Knative Serverless to architect atomically decomposed components

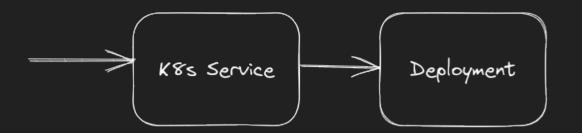
Maarten Vandeperre Specialist Solution Architect Red Hat Bart Joris
Account Solution Architect
Red Hat



A simple application on Kubernetes

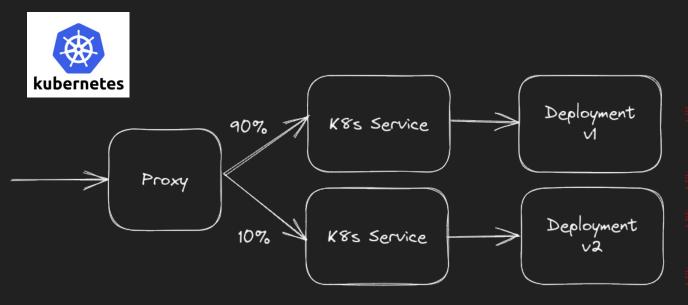






A/B testing a simple application





Need to:

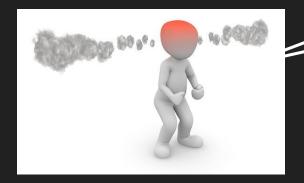
- Create k8s deployments
- Create k8s services
- Create proxy
- Configure proxy
- (Configure k8s ingress)



Serverless computing - What's in a name?



- Simplify deployment of applications
- Don't worry about underlying infrastructure (including kubernetes)
- Only use resources when needed



There are always servers running somewhere!

What is Knative?



Knative is a Kubernetes Native implementation of Serverless technology

CNCF incubating project

From the Open Source project site - "Knative is an Open-Source Enterprise-level solution to build Serverless and Event Driven Applications"

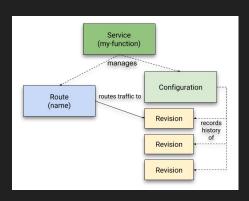
Knative provides a set of additional Objects (through CRD's) that allow the management of Serverless components.

More yaml? => kn CLI

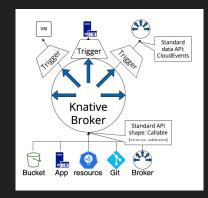
Knative overview



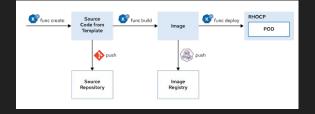
Serving



Eventing



Functions



Builds



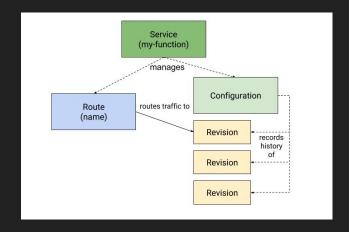


Knative Serving Basics



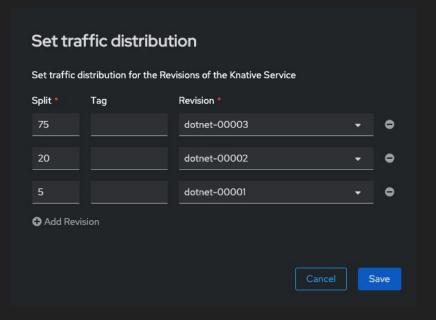
Knative Serving is realised using four CRD components:

- Services controller object that orchestrates the other components
- 2. Configurations defines the state of the deployment. Editing the configuration generates a new Revision
- **3. Revision** immutable point-in-time snapshot of configuration
- Routes maps a network endpoint to one or more Revisions



Revisions





A new **Revision** is created when the configuration of a knative Serving Application is changed

Revisions are immutable

Routes allow for changing the distribution of traffic arriving to push traffic to multiple **Revisions**

Allows for Blue/Green deployments, A/B testing, canary release

Revisions can have their own entry point

Knative Serving Autoscaling



Knative Serving provides auto-scaling to match incoming demand

Scaling based on HTTP requests!

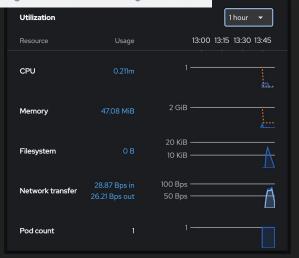
Scale-to-zero (can be disabled)

=> More efficient resource consumption

Configure scaling:

- **concurrency** (how many requests can be simultaneously processed by each replica), **or**
- requests-per-second (per replica)
- configured per **Revision** (annotations) or globally (config-map or operator)

apiVersion: serving.knative.dev/v1
kind: Service
metadata:
 name: helloworld-go
 namespace: default
spec:
 template:
 metadata:
 annotations:
 autoscaling.knative.dev/target: "200"



Direct Control of Knative objects



Knative also provides a command-line utility called **kn**

kn allows you to create, modify and delete all knative objects directly on a target Kubernetes/OpenShift cluster

```
serverless1 — -zsh — 147×38
uther@ilawson-mac serverless1 % kn service list
                                                                                            LATEST
                                                                                                                                READY
           https://dotnet-serverlessdemo.apps.cluster-s9kn8.s9kn8.sandbox716.opentlc.com
                                                                                            dotnet-00003
                                                                                                                                True
dotnet
          https://svctest1-serverlessdemo.apps.cluster-s9kn8.s9kn8.sandbox716.opentlc.com
uther@ilawson-mac serverless1 % kn broker list
                                                                                                                     REASON
NAME
                                                                                                             READY
default
        http://broker-ingress.knative-eventing.svc.cluster.local/serverlessdemo/default
uther@ilawson-mac serverless1 % kn trigger list
                          SINK
                                           AGE
                                                 CONDITIONS READY
                                                                      REASON
trigger-quarkus default ksvc:svctest1 75m 6 OK / 6
uther@ilawson-mac serverless1 % kn trigger describe trigger-quarkus
             trigger-quarkus
             serverlessdemo
              app.kubernetes.io/instance=serverlesstest, eventing.knative.dev/broker=default
Annotations: eventing.knative.dev/creator=system:serviceaccount:openshift-gitops:openshift-gitop ...
Age:
Broker:
             default
Filter:
 type:
             quarkusevent
Sink:
 Resource: Service (serving.knative.dev/v1)
Conditions:
 OK TYPE
                              AGE REASON
  ++ Ready
                               1h
                               1h
 ++ BrokerReady
  ++ DeadLetterSinkResolved
                               1h DeadLetterSinkNotConfigured
  ++ DependencyReady
  ++ SubscriberResolved
                               1h
  ++ SubscriptionReady
uther@ilawson-mac serverless1 %
```

Knative Serving



kn service create myservice --image docker.io/openshift/hello-openshift --env RESPONSE="hello1"

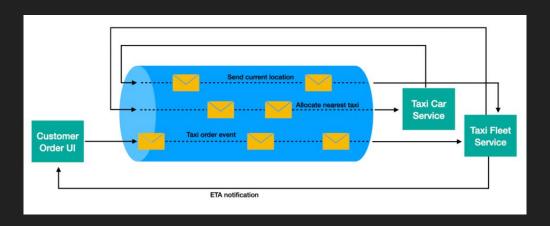
kn service update myservice --env RESPONSE="hello2"

kn service update myservice --traffic myservice-00001=50,myservice-00002=50



Event-driven architecture





Microservices that exchange info through events

Asynchronous (contrast to REST architectures)

Loosely coupled

Resilient

Scalable

CloudEvents





cloudevents

CNCF incubating project - https://cloudevents.io/

Provides a common event schema => Interoperability, portability

Extensible through extension attributes

SDKs for different programming languages

Protocol-agnostic (HTTP, AMQP, MQTT, ...)

Wide adoption

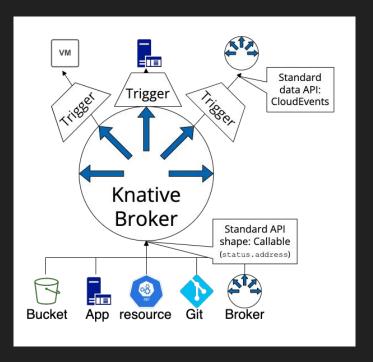
Knative Eventing



Eventing is a set of APIs for routing events from **Producers** to **Consumers** (known as **Sinks**)

CloudEvent specification

allows for the creation of Serverless components that are driven by Event rather than Traffic



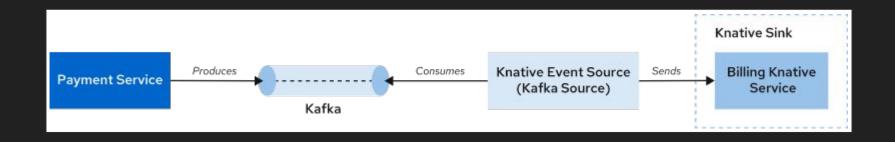
Knative Source



Knative Source:

- PingSource
- APIServerSource
- KafkaSource
- SinkBinding
- ..

```
kn source ping create <pingsource-name> \
    --namespace <namespace> \
    --schedule "<cron-schedule>" \
    --data '<data>' \
    --sink <sink-name>
```



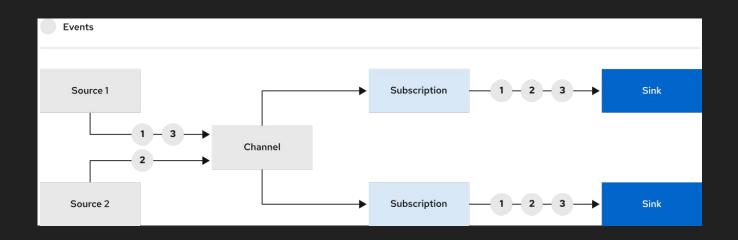
Channels and Subscriptions



provides a delivery mechanism for Cloudevents, through Subscriptions, to multiple destinations or Sinks

Sinks are consumers of events and can include Brokers and Knative Services

Subscriptions are used to hook together the Channel and the Sink



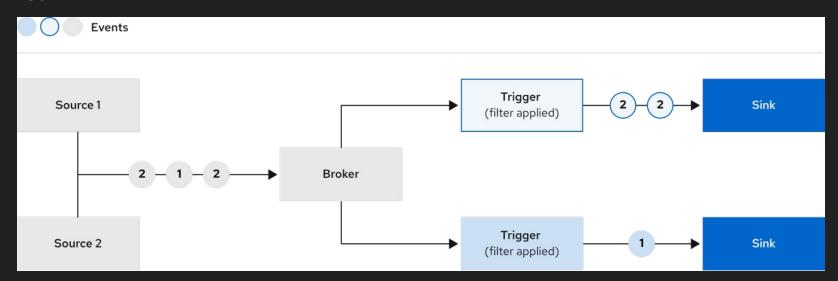
Brokers and Triggers



Brokers are namespace bound Cloudevent receivers/forwarders

Triggers are forwarders that push events to **Consumers**

Triggers use the cloudevent metadata (attributes) to filter cloudevents



Knative Eventing



kn broker create <broker-name>

kn trigger create <name> --sink <sink> --broker
broker-name> --filter <attribute>=<value>

kn source binding <name> --subject "Service:serving.knative.dev/v1:<service-name>" --sink broker:
broker-name>

Ephemeral or Persistent



Channel options (*):

- InMemoryChannel
- KafkaChannel
- NatssChannel

Broker options (*):

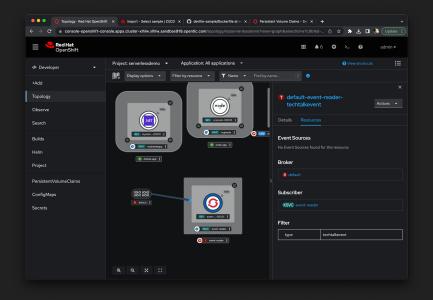
- Channel based broker
 - default
 - backed by InMemoryChannel
 - o ephemeral
- Knative Broker for Apache Kafka
- RabbitMQ broker

```
apiVersion: eventing.knative.dev/v1
kind: Broker
metadata:
  annotations:
    # case-sensitive
    eventing.knative.dev/broker.class: Kafka
    # Optional annotation to point to an externally
managed kafka topic:
    # kafka.eventing.knative.dev/external.topic:
<topic-name>
  name: default
  namespace: default
spec:
  # Configuration specific to this broker.
  config:
    apiVersion: v1
    kind: ConfigMap
    name: kafka-broker-config
    namespace: knative-eventing
```

^{*} Brokers/channels lists on https://knative.dev/docs/eventing/

Knative Eventing vs Serving



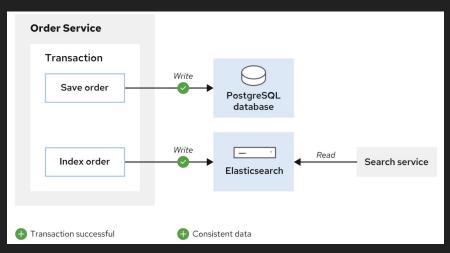


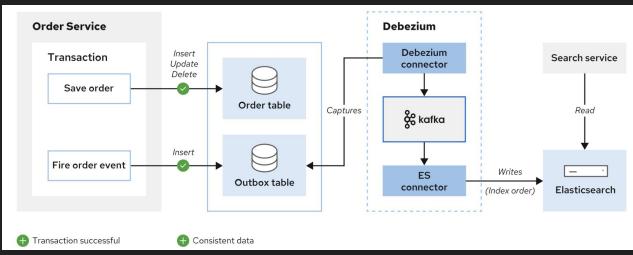
Serving allows us to create services that can scale dynamically

Eventing allows for the creation of Event-driven architectures

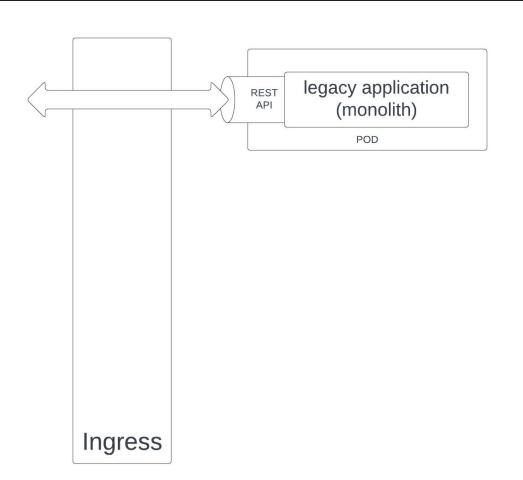
Knative Serving applications can act as **sinks** for Eventing => a combination of EDA and efficient resource Applications

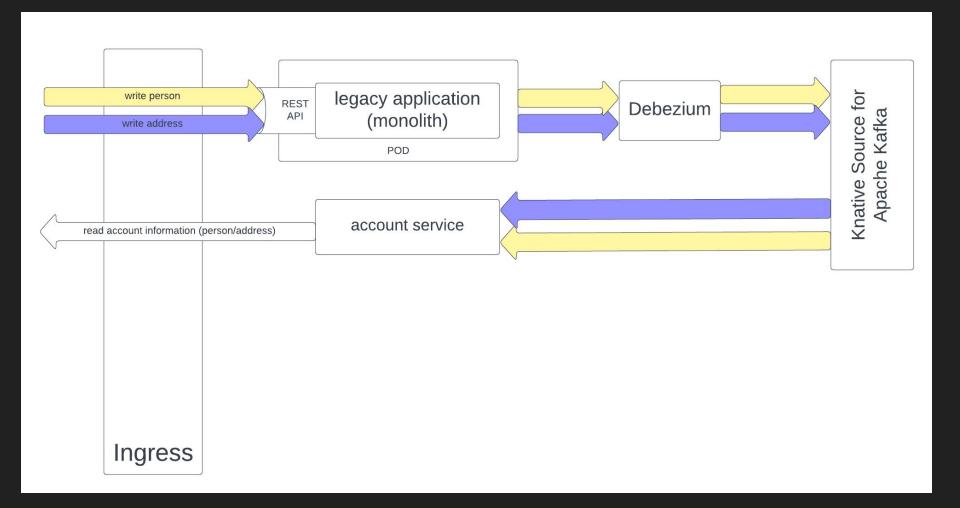


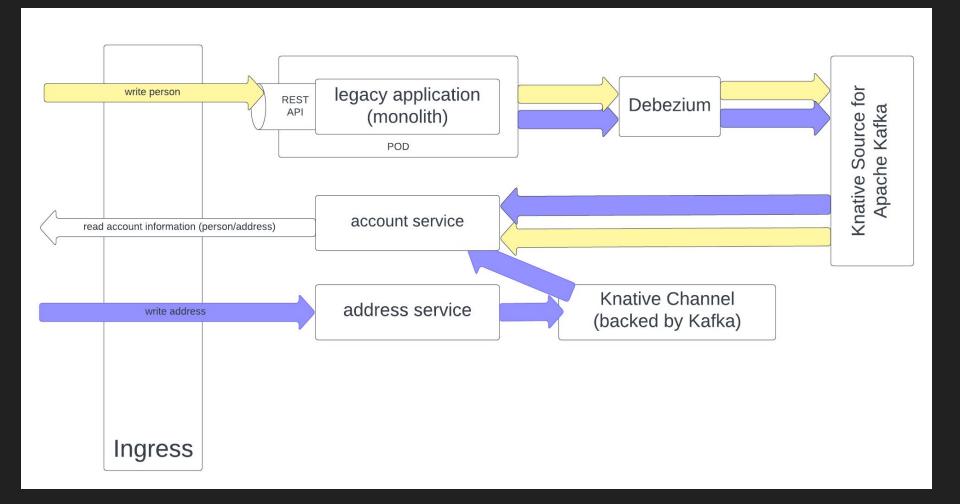


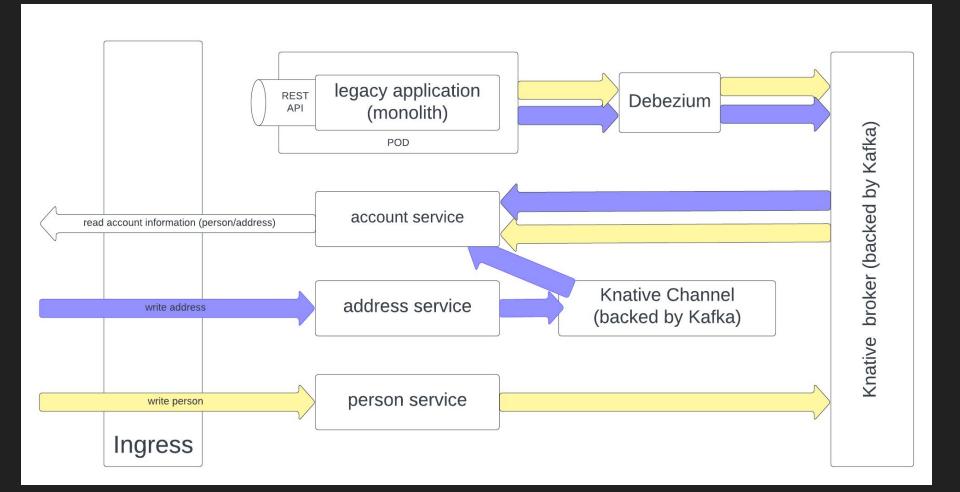


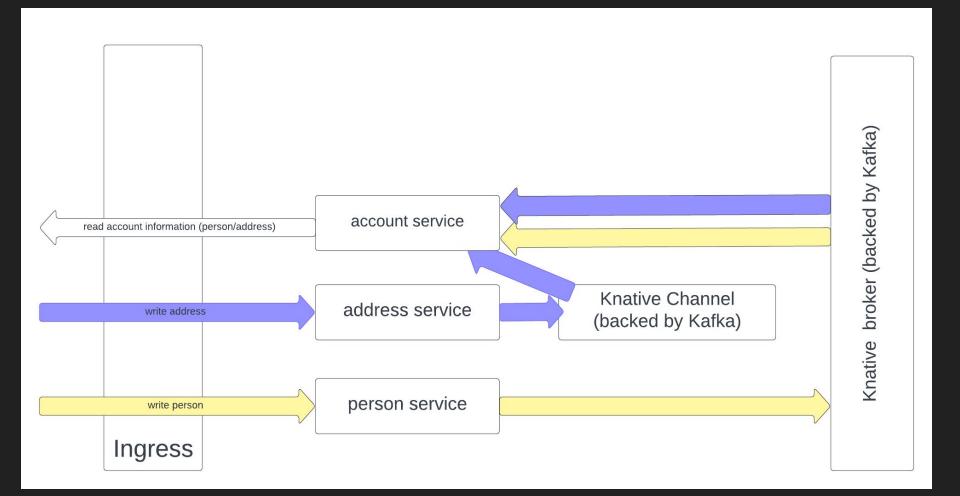
REST API legacy application (monolith)













Want a quick tinker?



Red Hat provides a free-to-use OpenShift experience at:

https://developers.redhat.com/developer-sandbox

