Introduction

Table of Contents

This document will help you get started building applications with Spring Boot that run on Kubernetes and various distributions of Kubernetes (ie, OpenShift v3.x).

Make JAR not WAR. We need Moar!

Spring Boot is a great way to build simple Java microservices and has a vibrant ecosystem to help facilitate. Spring Boot and its surrounding ecosystem go to great lengths to help developers get started with microservices including taking some of the pain out of configuration, health checking, boostrapping etc. For example, Spring Boot prefers an "uber-jar" model of packaging a microservice which is executable on its own. This helps reduce the many mistakes that can happen when trying to juggle application servers + WARs/EARs as well as simplifies the classloading model for those apps that can get away with it. When we're deploying lots of microservices, we want to eliminate configuration drift as much as possible and reason about our apps without adding complications between environments.

Building to uber jars does help with this but is not the end of the story. As we further our quest to reduce configuration drift between environments for our apps we must also be aware of this fact: A Java based microservice depends fully on a JVM. The JVM *is* a very important implementation detail of our Java application. As are the dependencies and transitive dependencies of the JVM (libc, etc). A developer that created an app using a particular JVM on, let's say, Windows could potentially behave quite differently on a different JVM running on Linux (in a QA environment, let's say). You'll want a way to capture the complete snapshot of binaries that make up you application and Linux Containers and associated image formats is a great way to do that.

Isn't Docker and Kubernetes complicated?

There's an interesting dynamic between "yah we need this" and "well, it's too complicated". We can hope and pray and ignore it? But really, what if we just made it easier to build your Docker images and deploy to Kubernetes? That's what the fabric8 tooling does for you. It allows you to just use the same developer tools you use today and take advantage of building and deploying cloud-native applications on Kubernetes.

How easy is it?

Lots of Java developers are used to using application servers, creating deployment units (jars/wars), deploying them and running them in app servers using Maven. Then they can use Maven from the command line or easily inside their IDE to do most of the work.

So we figured, why not make Kubernetes look and feel like an application server to a Java developer? So you build and deploy your application from maven like you would with other maven plugins like spring-boot, tomcat, jetty, wildfly, karaf et al. Then you can get started quickly by just treating Kubernetes as a kind of application server.

Kubernetes is actually way more awesome than an application server; its more like an application cloud as:

- kubernetes can keep running multiple instances of each of your apps including automatic restarts on software or hardware failures
- automatic load balancing when invoking your apps
- each app instance is isolated as a separate process so its much easier to monitor metrics and logs

Keep reading to get started with Spring Boot, or checkout how we've made it easy for tomcat, jetty, wildfly, karaf, standalone, etc too!

Do I need to do anything different for my Spring Boot app?

No! Just create your Spring Boot app (or use an existing one if you've got one) and add a single maven plugin. That's it! Here's how we get started.