CLOUD NATIVE
COMPUTING FOUNDATION

How to Build Deep Learning Inference Through Knative Serverless Framework

Huamin Chen - Red Hat
Yehuda Sadeh-Weinraub - Red Hat

Dec, 2018

Overview

- Knative Overview
- Expand Knative: a Ceph RGW PubSub Case Study
- Write Knative Functions
- Deployment Instructions

Knative: a Kubernetes Native Serverless Framework

- Eventing: Reliable event delivery to single or multiple data sinks
- Serving: Route traffic to functions; Scale up/down function containers
- Build: source to image build steps and pipeline

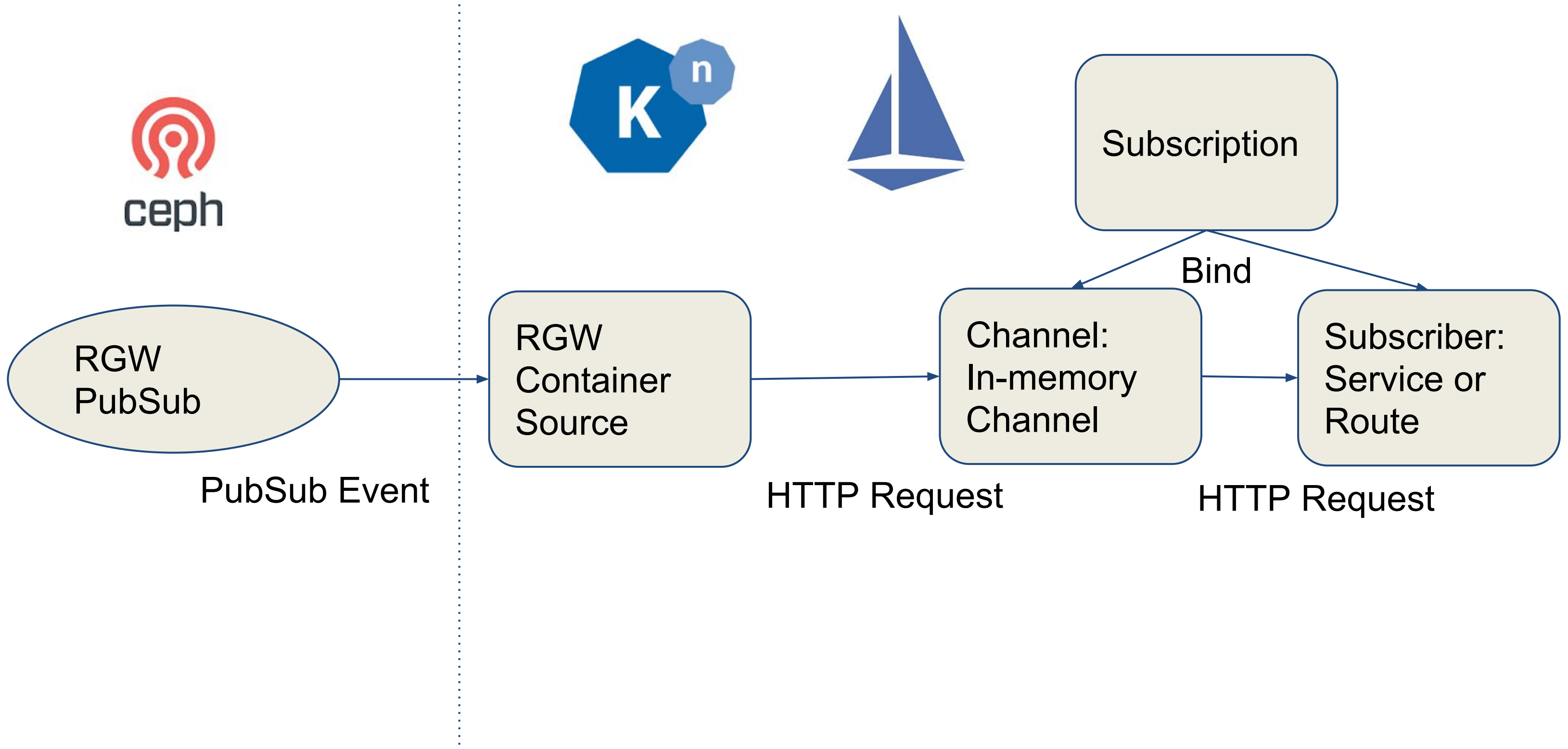
Eventing: Event Retrieval and Delivery

- CRDs
  - Channel
  - Subscription
  - Sources
    - ContainerSource, Cron Jobs, K8s Events, GCP PubSub, Github...
- Controllers
  - Channel Provisioner and Message Dispatcher
  - Channels:
    - In-memory
    - GCP PubSub
    - Apache Kafka
    - NATS
    - ...

Ceph RGW PubSub

- RGW multisite sync plugin
  - Hooks into change events tracking of all objects
  - Provides REST api
- Topic
  - Aggregates different published events
- Publish
  - Changes on bucket are published to a topic
    - Notification config: bucket, event type
- Subscribe
  - Can have multiple subscriptions to a single topic

Expand Eventing: Ceph RGW Pubsub Event Source



ContainerSource

- ContainerSource is a kind of Eventing Source for quick prototyping and deployment.
  - No need to add new Eventing Source CRD
  - Though more to be done: Secret
- Sink is automatically appended to the Container
  - Either as an arg, if not already used: --sink, or EnvVar: SINK

```go
func main() {
	target := flag.String("sink", "", "uri to send events to")
	flag.Parse()
	if target == nil || *target == "" {
		log.Fatalf("No sink target")
	}
	......
	postMessage(*target, &e)
	......
}
```

Get --sink argument

Post message to sink

Source at https://github.com/ceph/rgw-pubsub-api/blob/master/go/examples/knative-eventing-source/container-source/main.go

Source

RGW Event Source is based on ContainerSource

```
apiVersion: sources.eventing.knative.dev/v1alpha1
kind: ContainerSource
metadata:
 labels:
   controller-tools.k8s.io: "1.0"
 name: containersource-rgwpubsub
 namespace: rgwpubsub
spec:
 image: docker.io/rootfs/rgwpubsub-knative-source
 sink:
   apiVersion: eventing.knative.dev/v1alpha1
   kind: Channel
   name: rgw-ps-channel
```

Image Source

Eventing Source: https://github.com/ceph/rgw-pubsub-api/tree/master/go/examples/knative-eventing-source/container-source

Channel

```yaml
apiVersion: eventing.knative.dev/v1alpha1
kind: Channel
metadata:
 name: rgw-ps-channel
 namespace: rgwpubsub
spec:
 provisioner:
   apiVersion: eventing.knative.dev/v1alpha1
   kind: ClusterChannelProvisioner
   name: in-memory-channel
```

Use in-memory-channel

Subscription

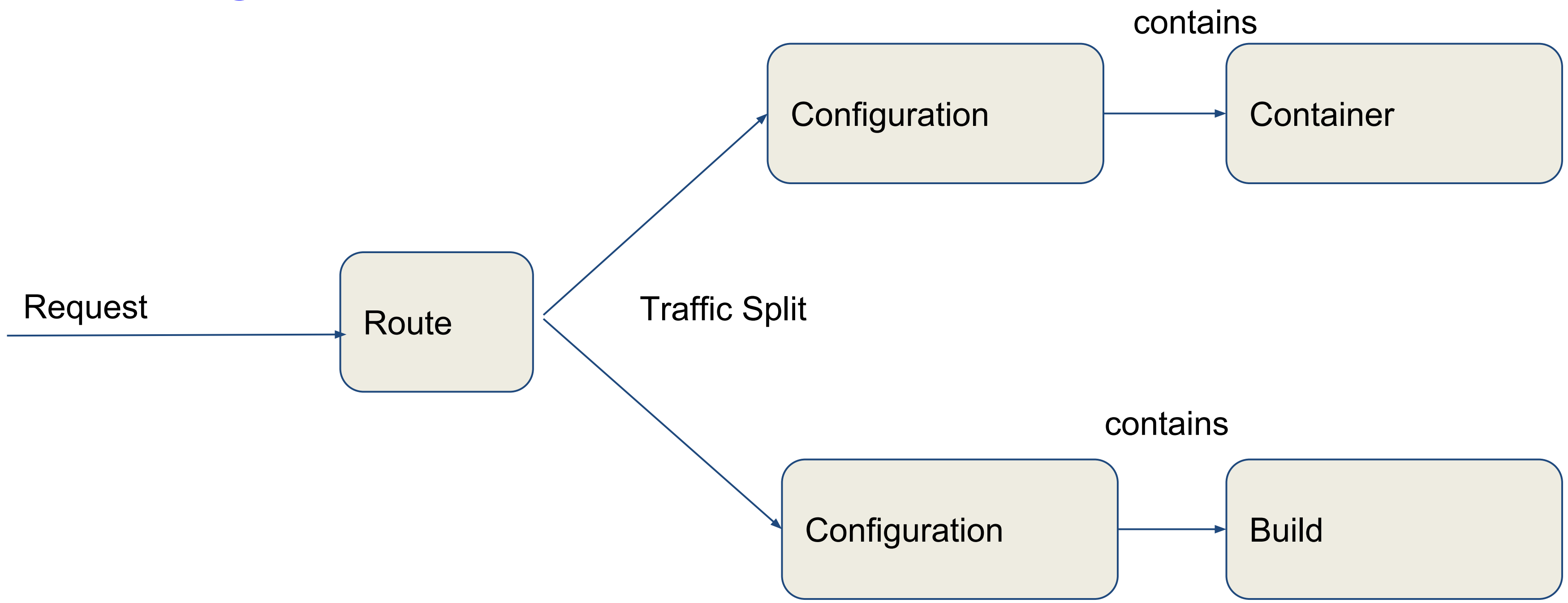
```
apiVersion: eventing.knative.dev/v1alpha1
kind: Subscription
metadata:
 name: rgw-ps-subscription
 namespace: rgwpubsub
spec:
 channel:
   apiVersion: eventing.knative.dev/v1alpha1
   kind: Channel
   name: rgw-ps-channel
 subscriber:
   ref:
     apiVersion: serving.knative.dev/v1alpha1
     kind: Service
     name: rgwpubsub-svc
```

Service CRD

Serving: Serverless Functions

- CRDs
  - Route
    - How traffic is routed
  - Configuration
    - How Function Container is configured
  - Revision
    - Identify Container name or Build to use
  - Service
    - Route + Configuration

Serving: A Simplified View



Serving: Service

```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
 name: rgwpubsub-svc
 namespace: rgwpubsub
spec:
 runLatest:
   configuration:
     revisionTemplate:
       spec:
         container:
           image: docker.io/rootfs/rgwpubsub-knative-receiver
```

Serving Function

Receiver Function source: https://github.com/ceph/rgw-pubsub-api/tree/master/go/examples/knative-eventing-source/receiver

Change Serving Function

```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
 name: rgwpubsub-svc
spec:
 runLatest:
   configuration:
     revisionTemplate:
       spec:
         container:
           image: docker.io/rootfs/rgwpubsub-knative-vision
```

Serving Function

Vision Function Source: https://github.com/ceph/rgw-pubsub-api/tree/master/go/examples/knative-eventing-source/vision

Serving: Route

```yaml
apiVersion: eventing.knative.dev/v1alpha1
kind: Subscription
metadata:
 name: rgw-ps-subscription
spec:
 channel:
   apiVersion: eventing.knative.dev/v1alpha1
   kind: Channel
   name: rgw-ps-channel
 subscriber:
   ref:
     apiVersion: serving.knative.dev/v1alpha1
     kind: Route
     name: rgwpubsub-route
```

Route CRD

Route: Split Traffic

Route CRD

Reference to Configurations

```
apiVersion: serving.knative.dev/v1alpha1
kind: Route
metadata:
 name: rgwpubsub-route
spec:
traffic:
- configurationName: google-vision-configuration
  percent: 50
- configurationName: resnet-configuration
  percent: 50
```

Functions

- Function Entrypoint: HTTP Request
- Can write in Golang, Python, Javascript…

```
func handler(ctx context.Context, e *rgwpubsub.RGWEvent) {
    metadata := cloudevents.FromContext(ctx)
    log.Printf("Object: %q  Bucket: %q", e.Info.Key.Name, e.Info.Bucket.Name)
}
func main() {
    http.ListenAndServe(":8080", cloudevents.Handler(handler))
}
```

Business logic comes here

Entrypoint

Function examples https://github.com/ceph/rgw-pubsub-api/blob/master/go/examples/knative-eventing-source/receiver/main.go

ResNet Function Skeleton

TF Serving gRPC client

```
import (
     "github.com/ceph/rgw-pubsub-api/go/examples/knative-eventing-source/resnet-grpc/pkg/resnet"
)
func getAnnotation(bucket, key string) {
      reader, err := rgwDownloader.Download(bucket, key)
      tp := resnet.Predict(servingEndpoint, reader)
      log.Printf("classes: %v", tp.Int64Val)
}
func handler(ctx context.Context, e *rgwpubsub.RGWEvent) {
      metadata := cloudevents.FromContext(ctx)
      getAnnotation(e.Info.Bucket.Name, e.Info.Key.Name)
}
func main() {
      http.ListenAndServe(":8080", cloudevents.Handler(handler))
}
```

Download RGW Object

Image Classification

Business logic

Entrypoint

Source: https://github.com/ceph/rgw-pubsub-api/blob/master/go/examples/knative-eventing-source/resnet-grpc/resnet-grpc.go

Google Vision Function Skeleton

Google Vision REST client

Download RGW Object

Image Classification

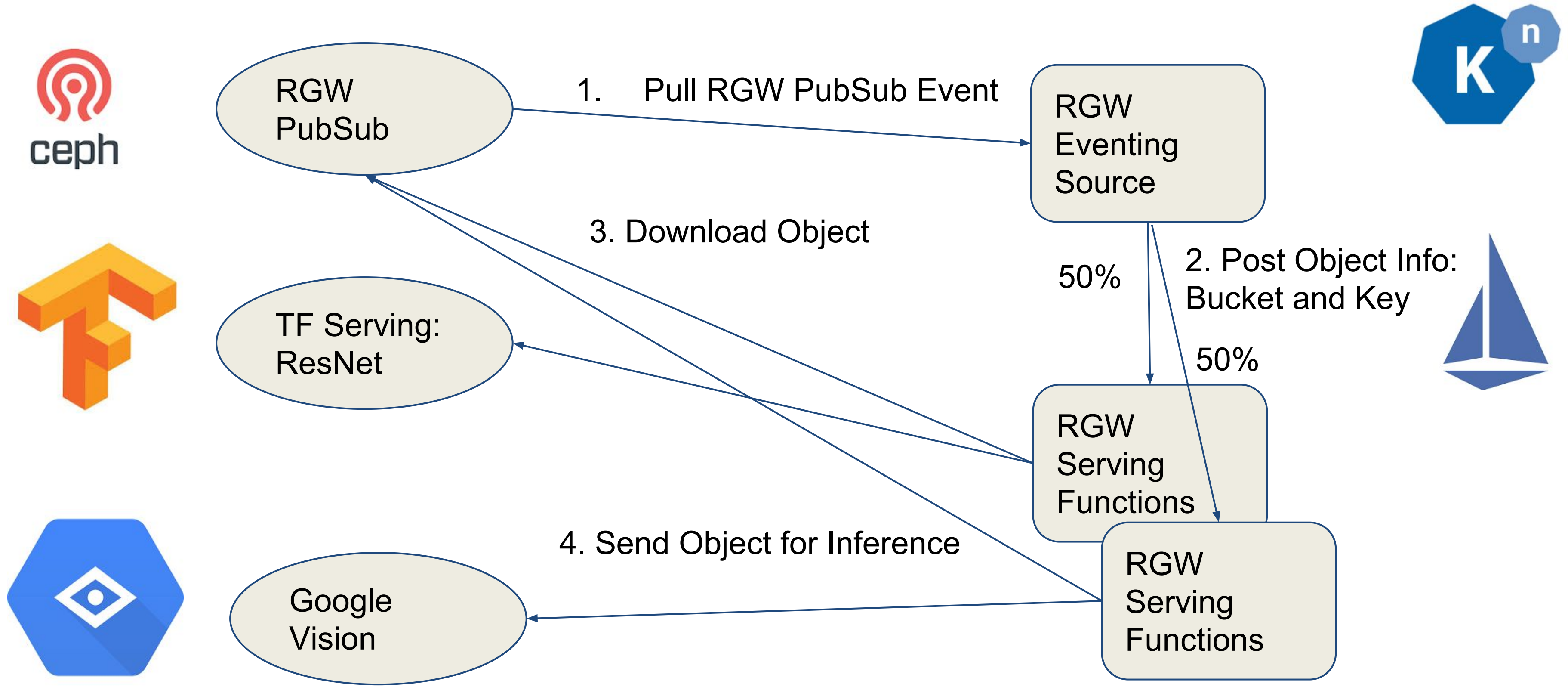
Business logic

Entrypoint

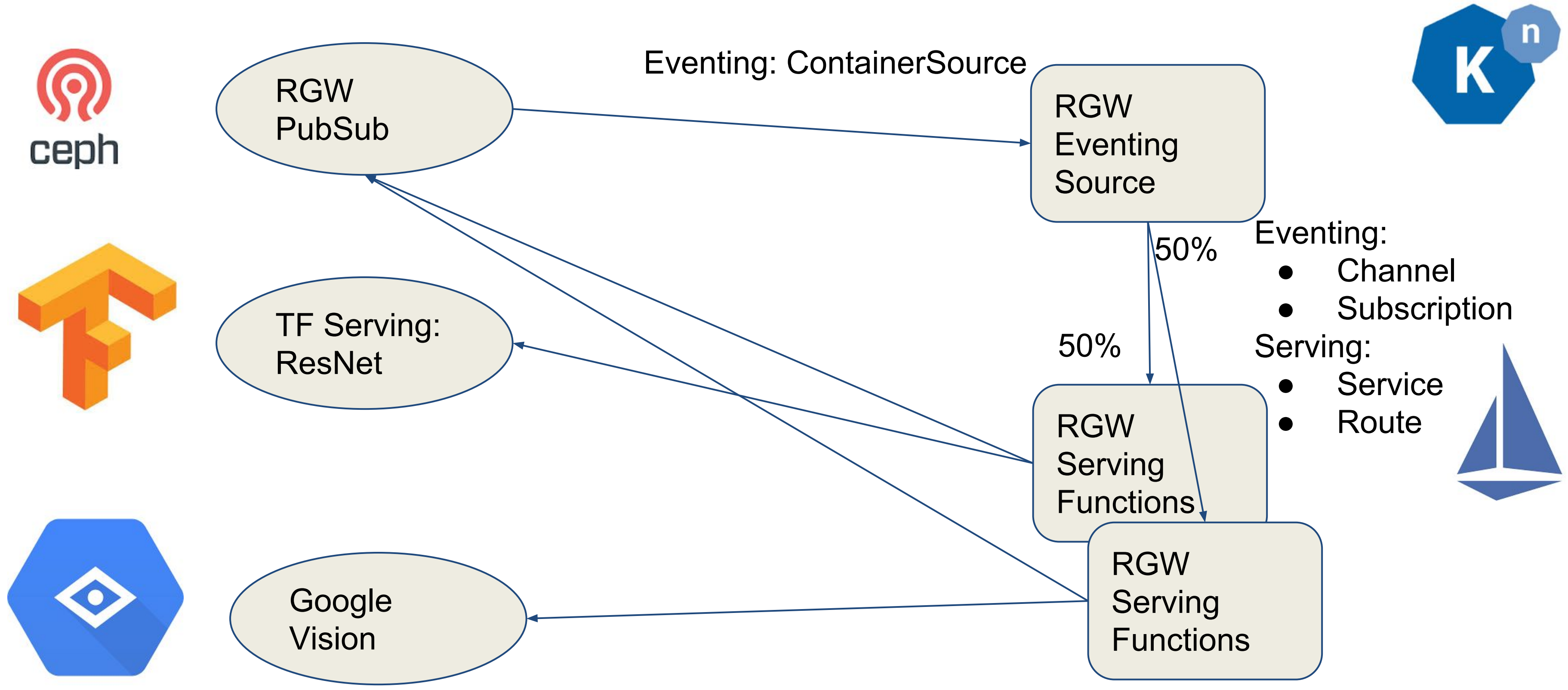
```go
import (
    "github.com/ceph/rgw-pubsub-api/go/examples/knative-eventing-source/vision/pkg/googlevision"
)
func getAnnotation(bucket, key string) {
    reader, err := rgwDownloader.Download(bucket, key)
    if annotations := googlevision.AnnotateImage(apiKey, *numAnnInt, reader); len(annotations) > 0 {
        for i := 0; i < len(annotations); i++ {
            label := annotations[i].Description
            score := annotations[i].Score
            log.Printf("label: %s, Score: %f\n", label, score)
        }
    }
}
func handler(ctx context.Context, e *rgwpubsub.RGWEvent) {
    metadata := cloudevents.FromContext(ctx)
    getAnnotation(e.Info.Bucket.Name, e.Info.Key.Name)
}
func main() {
    http.ListenAndServe(":8080", cloudevents.Handler(handler))
}
```

Source: https://github.com/ceph/rgw-pubsub-api/blob/master/go/examples/knative-eventing-source/vision/vision.go

Put Everything Together



Knative's View



Eventing:

- Channel
- Subscription

Serving:

- Service
- Route

Deploy Instructions

- Create Eventing Source

```
kubectl apply -f deploy/sources_v1alpha1_containersources_rgwpubsub.yaml
```

- Create Channel

```
kubectl apply -f deploy/channel.yaml
```

- Create Subscription, Route, Configuration, Service Entry and Secret

```
kubectl apply -f deploy/split-traffic/route.yaml
```

Details at https://github.com/ceph/rgw-pubsub-api/tree/master/go/examples/knative-eventing-source

Test it

`Upload Some images to RGW`

```
# wget https://r.hswstatic.com/w_907/gif/tesla-cat.jpg
# for i in $(seq 1 10); do ./s3 put buck/cat-${i}.jpg --in-file=./tesla-cat.jpg; done
```

Serving Function Logs

ResNet Serving function:

ResNet Function

```
# kubectl logs -lserving.knative.dev/configuration=resnet-configuration -c user-container
2018/12/03 15:20:49 Ready and listening on port 8080
2018/12/03 15:24:10 [2018-12-03T15:24:10Z] application/json rgwpubsub. Object: "cat-7.jpg"  Bucket:
"buck"
2018/12/03 15:24:11 classes: [286]
```

Cat #7

Class 286 in ImageNet is ‘cougar, puma, catamount, mountain lion, painter, panther, Felis concolor’

Google Vision Serving function:

Google Vision Function

```
# kubectl logs -lserving.knative.dev/configuration=google-vision-configuration -c user-container
2018/12/03 15:20:48 Ready and listening on port 8080
2018/12/03 15:24:11 [2018-12-03T15:24:11Z] application/json rgwpubsub. Object: "cat-1.jpg"  Bucket:
"buck"
2018/12/03 15:24:11 label: cat, Score: 0.993347
```

Cat #1

What Next?

- More Libraries and Functions for More Use Cases
  - Auditing and Compliance
  - Vulnerabilities Detection
  - RGW Object Metadata tagging
- Support More Event Types
- Make RGW PubSub Easily Deployable
  - Through Rook