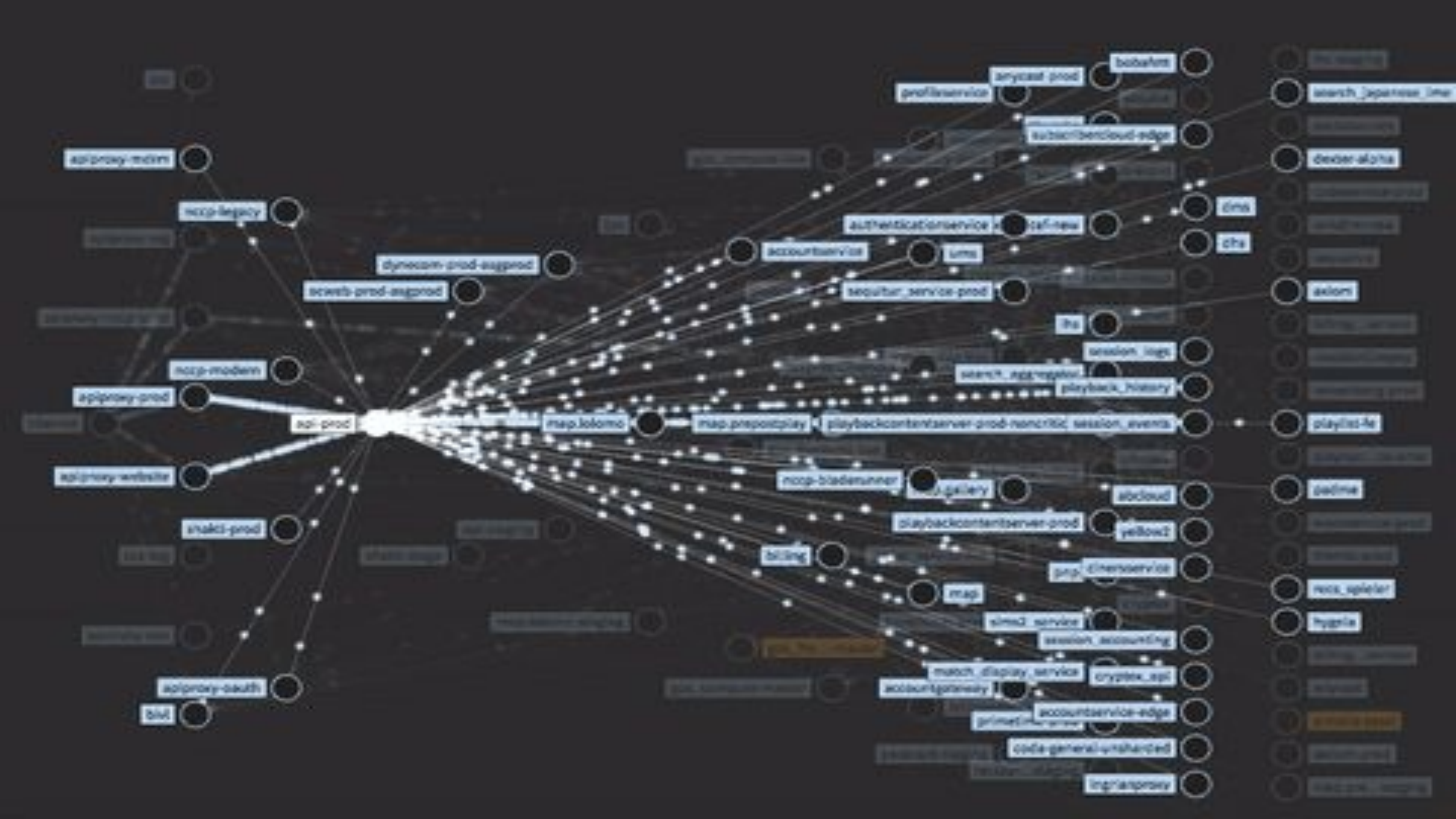
Cloud-Native and Scalable Kafka

Allen Wang

@allenxwang

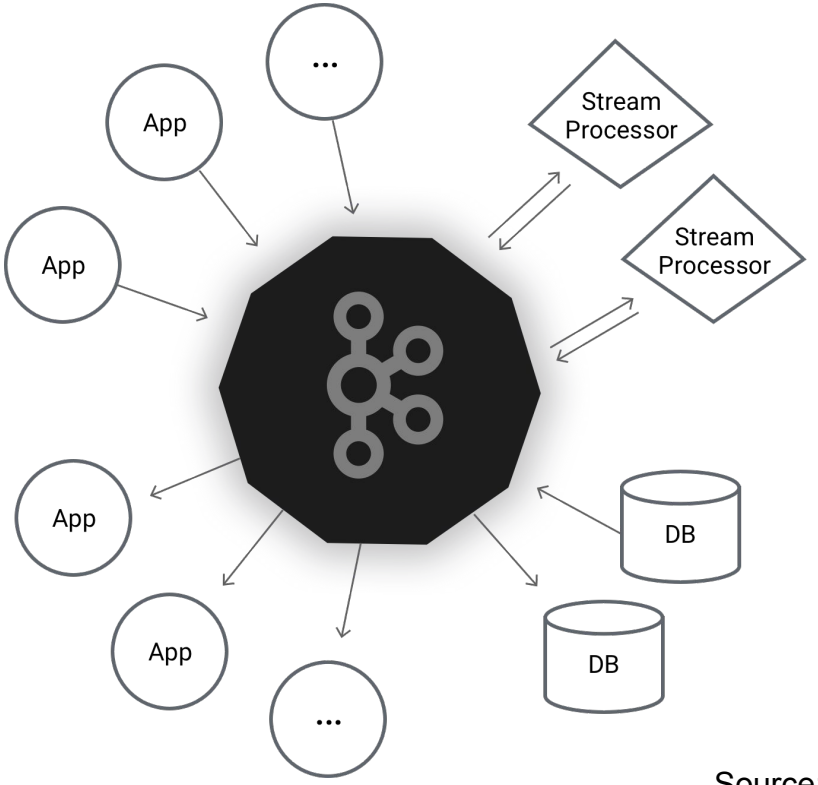
About Me

- Real Time Data Infrastructure @ Netflix
- Apache Kafka contributor (KIP-36 Rack Aware Assignment)
- NetflixOSS contributor (Archaius and Ribbon)
- Previously
  - Cloud platform @ Netflix
  - VeriSign, Sun Microsystems



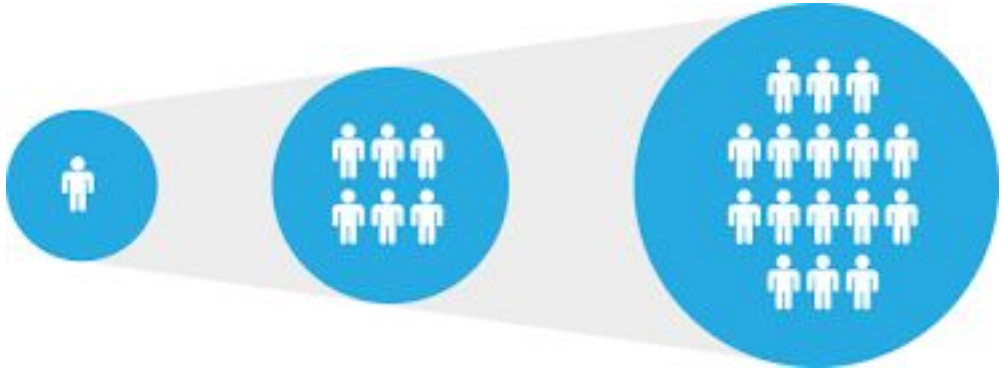
api-prod
apiproxy-prod
apiproxy-website
apiproxy-oauth
nccp-legacy
nccp-modern
nccp-bladerunner
dynecom-prod-asgprod
accountservice
anycast-prod
profileservice
subscribercloud-edge
dexter-alpha
cms
chs
authenticationservice
ums
sequitur_service-prod
axiom
ihs
session_logs
search_aggregator
playback_history
map.lolomo
map.prepostplay
playlist-fe
padme
abcloud
playbackcontentserver-prod
billing
cinemaservice
map
recs_spieler
hygeia
sims2_service
session_accounting
cryptex_api
accountgateway
accountservice-edge
coda-general-unsharded
ingrianproxy

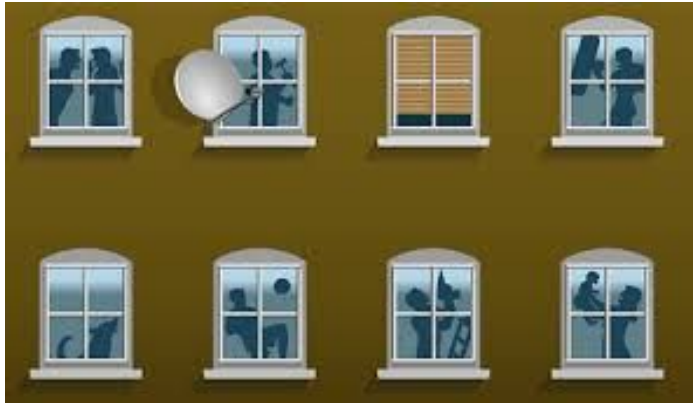
They All Come To One Place



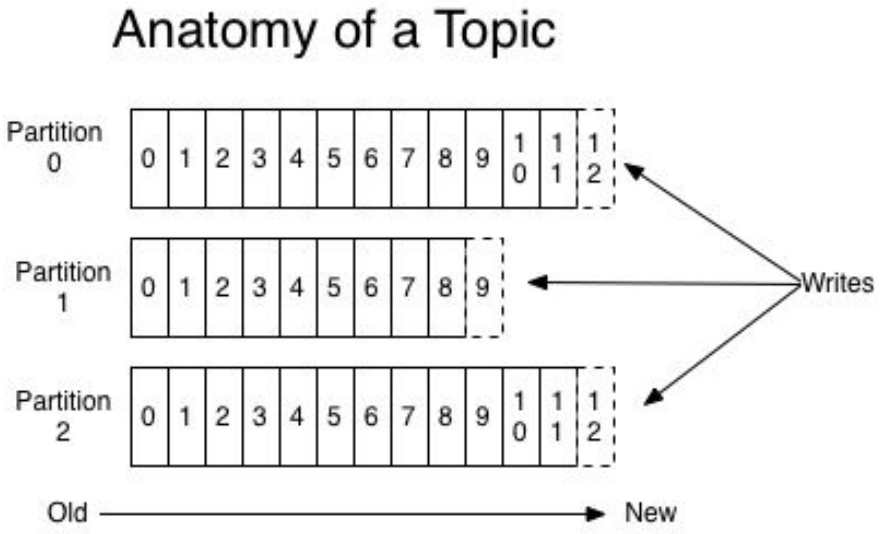
Source: http://kafka.apache.org

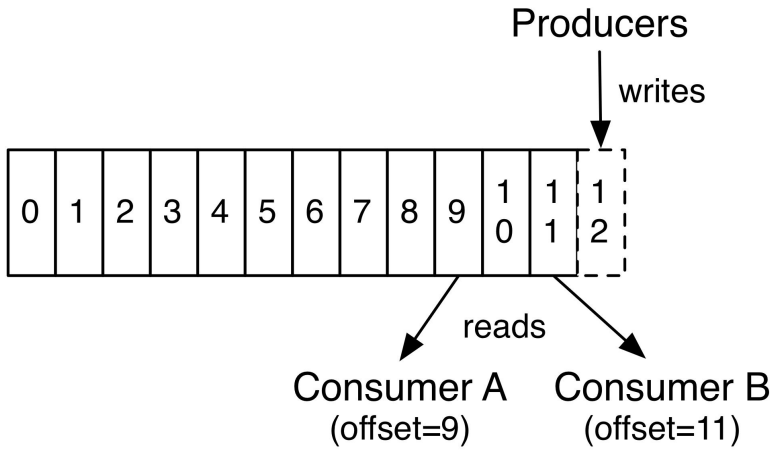
What's In the Talk





Kafka - Distributed Streaming Platform





Source: http://kafka.apache.org

Kafka @ Netflix

- Data Pipeline and stream processing
  - Business and analytical data
  - System related
- Huge volume but non-transactional data
- Order is not required for most of topics

Kafka @ Netflix Scale

- 4,000+ brokers and ~50 clusters in 3 AWS regions
- > 1 Trillion messages per day
- At peak (New Years Day 2018)
  - 2.2 trillion messages (1.3 trillion unique)
  - 6 Petabytes

A Typical Netflix Kafka Cluster

- 20 to 200 brokers
- 4 to 8 cores, Gbps network, 2 to 12 TB local disk
- Brokers on Kafka 0.10.2
- Span across three availability zones within a region with rack aware assignment
- MirrorMaker for cross region replication for selected topics

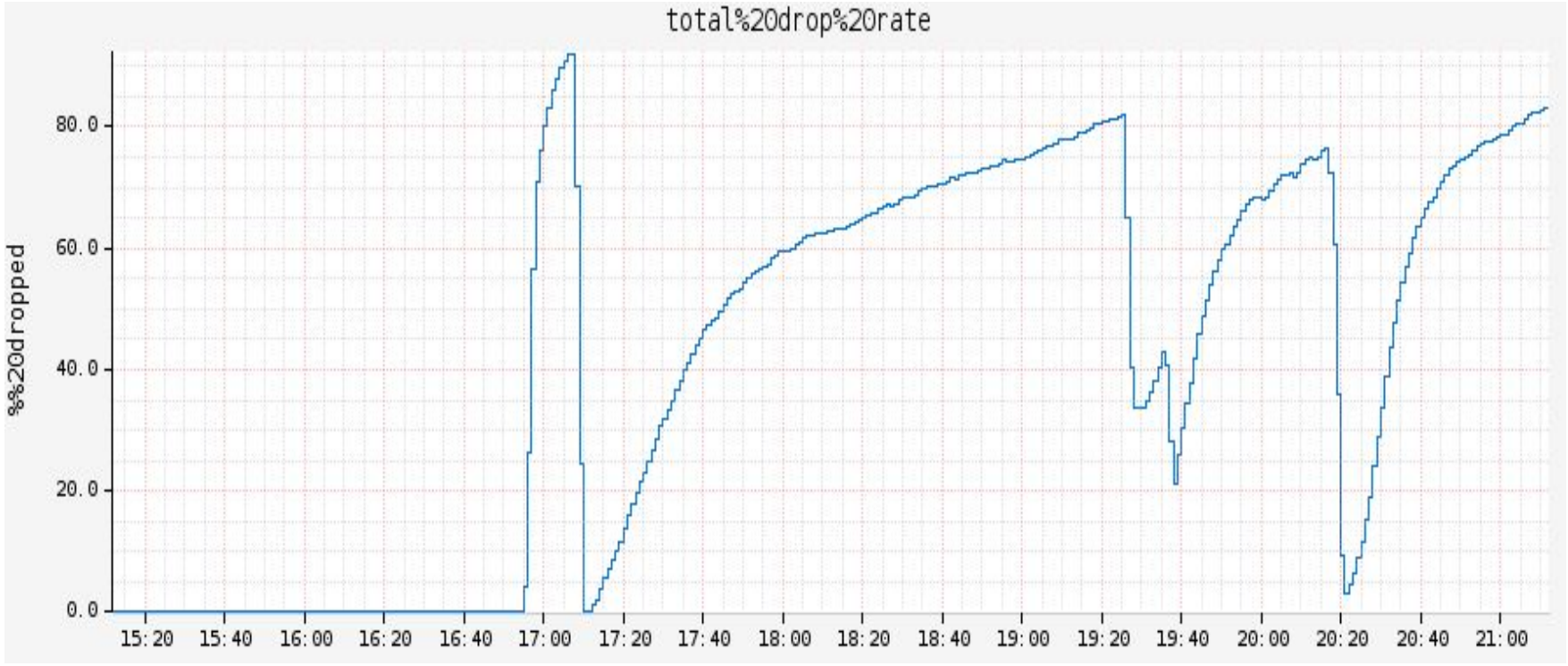
Challenges

Availability

Availability Defined

- Ratio of messages successfully produced to Kafka vs. total attempts

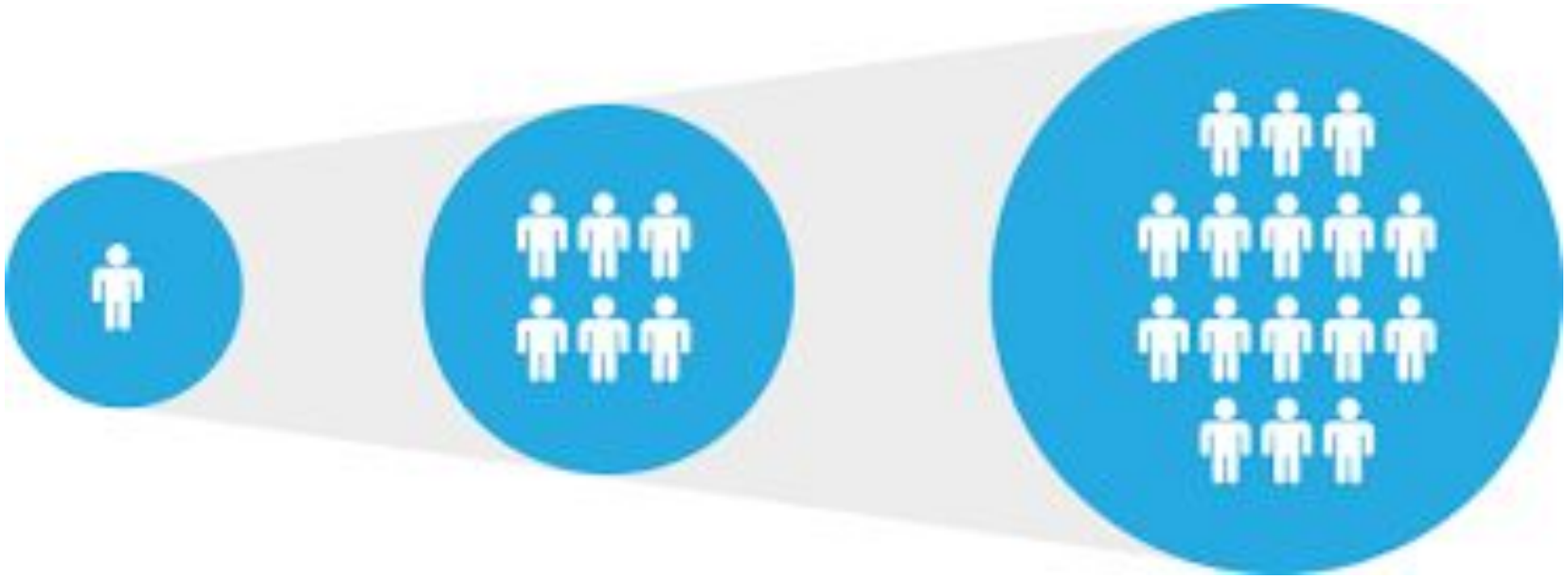
Availability Challenge



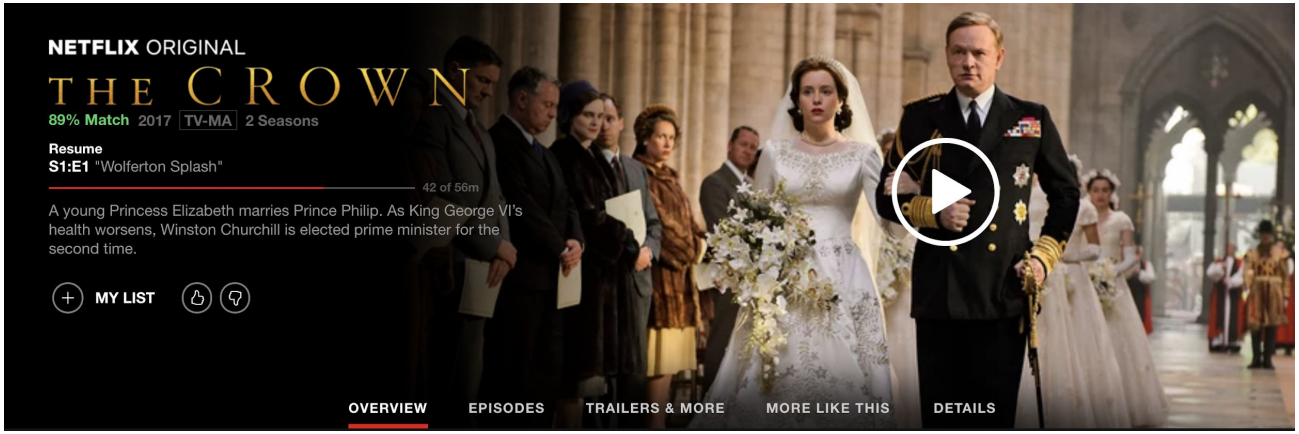
Availability Challenge

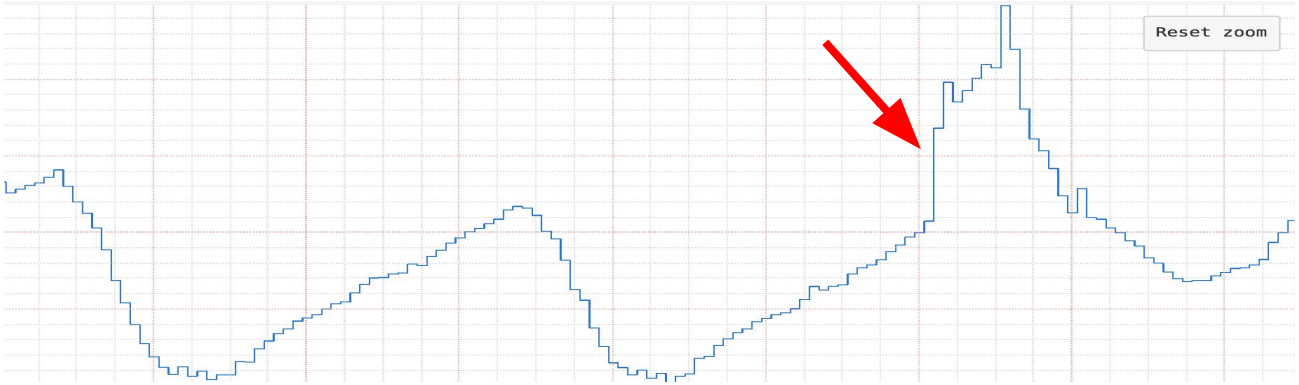
- We have improved
  - Over 99.999% availability
- Failover is must to have

Scalability

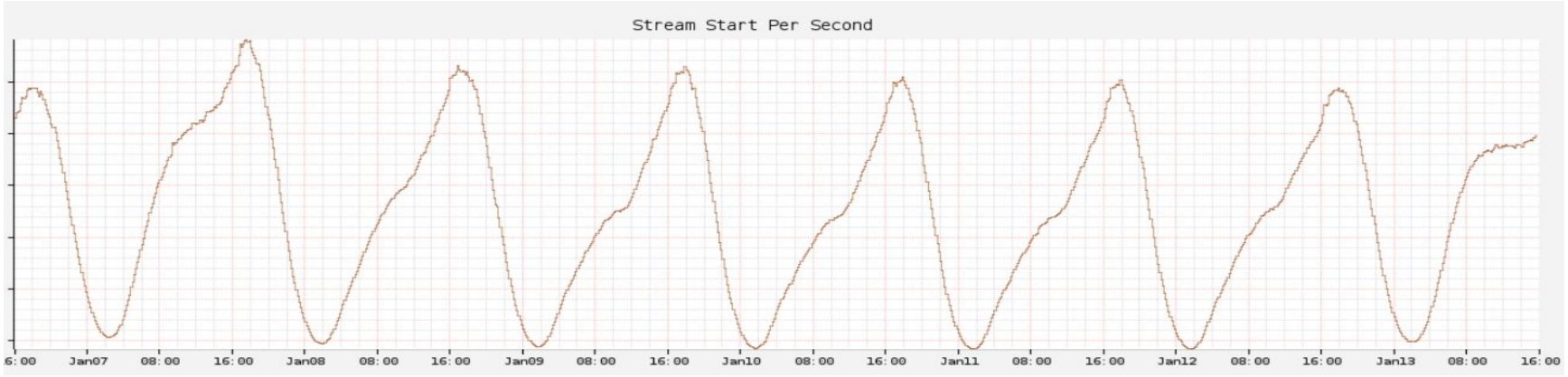


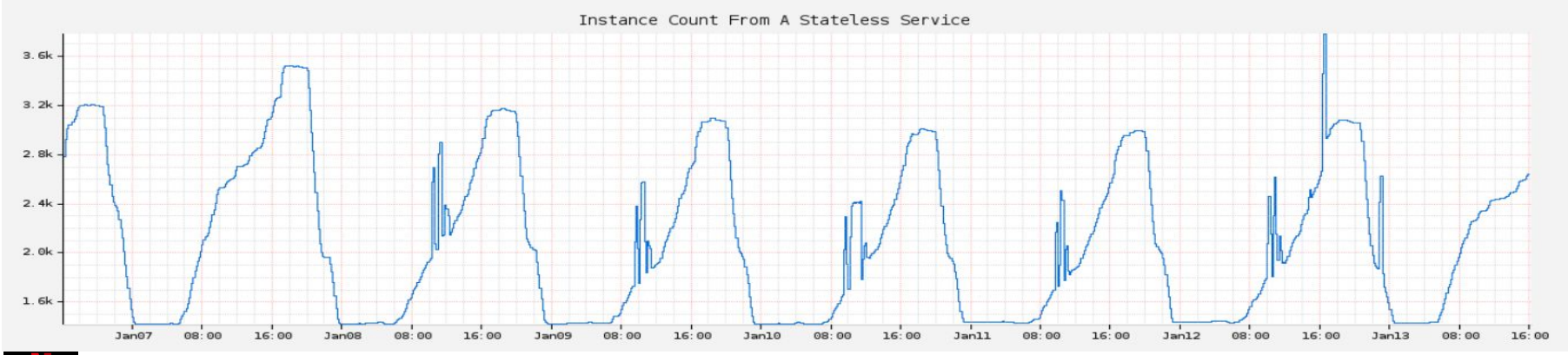
Scalability Challenge

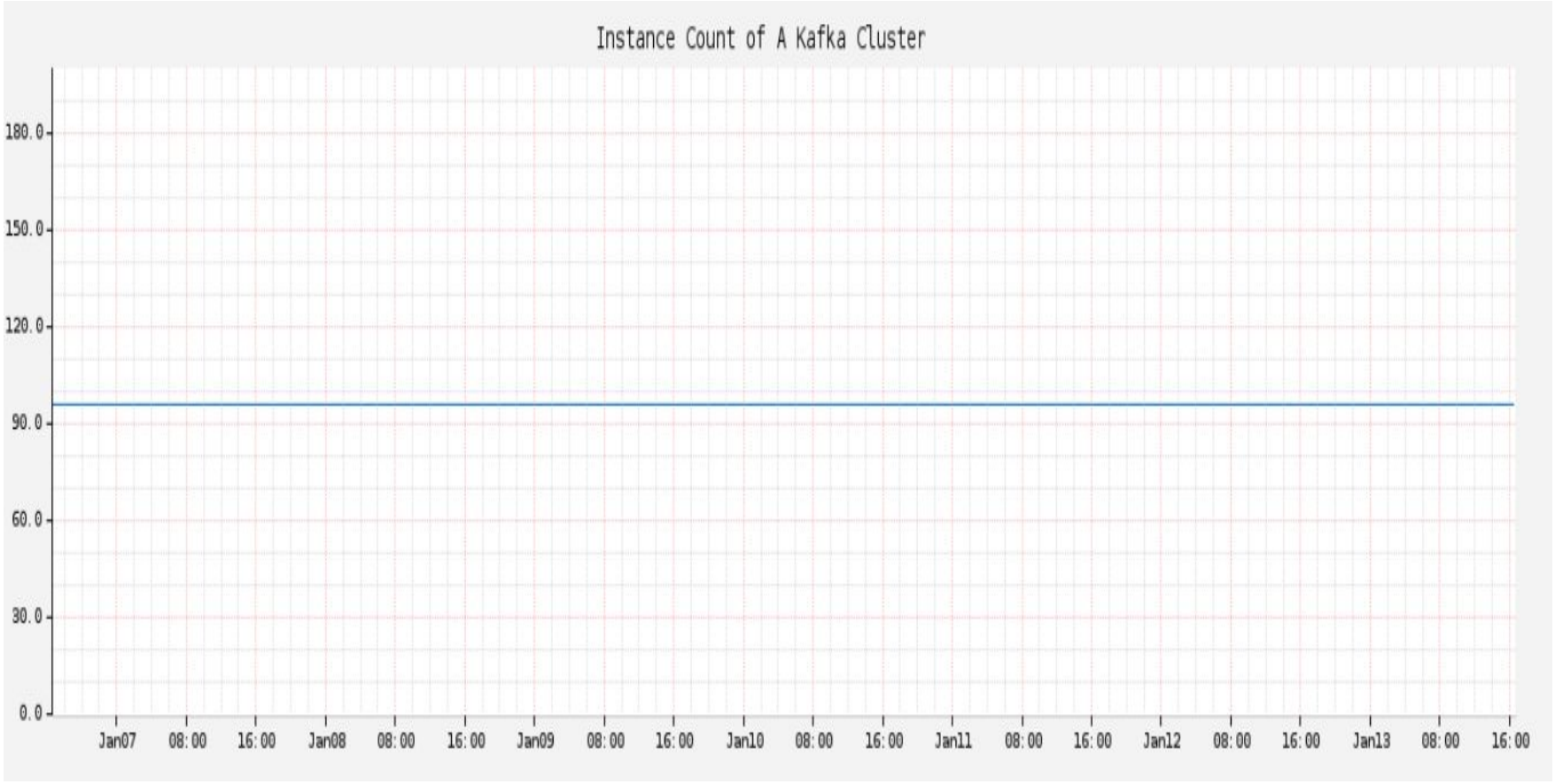




Desired Autoscale



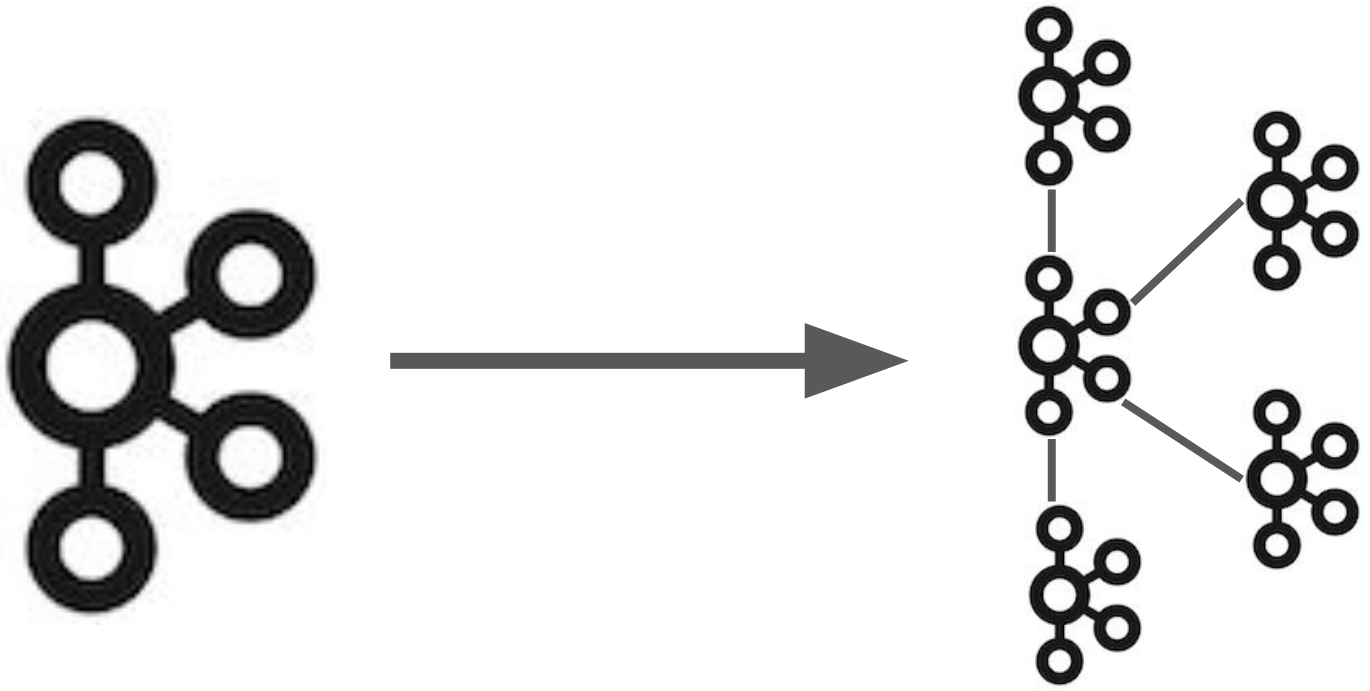



Instance Count of A Kafka Cluster
180.0
150.0
120.0
90.0
60.0
30.0
0.0
Jan07
08:00
16:00
Jan08
08:00
16:00
Jan09
08:00
16:00
Jan10
08:00
16:00
Jan11
08:00
16:00
Jan12
08:00
16:00
Jan13
08:00
16:00

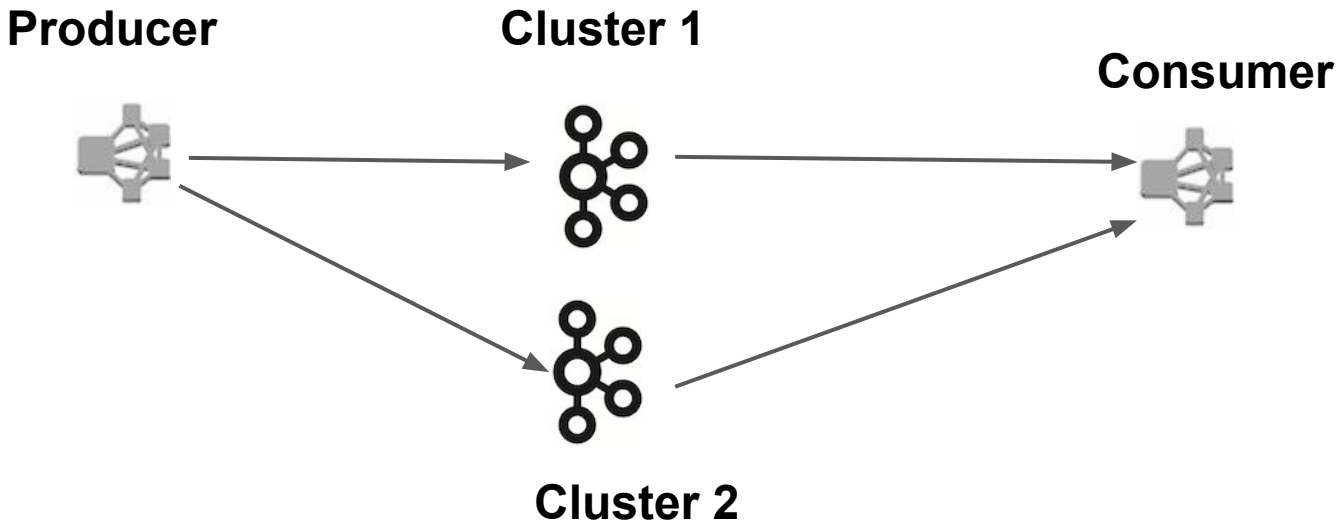
Why Scaling is Difficult

- Add brokers and partitions
  - Currently does not work well with keyed messages
  - Practical limit of number of partitions
  - Watch for KIP-253: In order message delivery with partition expansion and deletion
- Partition reassignment
  - Data copying is time consuming
  - Increased network traffic

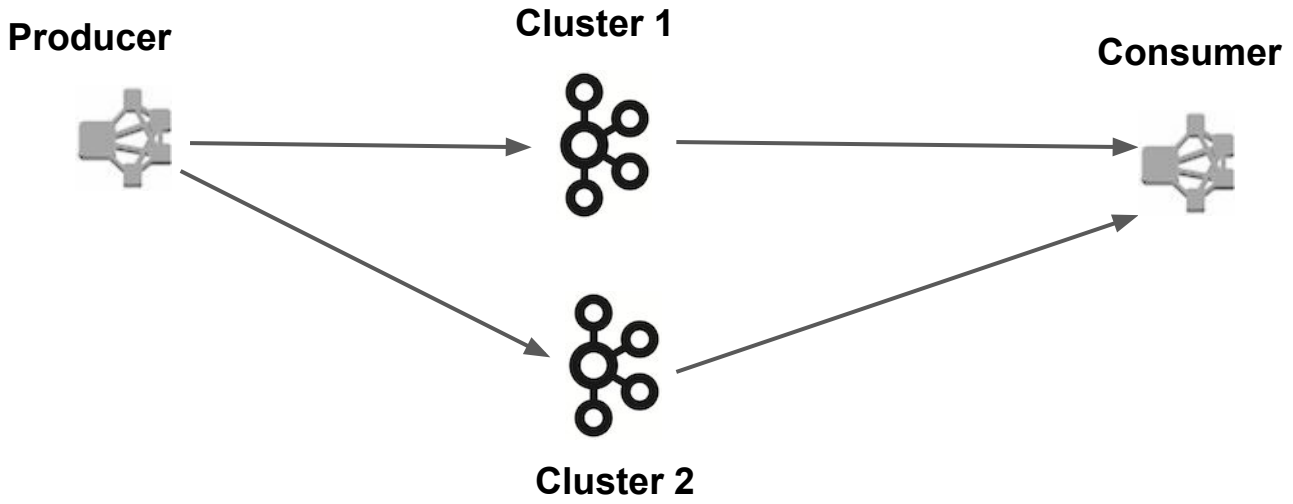
Think Out Of the Box



Scale with Traffic



Topic Move/Failover



Failover with Traffic Migration

- Netflix operates in island model
- In region Kafka failover
  - Failover by switching client traffic to a different cluster
  - No extra cost for redundancy or cross DC traffic
  - No ordering guarantee
  - Best case: exactly once
  - Worst case: data loss

Better Scalability with Multi-Cluster

- No data copying!
- Built-in failover capability
- Requires built-in client support to switch traffic
  - Currently implemented with client dynamic properties
- Does not work with keyed messages - still WIP

Improvement on Availability

Let’s Prove It

- Divide one big cluster into **s** clusters
- Assumptions
  - Replication factor **k** in both cases
  - losing k brokers always lead to unavailability
- Small clusters can be $\mathbf{s^{k-1}}$ times more reliable than one big cluster

The Math

Compare number of combinations to choose **k** brokers from a cluster of size **n** vs. from any one of **s** clusters of size **m**

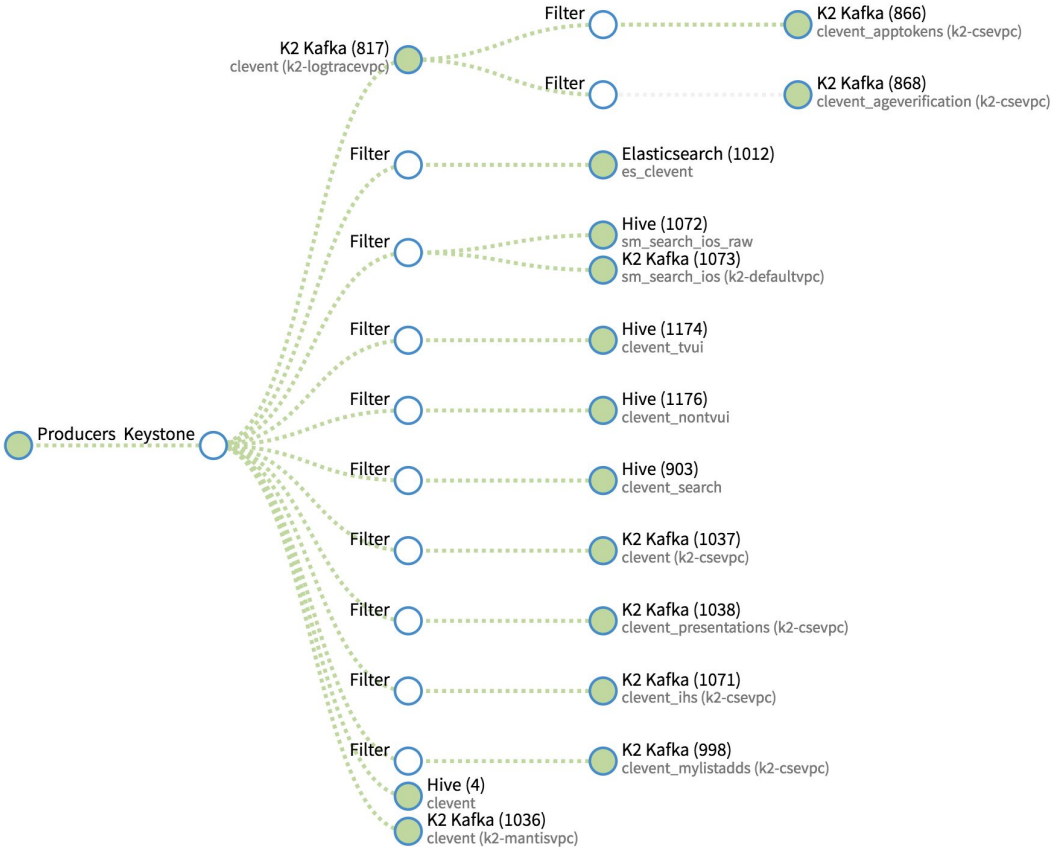
$$F_n = C_n^k = \frac{n\,(n-1)(n-2)\,\ldots\,(n-k+1)}{k!}$$

$$F_m = sC_m^k = s\,\frac{m\,(m-1)(m-2)\ldots(m-k+1)}{k!}$$

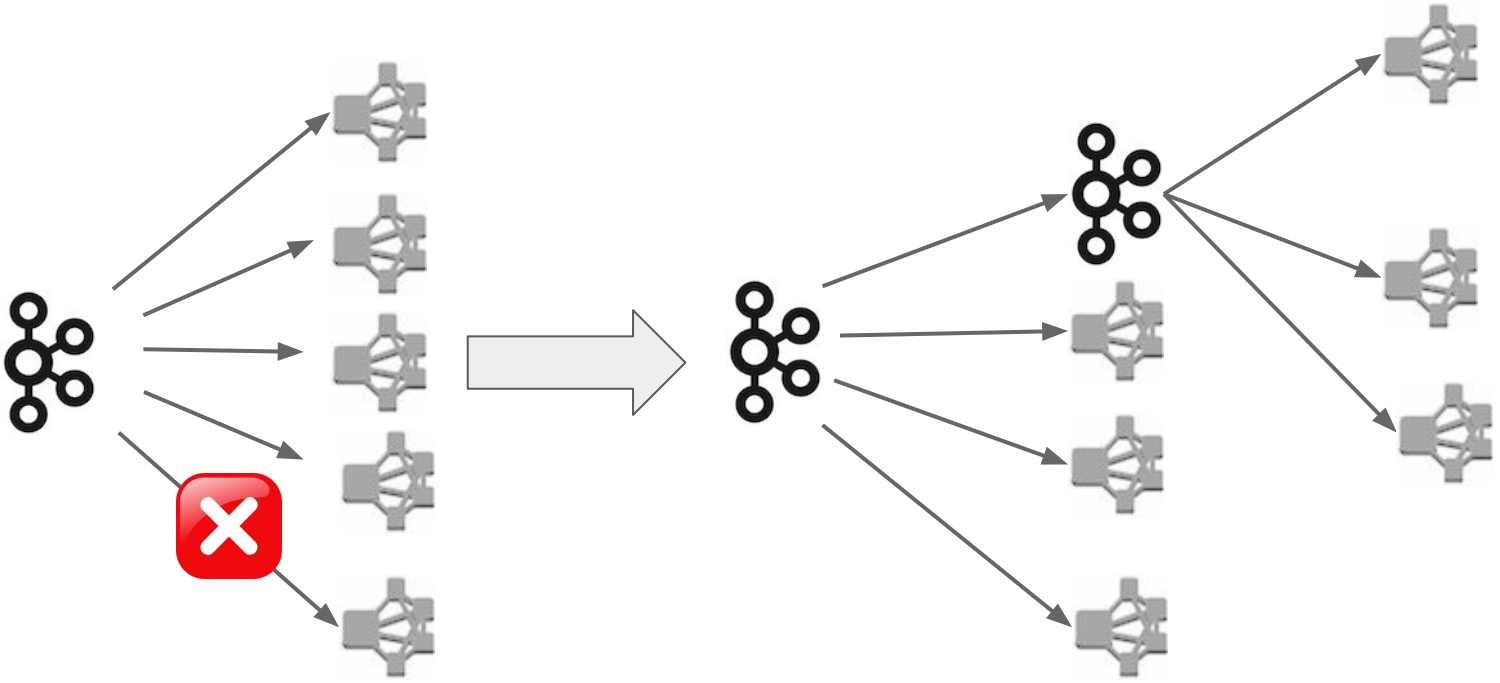
$$n = sm$$

$$\frac{F_n}{F_m} = \frac{(n-1)(n-2)\ldots(n-k+1)}{(m-1)(m-2)\ldots(m-k+1)} \geq s^{k-1}$$

Challenge From High Data Fan-Out



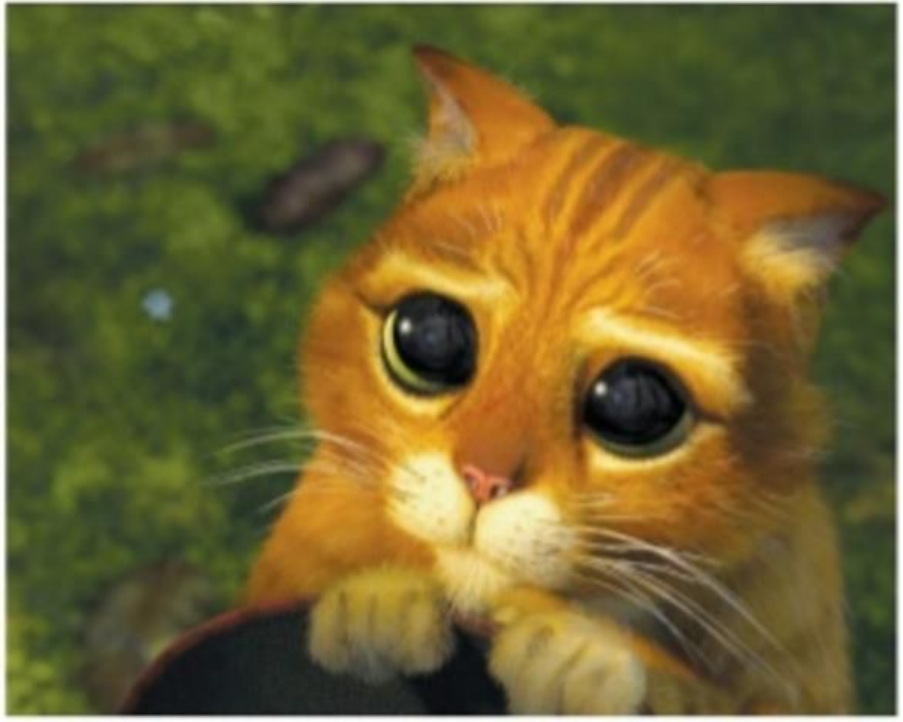
Scaling with Cluster Chaining



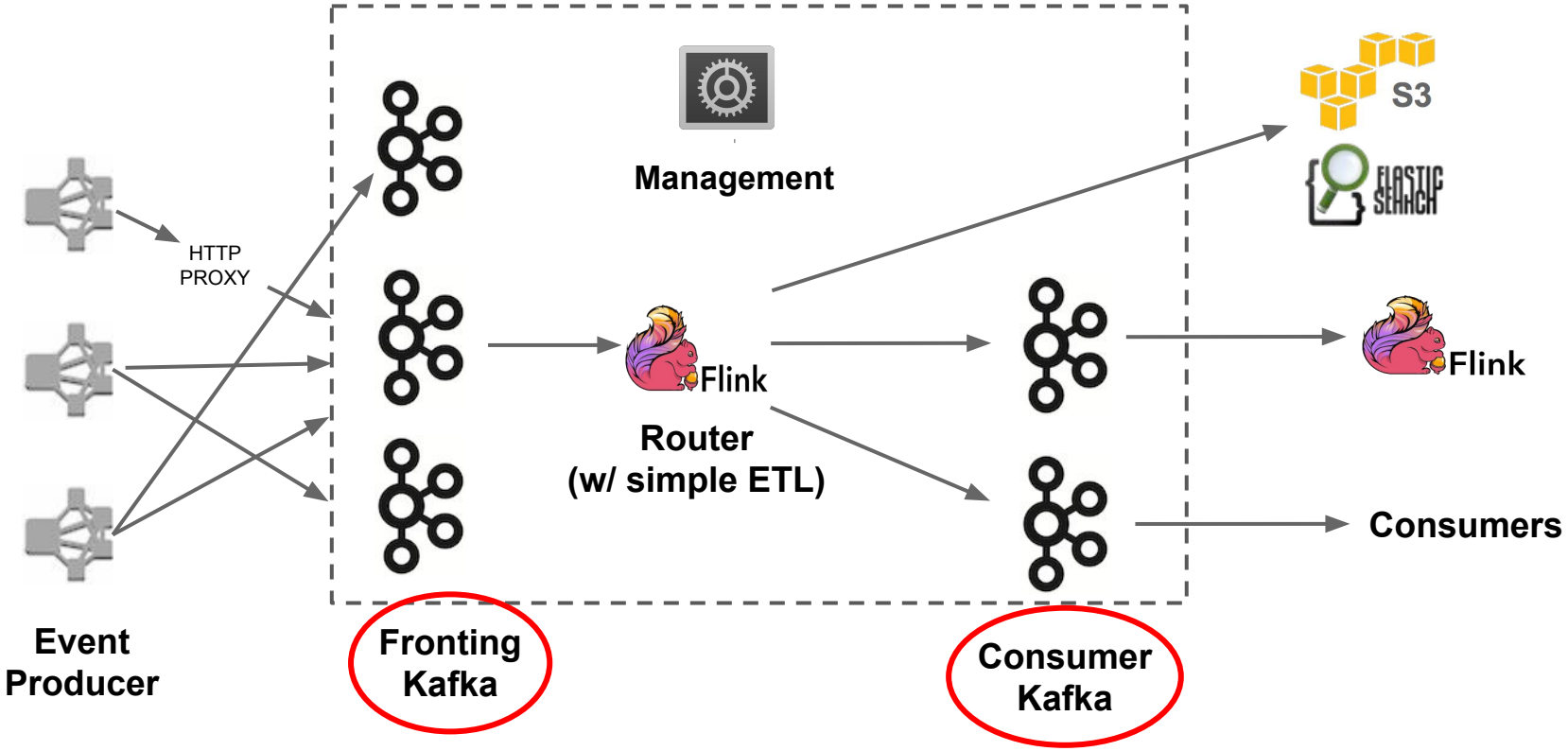
The Ideas of Multi-Cluster

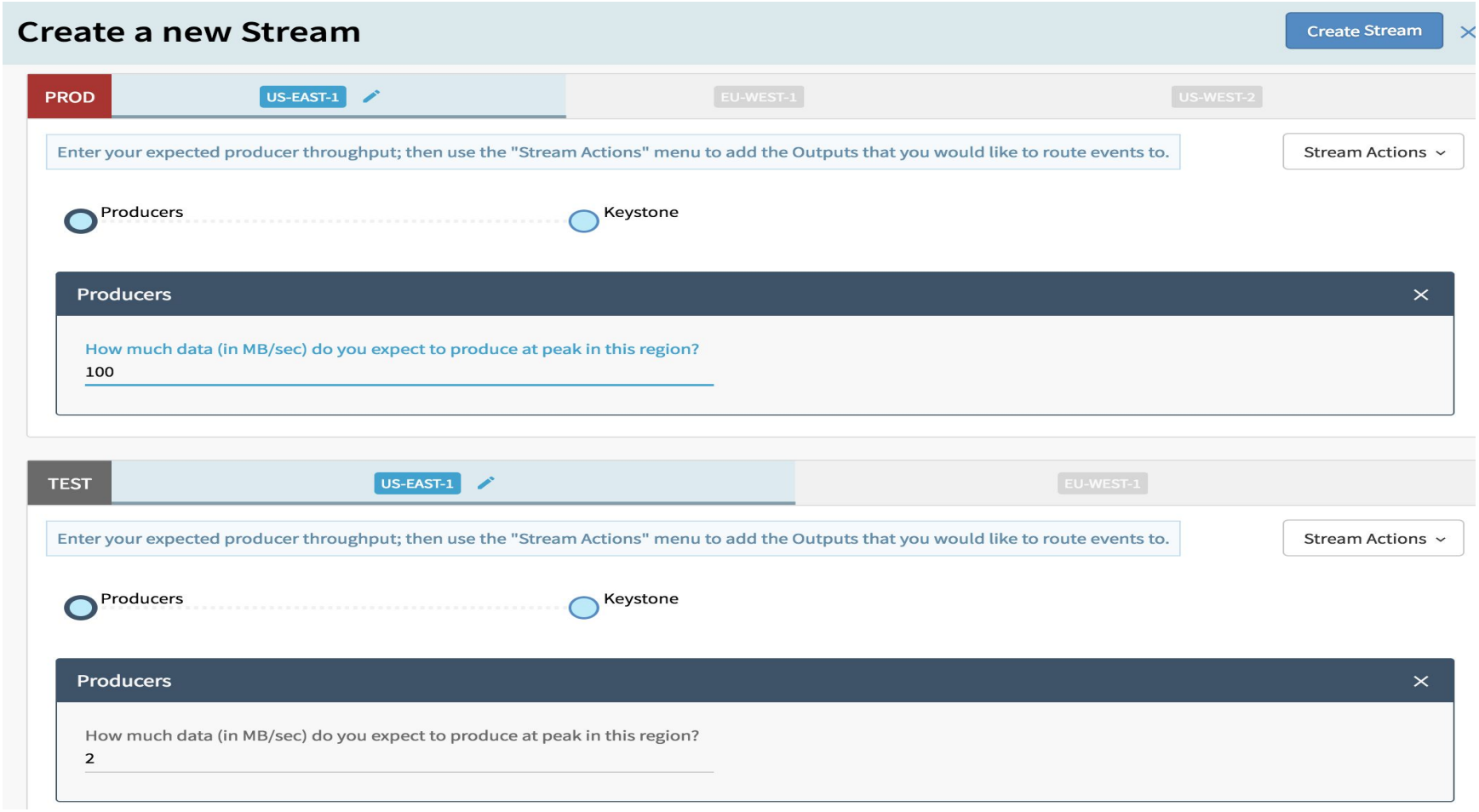
- Break up big clusters into small clusters
  - Mostly immutable
  - Scale by adding/removing clusters
  - Improve availability by failover with client traffic migration
- Connect clusters with routing services for high data fan-out
- Management service for automation and orchestration

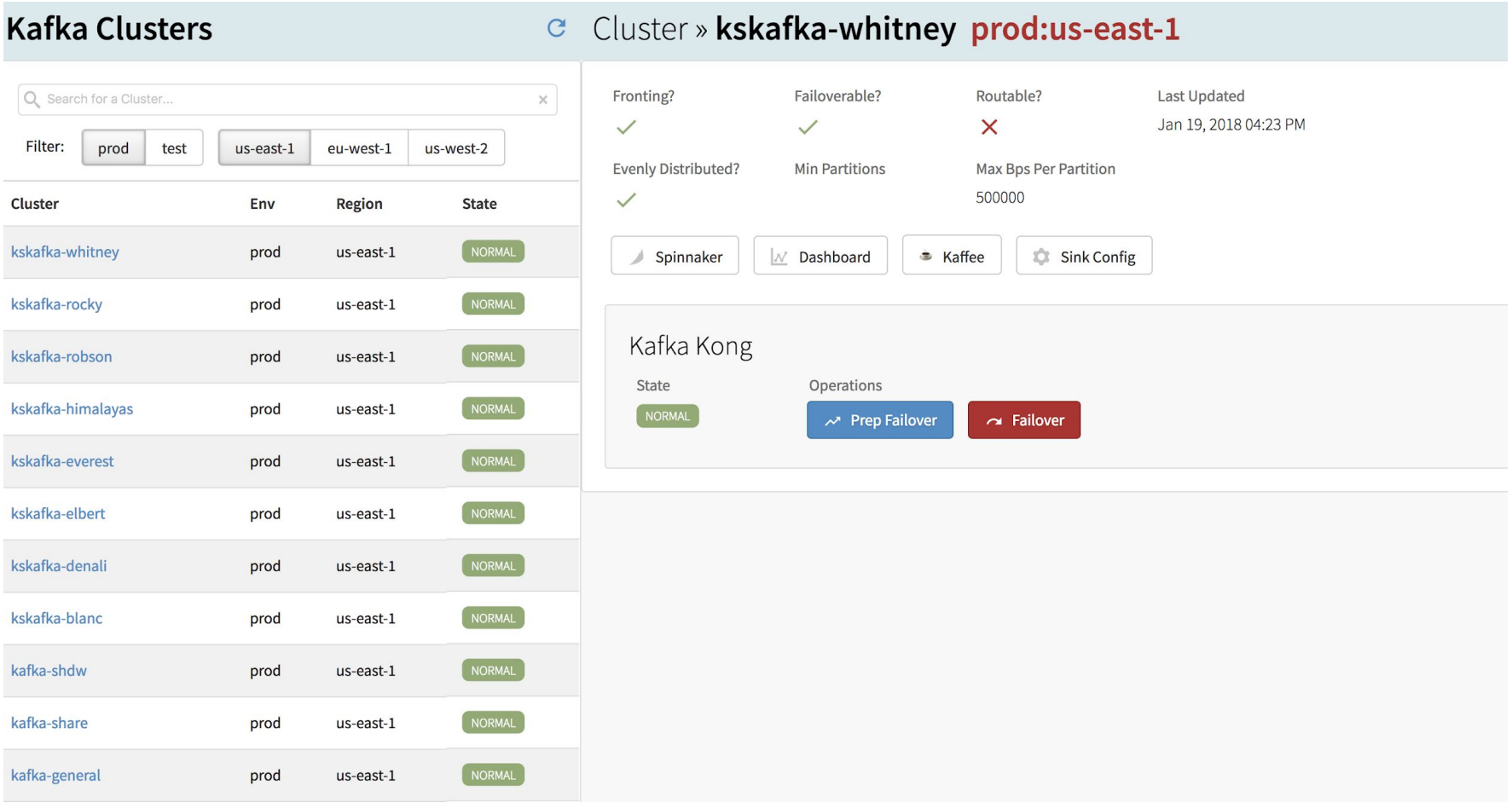
Pets To Cattle



Multi-Cluster Kafka Service At Netflix




Create a new Stream
Create Stream
PROD
US-EAST-1
EU-WEST-1
US-WEST-2
Enter your expected producer throughput; then use the "Stream Actions" menu to add the Outputs that you would like to route events to.
Stream Actions
Producers
Keystone
Producers
How much data (in MB/sec) do you expect to produce at peak in this region?
100
TEST
US-EAST-1
EU-WEST-1
Enter your expected producer throughput; then use the "Stream Actions" menu to add the Outputs that you would like to route events to.
Stream Actions
Producers
Keystone
Producers
How much data (in MB/sec) do you expect to produce at peak in this region?
2


Kafka Clusters
Cluster » kskafka-whitney prod:us-east-1
Search for a Cluster...
Filter:
prod
test
us-east-1
eu-west-1
us-west-2
Cluster
Env
Region
State
kskafka-whitney prod us-east-1 NORMAL
kskafka-rocky prod us-east-1 NORMAL
kskafka-robson prod us-east-1 NORMAL
kskafka-himalayas prod us-east-1 NORMAL
kskafka-everest prod us-east-1 NORMAL
kskafka-elbert prod us-east-1 NORMAL
kskafka-denali prod us-east-1 NORMAL
kskafka-blanc prod us-east-1 NORMAL
kafka-shdw prod us-east-1 NORMAL
kafka-share prod us-east-1 NORMAL
kafka-general prod us-east-1 NORMAL
Fronting?
Failoverable?
Routable?
Last Updated
Jan 19, 2018 04:23 PM
Evenly Distributed?
Min Partitions
Max Bps Per Partition
500000
Spinnaker
Dashboard
Kaffee
Sink Config
Kafka Kong
State
NORMAL
Operations
Prep Failover
Failover

Multi-Tenancy

Multi-Tenancy At Scale

- Cluster with the largest number of clients
  - Number of microservices accessing the cluster: **400+**
  - Average number of network connections per broker at peak: **33,000+**

The Goal

- Know your clients
- Ensure fair share of resources
- Better capacity planning

Client Registration

Authentication

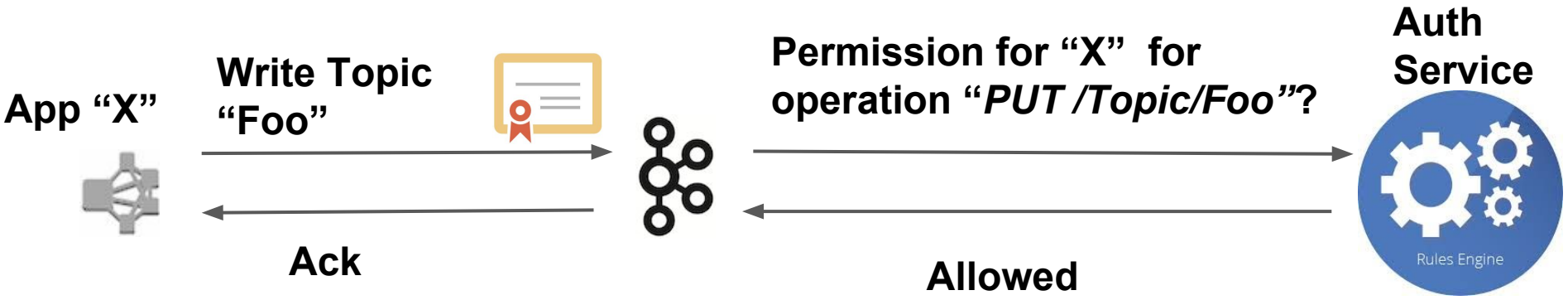
ACL and quota

Multi-Tenancy

- Identify your consumer - the old ways
  - Email, Slack …
  - Code search
  - TCPdump

Identity with Security

- Integrate with Netflix security system
  - Utilize standard Netflix client certs on every instance
  - Utilize Netflix authorization service to define policies
  - Map Kafka operations to HTTP methods
- Result - ACL and quota based on true application identity


App “X”
Write Topic “Foo”
Ack
Permission for “X” for operation “*PUT /Topic/Foo*”?
Allowed
Auth Service
Rules Engine

Takeaways

- Improve scalability and availability with multiple clusters
  - Scale with traffic by adding/removing clusters
  - Failover by migrating client traffic
  - Chain clusters to provide better solution for data fan-out
- Integrate with SSL infrastructure and your own auth service to lay the foundation of multi-tenancy management


Thank You