

Using funktion on the JVM

# Table of Contents

Examples .....	1
Getting started with Funktion and the JVM .....	1
Using the Fabric8 Microservices Platform .....	1
Create and use your funktion .....	2
How it works .....	5

Funktion is designed so that it can bind any events to any HTTP endpoint or any function source using a scripting language like nodejs, python or ruby. But you can also embed the funktion mechanism inside a JVM process.

To do that you:

- write a simple function in any programming language [like this](#).
- create a [funktion.yml](#) file and associate your function with an [event trigger endpoint URL](#) such as a HTTP URL or email address to listen on, a message queue name or database table etc.
- build and deploy the Java project in the usual way, such as via [Jenkins CI / CD pipeline](#) and your funktion will be deployed to your kubernetes cluster!

## Examples

Check out the following example projects which use a JVM and implement the functions in different JVM based languages:

- [funktion-java-example](#) is an example using a Java funktion triggered by HTTP
- [funktion-groovy-example](#) is an example using a [Groovy](#) funktion triggered by HTTP
- [funktion-kotlin-example](#) is an example using a [Kotlin](#) funktion triggered by HTTP

## Getting started with Funktion and the JVM

You can just fork one of the above examples and use command line tools to build and deploy it to a [Kubernetes](#) or [OpenShift](#) cluster.

However to make it easier to create, build, test, stage, approve, release, manage and iterate on your funktion code from inside your browser we recommend you use the [Fabric8 Microservices Platform](#) with its baked in [Continuous Delivery](#) based on [Jenkins Pipelines](#) together with integrated [Developer Console](#), [Management](#) (centralised logging, metrics, alerts), [ChatOps](#) and [Chaos Monkey](#).

When using the [Fabric8 Microservices Platform](#) you can create a new funktion in a few clicks from the [Create Application](#) button; then the platform takes care of building, testing, staging and approving your releases, rolling upgrades, management and monitoring; you just use your browser via the [Developer Console](#) to create, edit or test your code while funktion, Jenkins and Kubernetes take care of building, packaging, deploying, testing and releasing your project.

## Using the Fabric8 Microservices Platform

First you will need to install the [fabric8 microservices platform](#) on a cluster of [Kubernetes](#) (1.2 or later) or [OpenShift](#) (3.2 or later).

- follow one of the [fabric8 getting started guides](#) to get the [fabric8 microservices platform](#) up and

running on a Kubernetes or OpenShift cluster

- open the [Developer Console](#)
- select your **Team Dashboard** page

## Create and use your funktion

- from inside your **Team Dashboard** page click **Create Application** button then you will be presented with a number of different kinds of microservice to create
- select the **Funktion** icon and type in the name of your microservice and hit **Next**

- select the kind of funktion you wish to create (Java, Groovy, Kotlin, NodeJS etc) then hit **Next**
- you will now be prompted to choose one of the default CD Pipelines to use. For your first funktion we recommend **CanaryReleaseAndStage**
- selecting **Copy pipeline to project** is kinda handy if you want to edit your **Jenkinsfile** from your source code later on

- click **Next** then your app should be built and deployed. Please be patient first time you build a funktion as its going to be downloading a few docker images to do the build and runtime. You're second build should be much faster!
- once the build is complete you should see on the **App Dashboard** page the build pipeline run, the running pods for your funktion in each environment for your CD Pipeline and a link so you can easily navigate to the environment or ReplicaSet/ReplicationController/Pods in kubernetes
- in the screenshot below you can see we're running version **1.0.1** of the app **groovyfunktion** which currently has **1** running pod (those are all clickable links to view the ReplicationController or pods)
- for HTTP based funktions you can invoke the funktion via the open icon in the **Staging** environment (the icon to the right of the green **1** button next to **groovyfunktion-1: 1.0.1**)

## How it works

When you implement your **Funktion** using a JVM based language like Java, Groovy, Kotlin or Scala then your function is packaged up into a [Spring Boot](#) application using [Apache Camel](#) to implement the trigger via the various [endpoint URLs](#).

We've focussed **funktion** on being some simple declarative metadata to describe triggers via URLs and a simple programming model which is the only thing funktion developers should focus on; leaving the implementation free to use different approaches for optimal resource usage.

The creation of the docker images and generation of the kubernetes manifests is all done by the [fabric8-maven-plugin](#) which can work with pure docker on Kubernetes or reuse OpenShift's binary source to image builds. Usually this is hidden from you if you are using the [Continuous Delivery](#) in the [fabric8 microservices platform](#); but if you want to play with funktion purely from the command line, you'll need to [install Java](#) and [install Apache Maven](#).

Underneath the covers a [Kubernetes Deployment](#) is automatically created for your Funktion (or on OpenShift a [DeploymentConfig](#) is used) which takes care of scaling your funktion and performing [rolling updates](#) as you edit your code.