningResult()
                                        .build();
}
```

Uses env to retrieve ssh configuration

Can support any number of containers

ContainterBuilder

```
@Test
public void testCreate() throws Exception {
    System.err.println(executeCommand("fabric:create -n"));
    Set<Container> containers = ContainerBuilder.create()
                                        .withName("cnt")
                                        .assertProvisioningResult()
                                        .build();
}
```

Both type and conf are found in env.

Without changes in the source:

Devs can use say 3 “child” containers
Jenkins/Hudson can use 10 “remote”

ContainterBuilder Tips

Don't forget to cleanup

```
@After
public void tearDown() throws InterruptedException {
    ContainerBuilder.destroy();
}
```

Build all containers at once

Assign containers later

```
Set<Container> cnts = ContainerBuilder.create()
                                        .withName("cnt")
                                        .assertProvisioningResult()
                                        .build();
List<Container> cntList = new ArrayList<Container>(cnt);
List<Container> mqCnts = containerList.subList(0, cntList.size() / 2);
List<Container> camelCnts = containerList.subList(cntList.size() / 2, containerList.size());
```

Resources

Fuse Fabric

http://fuse.fusesource.org/fabric/

https://github.com/jboss-fuse/fuse

Pax-Exam

https://ops4j1.jira.com/wiki/display/paxexam/Pax+Exam

Questions ?