



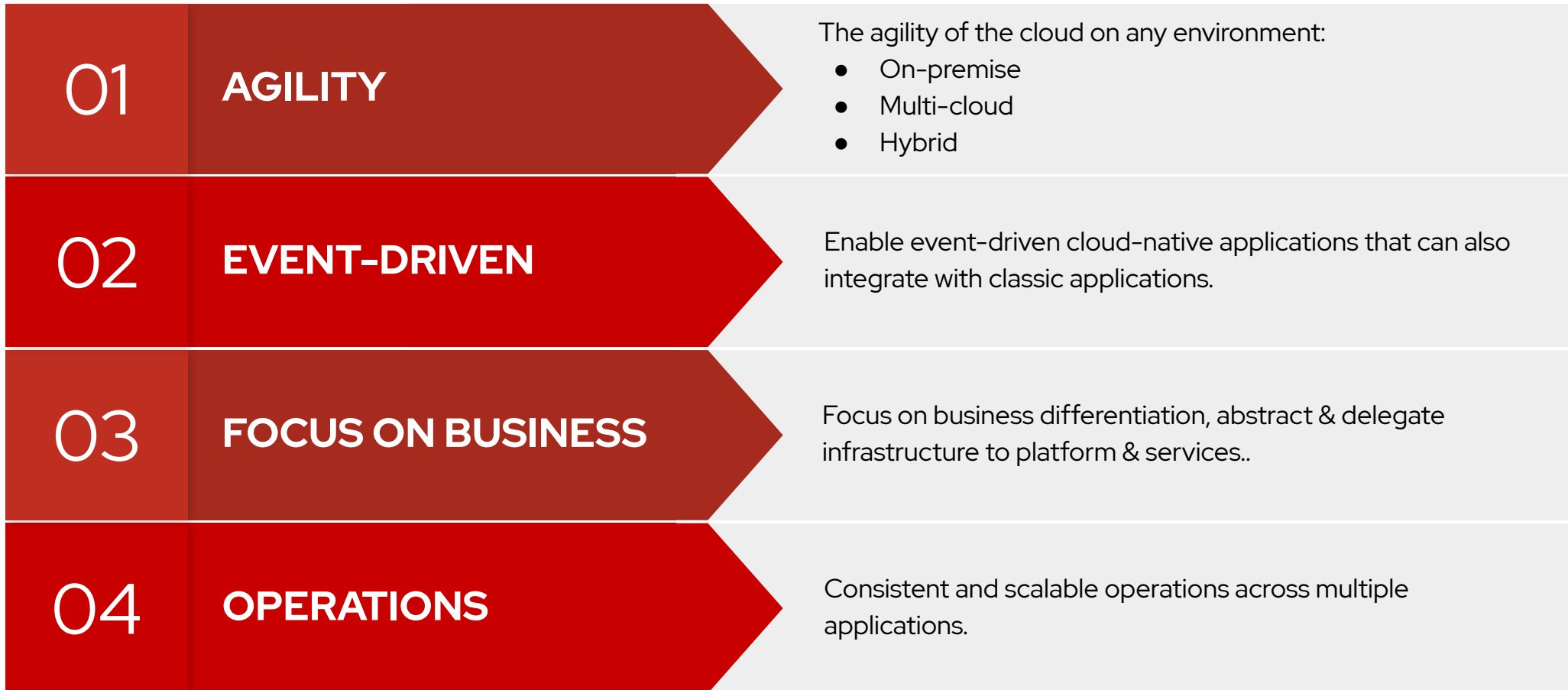
Red Hat Dallas Emerging Tech Summit

December 5, 2019

Developing Serverless Applications with Knative

Huamin Chen, Principal Software Engineer

Why do we need Serverless ?



Evolution of Serverless

1.0

AWS Lambda, Functions...

Serverless 1.0 was built around the FaaS component and by other services such as API Gateways. The genesis of the current is general is available but far from ideal for general computing, and with potential candidates for improvements.

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

Evolution of Serverless

1.0

AWS Lambda, Functions...

Serverless 1.0 was built around the FaaS component and by other services such as API Gateways. The genesis of the current is general is available but far from ideal for general computing, and with potential candidates 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**

Evolution of Serverless

1.0

AWS Lambda, Functions...

Serverless 1.0 was built around the FaaS component and by other services such as API Gateways. The genesis of the current is general is available but far from ideal for general computing, and with potential candidates 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 providers start adding the missing parts to make Serverless 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

CLOUD NATIVE DEVELOPMENT MARKET

Application Concerns



Routing & transformation
Technology Adapters
Error Handling
Development Patterns



| | | | |
|-----------|-----------|---------------|-----|
| Stateless | Batch Job | Microservices | ... |
| Stateful | Functions | Singleton | |

Declarative programming
Event orchestration
Activation & scale-to-zero
Service Binding



| | |
|-------------|--------------|
| SERVING API | EVENTING API |
|-------------|--------------|

Continuous Integration
GitOps
Continuous Delivery



| | | |
|------|----------|------|
| Task | Pipeline | Step |
|------|----------|------|

Traffic Routing
Network Resilience
Security



| | | |
|--------------|---------|----------------|
| ServiceEntry | Gateway | VirtualService |
|--------------|---------|----------------|

Infrastructure Concerns

Provisioning
Scheduling & Deployment
Scaling & Service Discovery
Monitoring and Recovery

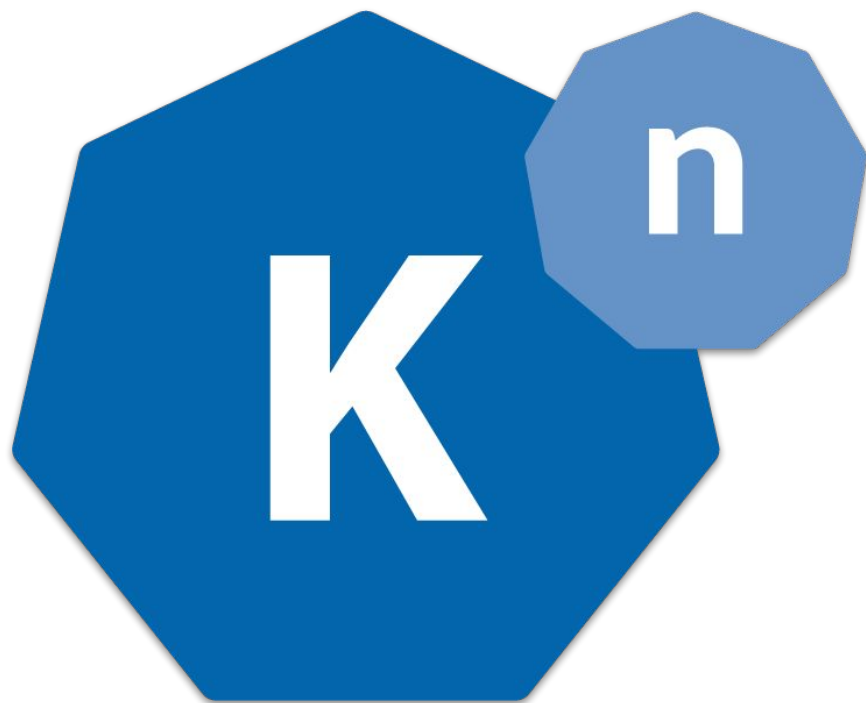


| | | |
|------------|-----------|------------|
| ReplicaSet | Stateful | Deployment |
| Service | Container | Logs |
| CronJob | ConfigMap | Health |

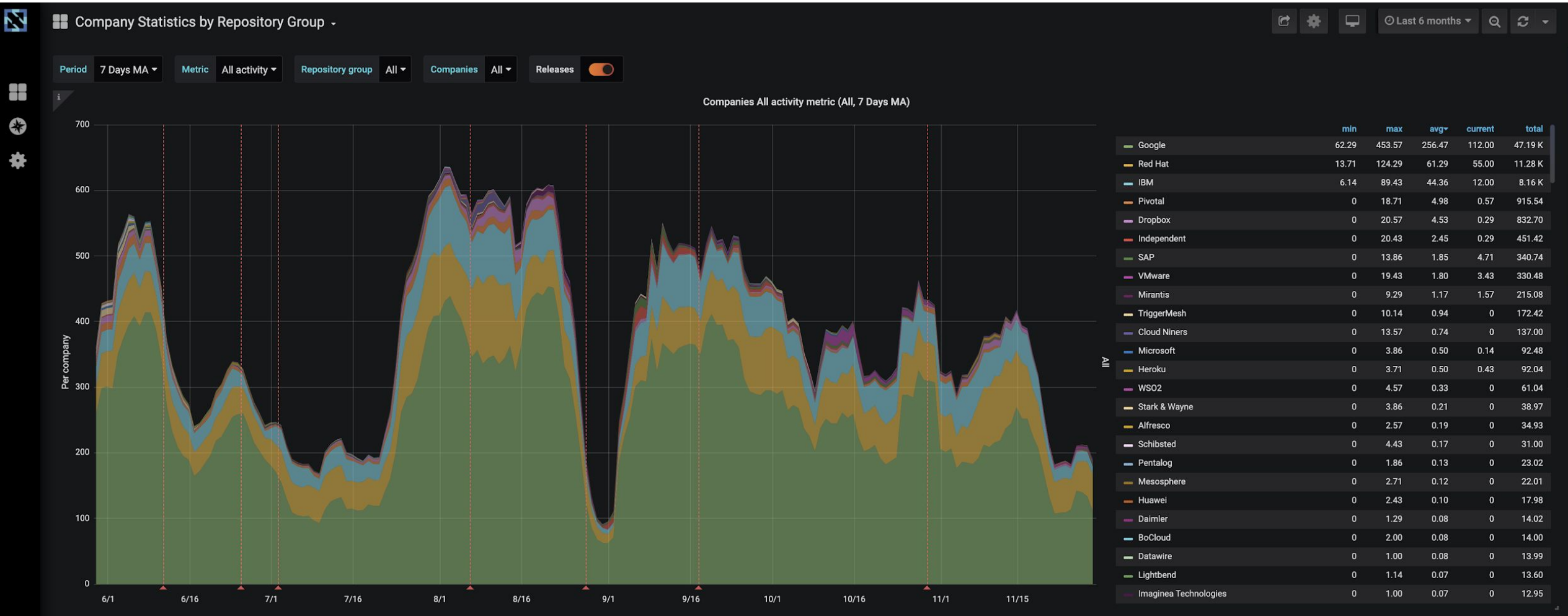
High-level primitives



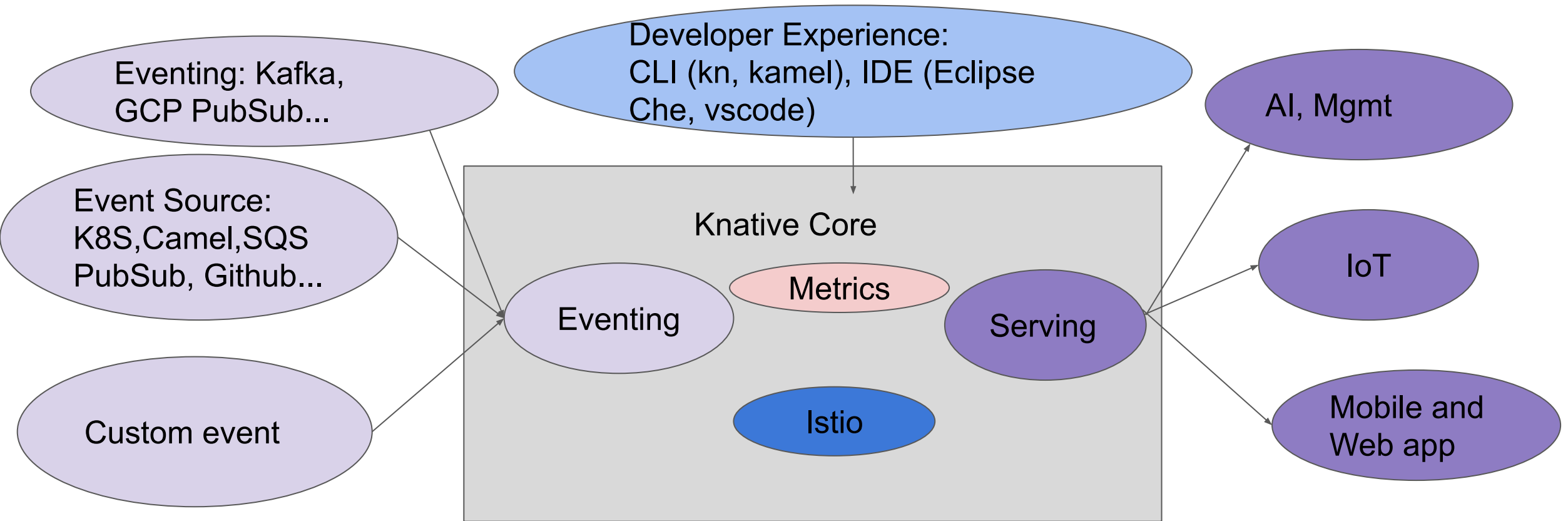
Low-level primitives



Who's Who (knative)

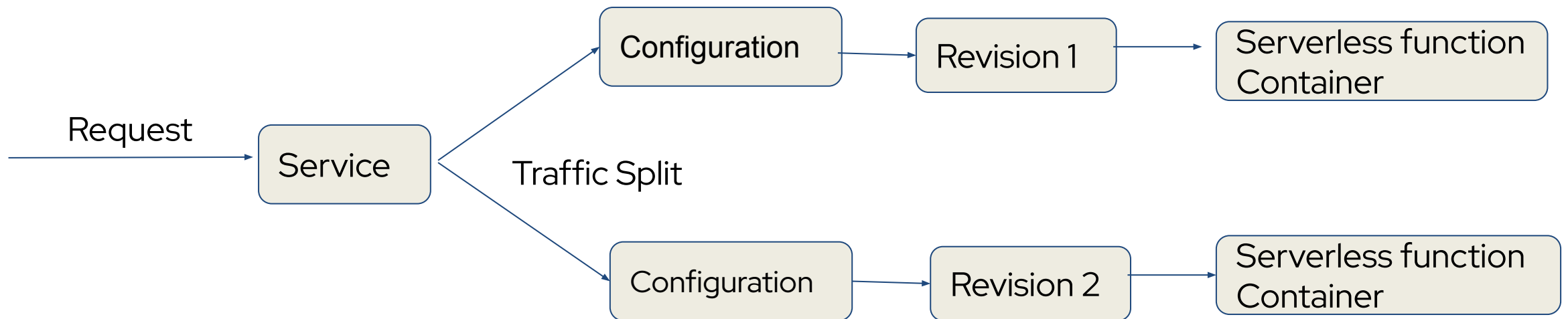


source: <https://knative.teststats.cncf.io/d/4/company-statistics-by-repository-group?orgId=1>

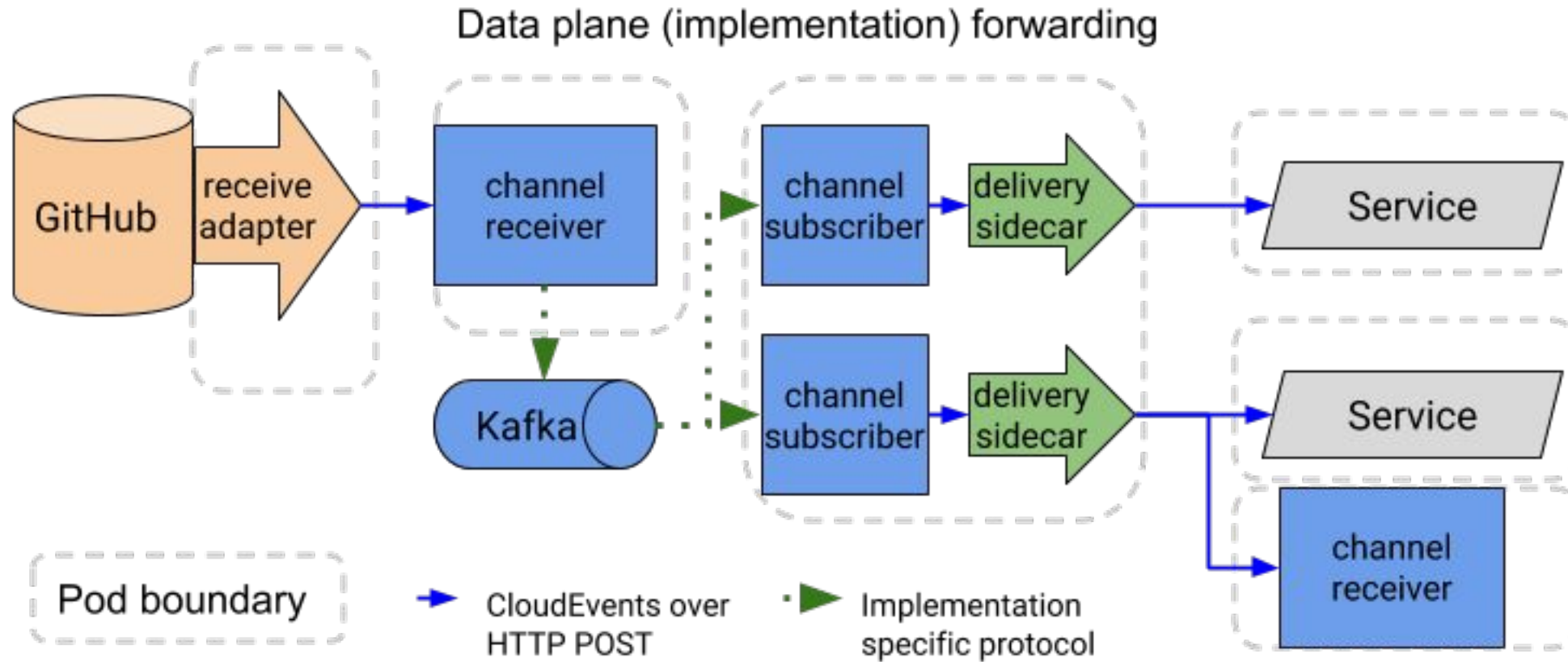


Knative Serving

- Configuration: How Function Container is configured
- Revision: Identify Container name or Image Tags to use
- Service: Traffic Routing



Knative Eventing



Source: <https://github.com/knative/docs>

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
```



Knative

```
apiVersion: -serving.knative.dev/v1alpha1
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



Simple use cases to get started... (No YAML)

Create a new service with 1 instance running all the time (no scale to zero) and limiting memory consumption

```
kn service create myService --image=.. --min-scale=1 --max-scale=100 --limits-memory=100m
```

Update a service with multiple Revisions to send 50% of traffic to each version

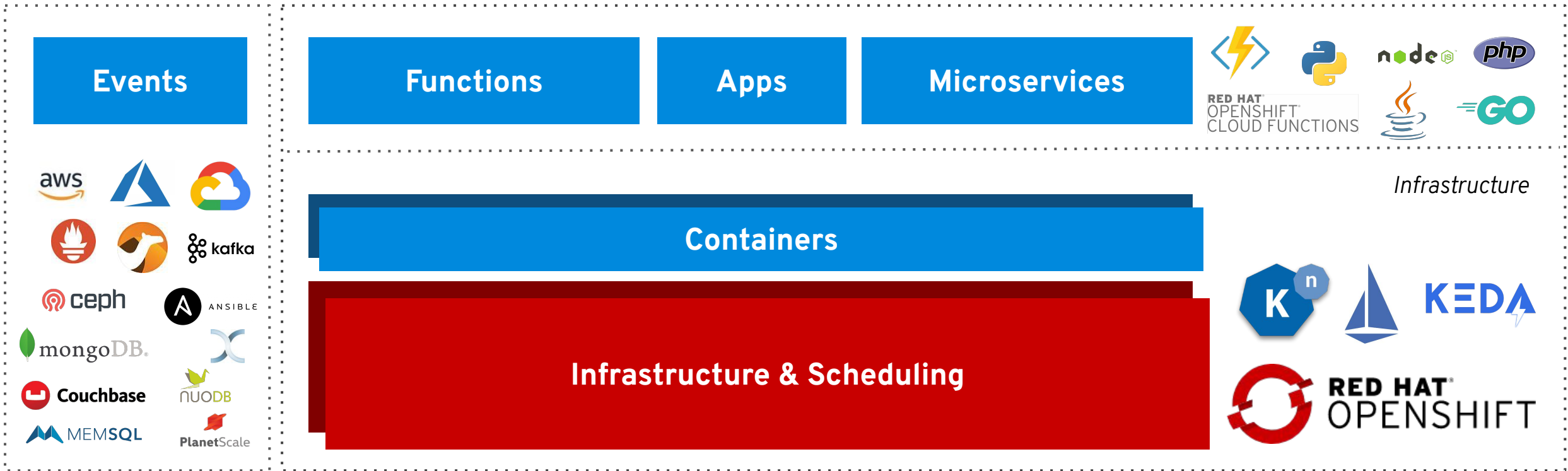
```
kn service update myService --traffic myService-rev1=50,myService-rev2=50
```

Output YAML file for a given service for version control

```
kn service describe myService -o yaml
```



Microservices, Functions and Apps + Events = OpenShift Serverless



OperatorHub.io

Red Hat OpenShift Container Platform

Administrator

You are logged in as a temporary administrative user. Update the [cluster OAuth configuration](#) to allow others to log in.

Project: markito

Services

Create Service

Filter by name...

| Name ↑ | Namespace ↓ | Domain ↓ | GenerationAge ↓ | ConditionsReady | Reason |
|----------------------------------|-------------------------|----------|--------------------|-----------------|--------|
| S kiosk-qrdecoder | NS markito | - | 1 Aug 19, 10:14 am | 3 OK / 3 True | - |
| S kiosk-qrgen | NS markito | - | 1 Aug 19, 10:11 am | 3 OK / 3 True | - |
| S quarkus-qrdecoder | NS markito | - | 2 10 minutes ago | 3 OK / 3 True | - |

Red Hat OpenShift Container Platform

Administrator

You are logged in as a temporary administrative user. Update the [cluster OAuth configuration](#) to allow others to log in.

Project: markito

Routes

| Name ↑ | Namespace ↓ | URL ↓ | Age ↓ | Conditions | Traffic |
|----------------------------------|-------------------------|---|------------------|------------|---------------------------------|
| R kiosk-qrdecoder | NS markito | http://kiosk-qrdecoder.markito.apps.openshift.codeready.cloud | Aug 19, 10:14 am | 3 OK / 3 | 100% → kiosk-qrdecoder-cvkmq-1 |
| R kiosk-qrgen | NS markito | http://kiosk-qrgen.markito.apps.openshift.codeready.cloud | Aug 19, 10:11 am | 3 OK / 3 | 100% → kiosk-qrgen-tjgy-1 |
| R quarkus-qrdecoder | NS markito | http://quarkus-qrdecoder.markito.apps.openshift.codeready.cloud | Aug 19, 10:26 am | 3 OK / 3 | 100% → quarkus-qrdecoder-gwbs-1 |


| Name ↑ | Namespace ↓ | URL ↓ | Age ↓ | Conditions | Traffic |
|----------------------------------|-------------------------|---|------------------|------------|---------------------------------|
| R kiosk-qrdecoder | NS markito | http://kiosk-qrdecoder.markito.apps.openshift.codeready.cloud | Aug 19, 10:14 am | 3 OK / 3 | 100% → kiosk-qrdecoder-cvkmq-1 |
| R kiosk-qrgen | NS markito | http://kiosk-qrgen.markito.apps.openshift.codeready.cloud | Aug 19, 10:11 am | 3 OK / 3 | 100% → kiosk-qrgen-tjgy-1 |
| R quarkus-qrdecoder | NS markito | http://quarkus-qrdecoder.markito.apps.openshift.codeready.cloud | Aug 19, 10:26 am | 3 OK / 3 | 100% → quarkus-qrdecoder-gwbs-1 |

The screenshot shows the OpenShift Container Platform console interface. At the top, it says "Red Hat OpenShift Container Platform" and "You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in." Below this, there are navigation menus for "Developer", "Topology", "Builds", "Pipelines", and "Advanced". The main area displays a service topology diagram for a project named "my-project". The diagram shows a CronJob (CJS) named "test-cronjob-so..." connected to a Service (S) named "store-app". The "store-app" service is backed by two Replicasets (R): "store-app-lcvcb..." and "store-app-bbgc...". Dashed arrows indicate a 50% traffic split between the two Replicasets. The interface includes search and zoom controls at the bottom.

This screenshot shows a detailed view of the "store-app" service in the OpenShift console. The left pane shows a zoomed-in view of the service topology diagram. The right pane displays the service details, including a "Resources" tab. Under the "Revisions" section, there is a "Set Traffic Distribution" button and a table showing the traffic split between two revisions.

| Revision | Traffic |
|-------------------|---------|
| store-app-bbgc-1 | 50% |
| store-app-lcvcb-2 | 50% |

Supporting Technologies

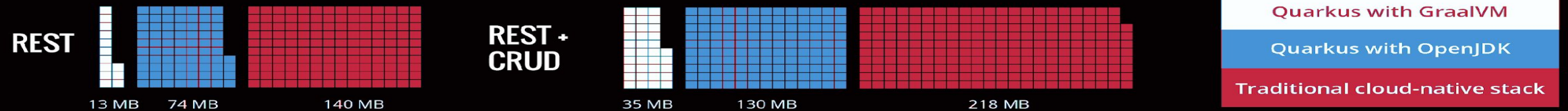


QUARKUS

Supersonic Subatomic Java

A Kubernetes Native Java stack tailored for OpenJDK HotSpot and GraalVM, crafted from the best of breed Java libraries and standards.

Memory (RSS) in Megabytes



Boot + First Response Time in Seconds



MEMORY & BOOT + FIRST RESPONSE TIME

```
mvn io.quarkus:quarkus-maven-plugin:1.0.0.CR1:create ...
mvn package -Pnative -Dnative-image.docker-build=true
kn service create gettingstarted-quarkus --image=markito/getting-started:v1
```



```
func init MyFunctionProj --docker # pick a runtime
cd MyFunctionProj
func new --name MyHttpTrigger --template "HttpTrigger"
docker build . hello-azure-func
kn service create hello-azure-func --image=markito/hello-azure-func:v1
```



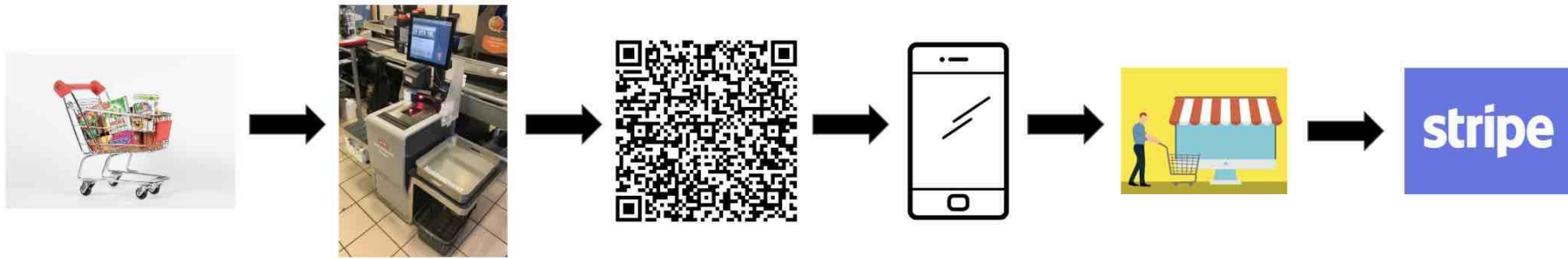
Apache Camel K

Integration:

```
from('timer:tick?period=3s')  
  .setBody().constant('Hello world from Camel K')  
  .to('log:info')
```

Deploy:

```
kamel run helloworld.groovy
```



<https://github.com/markito/kqr-pay>



```
kn service create kiosk --image=markito/kiosk:v1 --requests-memory=100Mi --concurrency-limit=1
kn service create payment-service --image=markito/payment-service:v1 --requests-memory=100Mi
--concurrency-limit=1
```

```
kn service create store-app --image=markito/store-app:v1 --requests-memory=300Mi
--concurrency-limit=10 --env PAYMENT_SERVICE="http://payment-service.markito.svc.cluster.local"
```

```
kamel run -t gc.enabled=false --dev src/main/groovy/request-router.groovy
--dependency=mvn:com.github.lburgazzoli/camel-k-kqr-pay-support/1.0.0 --secret=salesforce";
```



Kogito

- Build business automation applications, based on Drool.
- Complex Event Processing capabilities
- Support both Quarkus and Spring Boot



Kogito

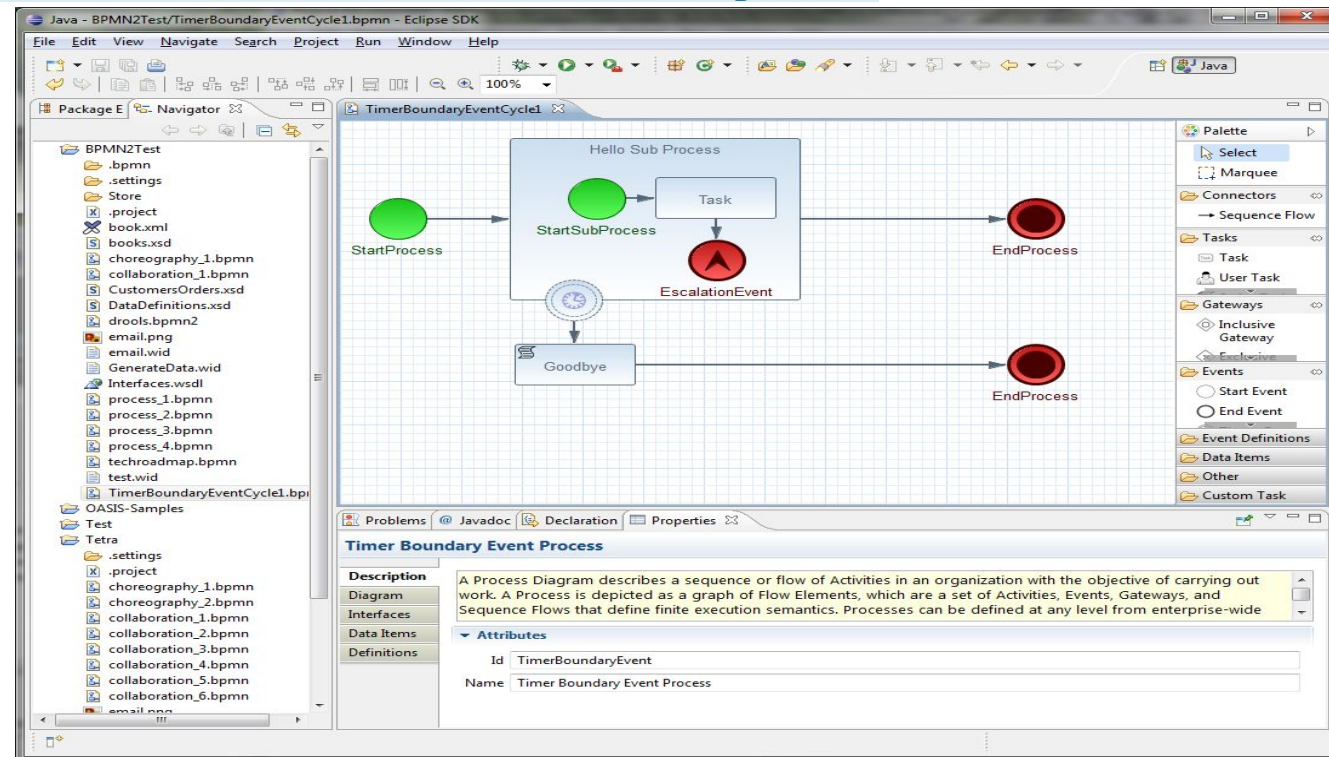
1. Create Application

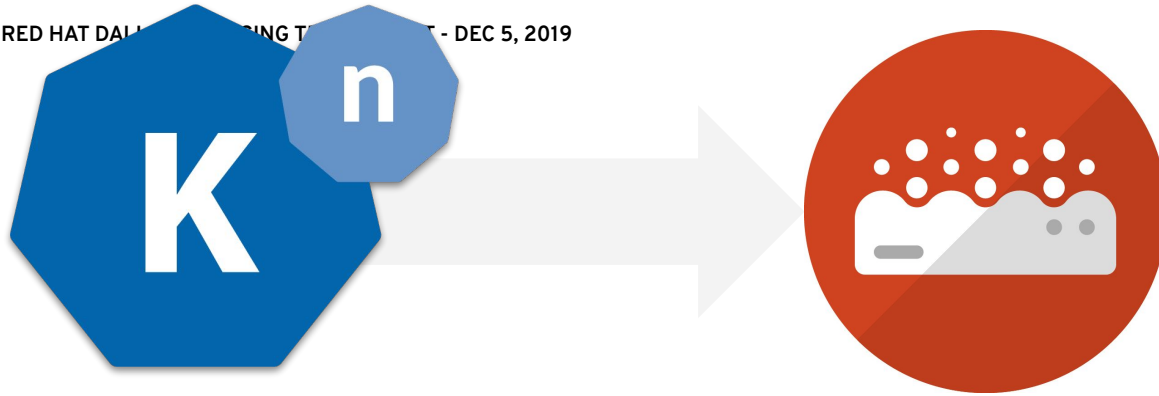
```
mvn io.quarkus:quarkus-maven-plugin:create -Dextensions="kogito"
```

2. In Eclipse Che IDE, edit business process

3. Build and Run Native Code

```
mvn clean package -Pnative
```





Learn more

OpenShift Serverless

Build and deploy serverless applications using an event-driven infrastructure on Red Hat® OpenShift®

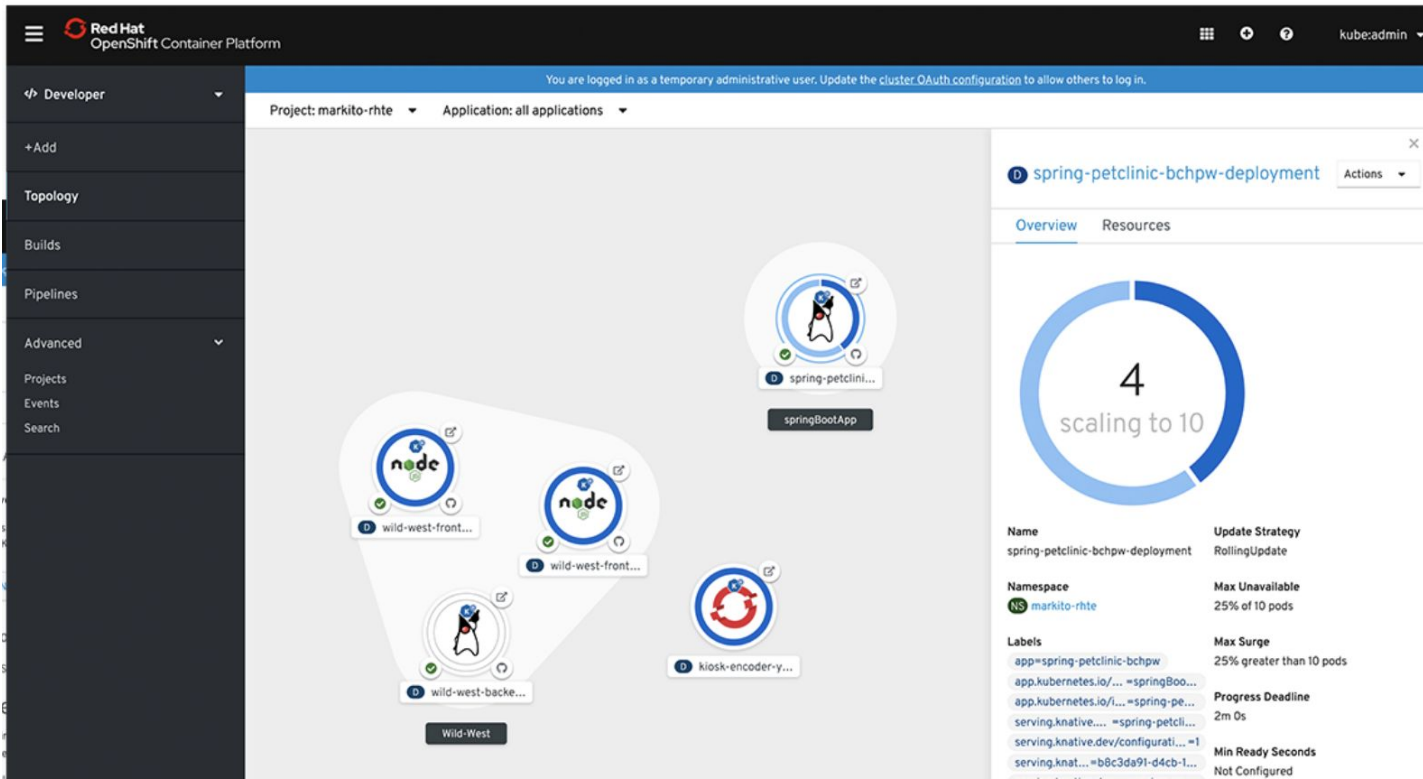
Tutorial

Get started with your serverless journey

Knative Blog series

Knative: Serving your Serverless Services

<https://www.openshift.com/learn/topics/serverless>



STAY ENGAGED

Developers.redhat.com

Your access point for no-cost developer tools and product subscriptions, how-tos, and demos

Red Hat User Groups

Meetups for networking and tech deep dives

www.meetup.com/Dallas-Red-Hat-Users-Group/

DevNation

Virtual and live events

Catch replays at

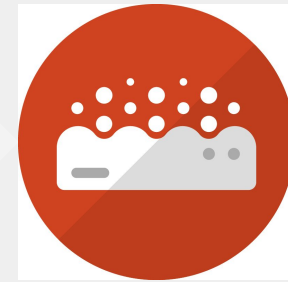
<https://developers.redhat.com/devnation/>

Next.redhat.com

Stay in touch with the Office of the CTO





Thank you



 linkedin.com/company/red-hat

 facebook.com/redhatinc

 youtube.com/user/RedHatVideos

 twitter.com/RedHat