



Writing a K8s Operator for Knative Functions

DevConf.CZ 2023

Luis Tomas Bolivar
Jose Castillo Lema



PHYSICS

Agenda

1. Introduction

- a. RH Research
- b. PHYSICS project

2. Serverless / Function as a Service (FaaS)

- a. Knative

3. Operator pattern

4. Hands on

- a. Lab description
- b. Prizes
- c. Useful links



whoami



Luis
TOMAS BOLIVAR



<https://ltomasbo.wordpress.com/>



<https://www.linkedin.com/in/luis-tomas-bolivar-a1022>



260/



<https://github.com/luis5tb>

ltomasbo@redhat.com

whoami



JOSE
CASTILLO LEMA



<https://josecastillolema.github.io/>



<https://www.linkedin.com/in/jose-castillo-lema>



<https://github.com/josecastillolema>



jlema@redhat.com

Introduction



Red Hat Research

Research Interest Groups (RIGs)

- EMEA
- Israel



Horizon 2020
European Union funding
for Research & Innovation



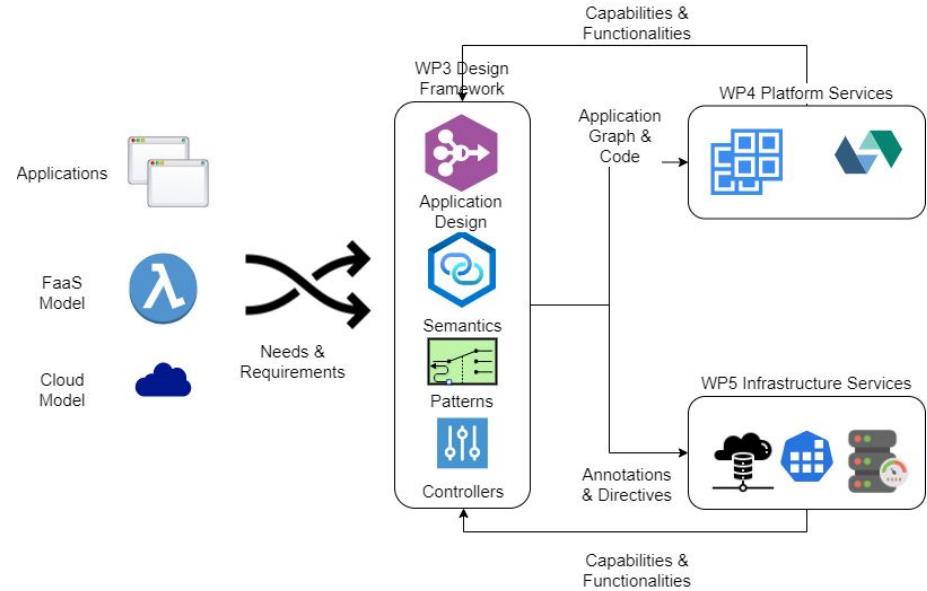
This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047



Project Goals

Visual programming environment to create serverless workflows with reusable patterns and increased semantics

OPTIMIZED HYBRID SPACE-TIME SERVICE CONTINUUM IN FAAS



Platform-level functionalities to orchestrate and deploy FaaS workflows and optimize cloud/edge interplay

Provider-local resource management mechanisms to offer competitive and optimized services execution

Challenges targeted by PHYSICS



Abstract usage of **service offerings and clusters across the Continuum**



Adaptation of code to new **serverless** computing paradigms



Investigation of **space** (location of execution)-**time** (duration of execution) in the continuum

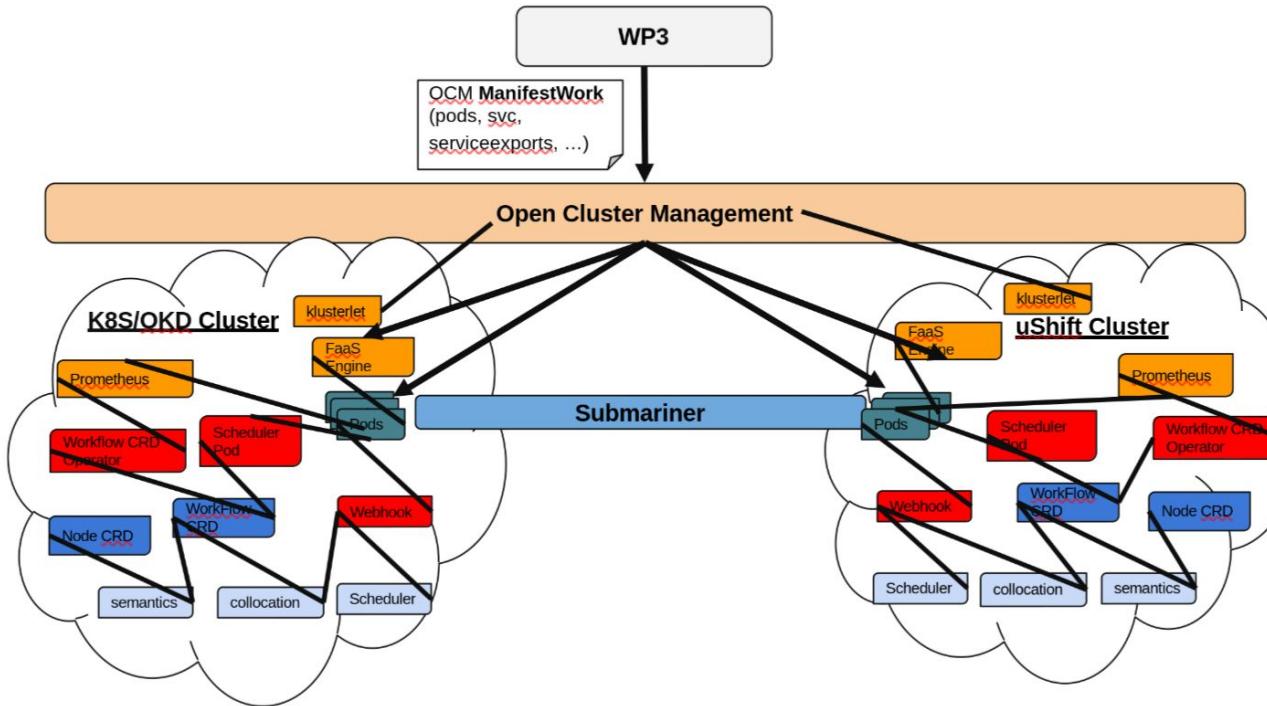


Optimization of resource **selection** and operation (**global** and **local** level)



Multiple Exploitation Channels and **Reusable Artefacts**

Architecture / Baseline technologies



Sunday, June 18 • 2:45pm - 3:20pm



Design & Deployment of FaaS apps at the edge-cloud

Click here to remove from My Schedule.

 <https://sched.co/1Mmi>

 Tweet

 Share

The talk will focus on a design and development environment coming from the H2020 PHYSICS project, that aims to ease application evolution to the new FaaS model. It uses the Node-RED open source tool as the main function and workflow runtime. The goal of the environment is to enable a more user friendly and abstract function and workflow creation process for complex FaaS applications. To this end, it provides an extendable, pattern-enriched palette of ready-made, reusable functionalities such as workload parallelization, data collection at the edge, function orchestration creation among others. The environment embeds seamless DevOps processes for generating the deployable artefacts of the FaaS platform (Openwhisk). Annotation mechanisms are also available for the developer to dictate diverse execution options towards the deployment stacks, including sizing and locality considerations, as well as abilities for dynamic FaaS applications to continuously leverage the edge-cloud continuum.



Saturday, June 17 • 5:00pm - 5:35pm



Towards Container-layer-aware Scheduling Policies

Click here to remove from My Schedule.

 <https://sched.co/1MYe>

 Tweet

 Share

Serverless has been gaining popularity as a new way to program and deploy applications on clouds. Function as a service (FaaS) is an approach encompassed by serverless, extending the FaaS concept by avoiding server infrastructure management.

In this context, functions rely on containers, and deploying new containers can cause several overheads to the platforms and the function's execution (cold start delays).

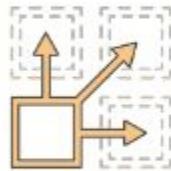
Kubernetes-based platforms are used for serverless proposes, and K8S provides an ImageLocality mechanism to address it, but it relies on entire warm containers and not on layers.

Therefore, we propose and implement on K8S two new scheduling policies. The first is a ContainerLayer-Aware policy that optimizes function's placements by selecting machines with the biggest rate of container layers that can be shared. The second is a Multi-Objective policy for heterogeneous platforms that reduces at the same time the makespan and the data transferred by functions I/O and container layers.

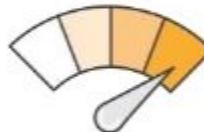
Knative

Serverless architectures

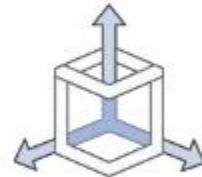
Removes the need for....



Provisioning
and Utilization



Operations
and Management



Scaling



Availability and
Fault Tolerance

The Core Serverless Traits



FaaS in the cloud



IBM Cloud Functions

Azure Functions



Google Cloud
Functions

OpenShift Serverless

FaaS in the K8s



OPENFAAS



Kubeless



Apache OpenWhisk

v0.0.0.0.0.0



Knative

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

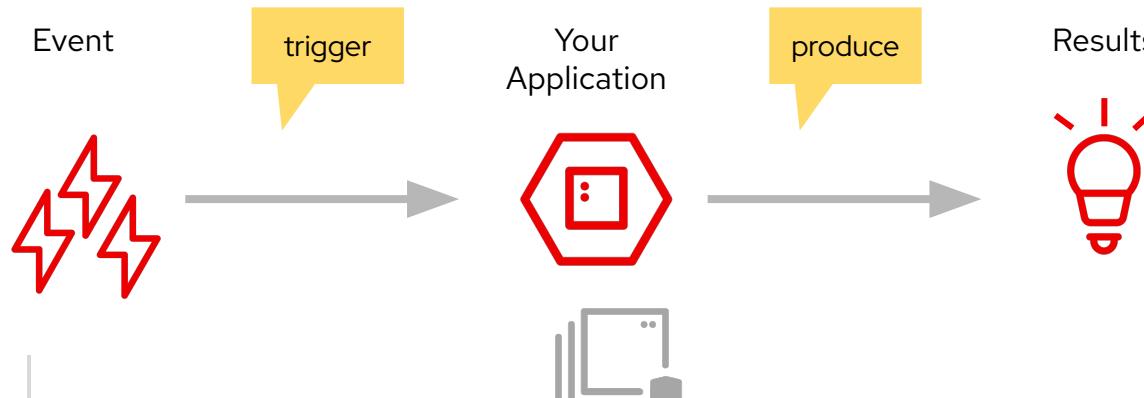
A **programming** model that lets you focus on **just your code** for faster iterations.



CLIENT (kn)

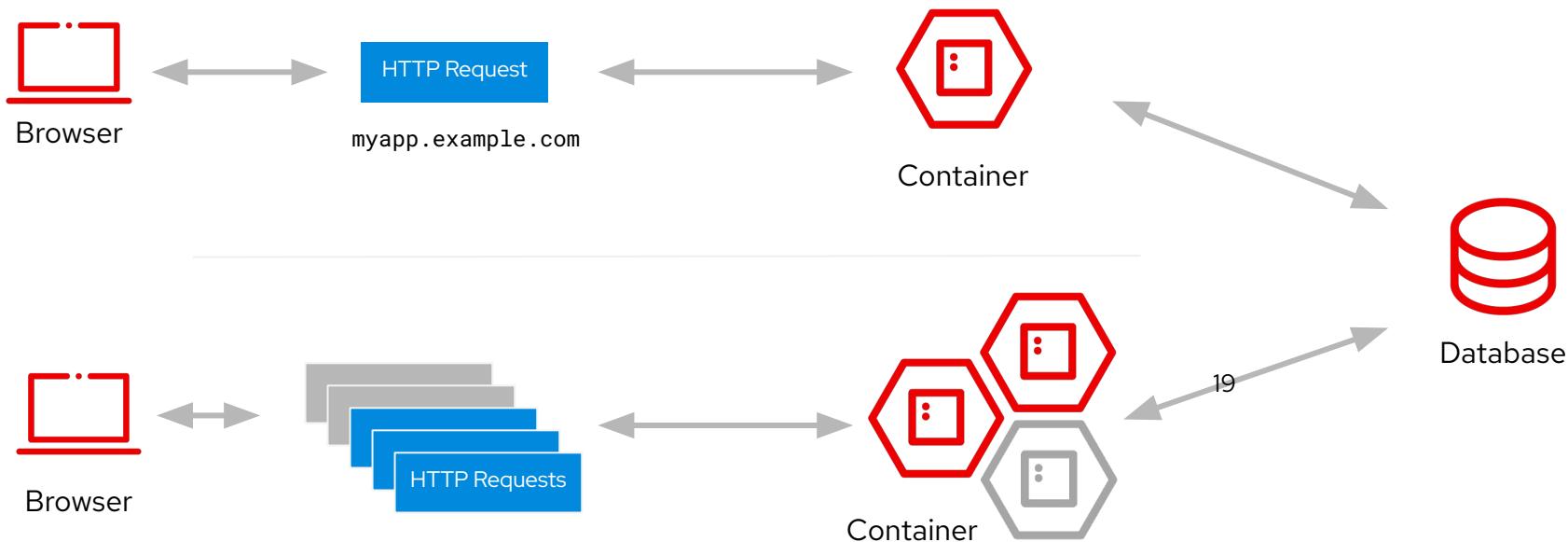
Allows you to create resources interactively from the **command line** or from within scripts

The "Serverless Pattern"



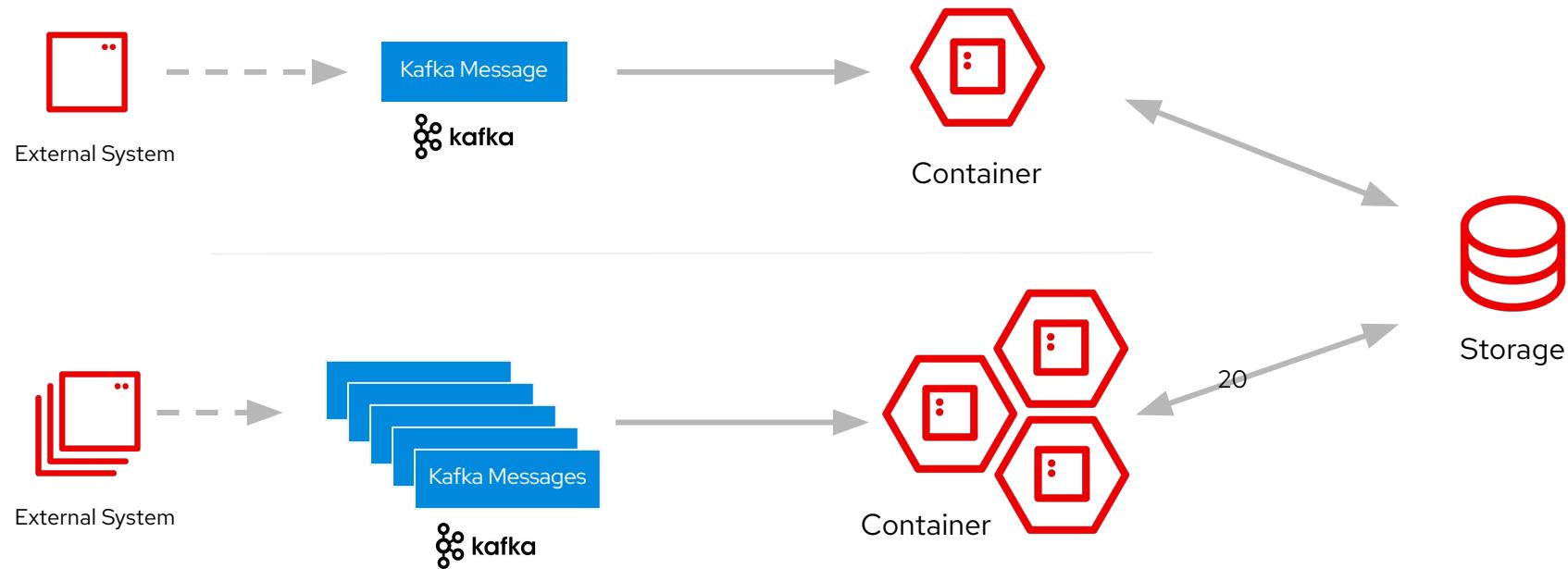
The "Serverless Pattern"

A serverless web application



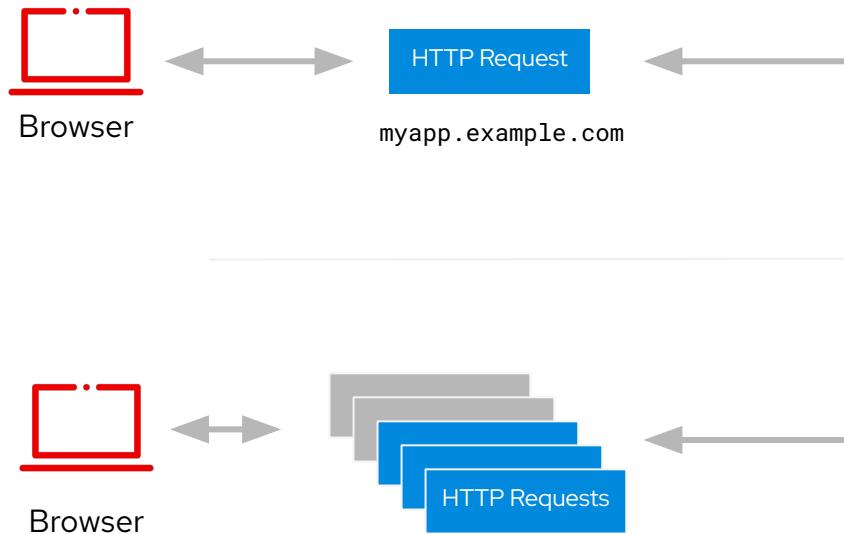
The "Serverless Pattern"

Processing a Kafka message



The "Serverless Pattern"

A serverless web application

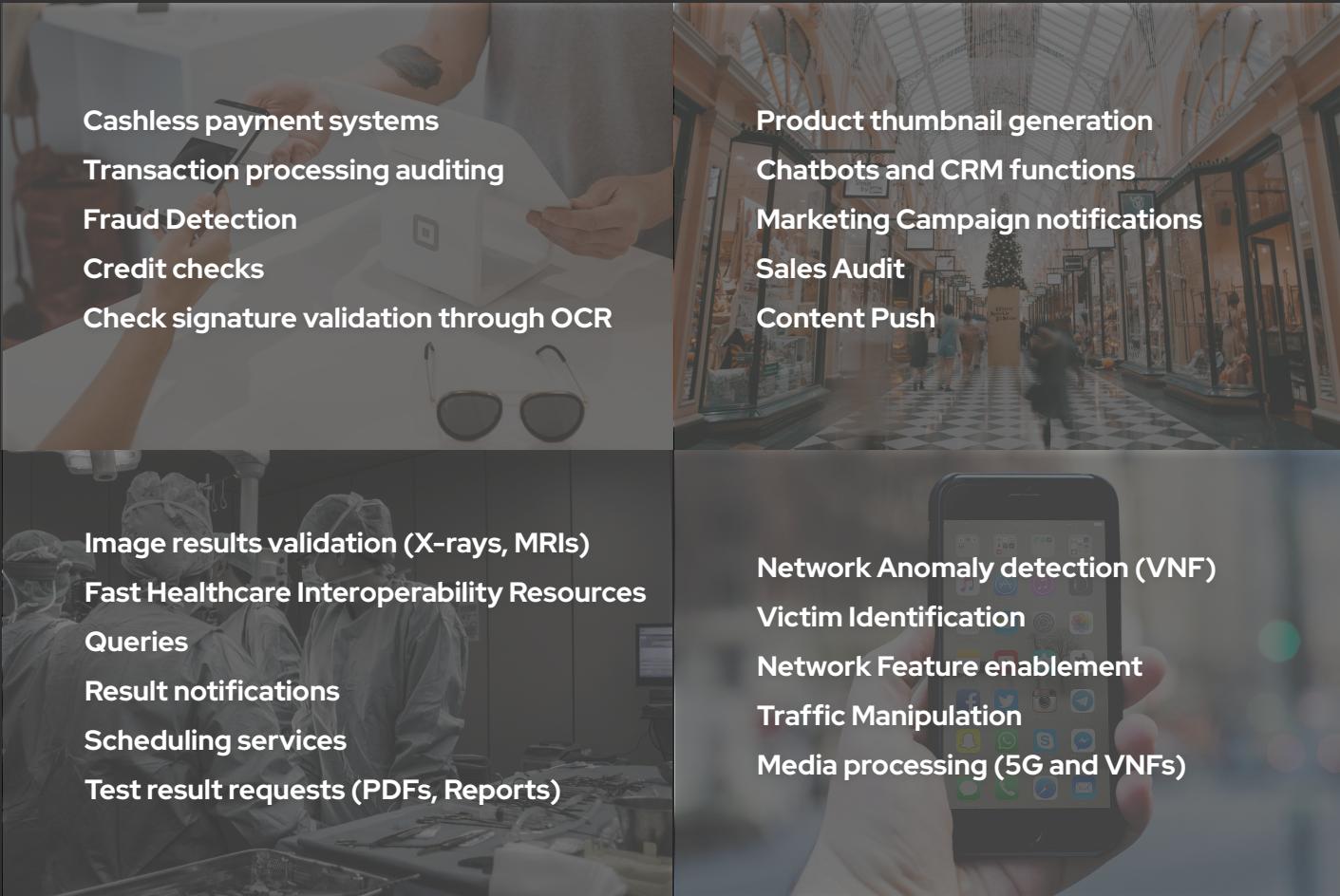


Benefits of this model:

- No need to setup auto-scaling and load balancers
 - Scale down and save resources when needed.
 - Scale up to meet the demand.
- No tickets to configure SSL for applications
- Enable Event Driven Architectures (EDA) patterns
- Enable teams to associate cost with IT
- Modernize existing applications to run as serverless containers

Where Serverless?

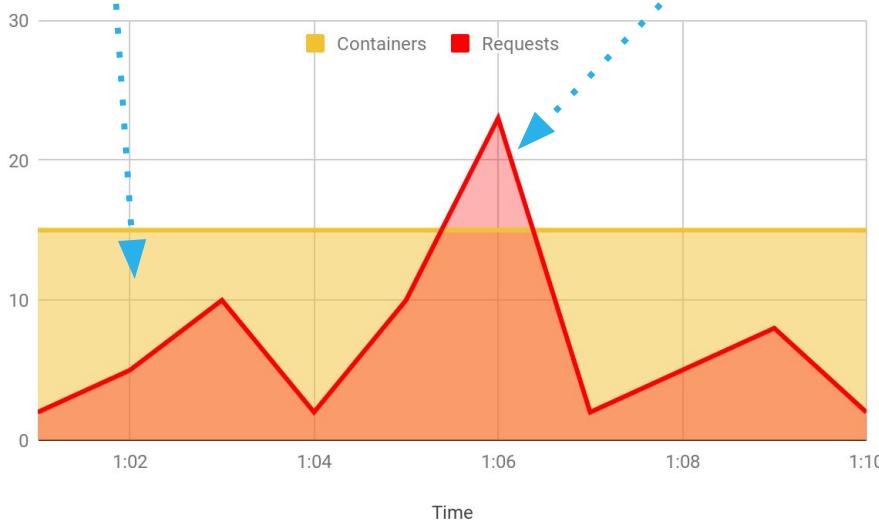
- Application with unpredictable or bursty number of requests.
- Maximum resource utilization to reduce the carbon footprint
- Building event-driven, loosely coupled systems
- Low barrier for Developers (Kubernetes is hard)
- A/B testing or canary deployments
- Seasonal or periodic workload.
- microservices or containers and want to leverage serverless



Serverless Operational Benefits

Over provisioning

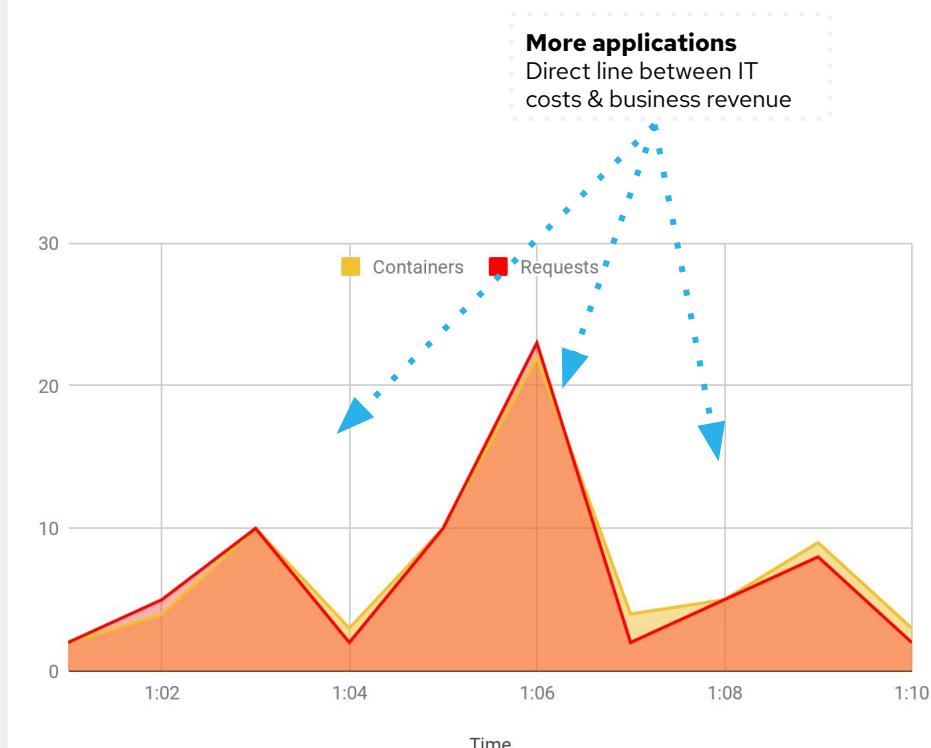
Time in capacity planning
IT cost of idle resources



NOT Serverless

Under provisioning

Lost business revenue
Poor quality of service

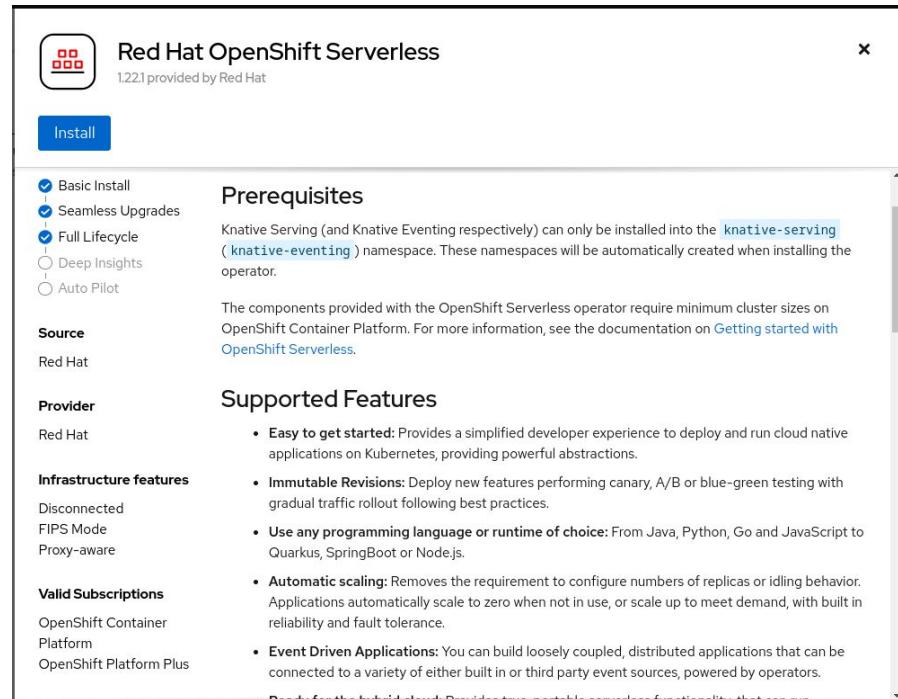


with Serverless

Installation experience

"Easy day 1 and even better for day 2"

- Click Install experience
- Developer & admin experience in Console
- Built-in event sources
- No external dependencies.
- **"Just works."**



The screenshot shows the Red Hat OpenShift Serverless Operator's "Install" page. At the top, there's a logo and the text "Red Hat OpenShift Serverless 1.22.1 provided by Red Hat". Below that is a large blue "Install" button. To the left of the main content area, there are several configuration sections:

- Basic Install**: Includes checkboxes for "Basic Install", "Seamless Upgrades", and "Full Lifecycle". There are also radio buttons for "Deep Insights" and "Auto Pilot".
- Source**: Set to "Red Hat".
- Provider**: Set to "Red Hat".
- Infrastructure features**: Includes "Disconnected", "FIPS Mode", and "Proxy-aware".
- Valid Subscriptions**: Includes "OpenShift Container Platform" and "OpenShift Platform Plus".

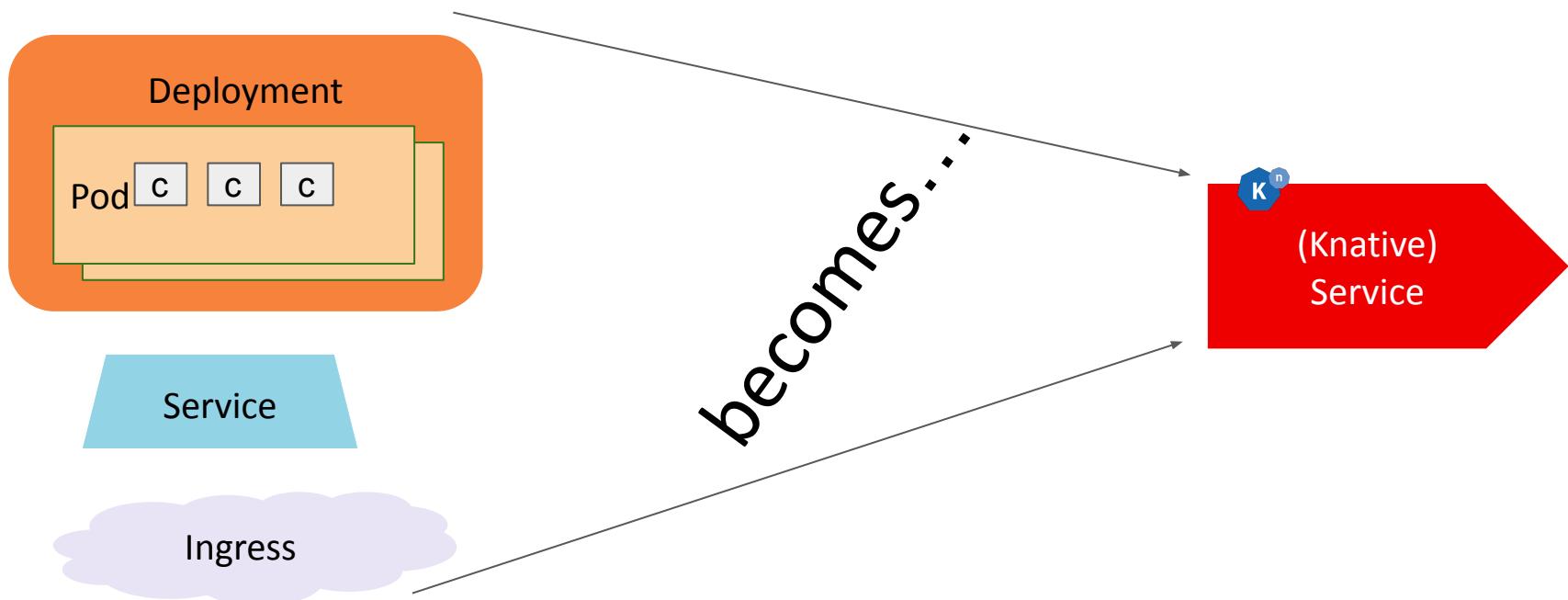
Prerequisites: A note states that Knative Serving (and Knative Eventing respectively) can only be installed into the `knative-serving` (`knative-eventing`) namespace. These namespaces will be automatically created when installing the operator.

Supported Features:

- Easy to get started**: Provides a simplified developer experience to deploy and run cloud native applications on Kubernetes, providing powerful abstractions.
- Immutable Revisions**: Deploy new features performing canary, A/B or blue-green testing with gradual traffic rollout following best practices.
- Use any programming language or runtime of choice**: From Java, Python, Go and JavaScript to Quarkus, SpringBoot or Node.js.
- Automatic scaling**: Removes the requirement to configure numbers of replicas or idling behavior. Applications automatically scale to zero when not in use, or scale up to meet demand, with built in reliability and fault tolerance.
- Event Driven Applications**: You can build loosely coupled, distributed applications that can be connected to a variety of either built in or third party event sources, powered by operators.
- Ready for the hybrid cloud**: Provides true portable serverless functionality that can run

Simplify application development/deployment on K8

Reduce developer toil and cognitive overhead with **Knative tools**



Kubernetes

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: guestbook
spec:
  selector:
    matchLabels:
      app: guestbook
      tier: frontend
  replicas: 1
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
        - image: markusthoemmes/guestbook
          name: guestbook
          resources:
            requests:
              cpu: 100m
              memory: 100Mi
          env:
            - name: GET_HOSTS_FROM
              value: dns
          ports:
            - containerPort: 80
```

~70 lines

```
apiVersion: extensions/v1beta1
kind: HorizontalPodAutoscaler
metadata:
  name: guestbook
  namespace: default
spec:
  scaleRef:
    kind: ReplicationController
    name: guestbook
    namespace: default
    subresource: scale
  minReplicas: 1
  maxReplicas: 10
  cpuUtilization:
    targetPercentage: 50
  ---
  apiVersion: v1
  kind: Service
  metadata:
    name: frontend-service
  labels:
    app: guestbook
    tier: frontend
  spec:
    ports:
      - port: 80
        selector:
          app: guestbook
          tier: frontend
  ---
  apiVersion: route.openshift.io/v1
  kind: Route
  metadata:
    name: frontend-route
  spec:
    to:
      kind: Service
      name: frontend-service
```

CONFIDENTIAL



Knative

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: frontend
spec:
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
        - image: markusthoemmes/guestbook
          resources:
            requests:
              cpu: 100m
              memory: 100Mi
          env:
            - name: GET_HOSTS_FROM
              value: dns
          ports:
            - containerPort: 80
```

22 lines

voooooooo

Knative Functions

Powerful Developer experience

- ✓ Local Developer Experience
- ✓ IDE Developer Experience
- ✓ Offers multiple build strategies
- ✓ Deploy as Knative Service
- ✓ Project templates
- ✓ Support for Cloud Events/HTTP
- ✓ On cluster build using Tekton/Pipelines
- ✓ **Runtimes:**



```
$ kn func help
Usage:
  func [command]

Available Commands:
  build      Build a Function project as a container image
  completion Generate completion scripts for bash, fish and zsh
  config     Configure the Function
  create     Create a Function project
  delete     Undeploy a Function
  deploy     Deploy a Function
  help       Help about any command
  info       Show details of a Function
  invoke    Invoke a Function
  list      List Functions
  repository Manage installed template repositories
  run        Run the Function locally
  version   Show the version
```

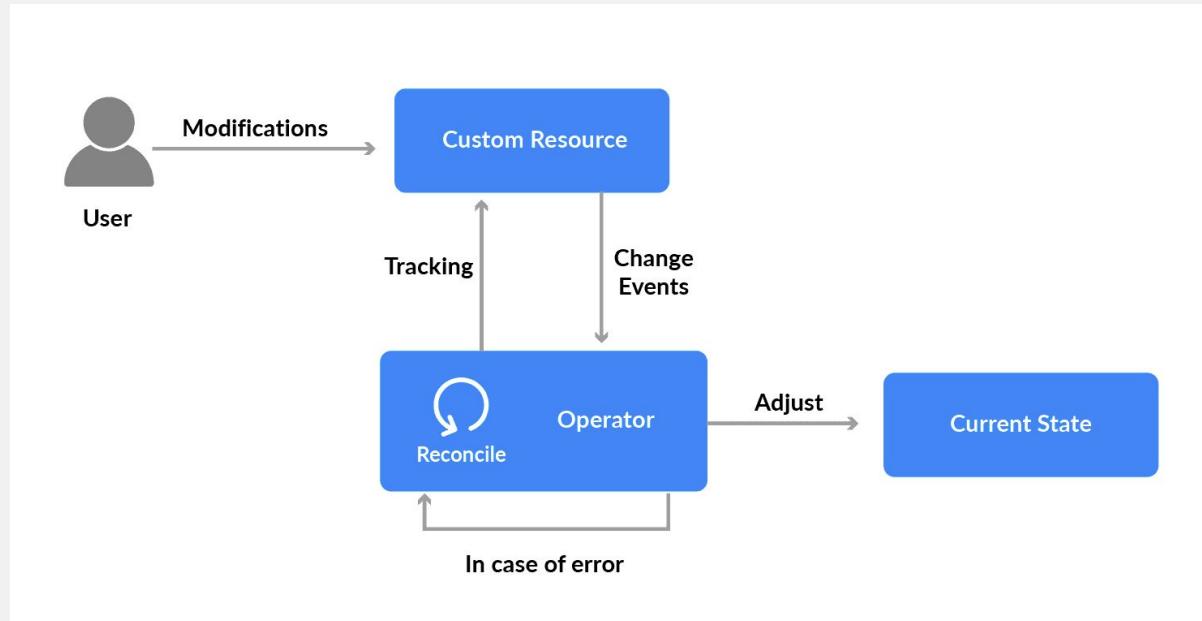


Operator pattern

Example: Kafka operator

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  labels:
    app: my-cluster
  name: my-cluster
  namespace: myproject
spec:
  # ...
  kafka:
    replicas: 3
    # ...
```

Operator pattern



Kubernetes Operators

Operators simplify management of complex applications
on Kubernetes

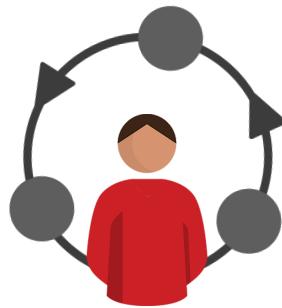
- Encode human operational knowledge
- **Automatically patch, upgrade, recover, and tune container-based apps and services**
- Kubernetes-native
- Purpose-built for a specific application or service
- Enable “day 2” management



OPERATOR
SDK



Encoding and automating Ops knowledge



WITHOUT OPERATORS: REACTIVE

- Continually checks for anomalies
- Alert humans for response
- Requires manual change to fix



WITH OPERATORS: PROACTIVE

- Continually adjusts to optimal state
- Automatically acts in milliseconds



Choosing the right tool

HELM

Implementation is declarative and simple

Operator functionality is limited to Helm features

Operator manifest bundle files (CRD, RBAC, Operator Deployment) are automatically generated

ANSIBLE

Implementation is declarative and human-readable

Ansible can express almost any operator functionality

Operator manifest bundle files (CRD, RBAC, Operator Deployment) are automatically generated

GO

Implementation is imperative and more complex

There is no limit on the functionality you want to implement

Operator manifest bundle files (CRD, RBAC, Operator Deployment) are generated from the Go source code



Hands on

Lab description

Goal:

- Create a golang operator that deploys an existing knative function
 - The Knative function has previously been built and pushed to a private registry
- Extend the operator to build, push and deploy a function located on a github repository

Example CRD:

```
apiVersion: knf.example.com/v1alpha1
kind: KnativeFunction
metadata:
  name: knativefunction-sample
spec:
  name: test-function
  image: localhost:50000/kn-user/test-hw@sha256:79c456
  maxscale: "2"
  minscale: "1"
  concurrency: 1
```

Example output:

```
$ kubectl get knativefunction
NAME          AGE
knativefunction-sample  5m8s

$ kubectl get ksvc
NAME      URL
test-function  http://test-function.default.127.0.0.1.sslip.io

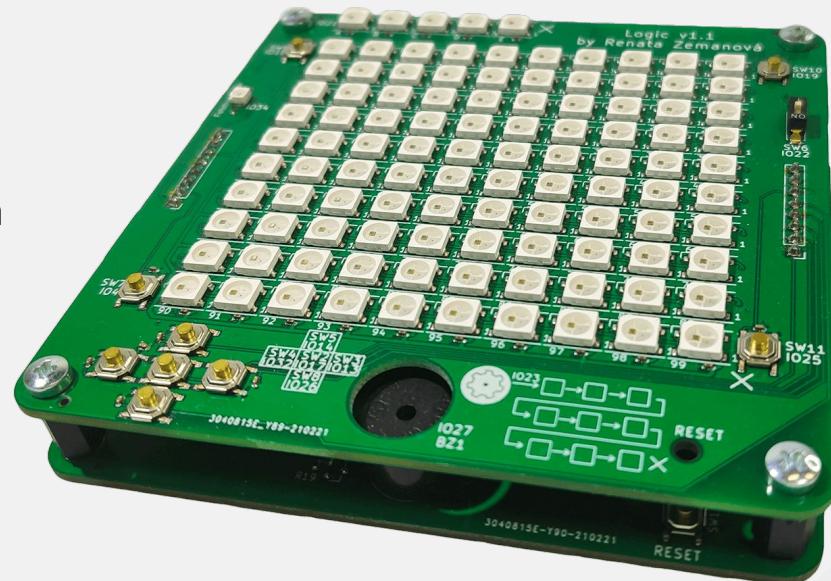
$ curl http://test-function.default.127.0.0.1.sslip.io
DevConf.cz 2023!
```



Prizes

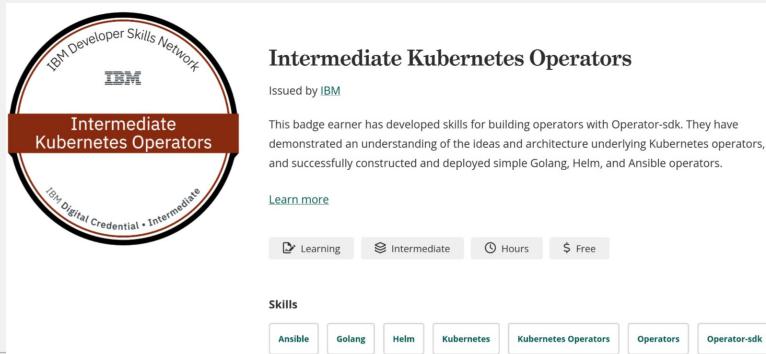
3 Logic | Robotárna boards - [github](#)

- Universal programmable toy designed for teaching programming
- Resembles a game console
- It has 100 RGB LEDs that can serve as a display, it has numerous buttons and a buzzer
- Powered by ESP32 microcontroller
- Kids can create custom games and learn programming while doing so
- Possible to run multiplayer games as the on-board processor features both, WiFi and Bluetooth 4



Useful links

- Knative documentation: <https://knative.dev/docs/>
- Knative.dev/client Golang API: <https://pkg.go.dev/knative.dev/client>
- Operator SDK - Go Operator tutorial:
<https://sdk.operatorframework.io/docs/building-operators/golang/tutorial/>
- Example memcached operator:
<https://github.com/operator-framework/operator-sdk/tree/master/testdata/go/v3/memcached-operator>
- Intermediate Kubernetes Operators on IBM Developer Skills Network:
<https://courses.course-dev.skills.network/courses/course-v1:IBMSkillsNetwork+CO0201EN+2021T1/course/>



Thank you

Red Hat is the world's leading provider of enterprise open source software solutions.

Award-winning support, training, and consulting services make

Red Hat a trusted adviser to the Fortune 500.



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



twitter.com/RedHat