

Scaling the Yelp's logging pipeline with Apache Kafka

Enrico Canzonieri

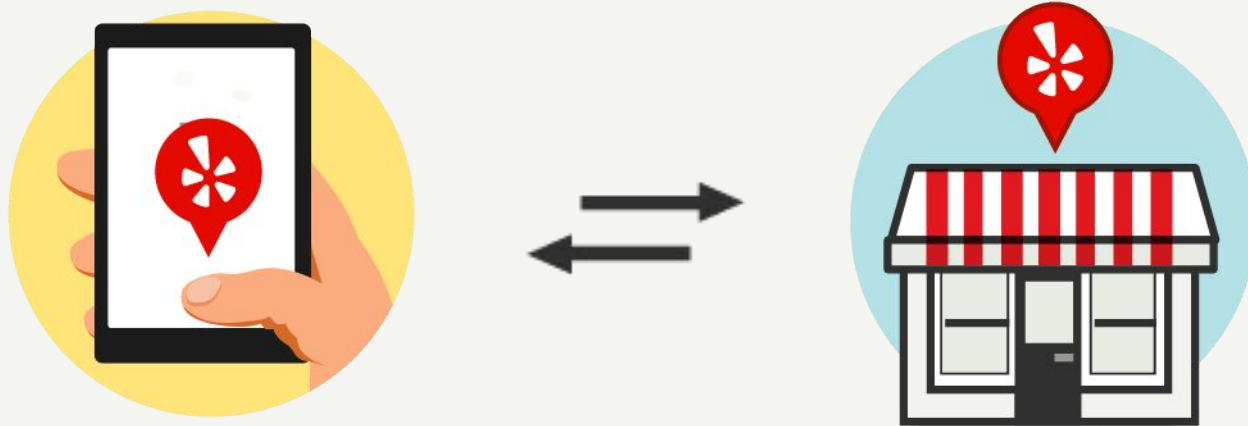
enrico@yelp.com

@EnricoC89



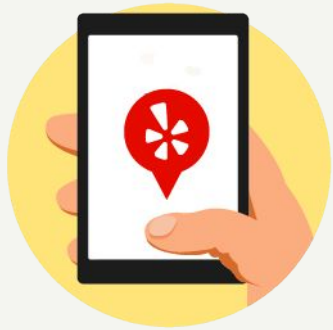
Yelp's Mission

Connecting people with great local businesses.



Yelp Stats

As of Q1 2016



90M



102M



70%



32



What to expect

- High-level architecture overview
- Kafka best-practices
- No code



Logging? What is that?



A log is a stream of events

```
12:38:17.687 RD00155DA946E3 DEBUG Starting to save logs 64
12:38:17.687 RD00155DA946E3 DEBUG Got nest client 64c91e09
12:38:17.687 RD00155DA946E3 DEBUG Starting to see if index
12:38:17.687 RD00155DA946E3 DEBUG Saving logs to index c27
12:38:17.687 RD00155DA946E3 DEBUG Saving logs to index c27
12:38:17.687 RD00155DA946E3 DEBUG Saving logs to index com
12:38:17.687 RD00155DA946E3 DEBUG Got nest client ff8b1dc6
12:38:17.687 RD00155DA946E3 DEBUG Starting to see if index
12:38:17.687 RD00155DA946E3 DEBUG Saving logs to index c27
12:38:17.687 RD00155DA946E3 DEBUG Saving logs to index com
12:38:17.687 RD00155DA946E3 DEBUG Starting to see if index
12:38:17.687 RD00155DA946E3 DEBUG Got nest client 61fd2994
```

```
12: }
12: }
176 warn("Replica %d for partition %s reset its fetch offset from %d to current leader %d's latest offset %d"
177     .format(brokerConfig.brokerId, topicAndPartition, replica.logEndOffset.messageOffset, sourceBroker.id, leaderEndOffset))
178     replicaMgr.logManager.truncateTo(Map(topicAndPartition -> leaderEndOffset))
```

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326 "http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I ;Nav)"
```

```
Jul 23 22:09:36 Example-iMac kernel[0]: memorystatus_thread: idle exiting pid 34364 [com.apple.ShareK]
Jul 23 22:09:36 Example-iMac com.apple.launchd.peruser.89[33810] (com.apple.cfprefsd.xpc.agent[33814]): Exited: Killed:
Jul 23 22:09:36 Example-iMac kernel[0]: memorystatus_thread: idle exiting pid 33814 [cfprefsd]
Jul 23 22:09:38 Example-iMac com.apple.launchd.peruser.501[237] (com.apple.coreservices.appleid.authentication[33785]):
```

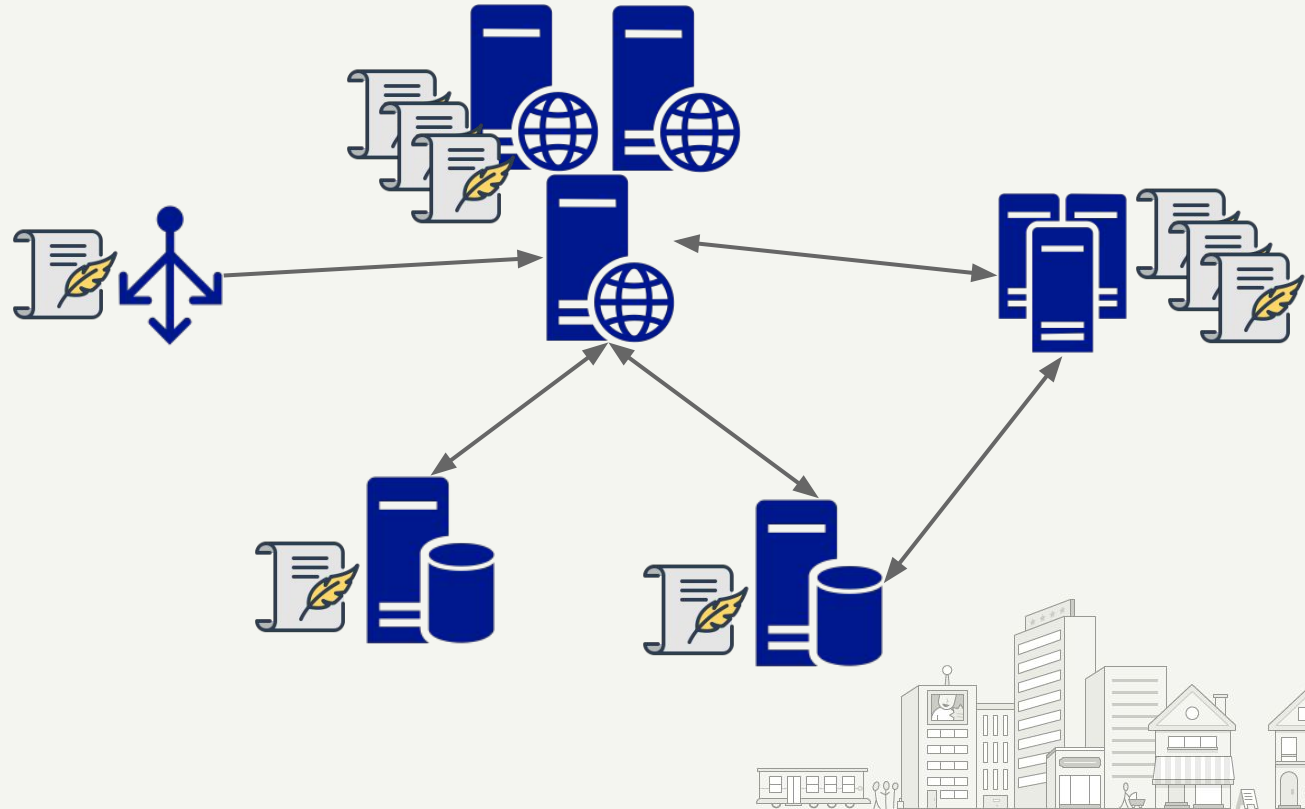
Logs provide valuable
operational and business
visibility.



Producing and consuming logs should be easy



Logs at scale



Consuming logs becomes hard



Solution: Centralized logging!



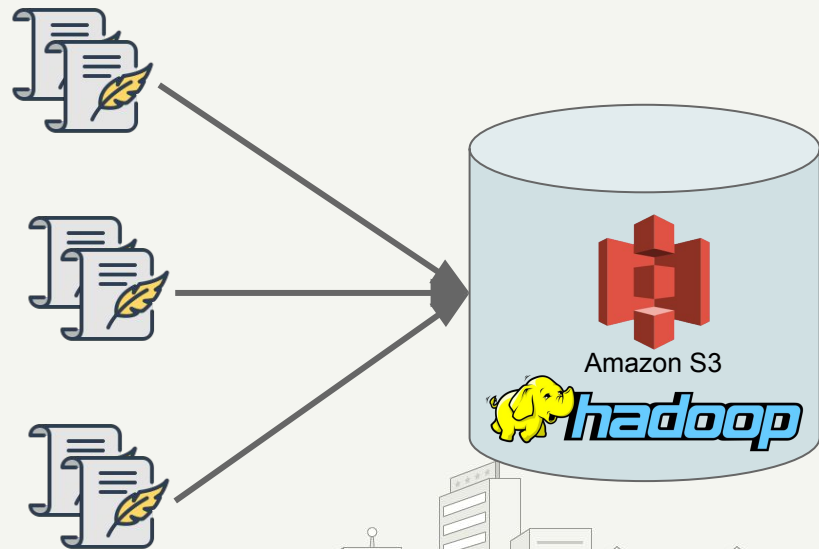
Simple aggregation strategy: upload all logs to a centralized datastore

Pros:

- Easy to implement (cron, logrotate, etc)
- Unique place to access logs

Cons:

- No real time streaming
- Colocation not aggregation



Log aggregation systems



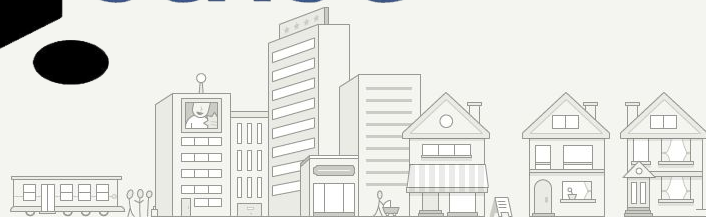
fluentd



syslog-ng

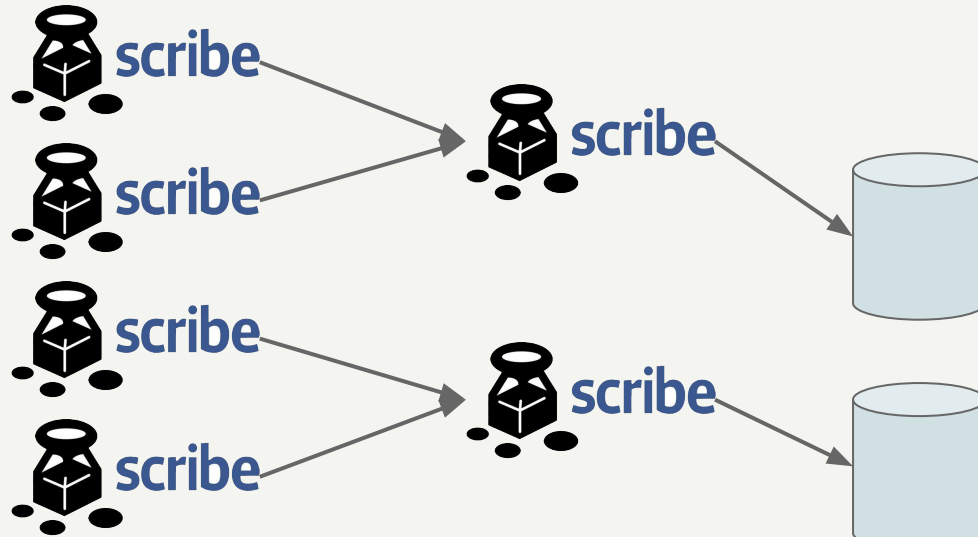


scribe

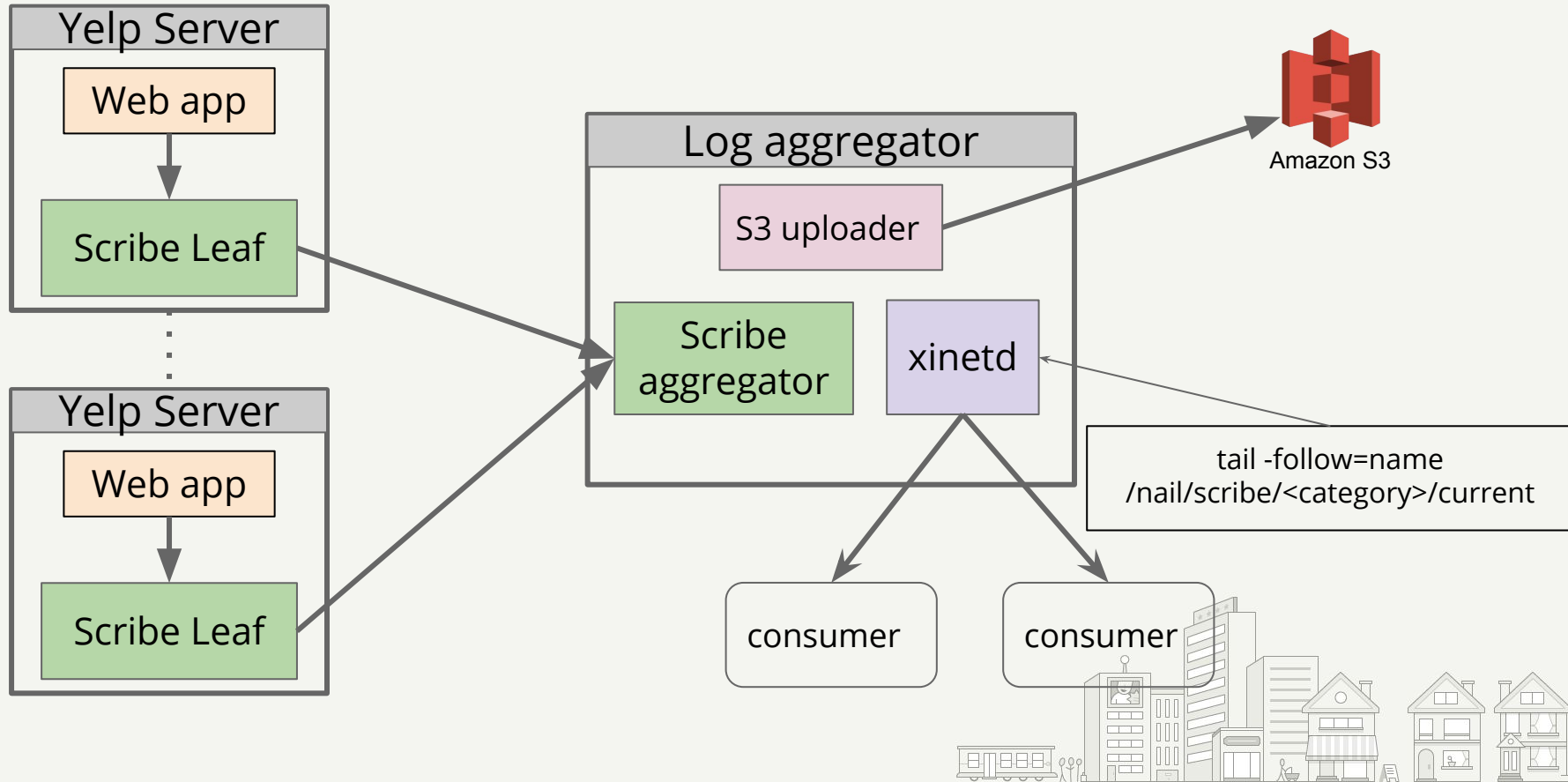


Scribe

Push based architecture



Yelp's logging pipeline V1.0



Why use Scribe?

- Simple to configure
- Multilang support (via Thrift)
- Stable (most of the times) and fast
- Support arbitrary message size

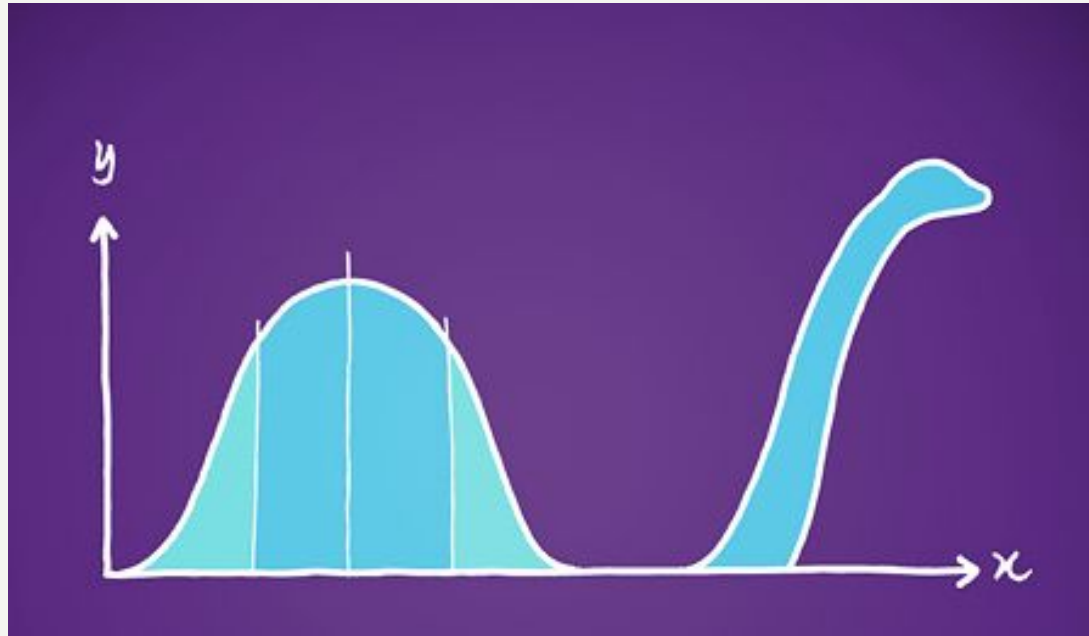


Why move away from Scribe?

- No support for log consumption
- Doesn't really scale
- No replication
- Abandoned project
- Lack of plugins



Stream processing: Statmonster



Stream processing: Statmonster

- Statmonster is a real-time metrics pipeline
- The first stage extract multiple metrics from each line
- Consumes most of the high volume logs



Stream processing: Statmonster

- In 2013 the metrics from high volume logs started falling behind
- The first stage of the pipeline was too slow to keep up with the increased volume of logs



Stream processing: Statmonster

- Short term solution: sample logs to save timeliness
- Long term solution: run Statmonster on Apache Storm.



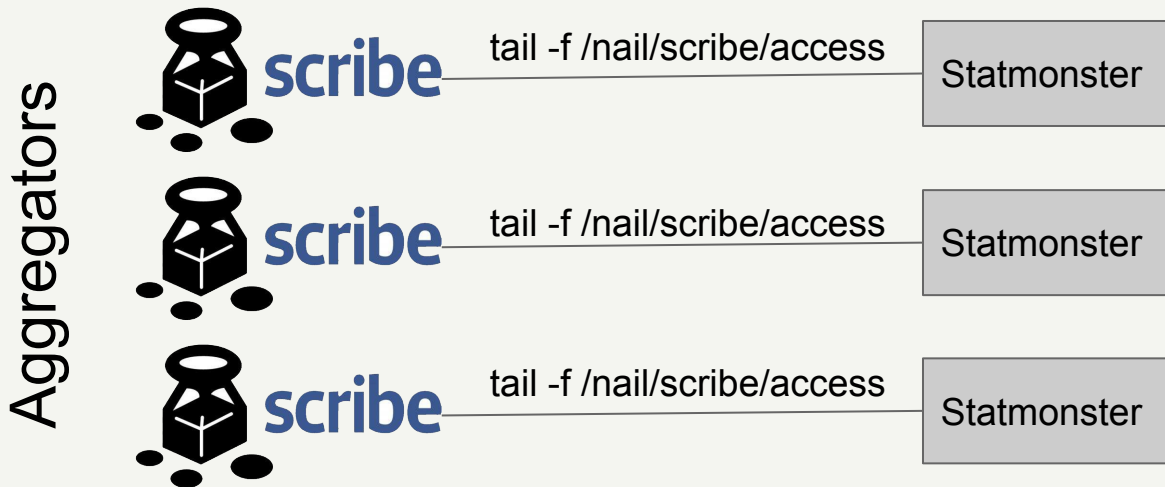
Stream processing: Statmonster

- Short term solution: sample logs to save timeliness
- ~~● Long term solution: run Statmonster on Apache Storm.~~



The real bottleneck

Can't scale the consumer without increasing the number of physical log aggregators

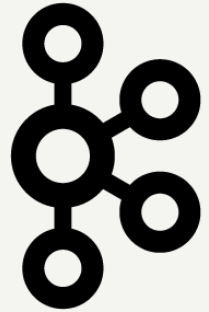


Need to scale the source!

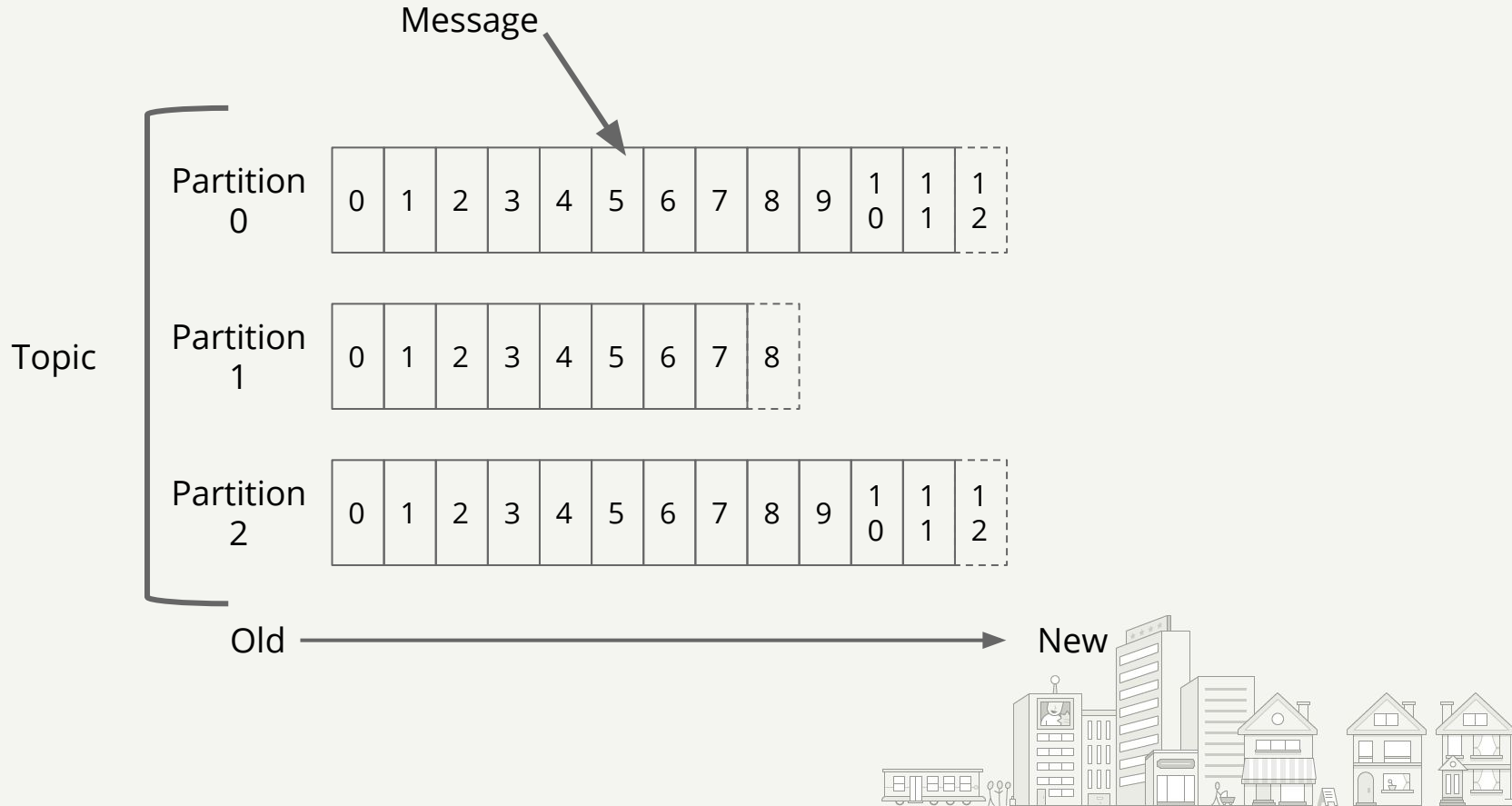


Kafka into the mix

- High-throughput Publish-subscribe messaging system
- Distributed, replicated commit log
- Client library available for a variety of languages
- Configurable retention



Kafka Message (topic, partition, offset)



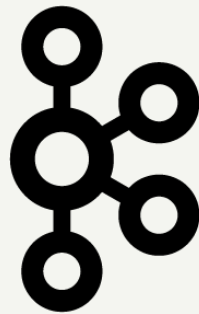
Kafka Producer

- Java producer is async by default
- Messages are produced in batch
- Support either *at-least-once* or *at-most-once* semantic
- Support compression out of the box



Kafka Consumer

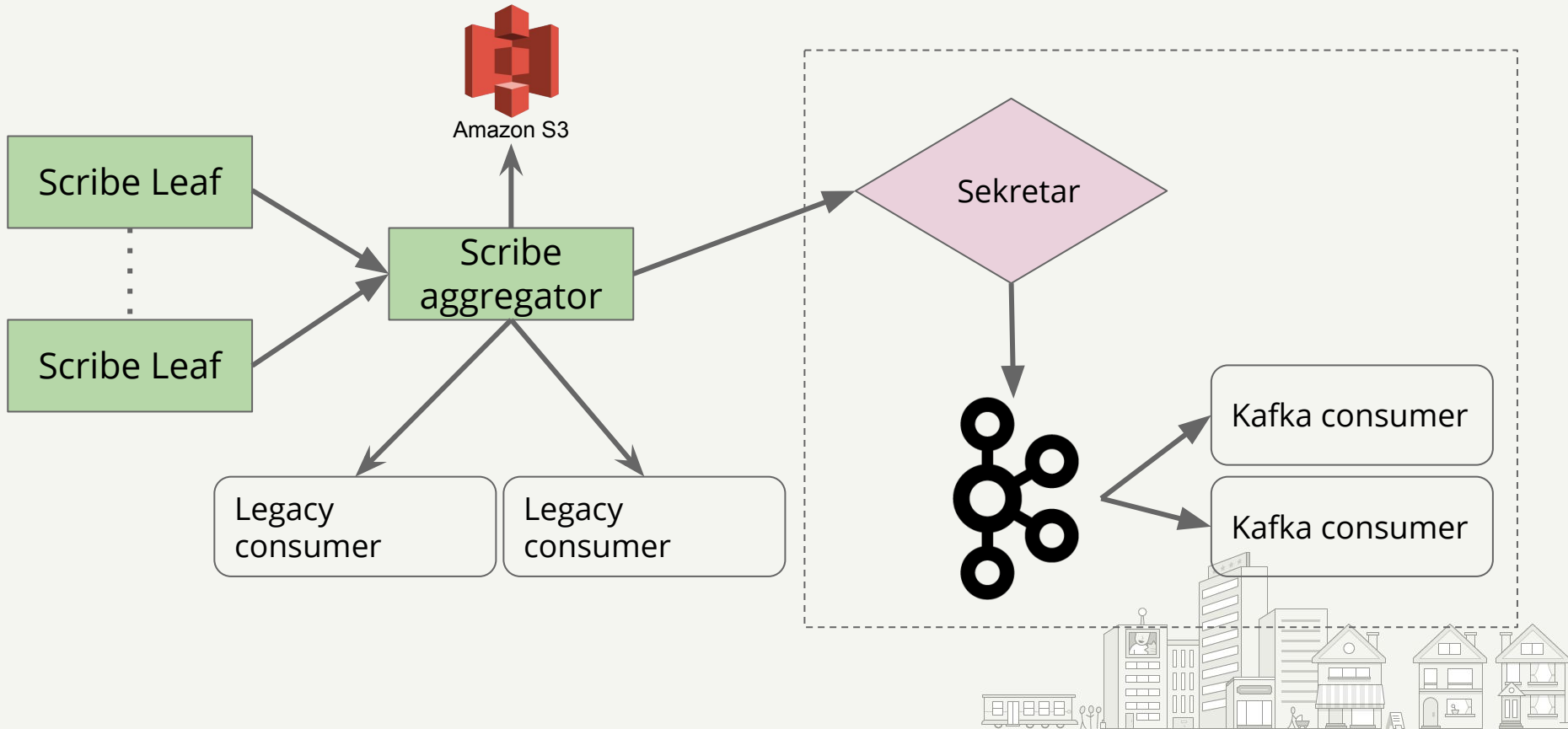
- Identified by a *group id*
- Consumers coordinate to consume from different partitions
- Ability to re-play messages
- Native support for offset commit



Architecture overview



Yelp's logging pipeline V1.5

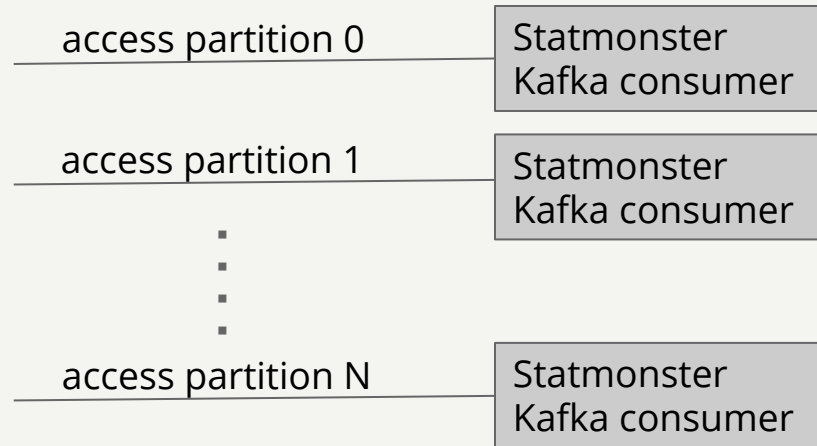
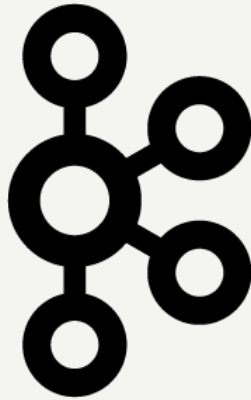


Sekretar

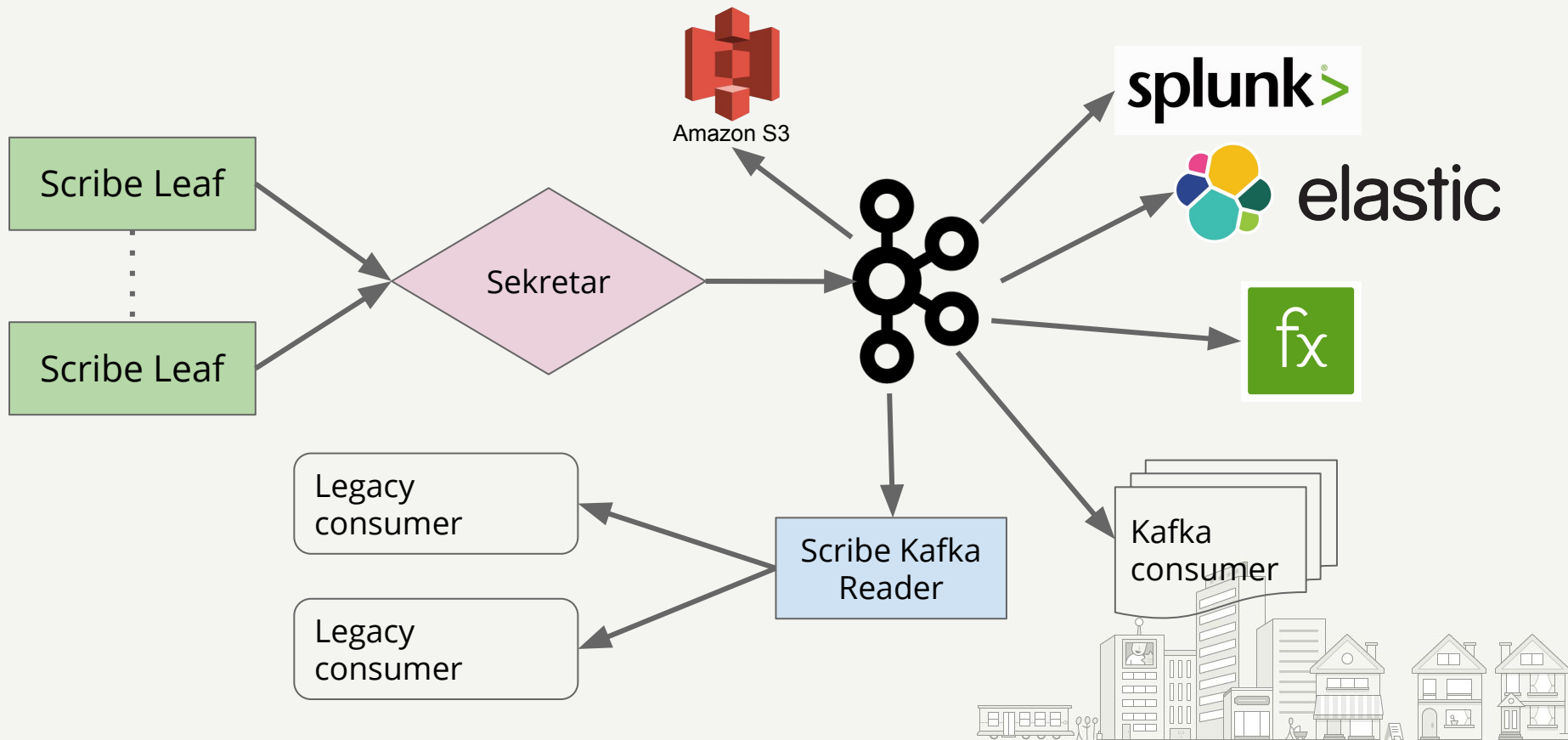
- Act as a Scribe aggregator speaking the Scribe protocol
- Produce messages to Kafka
- Map a log category into a topic
- Act as an intermediate buffer
- Handle big messages



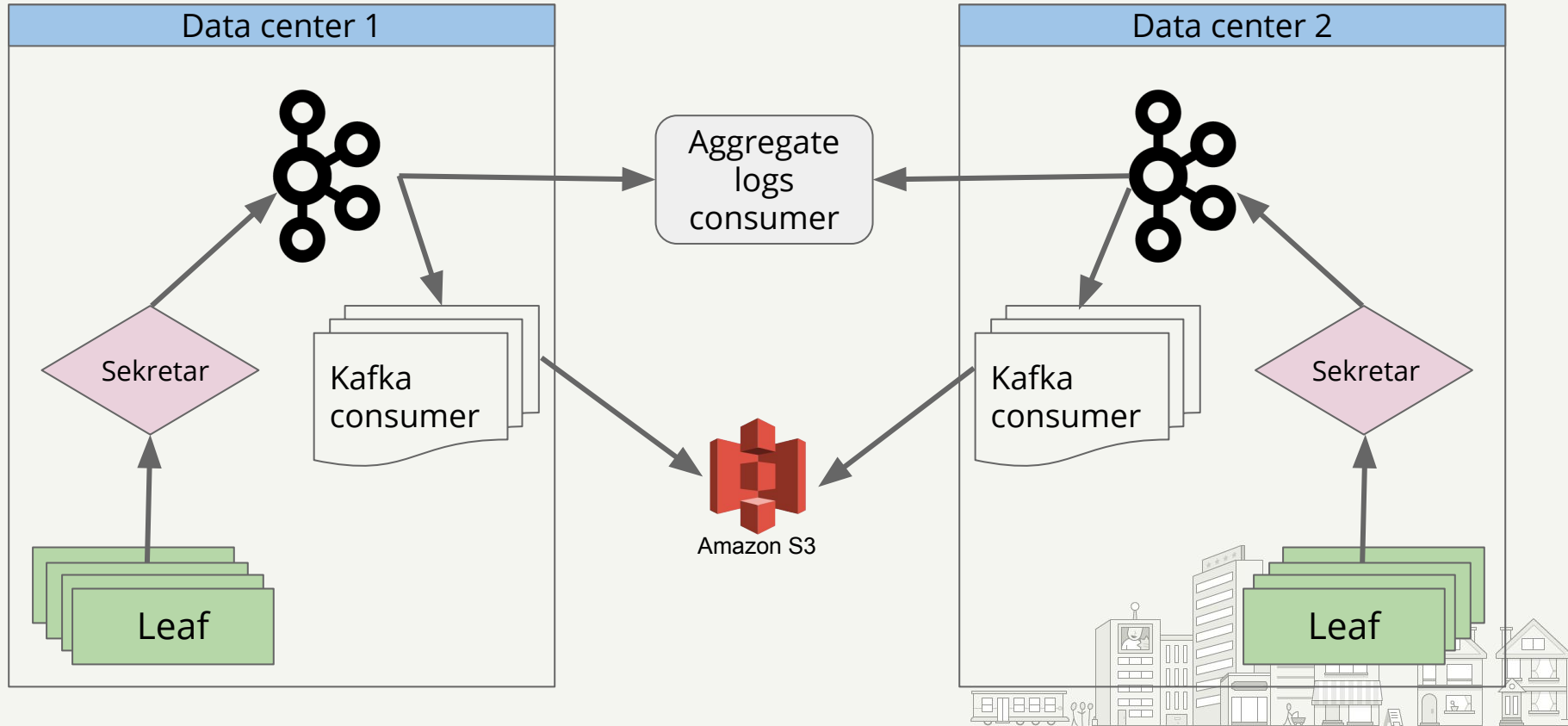
Statmonster 2.0



Yelp's logging pipeline V2.0



Multi-regional architecture



Configuration



Availability vs Consistency

Cluster side:

- **Unclean leader election** → Disabled
unclean.leader.election.enable
- **Replication factor** → 3: 1 leader + 2 followers
default.replication.factor
- **Minimum in-sync replicas (ISR)** → Quorum: $3/2 + 1 = 2$ min.
insync.replicas

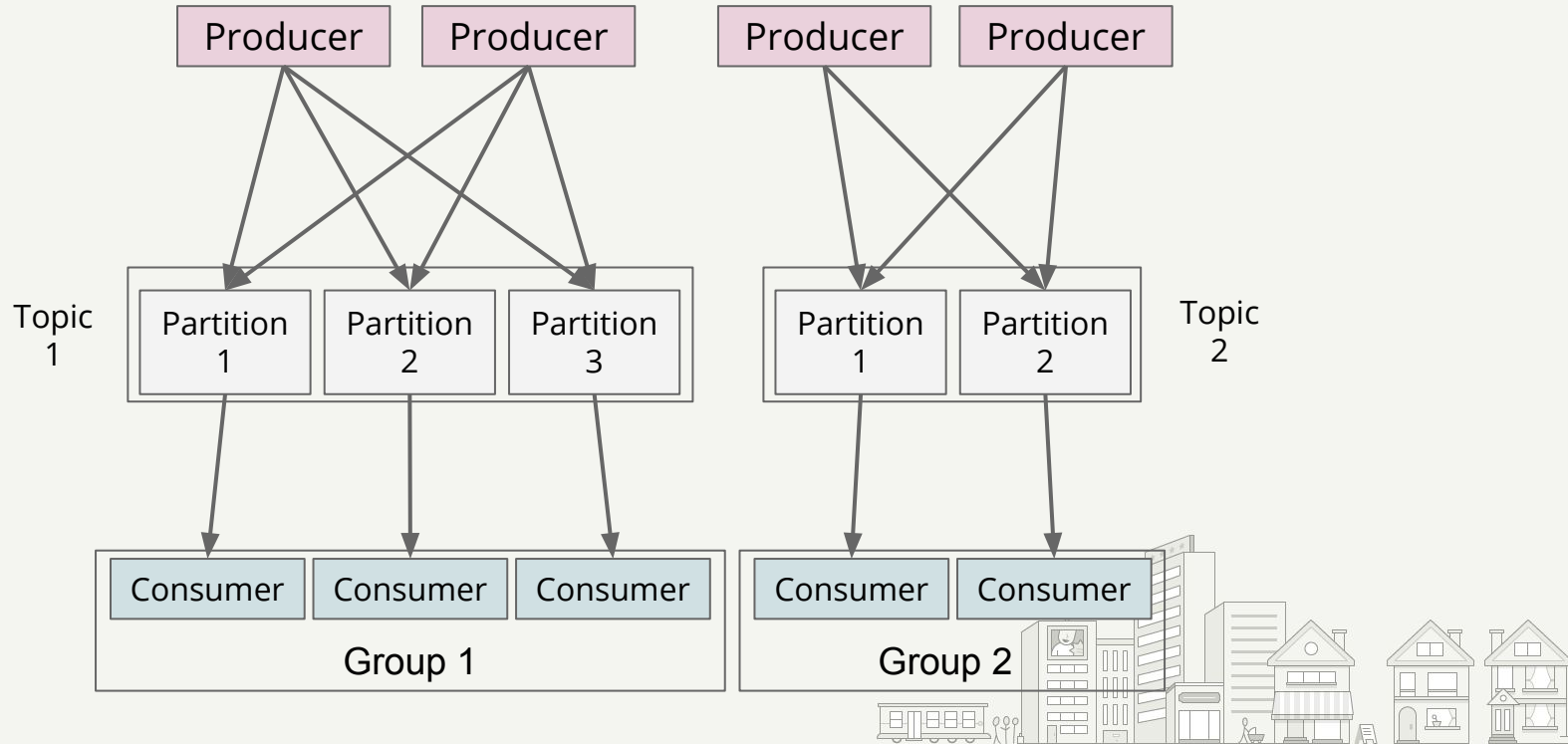
Producer side:

- **acks** → All (or -1): all replicas in the ISR need to ack
acks



Provisioning partitions

How many partitions for a topic?



Criteria 1: consumer

Consumer speed

Number of different consumer groups



Criteria 2: Kafka

Physical Limits of a single broker egress/ingress

Log retention

Recovery speed upon broker failures



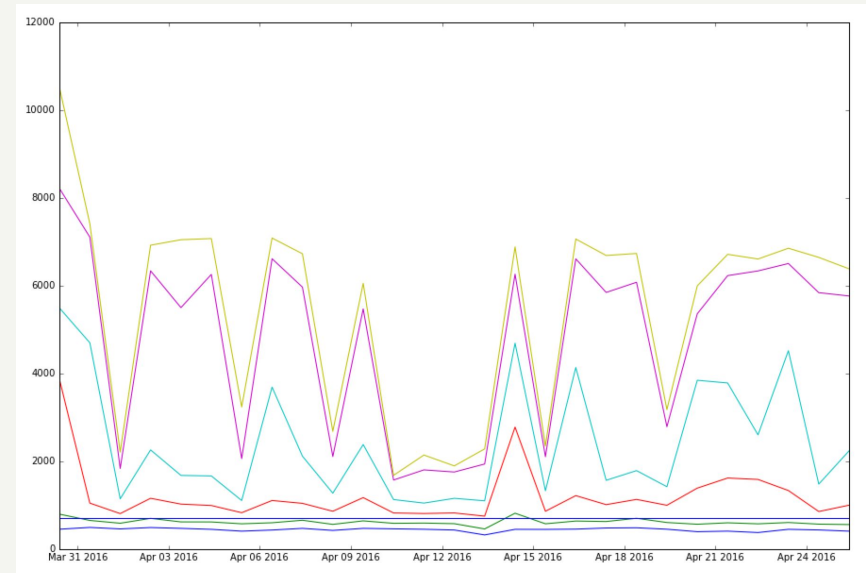
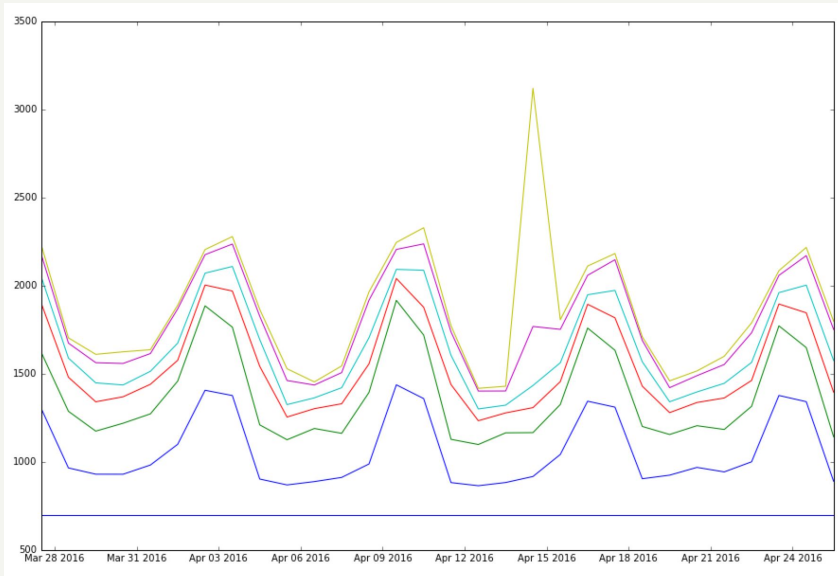
Criteria 3: producer

Ingress rate into Kafka in msg/s and bytes/s



Accounting for spikes

Message rate (msg/s) percentiles: 50, 75, 90, 95, 99, 99.5



Partitions auto-creation

Regular traffic increase

Automatically increase the number of partitions based on 99th percentile of bytes/s and msg/s over a time window of 3 days.

Use a combined threshold of 700 msg/s and 500 KiB/s.

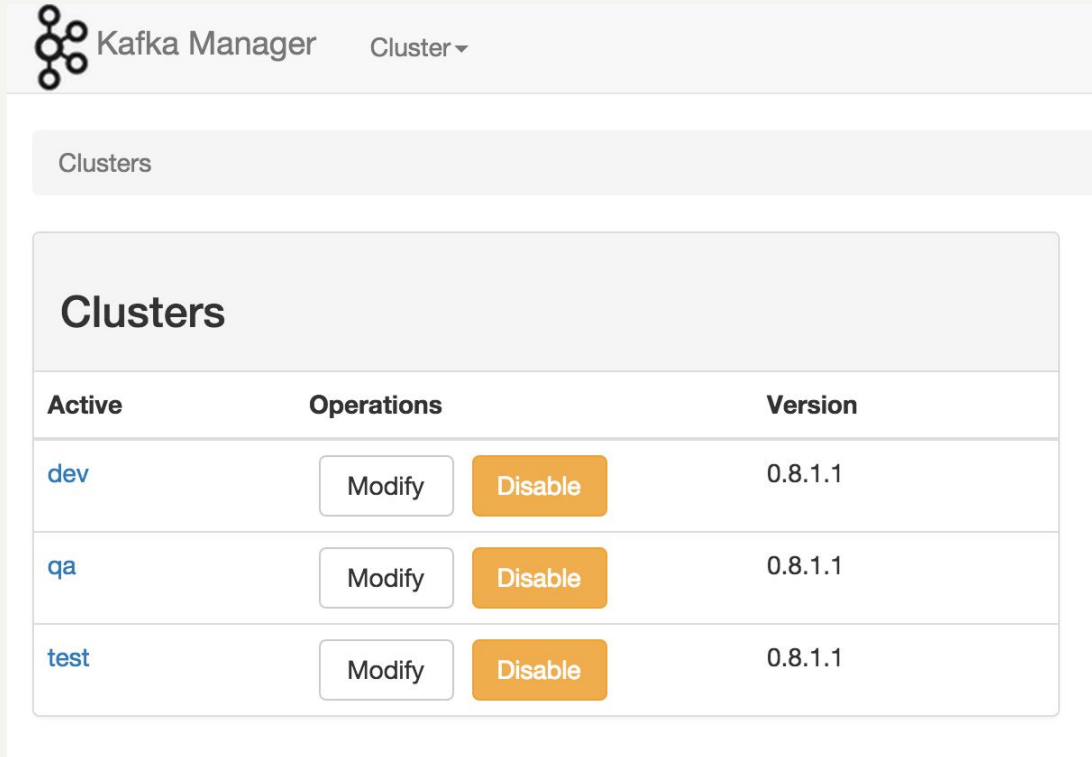


Tooling



Kafka Manager

<https://github.com/yahoo/kafka-manager>



The screenshot shows the Kafka Manager web interface. At the top left is the Kafka Manager logo (a cluster of nodes) and the text "Kafka Manager". To its right is a "Cluster" dropdown menu. Below this is a "Clusters" section header. The main content is a table with three columns: "Active", "Operations", and "Version". The table lists three clusters: "dev", "qa", and "test". Each cluster has a "Modify" button and a "Disable" button. The "Version" for all clusters is "0.8.1.1".

Active	Operations	Version
dev	<input type="button" value="Modify"/> <input type="button" value="Disable"/>	0.8.1.1
qa	<input type="button" value="Modify"/> <input type="button" value="Disable"/>	0.8.1.1
test	<input type="button" value="Modify"/> <input type="button" value="Disable"/>	0.8.1.1



Kafka Utils

Contains several command line tools to help operating and maintaining a Kafka cluster.

<https://github.com/Yelp/kafka-utils>

- Cluster rolling restart
- Consumer offset management
- Cluster rebalance and broker decommission
- Healthchecks



Monitoring

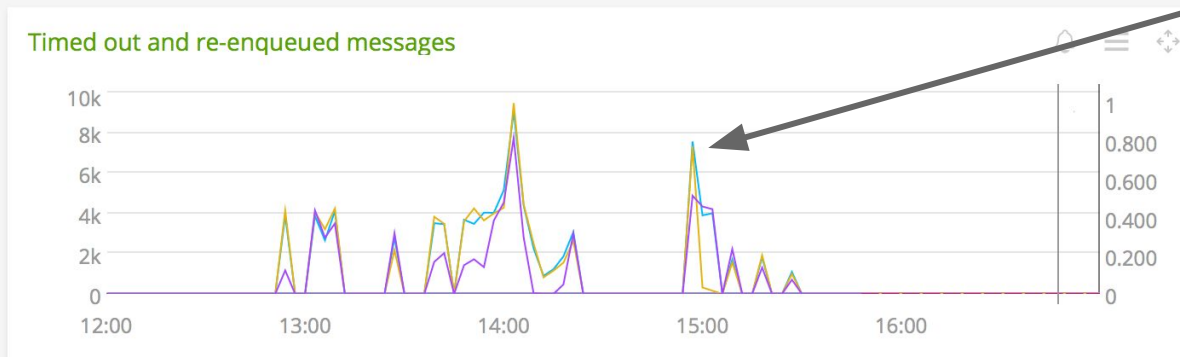
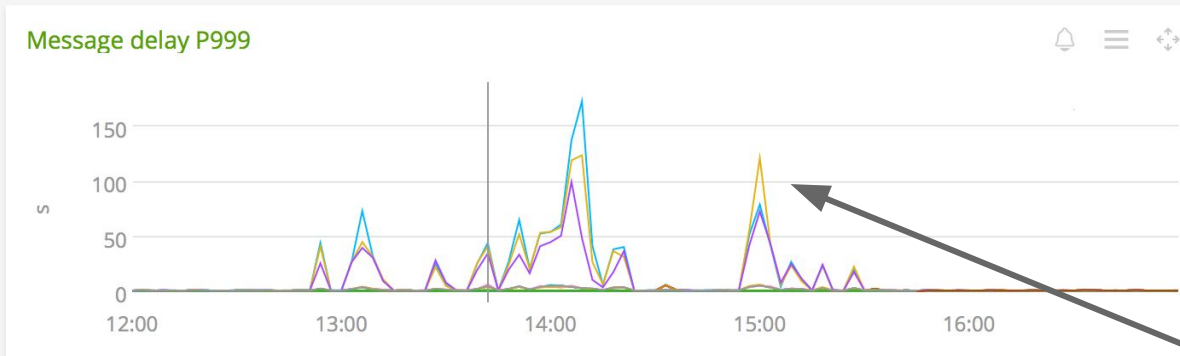


Monitoring Sekretar

- Ingress: msg/s, bytes/s
- Egress: msg/s, (compressed) bytes/s
- Message delay
- Producer latency
- Producer memory buffer
- Timed-out message count



Monitoring Sekretar



Unhealthy
Kafka cluster



Kafka monitoring

Kafka exports metrics via jmx (<http://docs.confluent.io/1.0/kafka/monitoring.html>):

- Offline partitions
- Under replicated partitions
- Controller count

Kafka-Utils includes a check for min.isr



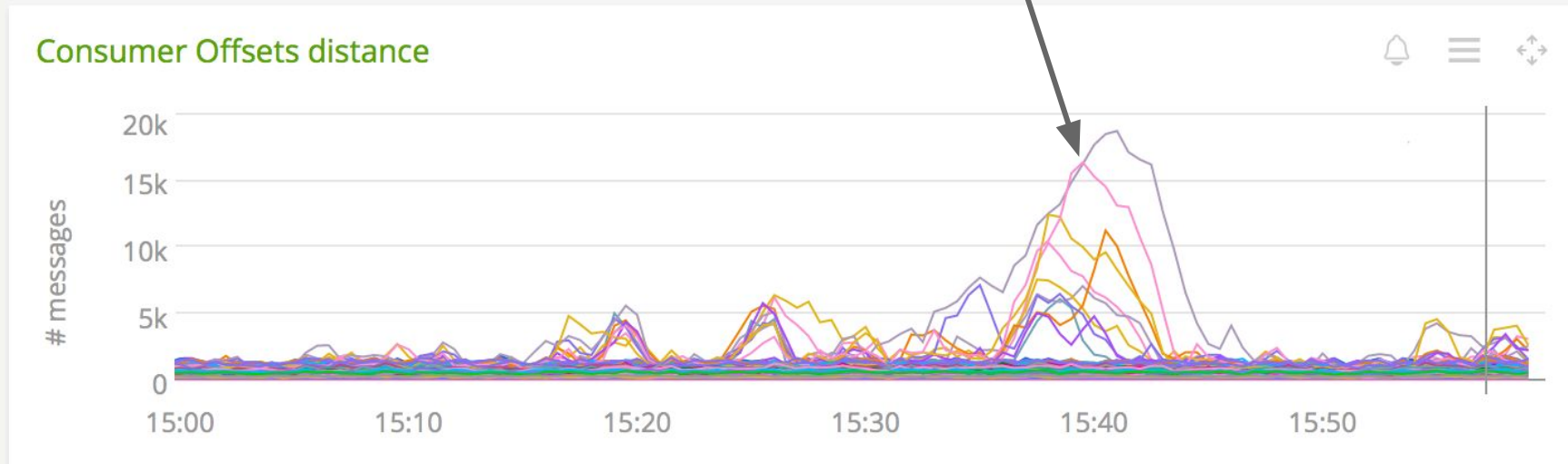
Consumer monitoring

- Consumer speed (msg/s)
- Kafka speed
- Consumer lag (msg)



Consumer monitoring

Consumer slow down or
logs volume increase



Apache Kafka @ Yelp

18 clusters

~ 100 brokers

> 20K topics

~ 45K partitions

> 5TB of data per day

~ 25 Billion messages per day





fb.com/YelpEngineers



[@YelpEngineering](https://twitter.com/YelpEngineering)



engineeringblog.yelp.com



github.com/yelp