Event-Driven Architecture and Serverless with Red Hat

Red Hat

“Serverless,” Defined

Serverless computing is a cloud computing execution model in which the applications are written by the cloud consumer and the infrastructure is managed by the cloud provider.

“Serverless,” Defined

Serverless computing is...

- a cloud computing execution model
- where the applications are written by the cloud consumer
- and infrastructure is managed by the cloud provider

“Serverless,” Defined

Serverless computing is...

- a cloud computing **execution model**
- where the applications are written by the cloud consumer
- and infrastructure is managed by the cloud provider

“Infrastructure”: from hardware all the way up the stack to the ***number of instances of applications running***

Serverless, Defined

- execution model
- applications are written by the cloud consumer
- infrastructure is managed by the cloud provider.

Serverless vs FaaS

Functions as a Service (FaaS): small pieces of code that you trust someone else to run for you. In general, architecture options are limited to what they run and how they run it.

Serverless *isn't* the same as FaaS - but FaaS systems do generally use the Serverless execution model to manage how they run.

Serverless uses the power of containers and automation to minimize the thought and work needed to run applications.

Okay, but *why?*

“Running containers is pretty easy!”
or
“Just getting to containers and OpenShift is so huge for us, let’s look at that first!”

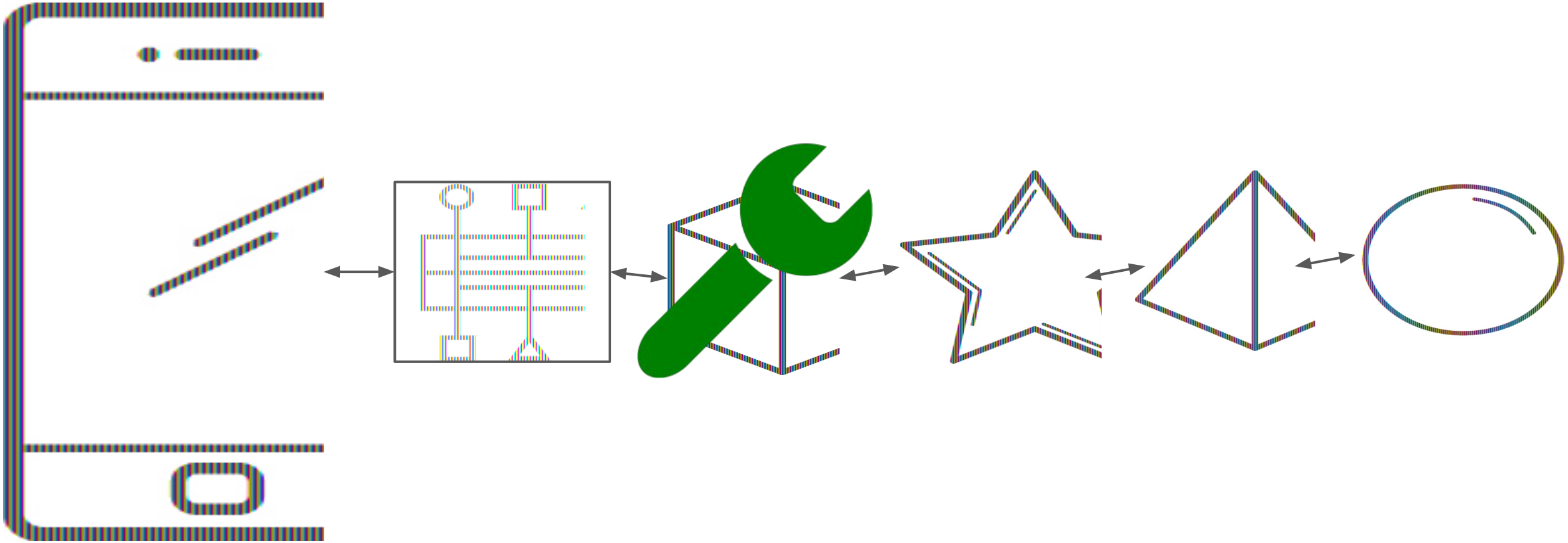
Why do I need to understand this new thing?

Weaknesses of Traditional Architecture

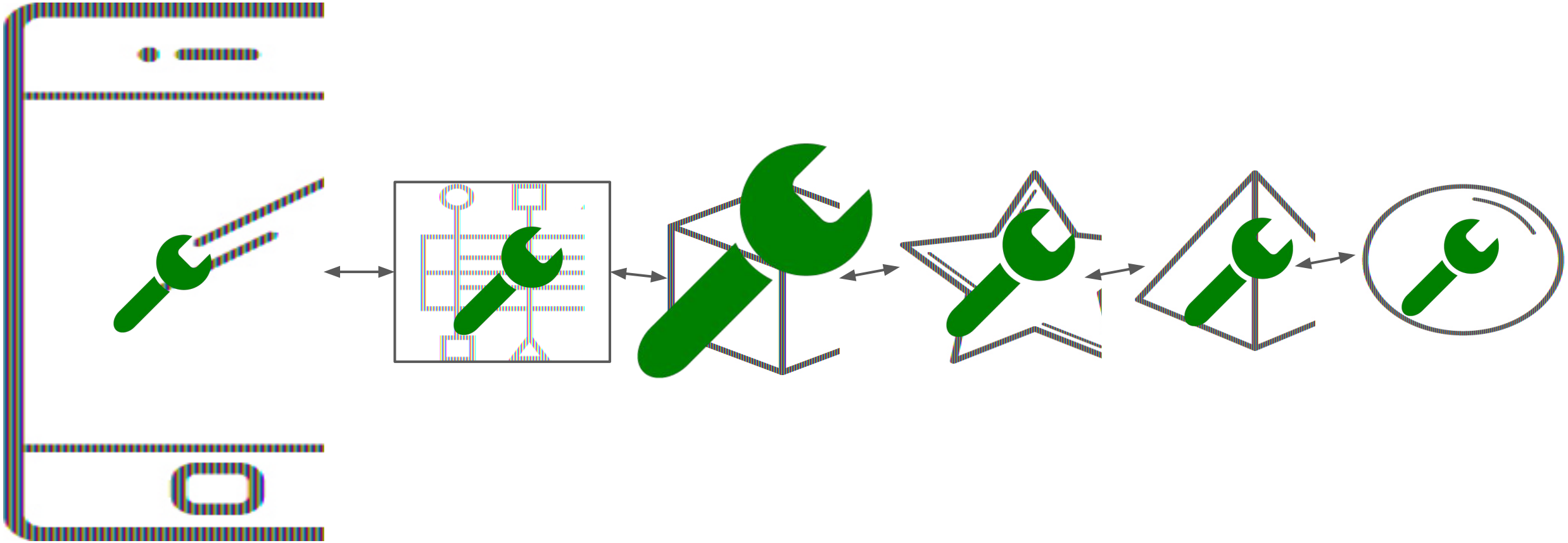
Weakness 1: Tight Coupling

When one thing changes, everything changes.

Tight Coupling



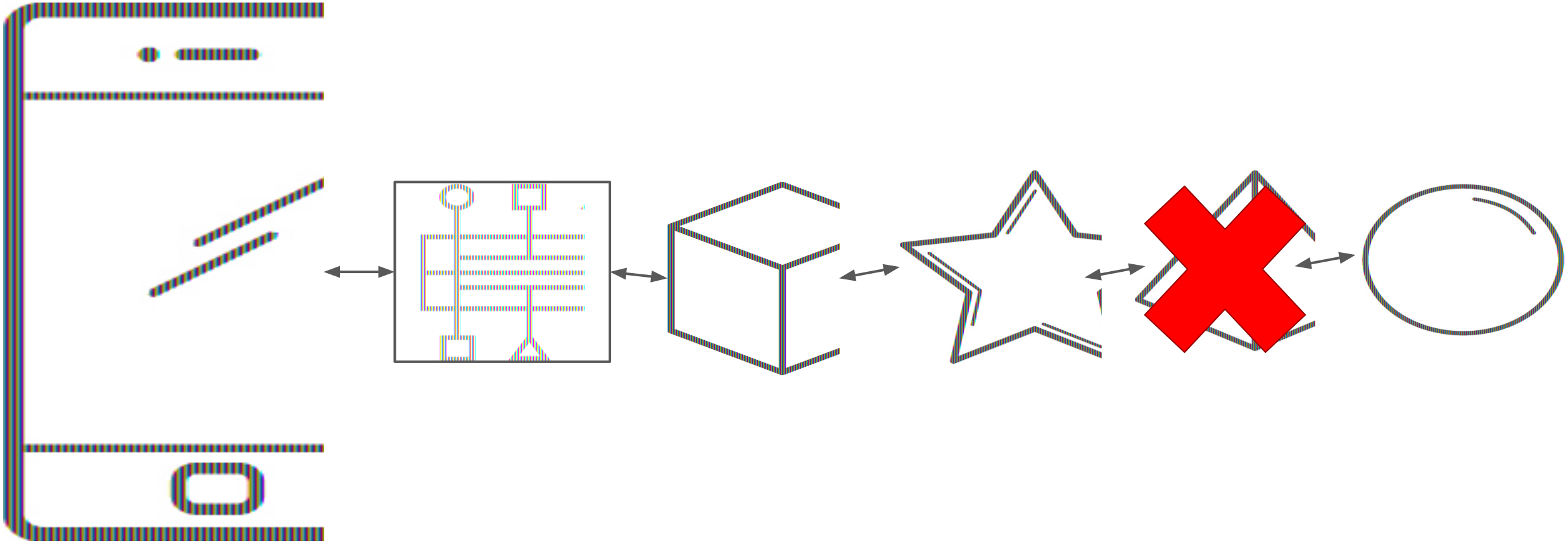
Tight Coupling (Cascading Change)



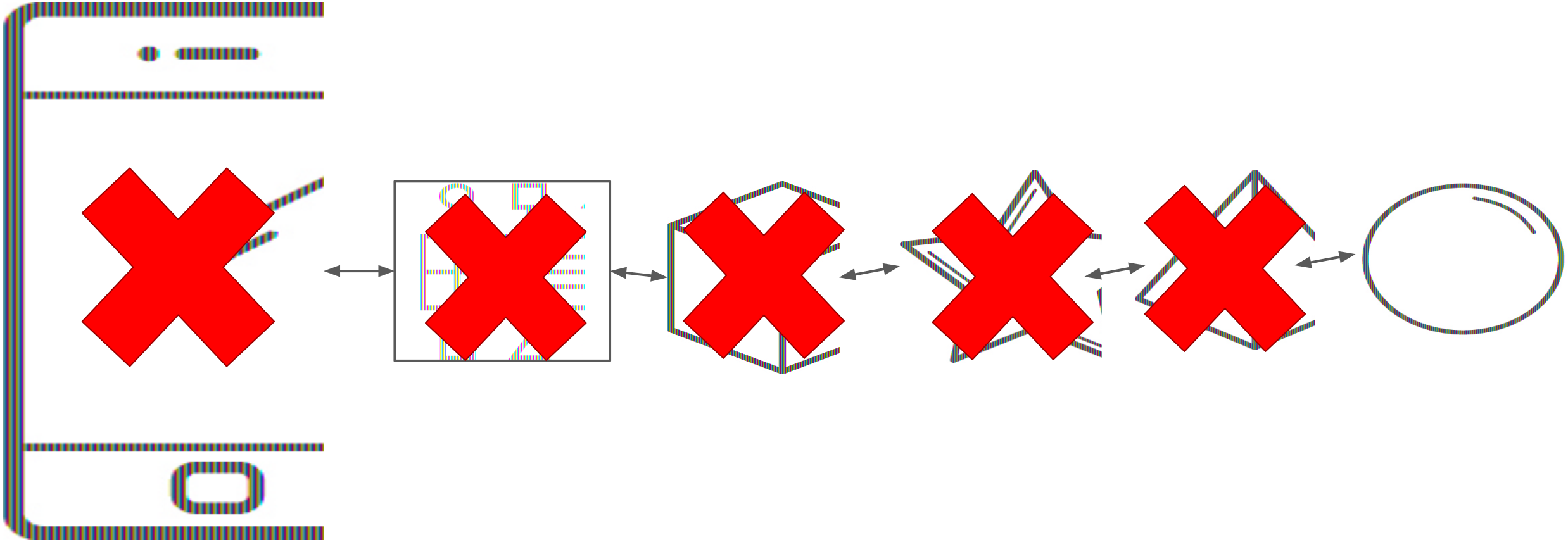
Weakness 2: Baton Dropping

When something breaks, everything breaks.

Baton Dropping



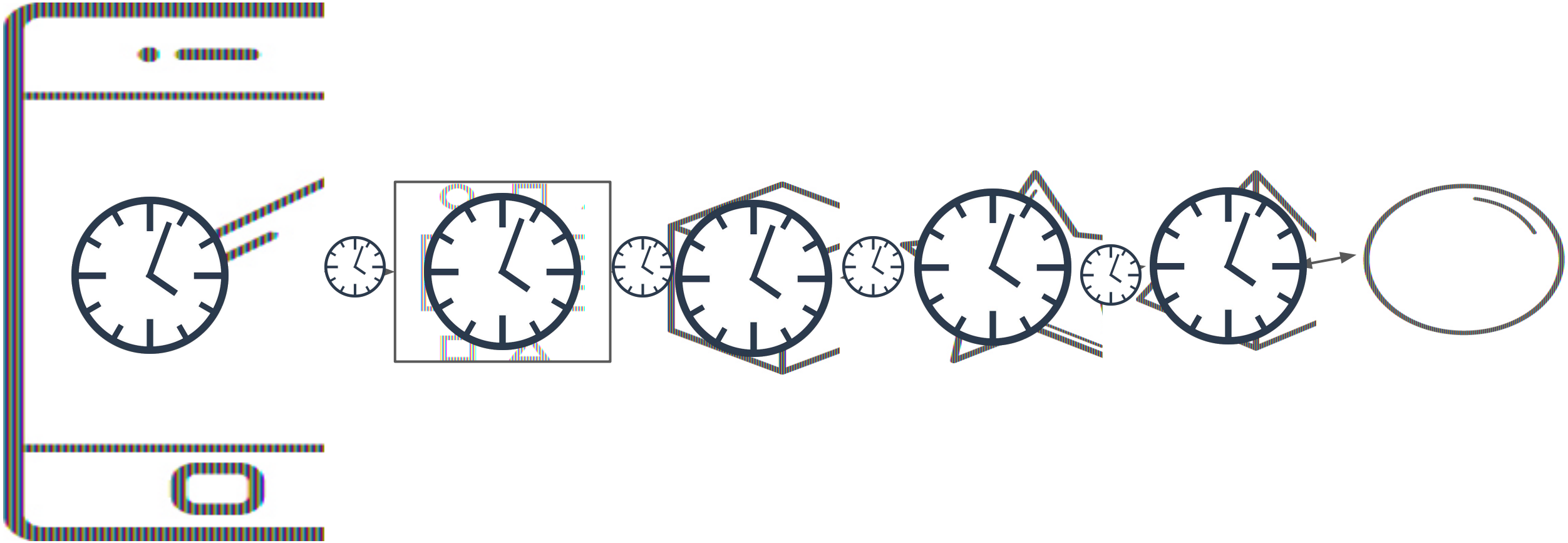
Baton Dropping (Cascading Failure)



Weakness 3: Call Chain Latency

- ➔ *Every call in the chain takes time to execute the code.*
- ➔ *Every hop between calls takes network time.*
- ➔ *Jumping between data centers and clouds add MORE time/latency.*

Call Chain Latency



Weakness 4: Sizing and Managing Scale

How many instances of the running application should we have?
1? 2? 5?

It's *expensive* to guess too many (because of wasted infrastructure!) and even **more** expensive to guess too few - because customers expect responsiveness.

Red Hat

Recap: Traditional Architecture Weaknesses

1. Tight coupling
2. Cascading failures
3. Call chain latency
4. Scaling

Effects of Microservices and Clouds

1. *Longer* call chains
2. *More* network latency
3. Splitting call chains
4. Scaling is even *harder* - so many services!

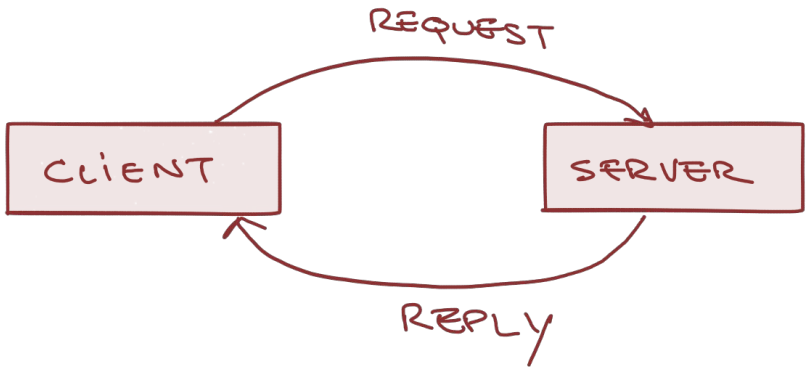
Architecture Problems
Tight coupling
Cascading failures
Call chain latency
Cloud latency
Getting scaling right

A Solution Appears! Asynchronous Processing + Event-Driven Architecture

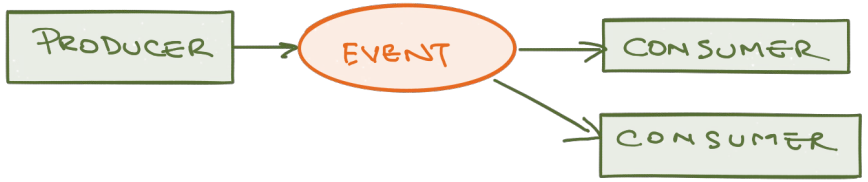
A Human Analogy

- Visiting the Secretary of State
- Filling out a form online

Request-Reply vs. Event-Driven



Synchronous
Ephemeral
Low composability
Simplified model
Low tolerance to failure
Best practices evolved as REST



Asynchronous
Optionally Persistent
Highly composable
Complex model
High tolerance to failure
Best practices still evolving
Decoupled

Asynchronous Processing Notes

1. Asynchronous processing
2. Messages
3. Think of the secretary of state - waiting
4. “Avoid the wait” - just like emailing or filling out a form
5. Messaging technology has existed a long time, and is now very mature
6. Kafka
7. Nearly Everything can be asynchronous
8. Serverless can be workload-aware, and scale up/down based on the amount of incoming messages

All About Events

“Event” - an action or occurrence that happened in the past as a result of something (usually an end user, could also be another system) interacting with a system. Like...

- order created
- new account opened
- claim created

All About Events

Characteristics of an “event:”

- Immutable
- Can be persisted
- Shareable

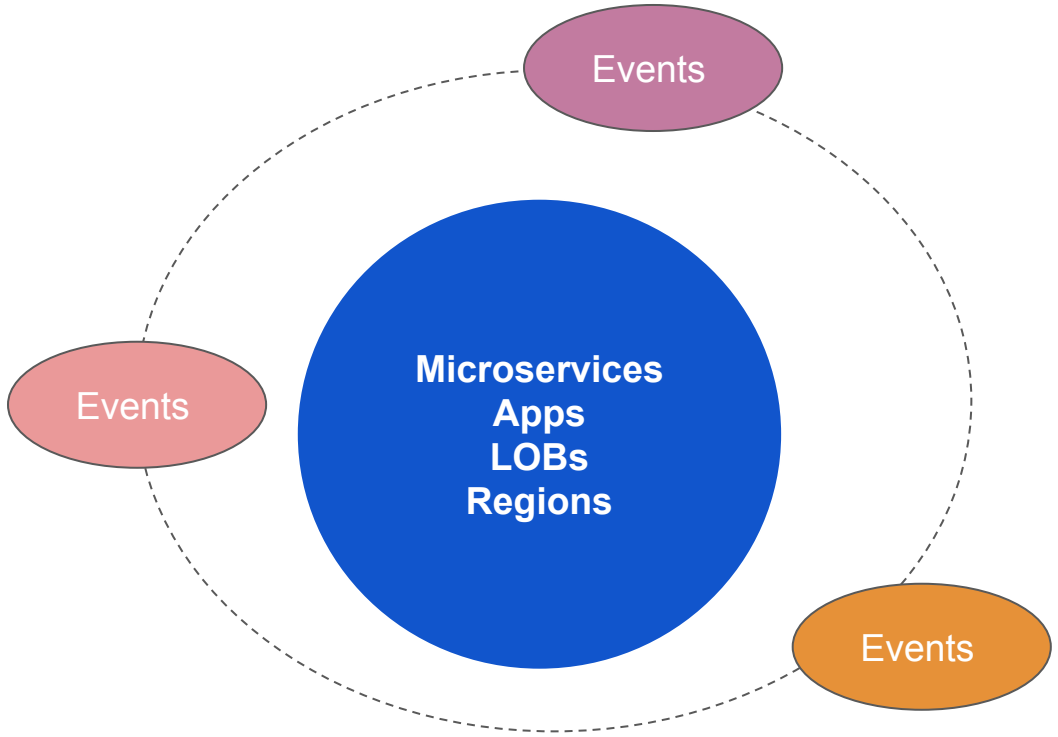
All About Events

Event types:

- Notification
- State Transfer (Command)
- Event-Sourcing/CQRS

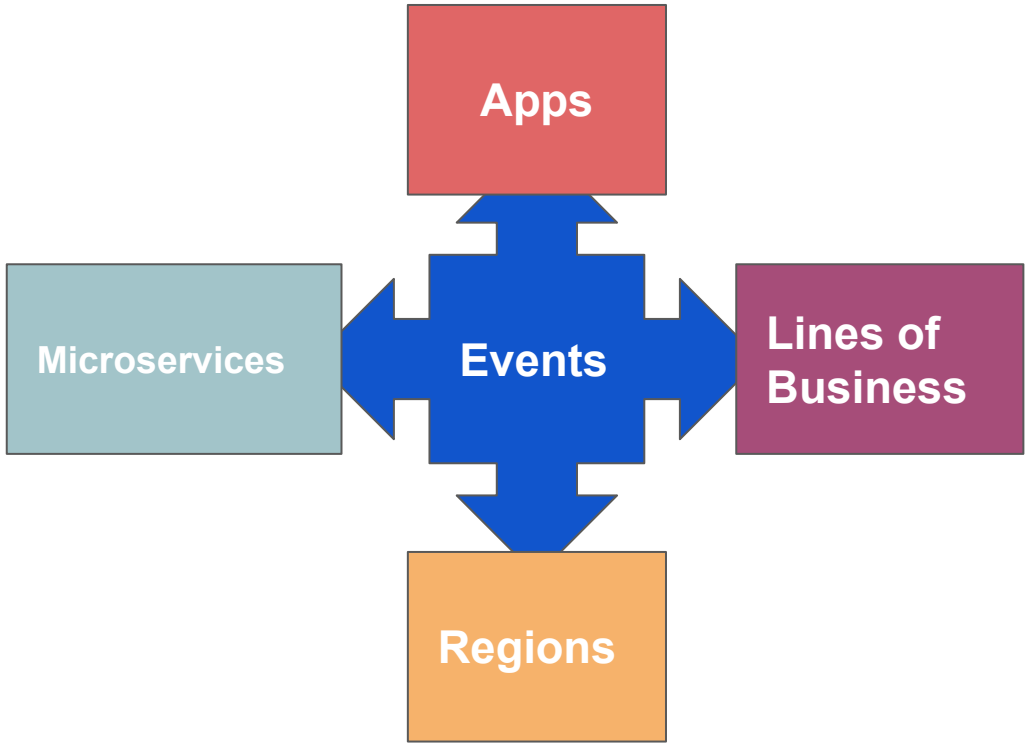
https://martinfowler.com/articles/201701-event-driven.html

A Change in Thinking



System-centric, and data-centric

Events are ephemeral, intended to make systems work, while systems own their own systems of record

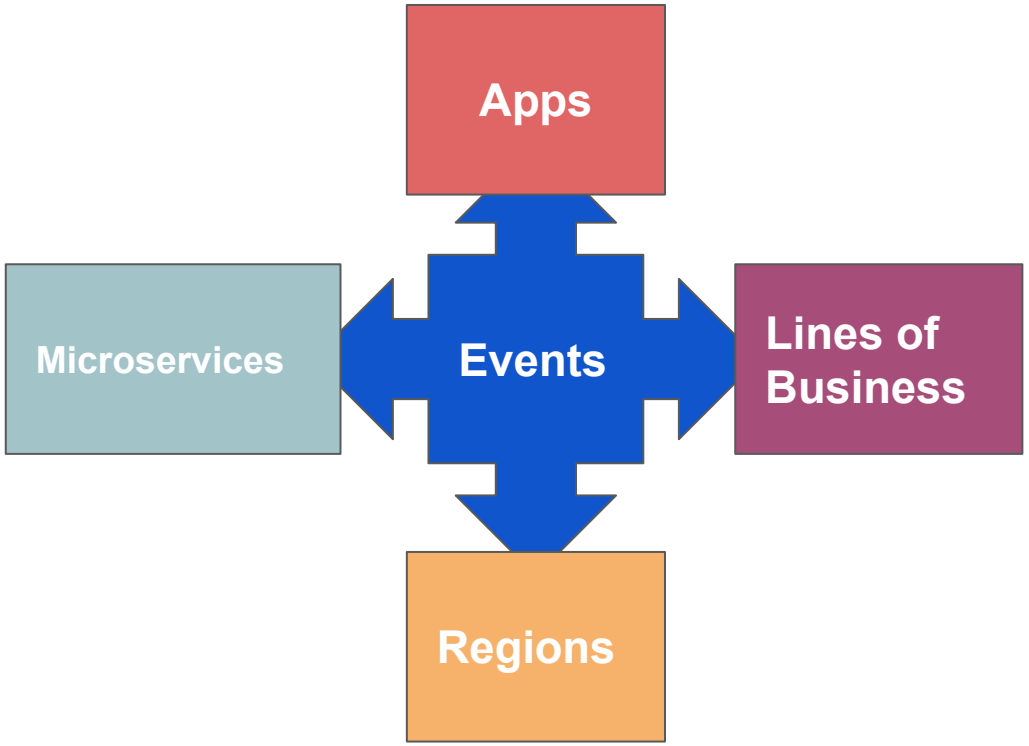


Event-centric

Events are long-lived or permanent; designed to serve as a first-class enterprise information store

Red Hat

A Change in Thinking: Event-Centric

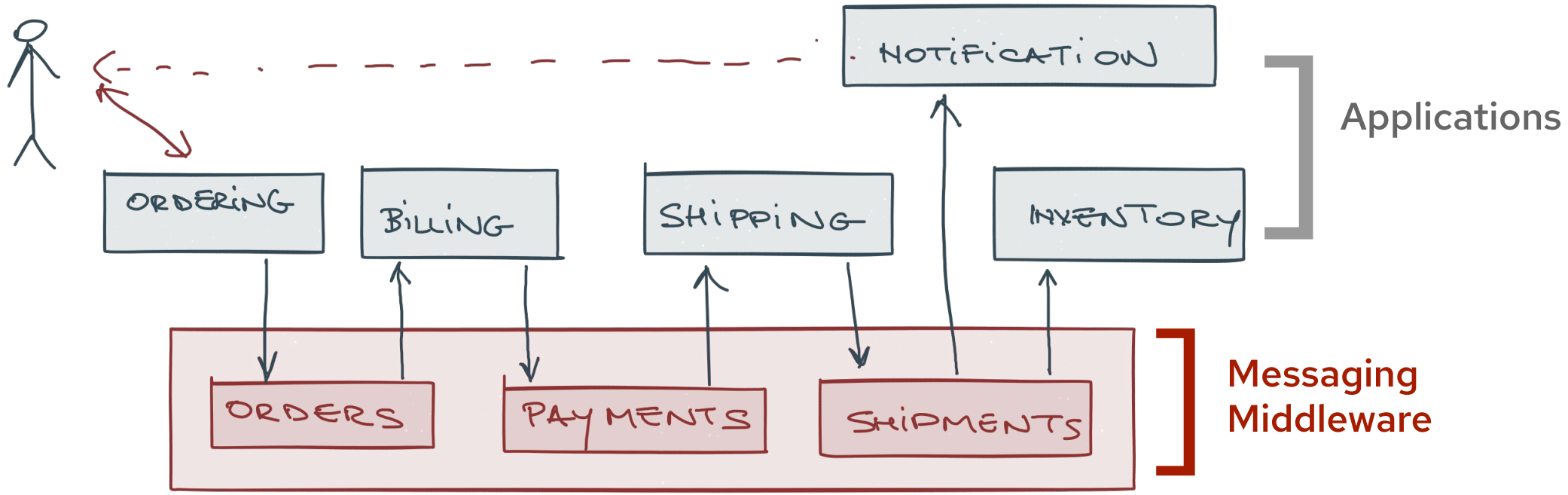


Event-centric

Advantages of this approach:

- Services can be simpler & stateless
- Communication patterns are clearer and easier to follow
- Data silos can be decreased while keeping ownership clear

Event-Driven Microservices: A Model



Red Hat

TRADITIONAL MESSAGING

EVENT STREAMING

Advantages

- Store-and-forward
- individual message exchanges (transactionality, acknowledgment, error handling/DLQs), P2P/competing consumer support
- Publish-subscribe support with limitations

Trade-offs

- No replay support
- Requires fast and/or highly available storage infrastructure
- No ordering at scale

Advantages

- long-term persistence, replay, semantic partitioning, large publisher/subscriber imbalances, replay and late-coming subscribers
- Shared nothing data storage model
- Repeatable ordering at scale

Trade-offs

- Weak support for individual message acknowledgment, p2p/competing consumers
- Larger data footprint and extremely fast storage access

Red Hat

Architecture Problems	Architecture Solutions
Tight coupling	
Cascading failures	
Call chain latency	
Cloud latency	
Getting scaling right	

Architecture Problems	Architecture Solutions
Tight coupling	Event-Driven messages
Cascading failures	
Call chain latency	
Cloud latency	
Getting scaling right	

Architecture Problems	Architecture Solutions
Tight coupling	Event-Driven messages
Cascading failures	Event-Driven messages
Call chain latency	
Cloud latency	
Getting scaling right	

Architecture Problems	Architecture Solutions
Tight coupling	Event-Driven messages
Cascading failures	Event-Driven messages
Call chain latency	Event-Driven messages
Cloud latency	
Getting scaling right	

Architecture Problems	Architecture Solutions
Tight coupling	Event-Driven messages
Cascading failures	Event-Driven messages
Call chain latency	Event-Driven messages
Cloud latency	Event-Driven messages
Getting scaling right	

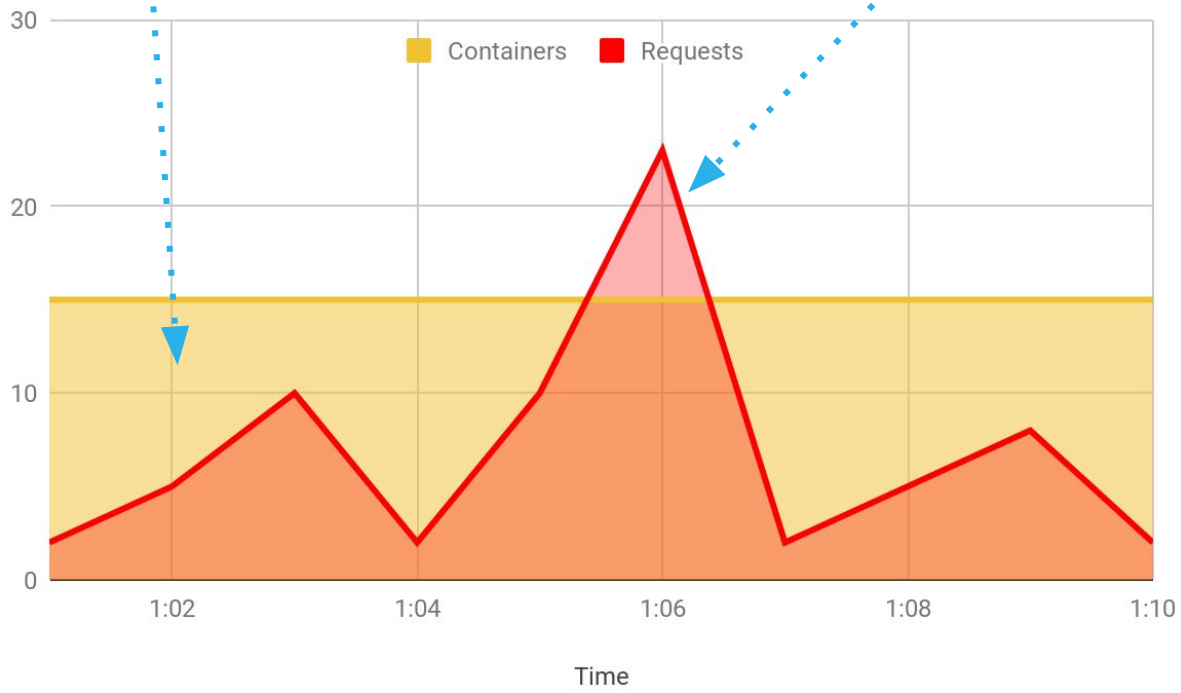
Architecture Problems	Architecture Solutions
Tight coupling	Event-Driven messages
Cascading failures	Event-Driven messages
Call chain latency	Event-Driven messages
Cloud latency	Event-Driven messages
Getting scaling right	???

Serverless Architecture: Automatic *Scaling* for Event-Driven Architecture

Serverless Operational Benefits

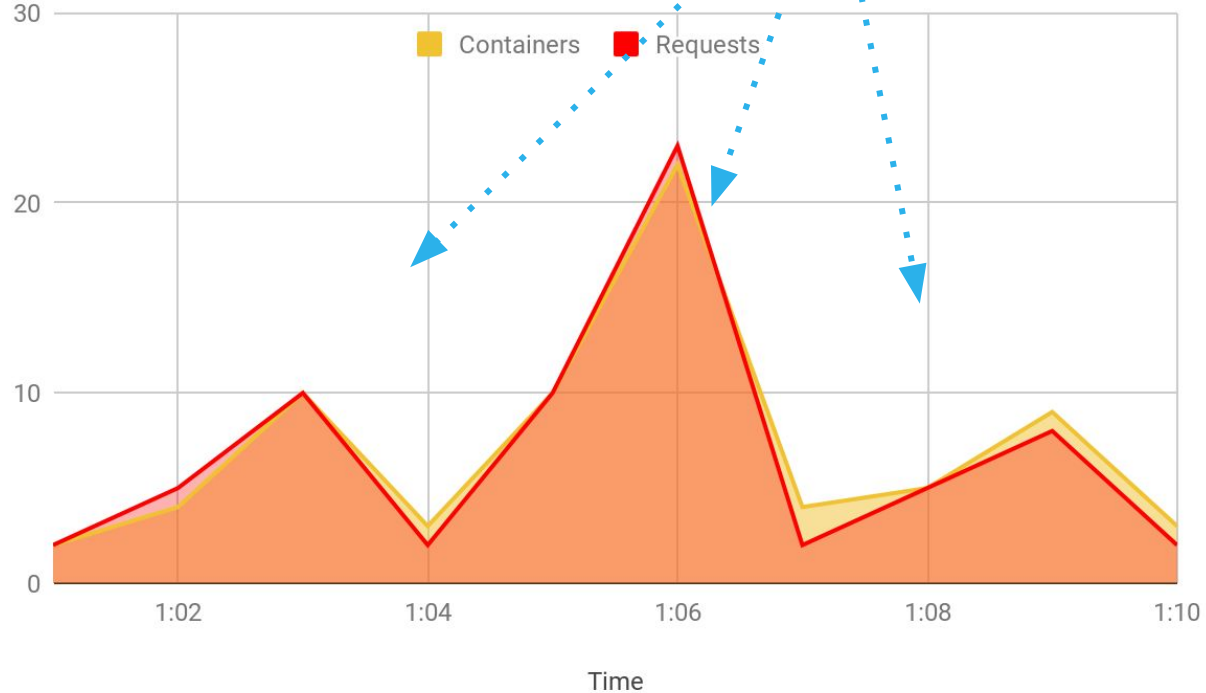
Over provisioning
Time in capacity planning
IT cost of idle resources

Under provisioning
Lost business revenue
Poor quality of service



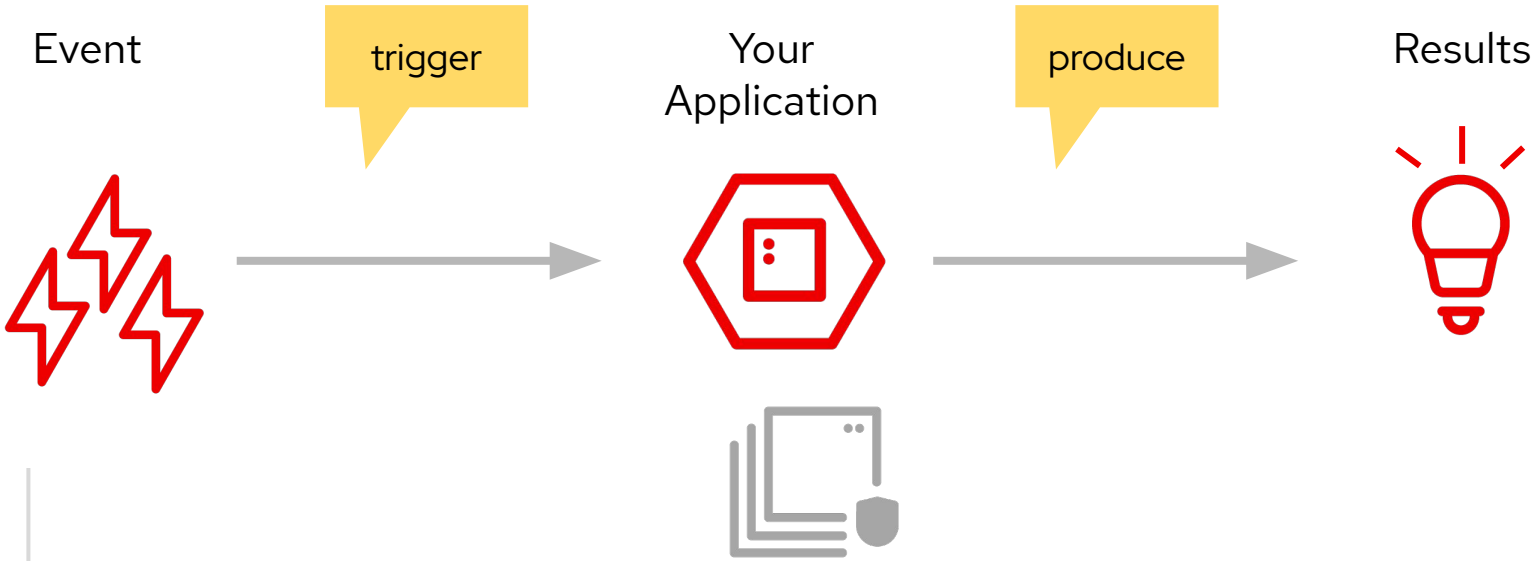
NOT Serverless

More applications
Direct line between IT
costs & business revenue



with Serverless

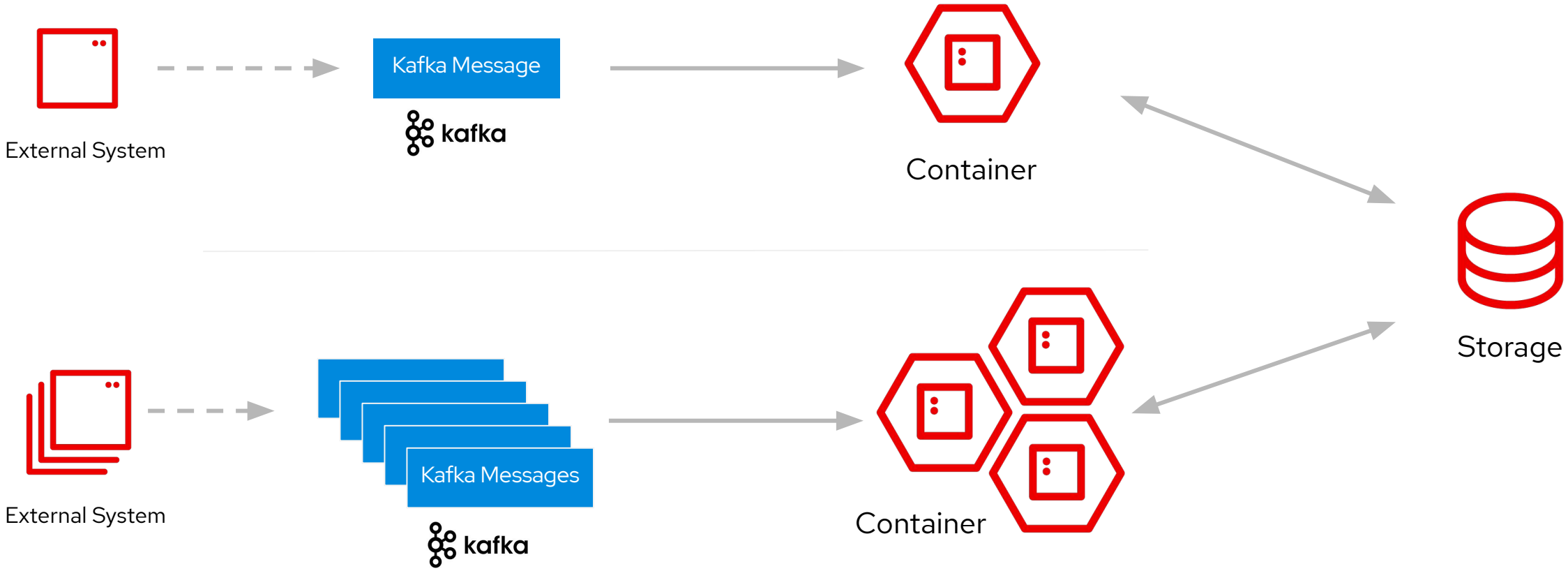
The "Serverless Pattern"



- HTTP Requests
- Kafka Messages
- Image Uploaded
- New Order
- Login from user

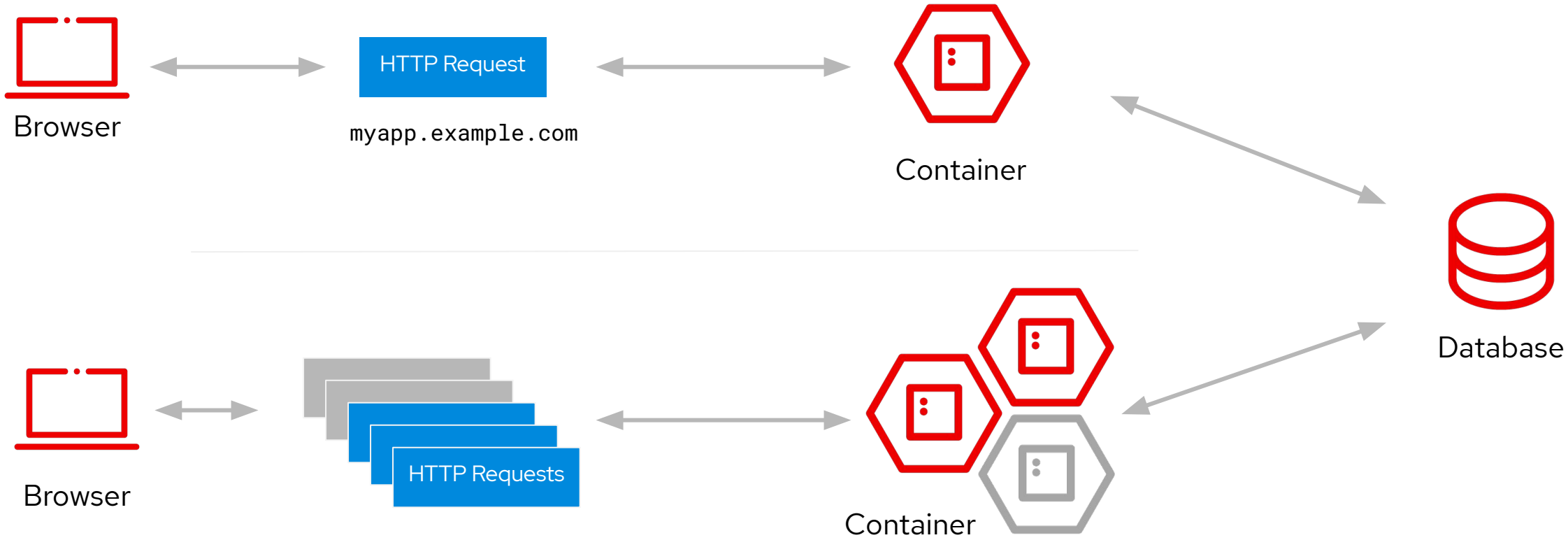
The "Serverless Pattern"

Processing a Kafka message



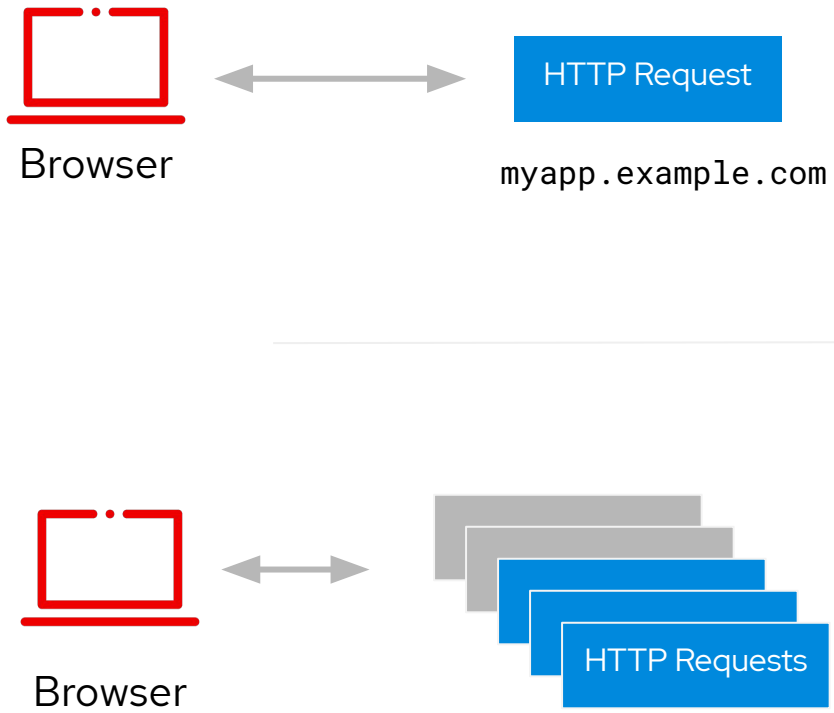
The "Serverless Pattern"

A serverless web application



The "Serverless Pattern"

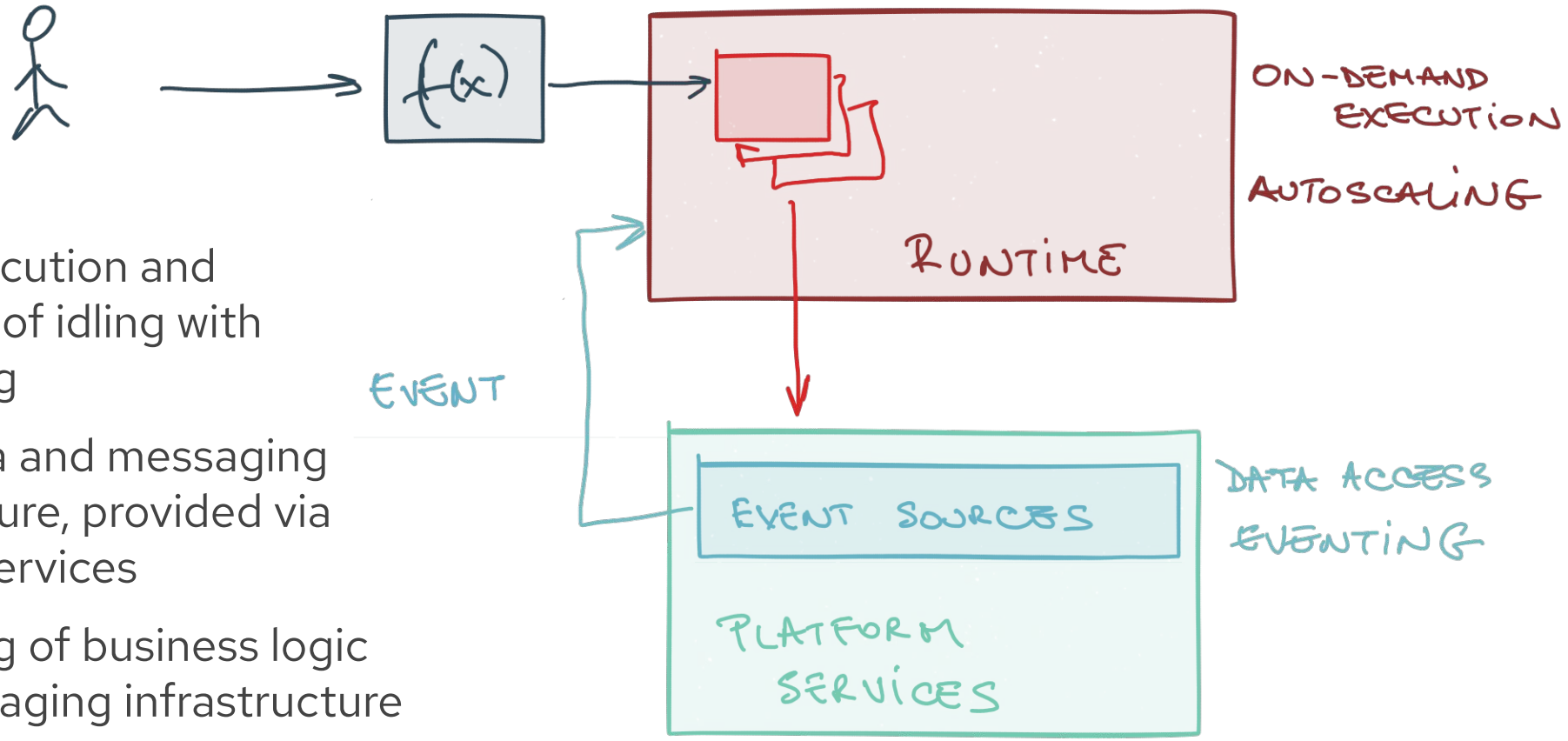
A serverless web application



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

Event-Based Serverless Architecture: A Model



- Elastic execution and avoidance of idling with autoscaling
- Utility data and messaging infrastructure, provided via platform services
- Decoupling of business logic from messaging infrastructure

Red Hat

Architecture Problems	Architecture Solutions
Tight coupling	Event-Driven messages
Cascading failures	Event-Driven messages
Call chain latency	Event-Driven messages
Cloud latency	Event-Driven messages
Getting scaling right	Serverless

OpenShift Serverless

1.0

AWS Lambda, Functions...

Built around the FaaS components and other services such as API Gateways. It enabled a variety of use cases but it is far from ideal for general computing and with room for improvements.

- ➔ HTTP and other few Sources
- ➔ **Functions only**
- ➔ **Limited execution time (5 min)**
- ➔ No orchestration
- ➔ Limited local development experience

1.5

Serverless Containers

With the advent of Kubernetes, many frameworks and solutions started to auto-scale containers. Cloud providers created offerings using managed services completely abstracting Kubernetes APIs.

- ➔ Red Hat joins Knative
- ➔ Kubernetes based auto-scaling
- ➔ **Microservices and Functions**
- ➔ Easy to debug & test locally
- ➔ **Polyglot & Portable**

2.0

Integration & State

The maturity and benefits of Serverless are recognized industry wide and it adds the missing parts to make pattern suitable for general purpose workloads and used on the enterprise.

- ➔ Basic state handling
- ➔ **Enterprise Integration Patterns**
- ➔ Advanced Messaging Capabilities
- ➔ **Blended with your PaaS**
- ➔ Enterprise-ready event sources

Serverless is still evolving...

Serverless Market Trends

"Use Serverless To optimize The Benefits of The cloud" [2]

40%

of enterprises adopted Serverless technologies or practices with expected growth coming in the next 12 to 18 months.[1]

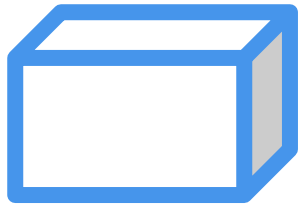
Vendor lock-in is the second biggest concern when adopting Serverless technologies.[1]

60%

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

Source:
1. https://www.oreilly.com/radar/oreilly-serverless-survey-2019-concerns-what-works-and-what-to-expect/
2. Forrester - Now Tech: Serverless, Q4 2019

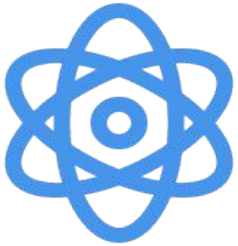
Application Architecture Choices



Monolith

Cloud Native

Serverless

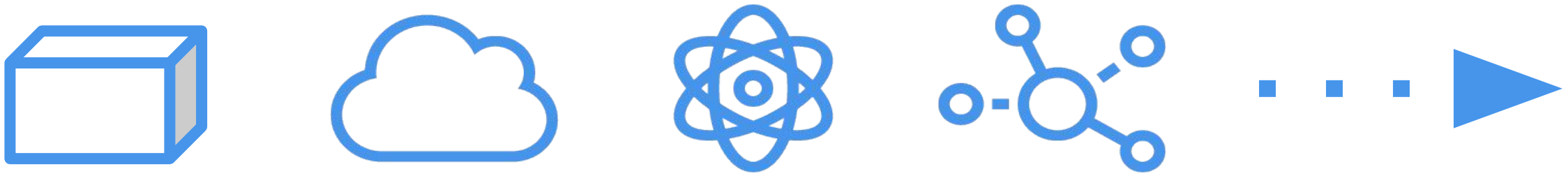


Request/Reply Architecture

Event-Driven Architecture

Red Hat

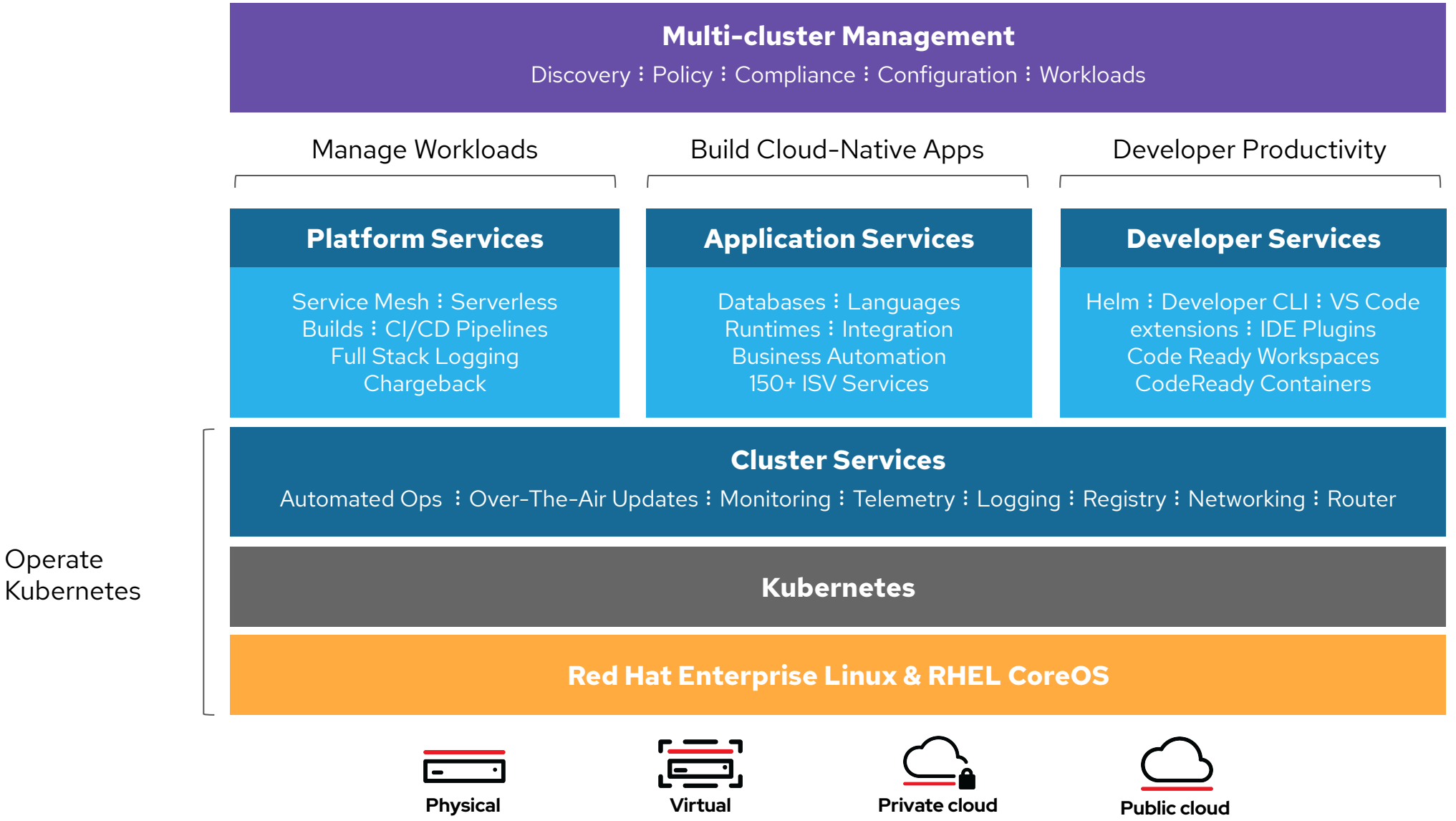
Common Deployment Tools



Monolith | Cloud Native | Request/Reply | Serverless | Event-Driven

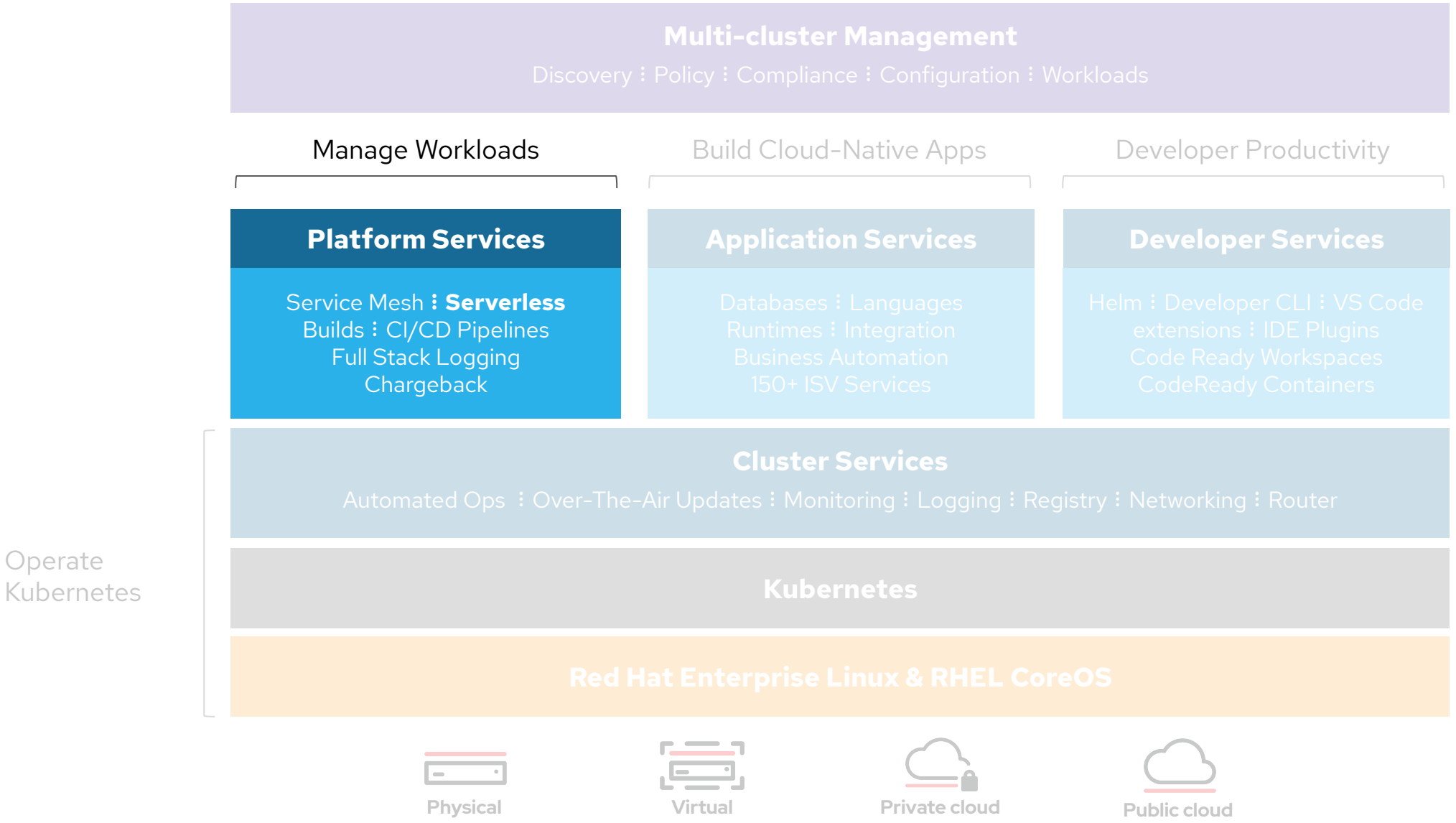
kubernetes | Istio | Knative

OpenShift Container Platform



Red Hat

OpenShift Container Platform



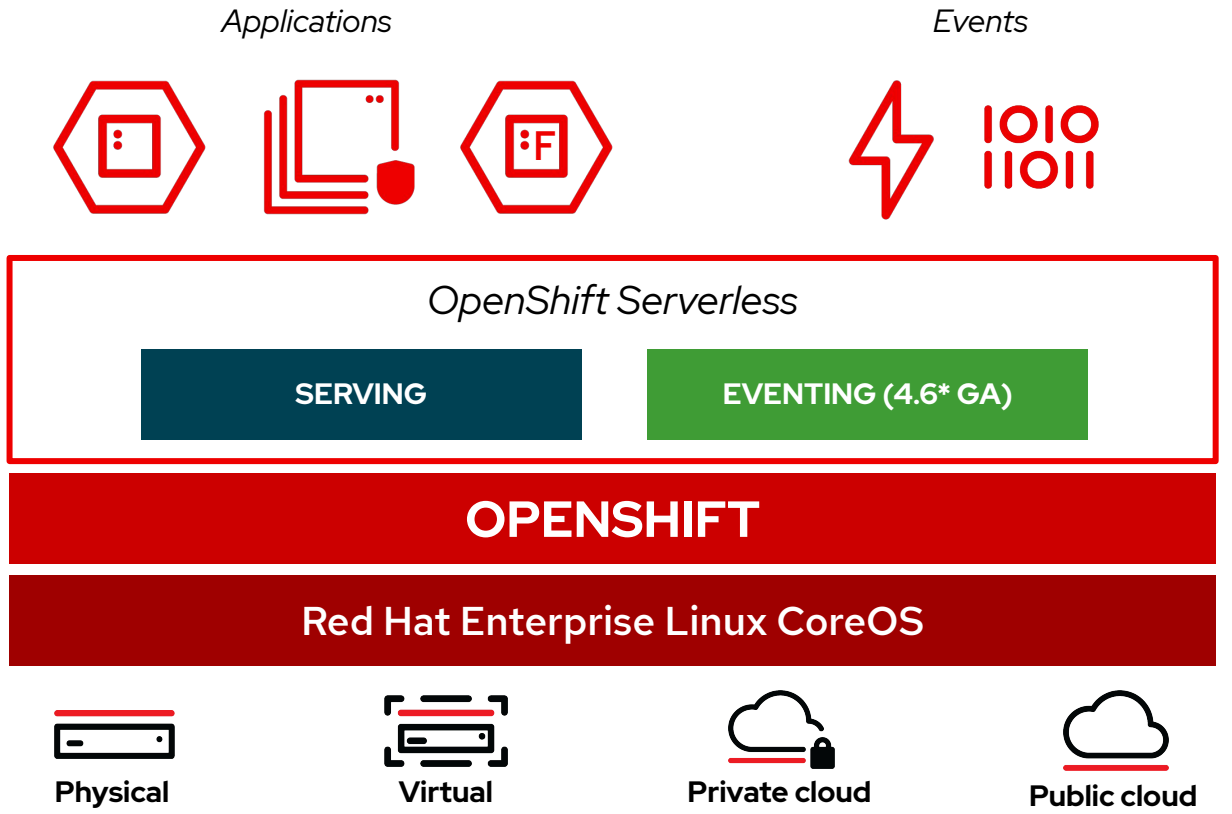
Red Hat

OpenShift Serverless

Event-driven, serverless containers and functions

GA in OpenShift 4.5

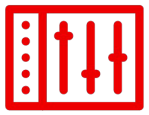
- Deploy and run **serverless containers**
- Use any programming language or runtime
- Modernize existing applications to run serverless
- Powered by a rich ecosystem of event sources
- Manage serverless apps natively in Kubernetes
- Based on open source project, **Knative**
- Run anywhere OpenShift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



* 4.6 is the target, this could change!

Red Hat

Serverless Themes

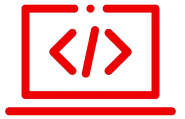


Monitoring and Automation

Powerful monitoring capabilities with configuration and automation for GitOps and modern CI/CD practices.

Integrations and Ecosystem

Eventing capabilities enabling a rich ecosystem of Event Sources from Red Hat and Partner products.



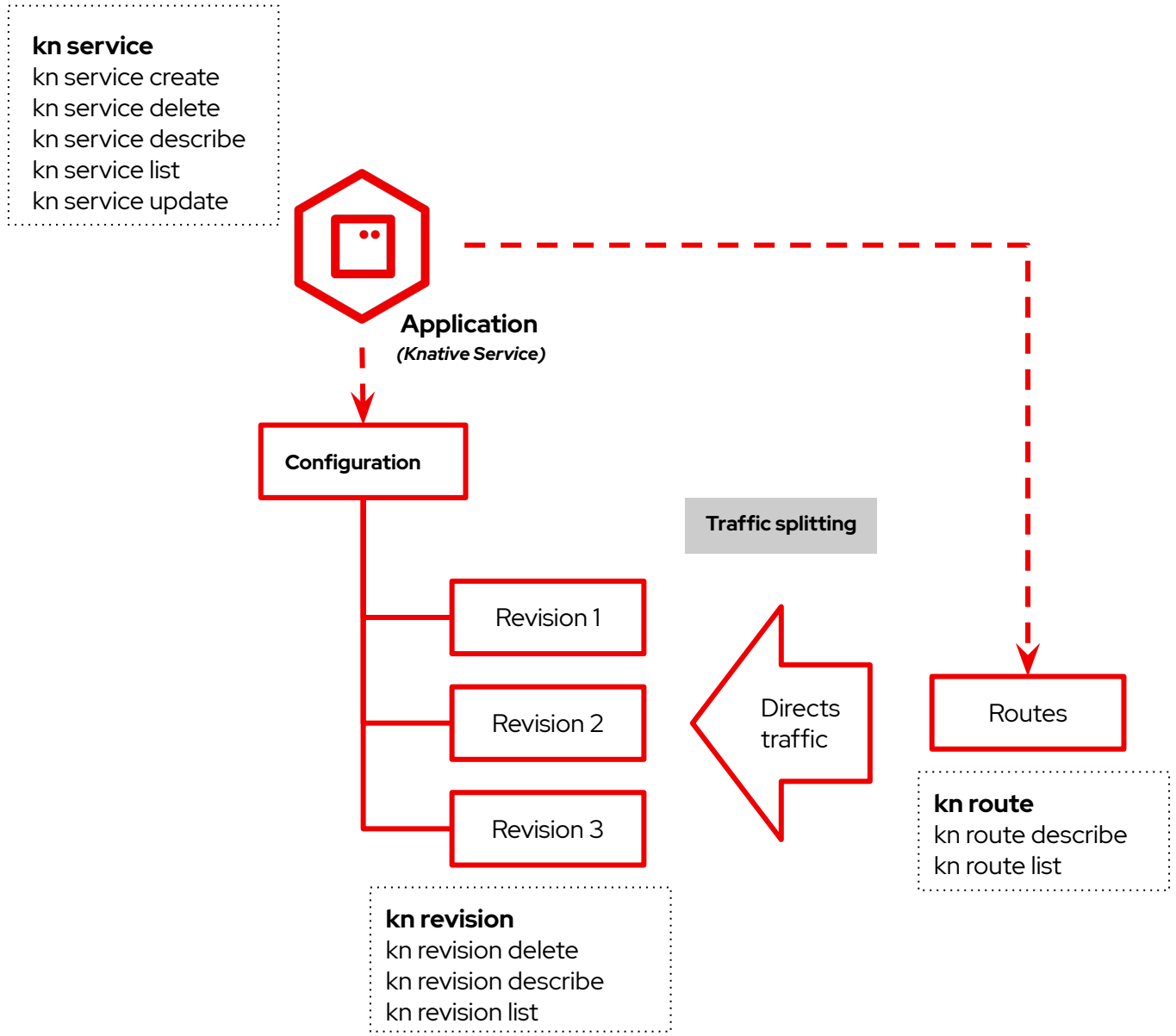
Developer Experience

Intuitive developer experience through the Developer Console and CLI/IDE with Functions support.

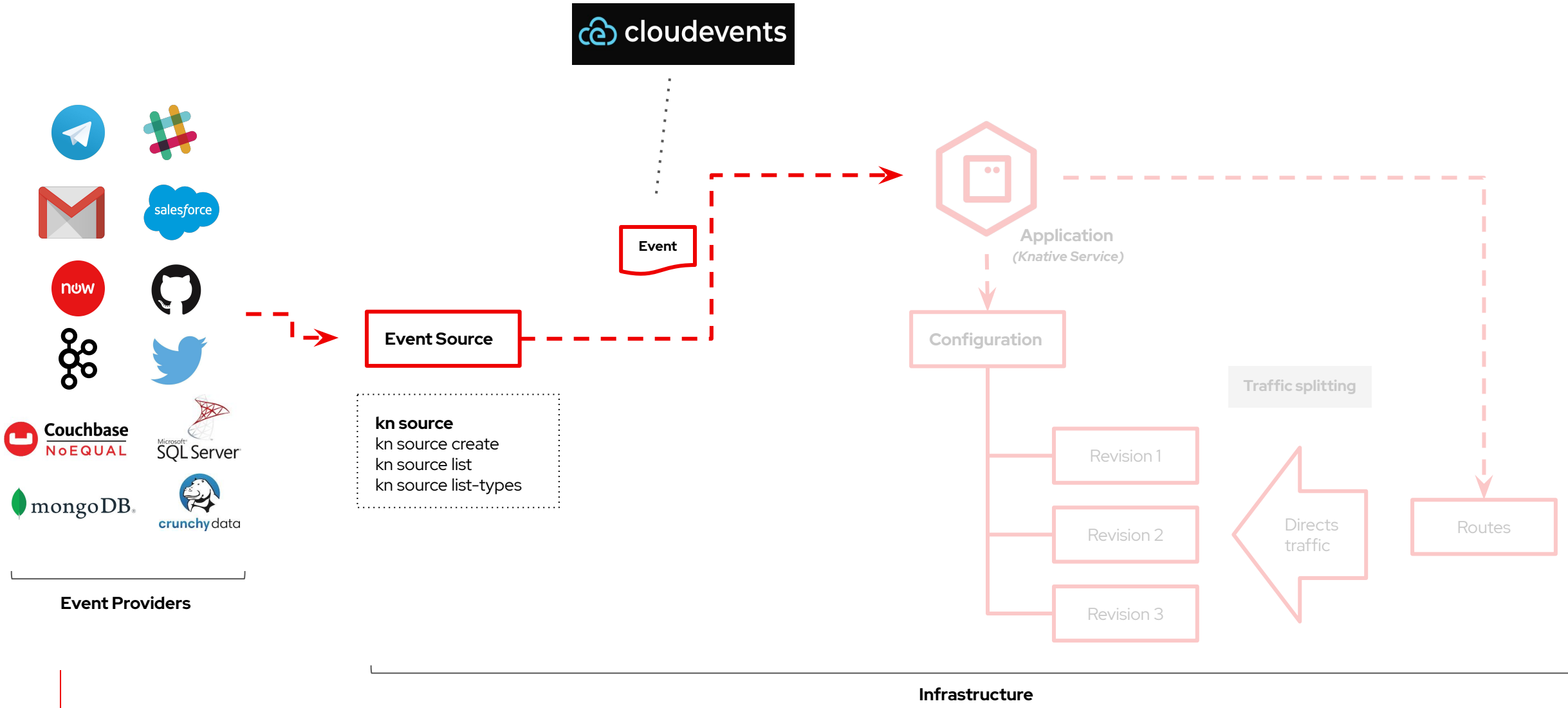
Serving

- From container to URL within seconds
- Easier developer experience for Kubernetes
- Built-in versioning, traffic split and more
- Simplified Installation experience with Kourier *new*
- Automatic TLS/SSL for Applications *new*

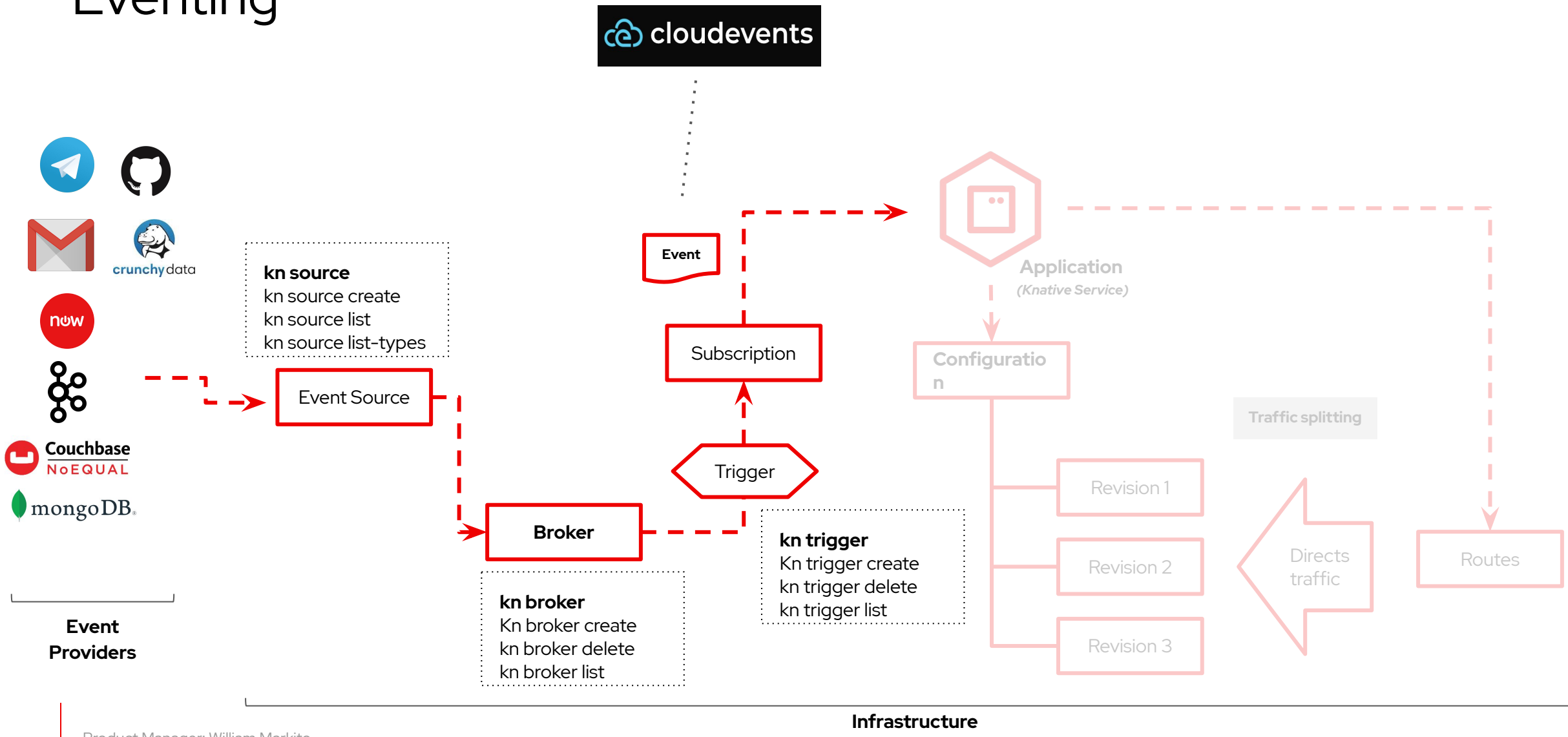
```
$ kn service create --image=<container>
```



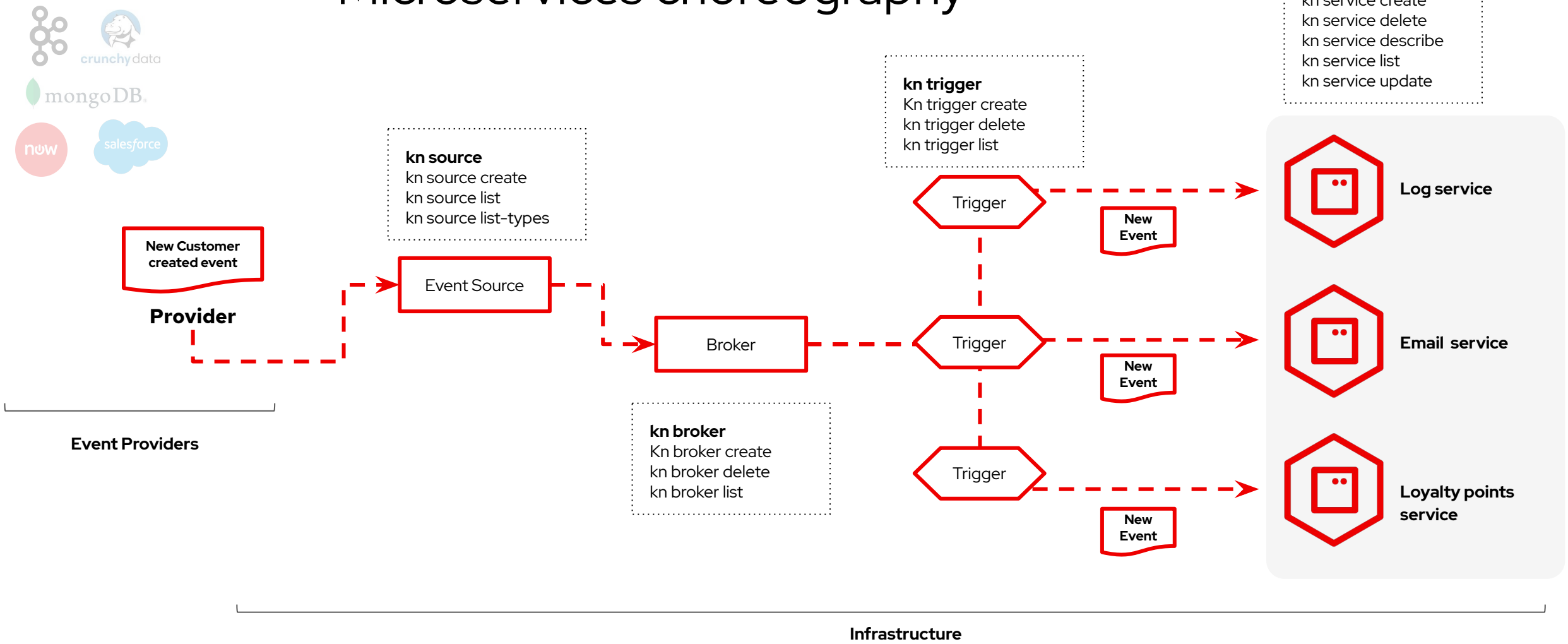
Eventing


cloudevents
Event
Application
(Knative Service)
Configuration
Event Source
kn source
kn source create
kn source list
kn source list-types
Traffic splitting
Revision 1
Revision 2
Revision 3
Directs traffic
Routes
Couchbase
NoEQUAL
Microsoft SQL Server
mongoDB.
crunchy data
salesforce
Event Providers
Infrastructure

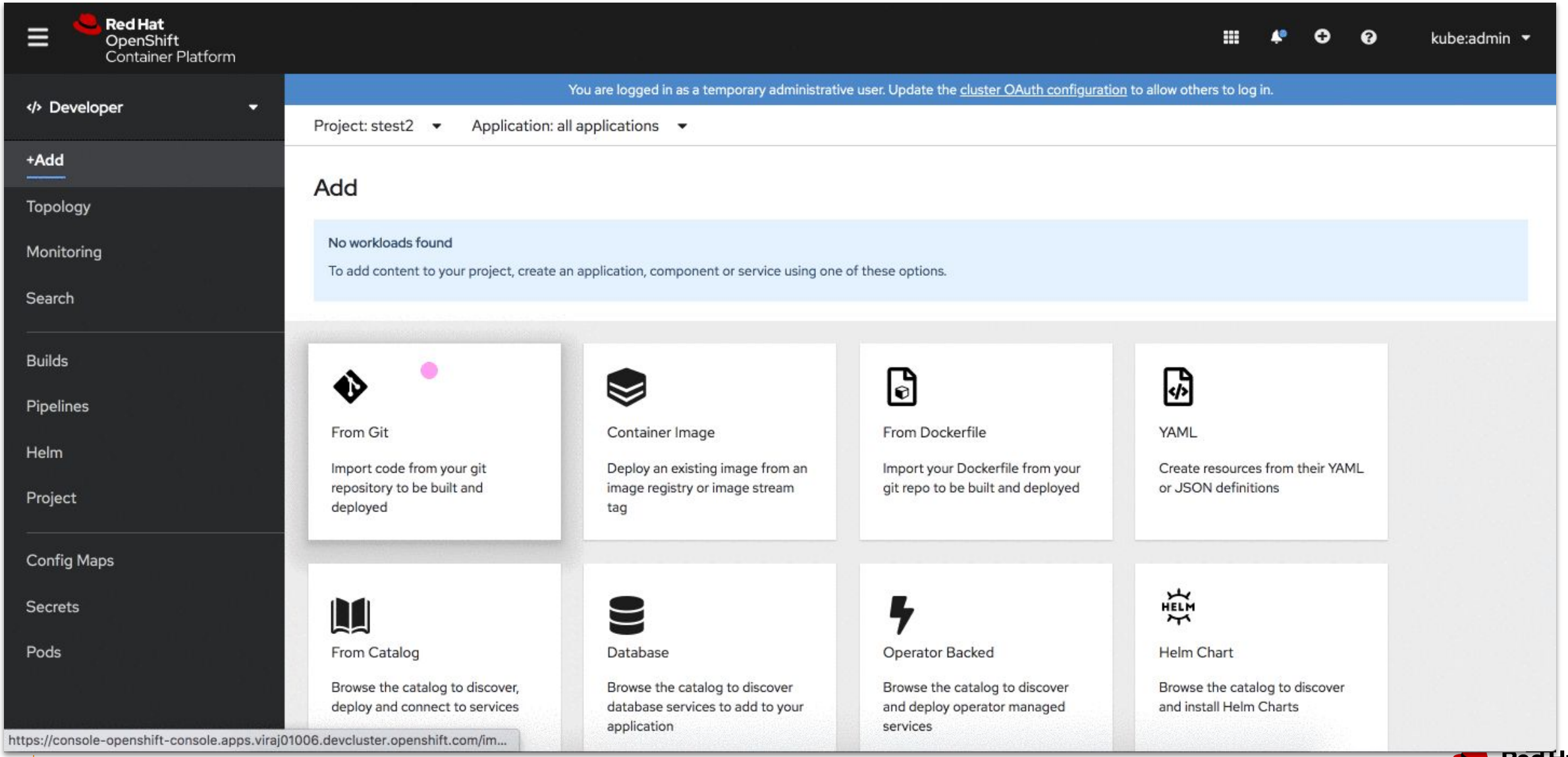
Eventing


cloudevents
Event
Application
(Knative Service)
Configuratio
n
Subscription
Trigger
Event Source
Broker
kn source
kn source create
kn source list
kn source list-types
kn trigger
Kn trigger create
kn trigger delete
kn trigger list
kn broker
Kn broker create
kn broker delete
kn broker list
Traffic splitting
Revision 1
Revision 2
Revision 3
Directs
traffic
Routes
crunchydata
Couchbase
NoEQUAL
mongoDB.
Event
Providers
Infrastructure

Microservices choreography



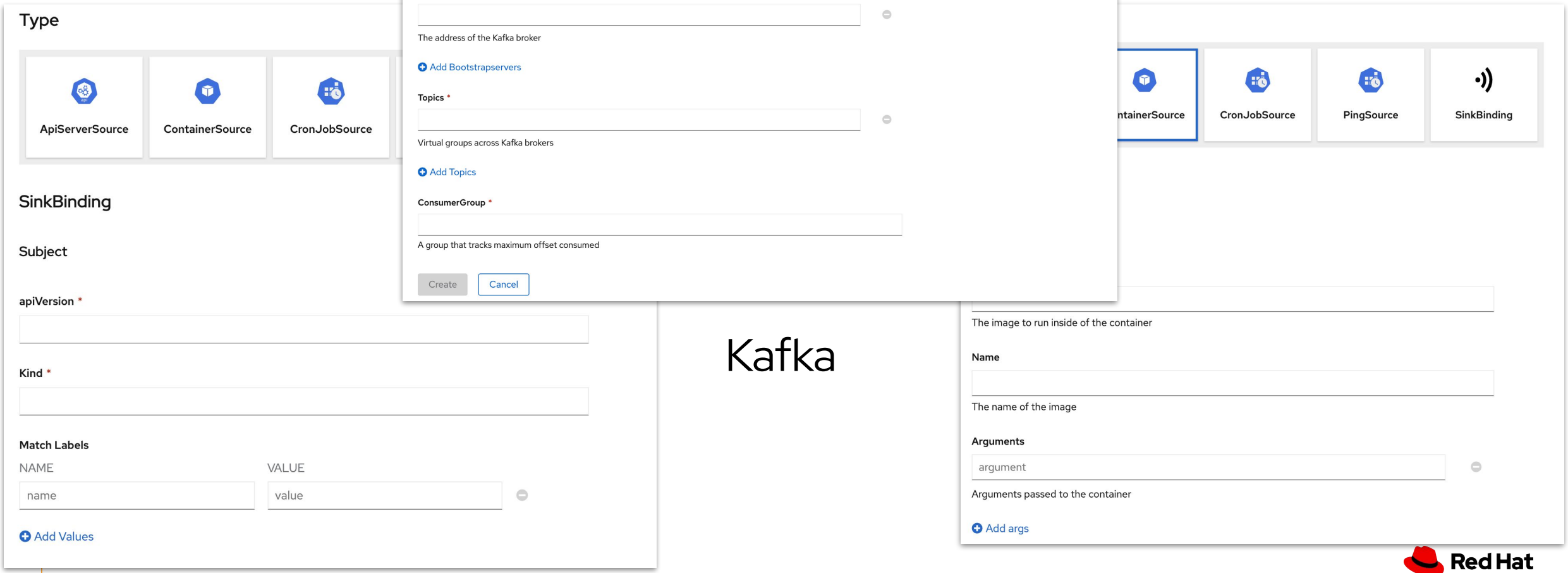
Event Sources in the Developer Console



Sink Binding

Container Source

Kafka

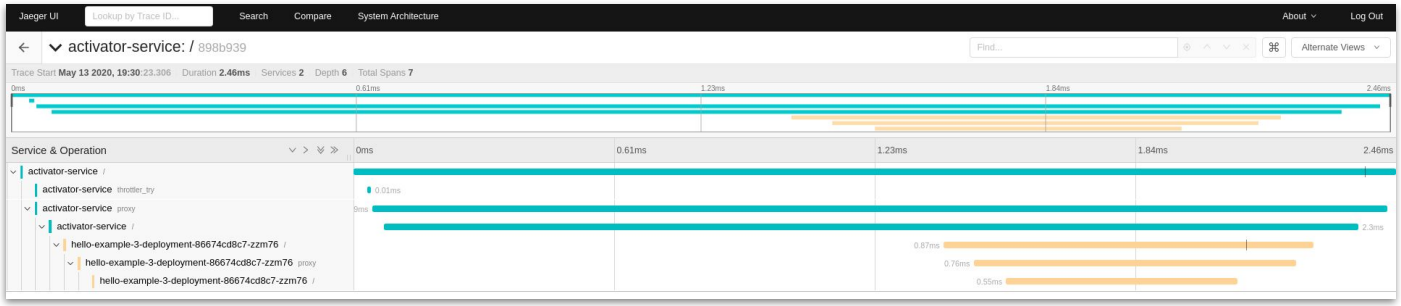


Product Manager: William Markito

Red Hat

Developer Experience

Jaeger Support [2]



GPU Support [1] for Serverless Applications

```
kn service create hello --image \ docker.io/knativesamples/hellocuda-go
--limit nvidia.com/gpu=1
```

NVIDIA

Product Manager: William Markito

[1] https://docs.nvidia.com/datacenter/kubernetes/openshift-on-gpu-install-guide/index.html
[2] https://docs.openshift.com/container-platform/4.4/serverless/serverless-tracing.html

Red Hat

Next Steps:
Your Technology Radar for Event-Driven and Serverless

- Service Mesh (Istio):
  - Provide microservice interconnectivity and visibility
- Serverless platforms (Knative)
  - Container build and on-demand scheduling
- Container-native frameworks (Quarkus)
  - Optimize Java workloads for serverless architecture

Red Hat

Next Steps:
Your Technology Radar for Event-Driven and Serverless

- Strimzi
  - Kafka operator for Kubernetes/OpenShift
- EnMasse
  - Messaging-as-a-Service for Kubernetes/OpenShift
- FaaS frameworks (e.g. Camel-K)
  - Schedule integration code directly on platform or via Knative

Red Hat

Next Steps:
Red Hat's Technology Radar for Event-Driven and Serverless

Red Hat AMQ

MESSAGING BACKBONE

Red Hat Data Grid

DISTRIBUTE, REACT ON DATA

Red Hat Application Runtimes

REACTIVE / FAAS FRAMEWORKS

Red Hat Middleware Portfolio

RULES EVALUATION
COMPLEX EVENTS
AUTOMATION

Red Hat Integration

CAMEL K
REACTIVE INTEGRATION
SERVERLESS

Red Hat CodeReady

REACTIVE DEVELOPER
TOOLING

Red Hat OpenShift

APP ENVIRONMENT
INFRASTRUCTURE
SERVERLESS / KNATIVE
OPERATOR HUB

Next Steps: Resources

Knative Tutorial on Red Hat Developer

Knative Cookbook on Red Hat Developer

OpenShift Serverless Tech Topic

Red Hat Services Overview of Serverless Blog

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

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat