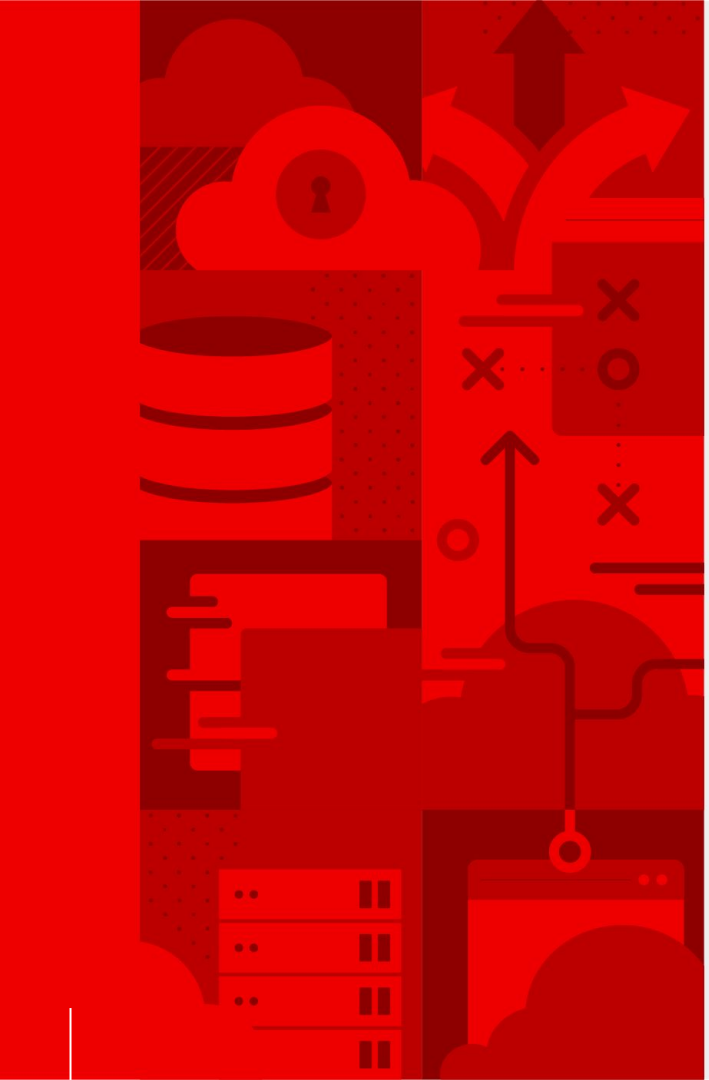


OpenShift Developer

Quarkus, Service Mesh, Serverless

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat

Agenda / etc.

Agenda

- Quarkus
- Service Mesh (istio.io)
- Serverless (knative)

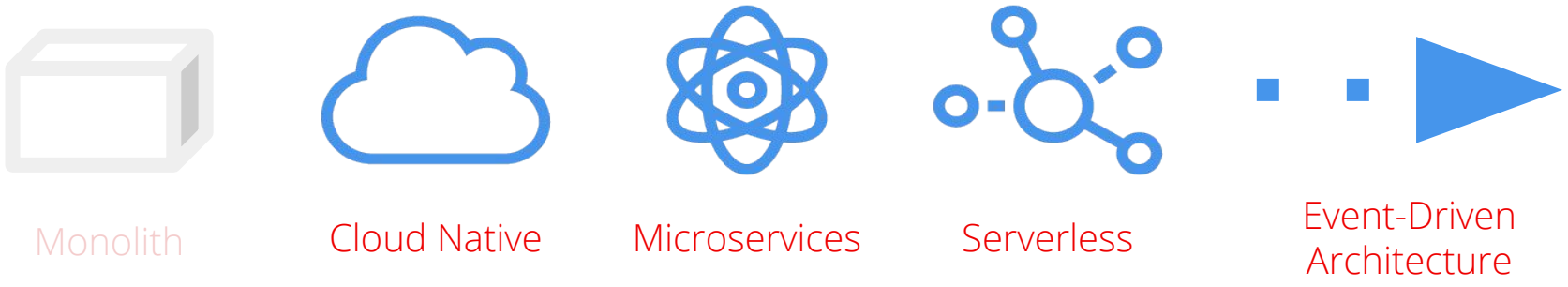
Quarkus Business Value

QUARKUS

Supersonic. Subatomic. Java.

Red Hat

Quarkus - Kubernetes Native Java



Red Hat

WHAT IS QUARKUS?

QUARK: elementary particle / **US**: hardest thing in computer science

Quarkus Business Value

“Supersonic, Subatomic Java”

Cost Savings

Cloud efficiency (low memory, fast startup, high density), serverless deployments

Faster Time to Market

Developer productivity, extensions ecosystem, low learning curve, keep competitive edge

Reliability

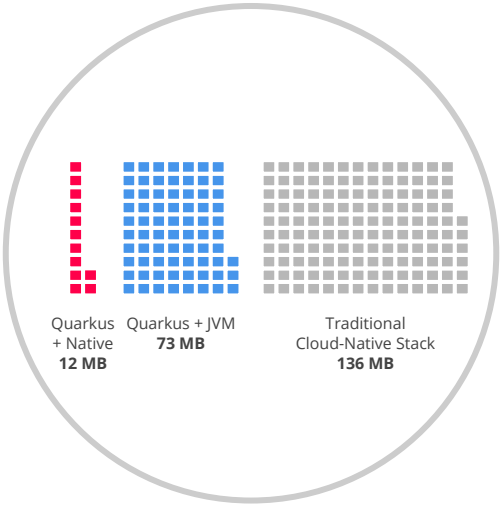
Trusted technology, active community, trusted sponsor, proven Java libraries and standards

Red Hat

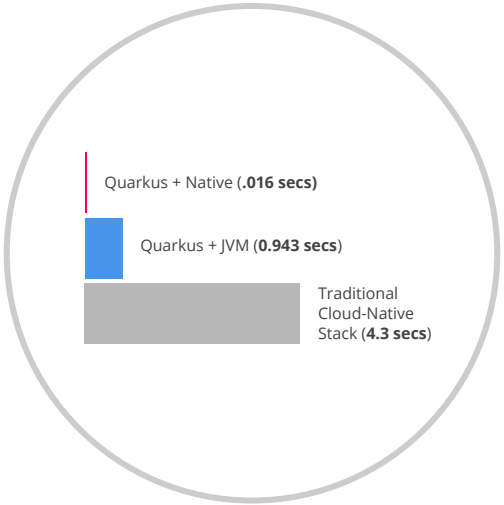
Cost Savings

"both jvm & native compilation modes"

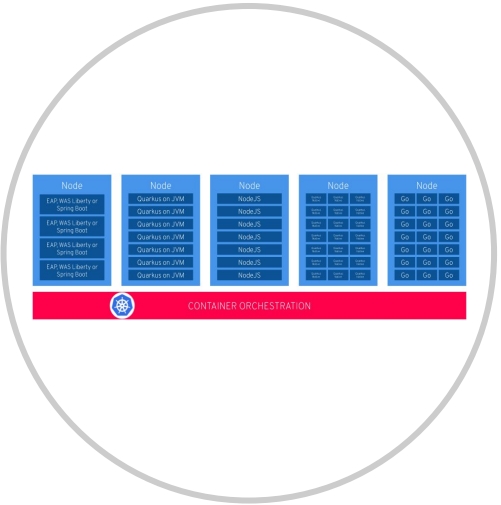
Reduced Memory Footprint



Fast Startup Time



Smaller Disk Footprint



Red Hat

Cost Savings

"from the mouths of production users"

*We could run **3x** denser deployments without sacrificing availability and response times of service.*

-Lufthansa Technik

*We went from **1 min** startup times to **400 ms**.*

-Suomen Asiakastieto Oy

*We went from **6-7 sec** startup time with plain Java on AWS Lambda to **10 ms** with Quarkus.*

-Ennovative Solutions

Red Hat

Faster Time to Market

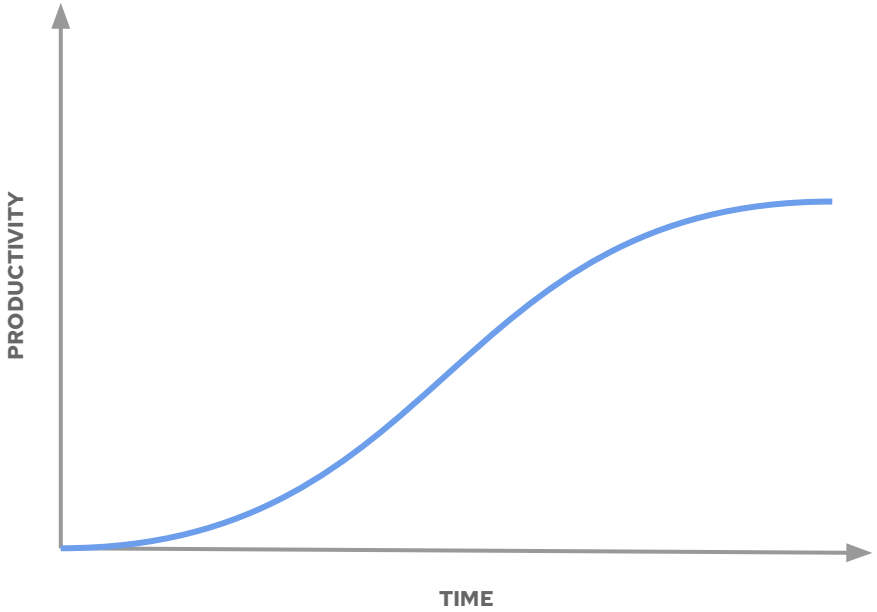
"Platform that feels familiar but new"

Quarkus allows developers to leverage their existing Java expertise to reduce their time to productivity. It feels familiar and innovative as it unlocks the ability to utilize their Java skills to build modern, cloud-native microservices and serverless applications.

Derived from Java standards like Jakarta EE and Eclipse MicroProfile

Crafted from well known Java libraries such as Hibernate and Spring

Developer tooling and extension library to build applications quicker



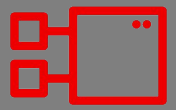
Red Hat

Why Quarkus?

"Quarkus is a great fit if you are looking for..."

IMPROVED APP PERFORMANCE & COST

- ❑ Reduce memory consumption
- ❑ Increase startup speeds
- ❑ Reduce application size
- ❑ Looking for Spring or Java alternatives



DIGITAL TRANSFORMATION

- ❑ Modernize or optimize existing applications
- ❑ Move to a cloud-native or microservices architecture
- ❑ Need lightweight connectivity between services
- ❑ Next generation business automation

SERVERLESS

- ❑ Have seldom used but critical services
- ❑ Interested in serverless or FaaS (function as a service)

Use Cases

"Quarkus is an ideal runtime for"

NET NEW

Low memory footprint + lightning fast startup time + small disk footprint = an ideal runtime for Kubernetes-native microservices

MONO 2 MICRO

Quarkus is a great choice to modernize existing monolithic applications by breaking it into smaller, loosely coupled microservices.

SERVERLESS

Scaling up or down (0) is extremely fast with Quarkus making it an ideal runtime for creating serverless applications.

EVENT-DRIVEN/REACTIVE

Quarkus utilizes an asynchronous, reactive event loop that makes it easy to create reactive applications.

Quarkus Technical Value

"Historical" Enterprise Java Stack

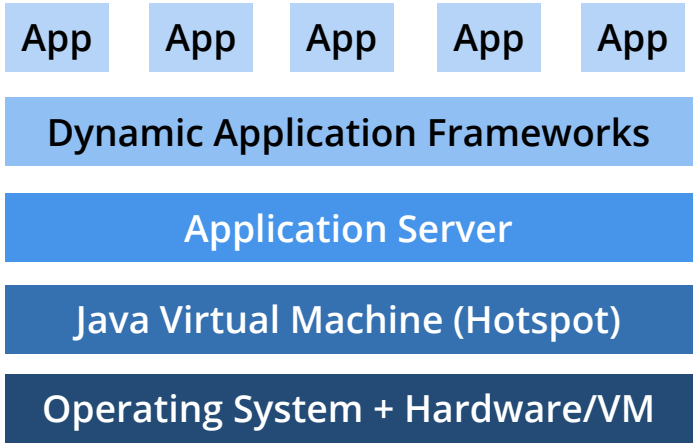
Architecture: Monoliths

Deployment: multi-app, appserver

App Lifecycle: Months

Memory: 1GB+ RAM

Startup Time: 10s of sec



"Modern" Enterprise Java Stack

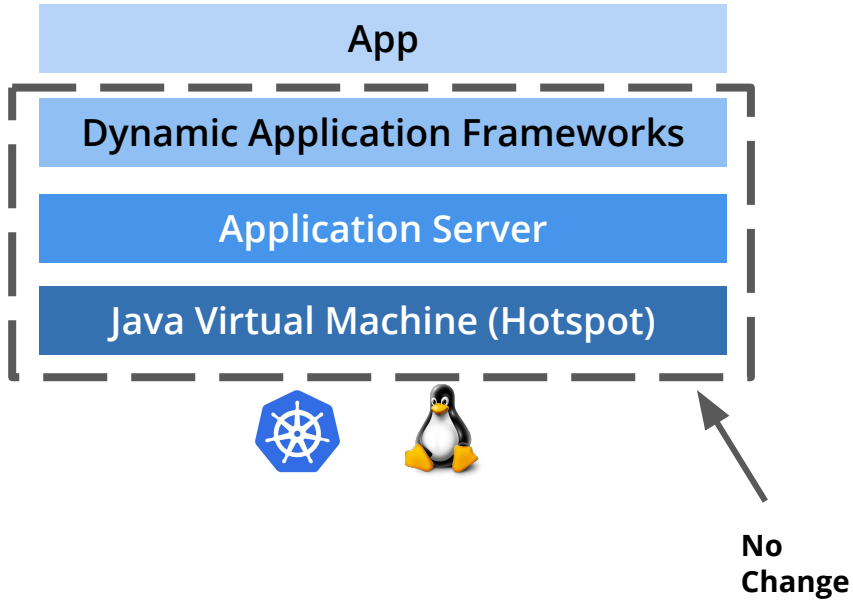
Architecture: Microservices

Deployment: Single App

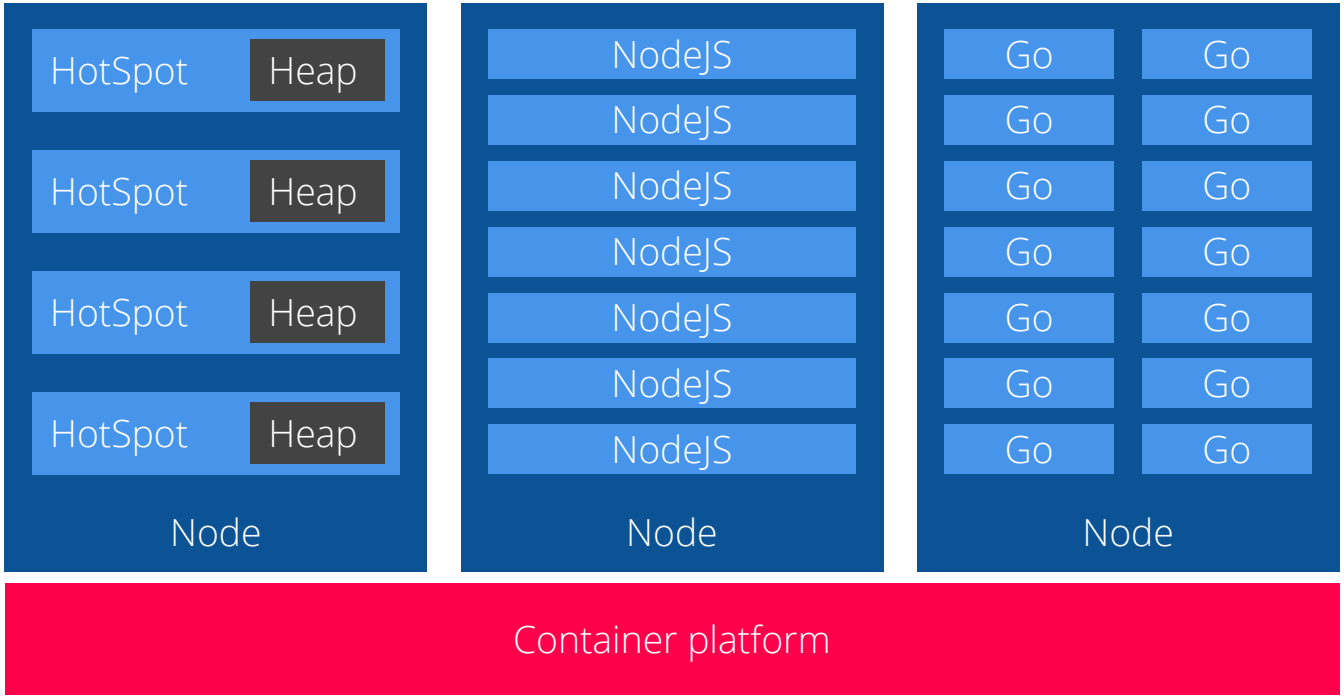
App Lifecycle: Days

Memory: 100MBs+ RAM

Startup Time: Seconds



Hidden Truth About Java + Containers



Red Hat

THERE IS A NEED FOR A NEW JAVA STACK FOR CLOUD-NATIVE AND SERVERLESS

Experts from cloud-native Java OS projects

Eclipse Vert.x | Hibernate | RESTEasy | Eclipse MicroProfile | WildFly | Undertow | OpenJDK

QUARKUS

Differentiators

Container First

Tailors your app for HotSpot & GraalVM

Fast boot time and low RSS memory

Serverless fit

Developer Joy

Live coding

Unified configuration

Unifies Imperative & Reactive

Combines blocking and non-blocking

Built-in event bus

Best of Breed Libraries & Standards

90+ extensions

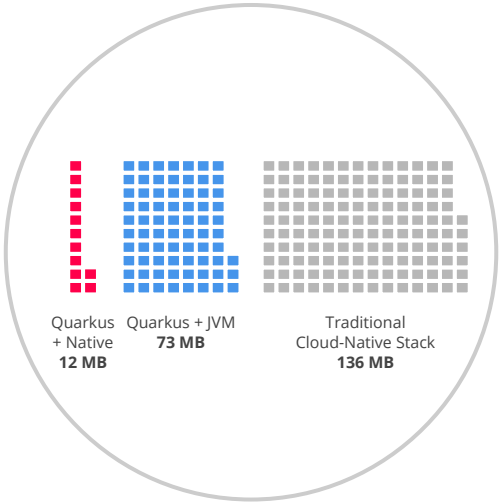
“Powered by Quarkus” applications

Red Hat

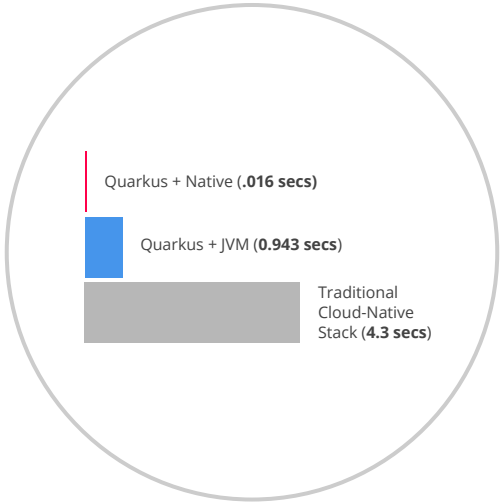
Benefit No. 1: Container First

*"We went from **1-min** startup times to **400 milliseconds**"*

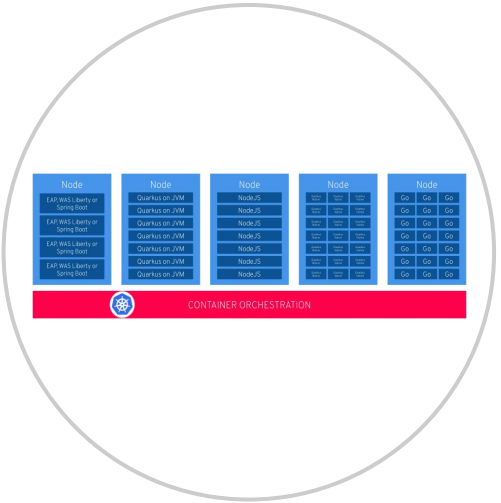
Reduced Memory Footprint



Fast Startup Time



Smaller Disk Footprint



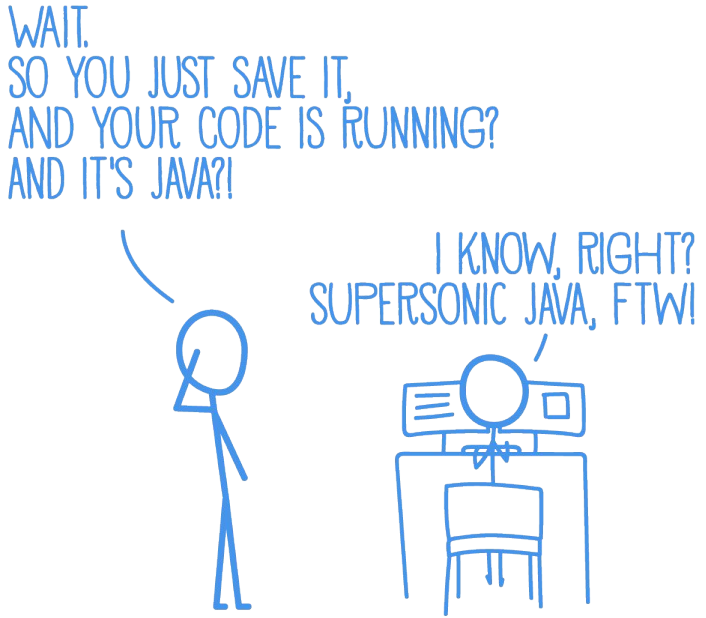
Red Hat

Benefit No. 2: Developer Joy

*"Our developers used to wait **2 to 3 mins** to see their changes. **Live coding** does away with this."*

A cohesive platform for optimized developer joy:

- Based on standards and more
- Unified configuration
- Live coding
- Streamlined code for the 80% common usages, flexible for the 20%
- No hassle native executable generation



Red Hat

Benefit No. 3: Unifies Imperative and Reactive

```
@Inject
SayService say;

@GET
@Produces(MediaType.TEXT_PLAIN)
public String hello() {
      return say.hello();
}
```

```
@Inject @Stream("kafka")
Publisher<String> reactiveSay;

@GET
@Produces(MediaType.SERVER_SENT_EVENTS)
public Publisher<String> stream() {
      return reactiveSay;
}
```

- Combine both Reactive and imperative development in the same application
- Inject the EventBus or the Vertx context
- Use the technology that fits your use-case
- Key for reactive systems based on event driven apps

Benefit No. 4: Best of Breed Frameworks & Standards

"When you adopt Quarkus, you will be productive from day one since you don't need to learn new technologies."

Eclipse Vert.x

Hibernate

RESTEasy

Apache Camel

Eclipse MicroProfile

Netty

Kubernetes

OpenShift

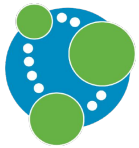
Jaeger

Prometheus

Apache Kafka

Infinispan

Flyway



Neo4j

MongoDB

MQTT

KeyCloak

Apache Tika

Red Hat

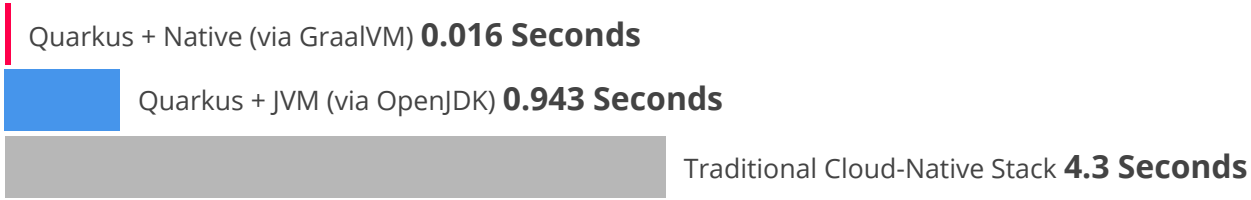
Supersonic, Subatomic

Fast.
Blazing fast to start.
Millisecond fast!

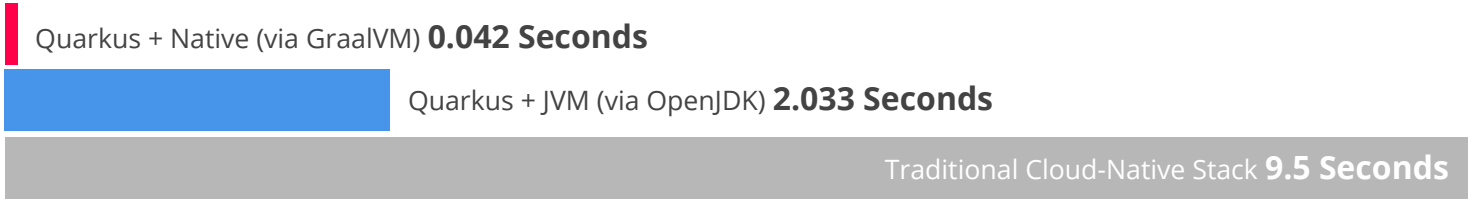
Red Hat

Supersonic, Subatomic Java

REST



REST + CRUD

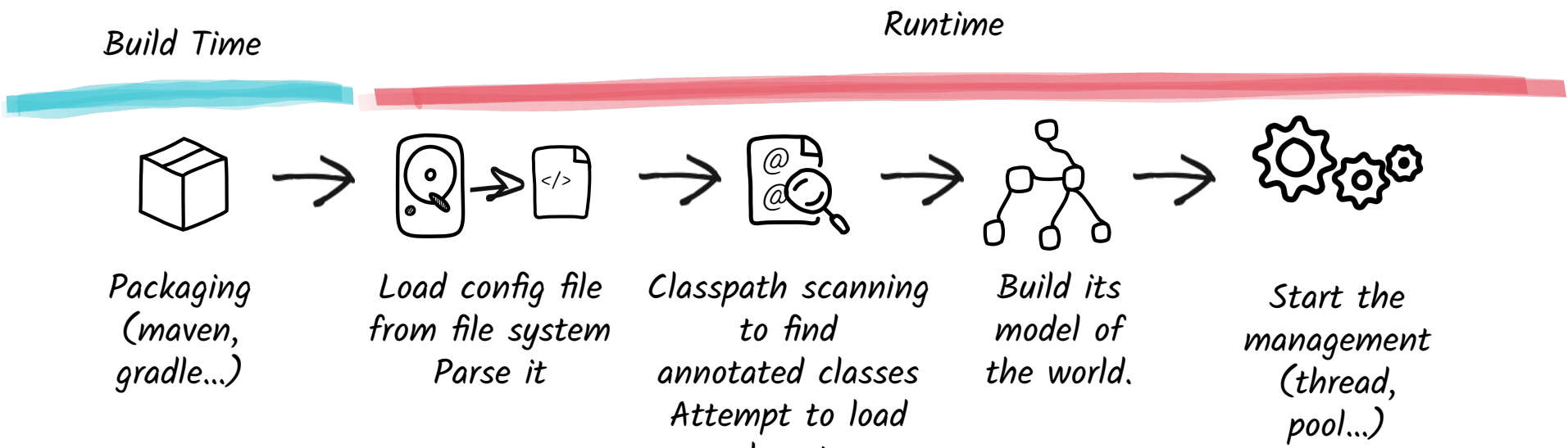


Time to first response

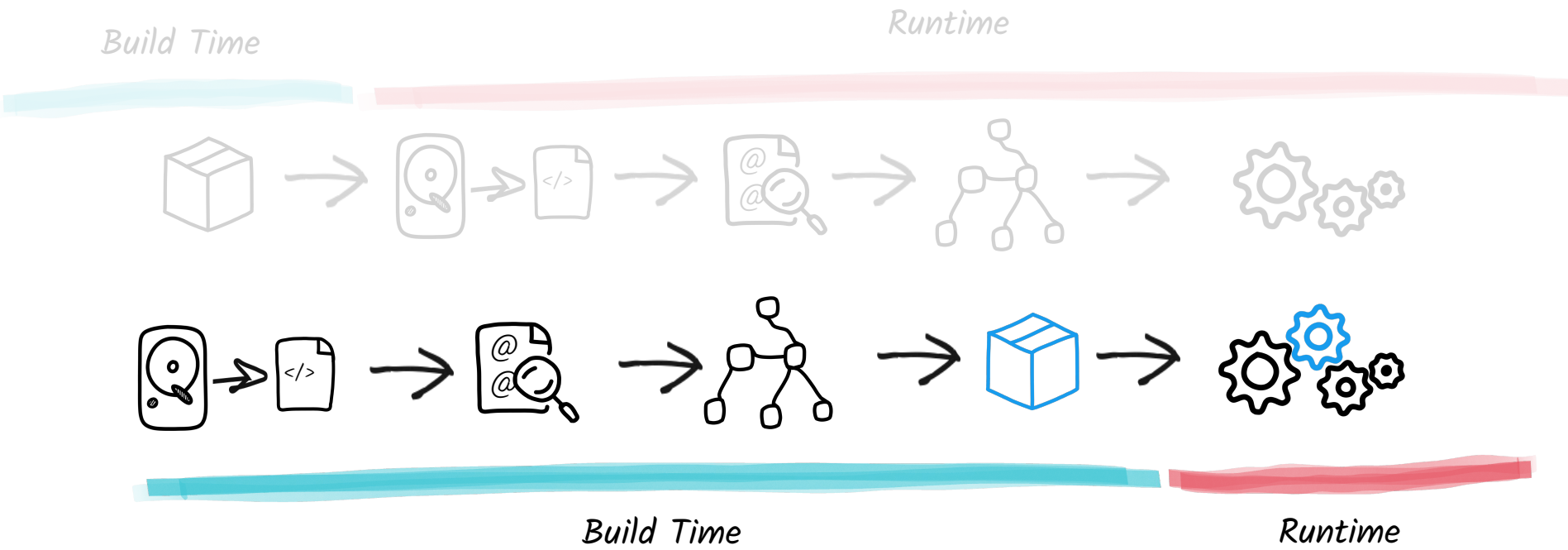
Red Hat

HOW DOES QUARKUS WORK?

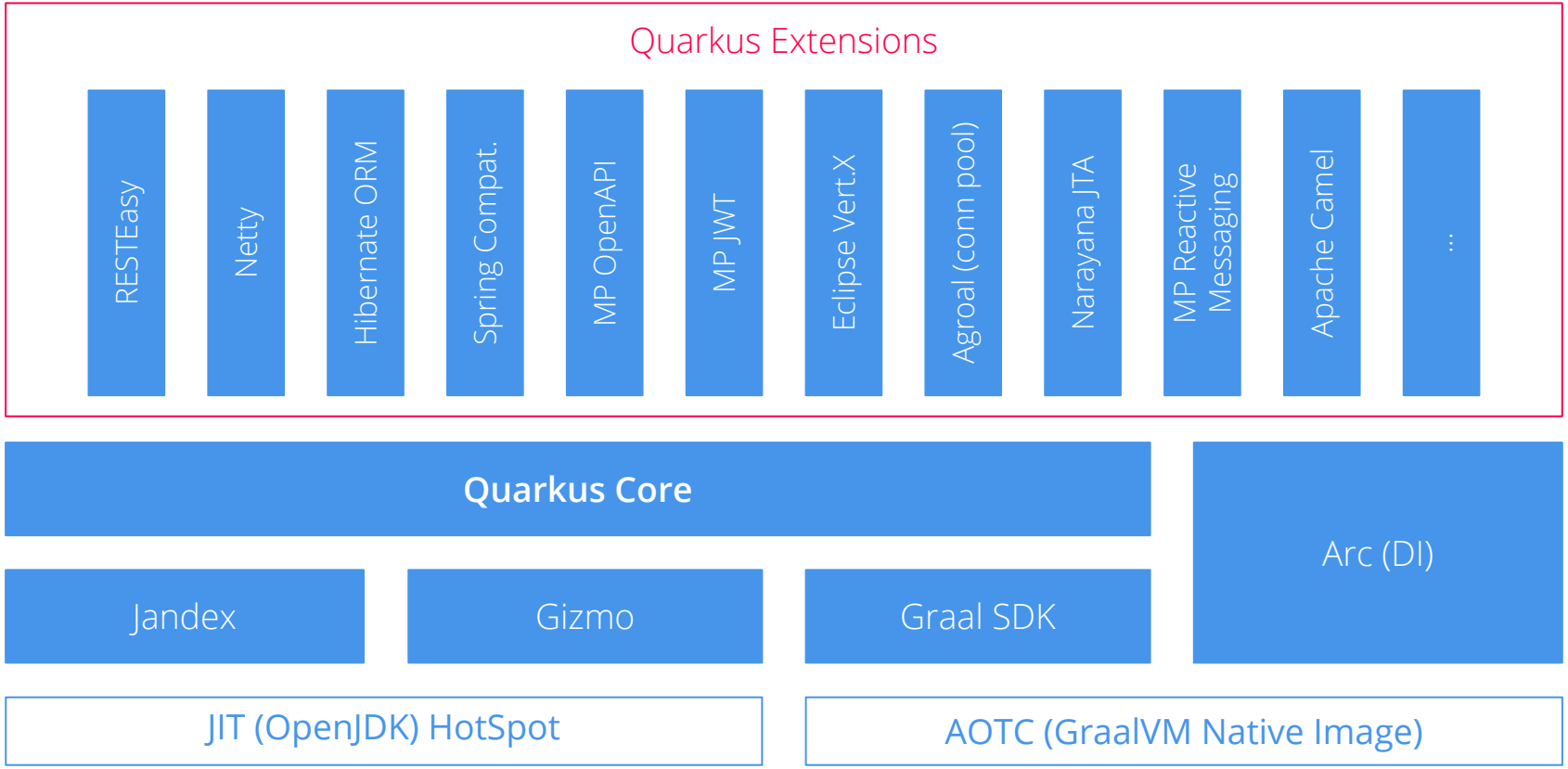
How Does a Framework Start?



The Quarkus Way



An ahead-of-time, build-time, runtime


Quarkus Extensions
RESTEasy
Netty
Hibernate ORM
Spring Compat.
MP OpenAPI
MP JWT
Eclipse Vert.X
Agroal (conn pool)
Narayana JTA
MP Reactive Messaging
Apache Camel
...
Quarkus Core
Arc (DI)
Jandex
Gizmo
Graal SDK
JIT (OpenJDK) HotSpot
AOTC (GraalVM Native Image)

Red Hat

The Right VM For the Right Deployment

JIT (OpenJDK HotSpot)

High memory density requirements
High request/s/MB
Fast startup time

Best raw performance (CPU)
Best garbage collectors
Higher heap size usage

Known monitoring tools
Compile Once, Run anywhere
Libraries that only works in standard JDK

AOT (GraalVM native image)*

Highest memory density requirements
Highest request/s/MB
    for low heap size usages
Faster startup time
    10s of ms for Serverless

*Currently in Tech Preview

Red Hat

Quarkus Tools - Build


```
mvn io.quarkus:quarkus-maven-plugin:1.3.2.Final-redhat-00001:create \
    -DprojectGroupId=org.acme \
    -DprojectArtifactId=getting-started \
    -DplatformGroupId=com.redhat.quarkus \
    -DplatformVersion=1.3.2.Final-redhat-00001 \
    -DclassName="org.acme.quickstart.GreetingResource" \
    -Dpath="/hello"
cd getting-started
```

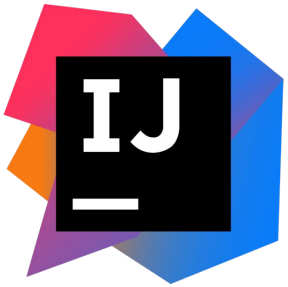
*community supported

Red Hat

Quarkus Tools - IDE

VSCode

Eclipse



IntelliJ

che.openshift.io

DEMO

https://github.com/wpernath/quarkus-worldtour

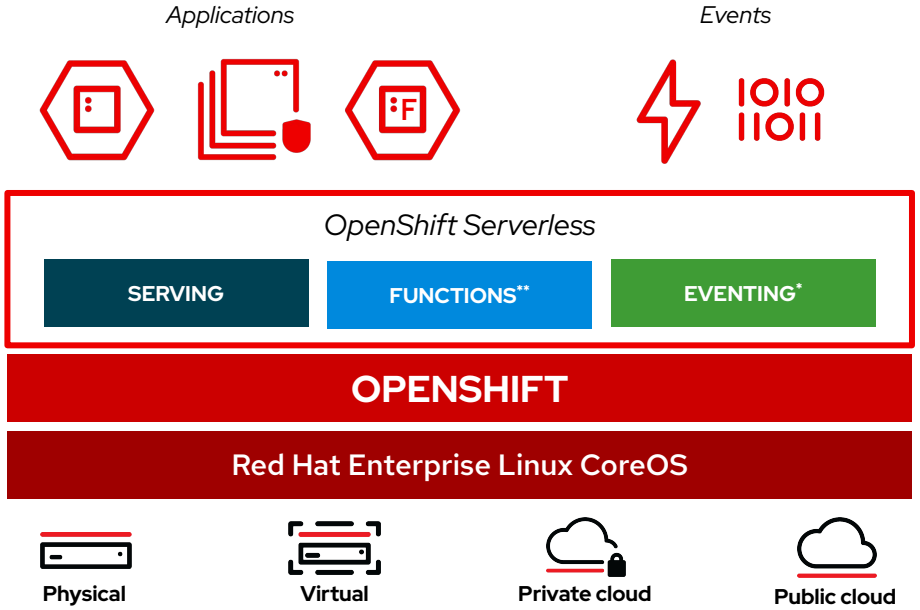
Optional - Serverless / knative

Generally Available

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

Red Hat

OpenShift Serverless

Key Features

Containers made easy

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

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.

Immutable revisions

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

Any programming language

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

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.

Event Driven Architectures

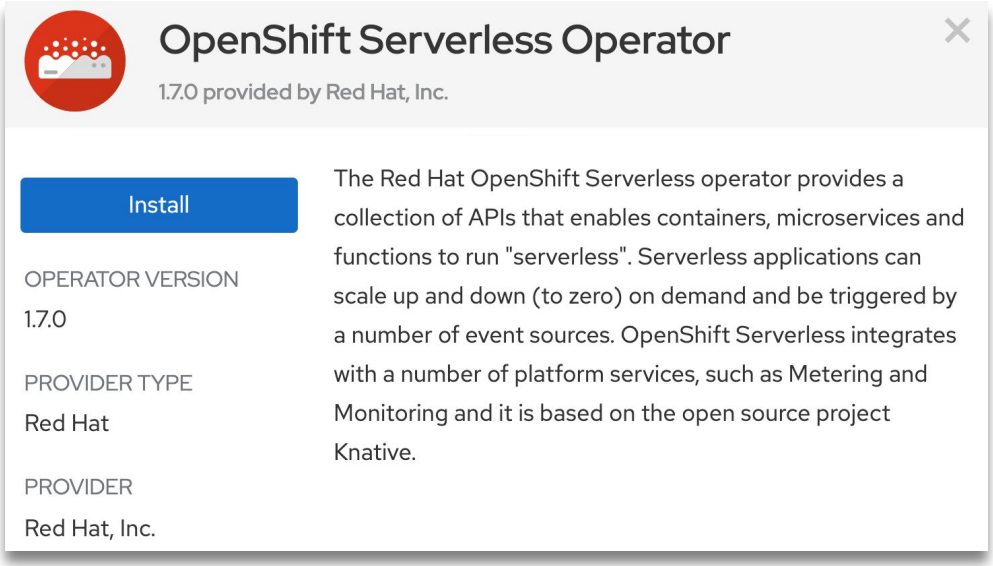
Build loosely coupled & distributed apps connecting with a variety of built-in or third-party event sources or connectors powered by Operators.

Red Hat

Installation experience

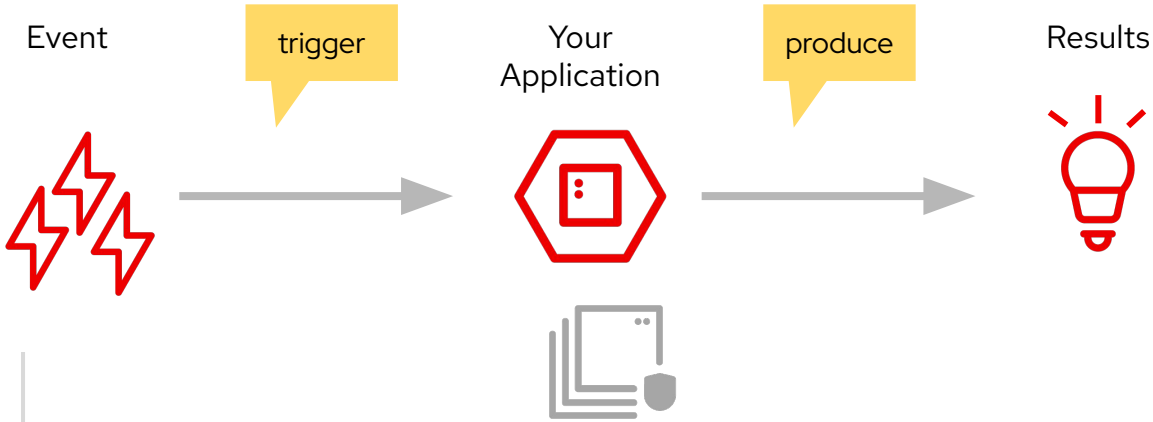
"Easy day 1 and even better for day 2"

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



Red Hat

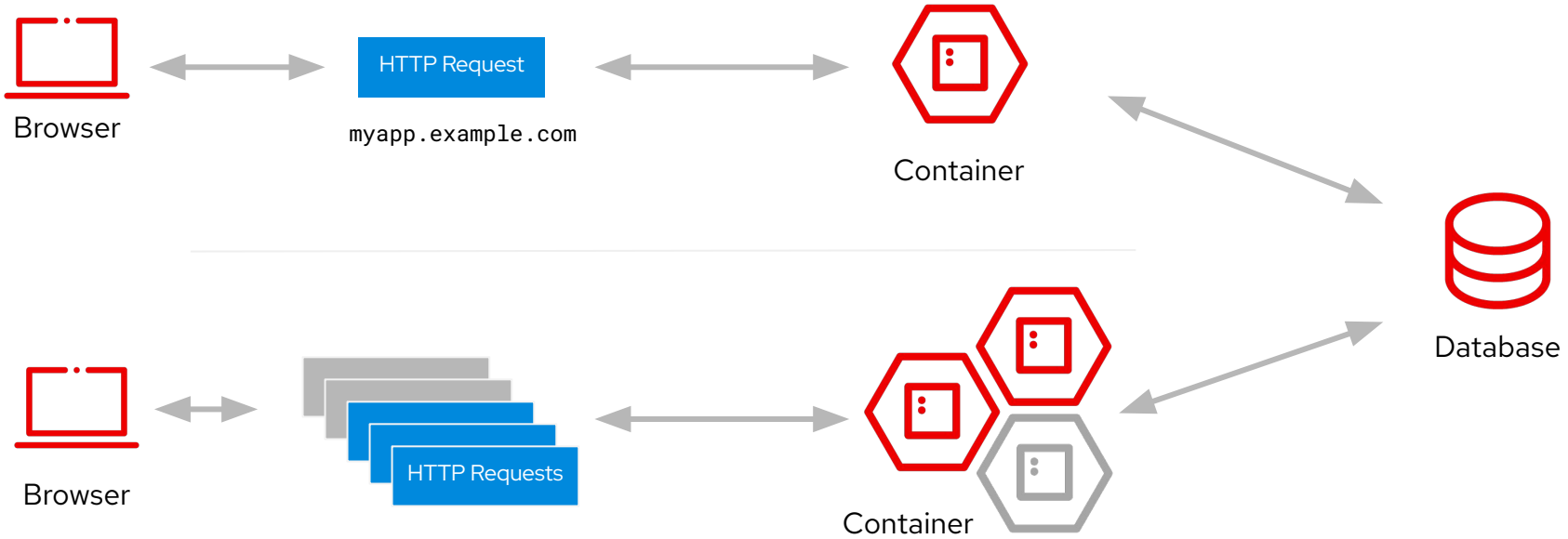
The "Serverless Pattern"



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

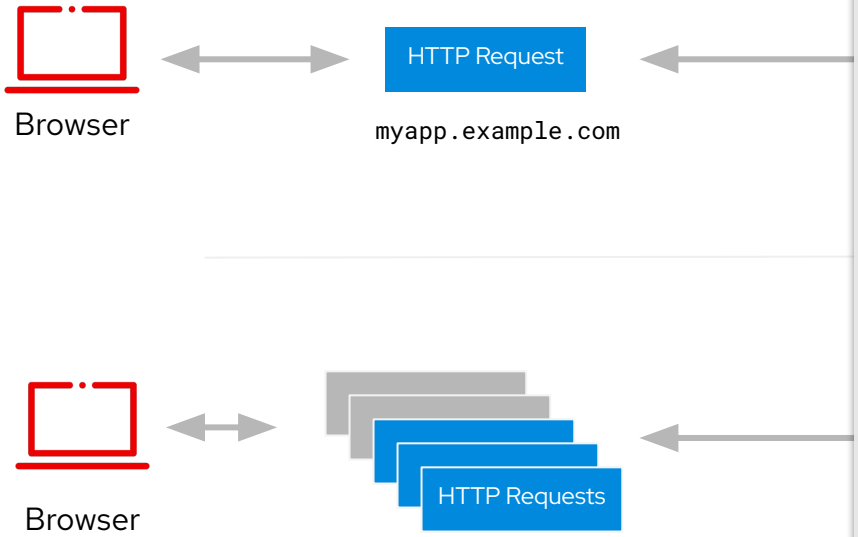
The "Serverless Pattern"

A serverless web application



The "Serverless Pattern"

A serverless web application

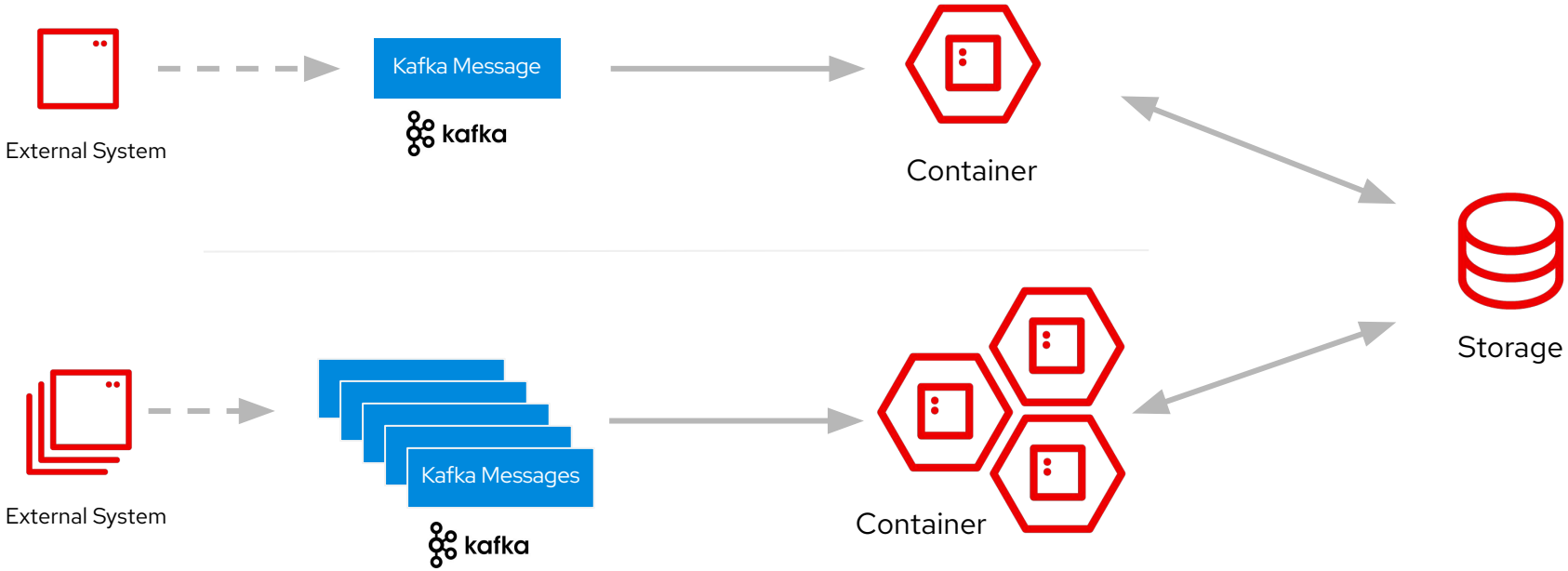


Benefits of this model:

- 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"

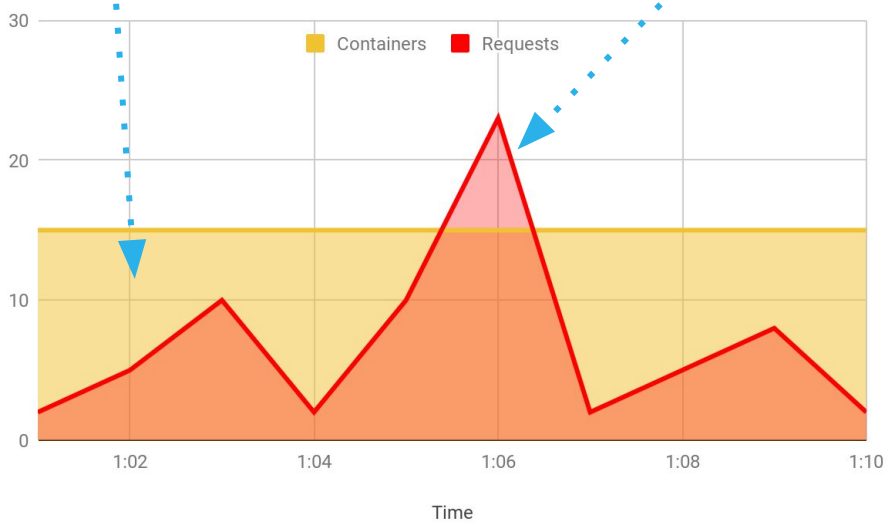
Processing a Kafka message



Serverless Operational Benefits

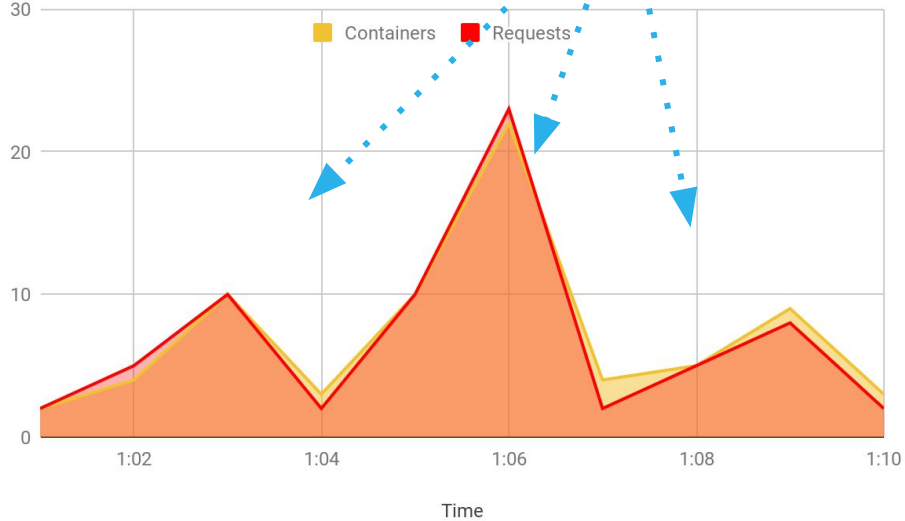
Over provisioning
Time in capacity planning
IT cost of idle resources

Under provisioning
Lost business revenue
Poor quality of service



NOT Serverless

More applications
Direct line between IT
costs & business revenue



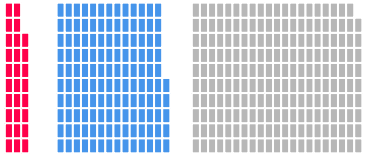
with Serverless

Choosing the Right Tool

Your Team

The Ecosystem

Performance

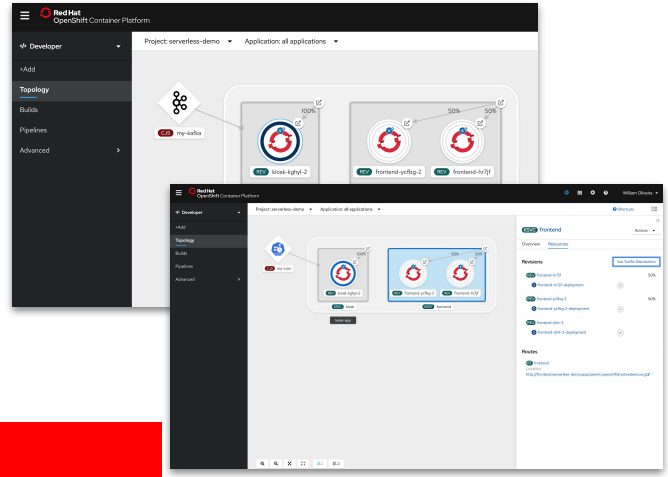


LOGIC

Developer experience

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

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



CLI

UI

OpenShift Serverless

OPENSHIFT

Red Hat Enterprise Linux CoreOS

Physical

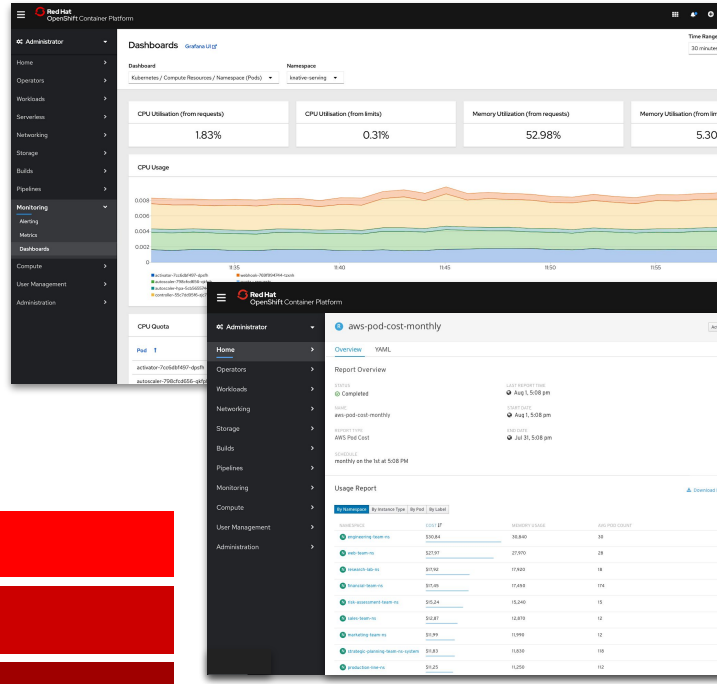
Virtual

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



UI

OpenShift Serverless

OPENSHIFT

Red Hat Enterprise Linux CoreOS

Physical | Virtual | Private cloud | Public cloud

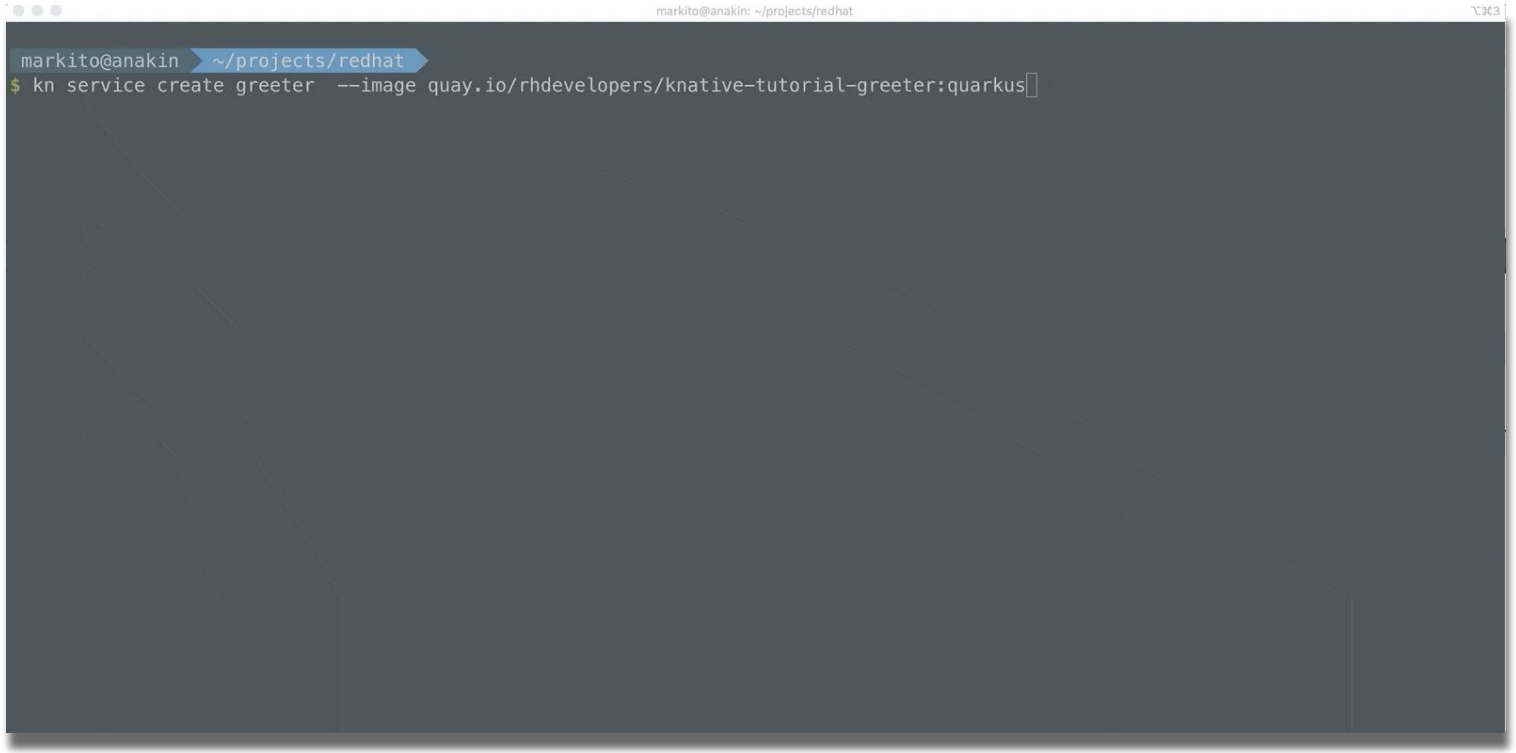
Command Line Experience

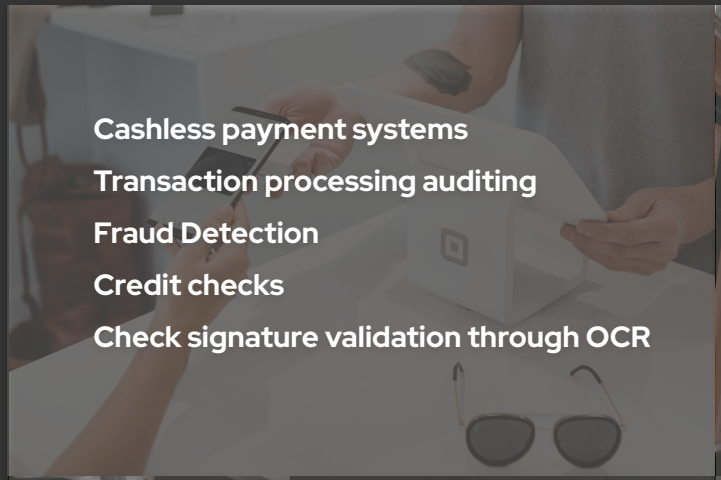
```
$ kn service create myService --image=[registry/mycontainer:v1] --min-scale=1 --max-scale=100
```

```
$ 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!





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

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

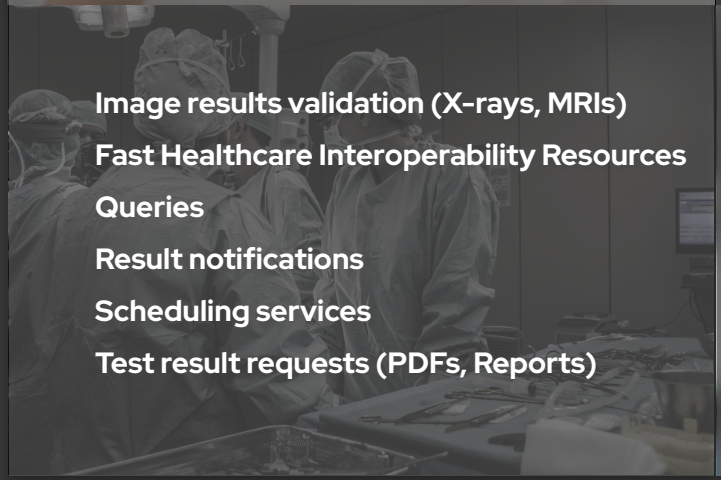
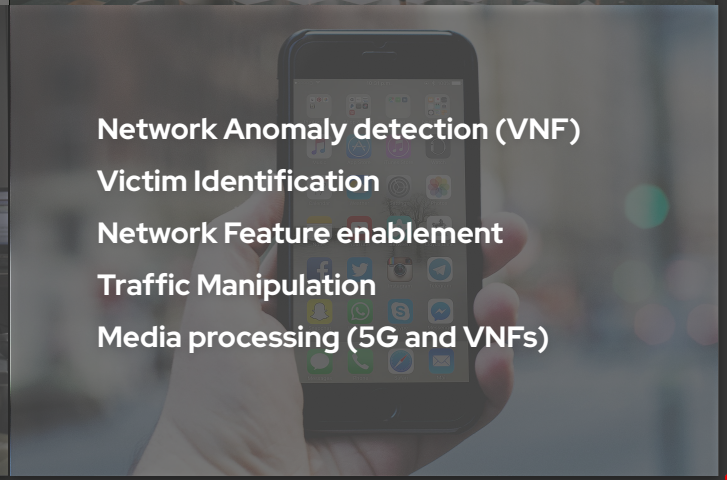
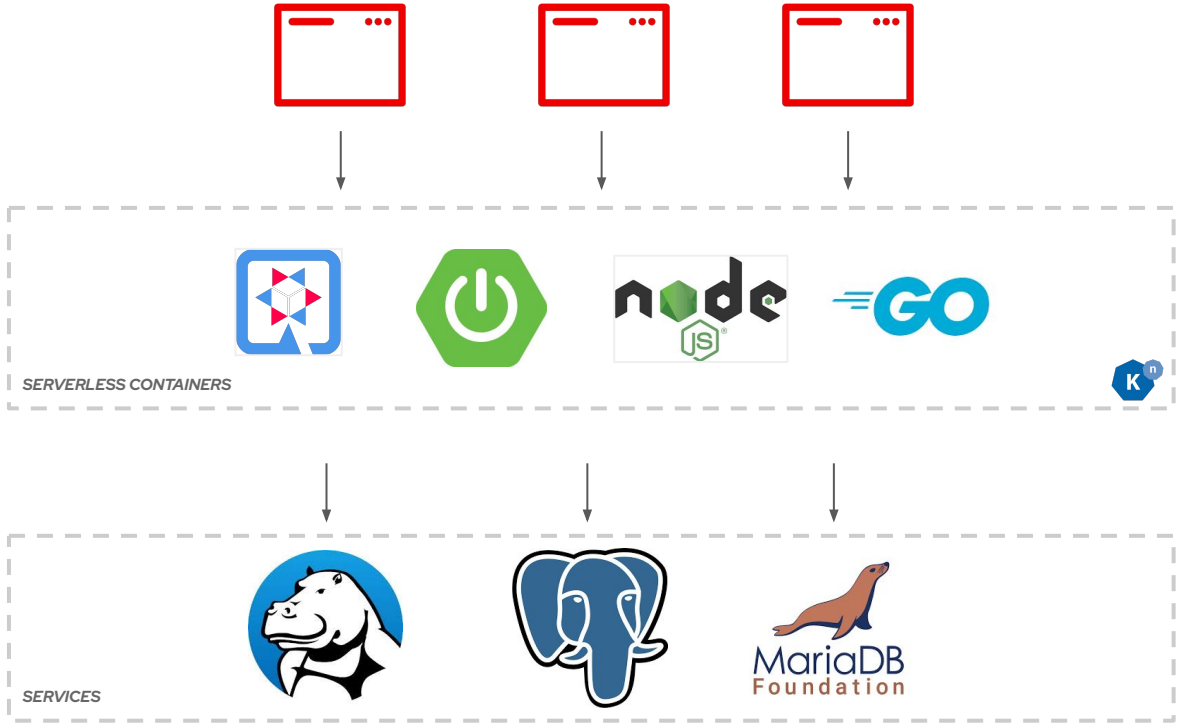


Image results validation (X-rays, MRIs)
Fast Healthcare Interoperability Resources
Queries
Result notifications
Scheduling services
Test result requests (PDFs, Reports)



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:

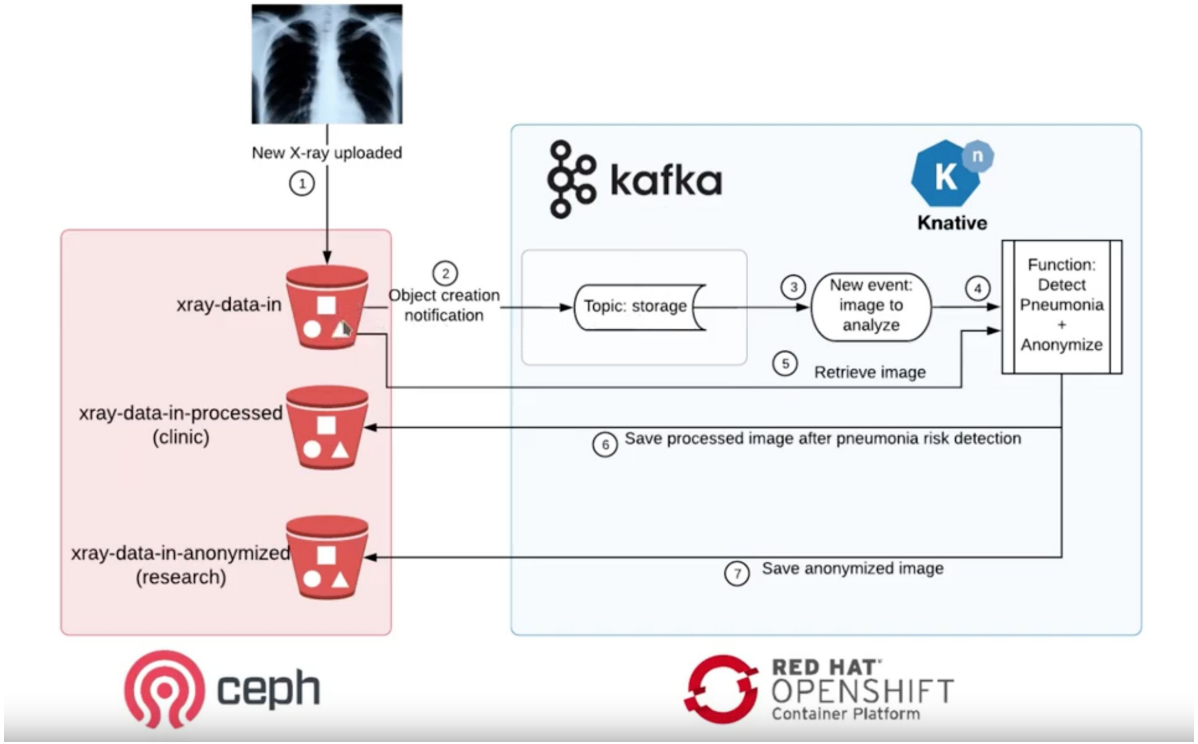
OpenJDK
python
django
VERT.X
JS
RAILS

Red Hat

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



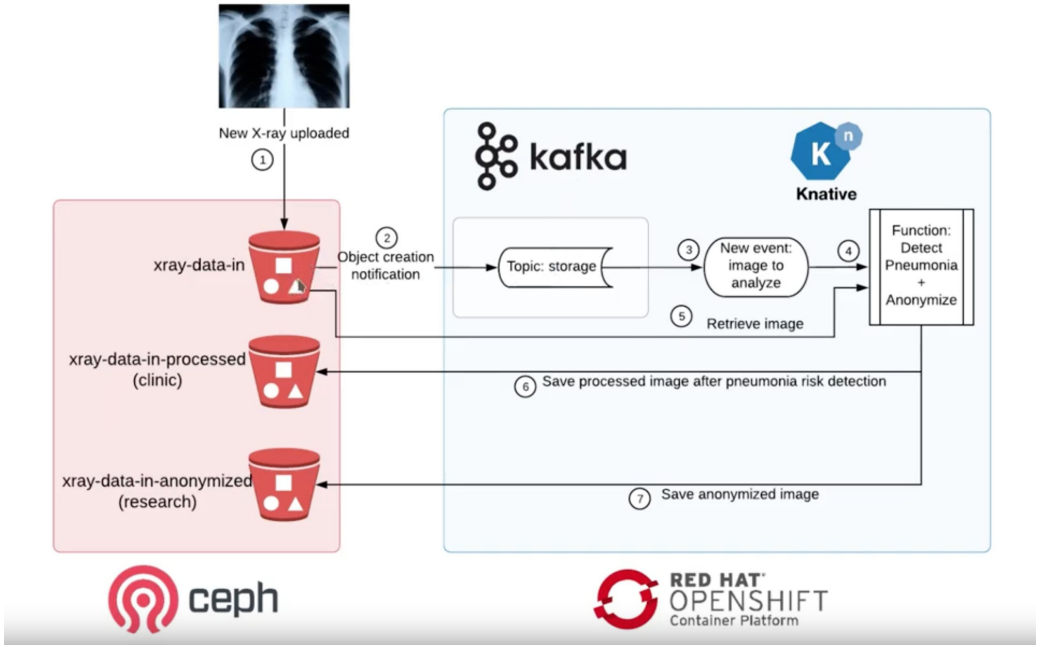
* Article: https://medium.com/analytics-vidhya/automated-data-pipeline-using-ceph-notifications-and-kserving-5e1e9b996661

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



TensorFlow

https://medium.com/analytics-vidhya/automated-data-pipeline-using-ceph-notifications-and-kserving-5e1e9b996661

Red Hat

Building on OpenShift Serverless with Red Hat Services

Connected Services
How Knative services interact with the outside world.

API Gateway
Next gen APIs still require management.

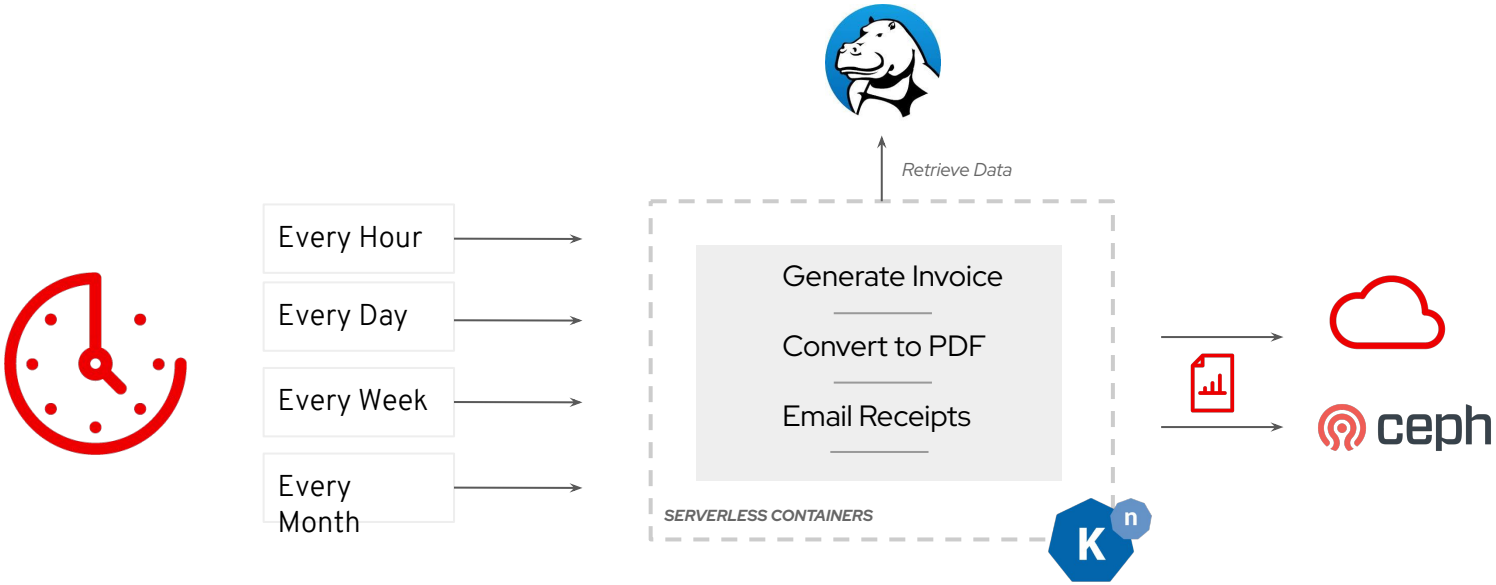
Service Orchestrator
Composing multiple services together into an application.

Implementing Services
Functions, languages, and the vagaries of cold starts.

Event Streaming
All modern architectures need some Kafka.

The Dirty Word in Serverless
Yep, you still need state to handle long-lived orchestration.

Scheduled Apps Cron Jobs



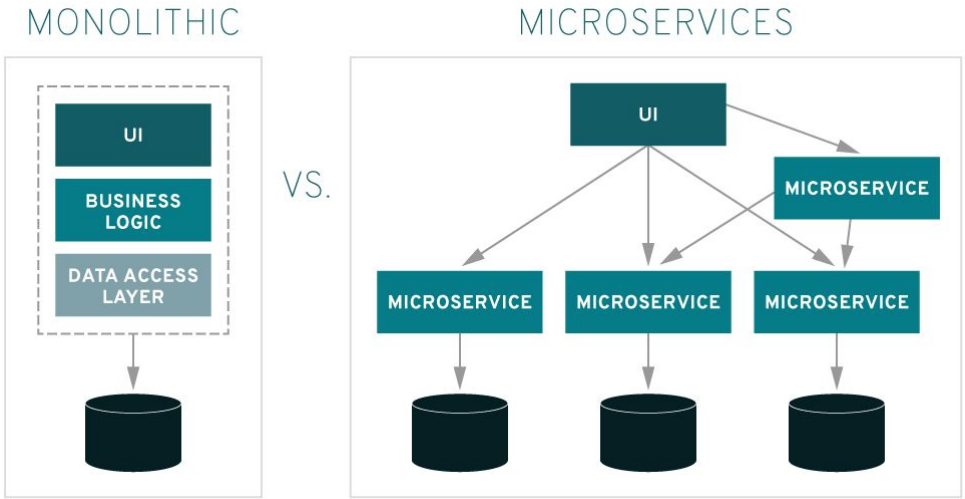
Red Hat

Optional - ServiceMesh / Istio

What are Microservices?

an architectural style that structures an application as a collection of services

- Single purpose
- Independently deployable
- Have their context bound to a biz domain
- Owned by a small team
- Often stateless



Benefits of Microservices

Agility

Deliver updates faster and react faster to new business demands

Highly scalable

Scale independently to meet temporary traffic increases, complete batch processing, or other business needs

Can be purpose-built

Use the languages and frameworks best suited for the

Many orgs have had success with Microservices - Netflix, Amazon, eBay, The Guardian

Resilience

Improved fault isolation restricts service issues, such as memory leaks or open database connections, to only

There is inherent complexity in adopting microservices

Some common areas where organizations stumble when adopting microservices

Tolerance to Faults

Cascading failure, partial outages, traffic spikes

DevOps and Deployments

More failure surface, version incompatibility, untracked svcs

Services Communication Needs

Latency, concurrence, distributed transactions

Inability to Monitor & Understand Performance

More to monitor & different types of monitoring required

Securing Services

Malicious requests, DoS, id & access control

Highly Distributed Logs

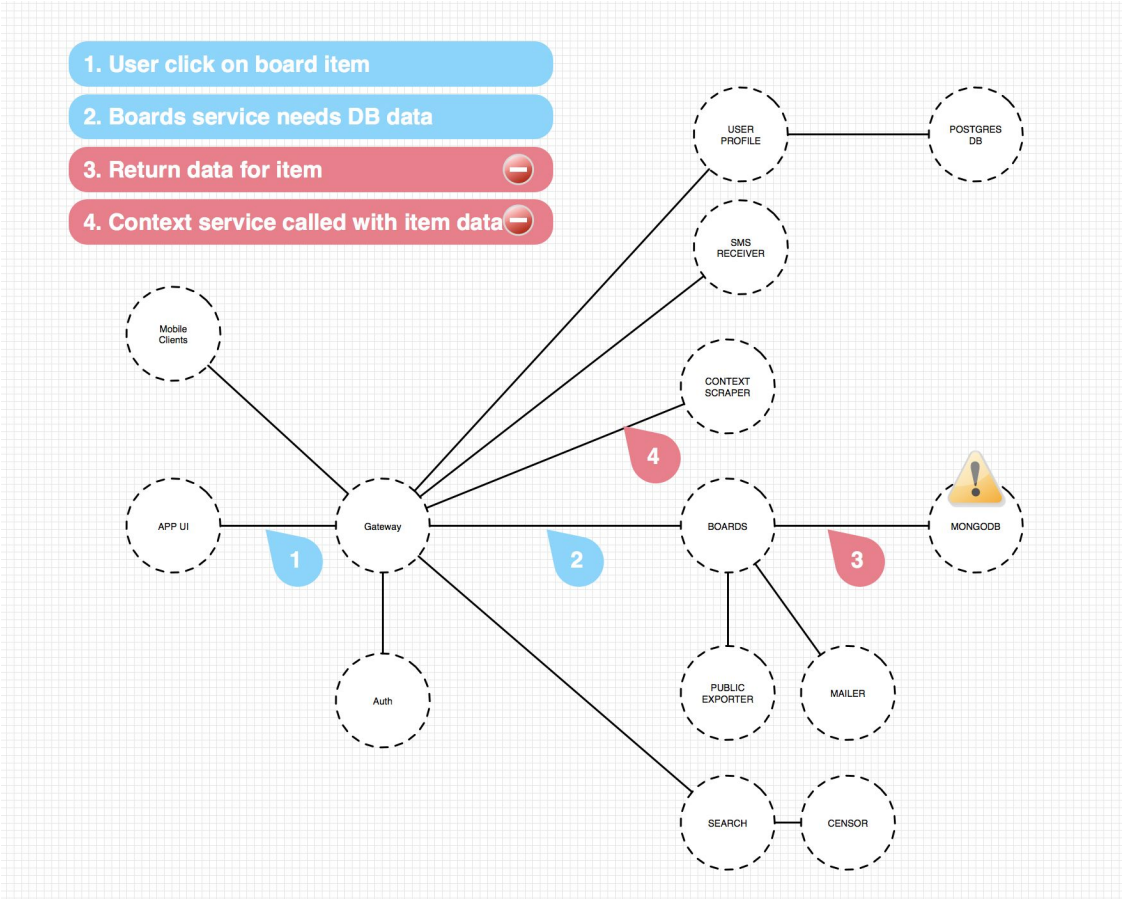
Scattered logs, lots more logs to manage, access control

Biggest challenge:
What used to be internal now needs to go across a network

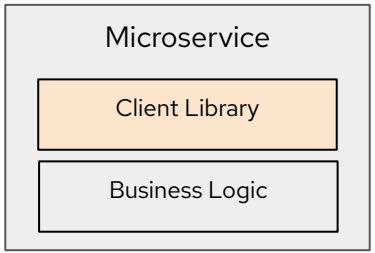
Partial Failure → Cascading Failures

- Route service's database goes offline
- Now no item data can be returned
- Default timeout for developer's HTTP framework is 2 minutes
  - This wait is happening repeatedly
- Item data is needed to pass to context scraper service - so it fails too
- User experience is poor

- Leads to unexpected case of users repeatedly mashing the refresh button
- Now the boards service begins to get more requests than than it can handle



Language Specific Libraries and Tools



Narrow Scope

Built to address a specific problem such as fault tolerance

Language Specific

Client libraries are tied to programming language

Thick Clients

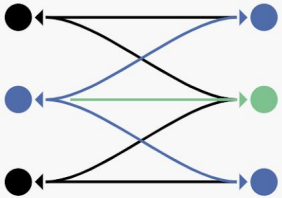
Solution results in clients with bulk of

Don’t force extra work on developers

There is a better way

Istio Service Mesh

A modern way to manage the complexity of microservice applications

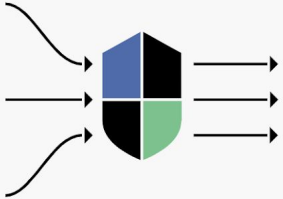


Connect

Intelligently control the flow of traffic and API calls between services, conduct a range of tests, and upgrade gradually with red/black deployments.

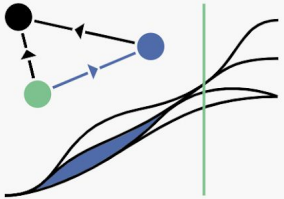
Secure

Automatically secure your services through managed authentication, authorization, and encryption of communication between services.



Control

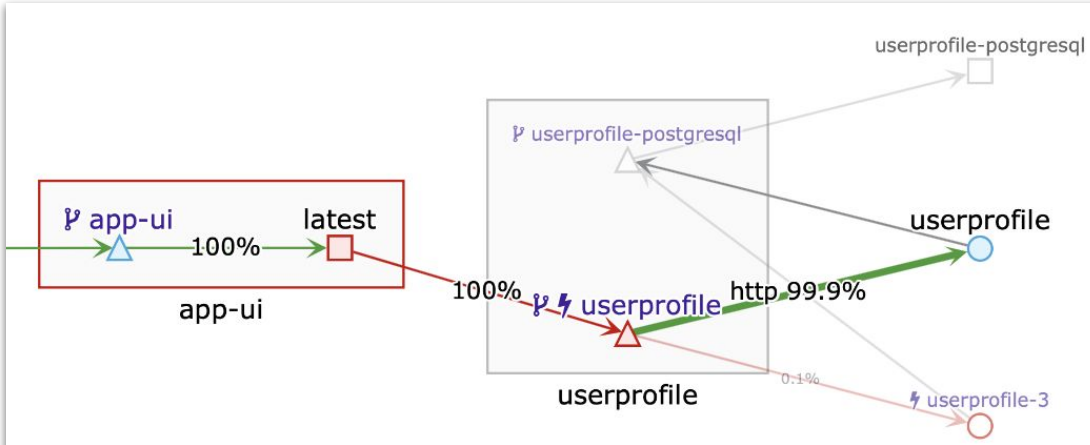
Apply policies and ensure that they're enforced, and that resources are fairly distributed among consumers.

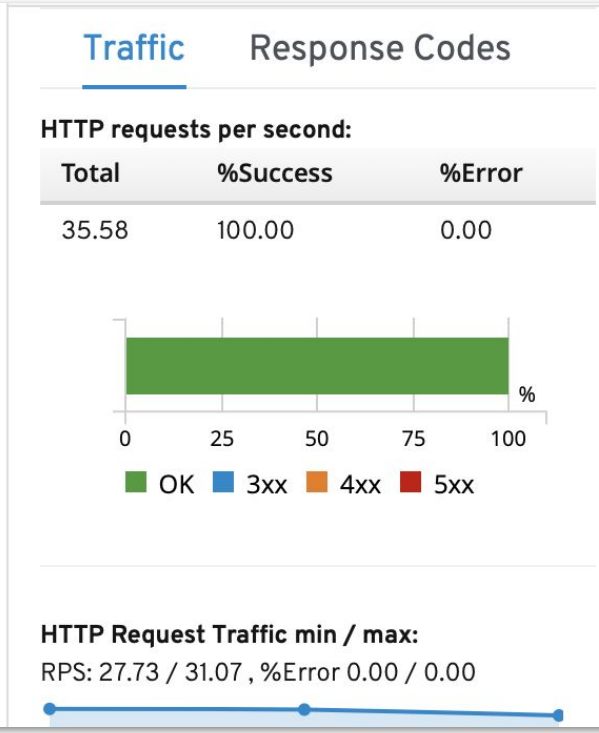


Observe

See what's happening with rich automatic tracing, monitoring, and logging of all your services.

Handling Partial Failures with the Service Mesh





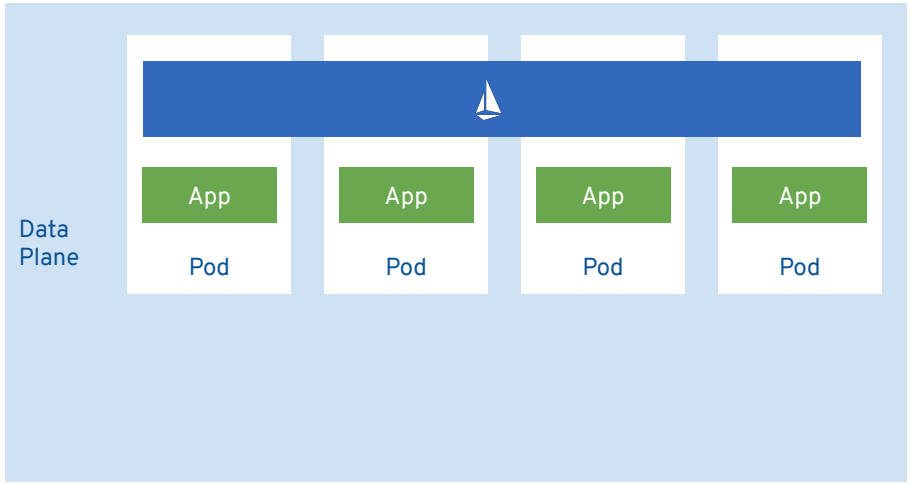

```
26      trafficPolicy:
27        connectionPool:
28          http:
29            http1MaxPendingRequests: 1
30            maxRequestsPerConnection: 1
31        outlierDetection:
32          consecutiveErrors: 1
33          interval: 1s
34          baseEjectionTime: 10m
35          maxEjectionPercent: 100
```

YAML configurable,
(Kubernetes native)

Red Hat

The service mesh is critical in addressing the inherent complexity of microservices

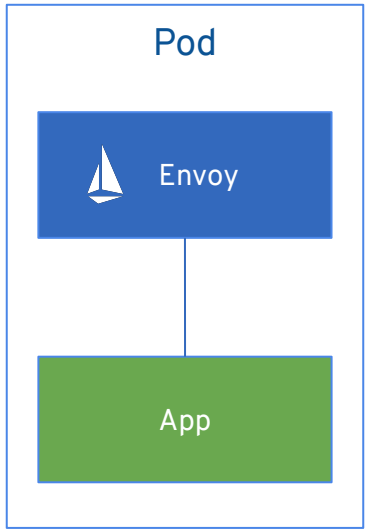
Your Services are in a “Data Plane”



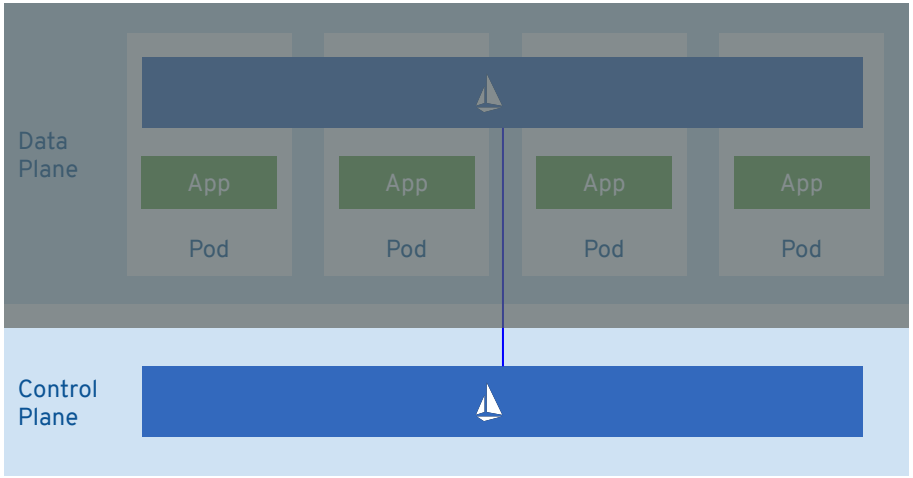
RED HAT® OPENSHIFT

Red Hat

What's a Sidecar have to do with containers?



Your Policy Makes Up a “Control Plane”



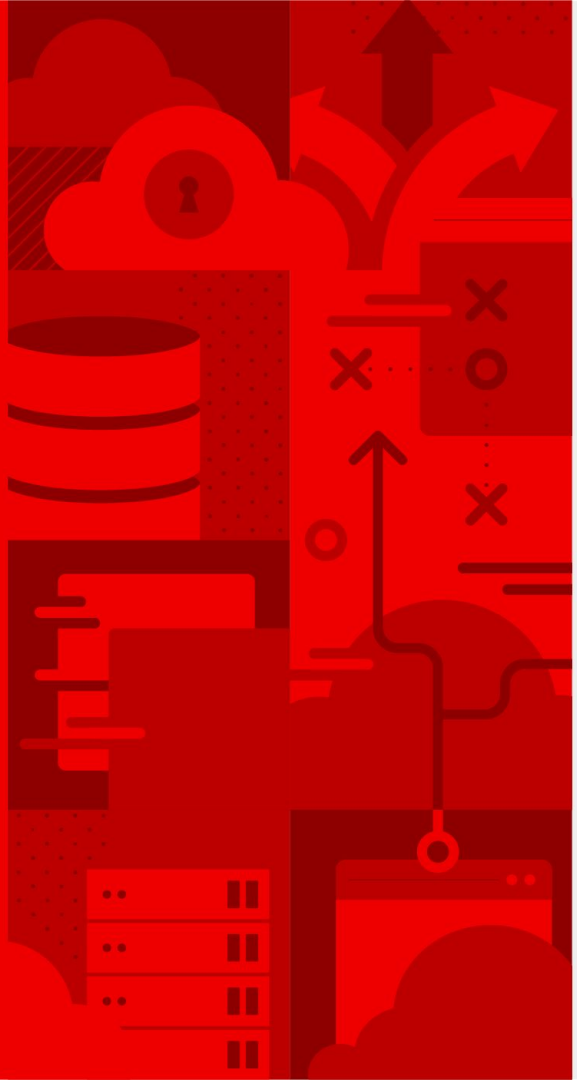
RED HAT® OPENSHIFT

Red Hat

Istio - Caveats

- Caveats
  - Be careful, istio is NOT a ServiceBus
  - Be careful, istio is NOT an API Manager
  - Be careful, istio is NOT meant to be a router
  - Be careful, istio is NOT a BPM tool

- Istio is meant to help solving some challenges
  - Centrally encryption / management of internal service communication
  - Distributed networking / communication of services
  - Increase Fault tolerance
  - Fault simulation



Thank you

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat