RED HAT®
DEVELOPER
PROGRAM

developers.redhat.com

Who am I ?

kamesh.sampath@hotmail.com

kameshsampath

@kamesh_sampath

Kamesh Sampath

Director of Developer Experience at Red Hat

- Active Open Source Contributor
  - OpenWhisk
  - Eclipse Che
  - fabric8 Platform https://fabric8.io/
- Creator vert.x-maven-plugin → https://vmp.fabric8.io/

Your Java Journey into the Serverless World

Developing Java Actions on Apache OpenWhisk

bit.ly/faas-tutorial

Serverless Defined

Serverless architectures refer to applications that significantly depend on third-party services (knows as **Backend as a Service or "BaaS"**) or on custom code that's run in ephemeral containers (**Function as a Service or "FaaS"**), the best known vendor host of which currently is AWS Lambda. By using these ideas, and by moving much behavior to the front end, such architectures remove the need for the traditional '**always on**' server system sitting behind an application. Depending on the circumstances, such systems can significantly **reduce operational cost** and **complexity** at a cost of vendor dependencies and (at the moment) immaturity of supporting services.

https://martinfowler.com/articles/serverless.html

Good and Bad about Serverless

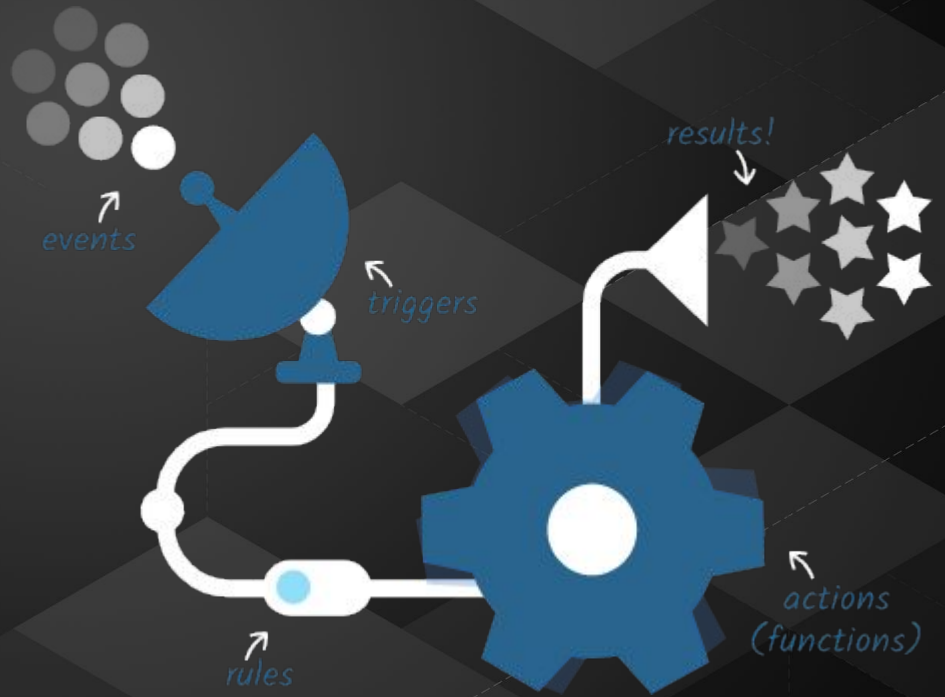
- Automatic Scalability
- Automatic Cost Reduction
- Quicker and Easier Development
- Better Capacity planning
- Delivery speed

- Debugging
- Deployment and Architectural complexity
- Deep learning curve
- Vendor Lock-in
- Monitoring

https://hackernoon.com/serverless-survey-77-delivery-speed-4-dev-workdays-mo-saved-26-aws-monthly-bill-d99174f70663

Apache OpenWhisk

- Open Source incubating under Apache
- A Cloud platform to execute functions written in:
  - JavaScript
  - Swift
  - Java
  - Python
  - PHP
  - Docker
  - Go
- Deployable on
  - Any platform where docker can be run
  - Kubernetes/OpenShift



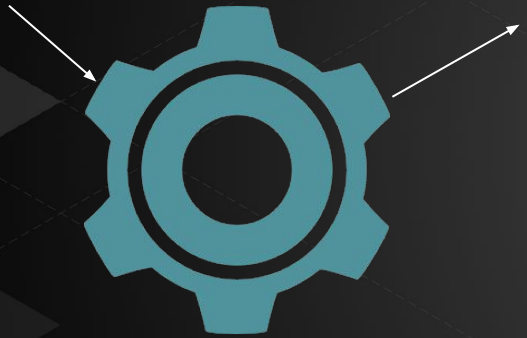
Some theory

- **Actions**
  - A **stateless** code snippet that gets executed
- **Package**
  - A **bundle** of actions and feeds synonymous to Java Packages
- **Feeds**
  - Acts as **Interface** between external **Event Sources** and OpenWhisk **Actions**
- **Triggers**
  - Stimulus of external Events via Feeds
- **Rules**
  - Ties one or more Triggers to an Action or vice-versa
- **Activation**
  - Records the output/response of Running/Ran actions

Tooling

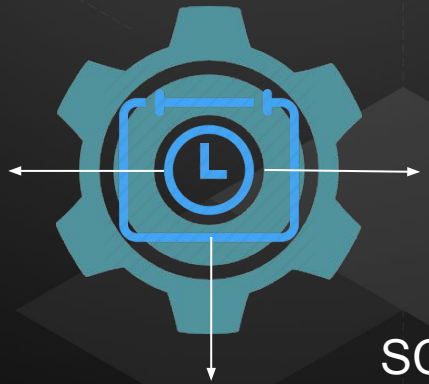
- wsk
  - The core command line interface to work OpenWhisk
- wskdeploy
  - The command line tool to allow deploying OpenWhisk actions, functions etc., using YAML base manifest
- Maven (only for Java)
  - Generate OpenWhisk Java Functions
- Serverless Framework
  - JavaScript based template framework
  - Support for generating OpenWhisk Templated project
  - Ability to deploy functions to OpenWhisk

Invocation Patterns



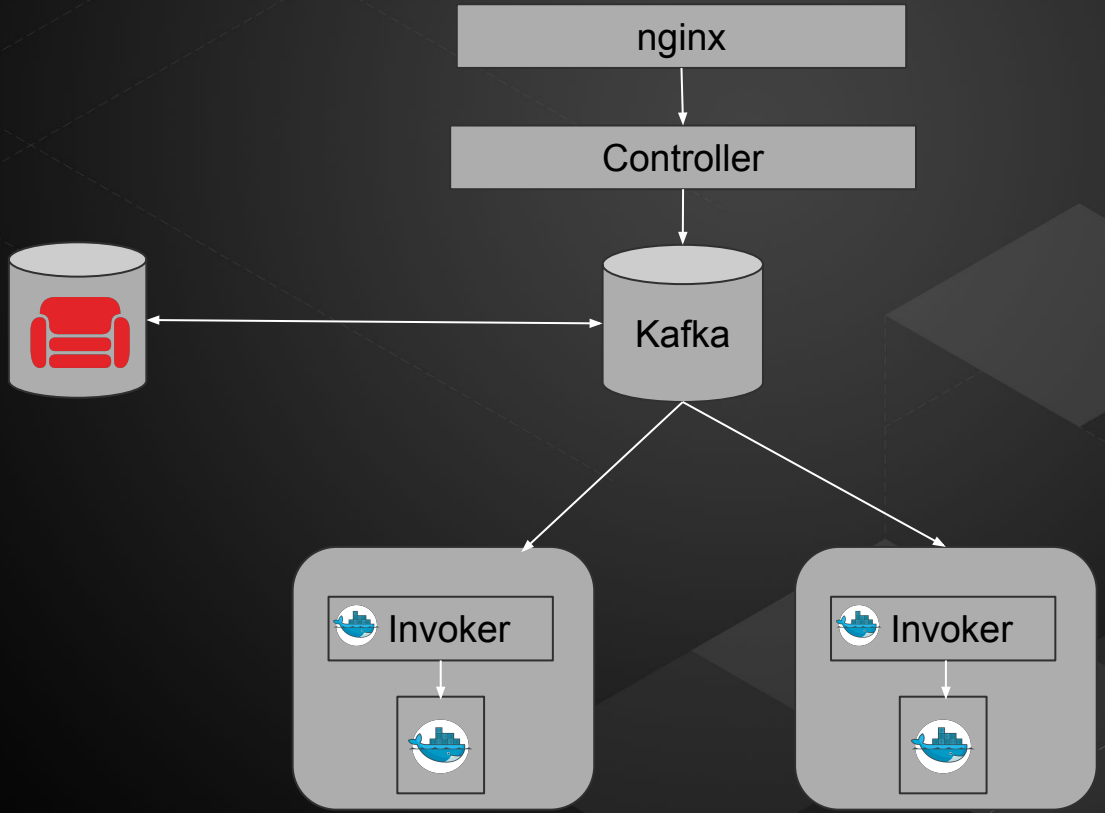
ASYNCHRONOUS

SYNCHRONOUS



SCHEDULED

Apache OpenWhisk - How does it work ?



Get the feel

Java Actions

Plain Old Function (POF)

- Pros
  - Java POJO
  - Simple and straightforward to understand
- Cons
  - Not portable across serverless providers
  - Dependencies need to bundled as an uberjar
  - Uses third-party libs e.g Google Gson for OpenWhisk Java Actions

Spring Cloud Functions (SCF)

- Pros
  - Uses standard and popular Spring Programming model
  - Standard Spring Boot Application Signature
    - No extension of specific provider e.g. AWS Lambda
  - Uses Java Function Model
    - Functions (Event Sinks)
      - Suppliers
      - Consumer
  - Adapters for major serverless providers OpenWhisk, AWS, Azure
    - Increases portability
- Cons
  - Heavy and Fat for serverless functional app
  - Startup time is longer owing to start the standard Spring Boot Application
  - Uses thin-jar for basic application and dependency resolution happens at runtime
  - Customization is not possible as some serverless providers might not allow custom docker images

WebAction

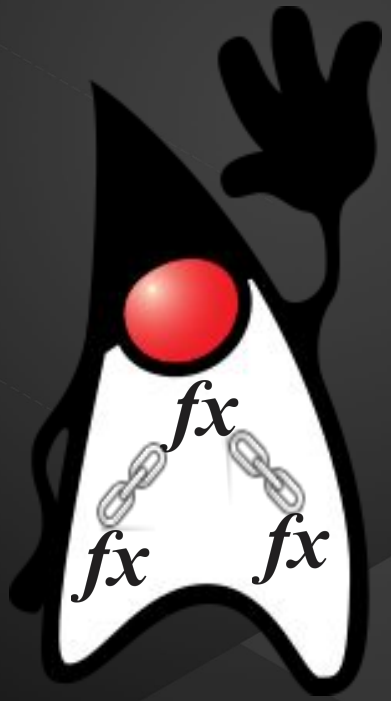
- Any action could be made web action
  - via annotation
- Web Action has URLS `wsk -i action get myaction --url`
- Any HTTP Verb could be used
- Automatic Content type detection via prefixing the url
  - E.g. https://some-url.**json**
- Can be invoked via URL
  - Query Parameters will be Action Parameter JSON
  - Request Body will be Action Parameter JSON
- If Content-Type header is ignored then the request body will be base64 encoded string
- It's NOT SYNCHRONOUS
- Some resources require authentication

Demo - Web Action

Action Chaining

- Actions can be chained
- Pre-defined chain of actions
  - **Sequences**
- Action Chaining could also be determined at runtime
  - **Conductors**
- Only first function in Sequence can accept parameter
  - The **output** of previous step(Action) is **input** to next

Demo - Action Chaining

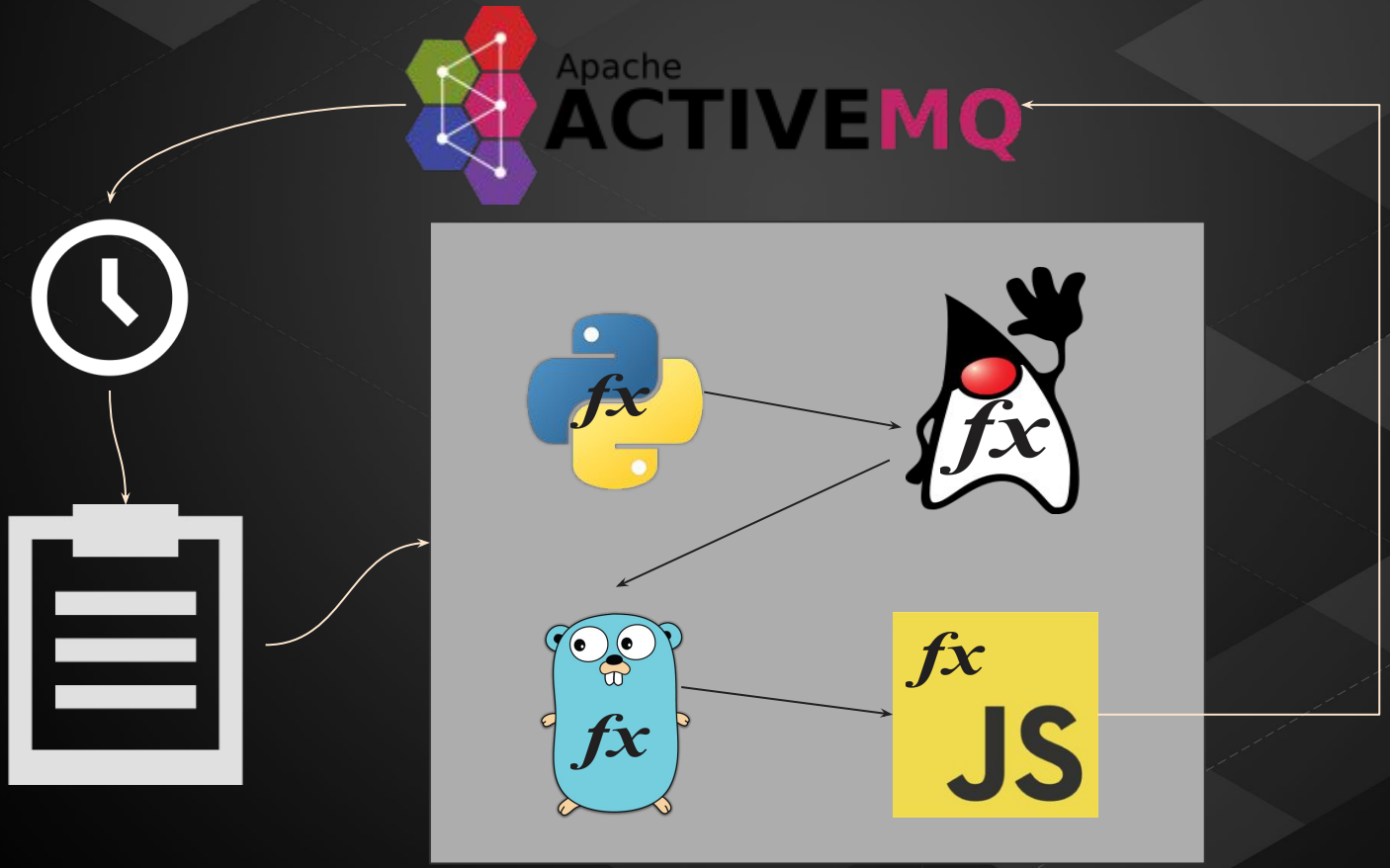


Event Driven Capabilities

- OpenWhisk functions can respond to Events
- Events could be
  - Another function
  - External event source called as Feed Provider
- Event Action is OpenWhisk Function that's invoked for a **Trigger**
- Event Actions are invoked via Triggers
  - **Rules** tie Actions to Triggers
- Trigger can be registered to Event Provider via **Feeds** (Stream of Events)
  - **Feed** can be:
    - Polling
    - Webhook
    - Persistent Connection

Feed the Trigger to Rule your Action

Demo - Feeds and Events



Thank you

bit.ly/faas-tutorial