Cloud-Native Streaming Platform:

Kafka on Kubernetes and Kafka as a Managed Service in the Cloud

@Hellmar_Becker @confluentinc
About Me

- Hellmar Becker
- Systems Engineer @ Confluent
- Formerly at Hortonworks, ING Bank (Amsterdam), IBM
- Lives near Munich
- Github: github.org/hellmarbecker
- Twitter: @Hellmar_Becker
Kubernetes has made huge improvements in the ability to run stateful workloads including databases and message queues, but I still prefer not to run them on Kubernetes.

https://twitter.com/kelseyhightower/status/963413508300812295
Kubernetes can only meet stateful workloads half way and I lack the expertise to manage a production configuration of Kafka, RabbitMQ, or Postgres on static infrastructure, let alone a Kubernetes cluster.

9:06 AM - 13 Feb 2018

18 Retweets  71 Likes

https://twitter.com/kelseyhightower/status/963414038603427840
DON'T DESPAIR...

“… not even over the fact that you don’t despair.

Just when everything seems over with, new forces come marching up, and precisely that means that you are alive”

Franz Kafka
KAFKA STREAMING ARCHITECTURE FUNDAMENTALS
Event Streaming Platform Architecture

- Kafka Brokers
- Zookeeper Nodes
- Schema Registry
- Kafka Connect
- REST Proxy
- Application
  - Native Client library
  - Kafka Streams
- KSQL
  - Kafka Streams
- Load Balancer *

@HELLMAR_BECKER  |  @CONFLUENTINC
KUBERNETES FUNDAMENTALS
2014 - We must adopt #microservices to solve all problems with monoliths
2016 - We must adopt #docker to solve all problems with microservices
2018 - We must adopt #kubernetes to solve all problems with docker

https://twitter.com/sahrizv/status/1018184792611827712
**ORCHESTRATION**

Compute
Networking
Storage
Service Discovery
Kubernetes

Schedules and allocates resources
Networking between Pods
Storage
Service Discovery
Refresher - Kubernetes Architecture

[kubectl]

https://thenewstack.io/kubernetes-an-overview/

@HELLMAR_BECKER  /  @CONFLUENTINC
POD

Basic Unit of Deployment in Kubernetes

A collection of containers sharing:

- Namespace
- Network
- Volumes
Persistent Volume (PV) & Persistent Volume Claim (PVC)

- Both PV and PVC are ‘resources’
Persistent Volume (PV) & Persistent Volume Claim (PVC)

- PV is a piece of storage that is provisioned dynamic or static of any individual pod that uses the PV
Persistent Volume (PV) & Persistent Volume Claim (PVC)

- PVC is a request for storage by a User
Persistent Volume (PV) & Persistent Volume Claim (PVC)

- PVCs consume PV
STATEFUL WORKLOADS
STATEFULSET

Rely on Headless Service to provide network identity
Ideal for highly available stateful workloads
STATEFULSET

Rely on **Headless Service** to provide network identity.
STATEFULSET

Ideal for highly available stateful workloads
WORKLOADS DEPLOYMENT
HELM CHARTS

Chart Files (yaml) → Chart Repository → Kubernetes

- Helm Client
- MySQL Chart Files
- Chart (packaged)

Helm Tiller
K8s API Server
Helm Release

API

REST/JSON
gRpc

MySQL Chart

Kubernetes (a running MySQL Pod)

@HELLMAR_BECKER  /  @CONFLUENTINC
HELM CHARTS

Chart Files (yaml)

MySQL
Chart Files

Chart Repository

MySQL Chart (packaged)

Package Chart Files

Helm Client

Helm Tiller

K8s API Server

Helm Release (a running MySQL Pod)

Kubernetes

install/delete release

API

GRpc

REST/JSON

@HELLMAR_BECKER / @CONFLUENTINC
HELM CHARTS

Chart Files (yaml)

Chart Repository

MySQL Chart Files

MySQL Chart (packaged)

Package Chart Files

Helm Client

> Helm Tiller

Kubernetes

install/delete release

APK

Helm Release
(a running MySQL Pod)

K8s API Server

REST/JSON

gRPC
HELM CHARTS

Package Manager

- Package multiple K8s resources into one deployment unit: Chart
KAFKA DEPLOYMENT CHECKLIST

PVC for Storage
StatefulSet for 3-node zk
Optional Pod Anti-Affinity to spread the ZK ensemble across nodes
Headless Service
ConfigMap for Prometheus JMX exporter

Uses ZK Headless Svc
PVC for Storage
StatefulSet for n-node Kafka
A group of NodePort Services for external traffic
ConfigMap for Prometheus JMX exporter
BASIC COMPONENTS ARE NOT ENOUGH
MEET KUBERNETES OPERATOR
KUBERNETES OPERATOR

Embedded with operational knowledge of both data software and Kubernetes

- Backup/restore
- Scale up/down
- Rebalance data
- Regular health checks
Controller

Brain behind Kubernetes resources

- e.g. replication controller, namespace controller etc.
CUSTOM RESOURCE DEFINITION (CRD)

Extend existing Kubernetes API

API: StatefulSet, ReplicaSet, ..., CRD

Controller: StatefulSet Controller, ReplicaSet Controller, ..., Custom Controller

Instance: StatefulSet, ReplicaSet, ..., Custom Resource
Usually works together **Custom Controller**
CUSTOM RESOURCE DEFINITION (CRD)

Users can create and access Customer Resources with **kubectl**, just as they do for built-in resources like **pods**.
Deploy and Manage your production streaming platform with Confluent Operator.

- Automated Provisioning
- Platform Operations
- Resiliency
- Monitoring
Each Confluent Platform component has specific characteristics:
- Security (SSL certificates)
- DNS names and zones
- Host selection
- Fault tolerance
- Scaling
CONFLUENT OPERATOR: AUTOMATED PROVISIONING

```yaml
> cat kafka.yml
## Kafka Cluster
##
kafka:
  name: kafka
  replicas: 3
  resources:
    cpu: 200m
    memory: 1Gi
  external:
    enabled: false
domain: ""
tls:
  enabled: true
domain: devoops.ru
  fullchain: |
  privkey: |
> helm install -f kafka.yml --name kafka
```

- `Load Balancer`
- `Ingress`
- `Kafka Pod`
- `Persistent Volumes`
- `Storage`
Automate scaling:
Spin up new broker pod(s)
Distribute partitions to the new broker(s)
Determine balancing plan
Execute balancing plan
Monitor resources
Automated rolling upgrade with no downtime for Kafka.
Stop broker
Wait for leader election to complete
Start broker with new version
Wait for zero under-replicated-partitions
Repeat
CONFLUENT OPERATOR

Automate provisioning

Monitor SLAs through Confluent Control Center or Prometheus

Scale your Kafkas and CP clusters elastically

Operate at scale with enterprise support from Confluent
ADVANCED USE CASES
DON'T DESPAIR!
As a distributed system for collecting, storing, and processing data at scale, Apache Kafka® comes with its own deployment complexities. Luckily for on-premises scenarios, a myriad of deployment options are available, such as the Confluent Platform which can be deployed on bare metal, virtual machines, containers, etc. But deployment is just the tip of the iceberg.
Is the service offered a managed service for Kafka?

- Key design decisions abstracted from the user
- Requires the user to take the wheel and decide
Why Kafka as a fully-managed service

Deploy in Minutes

Integrate with cloud services

Ensure service levels

Eliminate operational burden
Confluent Cloud | Apache Kafka re-engineered for cloud

DATA INTEGRATION
- Hadoop
- Database
- Data warehouse
- CRM

REAL-TIME APPS
- Transformations
- Custom apps
- Analytics
- Monitoring

HYBRID & MULTI-CLOUD | Replicator
- ENTERPRISE READY | Unlimited scale | Security | Compliance | SLAs
- FULLY-MANAGED | Schema Registry | KSQL | REST Proxy | Connectors*
- CLOUD-NATIVE | Elastic scaling | Usage-based billing

APACHE KAFKA® | Connect | Continuous Commit Log | Streams

*Business support through Confluent Cloud Enterprise subscription for self-managed connectors
## Two ways to stream

<table>
<thead>
<tr>
<th></th>
<th>Confluent Cloud Enterprise</th>
<th>Confluent Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td>24 x 7 Business Support</td>
<td>Optional add-on</td>
</tr>
<tr>
<td>Service Level Agreement</td>
<td>99.95% uptime</td>
<td>99.95% (limited regions)</td>
</tr>
<tr>
<td>Scale</td>
<td>Unlimited throughput</td>
<td>Max 100 MB/s write &amp; read</td>
</tr>
<tr>
<td></td>
<td>Unlimited retention</td>
<td>Max 5TB retention</td>
</tr>
<tr>
<td>Availability</td>
<td>Single AZ, Multi AZ</td>
<td>Single AZ</td>
</tr>
<tr>
<td>Network peering options</td>
<td>VPC Peering, Transit Gateway (AWS only)</td>
<td>Not available</td>
</tr>
</tbody>
</table>

Cloud-native experience for Apache Kafka
Confluent Cloud
Cloud-native experience for Apache Kafka

Cloud native agility
Scale elastically from 0 to 100 MB/s
No need to size or provision clusters

Consumption-based pricing
Only pay for what you actually stream
No minimums. No upfront commitments.

Complete event streaming service
FULLY-MANAGED:
Schema Registry
S3 Sink Connector
KSQL (in preview)

Scale to 10s of GB/s with our Enterprise plan
## Two ways to stream

<table>
<thead>
<tr>
<th>Confluent Cloud Enterprise</th>
<th>Confluent Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built for mission-critical applications</strong></td>
<td><strong>Cloud-native experience for Apache Kafka</strong></td>
</tr>
<tr>
<td><strong>Support</strong></td>
<td><strong>Optional add-on</strong></td>
</tr>
<tr>
<td>24 x 7 Business Support</td>
<td>99.95% (limited regions)</td>
</tr>
<tr>
<td><strong>Service Level Agreement</strong></td>
<td><strong>Max 100 MB/s write &amp; read</strong></td>
</tr>
<tr>
<td>99.95% uptime</td>
<td>Max 5TB retention</td>
</tr>
<tr>
<td><strong>Scale</strong></td>
<td><strong>Single AZ</strong></td>
</tr>
<tr>
<td>Unlimited throughput</td>
<td>Not available</td>
</tr>
<tr>
<td>Unlimited retention</td>
<td></td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td></td>
</tr>
<tr>
<td>Single AZ, Multi AZ</td>
<td></td>
</tr>
<tr>
<td><strong>VPC peering</strong></td>
<td></td>
</tr>
<tr>
<td>VPC Peering</td>
<td></td>
</tr>
<tr>
<td>Transit Gateway (AWS only)</td>
<td></td>
</tr>
</tbody>
</table>
### Environment overview

<table>
<thead>
<tr>
<th>CLUSTERS</th>
<th>SCHEMA REGISTRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster-name_01</td>
<td>Status: Running</td>
</tr>
<tr>
<td>Cloud Provider: Amazon AWS</td>
<td>Region: US-West1 (Oregon)</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td><strong>Write</strong></td>
</tr>
<tr>
<td>Provisioned: 300 MB/s</td>
<td>Provisioned: 1 MB/s</td>
</tr>
<tr>
<td>Peak usage: 200 MB/s</td>
<td>Peak usage: 0.9 MB/s</td>
</tr>
<tr>
<td>Average: 123 MB/s</td>
<td>Average: 0.45 MB/s</td>
</tr>
<tr>
<td>cluster-name_02</td>
<td>Status: Running</td>
</tr>
<tr>
<td>Cloud Provider: Amazon AWS</td>
<td>Region: US-West1 (Oregon)</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td><strong>Write</strong></td>
</tr>
<tr>
<td>Provisioned: 300 MB/s</td>
<td>Provisioned: 1 MB/s</td>
</tr>
<tr>
<td>Peak usage: 202.12 MB/s</td>
<td>Peak usage: 0.868 MB/s</td>
</tr>
<tr>
<td>Average: 2.091 MB/s</td>
<td>Average: 0.751 MB/s</td>
</tr>
</tbody>
</table>

### Confluent Cloud

**Enterprise**

**Reliability at scale**

<table>
<thead>
<tr>
<th>Scale</th>
<th>Unlimited throughput, unlimited retention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>99.95% uptime SLA</td>
</tr>
<tr>
<td>Durability</td>
<td>Multi-AZ with 3 availability zones (option)</td>
</tr>
<tr>
<td>Connection</td>
<td>VPC peering, AWS Transit Gateway (options)</td>
</tr>
<tr>
<td>Support</td>
<td>24x7 Business, option to upgrade to Premier</td>
</tr>
<tr>
<td>Terms</td>
<td>1 year commitment; flexible payment options</td>
</tr>
<tr>
<td>Cloud</td>
<td>GCP, Azure, AWS</td>
</tr>
</tbody>
</table>
Confluent Cloud | Enterprise grade security

01 Control access to cluster and resources
- SASL authentication
- VPC peering / AWS Transit
- ACLs for granular control

02 Protect users, applications and access to data
- SOC 1, SOC 2, SOC 3
- ISO 27001, PCI Level 2
- HIPAA & GDPR ready

03 Secure data against unwanted access (Encryption)
- Data at rest encryption
- Over the wire encryption (TLS Support)
Confluent Cloud | High-performance at GBps scale

- Sub-25 ms latencies* at massive scale
- Unlimited throughput and fanout
- Infinite retention

Enterprise

![Graphs showing performance metrics](image)

- Current Total Bytes In per Second: 1041.84 MB/s
- Current Total Bytes Out per Second: 1928.73 MB/s
- Current Total Messages In per Second: 2M M/s
Confluent Cloud | High availability for mission critical apps

- 3x replication
- Single & multi AZ
- 99.95% uptime SLA
Confluent Cloud | More than just Kafka

FULLY-MANAGED COMPONENTS

<table>
<thead>
<tr>
<th>Schema Registry</th>
<th>KSQL*</th>
<th>Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make data backwards compatible and future-proof</td>
<td>Develop real-time stream processing apps writing only SQL</td>
<td>Easily send data to cloud storage with AWS S3 Connector (sink) + more coming soon*</td>
</tr>
</tbody>
</table>

*Currently offered in preview with no support. Purchased support for CCE will include support for fully managed confluent components that are PA/GA
THANKS!

@Hellmar_Becker
hellmar@confluent.io