



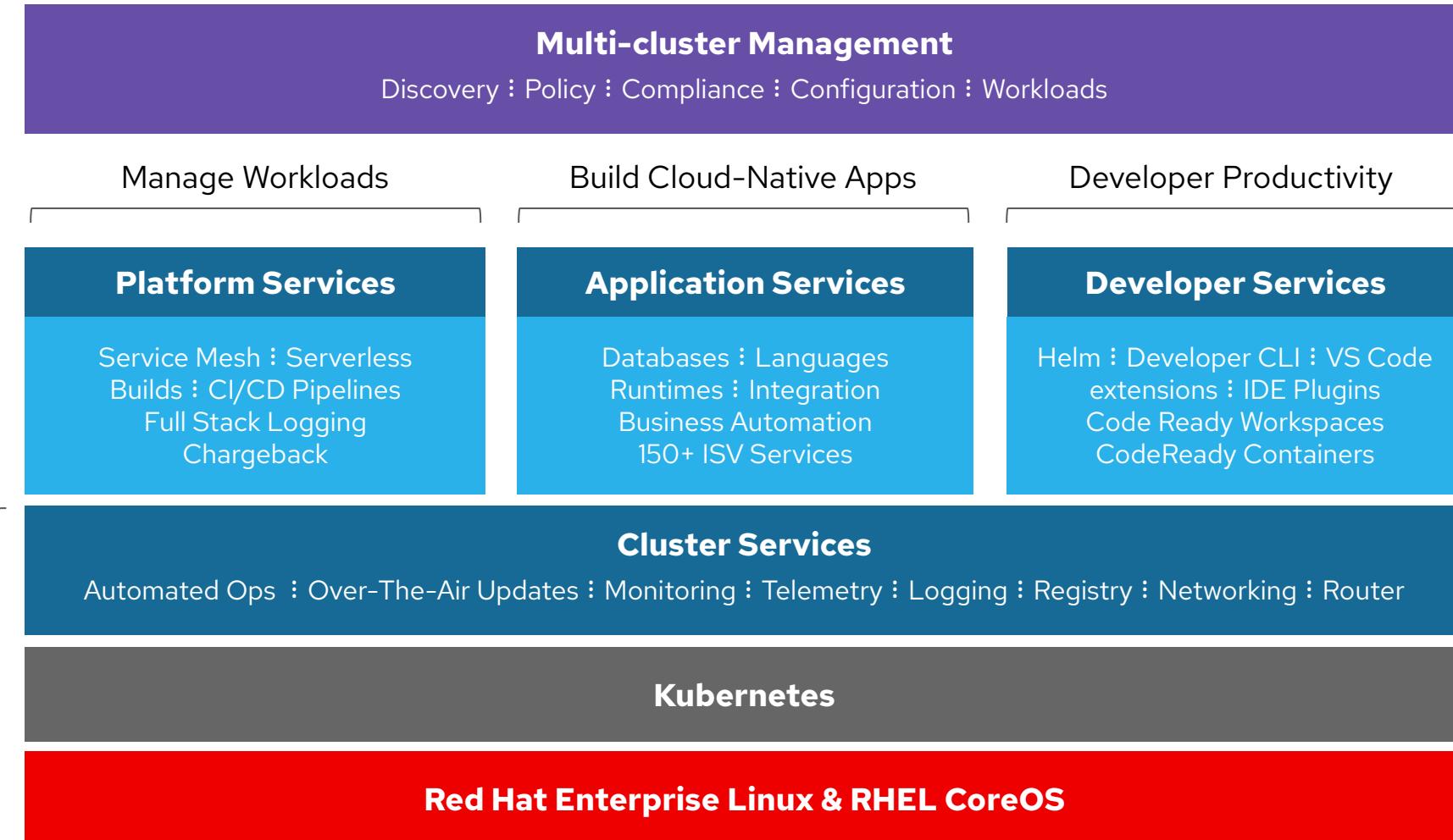
# OpenShift Serverless

Alfred Bach  
[abach@redhat.com](mailto:abach@redhat.com)

Updated: April 2020



# OpenShift Container Platform



Physical



Virtual



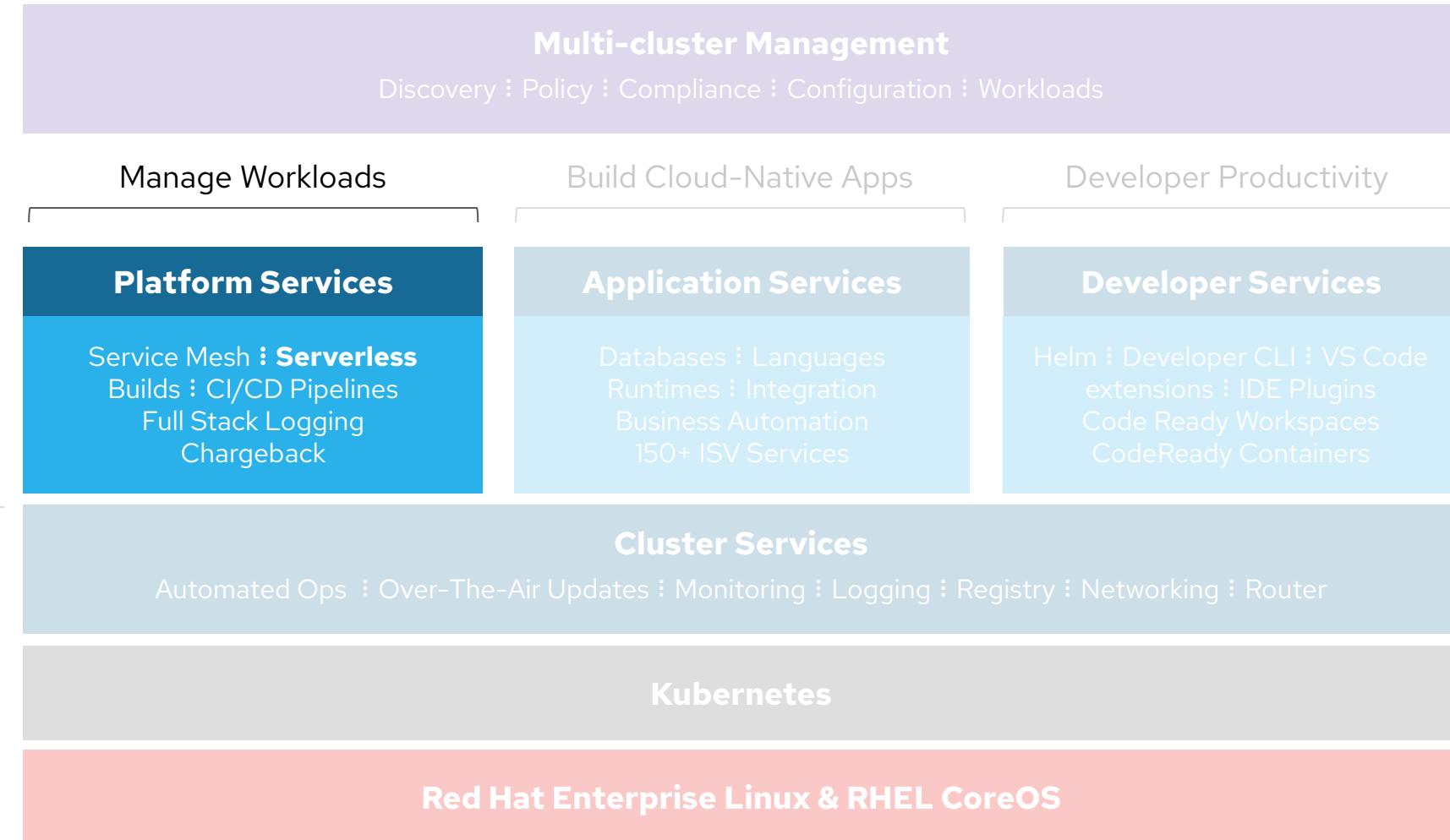
Private cloud



Public cloud



# OpenShift Container Platform



# Serverless Market Trends

*"Use Serverless To optimize The Benefits of The cloud"<sup>2</sup>*

40%

of enterprises adopted Serverless technologies or practices with expected growth coming in the next 12 to 18 months.<sup>1</sup>

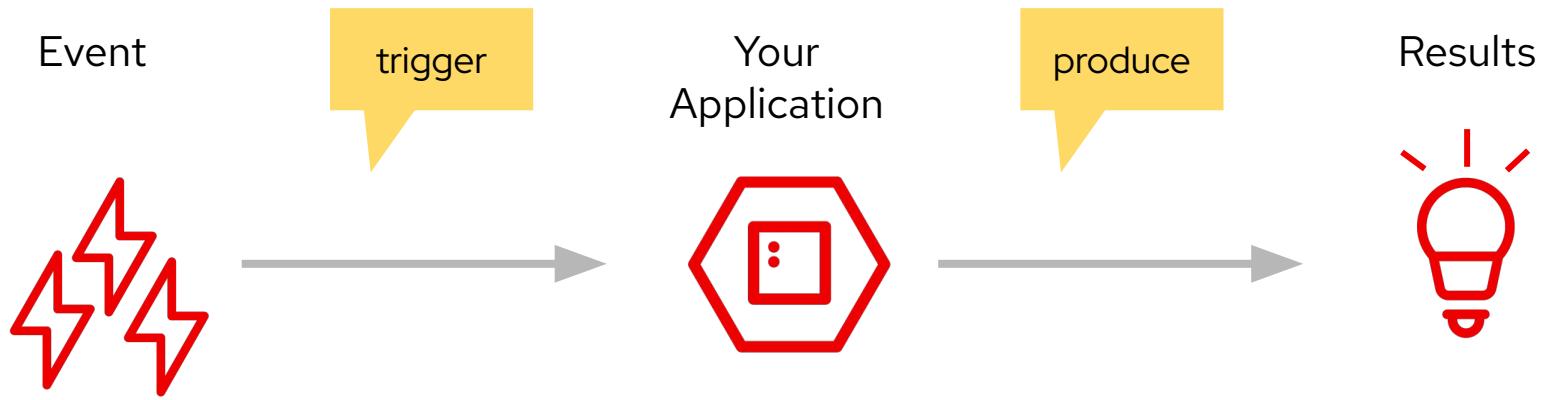


**Vendor lock-in** is the second biggest concern when adopting Serverless technologies.<sup>1</sup>

60%

of the serverless practitioners reported "*reduction of operational costs*" with the second biggest benefit being "*scale with demand automatically*"

# The "Serverless Pattern"



HTTP Requests

Kafka Messages

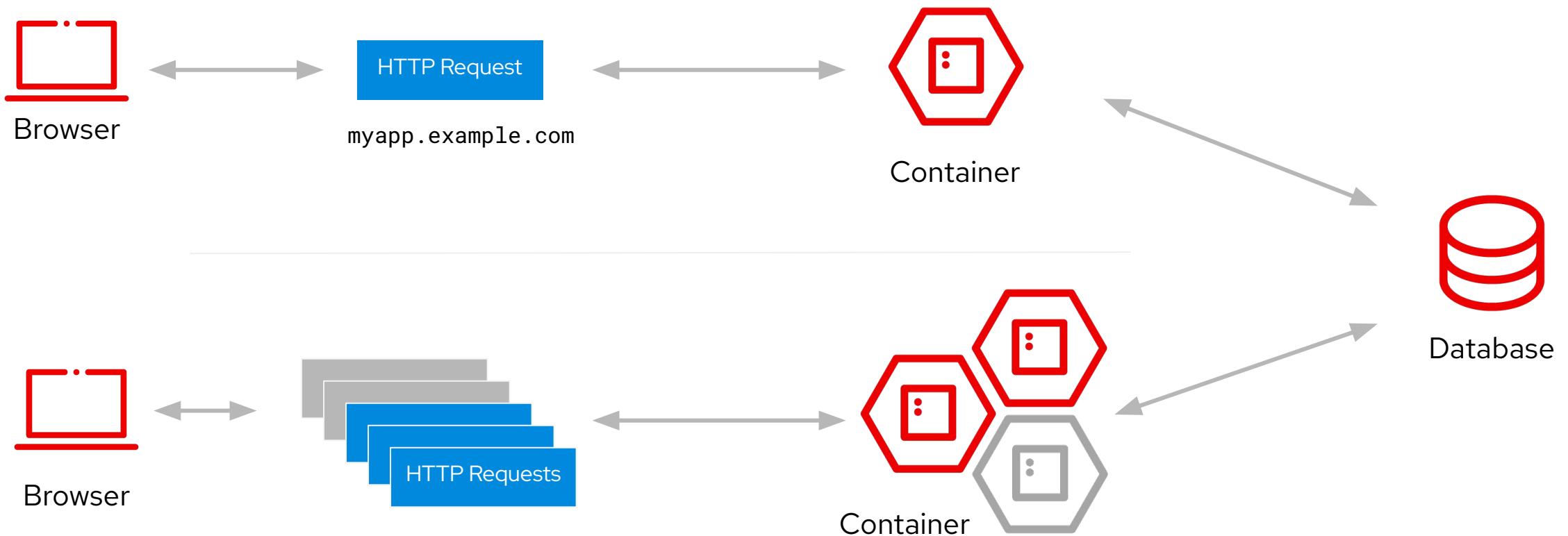
Image Uploaded

New Order

Login from user

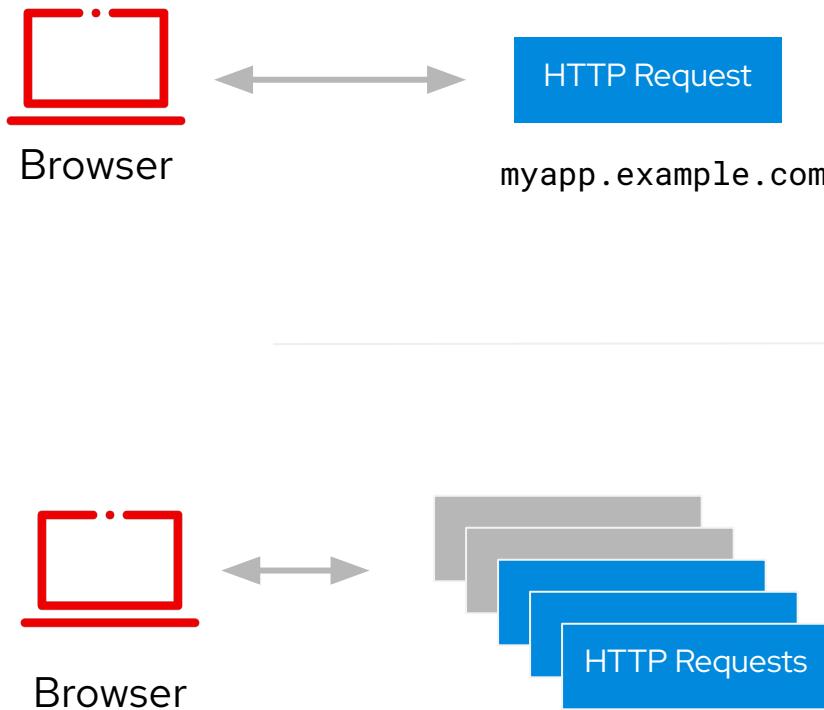
# The "Serverless Pattern"

A serverless web application



# The "Serverless Pattern"

A serverless web application

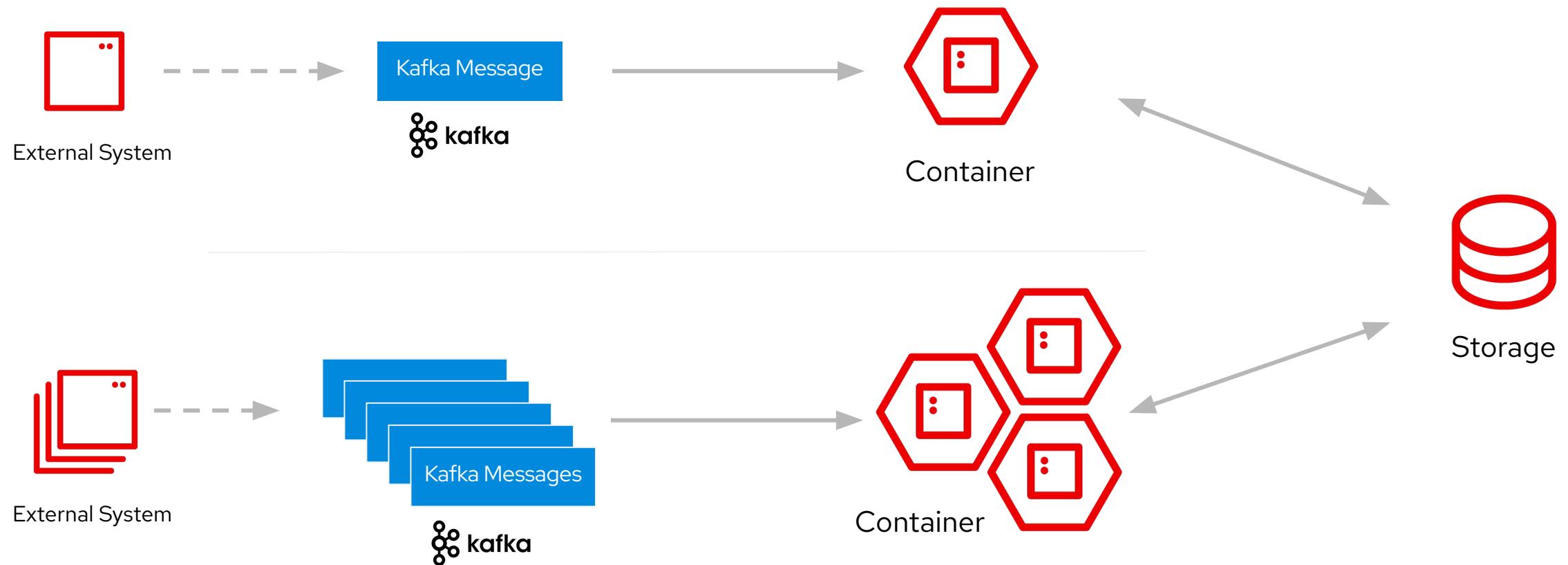


## Benefits of this model:

- 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

# The "Serverless Pattern"

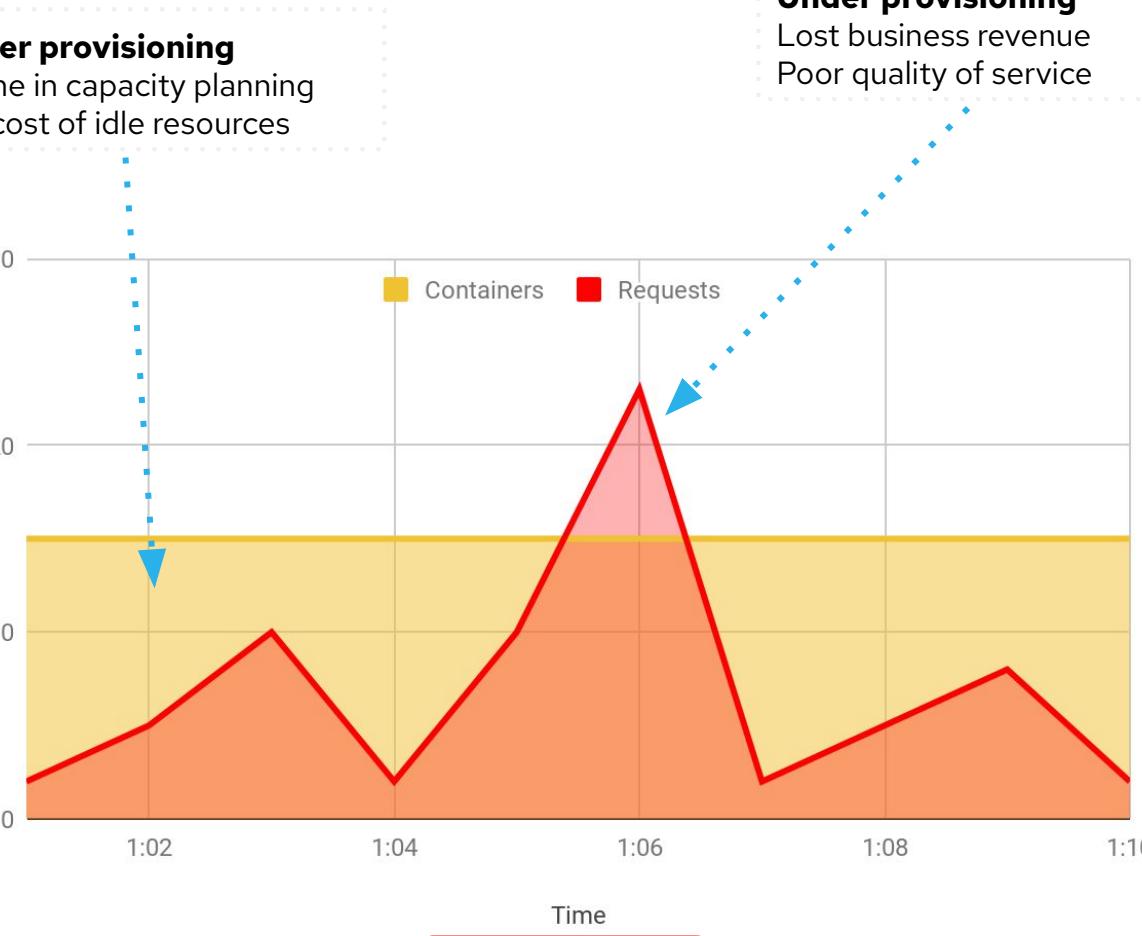
Processing a Kafka message



# Serverless Operational Benefits

## Over provisioning

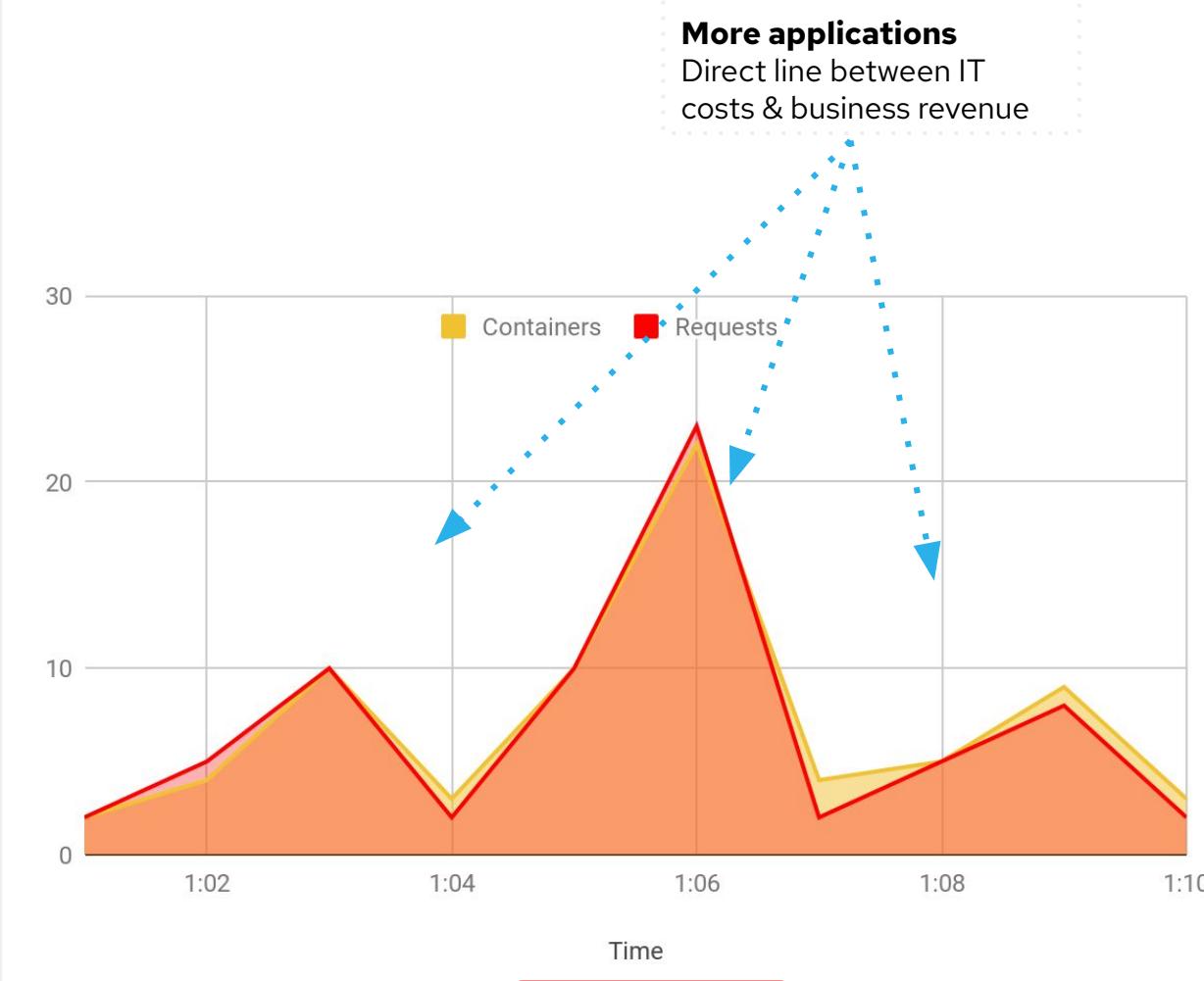
Time in capacity planning  
IT cost of idle resources



NOT Serverless

## Under provisioning

Lost business revenue  
Poor quality of service



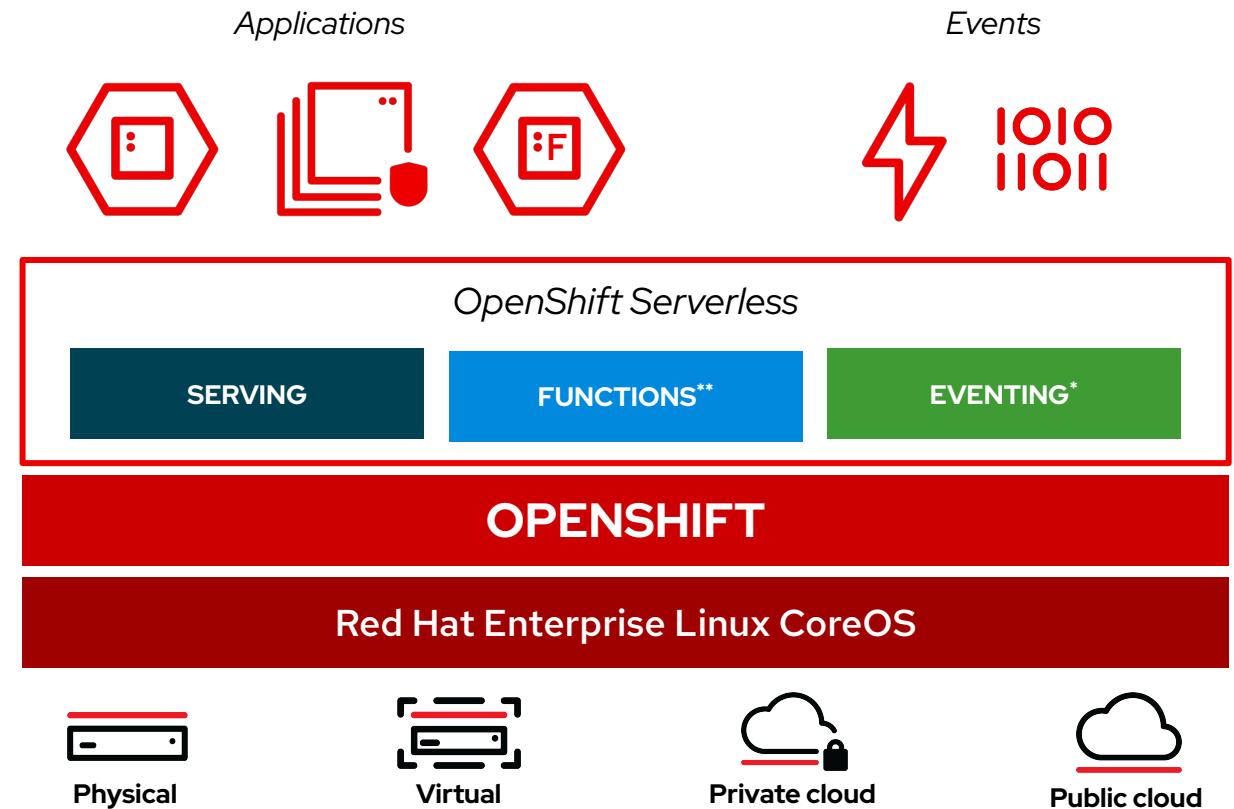
with Serverless



# OpenShift Serverless

Event-driven serverless containers and functions

- Deploy and run **serverless containers**
- Use any programming language or runtime
- Modernize existing applications to run serverless
- Powered by a rich ecosystem of event sources
- Manage serverless apps natively in Kubernetes
- Based on open source project **Knative**
- Run anywhere OpenShift runs



\* Eventing is currently in Technology Preview

\*\* Functions are currently a work in progress initiative

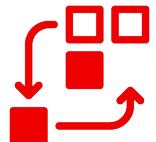
# OpenShift Serverless

## Key Features



### Containers made easy

Simplified developer experience to deploy applications/code on serverless containers abstracting infrastructure & focusing on what matters.



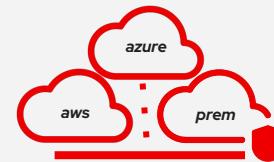
### Immutable revisions

Deploy new features: performing canary, A/B or blue-green testing with gradual traffic rollout with no sweat and following best practices.



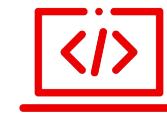
### Automatic scaling

No need to configure number of replicas, or idling. Scale to zero when not in use, auto scale to thousands during peak, with built-in reliability and fault-tolerance.



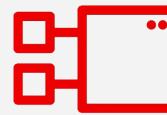
### Ready for the Hybrid Cloud

Truly portable serverless running anywhere OpenShift runs, that is on-premises or on any public cloud. Leverage data locality and SaaS when needed.



### Any programming language

Use any programming language or runtime of choice. From Java, Python, Go and JavaScript to Quarkus, SpringBoot or Node.js.



### Event Driven Architectures

Build loosely coupled & distributed apps connecting with a variety of built-in or third-party event sources or connectors powered by Operators.

# 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."**

 OpenShift Serverless Operator

1.7.0 provided by Red Hat, Inc.

[Install](#)

OPERATOR VERSION  
1.7.0

PROVIDER TYPE  
Red Hat

PROVIDER  
Red Hat, Inc.

The Red Hat OpenShift Serverless operator provides a collection of APIs that enables containers, microservices and functions to run "serverless". Serverless applications can scale up and down (to zero) on demand and be triggered by a number of event sources. OpenShift Serverless integrates with a number of platform services, such as Metering and Monitoring and it is based on the open source project Knative.

# Choosing the Right Tool

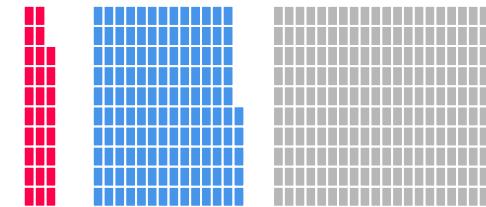
Your Team



The Ecosystem



Performance

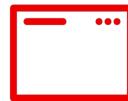




# Developer experience

- </> Java
- </> Node
- </> Python
- </> Go
- </> Ruby
- </> Ruby on Rails
- </> PHP
- </> Perl
- </> .NET Core

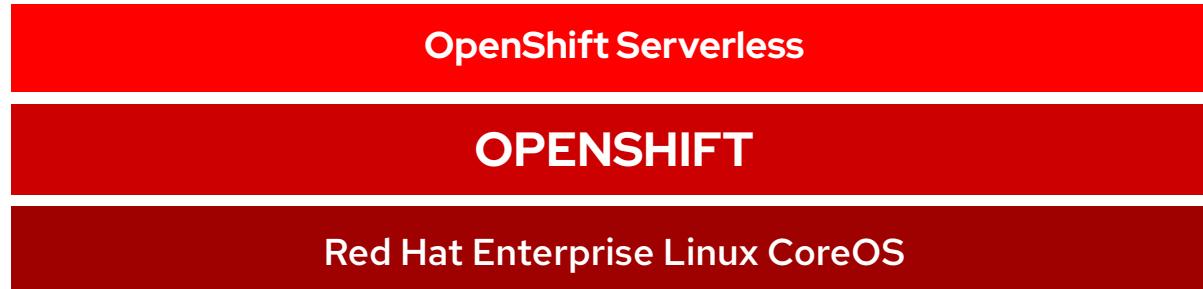
```
$ kn service create --image=
```



CLI



UI



Physical



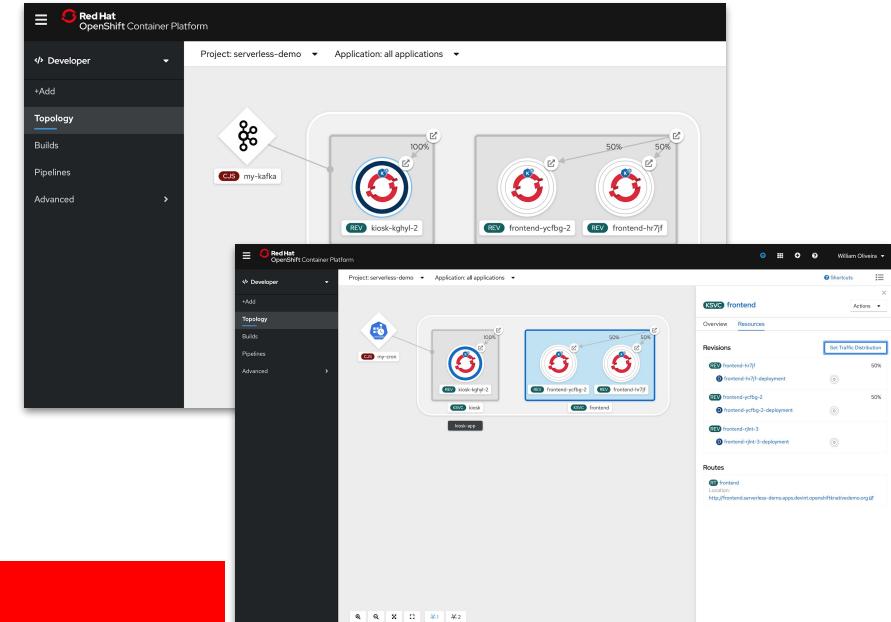
Virtual



Private cloud



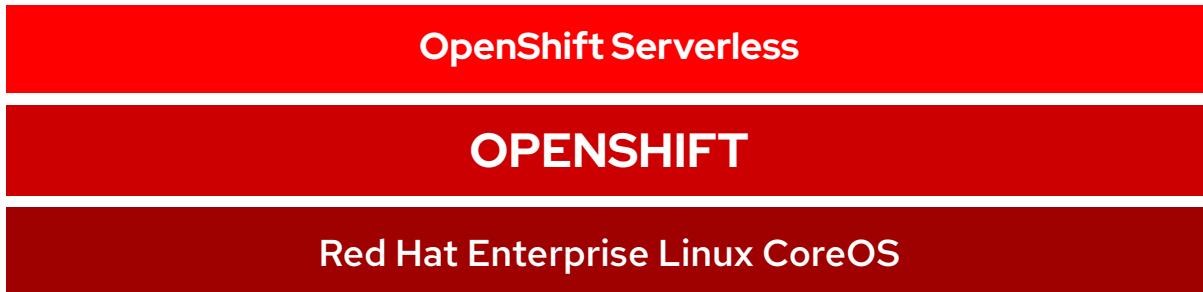
Public cloud





# Admin experience

- Monitoring, Metering and Logging
- Disconnected install support (air-gapped)
- Egress proxy with TLS support
- Over the air updates and patches



Physical



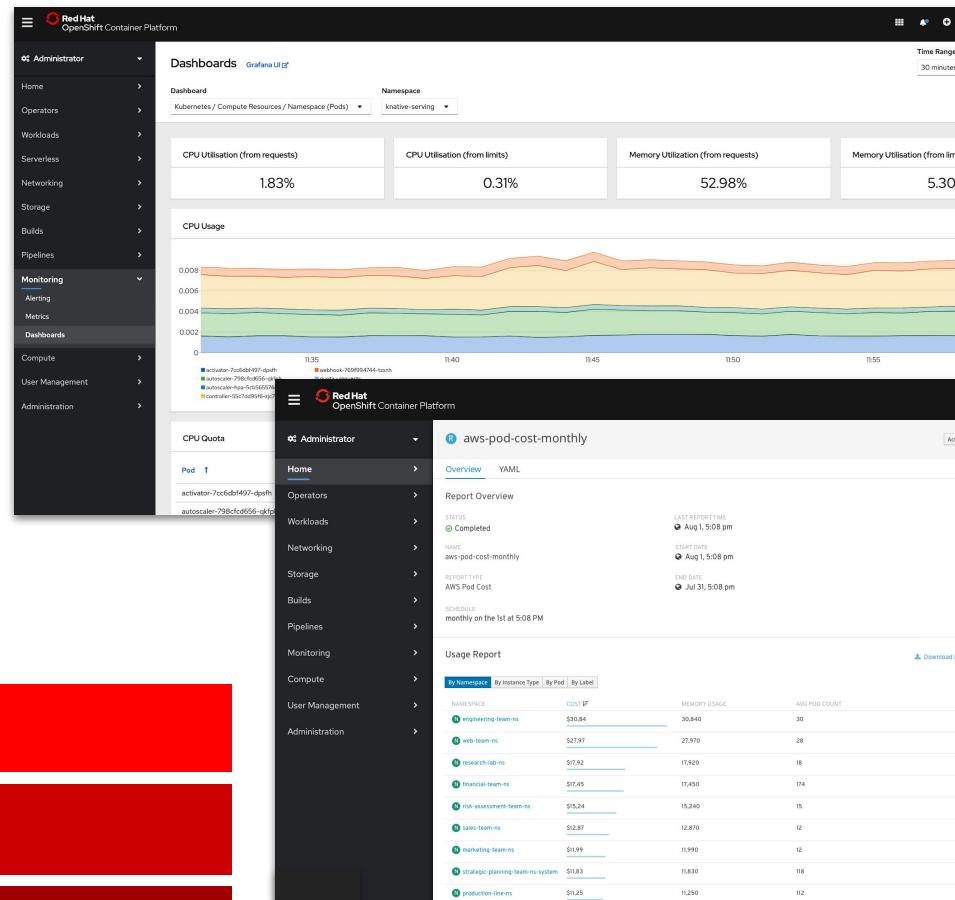
Virtual



Private cloud



Public cloud



# 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
```

# 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





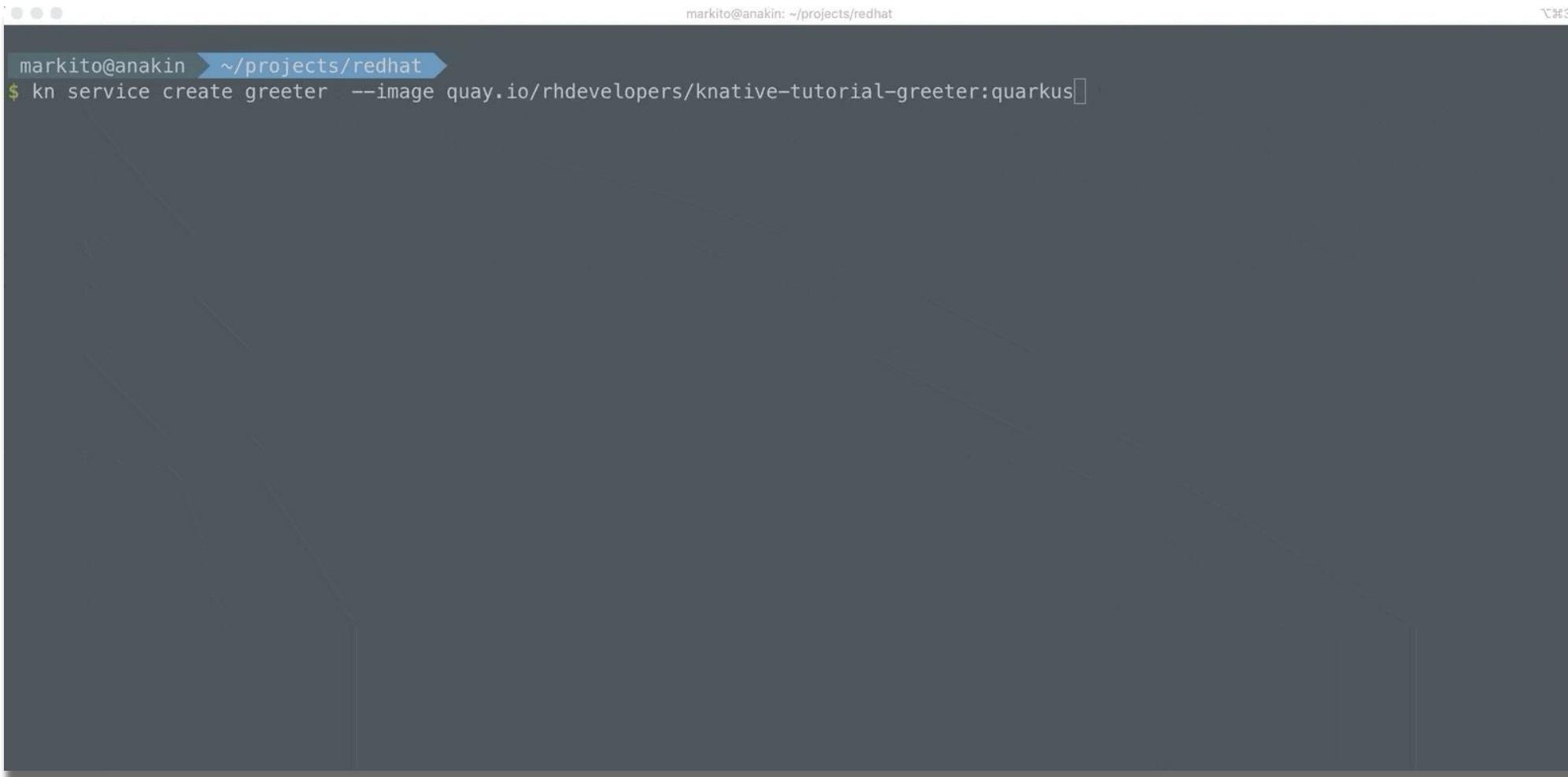
# Command Line Experience

```
$ kn service create myService --image=[registry/mycontainer:v1] --min-scale=1 --max-scale=100
```

```
$ kn service update myService --traffic myService-rev1=50,myService-rev2=50
```

```
$ kn source cronjob create my-cron --schedule "* * * * */1" --data "ping" --sink svc:myService
```

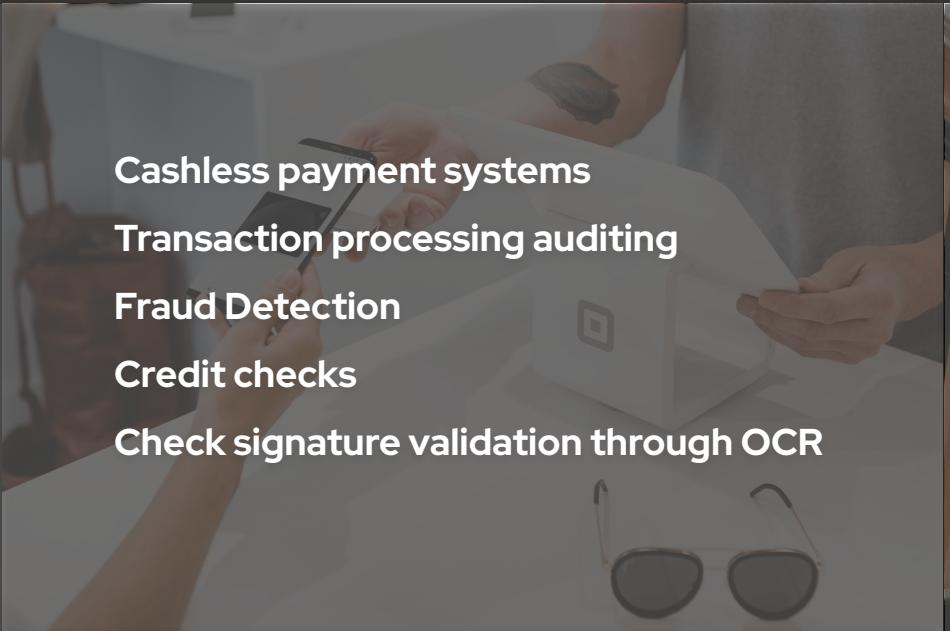
# Hello World with Quarkus!



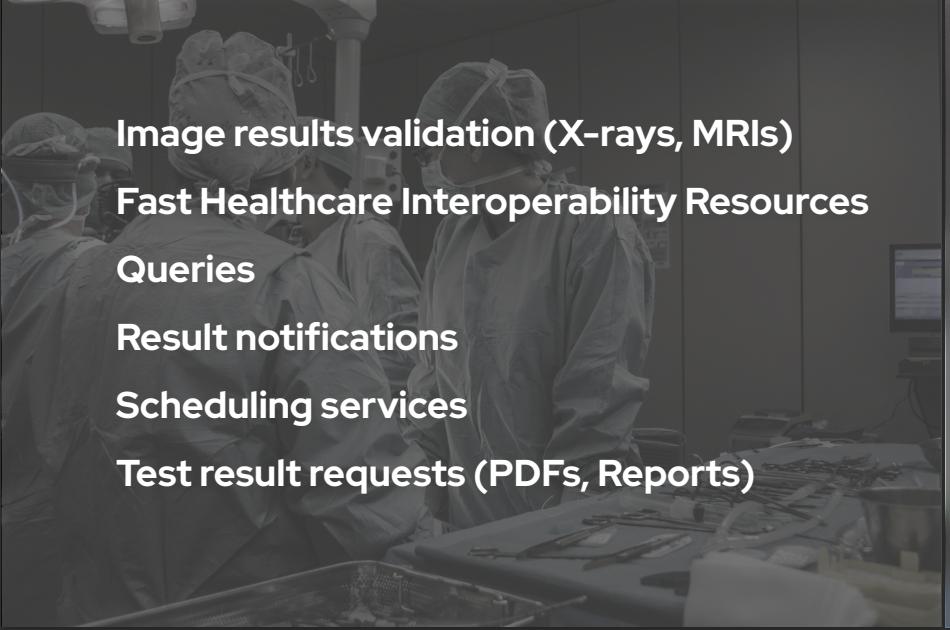
A screenshot of a terminal window titled "markito@anakin: ~/projects/redhat". The window shows the command \$ kn service create greeter --image quay.io/rhdevelopers/knative-tutorial-greeter:quarkus being typed into the terminal. The background of the slide features a faint, large watermark of a house.

```
markito@anakin ~]$ kn service create greeter --image quay.io/rhdevelopers/knative-tutorial-greeter:quarkus
```

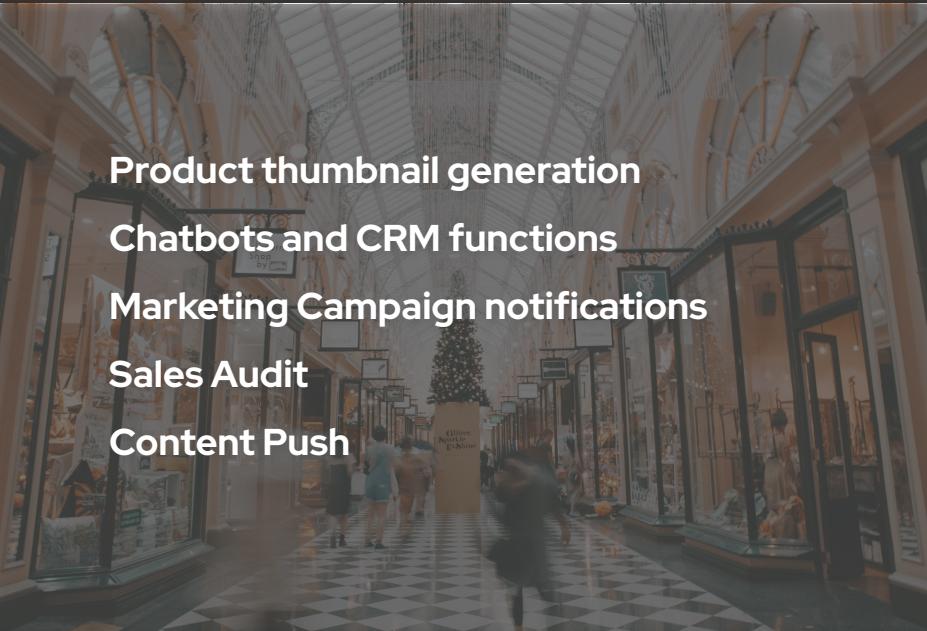
# Serverless Use cases



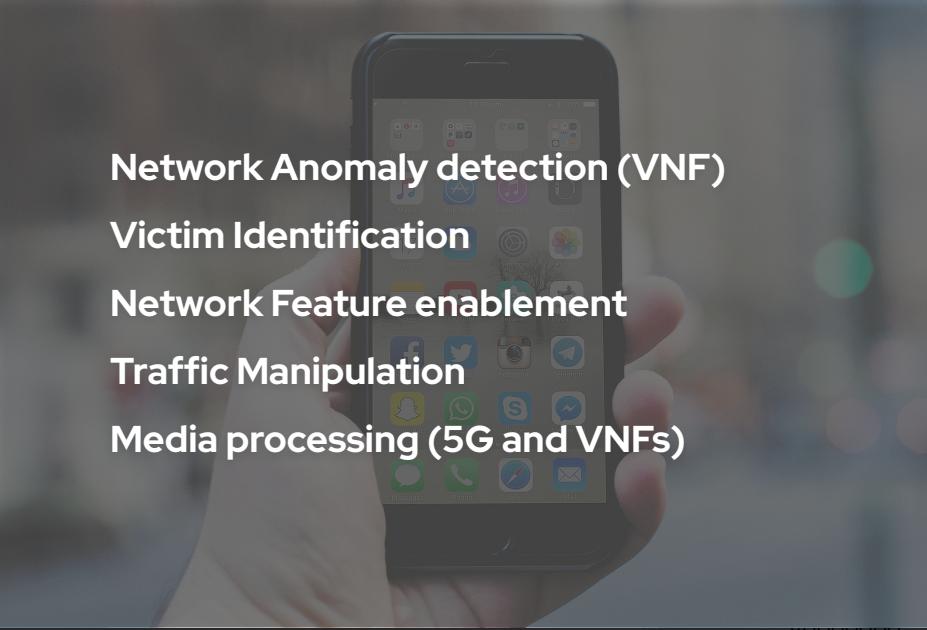
**Cashless payment systems**  
**Transaction processing auditing**  
**Fraud Detection**  
**Credit checks**  
**Check signature validation through OCR**



**Image results validation (X-rays, MRIs)**  
**Fast Healthcare Interoperability Resources**  
**Queries**  
**Result notifications**  
**Scheduling services**  
**Test result requests (PDFs, Reports)**



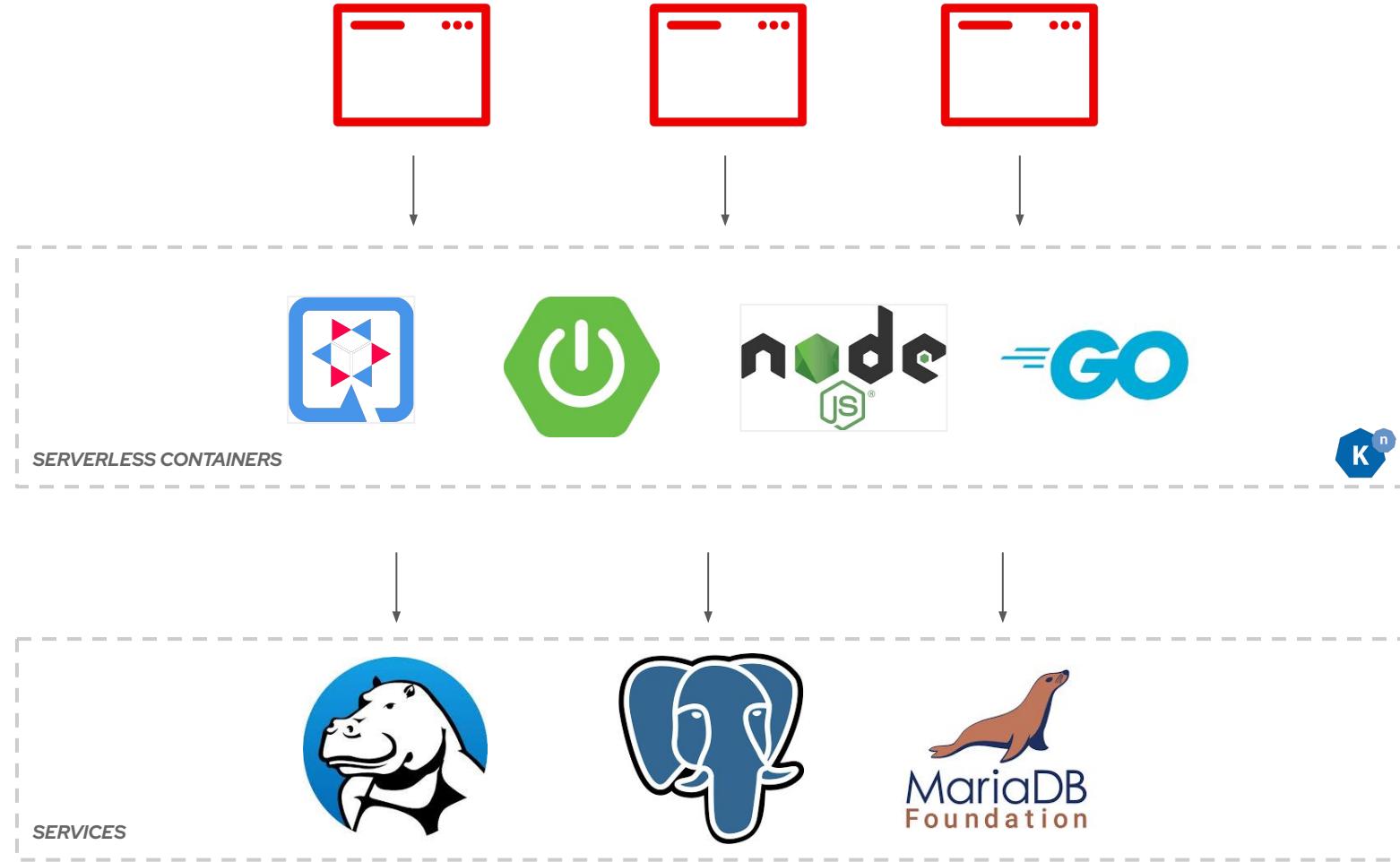
**Product thumbnail generation**  
**Chatbots and CRM functions**  
**Marketing Campaign notifications**  
**Sales Audit**  
**Content Push**



**Network Anomaly detection (VNF)**  
**Victim Identification**  
**Network Feature enablement**  
**Traffic Manipulation**  
**Media processing (5G and VNFs)**



# Web Applications and APIs



Language or runtime of your choice:

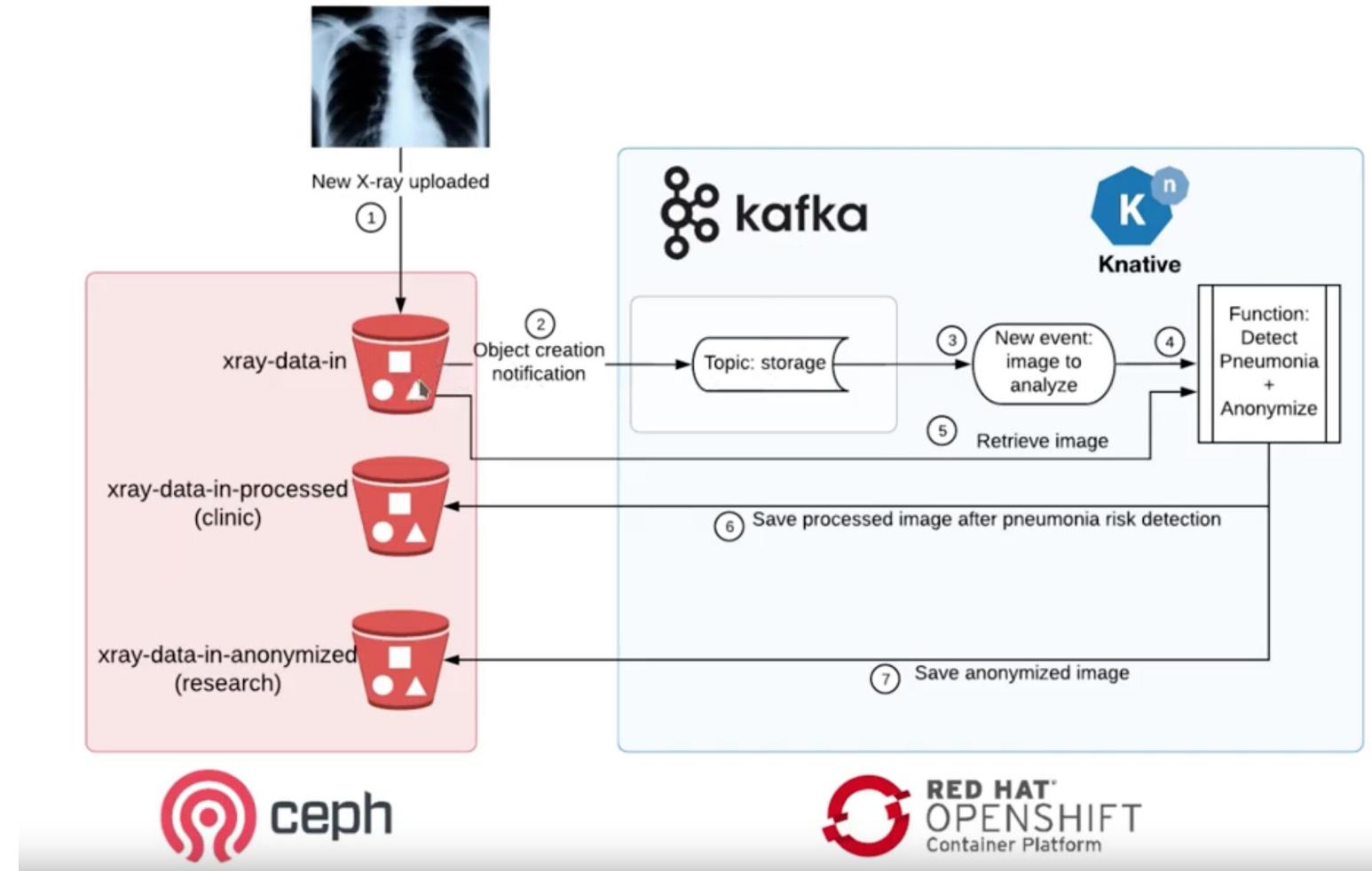




# Data Transformation

## Common Use cases

- Image analysis\* (medical, AI/ML, media)
- Video format transcoding
- File type conversion (financial, medical)
- Data extraction
- Lightweight Data Transformation
- Invoice Generation



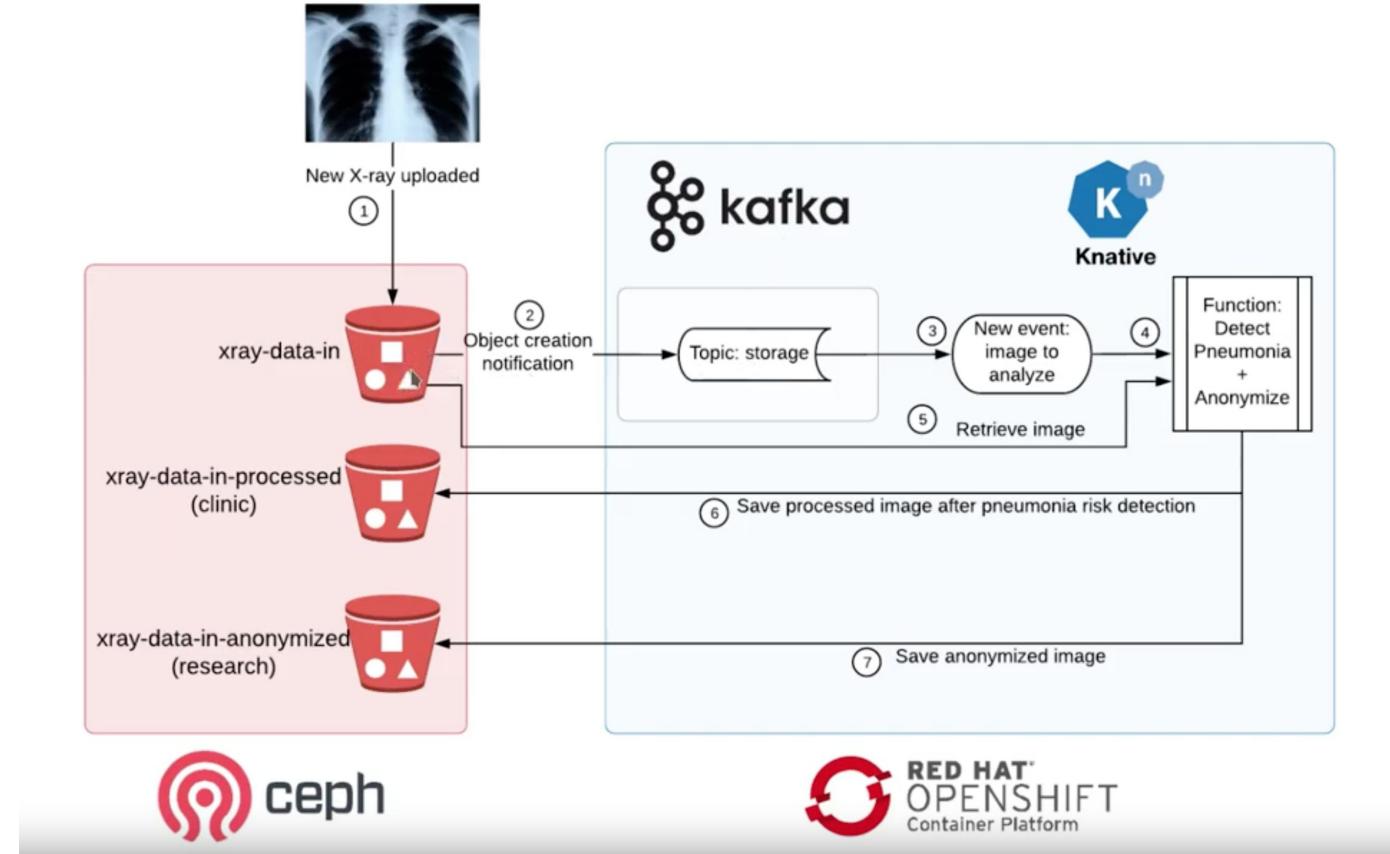


# Ceph + Kafka + Serverless

*"Automated AI/ML Data Pipelines"*

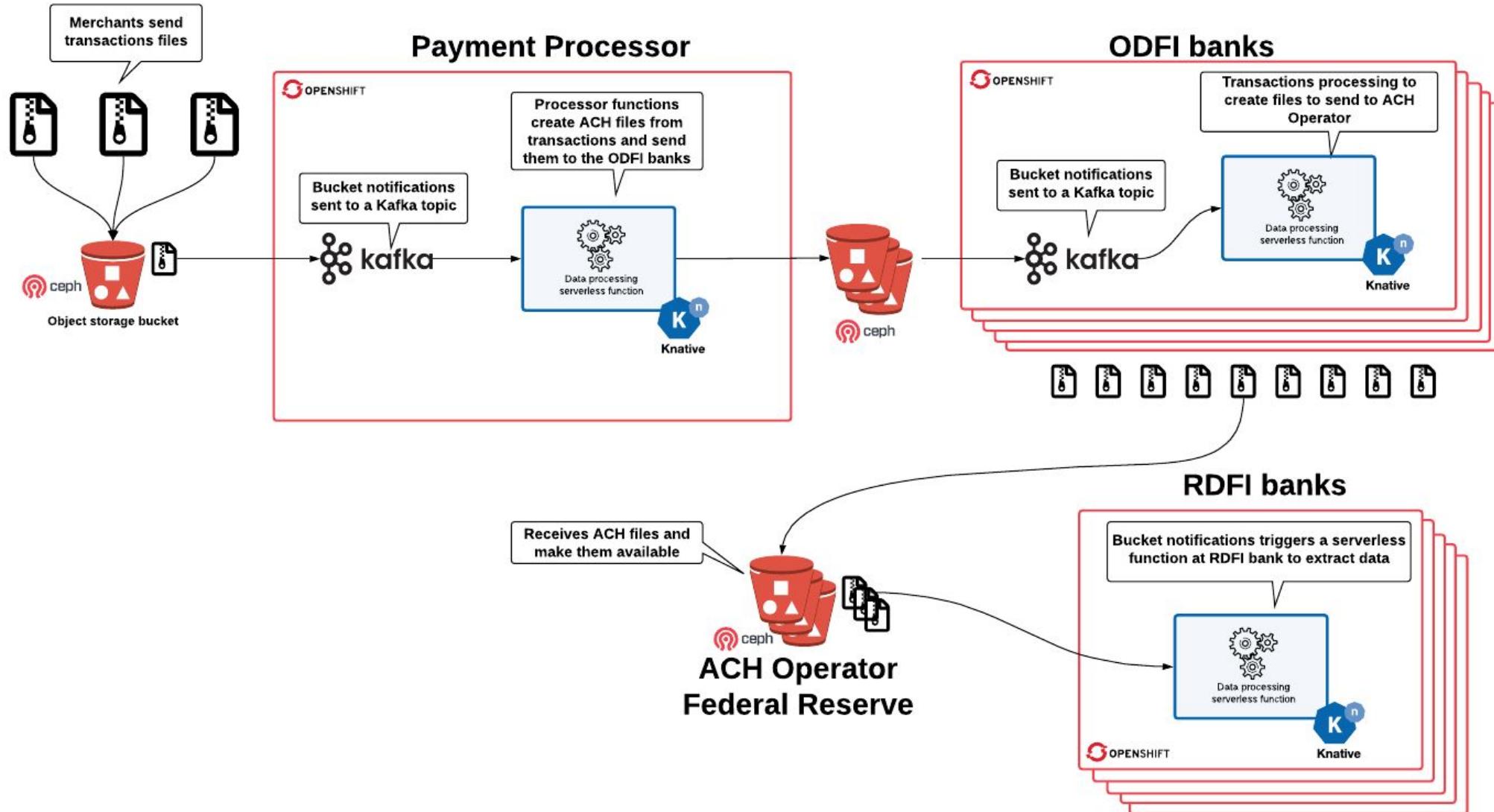
## Common Use cases

- Image analysis\*  
(medical, AI/ML, media)
- Video format transcoding
- File type conversion  
(financial, medical)
- Data extraction
- Data Transformation
- Invoice Generation





# Serverless Open Banking





# Building on OpenShift Serverless with Red Hat Services

## Connected Services

How Knative services interact with the outside world.



## Service Orchestrator

Composing multiple services together into an application.



## Event Streaming

All modern architectures need some Kafka.



## API Gateway

Next gen APIs still require management.



## Implementing Services

Functions, languages, and the vagaries of cold starts.



## The Dirty Word in Serverless

Yep, you still need state to handle long-lived orchestration.

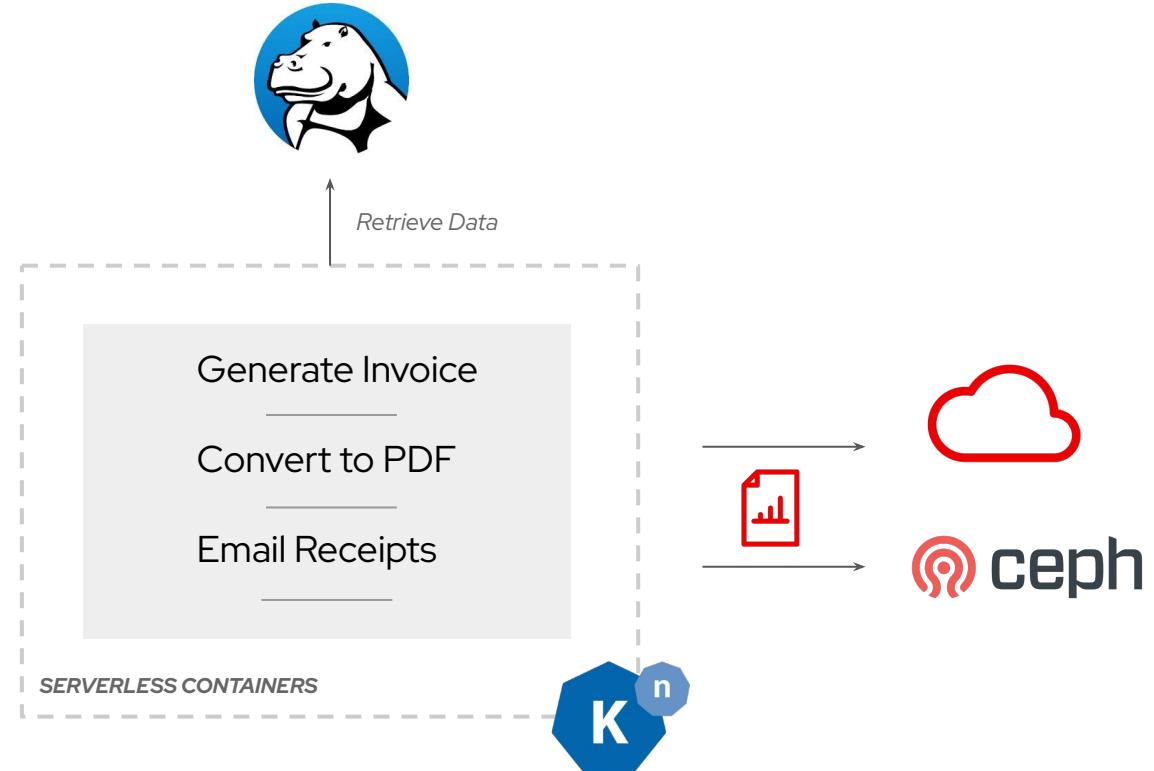




# Scheduled Apps Cron Jobs



- Every Hour →
- Every Day →
- Every Week →
- Every Month →



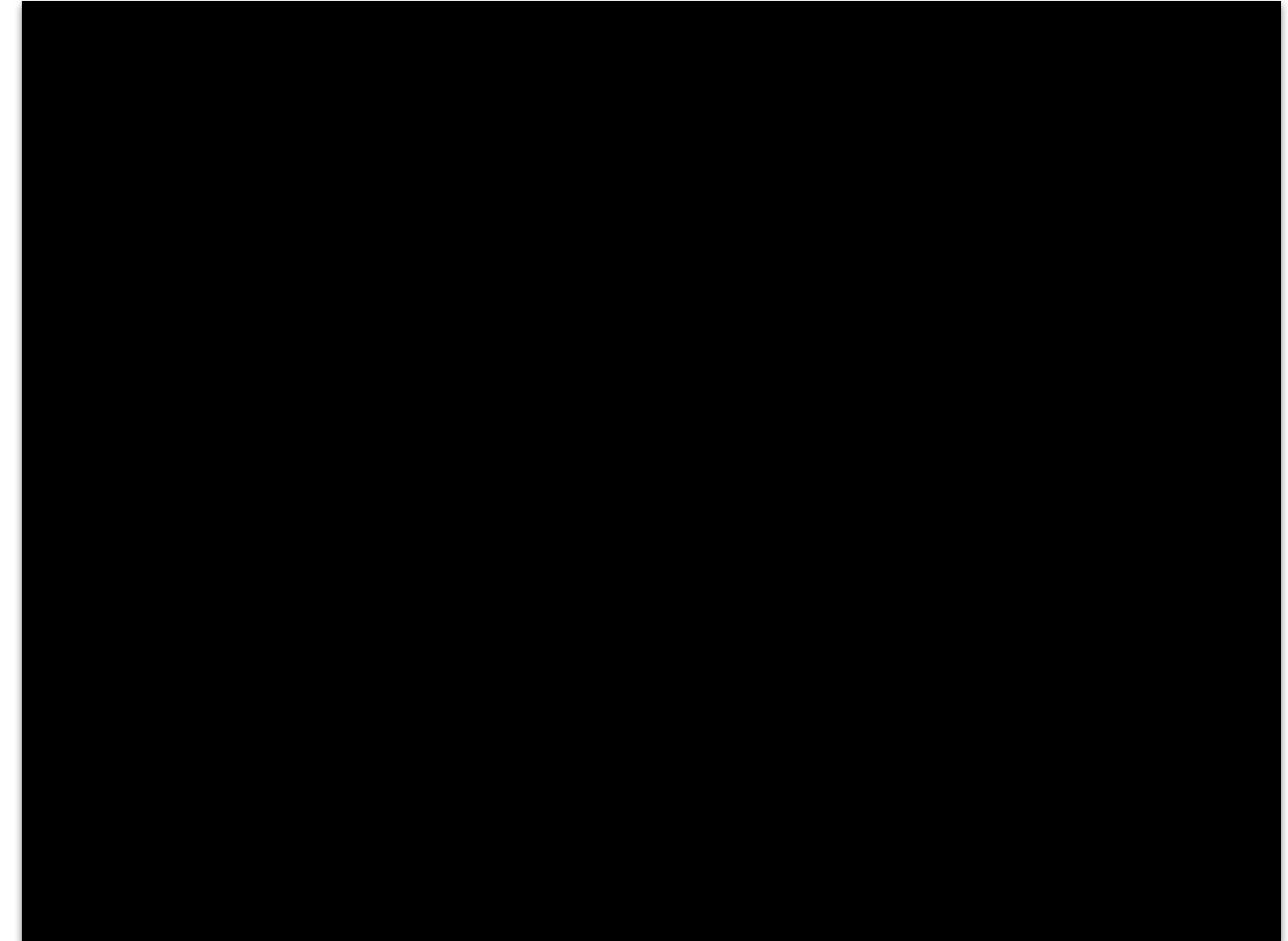
# Partners and ecosystem



★ Unstar 1.8k



" KEDA serves as a Kubernetes Metrics Server and allows users to define autoscaling rules and scalers for any containerized workload. It enables Azure Functions on Kubernetes.



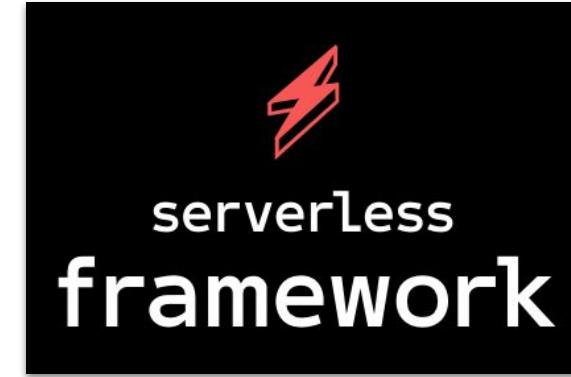
```
# Create a new Serverless service/project
$ serverless create --template knative-docker

# Change into the newly created directory
$ cd my-service

# deploy your application
$ serverless deploy

# Invoke from the CLI
$ serverless invoke -f hello

# Remove the service
$ serverless remove
```



Used by ▾ 8.6k   Watch ▾ 1k   Unstar 34.8k   Fork 4k

**"** The #1 framework to deploy functions in AWS Lambda can now be used with OpenShift and deploy containers and microservices.



**TRIGGERMESH**  
CLOUD NATIVE SERVERLESS INTEGRATION

” TriggerMesh has a certified Operator for OpenShift and is focusing on Serverless cloud native integration use cases, enabling a hybrid cloud even bus.



” IBM Cloud Services provide event sources that can trigger serverless applications running in OpenShift and will be using Knative as basis for a unified developer experience.

# OpenShift Serverless

## Links

- ✓ <https://www.openshift.com/serverless>
- ✓ [Knative Tutorial](#) by Red Hat Developers
- ✓ [Knative Cookbook](#) by Red Hat Developers
- ✓ <try.openshift.com/serverless> (coming soon)



## Red Hat's OpenShift Serverless for hybrid, legacy and greenfield

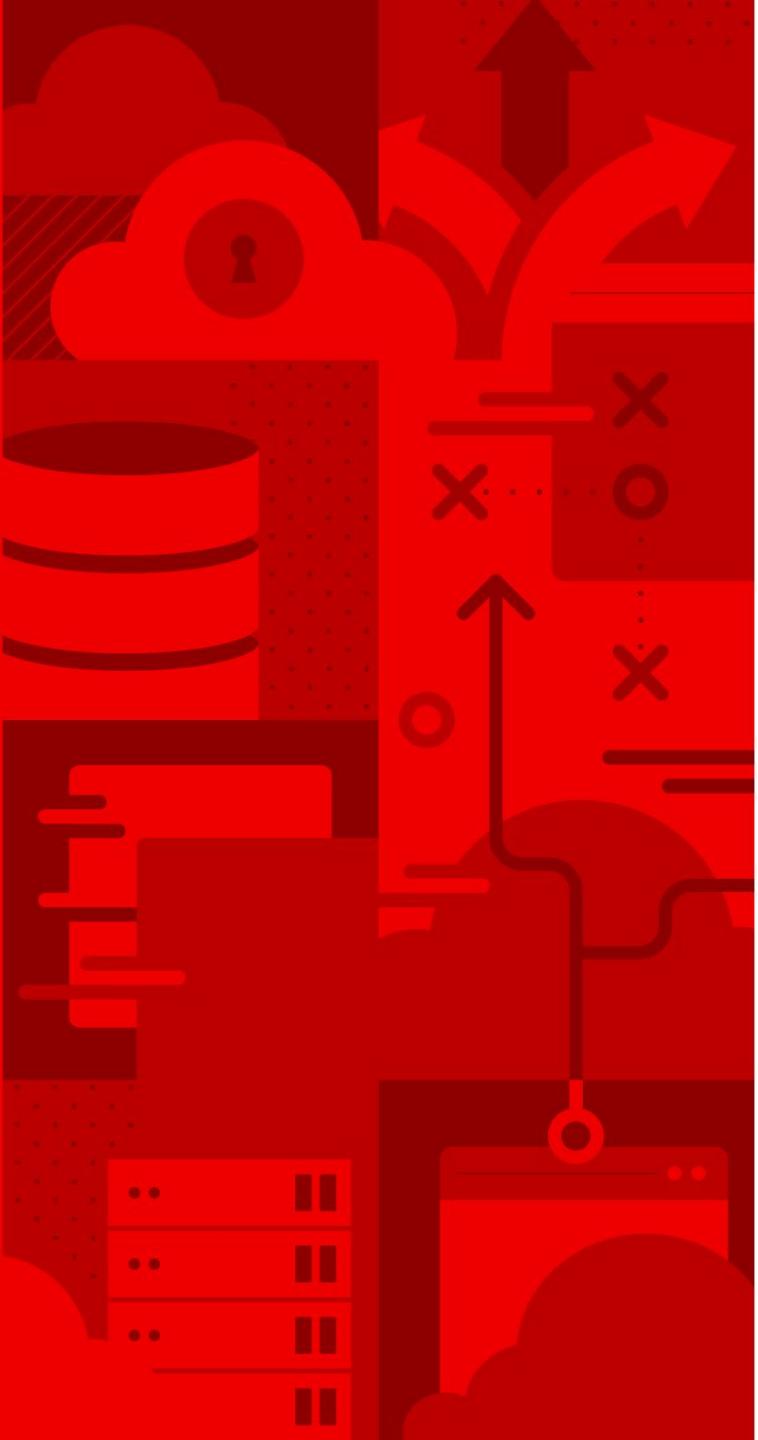
FEBRUARY 21 2020

By William Fellows

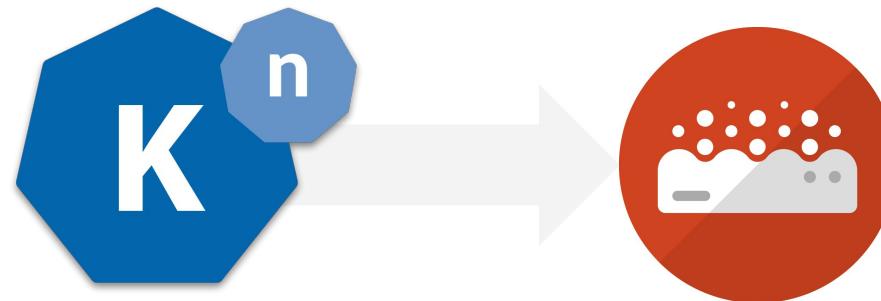
Kubernetes is complex and difficult to deploy – it autoscales based on available resources, not on requests themselves. OpenShift Serverless is designed to resolve this complexity to deliver the benefits ‘as advertised’ of quicker time to market and faster recovery.

Read the analyst report





# Thank you



[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](https://twitter.com/RedHat)