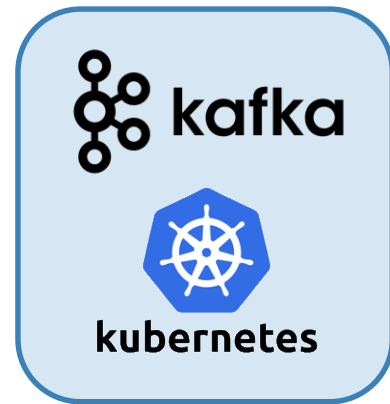


Running **Kafka** on Kubernetes GCP



Just because you *can* do it,
... *should* you be doing this?



Simon Aubury
[linkedin.com/in/simonaubury](https://www.linkedin.com/in/simonaubury)



Hello!

*I am **Simon Aubury***

Principal Data Engineer @ ThoughtWorks

I am here because I love streaming
(& really don't like ops)

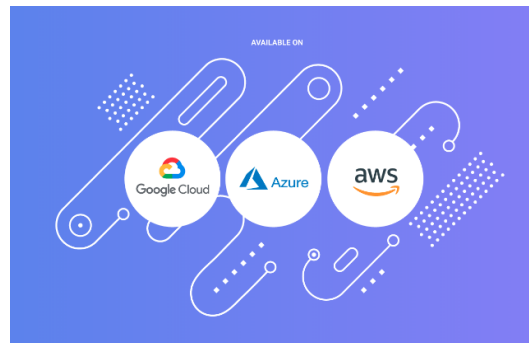


Let's start with a **confession ...**

- I'm wasting your time
- Use a managed service for Kafka
- Let's talk about plan "B"

Principle 3: Favor managed services

Cloud is more than just infrastructure. Most cloud providers offer a rich set of managed services, providing all sorts of functionality that relieve you of the headache of managing the backend software or infrastructure. However, many organizations are cautious about taking advantage of these services because they are concerned about being 'locked in' to a given provider. This is a valid concern, but managed services can often save the organization hugely in time and operational overhead.



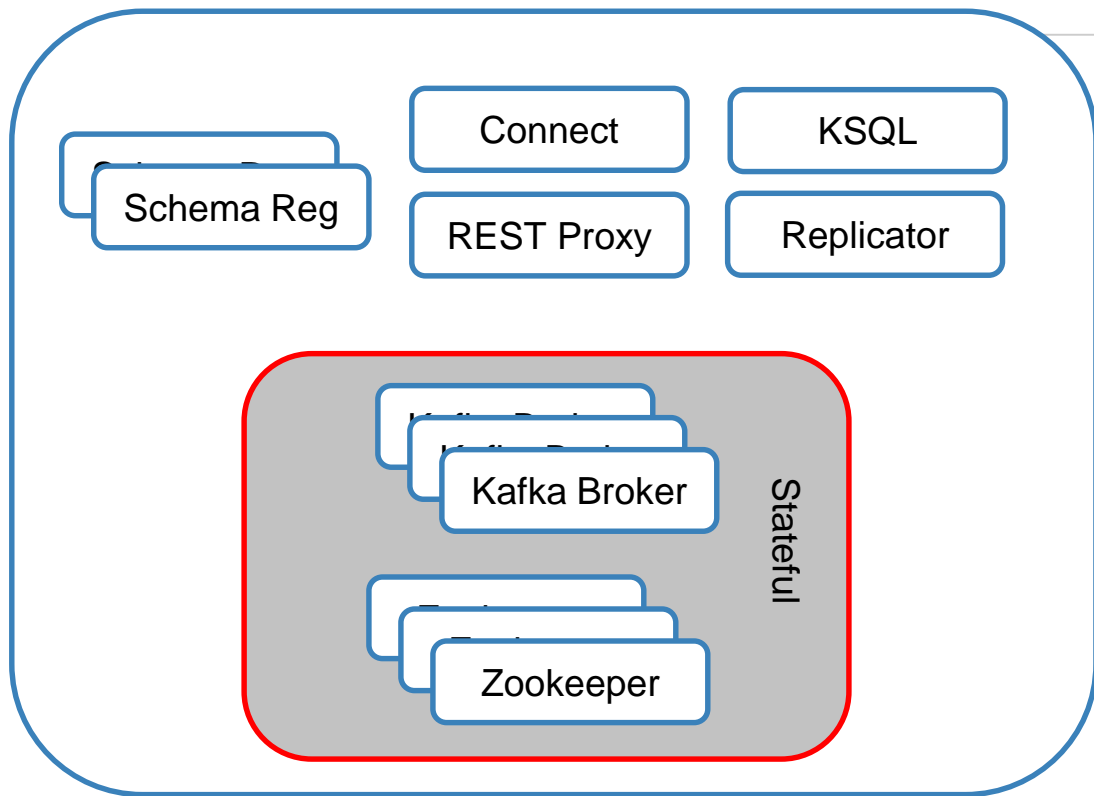


The Kafka typical build ... **has changed**

- Apache Kafka is a distributed event streaming platform
- What you “used to” do
- Stateful apps meet containers
- Kubernetes Operators

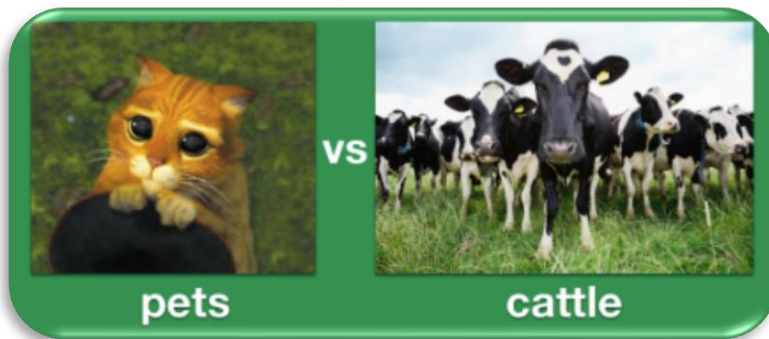


The usual setup



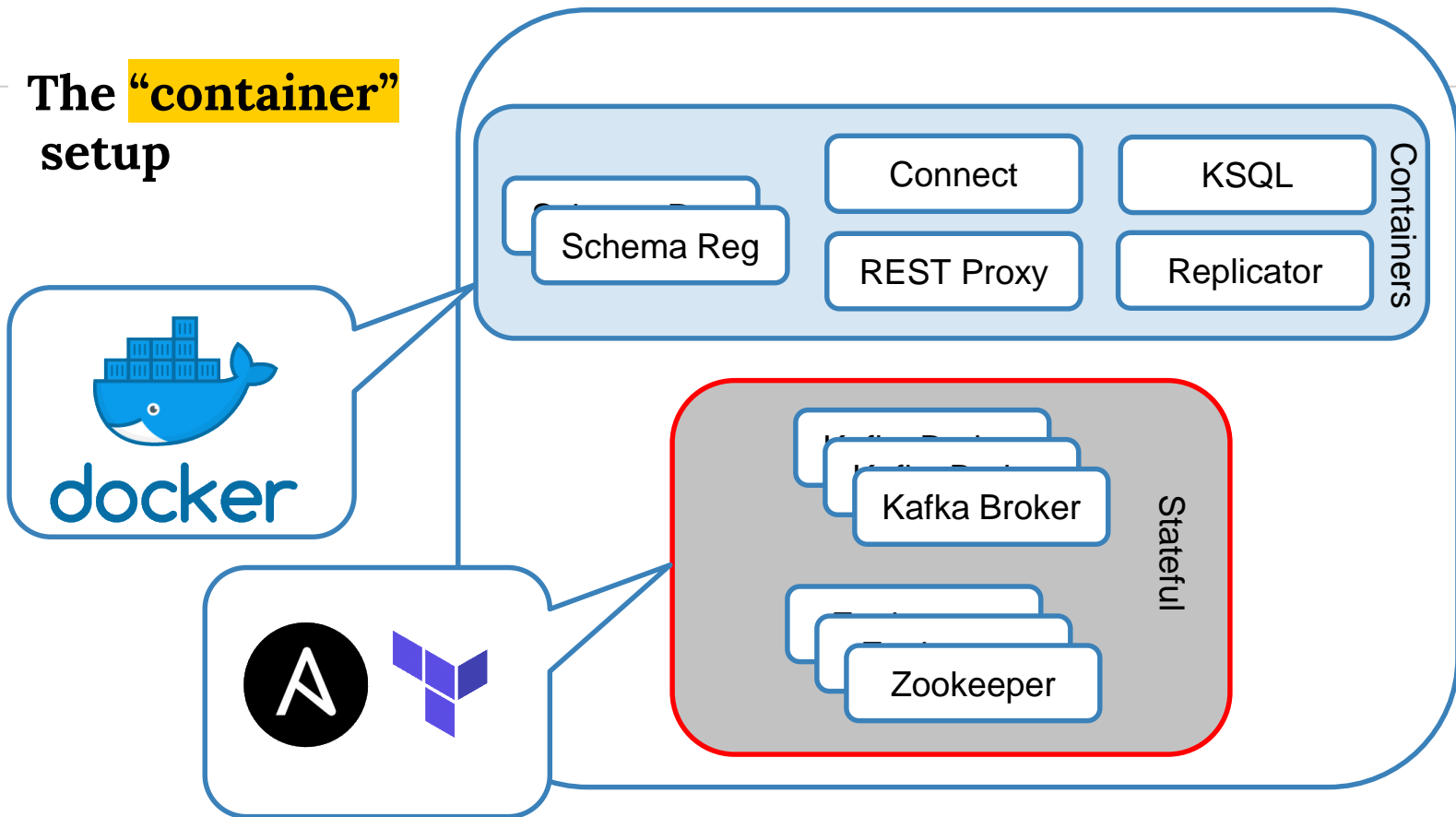


Managing state is hard





The “container” setup





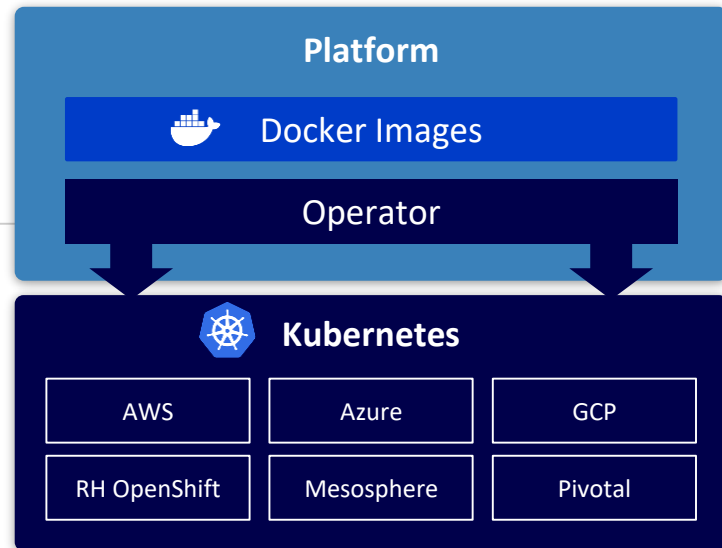
Kubernetes state **problem** ...

- Kubernetes has become the standard for orchestrating containerized applications
- Running stateful applications such as Kafka can be “challenging”



What's an **Operator**?

- Is a method of packaging, deploying & managing a k8s application
- Run complex (stateful) cloud-native apps on k8s
 - Adding a brokers or upgrade a cluster
 - Security and authentication
 - Networking
 - Configure storage!





And a **good operator** should ...



Container images



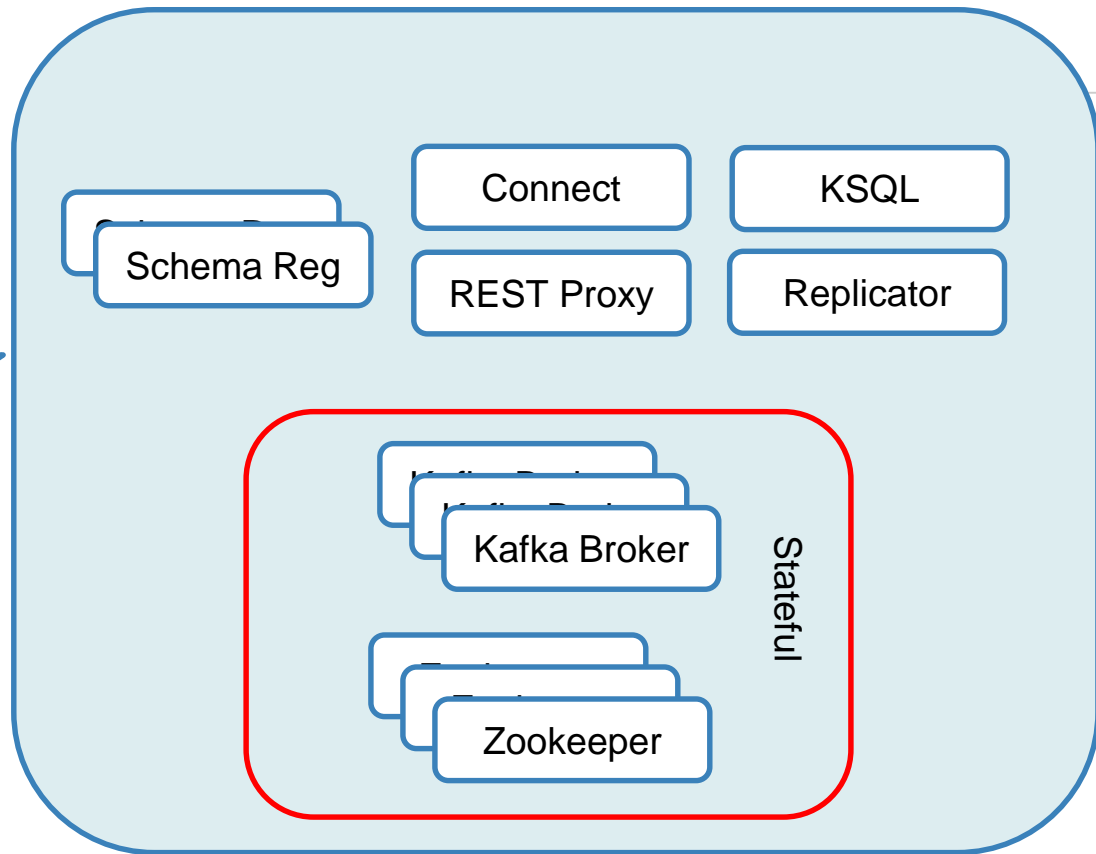
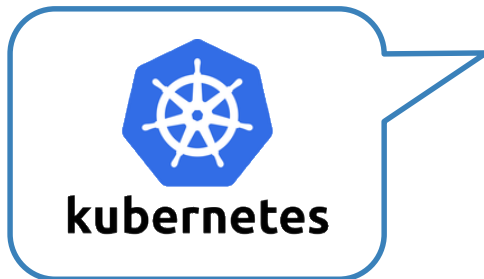
Operator Helm Charts
- yaml



- Do sensible things in a scalable, repeatable way
- Eg., rolling upgrade Kafka brokers
 - Stop the broker, upgrade Kafka
 - Wait for partition leader reassignment
 - Start the upgraded broker
 - Wait for zero under-replicated partitions
 - Upgrade the next broker



Kubernetes setup





Try this yourself ...

<http://bit.ly/kafkak8s>

- Kubernetes CLI setup -
- Google Cloud SDK –and setup CLI
 - Initialize
 - Set compute region & zone
 - Get credentials for cluster
- Download Operator – eg., Confluent Platform



Test **scaling** ...

Initial single broker Kafka cluster

```
helm install -f ./providers/gcp.yaml \  
  --name kafka --namespace operator \  
  --set kafka.enabled=true ./confluent-operator
```

```
kubectl get pods -n operator | grep kafka  
kafka-0                1/1      Running    0                47m
```

./providers/gcp.yaml

```
## Kafka Cluster  
##  
kafka:  
  name: kafka  
  replicas: 1  
  resources:  
    cpu: 200m  
    memory: 1Gi  
  loadBalancer:  
    enabled: true  
    domain: "mydevplatform.gcp.cloud"  
  tls:
```

enabled: false



☰

Google Cloud Platform

My First Project ▼

🔍

Kubernetes Engine

Clusters

Workloads

Services & Ingress

Applications

Configuration

Storage

Workloads

REFRESH

DEPLOY

DELETE

Workloads are deployable units of computing that can be created and managed in a cluster.

Is system object : False Filter workloads Columns

<input type="checkbox"/> Name ^	Status	Type	Pods	Namespace	Cluster
<input type="checkbox"/> cc-manager	OK	Deployment	1/1	operator	my-kafka-cluster
<input type="checkbox"/> cc-operator	OK	Deployment	1/1	operator	my-kafka-cluster
<input type="checkbox"/> connectors	OK	Stateful Set	1/1	operator	my-kafka-cluster
<input type="checkbox"/> controlcenter	OK	Stateful Set	1/1	operator	my-kafka-cluster
<input type="checkbox"/> kafka	OK	Stateful Set	1/1	operator	my-kafka-cluster
<input type="checkbox"/> ksql	OK	Stateful Set	2/2	operator	my-kafka-cluster
<input type="checkbox"/> schemaregistry	OK	Stateful Set	2/2	operator	my-kafka-cluster
<input type="checkbox"/> zookeeper	OK	Stateful Set	1/1	operator	my-kafka-cluster



Now **Scale!**

☉ Let's scale to 3

./providers/gcp.yaml

```
## Kafka Cluster
##
kafka:
  name: kafka
  replicas: 3
  resources:
    cpu: 200m
    memory: 1Gi
  loadBalancer:
    enabled: true
    domain: "mydevplatform.gcp.cloud"
  tls:
    enabled: false
    fullchain: |-
    privkey: |-
```

```
helm upgrade -f ./providers/gcp.yaml \
  --set zookeeper.enabled=true zookeeper ./confluent-operator
```

```
kubectl get pods -n operator | grep kafka
```

kafka-0	1/1	Running	0	68m
kafka-1	1/1	Running	0	5m57s
kafka-2	1/1	Running	0	4m36s



Google Cloud Platform

My First Project

Kubernetes Engine

Workloads

REFRESH

DEPLOY

DELETE

Clusters

Workloads

Services & Ingress

Applications

Configuration

Storage

Workloads are deployable units of computing that can be created and managed in a cluster.

Is system object : False Filter workloads

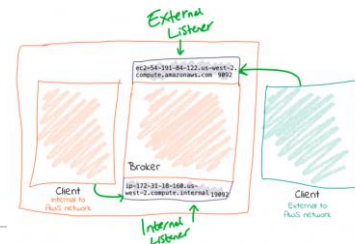
Columns

Name	Status	Type	Pods	Namespace	Cluster
cc-manager	OK	Deployment	1/1	operator	my-kafka-cluster
cc-operator	OK	Deployment	1/1	operator	my-kafka-cluster
connectors	OK	Stateful Set	1/1	operator	my-kafka-cluster
controlcenter	OK	Stateful Set	1/1	operator	my-kafka-cluster
kafka	OK	Stateful Set	3/3	operator	my-kafka-cluster
ksql	OK	Stateful Set	2/2	operator	my-kafka-cluster
schemaregistry	OK	Stateful Set	2/2	operator	my-kafka-cluster
zookeeper	OK	Stateful Set	3/3	operator	my-kafka-cluster



Broker connection strings and **listeners** ...

- Data in Kafka read from and written to the leader
 - A client (producer/consumer) will request metadata about which broker is the leader
 - The metadata returned will include the endpoints
 - The client will then use those endpoints
- TL;DR – Kafka broker need to know it's internal *and* external end-points



```
## Kafka Cluster
##
kafka:
  name: kafka
  replicas: 3
  resources:
    cpu: 200m
    memory: 1Gi
  loadBalancer:
    enabled: true
    domain: "mydevplatform.gcp.cloud"
  ...
  enabled: false
  fullchain: |-
  privkey: |-
  cacerts: |-
  metricReporter:
    enabled: false
```

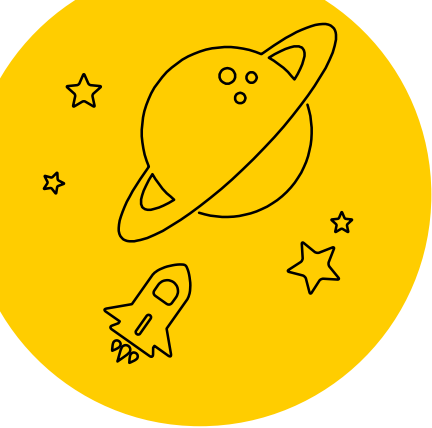


And the **issues** ...

! Create node pool "pool-4" in Kubernetes 12 hours ago
Engine cluster "my-cluster"
My First Project

Insufficient project quota to satisfy request: resource "CPUS_ALL_REGIONS": request requires '6.0' and is short '1.0'. project has a quota of '32.0' with '5.0' available. View and manage quotas at <https://console.cloud.google.com/iam-admin/quotas?usage=USED&project=fine-transit-248500>.

- Could not provision enough resource due to a quota on CPUs.
 - Navigate to IAM Admin Quotas and increase the Compute Engine API CPUs maximum
- Unable to schedule pods
 - Add a lot of capacity
 - I found 16 nodes of 2 vCPU's was enough to get going



So in **summary** ...

- Use a managed service for Kafka
- If you can't – use a Kubernetes Operator
 - Saves a lot of time!
- Test & get familiar with
 - Scaling and upgrades
 - Security and authentication
 - Networking



Thanks!

Any **questions** ?



[linkedin.com/in/simonaubury](https://www.linkedin.com/in/simonaubury)



github.com/saubury

<http://bit.ly/kafkak8s>

Presentation template by SlidesCarnival