Operating Multi-Tenant Kafka Services for Developers

Data Council SF 2019

Ali Hamidi - Heroku Data

HEROKU

Agenda

- Intro
- Motivation
- Single Tenant Dedicated
- Multi-tenancy
- Configuration & Tuning
- Testing
- Automation
- Limitations

Intro

I am… Ali Hamidi, an engineer on the Heroku Data team at Salesforce.

Heroku is... a cloud platform that lets companies build, deliver, monitor and scale apps.

Heroku Data is… the team that provides secure, scalable data services on the Heroku Platform.

Apache Kafka

- Distributed Streaming Platform

Apache Kafka

- Distributed Streaming Platform
- Publish/Subscribe (=> Produce/Consume)

Apache Kafka

- Distributed Streaming Platform
- Publish/Subscribe (=> Produce/Consume)
- Durable message store (commit log)

Apache Kafka

- Distributed Streaming Platform
- Publish/Subscribe (=> Produce/Consume)
- Durable message store (commit log)
- Highly available

Apache Kafka on Heroku

- Fully Managed Service

Apache Kafka on Heroku

- Fully Managed Service
- Opinionated

Apache Kafka on Heroku

- Fully Managed Service
- Opinionated
- Configured for best practices for most users*

Use Cases

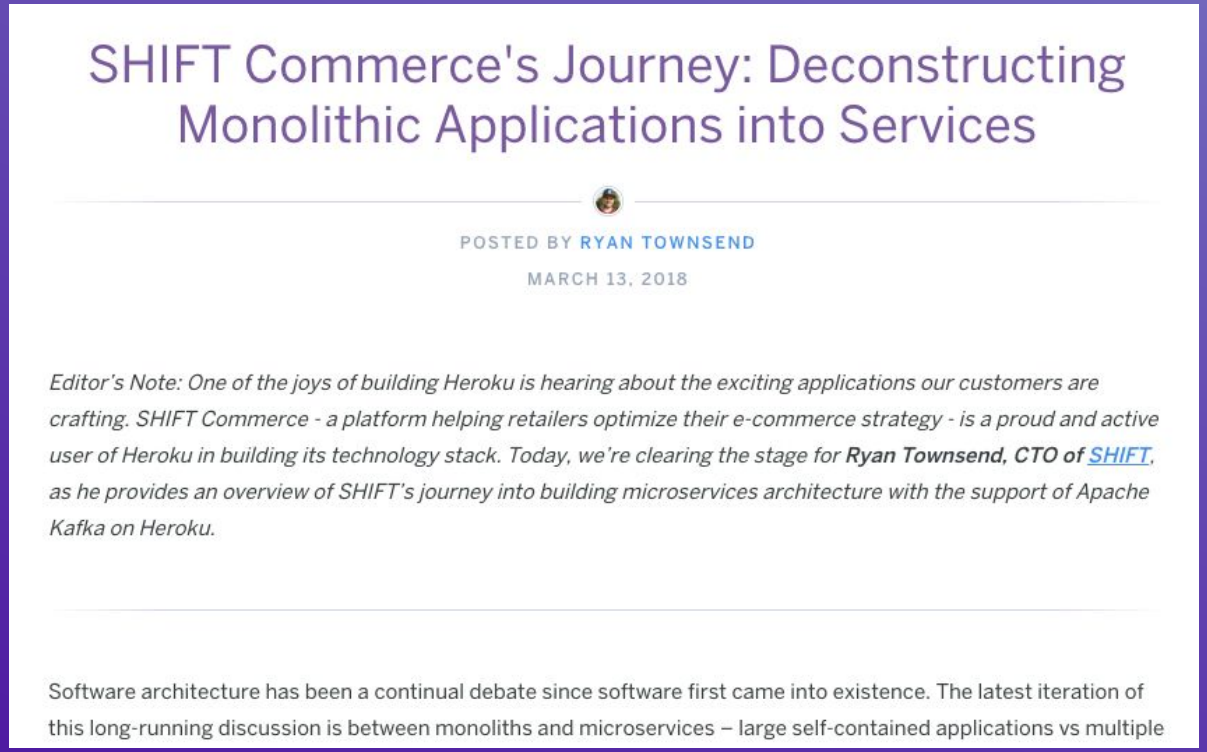
- Decompose a monolithic app

Use Cases

- Decompose a monolithic app
- Process high volume, real-time data streams

Use Cases

- Decompose a monolithic app
- Process high volume, real-time data streams
- Power a real-time, event-driven architecture



SHIFT Commerce's Journey: Deconstructing Monolithic Applications into Services

POSTED BY RYAN TOWNSEND

MARCH 13, 2018

Editor's Note: One of the joys of building Heroku is hearing about the exciting applications our customers are crafting. SHIFT Commerce - a platform helping retailers optimize their e-commerce strategy - is a proud and active user of Heroku in building its technology stack. Today, we're clearing the stage for ***Ryan Townsend, CTO of SHIFT****, as he provides an overview of SHIFT's journey into building microservices architecture with the support of Apache Kafka on Heroku.*

Software architecture has been a continual debate since software first came into existence. The latest iteration of this long-running discussion is between monoliths and microservices – large self-contained applications vs multiple

SHIFT Commerce

Decompose a monolithic app

Quoine

- QUOINE is a leading global fintech company that provides trading, exchange, and next generation financial services powered by blockchain technology
- Consume real-time cryptocurrency pricing data from individual markets and exchanges

Caesars Entertainment

- Ingest, aggregate, and process customer data in real-time to provide the best customer experience
- Real-time, event-driven architecture

The Motivation

Why Multi-tenant Kafka?

- More accessible
- Additional use cases
  - Development
  - Testing
  - Low volume production

Kafka Everywhere: New Plans and Pricing for Apache Kafka on Heroku

POSTED BY RAND FITZPATRICK

SEPTEMBER 14, 2017

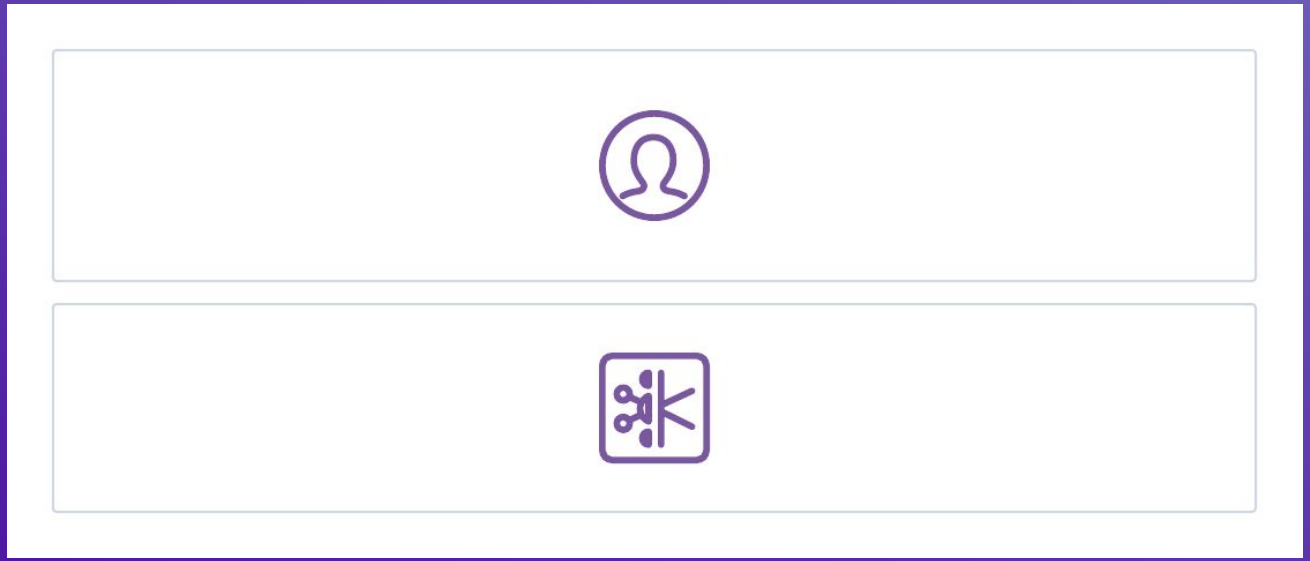
Event-driven architectures are on the rise, in response to fast-moving data and constellations of inter-connected systems. In order to support this trend, last year we released Apache Kafka on Heroku - a gracefully integrated, fully managed, and carefully optimized element of Heroku's platform that is the culmination of years of experience of running many hundreds of Kafka clusters in production and contributing code to the Kafka ecosystem.

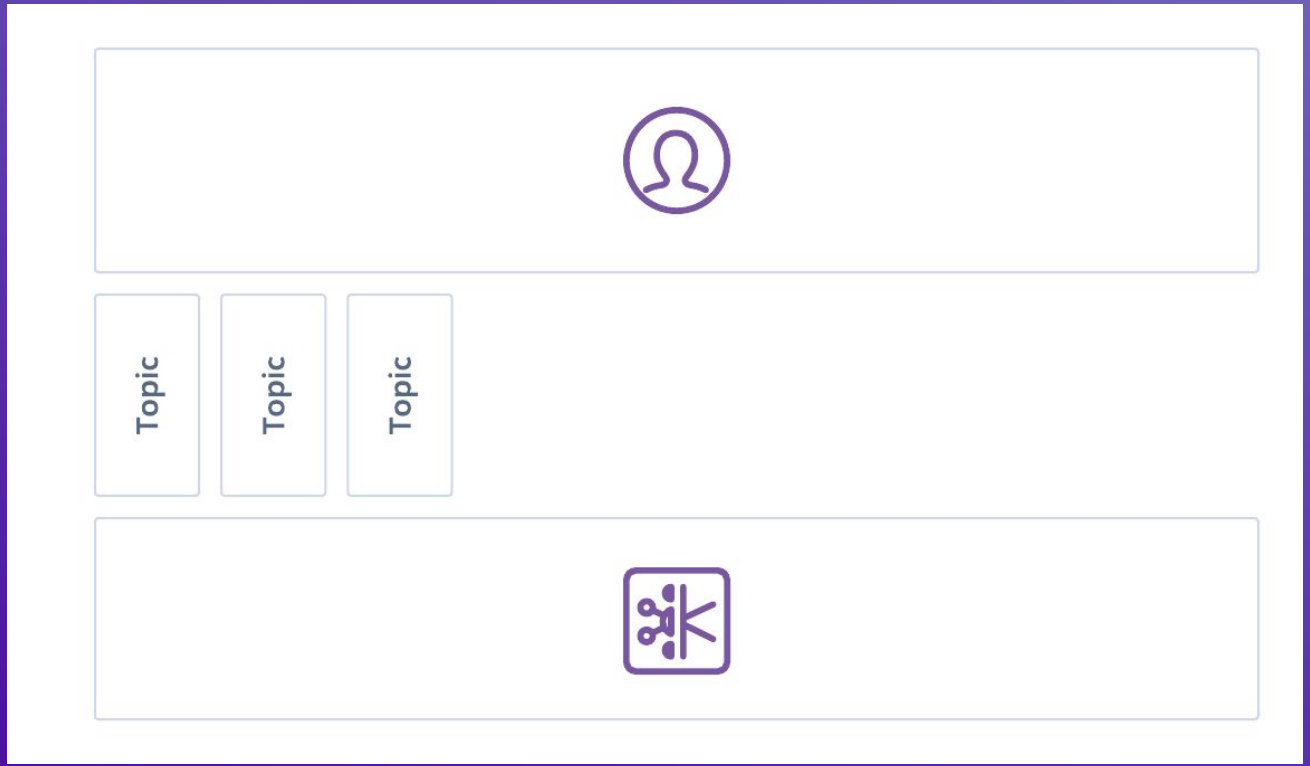
Today, we are excited to announce additional plans and pricing in our Kafka offering in order to make Apache Kafka more accessible, and to better support development, testing, and low volume production needs.

Apache Kafka on Heroku: Now With More Flexibility and Speed

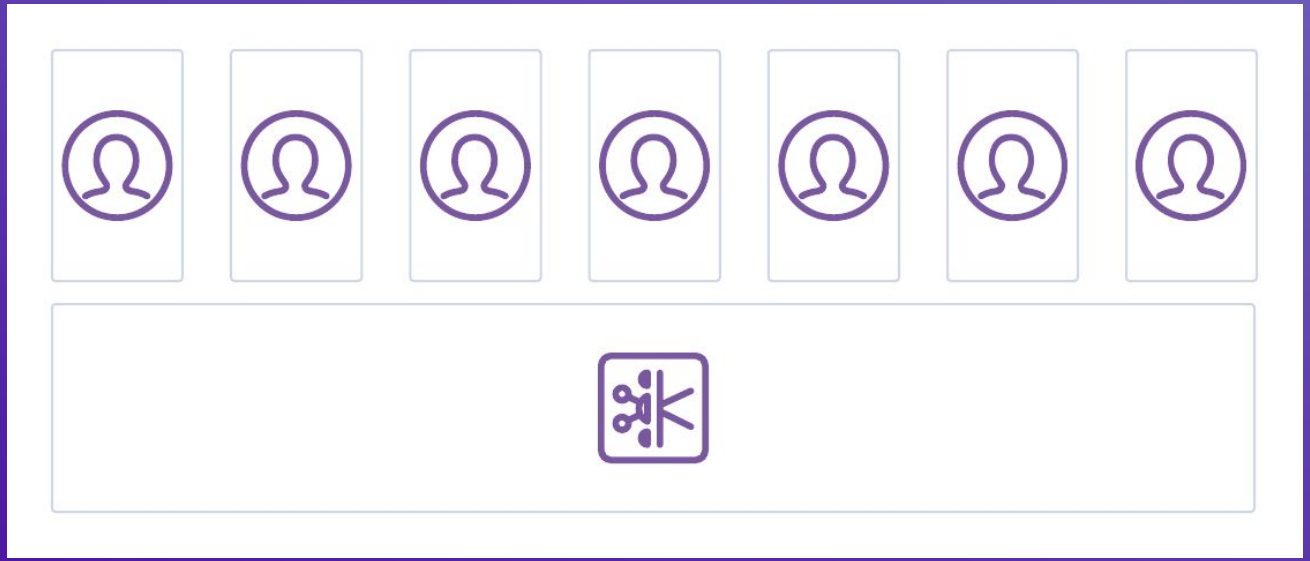
Apache Kafka is a powerful, distributed streaming platform, and the dominant open source solution in managing high scale event streams. Kafka enables you to easily design and implement architectures for many important use cases, such as elastic queuing, data pipelines & analytics, and microservices coordination. Apache Kafka on Heroku removes the complexity and cost of running Kafka, making its valuable resources available to a broad range of developers and applications.

Single Tenant Dedicated




Topic
Topic
Topic

Multi-tenancy



Multi-tenancy

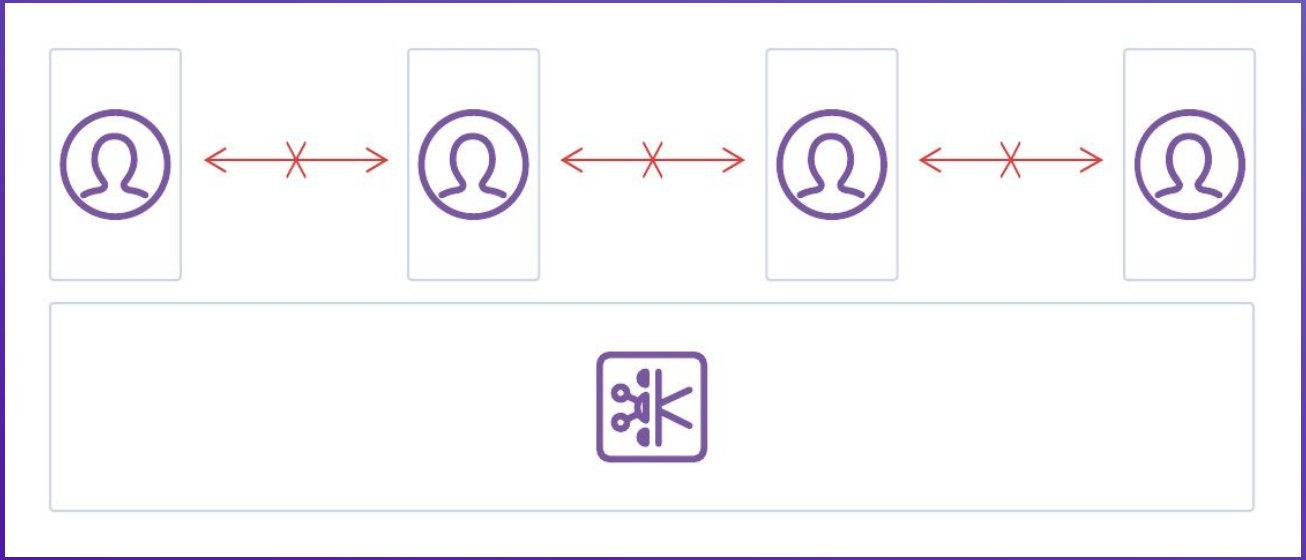
- Resource isolation
  - Security
  - Performance
  - Safety
- Parity
  - Feature
  - Behaviour
- Compatibility
- Costs
  - Resources
  - Operational

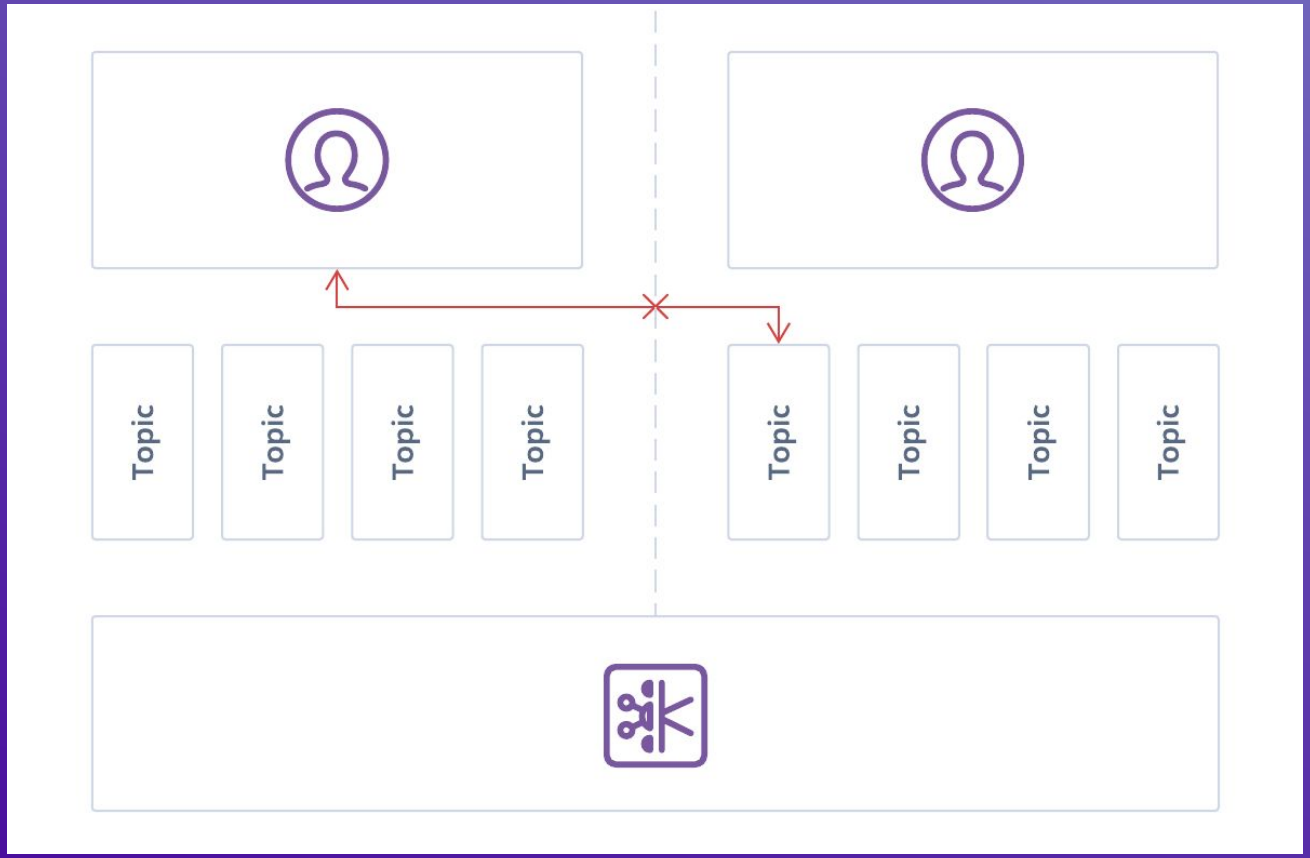
Multi-tenancy

- Resource isolation
  - Security
  - Performance
  - Safety
- Parity
  - Feature
  - Behaviour
- Compatibility
- Costs
  - Resources
  - Operational

Security

A tenant should not be able to access another tenant's data




Topic
Topic
Topic
Topic
Topic
Topic
Topic
Topic

Security

- Access Control Lists (ACLs)
- Namespacing

Security

- Access Control Lists (ACLs)
  - User A can carry out action B on resource C
- Namespacing

Security

- Access Control Lists (ACLs)
  - User A can carry out action B on resource C
- Namespacing
  - wabash-58779.events

Performance

A tenant should not adversely affect another tenant's performance

Performance

- Quotas
  - Produce
  - Consume

Safety

A tenant should not jeopardise the stability of the cluster

Safety

- Limits
  - Topics
  - Partitions
  - Consumer Groups
  - Storage
  - Throughput

Capacity = Message Throughput * Retention * Replication

Safety

- Limits
  - Topics
  - Partitions
  - Consumer Groups
  - ~~Storage~~ Capacity
  - Throughput

Safety

- Limits
  - Topics
  - Partitions
  - Consumer Groups
  - ~~Storage~~ Capacity
  - Throughput
- Monitoring

Safety

- Limits
  - Topics
  - Partitions
  - Consumer Groups
  - ~~Storage~~ Capacity
  - Throughput
- Monitoring
- Limit enforcement!

Multi-tenancy

- Resource isolation
  - Security
  - Performance
  - Safety
- Parity
  - Feature
  - Behaviour
- Compatibility
- Costs
  - Resources
  - Operational

Parity

For the service to be useful, it needs to behave like a normal cluster

Parity

- Access to a standard cluster

Parity

- Access to a standard cluster
- ...but with some limitations

Multi-tenancy

- Resource isolation
  - Security
  - Performance
  - Safety
- Parity
  - Feature
  - Behaviour
- Compatibility
- Costs
  - Resources
  - Operational

Compatibility

The service needs to support standard clients

No vendor lock-in

Compatibility

- Open Source Apache Kafka
- Not a fork
- No custom code required
- Use standard clients

Multi-tenancy

- Resource isolation
  - Security
  - Performance
  - Safety
- Parity
  - Feature
  - Behaviour
- Compatibility
- Costs
  - Resources
  - Operational

Costs

The service needs to be financially feasible

Resource Costs

- Packing Density
- Utilization

Resource Costs

- Cluster size?
- No over provisioning
- Seamless upgrading
  - Can’t move tenants (can’t migrate message offsets)

Operational Costs

- Minimal operational burden
- Minimize impact/blast radius

Operational Costs

- Safe defaults
- Similar clusters to our dedicated
- Automation (kind of our thing)
- Testing (lots)

Configuration & Tuning

Configuration & Tuning

- Partitions
- Quotas
- Topics & Consumer Groups
- Guard Rails

Partitions

- Lots of partitions
  - 48,000
- Max file descriptors
  - 500,000
- Max mmap count
  - 500,000

Quotas

- Per Broker!
- Counter intuitive enforcement

Topics & Consumer Groups

- Explicit Topic creation
- Explicit Consumer Group creation

Guard Rails

- Limit potential bad usage

Guard Rails

- Limit potential bad usage
- “Customers don’t make mistakes, we make bad tools”

```
# Heroku Data Control Plane
min_retention_time = 24.hours
```

```
# Heroku Data Control Plane
min_retention_time = 24.hours
max_retention_time = 7.days
```

```
# Heroku Data Control Plane
min_retention_time = 24.hours
max_retention_time = 7.days
default_replication_factor = 3
```

```
# Heroku Data Control Plane
min_retention_time = 24.hours
max_retention_time = 7.days
default_replication_factor = 3
min_replication_factor = 3
```

```
# Heroku Data Control Plane
min_retention_time = 24.hours
max_retention_time = 7.days
default_replication_factor = 3
min_replication_factor = 3
max_replication_factor = 3
```

```
# Heroku Data Control Plane
min_retention_time = 24.hours
max_retention_time = 7.days
default_replication_factor = 3
min_replication_factor = 3
max_replication_factor = 3


# Kafka
num.partitions=8
```

```
# Heroku Data Control Plane
min_retention_time = 24.hours
max_retention_time = 7.days
default_replication_factor = 3
min_replication_factor = 3
max_replication_factor = 3


# Kafka
num.partitions=8
replication=3
```

```
# Heroku Data Control Plane
min_retention_time = 24.hours
max_retention_time = 7.days
default_replication_factor = 3
min_replication_factor = 3
max_replication_factor = 3


# Kafka
num.partitions=8
replication=3
min.insync.replicas=2
```

Monitoring

- Custom tool
  - Agentless
- JMX is great!
- Jolokia + JMX is great(er)!

Testing

- How many partitions?
- What kind of throughput?
- Common operations?
- Failure scenarios?
- Max packing density?
- Bugs?

Testing - Automation!

- Internal tool
  - Leverages internal Heroku Data API
  - Creates large number of “tenants”
  - Creates topics and consumer groups for each tenant
  - Spins up pool of producers and consumers for each
  - Simulates different types of users

Bugs :(

- KAFKA-4725
- Contributed PR
- Fixed in 0.10.1.1

Automation

Heroku Data Control Plane

- Kafka does a lot

Heroku Data Control Plane

- Kafka does a lot...but it doesn't do everything!

Automate everything*

Automation

- Cluster resizing
- Node replacement
- Storage expansion
- Version upgrades
- Restarts*

Automation

- Cluster resizing
- Node replacement
- Storage expansion
- Version upgrades
- Restarts*
- ...

Limitations

- No AdminAPI

Limitations

- No AdminAPI
- Explicit Topic creation

Limitations

- No AdminAPI
- Explicit Topic creation
- Explicit Consumer Group creation

Limitations

- No AdminAPI
- Explicit Topic creation
- Explicit Consumer Group creation
- No Zookeeper access

Thank you!

Ali Hamidi @ahamidi

HEROKU.COM