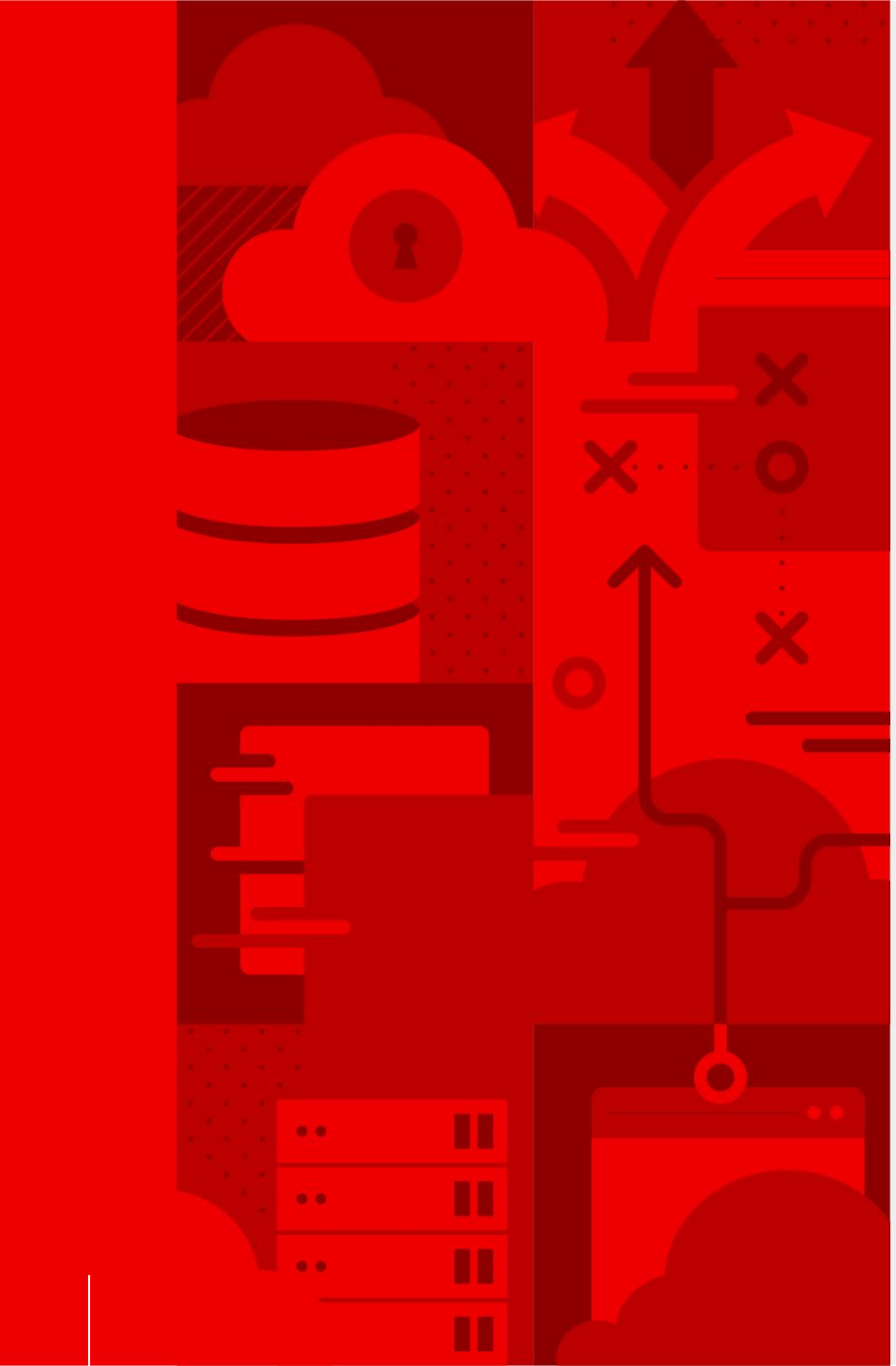


Containerful Serverless

Roland Huß **@ro14nd@hachyderm.io**
Senior Principal Software Engineer, Red Hat
OpenShift Serverless Architect
Ex-Knative TOC member

Matthias Weßendorf **@matzew@hachyderm.io**
Principal Software Engineer, Red Hat
OpenShift Serverless Engineer
Knative contributor

Red Hat

Wait … wat ?

-- CNCF Definition, https://www.cncf.io/blog/2018/02/14/cncf-takes-first-step-towards-serverless-computing/

Red Hat

What is “Serverless” ?

“Serverless computing 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”

-- CNCF Definition, https://www.cncf.io/blog/2018/02/14/cncf-takes-first-step-towards-serverless-computing/

Serverless vs. FaaS

Serverless is a **Deployment Model** that abstracts away the driving machine infrastructure.

- No server management required
- Executed, scaled and billed according to demand
- Defines a deployment packaging, but otherwise agnostic to the application

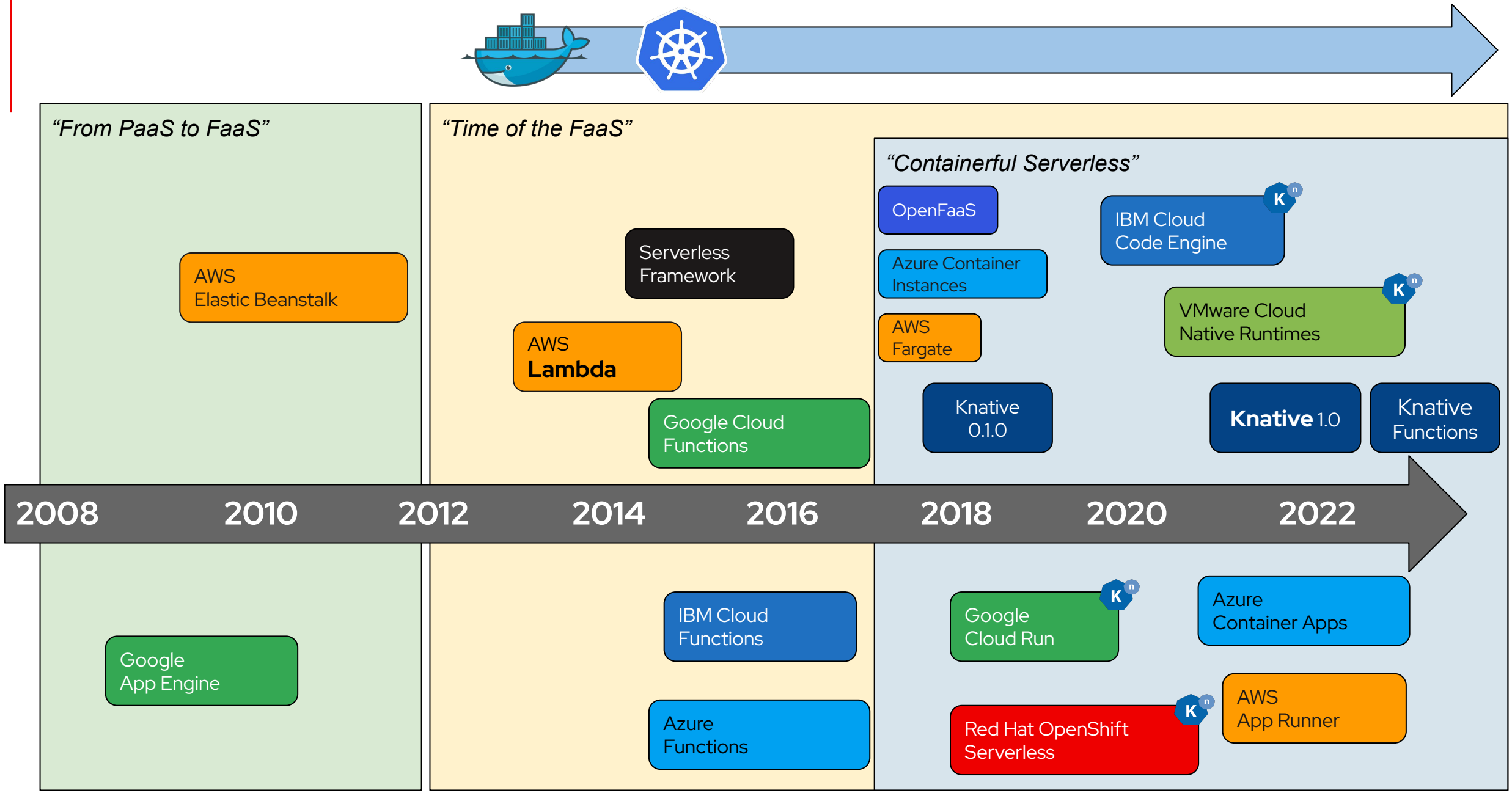
FaaS (Function-as-a-Service) is a **Programming Model** that mandates developing your application with fine grained function that match a given signature.

- Deployed as Serverless application
- Typically used as *glue code* to connect services

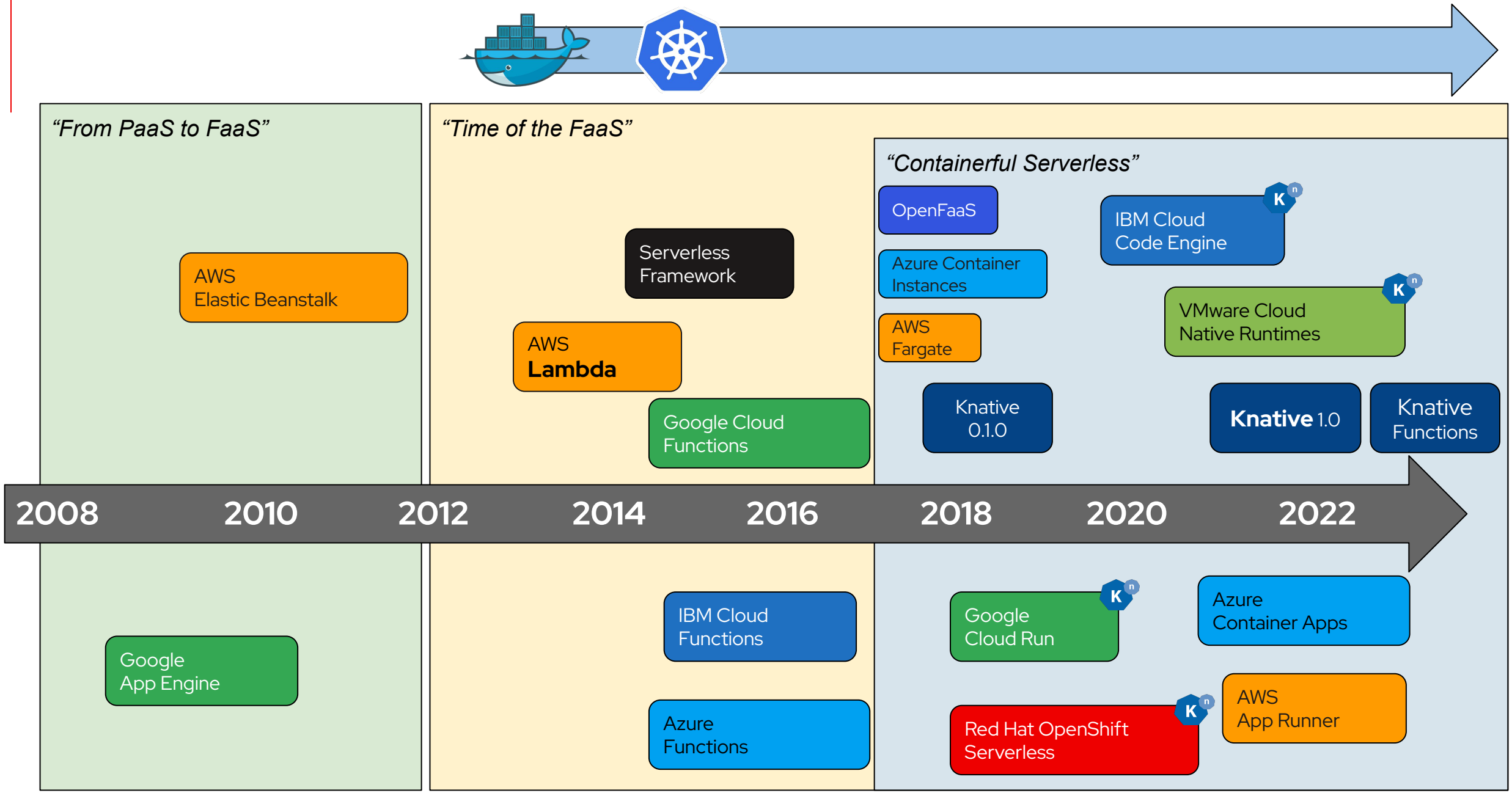
Serverless Workloads

The ideal Serverless Workload is ...

- ... **stateless**
- ... **short-running**
- ... **HTTP**-based web services or **Event-Driven** Applications


"From PaaS to FaaS"
"Time of the FaaS"
"Containerful Serverless"
AWS Elastic Beanstalk
Google App Engine
Serverless Framework
AWS Lambda
Google Cloud Functions
IBM Cloud Functions
Azure Functions
OpenFaaS
Azure Container Instances
AWS Fargate
Knative 0.1.0
IBM Cloud Code Engine
VMware Cloud Native Runtimes
Knative 1.0
Knative Functions
Google Cloud Run
Red Hat OpenShift Serverless
Azure Container Apps
AWS App Runner
2008
2010
2012
2014
2016
2018
2020
2022

Red Hat


“From PaaS to FaaS”
“Time of the FaaS”
“Containerful Serverless”
AWS
Elastic Beanstalk
Serverless
Framework
AWS
Lambda
Google Cloud
Functions
OpenFaaS
Azure Container
Instances
AWS
Fargate
IBM Cloud
Code Engine
VMware Cloud
Native Runtimes
Knative
0.1.0
Knative 1.0
Knative
Functions
2008
2010
2012
2014
2016
2018
2020
2022
Google
App Engine
IBM Cloud
Functions
Azure
Functions
Google
Cloud Run
Azure
Container Apps
Red Hat OpenShift
Serverless
AWS
App Runner

Red Hat

Serverless Packaging

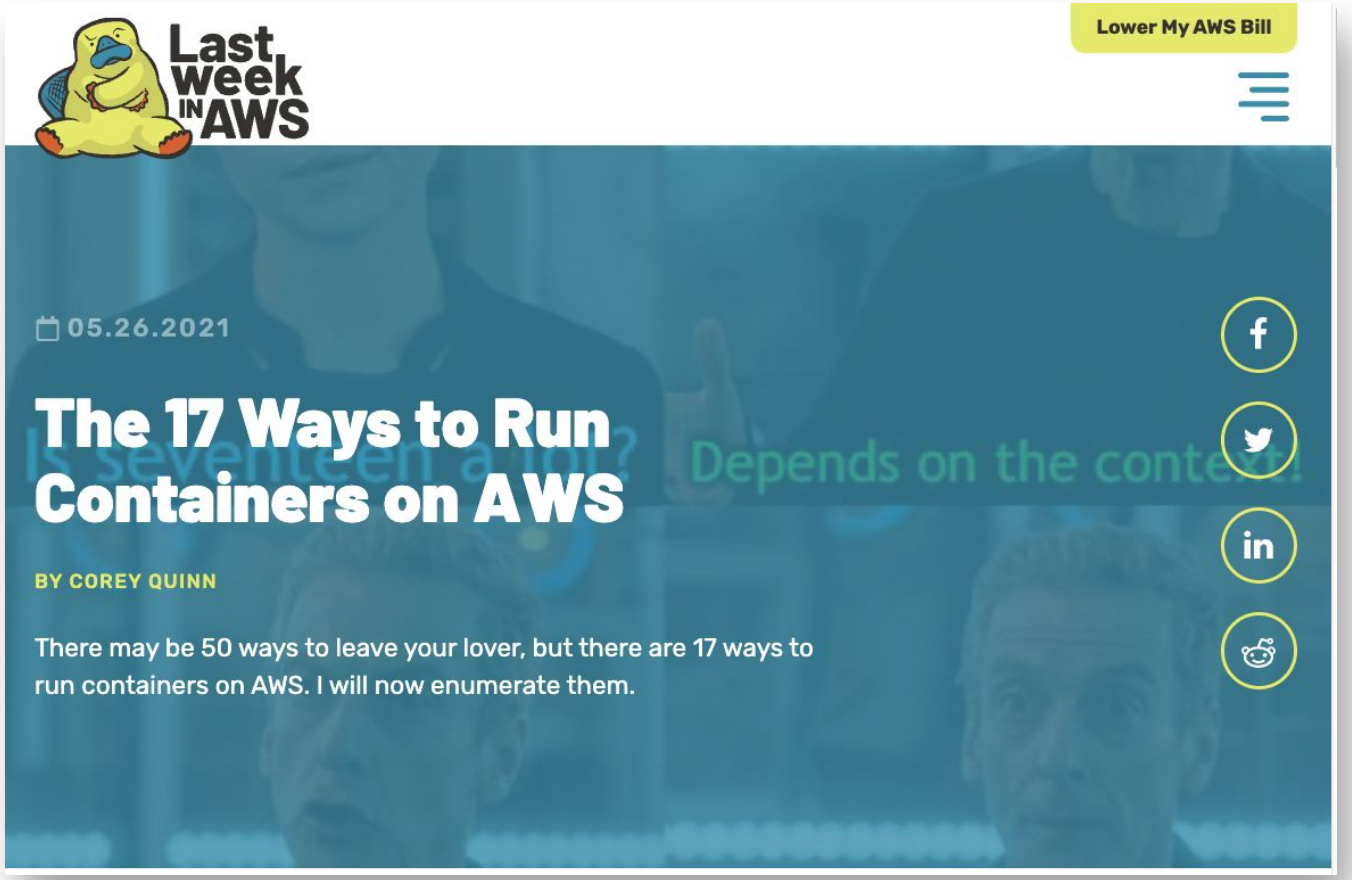
- **Custom** (zip, tar, ...)
  - All Cloud Providers
- **OCI Container Images**
  - Docker, Podman, Kubernetes, ...
- **Virtual Machines**
- **MicroVMs**
  - Firecracker, Kata containers, gVisor, ...
- **Web Assembly Modules (WASM)**
  - Krustlets, WasmEdge, Wasmtime

Containerful Serverless

- **AWS:** Lambda, Fargate, App Runner
- **Microsoft Azure:** Container Instances, Container Apps
- **OpenFaaS**
- **Knative**
  - **Google** Cloud Run, **IBM** Code Engine, **VMware** Cloud Native Runtimes for Tanzu, **Red Hat** OpenShift Serverless


AWS

Containers on AWS

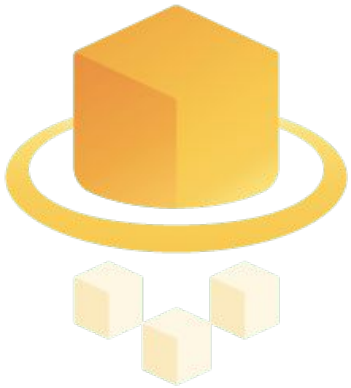


https://www.lastweekinaws.com/blog/the-17-ways-to-run-containers-on-aws/

AWS Lambda

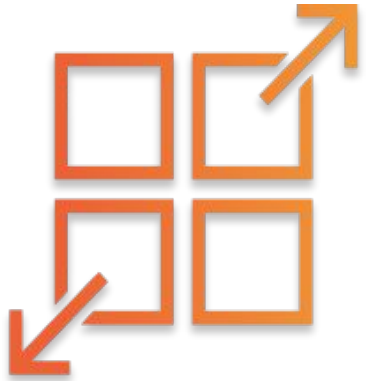
- Opinionated, **event-driven** computing platform
- Introduced **2014** with **Node.js** as runtime
- Main use case as '**glue code**' for connecting AWS services
- **Pay-as-you go** with 1 ms billing granularity
- **Container** packaging since 2019
  - up to 10 GB in size (instead of 50 MB zipped)
  - specific base images or clients needed to fullfill Lambda runtime contract

AWS Fargate



- Introduced in 2017 as a computing **backend for ECS**
- Deploying containers without managing EC2 instances.
- Fargate can be used in EKS do deploy Kubernetes Pods on Fargate
  - “Infinite large Kubernetes Node”

AWS App Runner



- Simplest and most opinionated way to deploy containers on AWS
- HTTP workloads
- Automatic Scaling based on traffic
- Automatic routing (public or private URL)
- No scale to zero

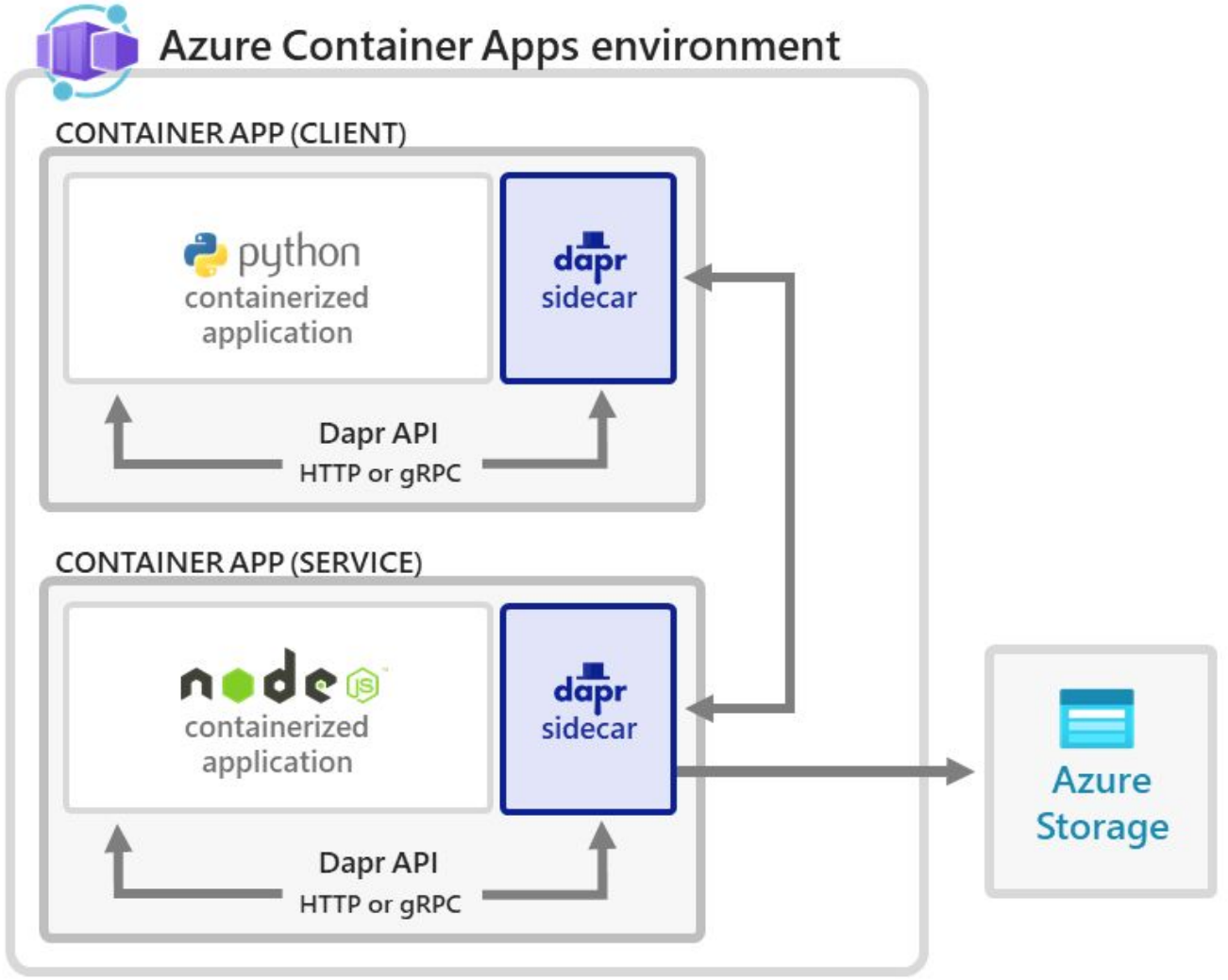

Azure

Container Instances (ACI)

- Started in 2017
- Similar scope as AWS Fargate
- No scaling or load balancing
- Unopinionated approach to start containers
- Simpler to use than ECS / Fargate on AWS
- Can be added as virtual nodes to AKS

Container Apps

- Started late 2021, since May 2022 GA
- Multiple revisions with traffic split
- Autoscaling options: HTTP, CPU/Memory, KEDA
- Native Dapr integration




OpenFaaS

OpenFaaS

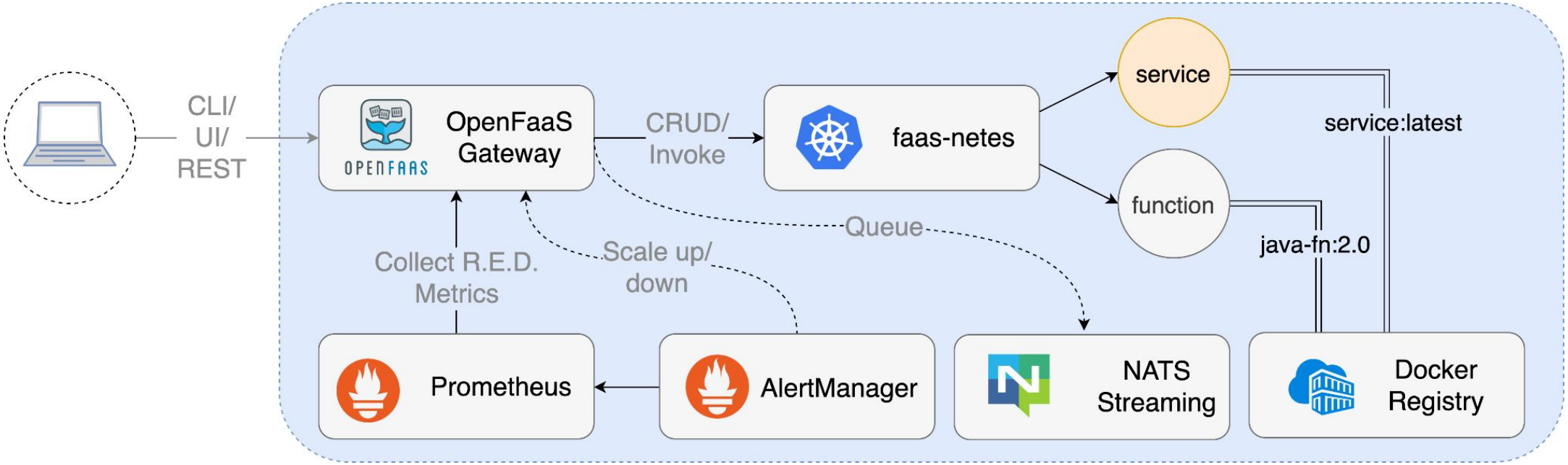
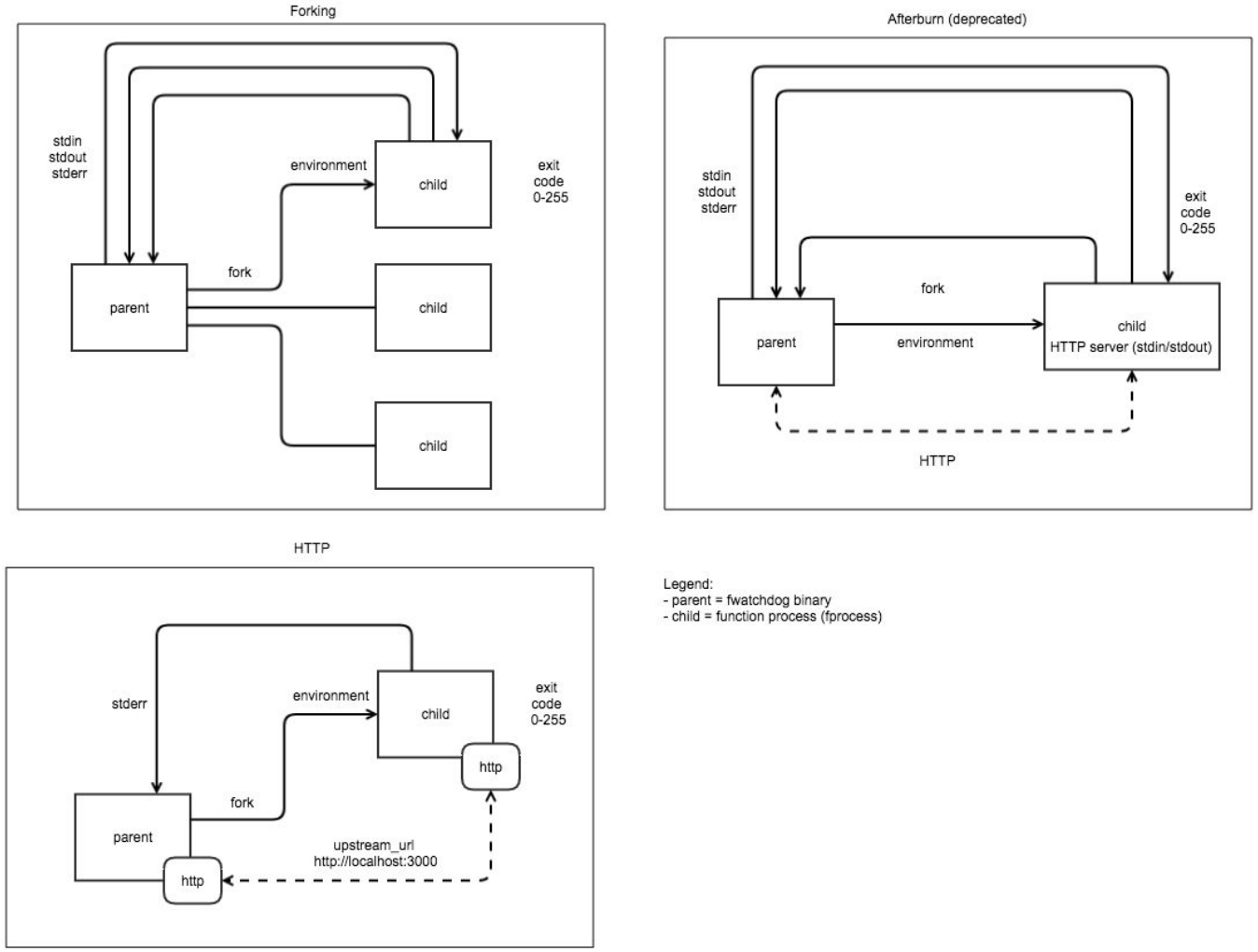


diagram from https://docs.openfaas.com/architecture/stack/

WatchDogs



reference: https://docs.openfaas.com/architecture/watchdog/

Open Core

- Exclusive Features of OpenFaaS Pro:
  - New Autoscaling engine including scale-to-zero
  - Retry of failed functions
  - New User Interface
  - Support for EDA application via triggers (Kafka, AWS SQS)
- No free trial (?)


Knative

Kubernetes-based platform to **deploy** and **manage** modern **serverless** workloads.

https://knative.dev

Modern Event-driven Workloads with Knative

W-JAX, November 10th, 12:00 - 13:00
Room Atlanta

Bringing Serverless applications to Kubernetes

Serving

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

Eventing

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

Functions

A programming model that lets you focus on just your code for faster iterations.

Background Information

- Started as an **Open Source** Project mid-2018 by Google
- CNCF Incubating Project since March 2022
- Community driven with a lot of vendor backing
  - https://github.com/knative
  - https://knative.dev
  - Support by Google*, Red Hat, IBM, VMware, Triggermesh and more
  - Organized in multiple Working Groups with weekly meetings
- Releases
  - Current: **v1.8**
  - quarterly release cadence (starting 2023)

Serving Concepts

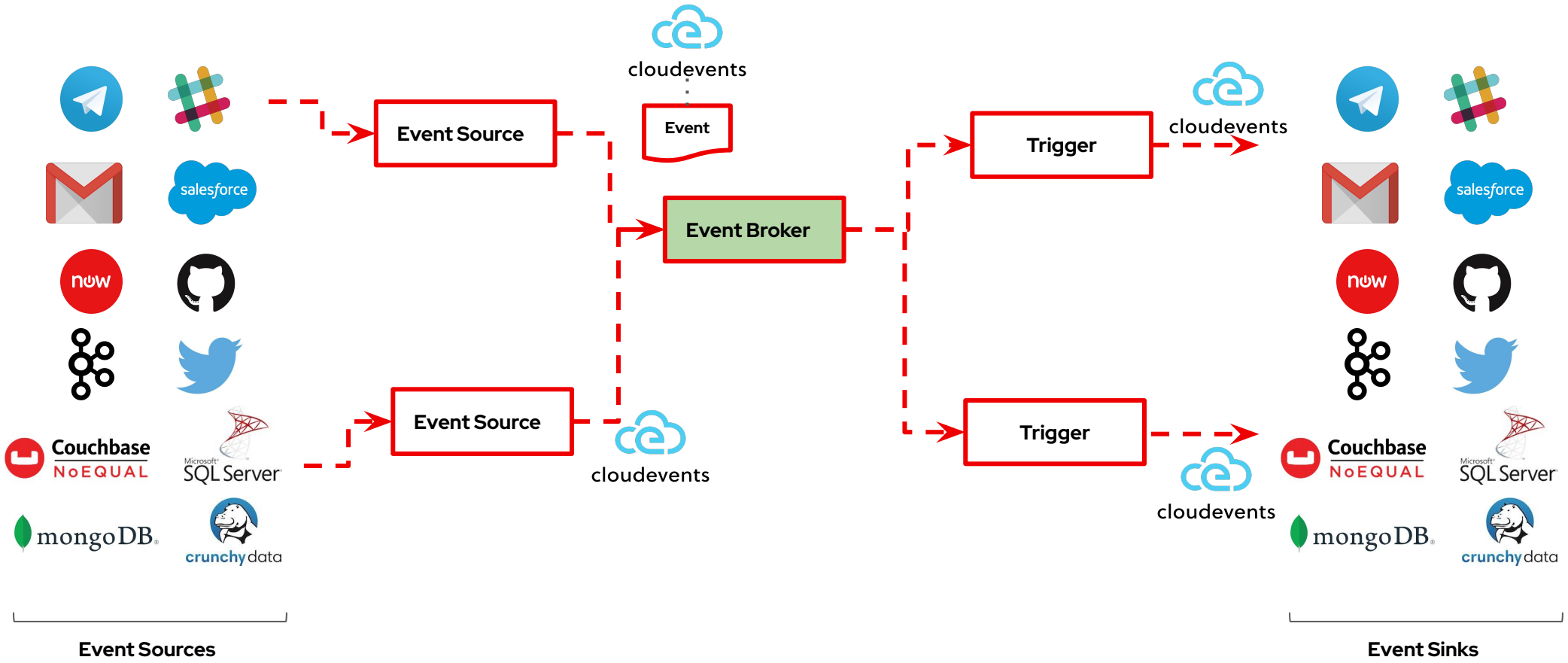
- **Demand-based autoscaling**, including scale-to-zero
- **Separation** of code and configuration
- Simplified deployment model catered for **stateless applications**
- Rich **traffic split capabilities** to enable custom rollout strategies of new versions

From **Deployment** to **KService**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: random
spec:
  replicas: 1
  selector:
    matchLabels:
      app: random
  template:
    metadata:
      labels:
        app: random
    spec:
      containers:
      - image: rhuss/random
        name: random
        ports:
        - containerPort: 8080
```

```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: random
spec:
  replicas: 1
  selector:
    matchLabels:
      app: random
  template:
    metadata:
      labels:
        app: random
    spec:
      containers:
      - image: rhuss/random
        name: random
        ports:
        - containerPort: 8080
```

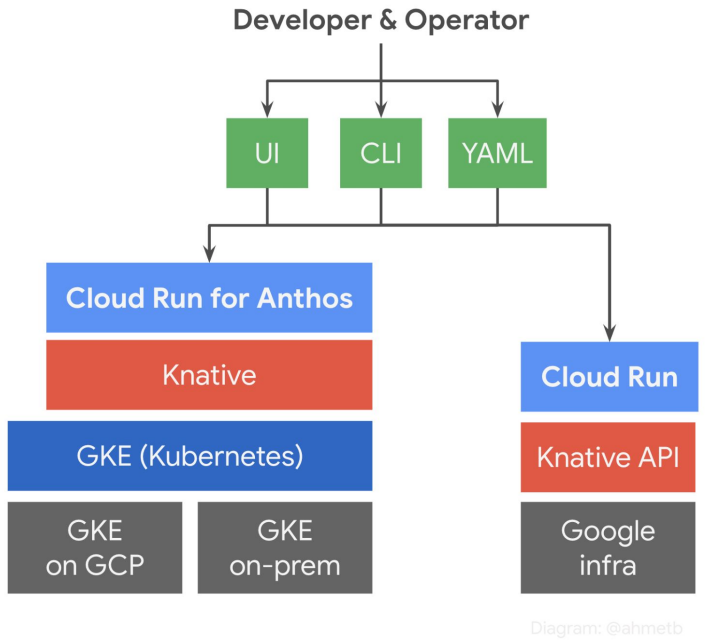
Event Driven Applications



Red Hat

Google Cloud Run

- Two variations:
  - Google Cloud Run → Knative API implemented on App Engine
  - Google Cloud Run For Anthos → Knative on Kubernetes
- Events for Cloud Run on Anthos
  - Knative Eventing on GKE
- Eventarc (inspired by Eventing)
- Not all Knative features implemented
  - see https://ahmet.im/blog/cloud-run-is-a-knative (?)



IBM Cloud Code Engine

- Knative on the IBM Cloud
- Runs on IKS
- KafkaSource support
- Support for Batch Jobs
- In-Cluster container builds via Cloud Native BuildPacks

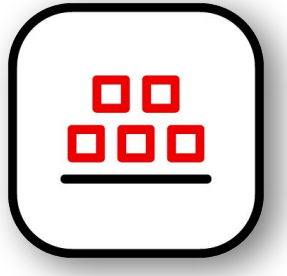
Cloud Native Runtimes for VMware Tanzu

- Knative distribution on-top of Kubernetes
- Integral part of the Tanzu Application Platform (TAP)
- Eventing Broker is backed by RabbitMQ
- Includes TriggerMesh Sources for Amazon Web Services (SAWS)

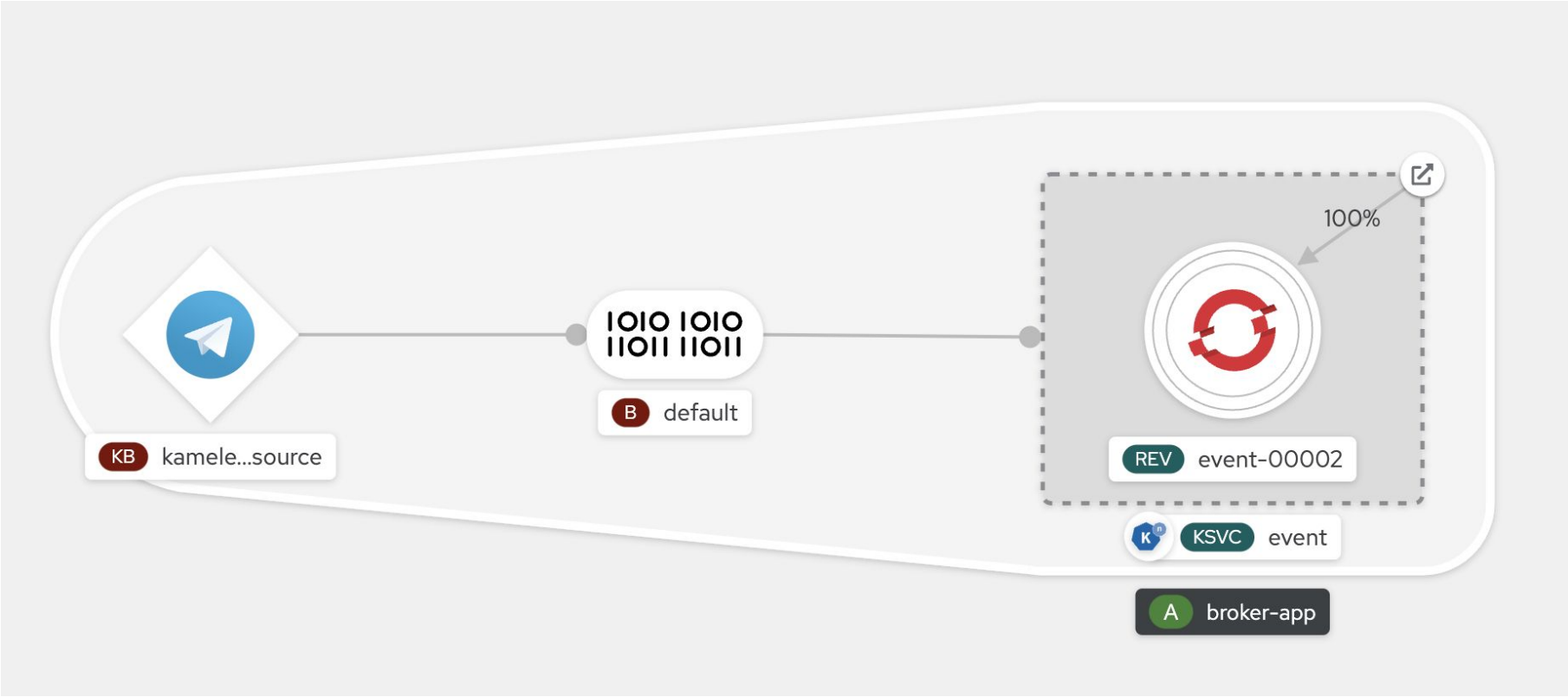
Red Hat OpenShift Serverless

- Knative bundled with Red Hat OpenShift
- Runs everywhere where OpenShift runs
  - on-prem
  - AWS (ROSA)
  - Azure (ARO)
  - IBM (ROKS)
- Adds operator-based installation experience
- Kafka-backed Knative broker
- Kafka EventSource and Kafka EventSink
- 50+ sources via Apache Camel connectors (Kamelets)

Red Hat OpenShift Serverless



- Great visualization in the OpenShift Console



Red Hat

Challenges

Coldstart

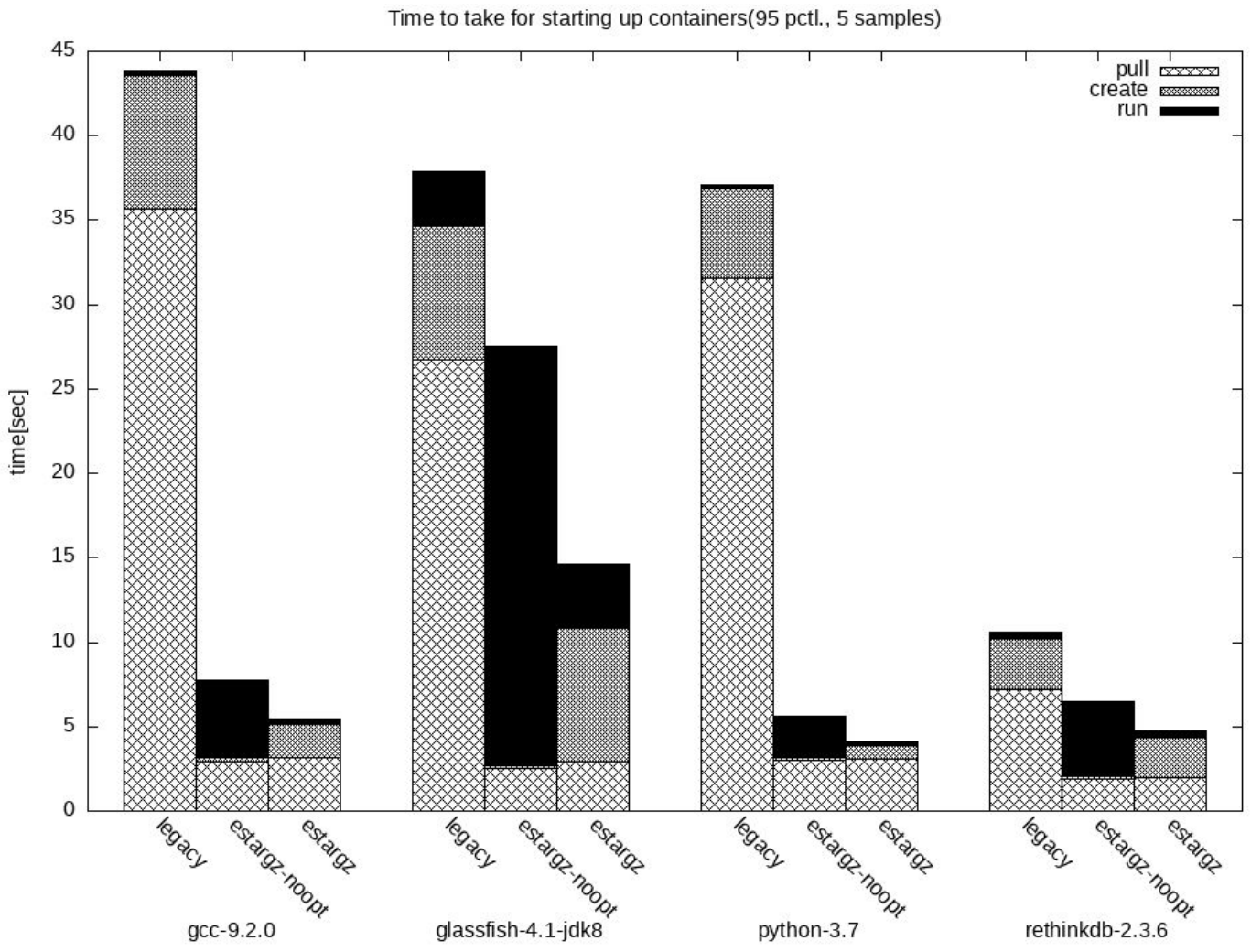
- Application startup times in Kubernetes:

- Scheduling time depends on cluster size and can take 10s for a 1000 nodes cluster
- Application startup depends on the application stack
  - Fast for optimized frameworks like Quarkus or Node
  - JavaEE servers are no good fit
- Pod serves requests when readiness probes succeed
  - Kubelet has a 1 s granularity for checking probes

Image Pull

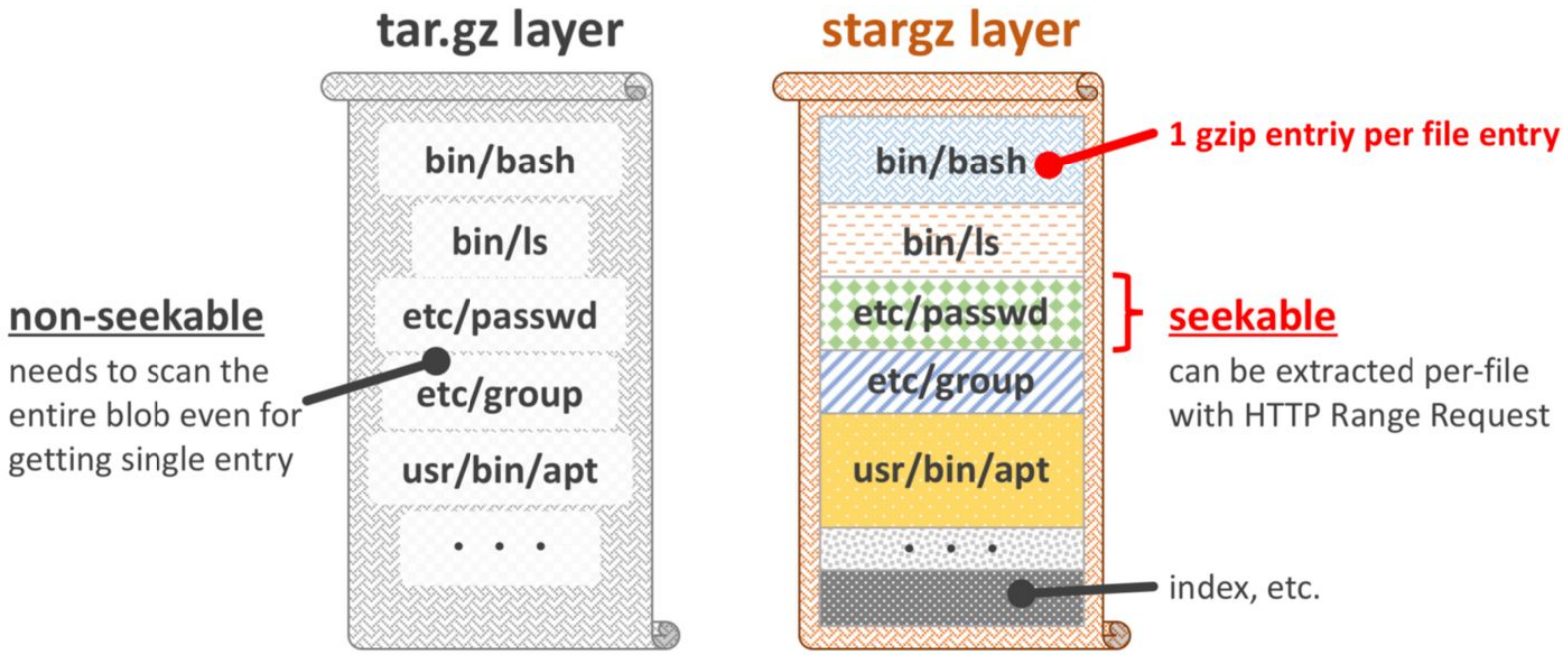
- Application image needs to be pulled from a container image registry
  - Cached at the Container Runtime for future usage
  - Workarounds: Pre-warm with a Daemonset
- Lazy pulling with stargz snapshotter
  - Files from layers only loaded when referenced
  - eStargz: Measure order of files access and rearrange the image

containerd/stargz-snapshotter



Red Hat

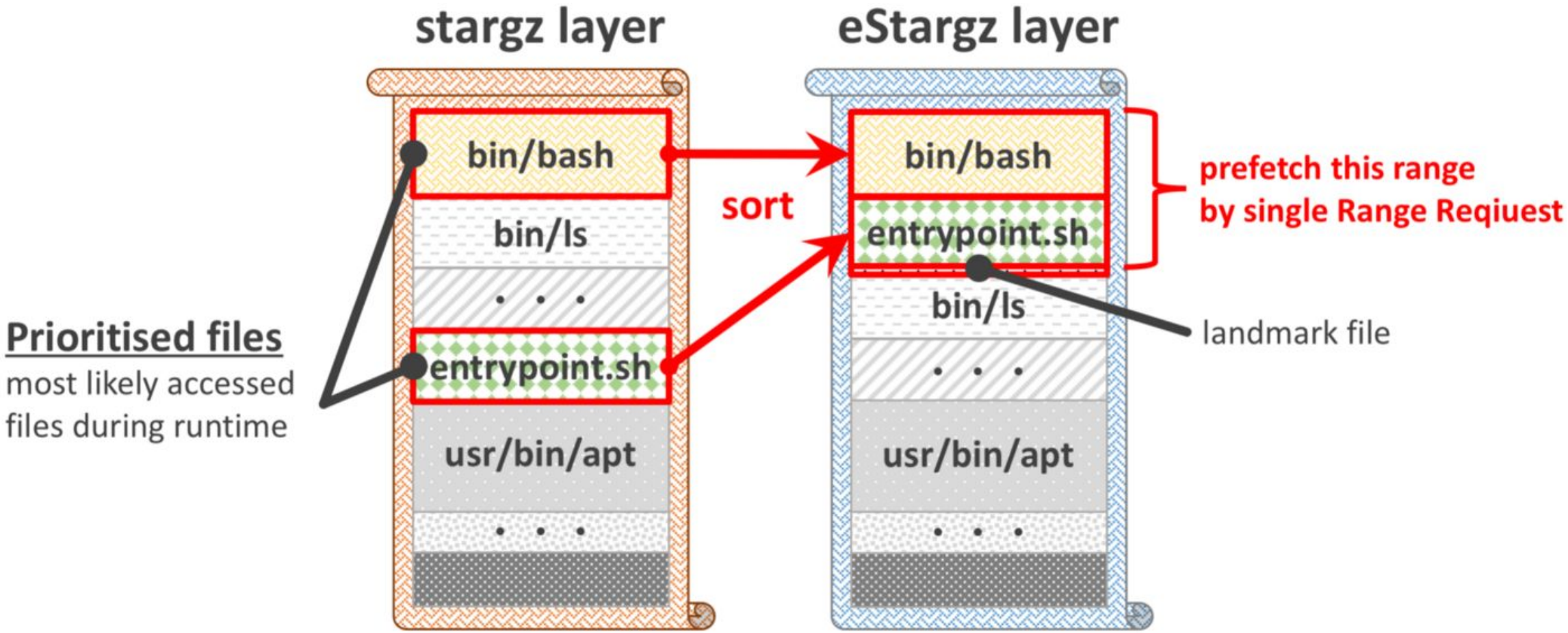
Stargz ("seekable tar.gz")



- tar: Non-indexed, gz: Non-Seekable
- Solution: Gzip tar-entries individually, concatenate and add gzipped-index file and tar footer

Source: https://medium.com/nttlabs/startup-containers-in-lightning-speed-with-lazy-image-distribution-on-containerd-243d94522361

eStargz Optimization

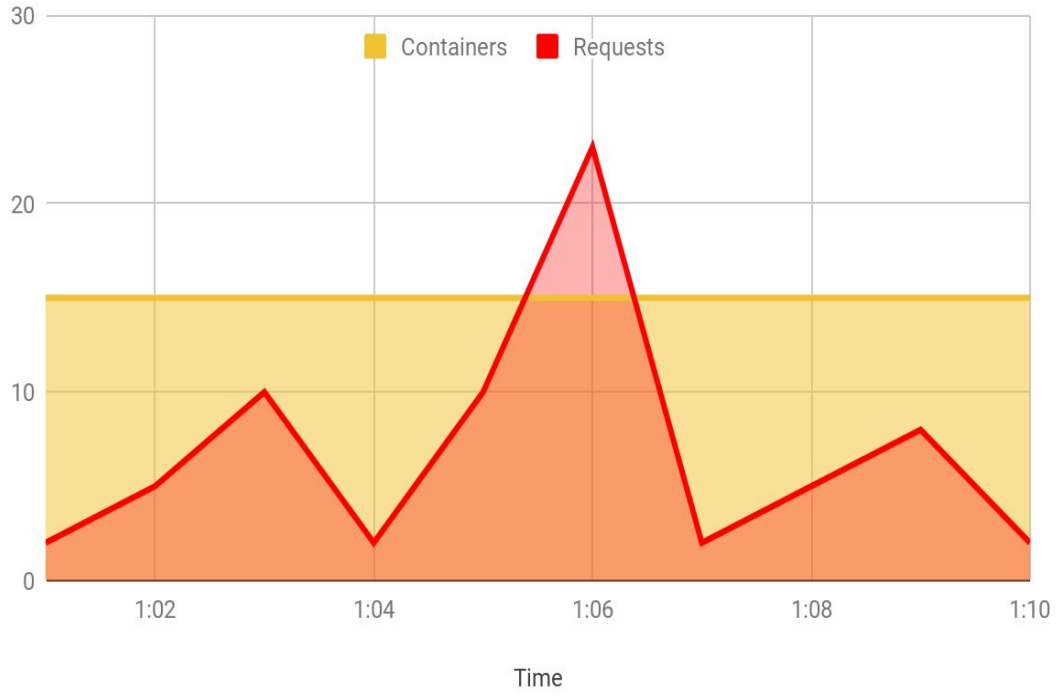


Source: https://medium.com/nttlabs/startup-containers-in-lightning-speed-with-lazy-image-distribution-on-containerd-243d945223ä61

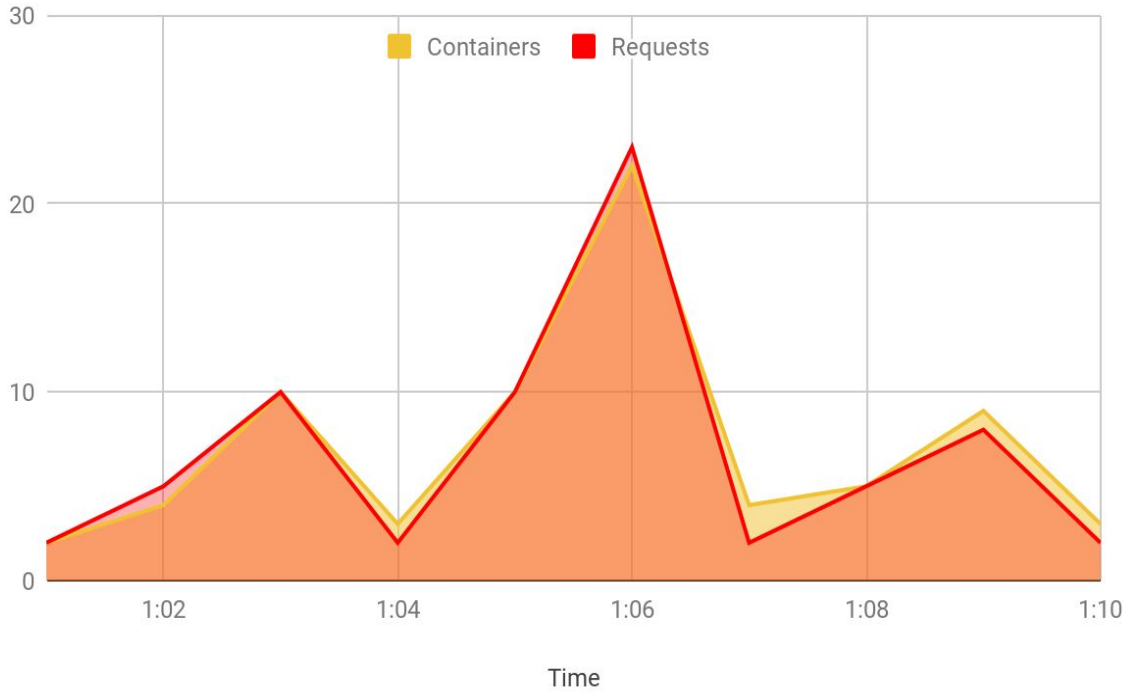
Possible Startup Optimizations

- Pooling of network interfaces
- Move the Pod Lifecycle Event Generator (PLEG) to an event-driven model instead of 1 s polling (requires change to the CRI)
- Precreated pool of “Pod hulls”
- More pooling in general
- See Cold start latency improvements in Kubernetes and Knative for more details

Autoscaling



NOT Serverless

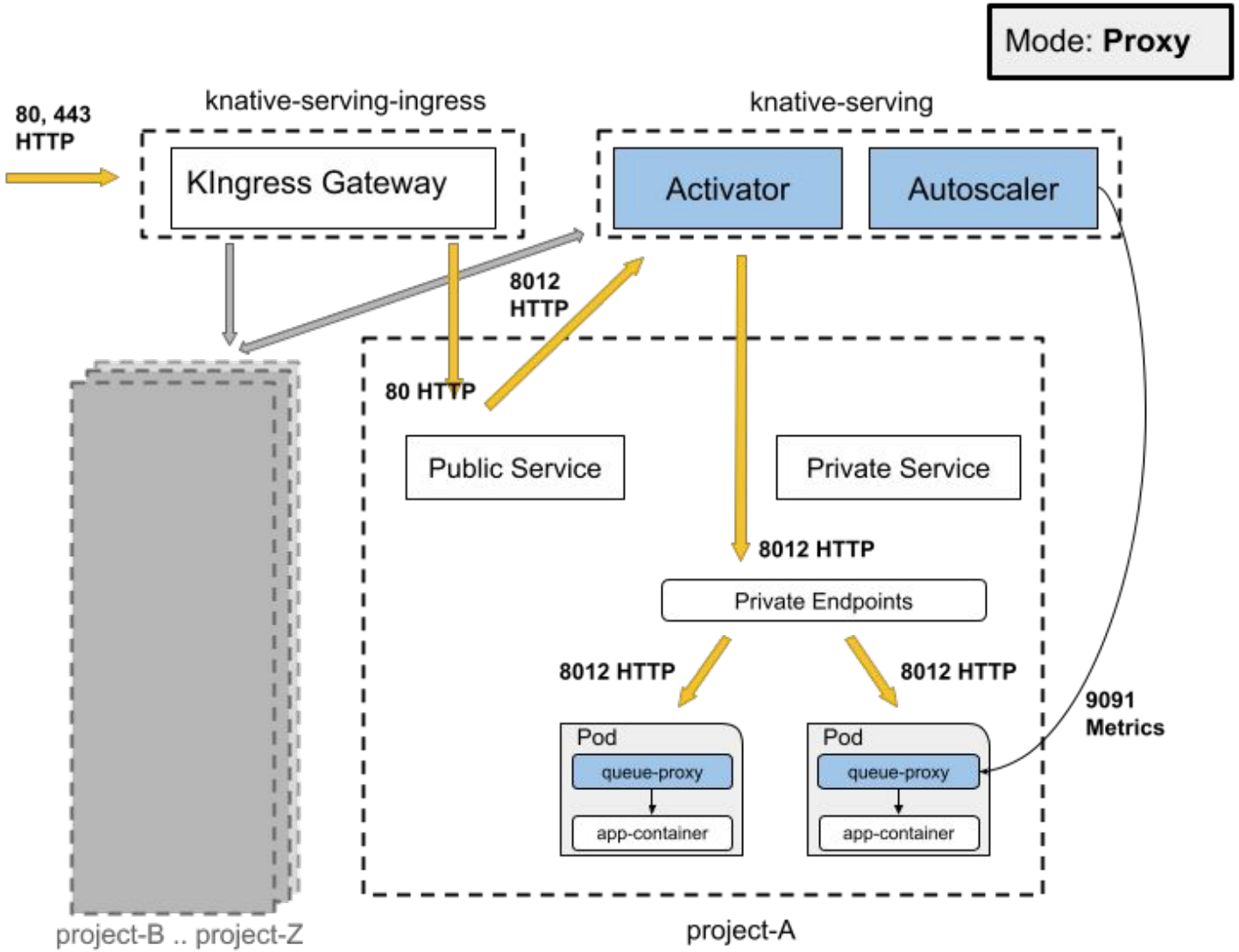


with Serverless

Autoscaling on Kubernetes

- Kubernetes Native:
  - Horizontal Pod Autoscaler (HPA)
  - Vertical Pod Autoscaler (VPA)
  - Cluster Autoscaler (CA)
- Knative Autoscaler
- KEDA

Knative Autoscaling



Red Hat


Pros & Cons

Container Drawbacks

- Container Image needs to be **built**
  - Explicit via Docker/Buildah tooling
  - Automatically by language specific tooling
    - Cloud Native Buildpacks, Shipwright, Eclipse JKube ...
  - Knative Functions
- **Coldstart** times
- Extra overhead because of **additional abstraction**

Container Benefits

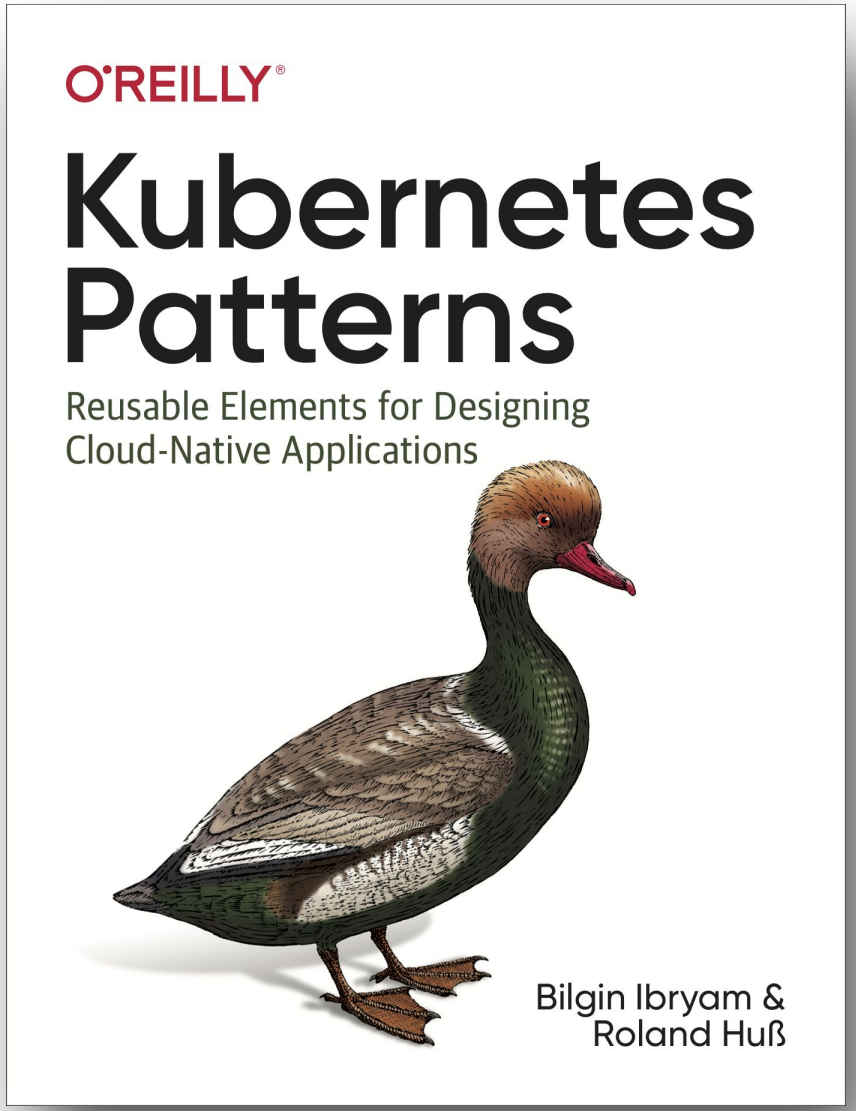
- **Standard** format (OCI)
- **No vendor locking** per se
  - Multitude cloud offering for container deployments
- No language restrictions
- Rich tooling landscape
- Hybrid scenarios (on-prem combined with cloud) easily possible
- Versioning
- Powerful orchestration platforms underneath (Kubernetes)

The Future

What's (maybe) next ?

- **Web Assembly** (WASM) as a popular deployment format for serverless functions
- **Serverless** Kubernetes **Control-Plane**
  - so that optional features don't use resources
- **Sub-second cold start** times become a reality
- **Scale-to-Zero** as a **core** Kubernetes feature
- **Hybrid Serverless** with multi tenancy

Thank you



https://k8spatterns.io

Red Hat

Picture Credits

https://www.pexels.com/photo/boat-island-ocean-sea-218999/

https://unsplash.com/photos/UGMf30W28qc

https://pixabay.com/photos/hamburg-speicherstadt-channel-2976711/

https://pixabay.com/photos/beer-machine-alcohol-brewery-1513436/

https://www.pexels.com/photo/people-art-abstract-blur-6077519/

https://www.pexels.com/photo/person-holding-a-glass-ball-1252893/

https://me.me/i/aws-lambda-is-just-glorified-cgi-bin-imgflip-com-change-my-mind-d0b715592ba34b08b79452ad02783ca2

https://unsplash.com/photos/vkQgb1lZZPQ

https://unsplash.com/photos/9JrBiphz0e0

https://www.pexels.com/photo/person-rock-climbing-3077882/