Serverless - Hype or Reality

James Allen
Senior Solution Architect, FSI
ja@redhat.com

Disclaimer

The content set forth herein is Red Hat confidential information and does not constitute in any way a binding or legal agreement or impose any legal obligation or duty on Red Hat.

This information is provided for discussion purposes only and is subject to change for any or no reason.

Agenda

The Myth - What is Serverless
Putting the puzzle together
A brief journey through history
The tools
Example Use Cases

"Serverless data center"

"Serverless data center"

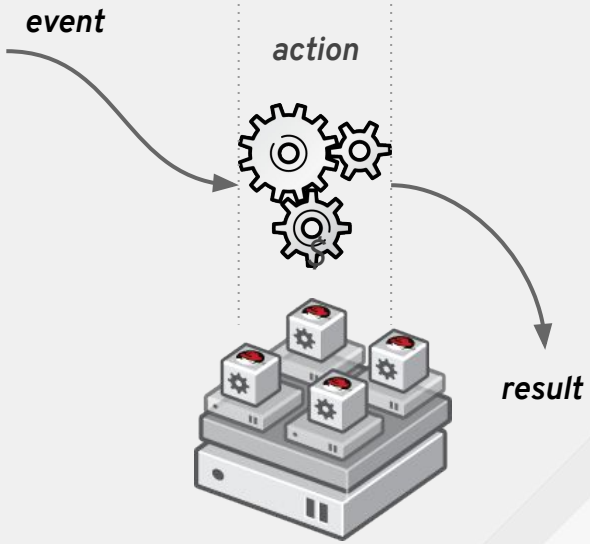
Yes, there are servers.

- Physical boxes running operating systems, VMs and containers.

...but **the platform** takes care of provisioning, scaling, dispatching, monitoring all of those.

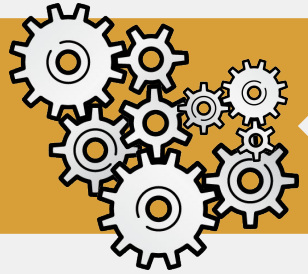
Serverless defined

"refers to the concept of building and running applications that do not require server management. It describes a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment"



Architectural evolution

Service



> Autonomous
> Loosely-coupled

Microservice

> Single Purpose
> Independently Scalable
> Automated

Function

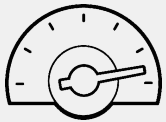
> Single Action
> Ephemeral

Control & High complexity

Productivity & Low Control

PORTABILITY

Why do we need serverless ?



***Agility** of the cloud on any environment*

- *On-premise*
- *Multi-cloud*
- *Hybrid*

*Enable **event driven** cloud-native applications but also **integrate** with classic applications*

*Focus on business differentiation, abstract & delegate infrastructure to **platform** & **services***

*Consistent and scalable **operations** across multiple applications*

Resource optimization & cost savings

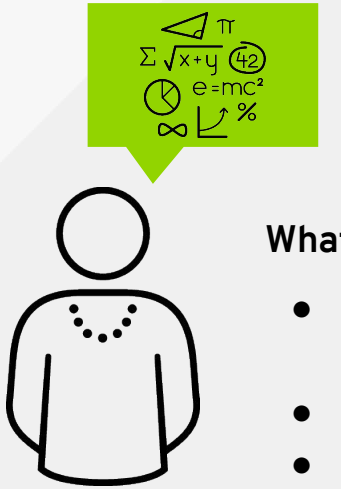
Scenario

- Knows about ***Microservices***
- Understands ***containers*** and ***Dockerfiles***
- Knows that ***Kubernetes*** "helps"
- Likes JavaScript and Python.
- Wants to start small
- Build an **evolutionary architecture**
- Incremental changes
- Exploratory development

- A developer with a new assignment: **Build a new Product Catalog**
- Works in Retail

Store new Product(s) submitted by a merchant

Scenario



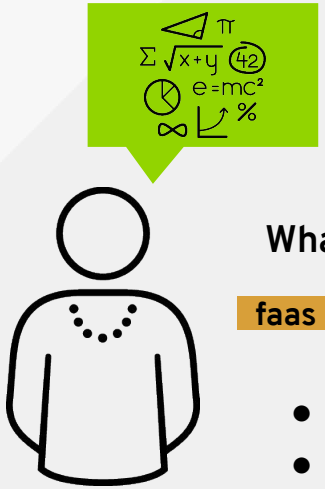
Store new Product(s) submitted by a merchant

What she doesn't know from day 1

- How many users ?
  (concurrently or over a day, week, month)
- How much storage for the whole system ?
- What's the size of the whole system ?
- What database to use ?
- How to build an evolutionary architecture ?
- How/when to resize the pictures ?
- How to send user notifications
  - Picture is missing
  - New product has been added

- What about security ?
  - Encryption ?
- What about day 2 ?
  - Monitoring
  - Patching
  - Error handling
  - Scale
- **Cost!**

Scenario



Store new Product(s) submitted by a merchant

What she doesn't know from day 1

faas How many users ?
(concurrently or over a day, week, month)
- How much storage for the whole system ?
- What's the size of the whole system ?
- What database to use ?
- How to build an evolutionary architecture ?

faas How/when to resize the pictures ?
- How to send user notifications
  - Picture is missing
  - New product has been added
- What about security ?
  - Encryption ?
- What about day 2 ?

faas
  - Monitoring

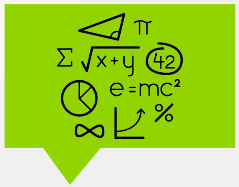
faas
  - Patching

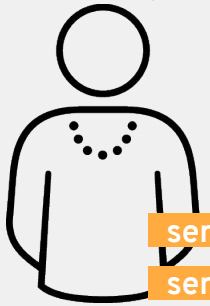
faas
  - Error handling

faas
  - Scale

- **Cost!**

Scenario





Store new Product(s) submitted by a merchant

What she doesn't know from day 1

faas How many users ?
(concurrently or over a day, week, month)

serverless How much storage for the whole system ?

serverless What's the size of the whole system ?

serverless What database to use ?

- How to build an evolutionary architecture ?

faas How/when to resize the pictures ?

serverless How to send user notifications
  - Picture is missing
  - New product has been added

serverless What about security ?
  - Encryption ?
- What about day 2 ?

faas
  - Monitoring

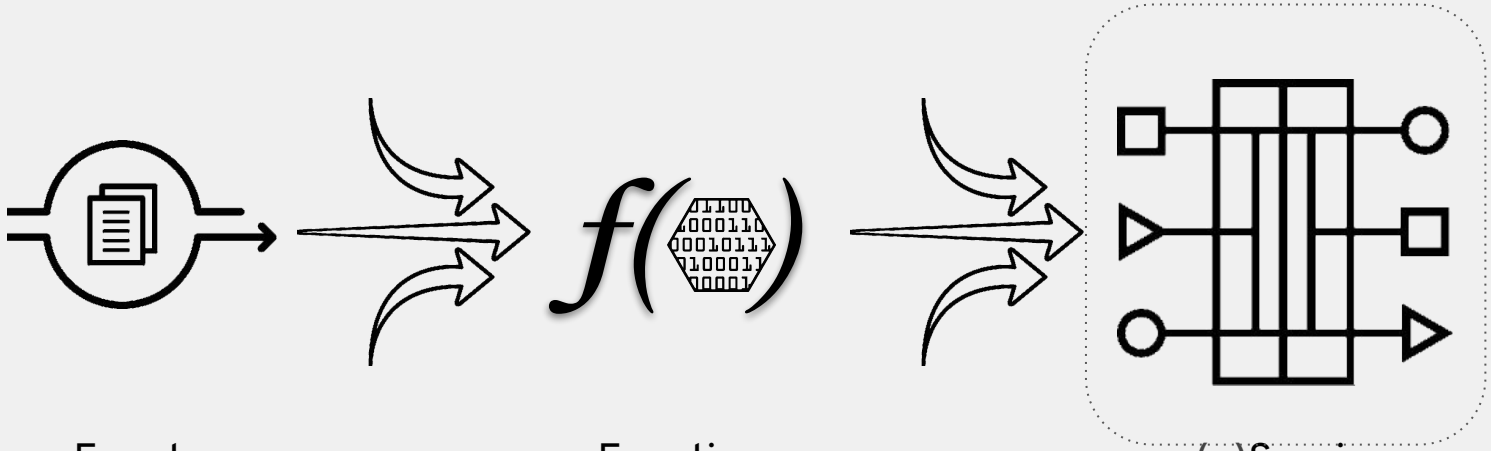
faas
  - Patching

faas
  - Error handling

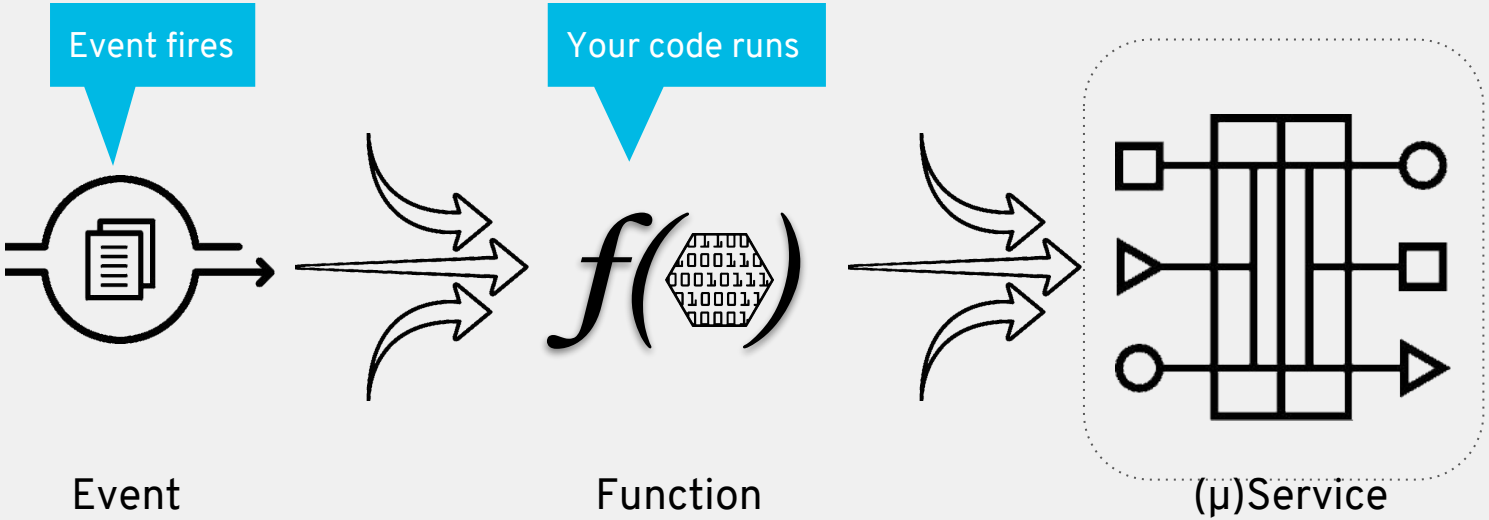
faas
  - Scale

serverless **Cost!**

How does it work ?

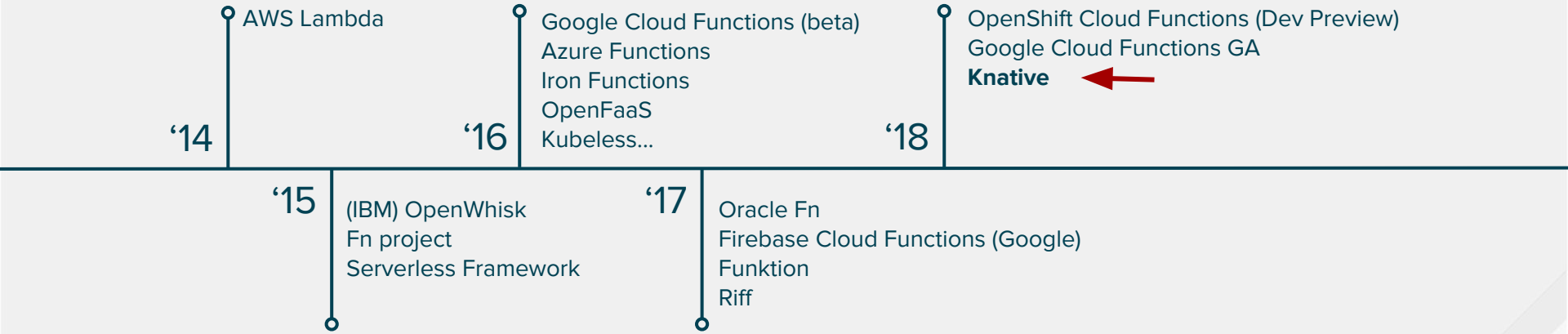


How does it work ?




```
function main() {
    return {payload: 'Hello world'};
}
```

Serverless Timeline



Serverless Cloud Native Landscape

v20180525

See the serverless interactive landscape at s.cncf.io

Greyed logos are not open source

Tools

dashbird, Epsagon, event gateway, Gleo, IO|pipe, Iron.io, Node Lambda, OVERCLOCK, python-λ, SIGMA, STACKERY, THUNDRA

Security

intrinsic, Protego, PURESEC, snyk

Framework

APEX, Architect, aws Chalice, AWS SAM, Claudia.js, FLOGO, gun.io, serverless, Shep, SPARTA, Spring Cloud Function

Platform

Hosted

ALGORITHMIA, AWS Lambda, Azure Functions, BINARIS, Clay, cloudboost, CLOUDFLARE, Google Cloud Functions, HUAWEI, HYPER.SH, IBM Cloud, NANO-LAMBDA, PubNub FUNCTIONS, spotinst, stdlib, syncano, twilio, WEBLAB, ZEIT

Installable

OpenWhisk, AppScale, DISPATCH, fission, fn, GALACTIC FOG, Kubeless, LunchBadger, nuclio, OPENFAAS, OpenLambda, riff

s.cncf.io

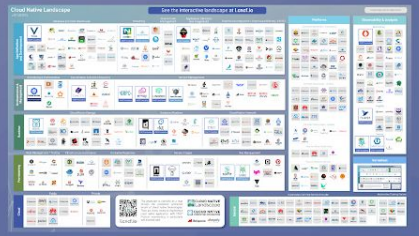
Serverless computing refers to a new model of cloud native computing, enabled by architectures that do not require server management to build and run applications. This landscape illustrates a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment.

CLOUD NATIVE Landscape

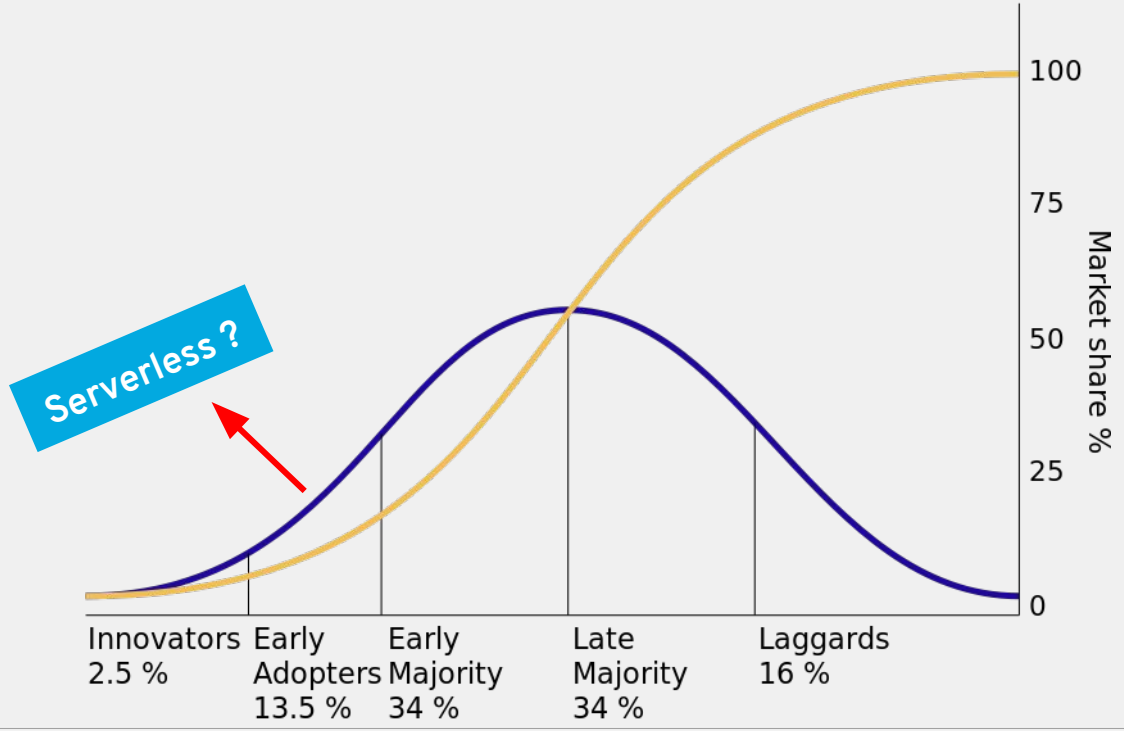
CLOUD NATIVE COMPUTING FOUNDATION

Redpoint

Cloud Native Landscape



Serverless Timeline



* *Everett Rogers, Diffusion of Innovation.*

Knative Overview

Knative Overview - Components

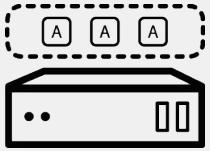
"...an extension to Kubernetes exposing building blocks to build modern, source-centric, and container-based applications that can run anywhere".

Build

A pluggable model for building artifacts, like jar files, zips or containers from source code.

Serving

An event-driven model that serves the container with your application and can "scale to zero".



Events

Common infrastructure for consuming and producing events that will stimulate applications.

Why Knative is not enough ?

- How do you deploy a function on Knative?
  - Write a web server, push the code somewhere, write some yaml, ***kubectl create*** …
    - Versus: write your function and *wsk action create helloWorld hello.js*
  - Which UX do we prefer:
    - Function developers to learn **kubectl** and the Kubernetes API ?
    - OR use tooling designed just for deploying functions ? 
- **Knative** gives us building blocks but explicitly chooses to **not be the entire platform**
  - *“Knative components are intended to be integrated into more polished products that cloud service providers or in-house teams in large enterprises can then operate.”* - https://github.com/knative/docs/#operators

Knative and Red Hat

OpenShift Cloud Functions

Value proposition

It's **your** FaaS

Run **on-premise** or **on cloud**

Custom memory & timeout limits

- More flexible than cloud providers.

Available in OpenShift Online.

Local development environment

- Support for Minishift

Enterprise ready

Robust security and authentication

Fully tested, patched and supported

Integrated monitoring interface*

Repackaged runtimes (CentOS/RHEL)

Supported OpenShift Online & OCP

Dev tools with Che support

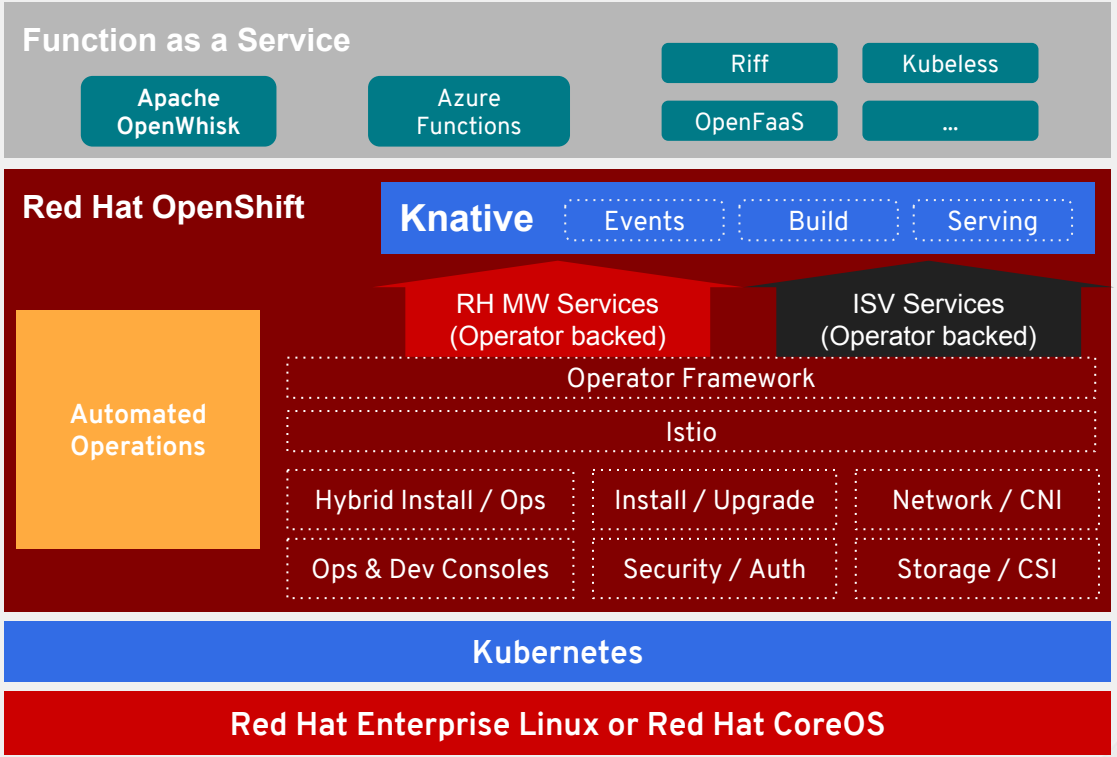
Red Hat OpenShift and Serverless

Developer experience
APIs, CLI, service binding

Building blocks for serverless
Source-centric and container-based

The leading enterprise Kubernetes platform
Automated Operations
Build an run anywhere (Hybrid Cloud)

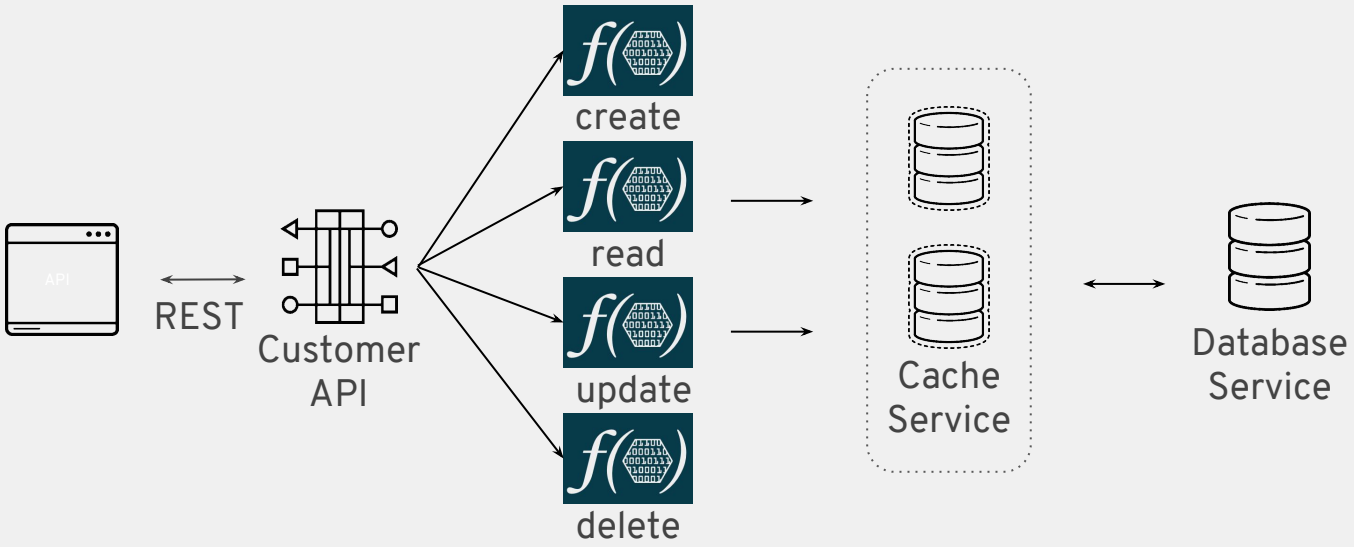
RED HAT® OPENSHIFT Container Platform



Serverless Use Cases

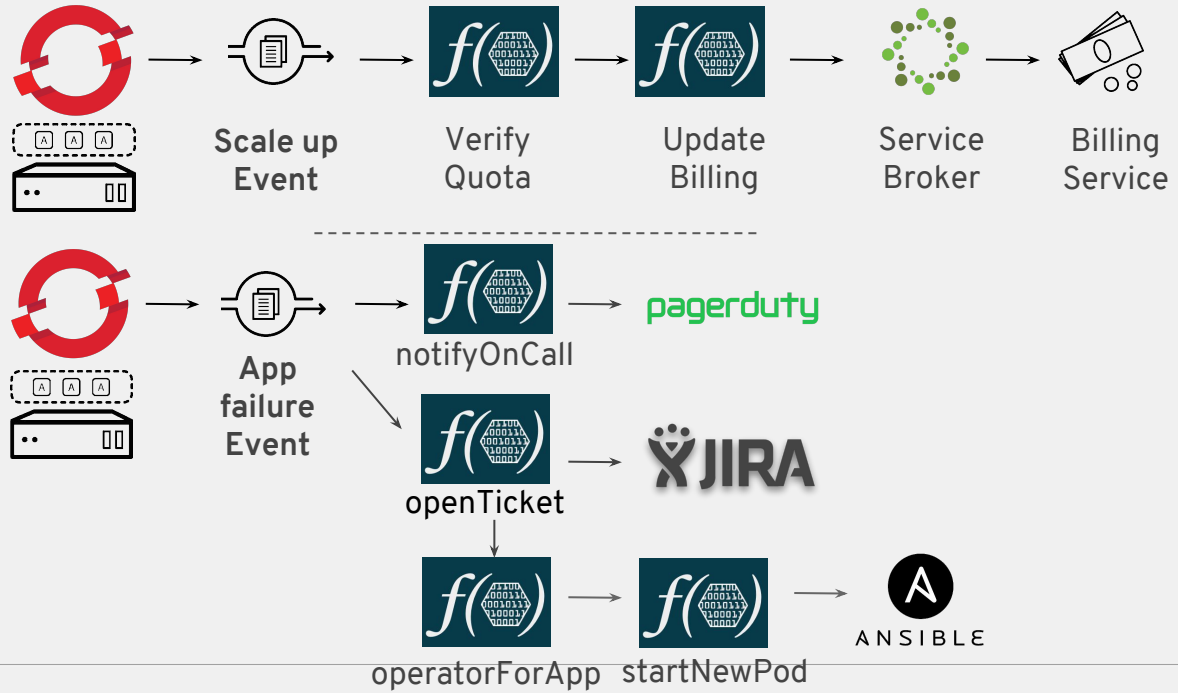
Serverless use cases

Web APIs



Serverless use cases

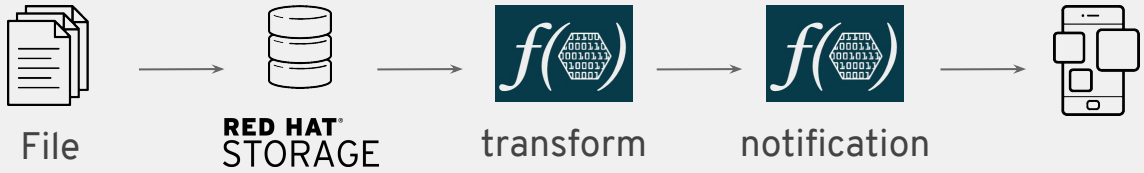
OpenShift event monitoring



Serverless use cases

Storage

File System / S3 Events



Machine Learning

Other common use cases…

- Processing web hooks
- Scheduled tasks (a la cron)
- Data transformation
- Mobile image manipulation (compression, conversion, and so on)
- Voice packet to JSON transformation (Alexa, Cortana, and so on)
- Mobile video analysis (frame-grabbing)
- PDF generation

Web

Mobile

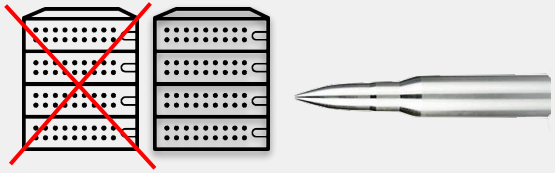
IoT

DevOps Automation

Focus on convenience and business value, no distractions.

When not to use serverless

- *Real-time, ultra-low latency applications*
- *Long running tasks that can't be split into steps*
- *Advanced or complex observability and monitoring requirements*
- *Memory or CPU requirements are very demanding and specific*
- *Can't deal with cold-start...*



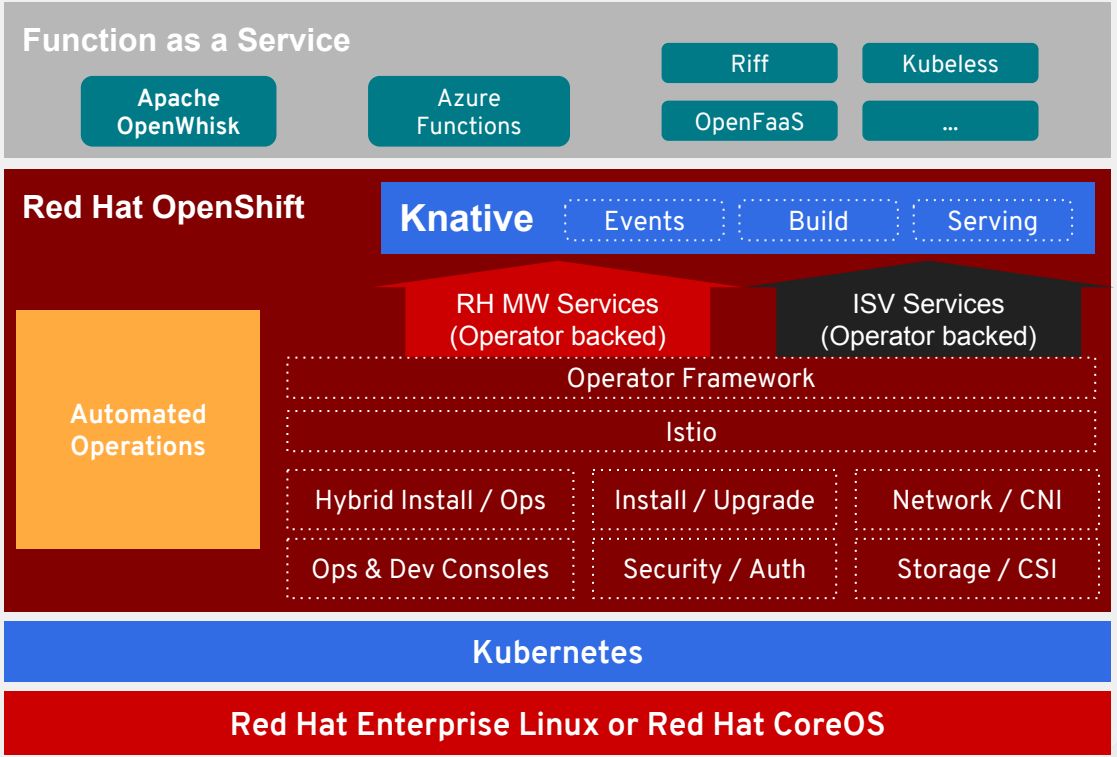
Red Hat OpenShift and Serverless

Developer experience
APIs, CLI, service binding

Building blocks for serverless
Source-centric and container-based

The leading enterprise Kubernetes platform
Automated Operations
Build an run anywhere (Hybrid Cloud)

RED HAT® OPENSHIFT Container Platform



Thankyou

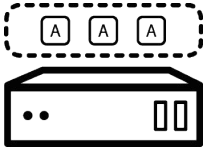
Appendix

Knative Overview - Builds

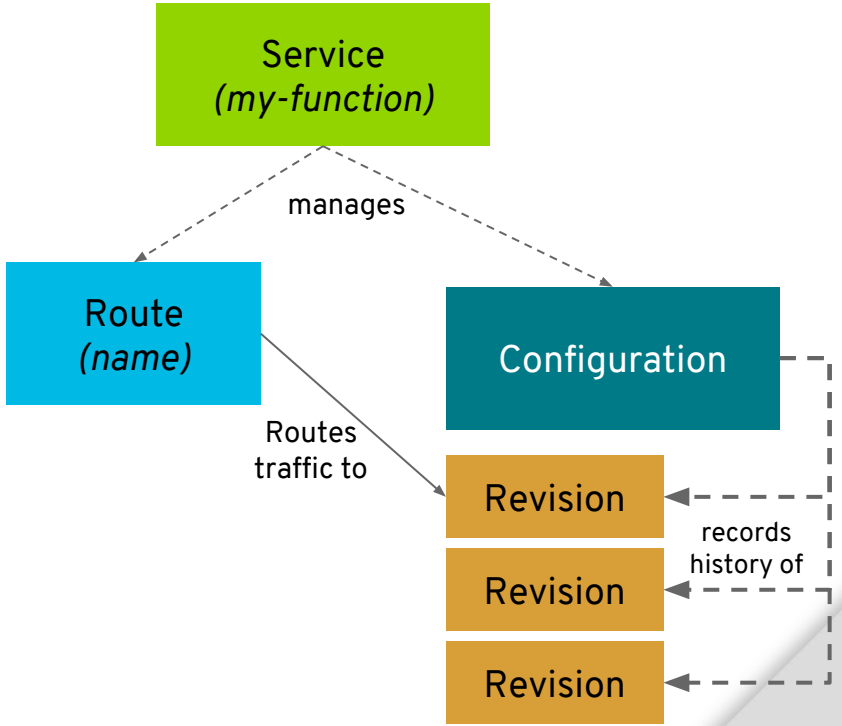
- A **Build** is a list of containers run in-order, with source mounted in
- **BuildTemplates** provide reusable, parameterized recipes that can be used to create Builds
- Pipelines ? TBD.
- Build Example
- "Source to URL"

```yaml
apiVersion: build.knative.dev/v1alpha1
kind: Build
metadata:
  name: example-build
spec:
  serviceAccountName: build-auth-example
  source:
    git:
      url: https://github.com/example/build-example.git
      revision: master
  steps:
  - name: centos-example
    image: centos
    args: ["centos-build-example", "SECRETS-example.md"]
  steps:
  - image: quay.io/example-builders/build-example
    args: ['echo', 'hello-example', 'build']
```

KNative Overview - Serving



- **Configurations** represent the 'floating HEAD' of a history of **Revisions**
- **Revisions** represent immutable snapshot of code and configuration
- **Routes** configure ingress over a collection of Revisions and/or Configurations
- **Services** (nope, not K8s services) are top-level controllers that manage a set of Routes and Configurations to implement a network service



KNative Overview - Events

- Goal is to be a generalized eventing framework
- Model is still in significant flux
- Latest proposal was unveiled this week (Sep 5th)
- Many influencers also active on CloudEvents specification

Current-ish API:

- **EventSources** represent systems that produce different **EventTypes**
- **Binds** represent triggers on EventSources that send Events to a destination
- **Channels** are named endpoints which accept event delivery
- Receivers use **Subscriptions** to register to receive traffic from a Channel
- **Busses** represents the software systems that back Channels
    - Kafka, GCP pubsub, etc.

OpenWhisk

- *Complete Serverless solution*
- *Incubating at Apache Software Foundation*
- *Community mostly IBM w/ Adobe and Red Hat as contributors*

- *Has yet to adopt Knative*
- *Community is concerned with graduation*
- *IBM already has a GA product (can wait…)*