



Moving Spotify's Infrastructure Management Up the Stack

with kro and kpop

KubeCon+CloudNativeCon Amsterdam 2026



Alexander Buck

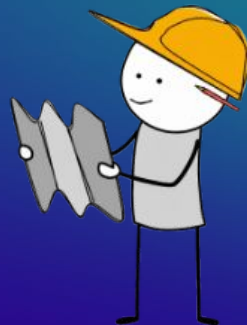


Tomas Aschan

We provide feature teams with less complex and more powerful abstractions, and allow platform teams to evolve their products independently



Feature developer



Platform developer



Managing 1M Resources

Designing the platform to manage change at scale

KubeCon Atlanta 2025



Fredrik Sommar



Oliver Soell



KubeCon



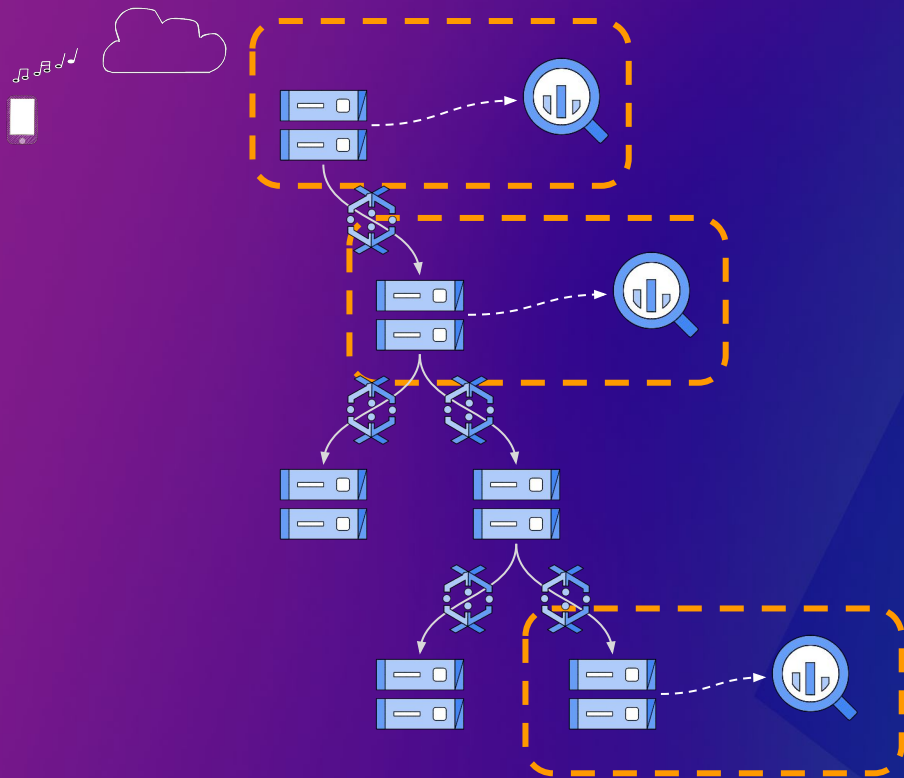
CloudNativeCon

— North America 2025 —



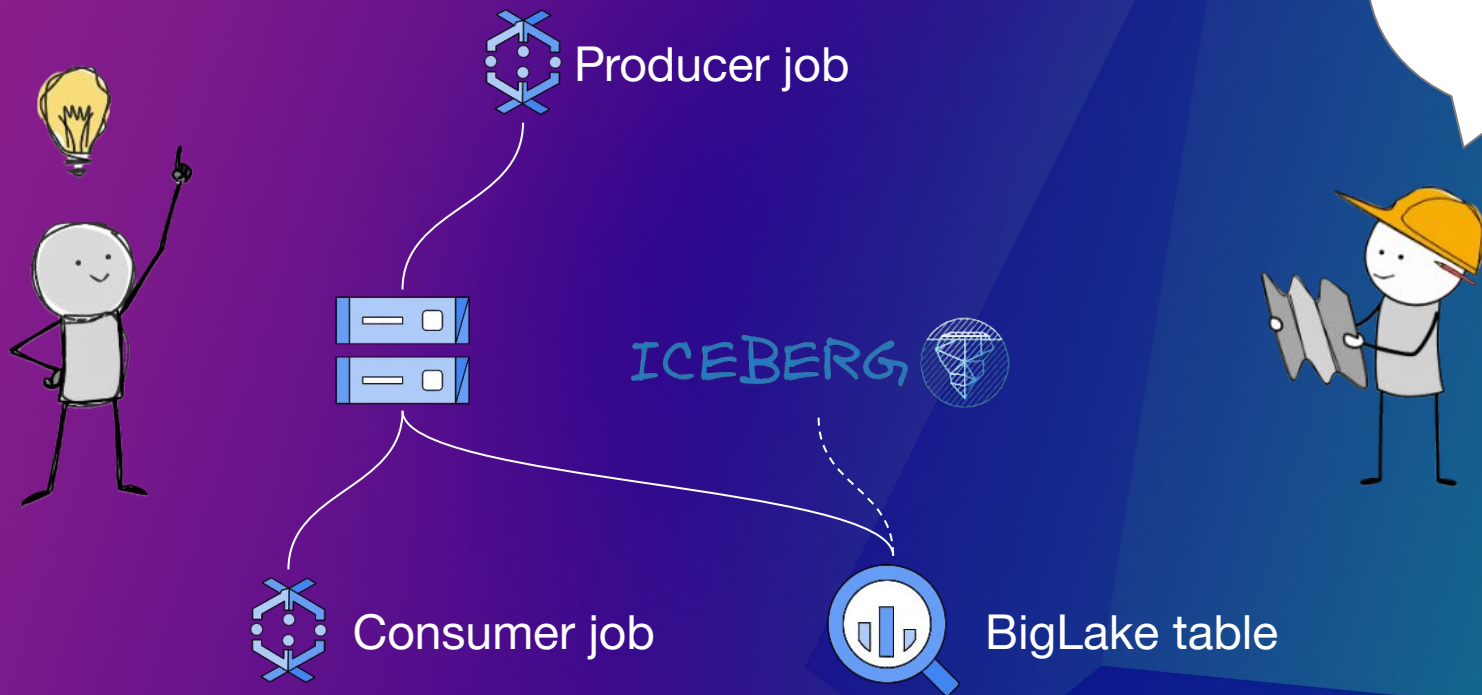
<https://youtu.be/Vg2FiJZReDg>

Case Study: Reducing Data Duplication



28%

Solution: We use BigLake tables to point to underlying data in GCS



Great! How do we set this up?



```
apiVersion: bigquery.cnrm.cloud.google.com/v1beta1
kind: BigQueryTable
metadata:
  name: iceberg-events-table
  namespace: kubecon-eu-21
spec:
  resourceID: iceberg-events-table
  datasetRef:
    name: iceberg-events-dataset
  description: "Iceberg events table"
  requirePartitionFilter: true
  externalDataConfiguration:
    autodetect: true
    connectionId: projects/kubecon-eu-21/locations/eu/connections/bigquery-connection-resource
    sourceFormat: ICEBERG
    sourceUris:
      - gs://iceberg-platform/events-endpoint/metadata/latest.metadata.json
```

bindings:

- role: roles/
- members:
- member: "serviceAccount:bigquery-connection-resource"

kubecon-eu-21 iam.googleapis.com

Interlude: Using Cloud Bigtable

```
apiVersion: bigtable.cnrm.cloud.google.com/v1beta1
kind: BigtableInstance
metadata:
  name: my-instance
  namespace: kubecon-eu-26
spec:
```

```
  cluster:
    - clusterId: my-instance
      zone: europe-west1-d
```

```
apiVersion: storage.spotify.com/v1alpha1
kind: BigtableBackupSchedule
metadata:
  name: my-instance-backups
  namespace: kubecon-eu-26
spec:
  instanceId: my-instance
```

```
apiVersion: iam.cnrm
kind: IAMPartialPolicy
metadata:
  name: my-instance
  namespace: kubecon
```

```
spec:
  bindings:
```

```
apiVersion: storage.spotify.com/v1alpha1
kind: DatabaseAutoscaler
metadata:
  name: my-instance-gew1d
  namespace: kubecon-eu-26
spec:
  duration:
```

```
apiVersion: bigtable.cnrm.cloud.google.com/v1beta1
kind: BigtableAppProfile
metadata:
  name: my-instance-euw1
  namespace: kubecon-eu-26
spec:
  resourceID: europe-west1
  instanceRef:
```

```
on-eu-26/instances/my-instance/clusters/my-instance-euw1
```

Interlude: Using Cloud Bigtable - But Good!

```
apiVersion: bigtable.cnrm.cloud.google.com/v1beta1
kind: BigtableInstance
metadata:
  name: my-instance
  namespace: kubecon-eu-26
spec:
  cluster:
    - clusterId: my-instance
      zone: europe-west1
```

```
apiVersion: storage.spotify.com/v1
kind: SpotifyBigtable
metadata:
  name: my-instance
  namespace: kubecon-eu-26
spec:
  access:
    users:
      - serviceAccount: kubecon-eu-26@service-project.iam.gserviceaccount.com
  backupStatus: enabled
  regions:
    - region: europe-west1
```

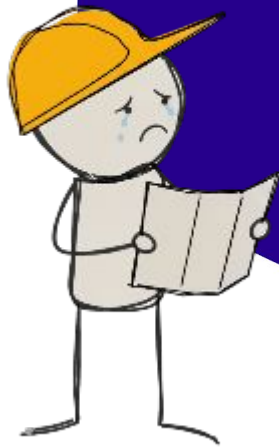
```
d.google.com/v1beta1
```

```
apiVersion: iam.cnrm.cloud.google.com/v1beta1
kind: IAMPolicy
metadata:
  name: my-instance
  namespace: kubecon-eu-26
spec:
  bindings:
    - role: roles/iam.policyAdmin
      members:
        - serviceAccount:kubecon-eu-26@service-project.iam.gserviceaccount.com
```

```
apiVersion: storage.spotify.com/v1alpha1
kind: DatabaseAutoscaler
metadata:
  name: my-instance-gew1d
  namespace: kubecon-eu-26
spec:
  duration:
```

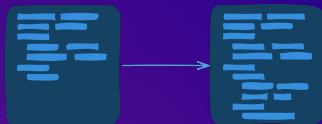
```
con-eu-26/instances/my-instance-gew1d/clusters/my-instance-euw1
```

...or not good



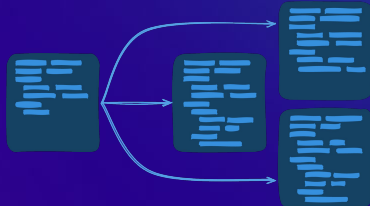
envtest
GVK typed client
event recording
client-go
CRD
kubectrl
markers
level-triggered
RBAC
make manifests
exponential backoff
observed state
watches
leader election
owner references
test environment
SetControllerReference
kustomize
work queue
cache sync
predicates
admission control
generation
cluster scoped
drift detection
namespace scoped
conditions
manager
scheme
idempotency
status
subresource
rate limiting
informer
CreateOrUpdate
code generation
desired state
operator pattern
Custom Resource
kubebuilder
controller-runtime
Reconciler
finalizers
webhook
API machinery

Generalizing the problem



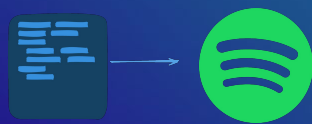
Abstractions

*What do I want
instead of
how do I get it*



Compositions

Bundle configuration for
improved ergonomics



API Calls

Expose new resource
types as CRDs



Config Connector (KCC)

Third-party operators to
configure third-party
resource types

kro

Kube Resource Orchestrator



No Code

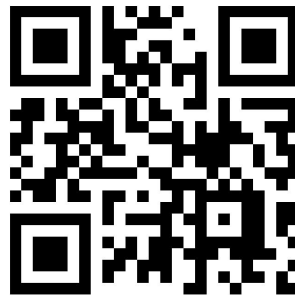
KRM and CEL -
the platform team
writes YAML!

Open Source

SIG Cloud
Provider

with active
maintainership
from AWS, Azure,
GCP and others

 kro.run





Kpop

Kubernetes **P**rotobuf **O**perators

Protobuf

A contract between platform developer and platform maintainer

CLI

Easy testing and validation without a cluster creates a tight inner loop

Deployment

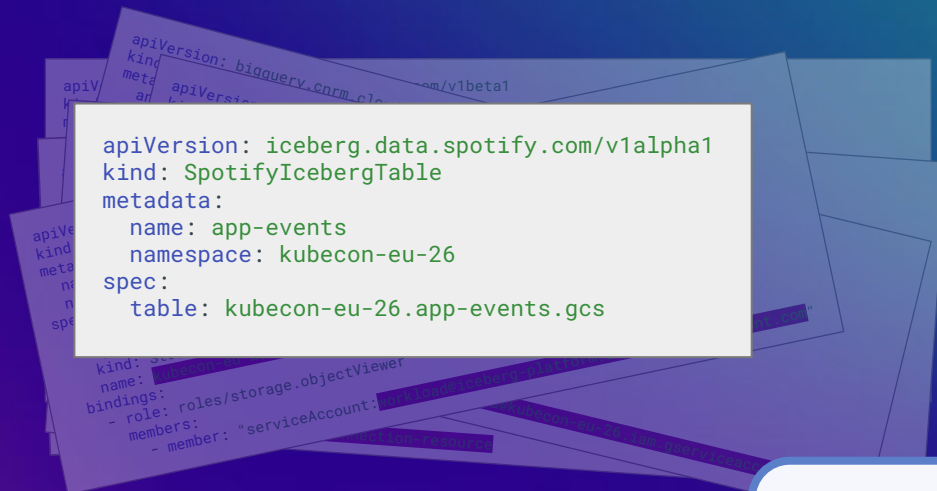
The platform manages controllers, which call out to gRPC services





Live Demo

kro glues together kpop and GCP resources





Hypothesis:

**kro + kpop solve more than 90% of use cases
for platform teams at Spotify**

and they enjoy using them!

What's next?



Thank you!