



Fuse Integration Services 2.0

A systems integration lecture

Mgr. David Simansky
Quality Engineer, Red Hat Middleware

2017-03-03

FIS 2.0

Introduction

Relation to JBoss Fuse

Karaf

Is OSGi in containers still needed?

Spring Boot

Is Winter really ending soon?

Fabric8 Maven Plugin

Maven workflow for OpenShift development

CI/CD

Connecting the dots/pipelines

FIS 2.0

What are the boundaries?

- OpenShift extension for JBoss Fuse
- Versions aligned to 6.3 release
- S2I Docker images
 - fis-java-openshift:2.0 (Spring Boot / generic uber-jar apps)
 - fis-karaf-openshift:2.0 (Karaf micro distributions)
- Karaf 4 Maven Plugin
- Spring Boot support in Camel 2.18
- Fabric8 Maven Plugin (f-m-p or FMP)

Karaf

Karaf

Is it still viable in cloud?

- Microservice ready
 - Define your own small distro with Karaf 4 Maven plugin
 - Health/Readiness checks
 - Benefit from separated classloader
 - Reload config from ConfigMap
- Container in container overhead (inception)
- There might be other drawbacks
 - Containers are immutable
 - No use for runtime patching / hot deployment
 - No need for service discovery

Karaf

Karaf 4 Maven Plugin

```
// Too long for snippet field
```

```
https://github.com/fabric8-quickstarts/karaf2-camel-rest-sql/blob/master/pom.xml
```

Karaf

Fabric8 Karaf Config Admin

```
amq.usr = ${k8s:secret:${env:ACTIVEMQ_SERVICE_NAME}/username}
amq.pwd = ${k8s:secret:${env:ACTIVEMQ_SERVICE_NAME}/password}
amq.url = tcp://${env+service:ACTIVEMQ_SERVICE_NAME}
```

Spring Boot

Spring Boot

Veni, vidi, spingbooted

- Spring application that you can “just run”
- Opinionated defaults out of the box
 - Ease up development
 - Reduce configuration complexity
 - Focus on business logic not framework options
- No code generation
- No requirement for XML configuration
- Embedded HTTP server
- Autoconfiguration
- Readiness/Health checks

Spring Boot

Autoconfigure

- Attempts to automatically configure your application
- Based on class path analysis
- @Enable*
 - @EnableAutoConfiguration
 - @EnableWebMvc
 - @EnableTransactionManagement
- @Conditional
 - @ConditionOnClass
 - @ConditionalOnMissingBean

Spring Boot & Camel

- Camel version 2.18
- Components starters out of the box
- Autoconfiguration support
- Auto detection of Camel routes

Spring Boot & Camel

HelloWorld

```
@SpringBootApplication
public class Application extends RouteBuilder {

    // must have a main method spring-boot can run
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Override
    public void configure() throws Exception {
        from("timer://foo?period=5000")
            .setBody().constant("Hello World")
            .log(">>> ${body}");
    }
}
```

Spring Boot & Camel

Auto detect routes

```
@Component
public class MyRouter extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from("jms:invoices").to("file:/invoices");
    }
}
```

Spring Boot & Camel

That simple?

```
<dependency>  
  <groupId>org.apache.camel</groupId>  
  <artifactId>camel-spring-boot-starter</artifactId>  
</dependency>
```

- <https://github.com/fabric8-quickstarts/spring-boot-camel-amq>

Fabric8 Maven Plugin

Fabric8 Maven Plugin

<https://maven.fabric8.io/>

- Purpose
 - Build Docker images from source (S2I)
 - Create OpenShift resources DC, BC, services, routes
- Based on opinionated defaults (as usual)
- Ironed-out and simplified workflow
 - “mvn clean install” is enough
- Detect cloud platform K8s vs. OpenShift
- Invoke S2I binary build

Fabric8 Maven Plugin

That simple?

```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>fabric8-maven-plugin</artifactId>
  <version>${fabric8.maven.plugin.version}</version>
  <executions>
    <execution>
      <goals>
        <goal>resource</goal>
        <goal>build</goal>
      </goals>
    </execution>
  </executions>
</plugin>
</plugins>
```

Fabric8 Maven Plugin

Goals

`fabric8:help`

`fabric8:resource`

`fabric8:resource-apply`

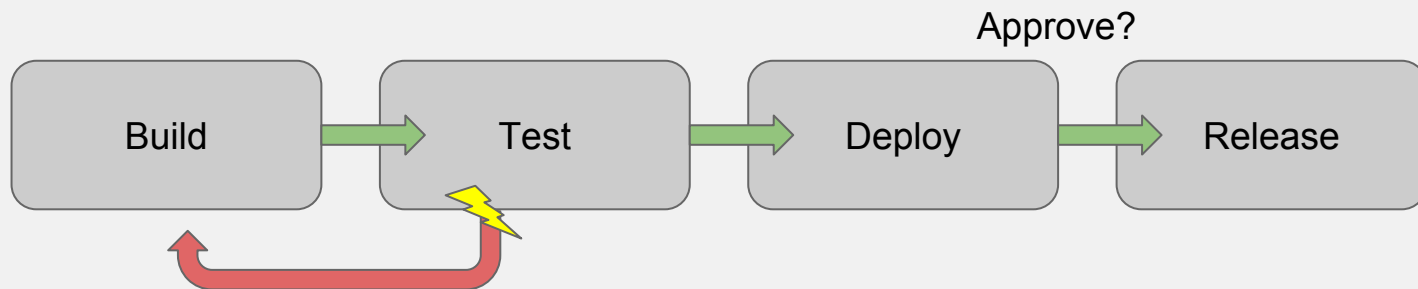
`fabric8:build`

`fabric8:deploy`

CI CD Pipelines

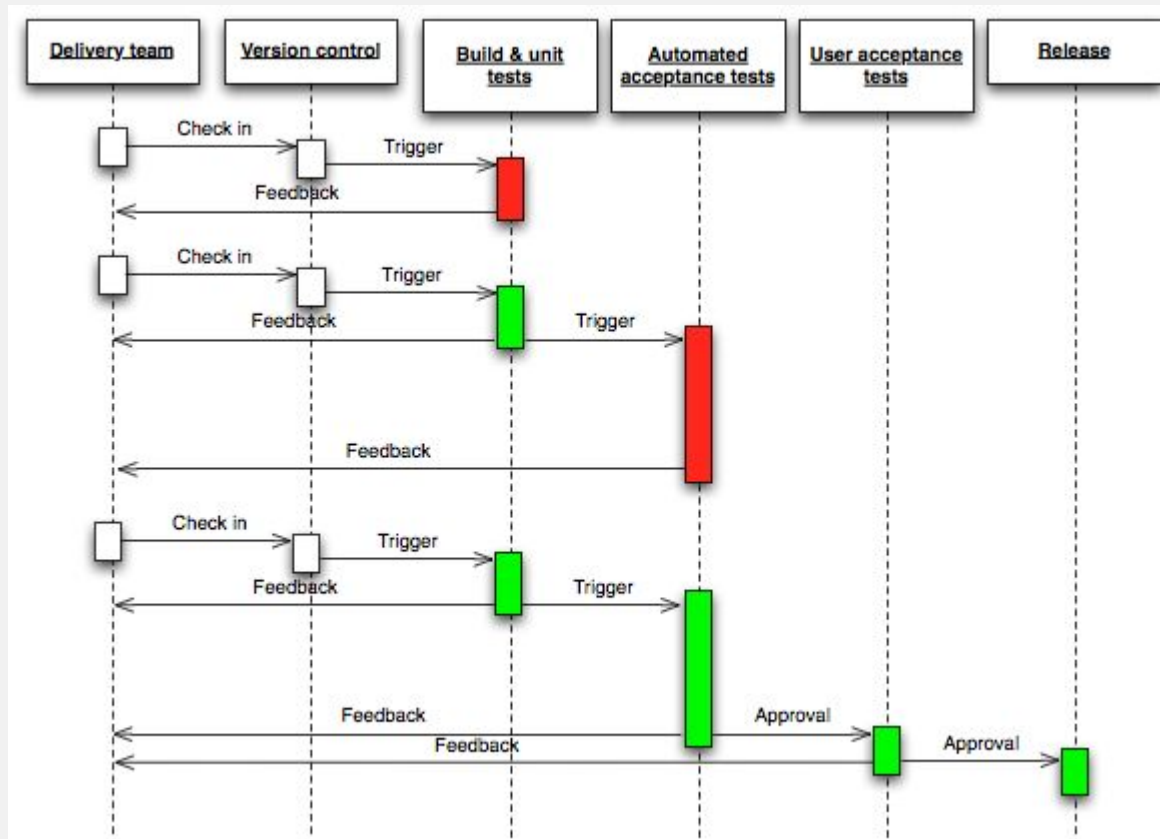
CI CD Pipelines

Motivation



CI CD Pipelines

Motivation



Source: https://en.wikipedia.org/wiki/Continuous_delivery

CI CD Pipelines

Jenkins (Workflow) Pipeline

- Freedom to define the steps of CI job
- Scripted workflow
- Mix shell commands with Jenkins plugins
- Build steps, post-build actions, build environments
- Deploy to cloud, stage, production
- Trigger other jobs or services

CI CD Pipelines

OpenShift

<https://github.com/iss-apps/workshop/tree/master/cicd-pipeline>

CI CD Pipeline

OpenShift console view

The screenshot displays the OpenShift console interface for a project named 'pipeline'. The left sidebar contains navigation options: Projects, Overview, Applications, Builds, Resources, Storage, and Monitoring. The main content area shows the 'Pipelines' section for the 'pipeline' project. It indicates that the pipeline was created 16 hours ago and provides a 'Start Pipeline' button. Below this, the 'Recent Runs' section shows a successful run for 'Build #4' (15 hours ago) with a 'View Log' link. The pipeline execution flow is visualized as a sequence of steps: 'Initialize' (1s), 'Build Maven store-backend' (1m 10s), 'Build Maven store' (16s), and 'Build images' (1m 30s). A 'Verify deployment' step (58s) is also shown below the main sequence. The overall average duration for the pipeline is 4m 25s. At the bottom of the console view, there are links for 'View History' and 'Edit Pipeline'.

CI CD Pipeline

OpenShift console edit view

OPENSIFT ORIGIN

[pipeline](#) » [Pipelines](#) » [cicd](#) » [Edit Pipelines](#)

Edit Build Config cicd — Jenkins Pipeline Build Strategy

Jenkins Pipeline Configuration

Jenkinsfile

```
1 def storeBackend="store-backend"
2 def store="store"
3 node("maven") {
4   stage("Initialize") {
5     sh "curl -O https://raw.githubusercontent.com/iss-apps/workshop/master/settings.xml"
6     stash name: "settings.xml", includes:"settings.xml"
7   }
8   stage("Build Maven ${storeBackend}") {
9     git url: "https://github.com/iss-apps/${storeBackend}.git", branch: "master"
10    unstash name:"settings.xml"
11    sh "mvn -s settings.xml clean package"
12    stash name:"${storeBackend}-jar", includes:"target/${storeBackend}.jar"
13  }
14  stage("Build Maven ${store}") {
15    git url: "https://github.com/iss-apps/${store}.git", branch: "master"
16    unstash name:"settings.xml"
17    sh "mvn -s settings.xml clean package"
18    stash name:"${store}-jar", includes:"target/${store}.jar"
19  }
}
```

No source inputs have been defined for this build configuration.

[Show advanced options](#)

Save

Cancel

CI CD Pipeline

Jenkins console view

The screenshot shows the Jenkins console view for a pipeline named 'pipeline/cicd'. The interface includes a top navigation bar with the Jenkins logo, 'Open Blue Ocean' button, a notification badge with '3', and a search bar. The left sidebar contains navigation options: Back to Dashboard, Status, Changes, Build Now, Delete Pipeline, Configure, Move, Full Stage View, and Pipeline Syntax. The main content area displays the pipeline name, project name, and a 'Recent Changes' section. Below this is the 'Stage View' table, which shows the duration of each stage for the last four builds. The stages are Initialize, Build Maven store-backend, Build Maven store, Build images, and Verify deployment. Build #2 is highlighted in red, indicating it failed during the 'Build images' stage. The 'Build History' section on the left lists the last four builds, all of which are OpenShift builds of the pipeline. At the bottom, there are permalinks for the last build, last stable build, and last successful build, all of which are 14 hours ago.

Pipeline pipeline/cicd

Project name: pipeline-cicd

[Recent Changes](#)

Stage View

	Initialize	Build Maven store-backend	Build Maven store	Build images	Verify deployment
Average stage times:	1s	1min 21s	16s	27s	53s
#4 Mar 02 22:02 No Changes	1s	1min 10s	16s	31s (paused for 59s)	58s
#3 Mar 02 21:46 No Changes	1s	1min 6s	16s	28s (paused for 3min 36s)	48s
#2 Mar 02 21:32 No Changes	1s	1min 49s	18s	21s (paused for 19s) failed	

Build History

find

- #4 Mar 2, 2017 9:02 PM
OpenShift Build pipeline/cicd-4
- #3 Mar 2, 2017 8:46 PM
OpenShift Build pipeline/cicd-3
- #2 Mar 2, 2017 8:32 PM
OpenShift Build pipeline/cicd-2
- #1 Mar 2, 2017 8:28 PM
OpenShift Build pipeline/cicd-1

[RSS for all](#) [RSS for failures](#)

Permalinks

- [Last build \(#4\), 14 hr ago](#)
- [Last stable build \(#4\), 14 hr ago](#)
- [Last successful build \(#4\), 14 hr ago](#)



redhat.

THANK YOU



plus.google.com/+RedHat



facebook.com/redhatinc



linkedin.com/company/red-hat



twitter.com/RedHatNews



youtube.com/user/RedHatVideos

CODE SNIPPET

When referencing code snippets, use the template below. Resize the snippet box to the appropriate size for your text.

```
Type code snippet here
```