Apache Kafka Development

Experience and Lesson Learned

Guozhang Wang

Kafka Meetup Beijing, April 15, 2017

*A Short History of **Kafka***

A Short History

- ***2010.10**: First commit of Kafka*

A Short History

- ***2010.10**: First commit of Kafka*
- ***2011.07**: Enters Apache Incubator*
  - *Release 0.7.0: compression, mirror-maker*

A Short History

- ***2010.10**: First commit of Kafka*
- ***2011.07**: Enters Apache Incubator*
  - *Release 0.7.0: compression, mirror-maker*
- ***2012.10**: Graduated to top-level project*
  - *Release 0.8.0: intra-cluster replication*

A Short History

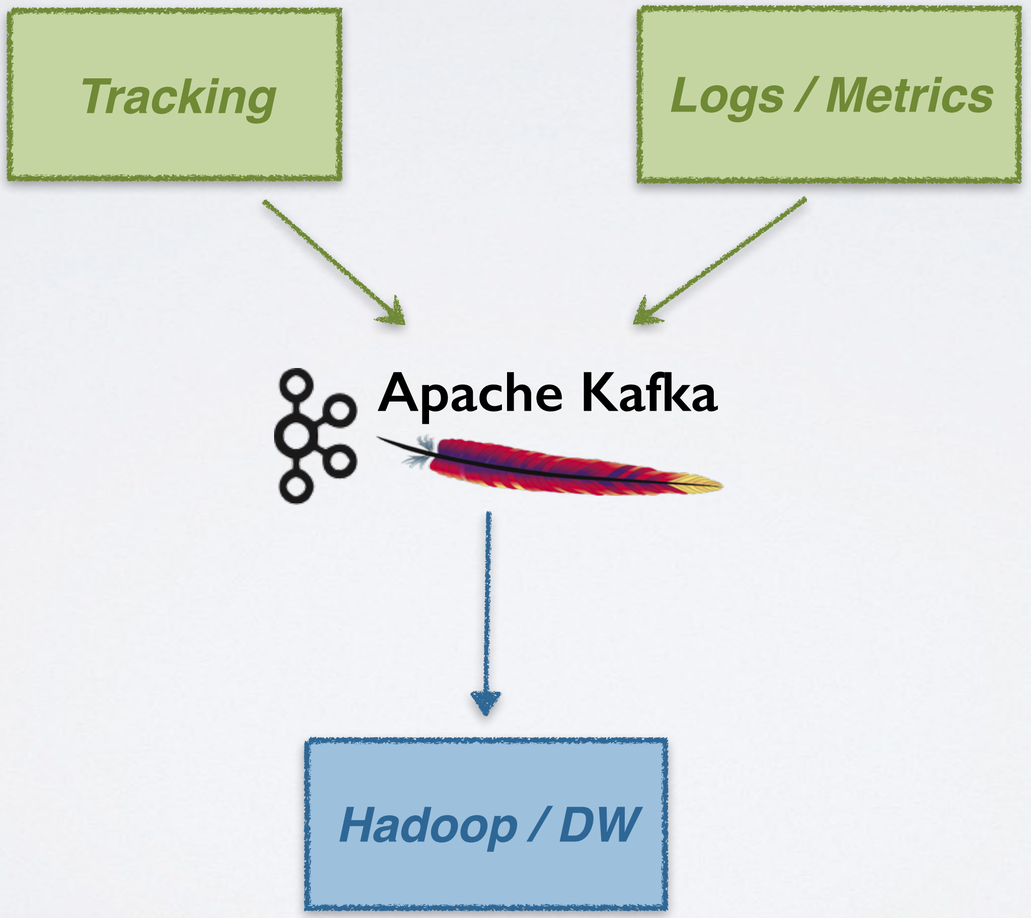
- ***2010.10**: First commit of Kafka*
- ***2011.07**: Enters Apache Incubator*
  - *Release 0.7.0: compression, mirror-maker*
- ***2012.10**: Graduated to top-level project*
  - *Release 0.8.0: intra-cluster replication*
- ***2014.11**: Confluent founded*
  - *Release 0.8.2: new producer, quota*
  - *Release 0.9.0: **Kafka Connect**, new consumer, security*
  - *Release 0.10.0: **Kafka Streams**, timestamps, rack awareness*

What is Kafka, Really?

a scalable pub-sub messaging system..

[NetDB 2011]

Example: Pub-Sub Messaging

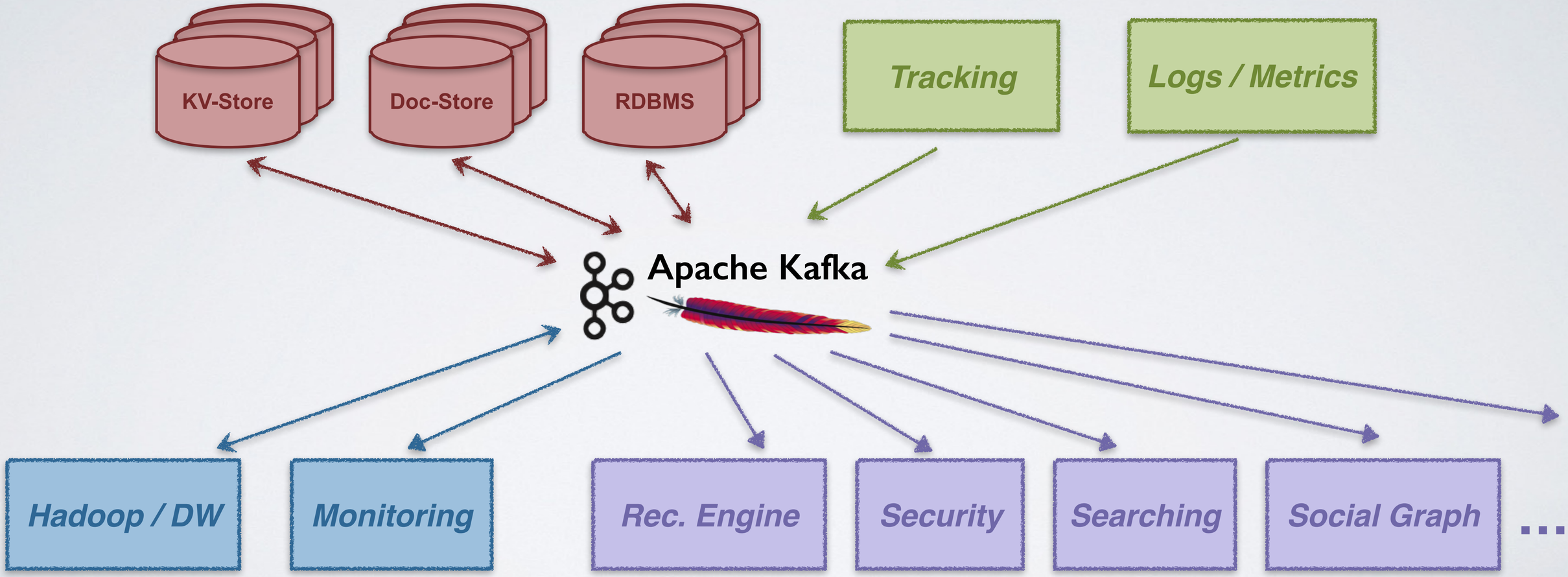


What is Kafka, Really?

a scalable pub-sub messaging system.. *[NetDB 2011]*

a real-time data pipeline.. *[Hadoop Summit 2013]*

Example: Centralized Data Pipeline



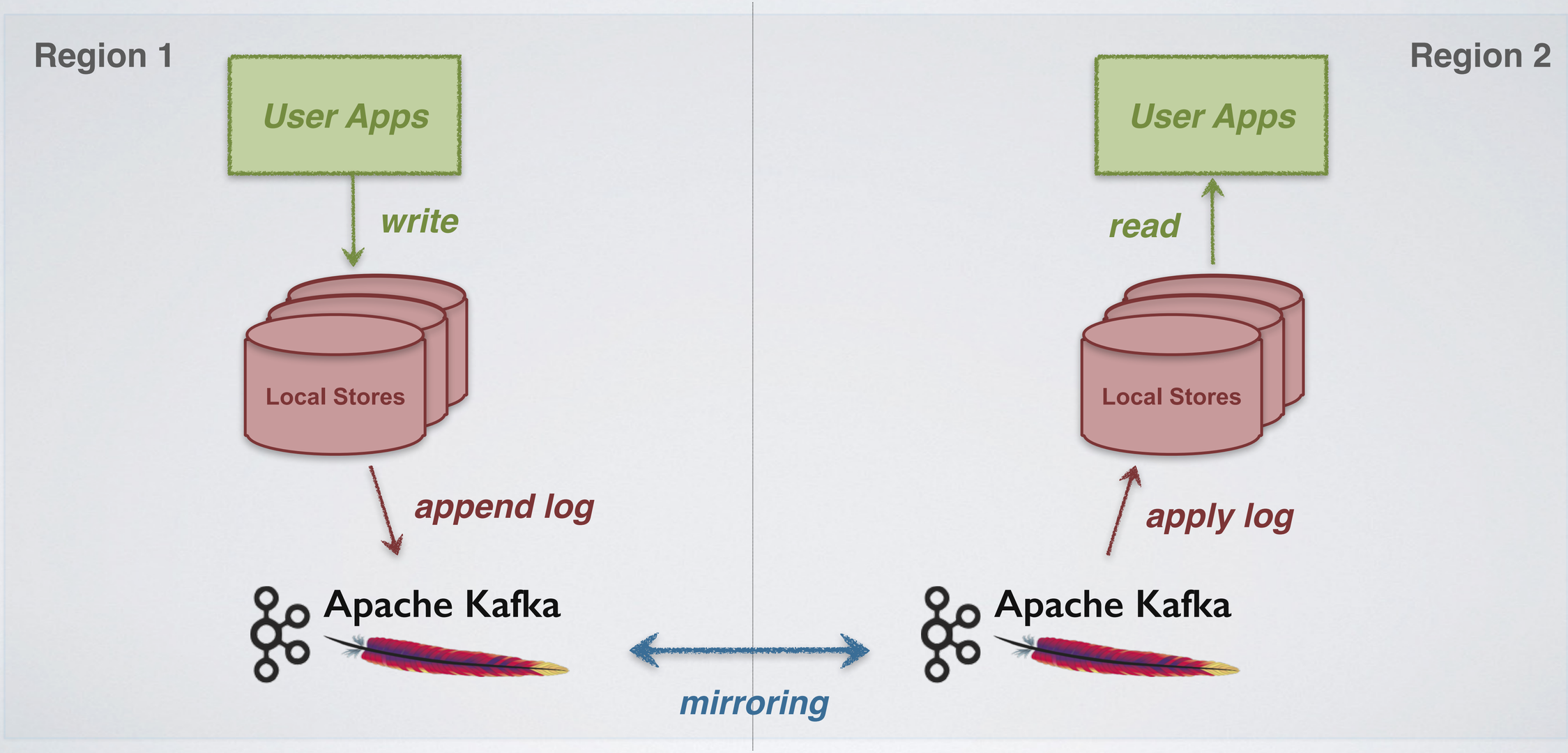
What is Kafka, Really?

a scalable pub-sub messaging system.. *[NetDB 2011]*

a real-time data pipeline.. *[Hadoop Summit 2013]*

a distributed and replicated log.. *[VLDB 2015]*

Example: Data Store Geo-Replication



What is Kafka, Really?

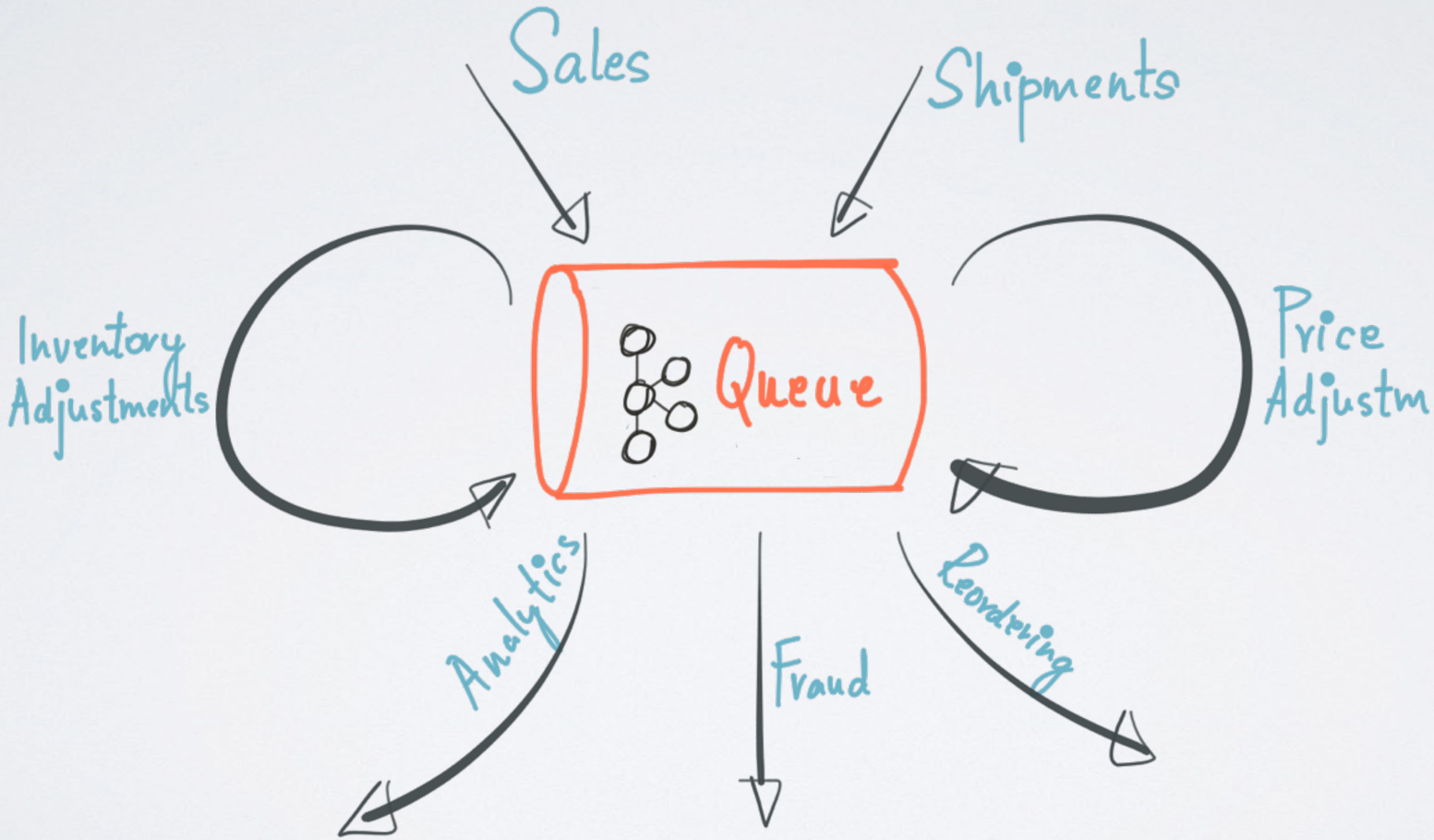
a scalable pub-sub messaging system.. *[NetDB 2011]*

a real-time data pipeline.. *[Hadoop Summit 2013]*

a distributed and replicated log.. *[VLDB 2015]*

a unified data integration stack.. *[CIDR 2015]*

Example: Async. Micro-Services



What is Kafka, Really?

a scalable pub-sub messaging system.. *[NetDB 2011]*

a real-time data pipeline.. *[Hadoop Summit 2013]*

a distributed and replicated log.. *[VLDB 2015]*

a unified data integration stack.. *[CIDR 2015]*

What is Kafka, Really?

a scalable Pub-sub messaging system.. *[NetDB 2011]*

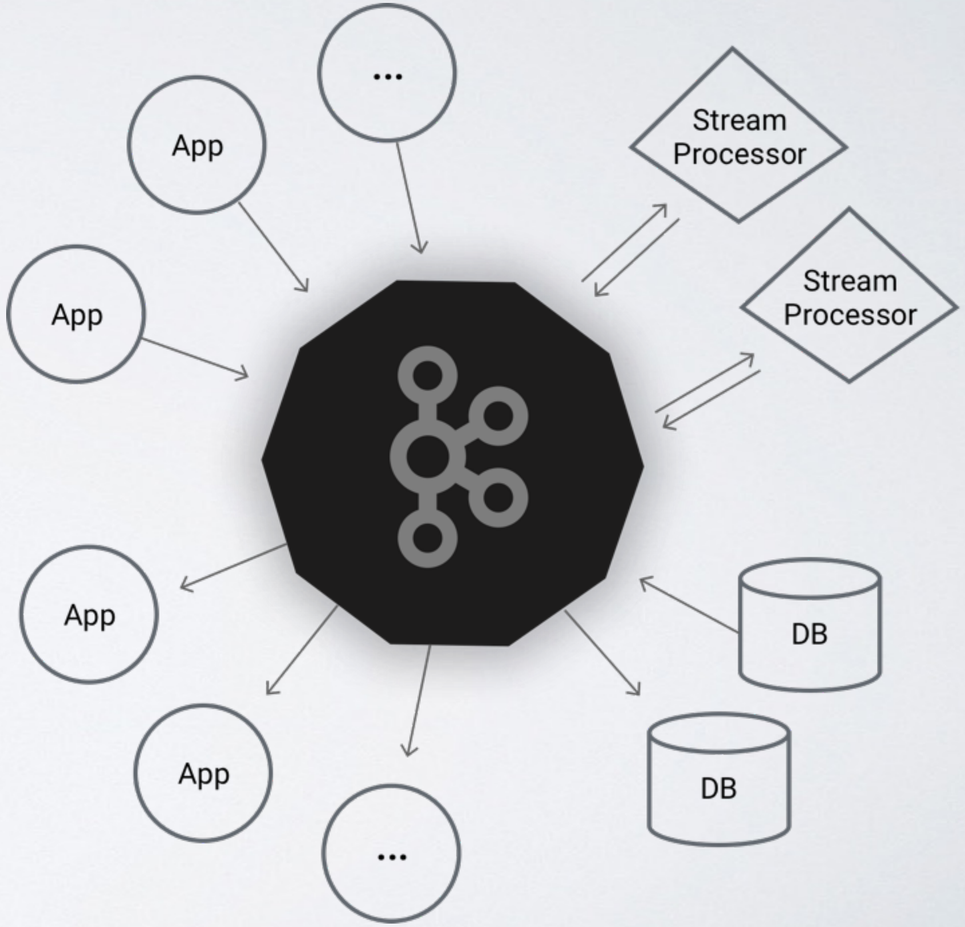
a distributed and replicated log.. *[VLDB 2015]*

a unified data integration stack.. *[CIDR 2015]*

All of them!

Kafka: Streaming Platform

- *Publish / Subscribe*
  - *Move data around as online streams*
- *Store*
  - *"Source-of-truth" continuous data*
- *Process*
  - *React / process data in real-time*



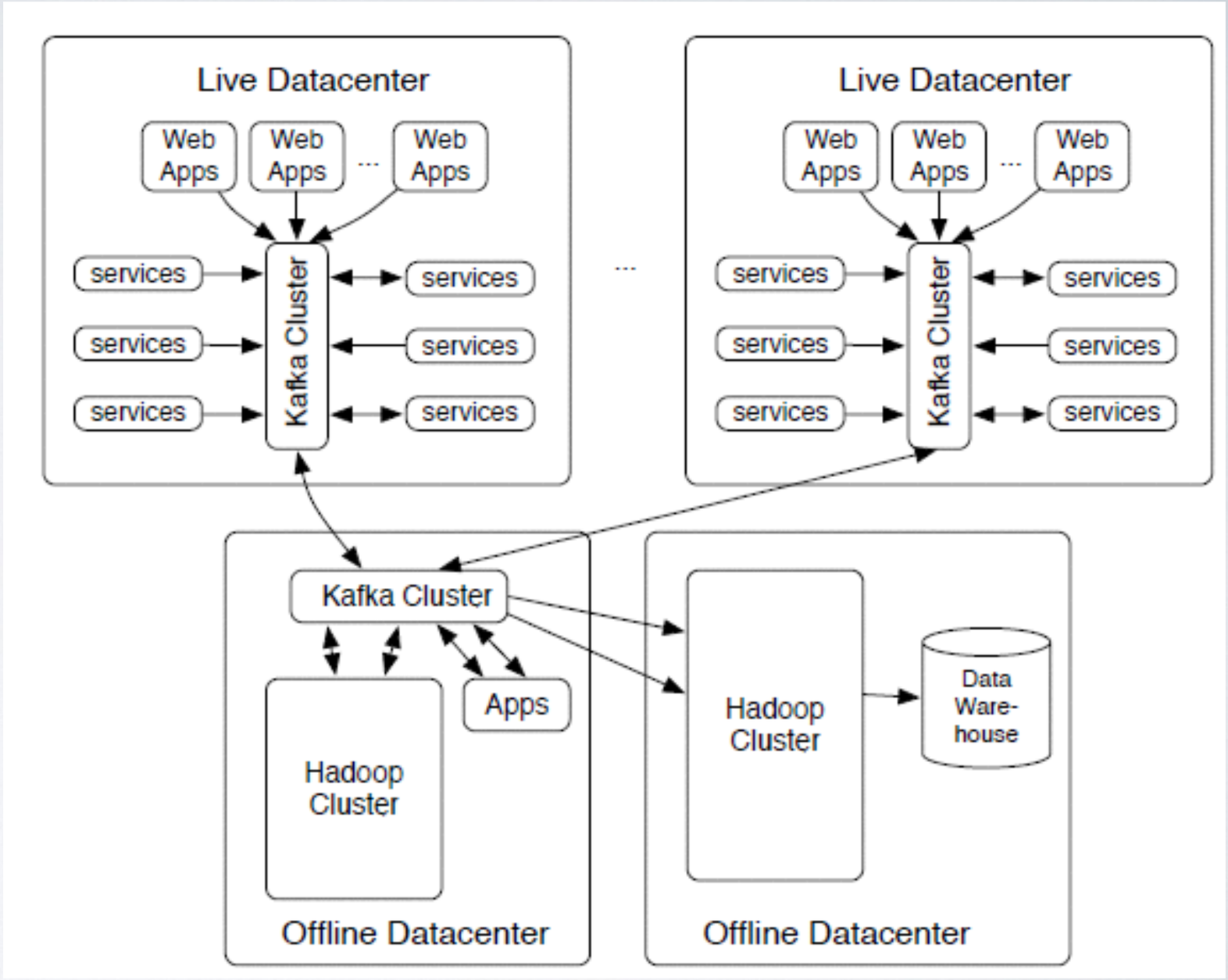
How did we get here?

***Lesson 1**: Build evolvable systems*

Upgrade Your Kafka Cluster is like ..

Kafka @ LI

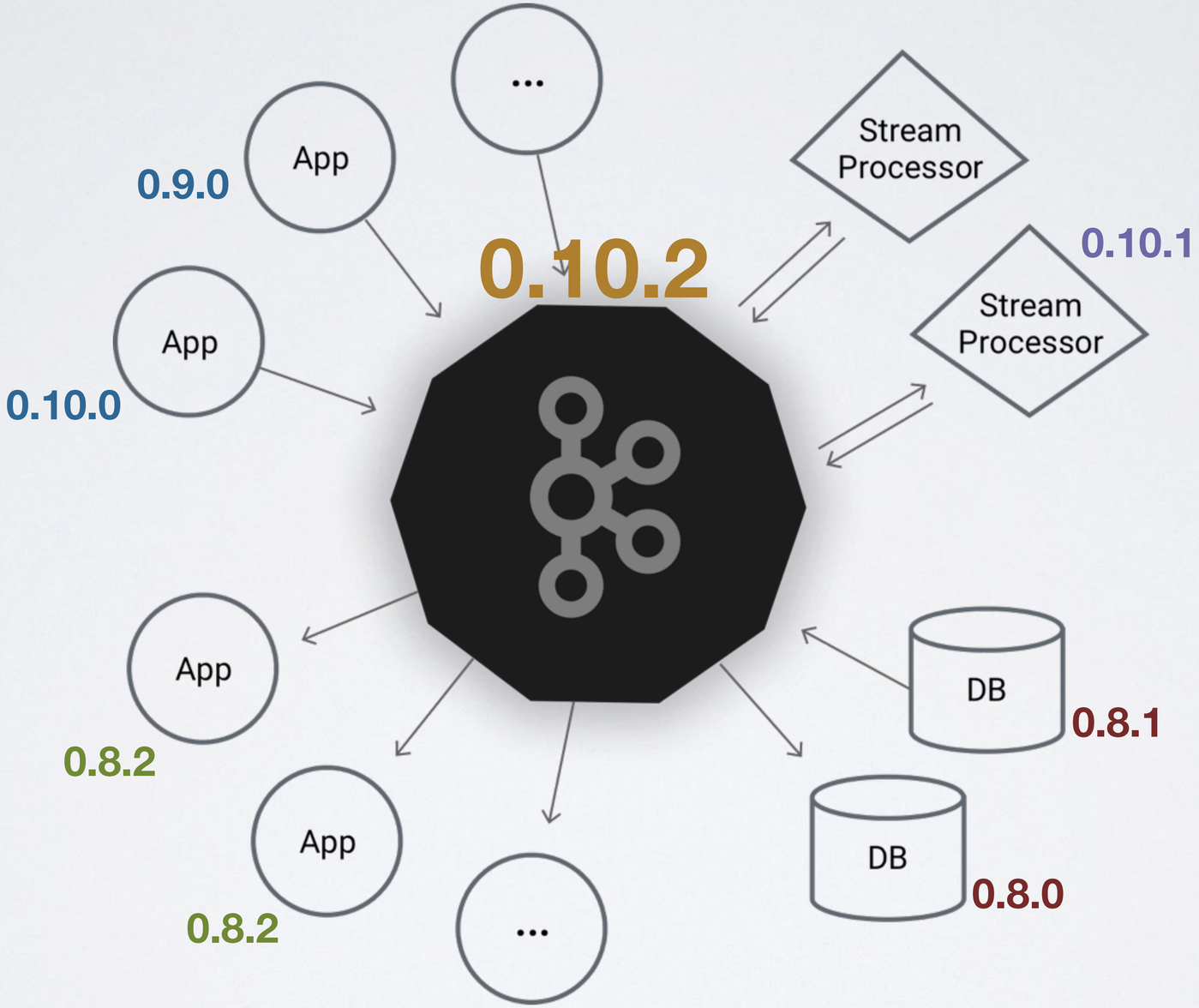
- *Release from trunk*
  - *Push frequency: daily*
- *Staging cluster*
  - *Full traffic of production*
  - *Full monitoring / alerting*
- *Production Ramp-up*
  - *Prepare to roll-back anytime*



Kafka: Evolvable System

- *Zero down-time*
  - *Maintenance outage? No such thing.*
- *All protocols versioned*
  - *Brokers can talk to older versioned clients*
  - *And vice versa since 0.10.2!*
- *One should do no more than rolling bounces*
  - *Staging before production*

Server-Client Compatibility



1.5 Upgrading From Previous Versions

Upgrading from 0.8.x, 0.9.x, 0.10.0.x or 0.10.1.x to 0.10.2.0

0.10.2.0 has wire protocol changes. By following the recommended rolling upgrade plan below, you guarantee no downtime during the upgrade. However, please review the notable changes in 0.10.2.0 before upgrading.

Starting with version 0.10.2, Java clients (producer and consumer) have acquired the ability to communicate with older brokers. Version 0.10.2 clients can talk to version 0.10.0 or newer brokers. However, if your brokers are older than 0.10.0, you must upgrade all the brokers in the Kafka cluster before upgrading your clients. Version 0.10.2 brokers support 0.8.x and newer clients.

For a rolling upgrade:

1. Update server.properties file on all brokers and add the following properties:
    - inter.broker.protocol.version=CURRENT_KAFKA_VERSION (e.g. 0.8.2, 0.9.0, 0.10.0 or 0.10.1).
    - log.message.format.version=CURRENT_KAFKA_VERSION (See potential performance impact following the upgrade for the details on what this configuration does.)
2. Upgrade the brokers one at a time: shut down the broker, update the code, and restart it.
3. Once the entire cluster is upgraded, bump the protocol version by editing inter.broker.protocol.version and setting it to 0.10.2.
4. If your previous message format is 0.10.0, change log.message.format.version to 0.10.2 (this is a no-op as the message format is the same for 0.10.0, 0.10.1 and 0.10.2). If your previous message format version is lower than 0.10.0, do not change log.message.format.version yet - this parameter should only change once all consumers have been upgraded to 0.10.0.0 or later.
5. Restart the brokers one by one for the new protocol version to take effect.
6. If log.message.format.version is still lower than 0.10.0 at this point, wait until all consumers have been upgraded to 0.10.0 or later, then change log.message.format.version to 0.10.2 on each broker and restart them one by one.

Note: If you are willing to accept downtime, you can simply take all the brokers down, update the code and start all of them. They will start with the new protocol by default.

Note: Bumping the protocol version and restarting can be done any time after the brokers were upgraded. It does not have to be immediately after.

Upgrading a 0.10.1 Kafka Streams Application

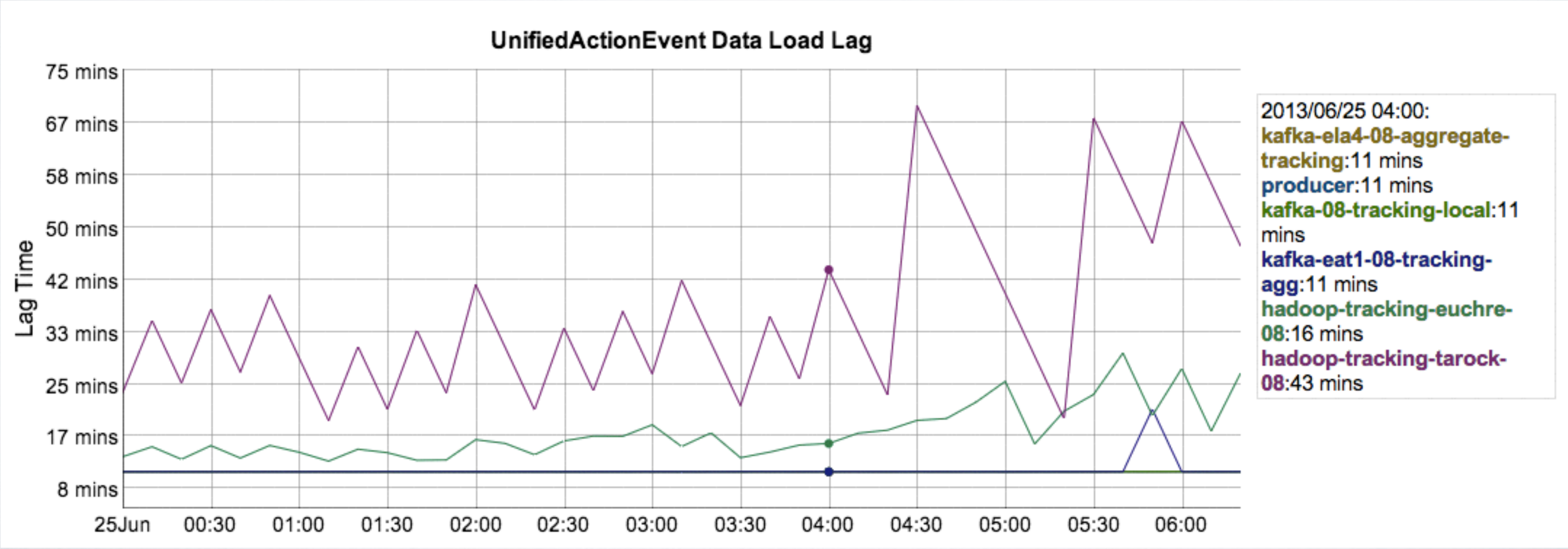
- Upgrading your Streams application from 0.10.1 to 0.10.2 does not require a broker upgrade. A Kafka Streams 0.10.2 application can connect to 0.10.2 and 0.10.1 brokers (it is not possible to connect to 0.10.0 brokers though).
- You need to recompile your code. Just swapping the Kafka Streams library jar file will not work and will break your application.
- If you use a custom (i.e., user implemented) timestamp extractor, you will need to update this code, because the `TimestampExtractor` interface was changed.
- If you register custom metrics, you will need to update this code, because the `StreamsMetric` interface was changed.
- See Streams API changes in 0.10.2 for more details.

Notable changes in 0.10.2.0

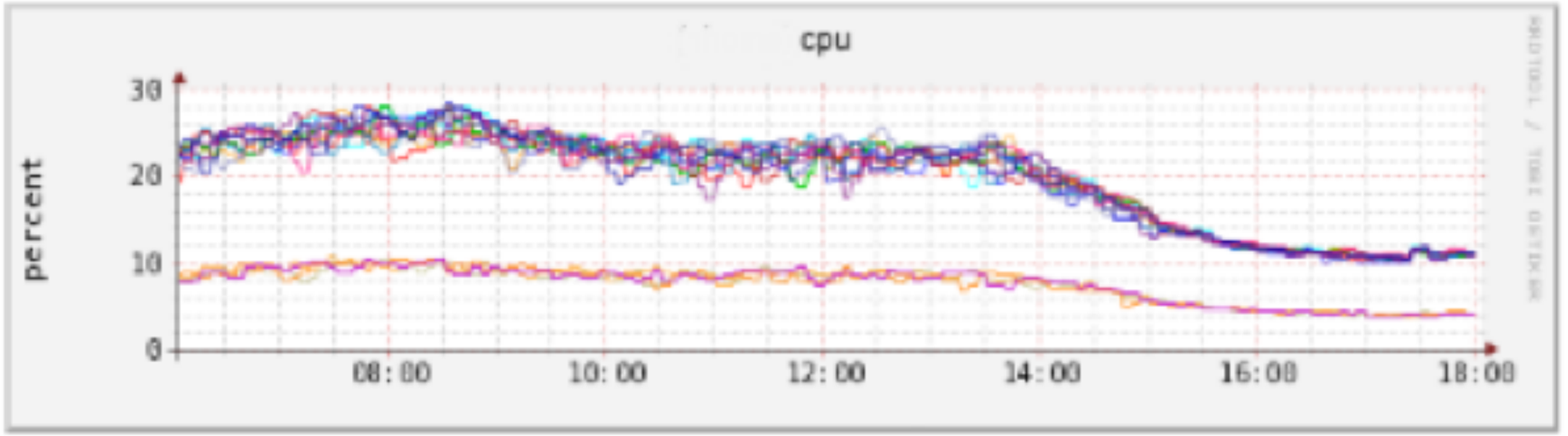
- The Java clients (producer and consumer) have acquired the ability to communicate with older brokers. Version 0.10.2 clients can talk to version 0.10.0 or newer brokers. Note that some features are not available or are limited when older brokers are used.
- Several methods on the Java consumer may now throw `InterruptException` if the calling thread is interrupted. Please refer to the `KafkaConsumer` Javadoc for a more in-depth explanation of this change.
- Java consumer now shuts down gracefully. By default, the consumer waits up to 30 seconds to complete pending requests. A new close API with timeout has been added to `KafkaConsumer` to control the maximum wait time.
- Multiple regular expressions separated by commas can be passed to MirrorMaker with the new Java consumer via the --whitelist option. This makes the behaviour consistent with MirrorMaker when used the old Scala consumer.
- Upgrading your Streams application from 0.10.1 to 0.10.2 does not require a broker upgrade. A Kafka Streams 0.10.2 application can connect to 0.10.2 and 0.10.1 brokers (it is not possible to connect to 0.10.0 brokers though).
- The Zookeeper dependency was removed from the Streams API. The Streams API now uses the Kafka protocol to manage internal topics instead of modifying Zookeeper directly. This eliminates the need for privileges to access Zookeeper directly and "StreamsConfig.ZOOKEEPER_CONFIG" should not be set in the Streams app any more. If the Kafka cluster is secured, Streams apps must have the required security privileges to create new topics.
- Several new fields including "security.protocol", "connections.max.idle.ms", "retry.backoff.ms", "reconnect.backoff.ms" and "request.timeout.ms" were added to StreamsConfig class. User should pay attention to the default values and set these if needed. For more details please refer to 3.5 Kafka Streams Configs.
- The `offsets.topic.replication.factor` broker config is now enforced upon auto topic creation. Internal auto topic creation will fail with a GROUP_COORDINATOR_NOT_AVAILABLE error until the cluster

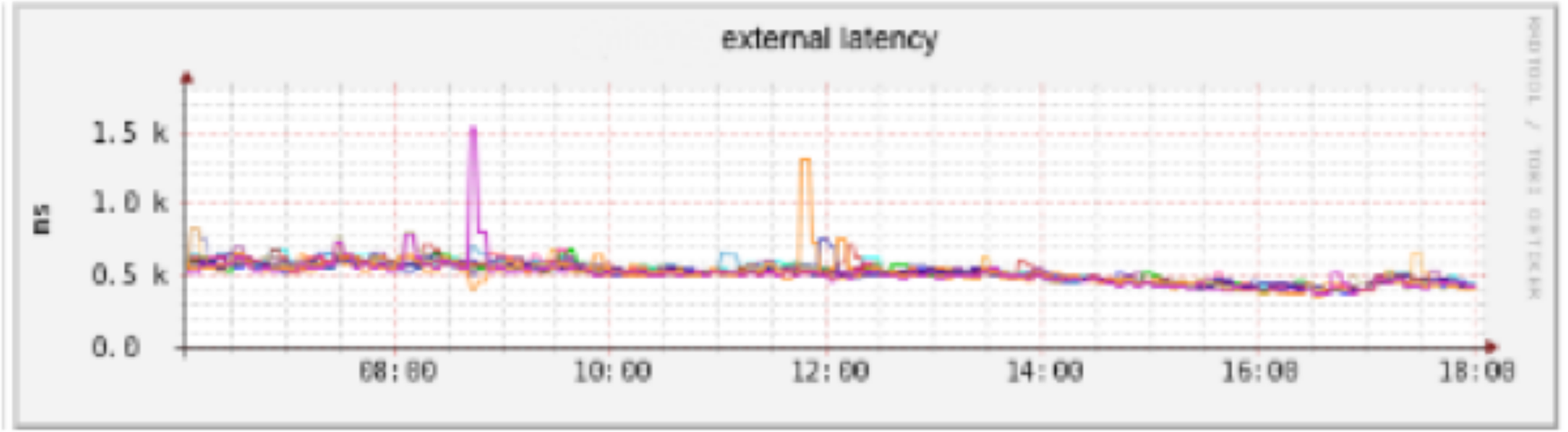
***Lesson 2**: What gets measured gets fixed*

Audit Trail @ LI



ingraphs


cpu
percent
30
20
10
0
08:00
10:00
12:00
14:00
16:00
18:00


external latency
ms
1.5 k
1.0 k
0.5 k
0.0
08:00
10:00
12:00
14:00
16:00
18:00

.. and a LOT of Them

6.6 Monitoring

Kafka uses Yammer Metrics for metrics reporting in both the server and the client. This can be configured to report stats using pluggable stats reporters to hook up to your monitoring system.

The easiest way to see the available metrics is to fire up jconsole and point it at a running kafka client or server; this will allow browsing all metrics with JMX.

We do graphing and alerting on the following metrics:

DESCRIPTION	MBEAN NAME	NORMAL VALUE
Message in rate	kafka.server:type=BrokerTopicMetrics,name=MessagesInPerSec	
Byte in rate	kafka.server:type=BrokerTopicMetrics,name=BytesInPerSec	
Request rate	kafka.network:type=RequestMetrics,name=RequestsPerSec,request={Produce|FetchConsumer|FetchFollower}	
Byte out rate	kafka.server:type=BrokerTopicMetrics,name=BytesOutPerSec	
Log flush rate and time	kafka.log:type=LogFlushStats,name=LogFlushRateAndTimeMs	
# of under replicated partitions (|ISR| < |all replicas|)	kafka.server:type=ReplicaManager,name=UnderReplicatedPartitions	0
Is controller active on broker	kafka.controller:type=KafkaController,name=ActiveControllerCount	only one broker in the cluster should have 1
Leader election rate	kafka.controller:type=ControllerStats,name=LeaderElectionRateAndTimeMs	non-zero when there are broker failures
Unclean leader election rate	kafka.controller:type=ControllerStats,name=UncleanLeaderElectionsPerSec	0
Partition counts	kafka.server:type=ReplicaManager,name=PartitionCount	mostly even across brokers
Leader replica counts	kafka.server:type=ReplicaManager,name=LeaderCount	mostly even across brokers
ISR shrink rate	kafka.server:type=ReplicaManager,name=IsrShrinksPerSec	If a broker goes down, ISR for some of the partitions will shrink. When that broker is up again, ISR will be expanded once the replicas are fully caught up. Other than that, the expected value for both ISR shrink rate and expansion rate is 0.
ISR expansion rate	kafka.server:type=ReplicaManager,name=IsrExpandsPerSec	See above
Max lag in messages btw follower and leader replicas	kafka.server:type=ReplicaFetcherManager,name=MaxLag,clientId=Replica	lag should be proportional to the maximum batch size of a produce request.
Lag in messages per follower replica	kafka.server:type=FetcherLagMetrics,name=ConsumerLag,clientId=([-.\w]+),topic=([-.\w]+),partition=([0-9]+)	lag should be proportional to the maximum batch size of a produce request.
Requests waiting in the producer purgatory	kafka.server:type=DelayedOperationPurgatory,name=PurgatorySize,delayedOperation=Produce	non-zero if ack=-1 is used
Requests waiting in the fetch purgatory	kafka.server:type=DelayedOperationPurgatory,name=PurgatorySize,delayedOperation=Fetch	size depends on fetch.wait.max.ms in the consumer
Request total time	kafka.network:type=RequestMetrics,name=TotalTimeMs,request={Produce|FetchConsumer|FetchFollower}	broken into queue, local, remote and response send time
Time the request waits in the request queue	kafka.network:type=RequestMetrics,name=RequestQueueTimeMs,request={Produce|FetchConsumer|Fe	

HOME
INTRODUCTION
QUICKSTART
USE CASES
DOCUMENTATION
Getting Started
APIs
Configuration
Design
Implementation
Operations
Security
Kafka Connect
Kafka Streams
PERFORMANCE
POWERED BY
PROJECT INFO
ECOSYSTEM
CLIENTS
EVENTS
CONTACT US
APACHE
Download
@apachekafka

Common monitoring metrics for producer/consumer/connect/streams

The following metrics are available on producer/consumer/connector/streams instances. For specific metrics, please see following sections.

METRIC/ATTRIBUTE NAME	DESCRIPTION	MBEAN NAME
connection-close-rate	Connections closed per second in the window.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
connection-creation-rate	New connections established per second in the window.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
network-io-rate	The average number of network operations (reads or writes) on all connections per second.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
outgoing-byte-rate	The average number of outgoing bytes sent per second to all servers.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
request-rate	The average number of requests sent per second.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
request-size-avg	The average size of all requests in the window.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
request-size-max	The maximum size of any request sent in the window.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
incoming-byte-rate	Bytes/second read off all sockets.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
response-rate	Responses received sent per second.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
select-rate	Number of times the I/O layer checked for new I/O to perform per second.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
io-wait-time-ns-avg	The average length of time the I/O thread spent waiting for a socket ready for reads or writes in nanoseconds.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
io-wait-ratio	The fraction of time the I/O thread spent waiting.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
io-time-ns-avg	The average length of time for I/O per select call in nanoseconds.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
io-ratio	The fraction of time the I/O thread spent doing I/O.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)
connection-count	The current number of active connections.	kafka.[producer|consumer|connect]:type=[producer|consumer|connect]-metrics,client-id=([-.\w]+)

Common Per-broker metrics for producer/consumer/connect/streams

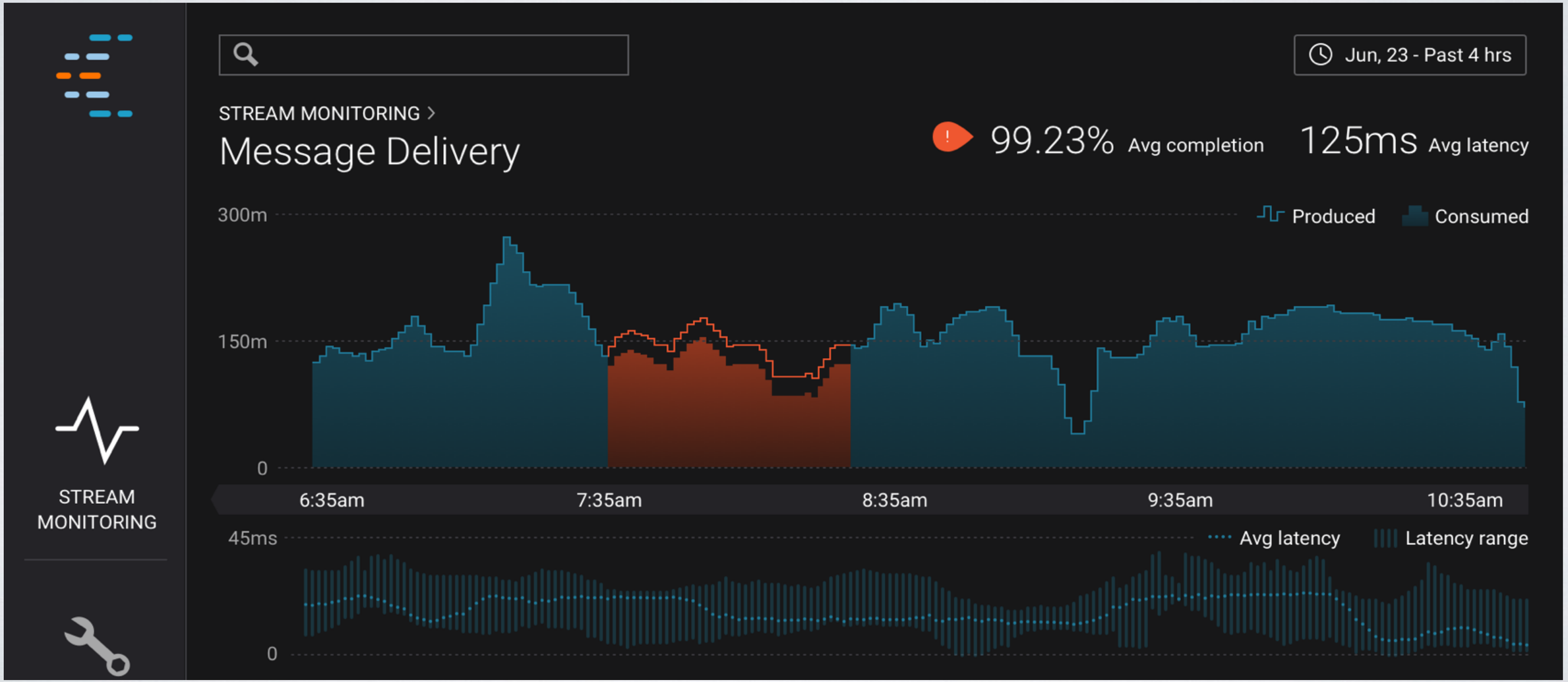
The following metrics are available on producer/consumer/connector/streams instances. For specific metrics, please see following sections.

METRIC/ATTRIBUTE NAME	DESCRIPTION	MBEAN NAME
outgoing-byte-rate	The average number of outgoing bytes sent per second for a node.	kafka.producer:type=[consumer|producer|connect]-node-metrics,client-id=([-.\w]+),node-id=([0-9]+)
request-rate	The average number of requests sent per second for a node.	kafka.producer:type=[consumer|producer|connect]-node-metrics,client-id=([-.\w]+),node-id=([0-9]+)
request-size-avg	The average size of all requests in the window for a node.	kafka.producer:type=[consumer|producer|connect]-node-metrics,client-id=([-.\w]+),node-id=([0-9]+)
request-size-max	The maximum size of any request sent in the window for a node.	kafka.producer:type=[consumer|producer|connect]-node-metrics,client-id=([-.\w]+),node-id=([0-9]+)
incoming-byte-rate	The average number of responses received per second for a node.	kafka.producer:type=[consumer|producer|connect]-node-metrics,client-id=([-.\w]+),node-id=([0-9]+)
request-latency-avg	The average request latency in ms for a node.	kafka.producer:type=[consumer|producer|connect]-node-metrics,client-id=([-.\w]+),node-id=([0-9]+)
request-latency-max	The maximum request latency in ms for a node.	kafka.producer:type=[consumer|producer|connect]-node-metrics,client-id=([-.\w]+),node-id=([0-9]+)
response-rate	Responses received sent per second for a node.	kafka.producer:type=[consumer|producer|connect]-node-metrics,client-id=([-.\w]+),node-id=([0-9]+)

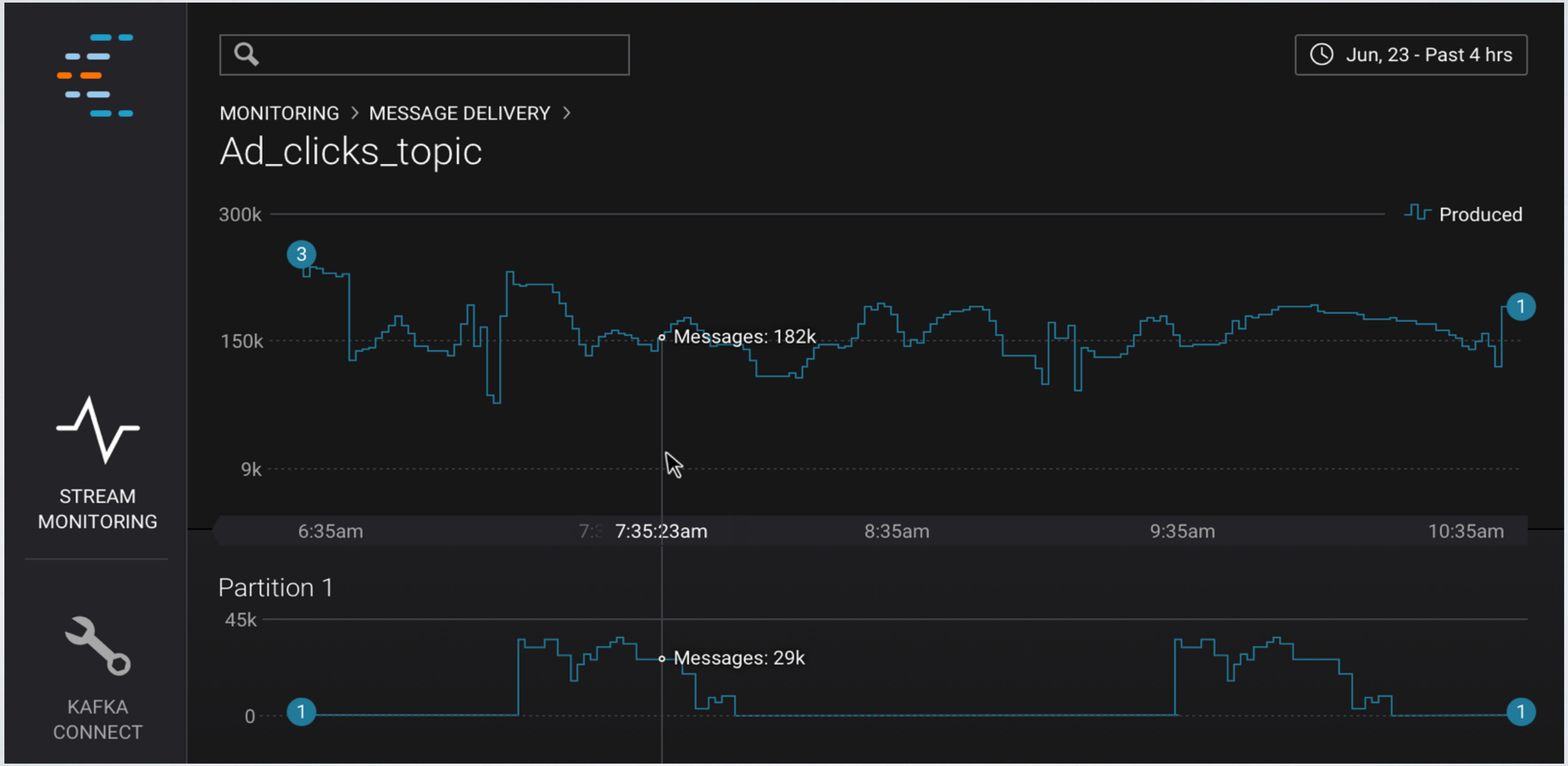
Metrics Reporter

- *JMXReporter*
  - *Included in AK: sensor / metrics -> mbean / attributes*
- *KafkaReporter*
  - *Send metrics back to Kafka!*
- *More..*
  - *Ganglia, Graphite, statsd ..*

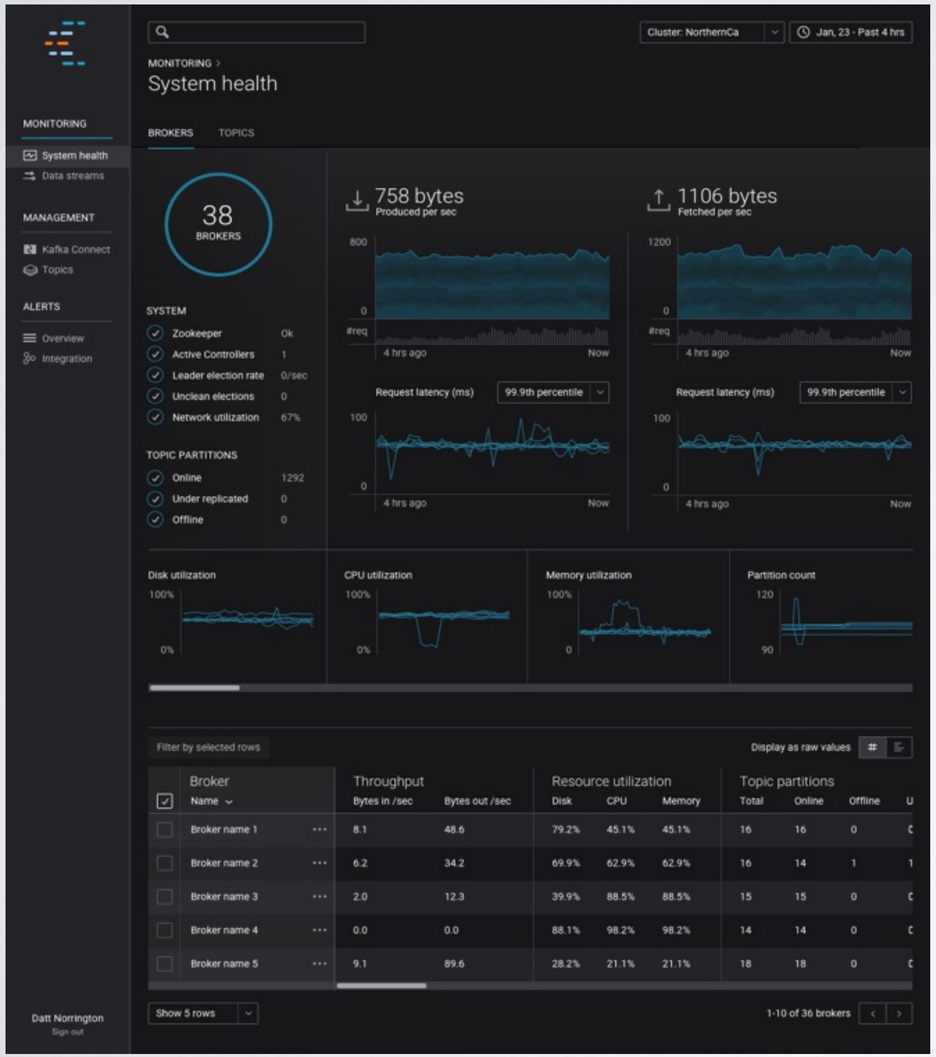
Confluent Enterprise: Delivery Tracking

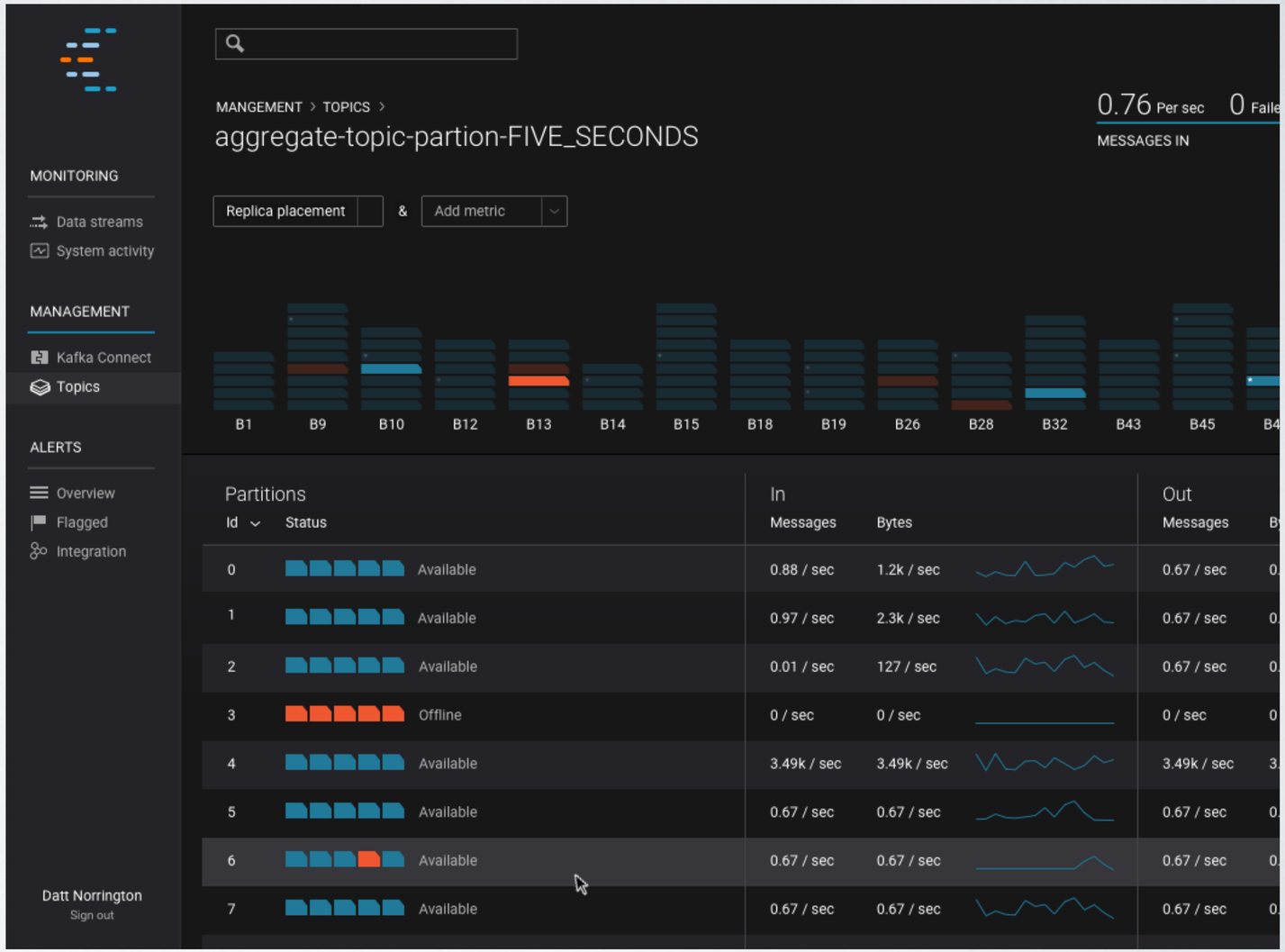


Confluent Enterprise: Delivery Tracking



Confluent Enterprise: Cluster Health





***Lesson 3**: APIs stay forever*

The Story of **KAFKA-1481**

Hey, we should stop allowing dashes / underscores in MBean name since they care used in hostnames / topics / etc.

Makes sense, let's do this!

DONE! (a few lines of key changes)

The Story of **KAFKA-1481**

The Story of **KAFKA-1481**

The Story of **KAFKA-1481**

OMG what happened?

We need to tell our SRE to change their monitoring metrics names now..

That's N cluster on M data centers ..

Kafka Improvement Proposals

Created by Jay Kreps, last modified by Matthias J. Sax yesterday at 09:57 PM

This page describes a proposed Kafka Improvement Proposal (KIP) process for proposing a major change to Kafka.

To create your own KIP, click on "Create" on the header and choose "KIP-Template" other than "Blank page".

Purpose

We want to make Kafka a core architectural component for users. We also support a large number of integrations with other tools, systems, and clients. Keeping this kind of usage health requires a high level of compatibility between releases — core architectural elements can't break compatibility or shift functionality from release to release. As a result each new major feature or public api has to be done in a way that we can stick with it going forward.

This means when making this kind of change we need to think through what we are doing as best we can prior to release. And as we go forward we need to stick to our decisions as much as possible. All technical decisions have pros and cons so it is important we capture the thought process that lead to a decision or design to avoid flip-flopping needlessly.

Hopefully we can make these proportional in effort to their magnitude — small changes should just need a couple brief paragraphs, whereas large changes need detailed design discussions.

This process also isn't meant to discourage incompatible changes — proposing an incompatible change is totally legitimate. Sometimes we will have made a mistake and the best path forward is a clean break that cleans things up and gives us a good foundation going forward. Rather this is intended to avoid accidentally introducing half thought-out interfaces and protocols that cause needless heartburn when changed. Likewise the definition of "compatible" is itself squishy: small details like which errors are thrown when are clearly part of the contract but may need to change in some circumstances, likewise performance isn't part of the public contract but dramatic changes may break use cases. So we just need to use good judgement about how big the impact of an incompatibility will be and how big the payoff is.

What is considered a "major change" that needs a KIP?

What is considered a "major change" that needs a KIP?

Any of the following should be considered a major change:

- Any major new feature, subsystem, or piece of functionality
- Any change that impacts the public interfaces of the project

What are the "public interfaces" of the project?

All of the following are public interfaces that people build around:

- Binary log format
- The network protocol and api behavior
- Any class in the public packages under clients
    - org/apache/kafka/common/serialization
    - org/apache/kafka/common
    - org/apache/kafka/common/errors
    - org/apache/kafka/clients/producer
    - org/apache/kafka/clients/consumer (eventually, once stable)
- Configuration, especially client configuration
- Monitoring
- Command line tools and arguments

Not all compatibility commitments are the same. We need to spend significantly more time on log format and protocol as these break code in lots of clients, cause downtime releases, etc. Public apis are next as they cause people to rebuild code and lead to compatibility issues in large multi-dependency projects (which end up requiring multiple incompatible versions). Configuration, monitoring, and command line tools can be faster and looser — changes here will break monitoring dashboards and require a bit of care during upgrades but aren't a huge burden.

For the most part monitoring, command line tool changes, and configs are added with new features so these can be done with a single KIP.

What should be included in a KIP?

A KIP should contain the following sections:

- **Motivation**: describe the problem to be solved
- **Proposed Change**: describe the new thing you want to do. This may be fairly extensive and have large subsections of its own. Or it may be a few sentences, depending on the scope of the change.
- **New or Changed Public Interfaces**: impact to any of the "compatibility commitments" described above. We want to call these out in particular so everyone thinks about them.
- **Migration Plan and Compatibility**: if this feature requires additional support for a no-downtime upgrade describe how that will work
- **Rejected Alternatives**: What are the other alternatives you considered and why are they worse? The goal of this section is to help people understand why this is the best solution now, and also to prevent churn in the future when old alternatives are reconsidered.

Who should initiate the KIP?

What is considered a "major change" that needs a KIP?

Any of the following should be considered a major change:

- Any major new feature, subsystem, or piece of functionality
- Any change that impacts the public interfaces of the project

What are the "public interfaces" of the project?

All of the following are public interfaces that people build around:

- Binary log format
- The network protocol and api behavior
- Any class in the public packages under clients
    - org/apache/kafka/common/serialization
    - org/apache/kafka/common
    - org/apache/kafka/common/errors
    - org/apache/kafka/clients/producer
    - org/apache/kafka/clients/consumer (eventually, once stable)
- Configuration, especially client configuration
- Monitoring
- Command line tools and arguments

Not all compatibility commitments are the same. We need to spend significantly more time on log format and protocol as these break code in lots of clients, cause downtime releases, etc. Public apis are next as they cause people to rebuild code and lead to compatibility issues in large multi-dependency projects (which end up requiring multiple incompatible versions). Configuration, monitoring, and command line tools can be faster and looser — changes here will break monitoring dashboards and require a bit of care during upgrades but aren't a huge burden.

For the most part monitoring, command line tool changes, and configs are added with new features so these can be done with a single KIP.

What should be included in a KIP?

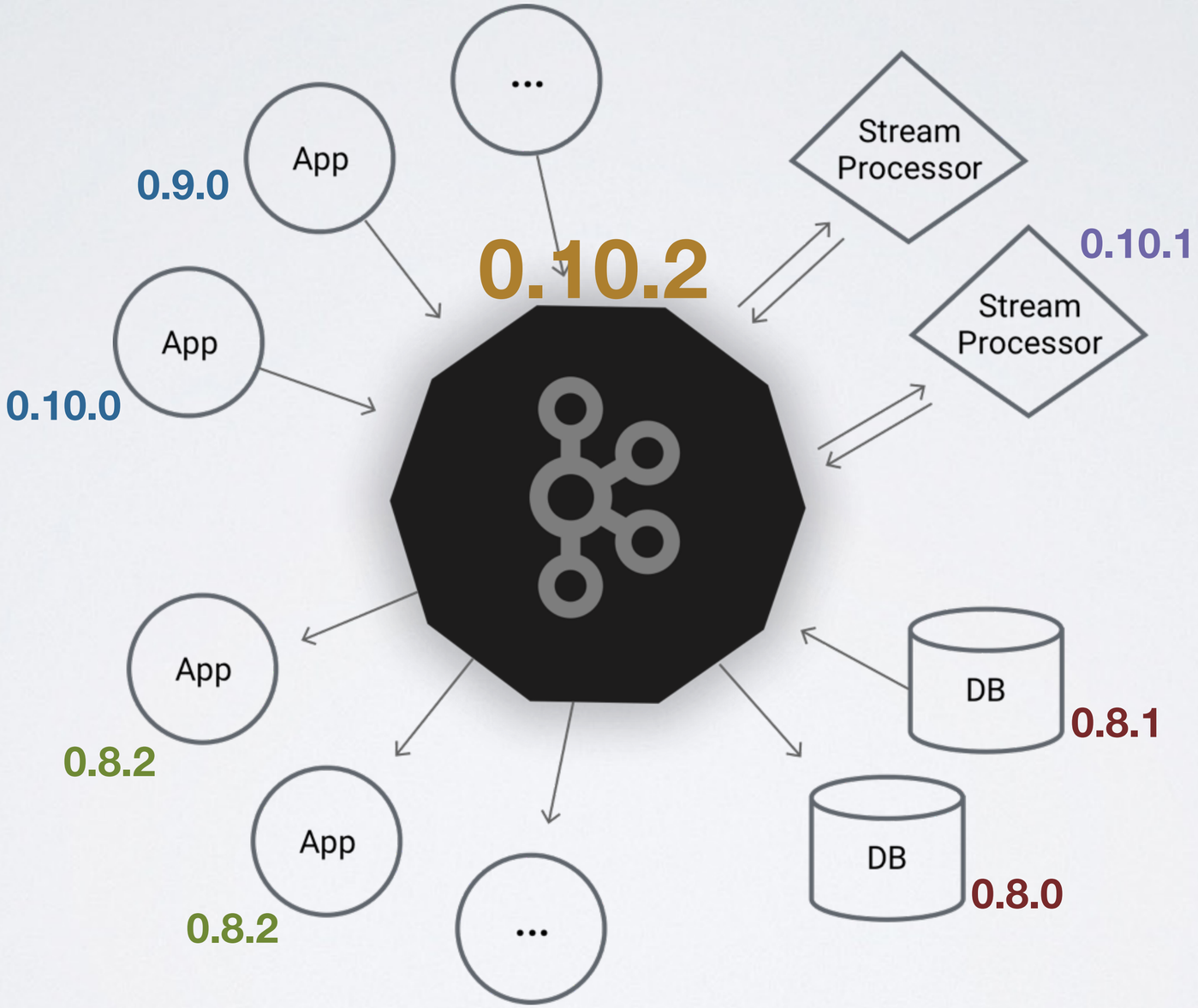
A KIP should contain the following sections:

- **Motivation**: describe the problem to be solved
- **Proposed Change**: describe the new thing you want to do. This may be fairly extensive and have large subsections of its own. Or it may be a few sentences, depending on the scope of the change.
- **New or Changed Public Interfaces**: impact to any of the "compatibility commitments" described above. We want to call these out in particular so everyone thinks about them.
- **Migration Plan and Compatibility**: if this feature requires additional support for a no-downtime upgrade describe how that will work
- **Rejected Alternatives**: What are the other alternatives you considered and why are they worse? The goal of this section is to help people understand why this is the best solution now, and also to prevent churn in the future when old alternatives are reconsidered.

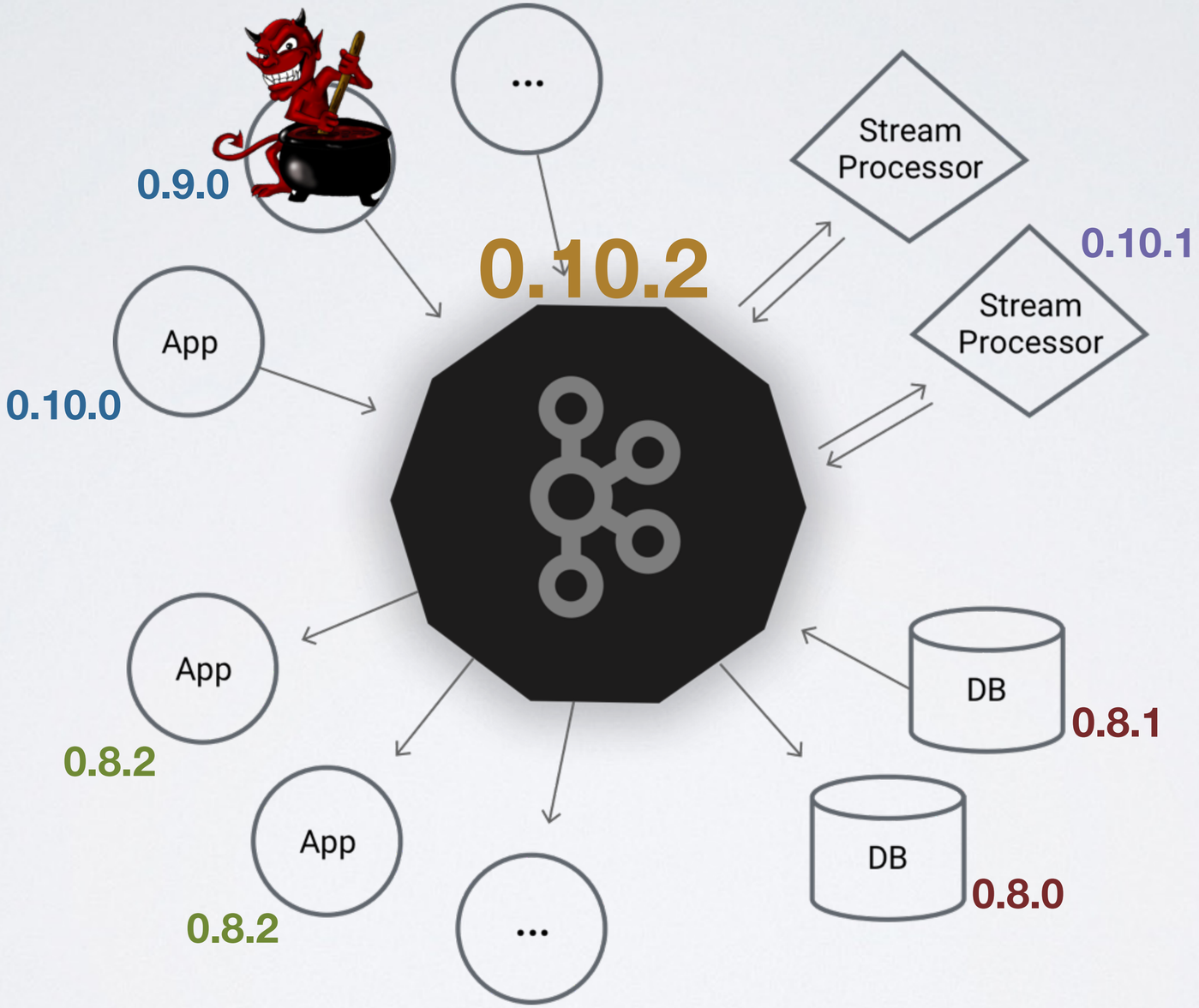
Who should initiate the KIP?

***Lesson 4**: Service needs gatekeepers*

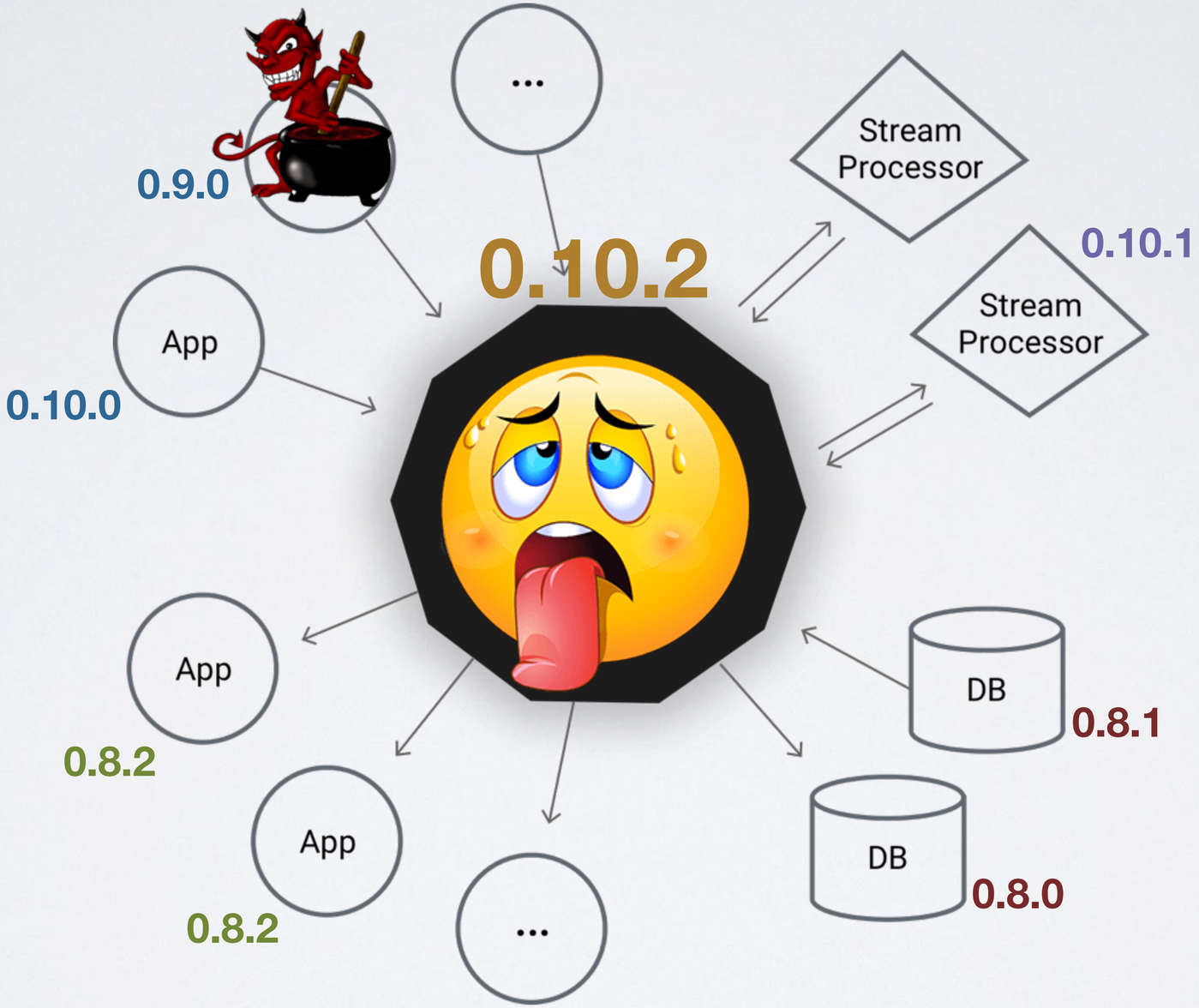
One naughty client can bother everyone ..



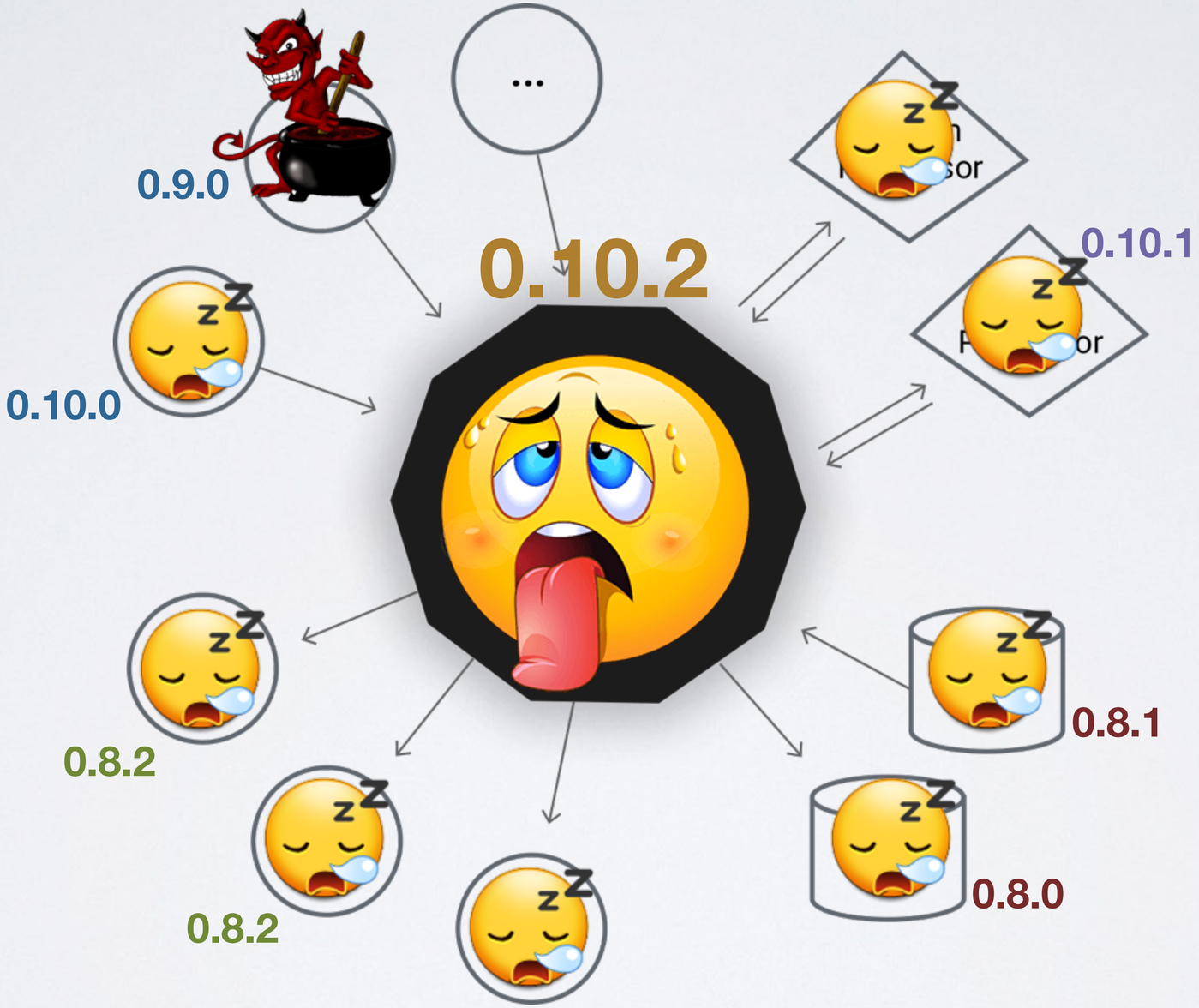
One naughty client can bother everyone ..



One naughty client can bother everyone ..



One naughty client can bother everyone ..

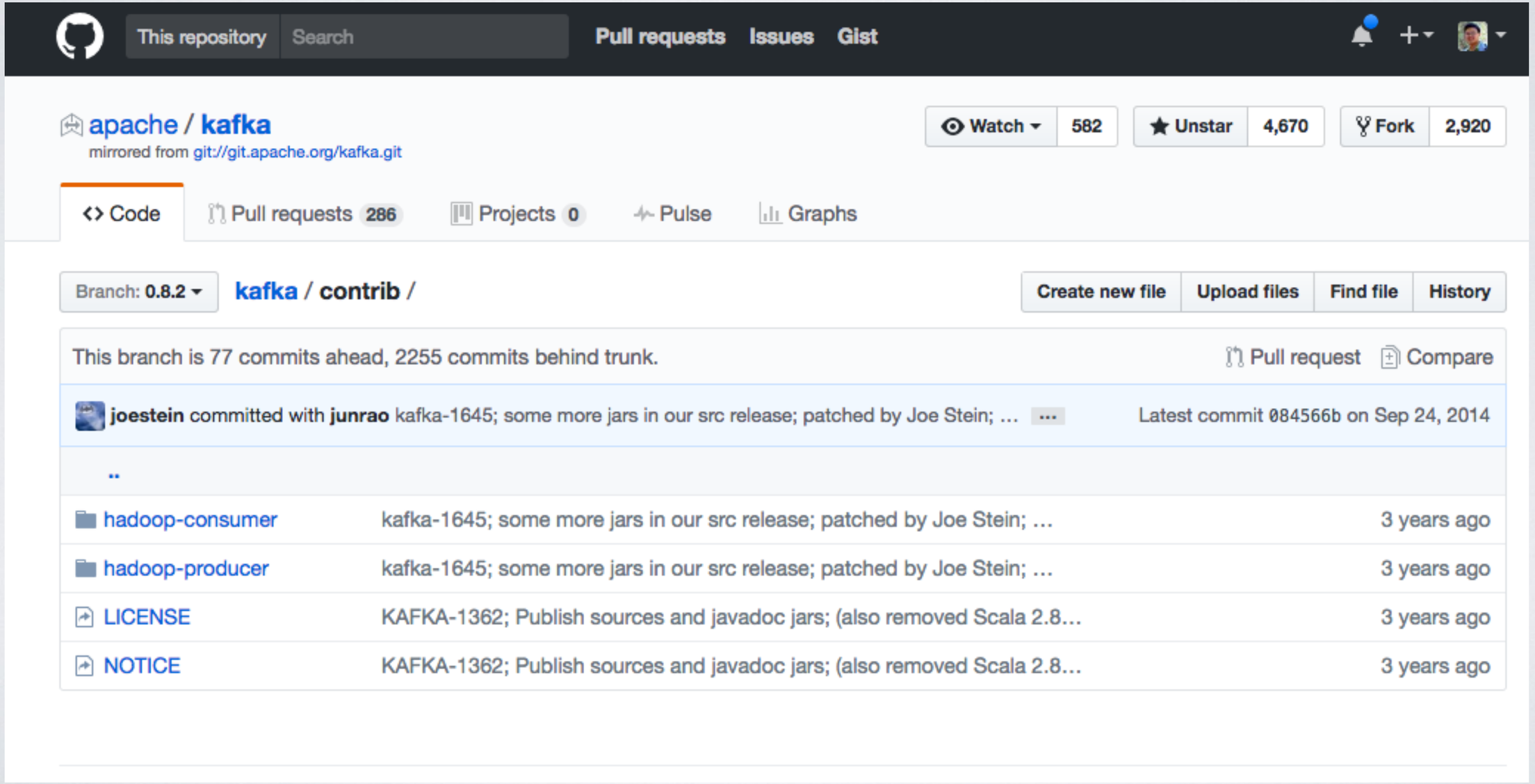


Multi-tenancy Services

- *Security from ground up* *[Release 0.9.0+]*
  - *Authentication*
  - *Authorization*
- *Resources under control* *[Release 0.9.0+]*
  - *Quota on bytes rate / request rate on clientId / GroupId etc*
  - *Quota on CPU resources*

***Lesson 5**: Ecosystems are the key*

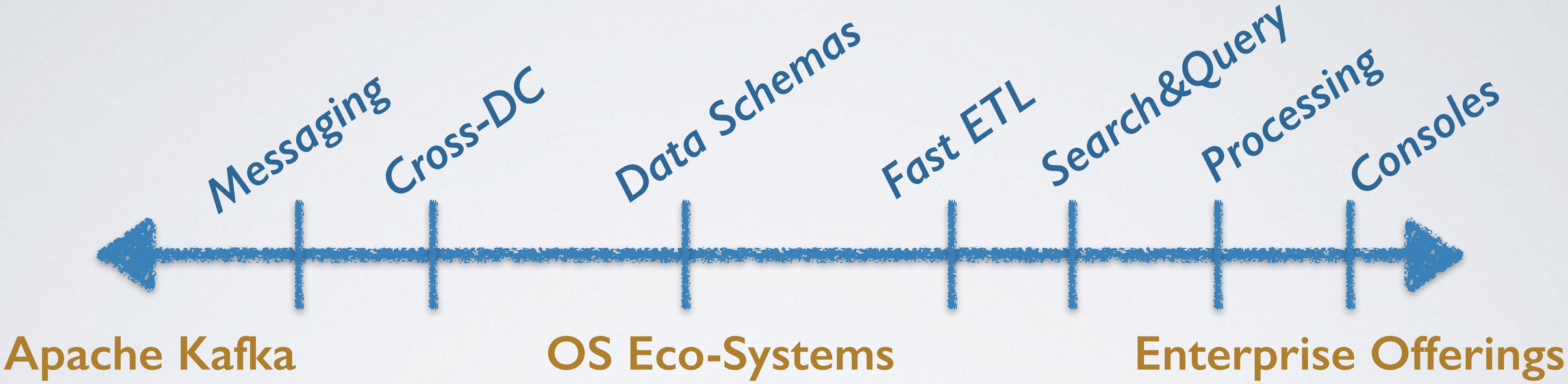
Remember Hadoop-Producer/Consumer?

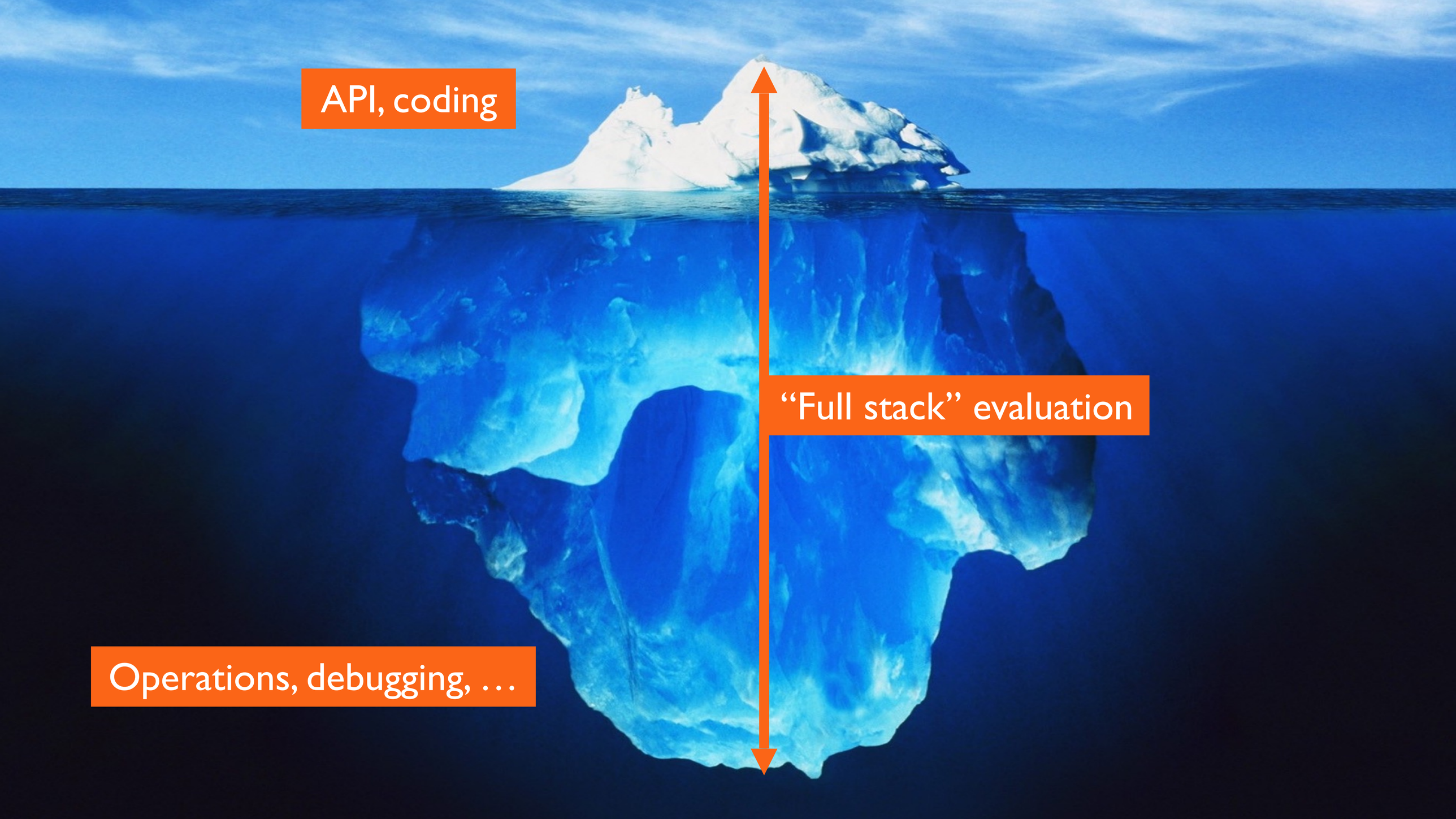


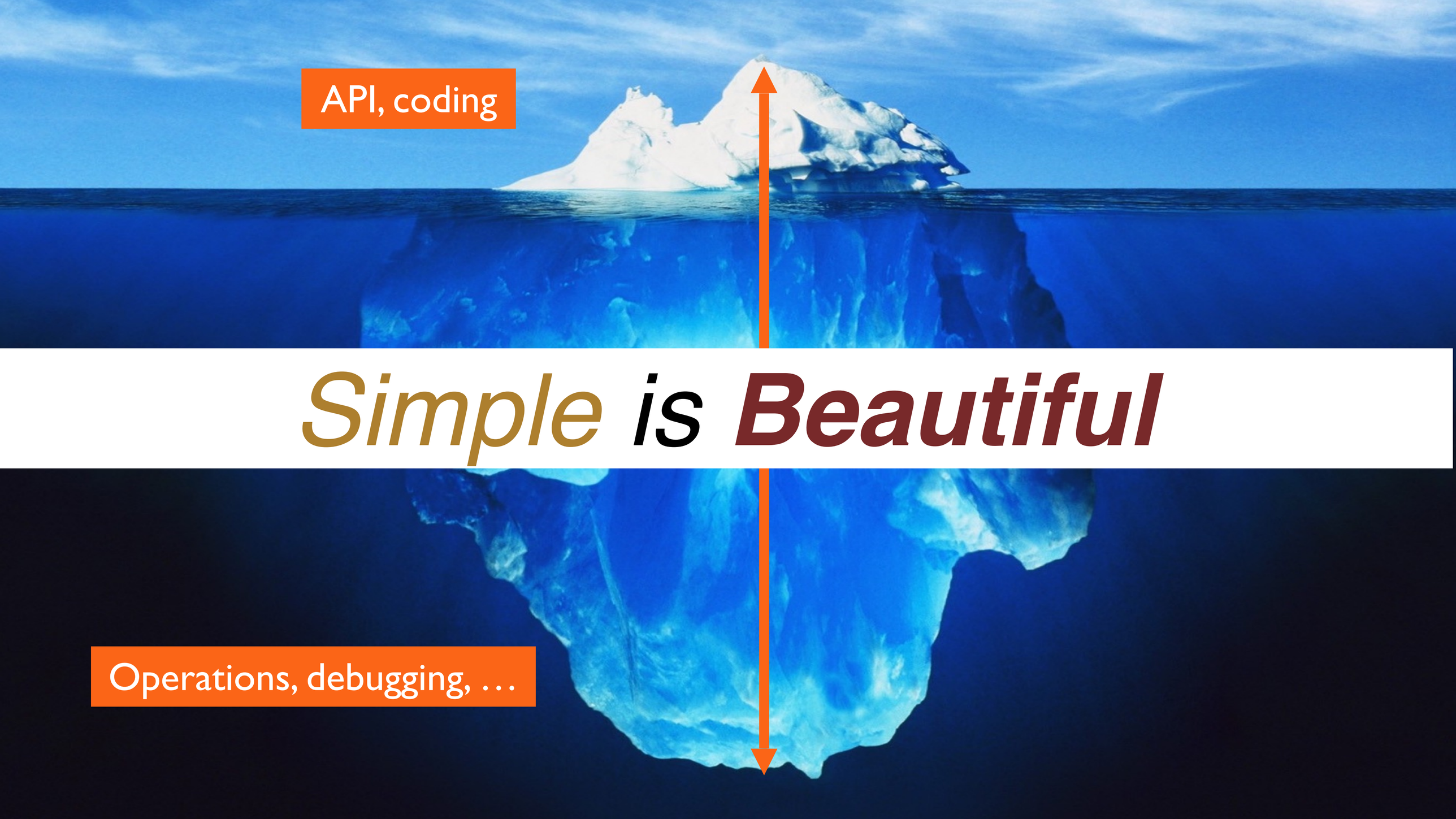
What Should Go into *AK* ?

- *Container Image?*
- *Kafka Manager GUI?*
- *REST Proxy APIs?*
- *Operation Tooling?*
- *Other Language Clients?*
- *Schema Registry?*
- *Hadoop / etc Integration?*
- *Stream Processing?*

Layered Architecture, not Monolithic




API, coding
“Full stack” evaluation
Operations, debugging, …


API, coding
Simple is Beautiful
Operations, debugging, …

APACHE kafka
A distributed streaming platform

How to contribute

We are always very happy to have contributions, whether for trivial cleanups or big new features.

If you don't know Java or Scala you can still contribute to the project. An important area is the clients. We want to have high quality, well documented clients for each programming language. These, as well as the surrounding ecosystem of integration tools that people use with Kafka™, are critical aspects of the project.

Nor is code the only way to contribute to the project. We strongly value documentation and gladly accept improvements to the documentation.

CONTRIBUTING A CODE CHANGE

To submit a change for inclusion, please do the following:

- If the change is non-trivial please include some unit tests that cover the new functionality.
- If you are introducing a completely new feature or API it is a good idea to start a wiki and get consensus on the basic design first.
- Make sure you have observed the recommendations in the style guide.
- Follow the detailed instructions in Contributing Code Changes.
- Note that if the change is related to user-facing protocols / interface / configs, etc, you need to make the corresponding change on the documentation as well. For wiki page changes feel free to edit the page content directly (you may need to contact us to get the permission first if it is your first time to edit on wiki); website docs live in the code repo under `docs` so that changes to that can be done in the same PR as changes to the code. Website doc change instructions are given below.
- It is our job to follow up on patches in a timely fashion. Nag us if we aren't doing our job (sometimes we drop things).

CONTRIBUTING A CHANGE TO THE WEBSITE

To submit a change for inclusion please do the following:

- Follow the instructions in Contributing Website Changes.
- It is our job to follow up on patches in a timely fashion. Nag us if we aren't doing our job (sometimes we drop things). If the patch needs improvement, the reviewer will mark the jira back to "In Progress" after reviewing.

FINDING A PROJECT TO WORK ON

The easiest way to get started working with the code base is to pick up a really easy JIRA and work on that. This will help you get familiar with the code base, build system, review process, etc. We flag these kind of starter bugs here.

Please contact us to be added the contributor list. After that you can assign yourself to the JIRA ticket you have started working on so others will notice.

Once you have gotten through the basic process of checking in code, you may want to move on to a more substantial project. We try to curate this kind of project as well, and you can find these here.

BECOMING A COMMITTER

We are always interested in adding new contributors. What we look for is a series of contributions, good taste, and an ongoing interest in the project. If you are interested in becoming a committer, let one of the existing committers know and they can help guide you through the process.

Search

Save as

Share | Export | Tools

Kafka | Type: All | Open | Assignee: All | Contains text | More | Advanced

Label: newbie, newbie++, beg...

1–50 of 68

Columns

T	Key	Summary	Assignee	Reporter	P	Status	Created	Updated	Labels
	KAFKA-1676	Ability to cancel replica reassignment in progress	*Unassigned*	Ryan Berdeen		OPEN	06/Oct/14	22/Jan/17	newbie
	KAFKA-4559	Add a site search bar on the Web site	*Unassigned*	Guozhang Wang		OPEN	19/Dec/16	19/Dec/16	newbie
	KAFKA-4722	Add application.id to StreamThread name	Sharad	Steven Schlansker		OPEN	01/Feb/17	28/Feb/17	beginner easyfix newbie
	KAFKA-1900	KAFKA-1722 / Add documentation on usage of code coverage in the project and how it works	Ashish Singh	Ashish Singh		OPEN	25/Jan/15	23/Jan/17	newbie
	KAFKA-4346	Add foreachValue method to KStream	Xavier Léauté	Xavier Léauté		OPEN	25/Oct/16	09/Jan/17	needs-kip newbie
	KAFKA-4594	Annotate integration tests and provide gradle build targets to run subsets of tests	*Unassigned*	Ewen Cheslack-Postava		OPEN	04/Jan/17	05/Jan/17	newbie
	KAFKA-205	Audit swallowing of exceptions/throwables	*Unassigned*	Chris Burroughs		OPEN	16/Nov/11	17/Oct/16	newbie
	KAFKA-4830	Augment KStream.print() to allow users pass in extra parameters in the printed string	*Unassigned*	Guozhang Wang		OPEN	01/Mar/17	01/Mar/17	newbie
	KAFKA-3729	Auto-configure non-default SerDes passed alongside the topology builder	Bharat Viswanadham	Fred Patton		OPEN	18/May/16	01/Feb/17	api newbie
	KAFKA-4566	Can't Symlink to Kafka bins	Akhilesh Naidu	Stephane Maarek		OPEN	21/Dec/16	29/Jan/17	newbie
	KAFKA-1506	Cancel "kafka-reassign-partitions" Job	*Unassigned*	Paul Lung		OPEN	23/Jun/14	16/Oct/14	newbie++
	KAFKA-1599	Change preferred replica election admin command to handle large clusters	Abhishek Nigam	Todd Palino		OPEN	15/Aug/14	11/Oct/16	newbie++
	KAFKA-2111	Command Line Standardization - Add Help Arguments & List Required Fields	*Unassigned*	Matt Warhaftig		OPEN	08/Apr/15	17/Oct/16	newbie
	KAFKA-3745	Consider adding join key to ValueJoiner interface	Sreepathi Prasanna	Greg Fodor		OPEN	22/May/16	13/Jan/17	api needs-kip newbie
	KAFKA-2526	Console Producer / Consumer's serde config is not working	Mayuresh Gharat	Guozhang Wang		OPEN	08/Sep/15	23/Jan/17	newbie
	KAFKA-1935	KAFKA-1326 / Consumer should use a separate socket for Coordinator connection	*Unassigned*	Guozhang Wang		OPEN	09/Feb/15	22/Feb/17	newbie
	KAFKA-1447	Controlled shutdown deadlock when trying to send state updates	*Unassigned*	Sam Meder		OPEN	12/May/14	08/Feb/15	newbie++
	KAFKA-2244	Document Kafka metrics configuration properties	Grant Henke	Stevo Slavic		OPEN	02/Jun/15	12/Aug/15	newbie
	KAFKA-4218	Enable access to key in ValueTransformer	*Unassigned*	Elias Levy		OPEN	24/Sep/16	02/Feb/17	api needs-kip newbie
	KAFKA-1056	Evenly Distribute Intervals in OffsetIndex	Guozhang Wang	Guozhang Wang		OPEN	16/Sep/13	23/Jan/17	newbie++ newbiee
	KAFKA-4772	Exploit #peek to implement #print() and other methods	*Unassigned*	Matthias J. Sax		OPEN	16/Feb/17	01/Mar/17	beginner newbie
	KAFKA-4304	Extend Interactive Queries for return latest update timestamp per key	*Unassigned*	Matthias J. Sax		OPEN	14/Oct/16	22/Jan/17	newbie++
	KAFKA-4831	Extract WindowedSerde to public APIs	*Unassigned*	Guozhang Wang		OPEN	01/Mar/17	01/Mar/17	newbie user-experience
	KAFKA-1850	Failed reassignment leads to additional replica	*Unassigned*	Alex Tian		OPEN	08/Jan/15	22/Jan/17	newbie
	KAFKA-1354	Failed to load class "org.slf4j.impl.StaticLoggerBinder"	*Unassigned*	RakeshAcharya		OPEN	31/Mar/14	23/Jan/17	newbie patch usability
	KAFKA-1677	Governor on concurrent replica reassignments	Geoff Anderson	Ryan Berdeen		OPEN	06/Oct/14	28/Feb/15	newbie++
	KAFKA-4217	KStream.transform equivalent of flatMap	*Unassigned*	Elias Levy		OPEN	24/Sep/16	09/Jan/17	api needs-kip newbie
	KAFKA-4408	KTable doesn't work with ProcessorTopologyTestDriver in Kafka 0.10.1.0	*Unassigned*	Byron Nikolaidis		OPEN	14/Nov/16	24/Jan/17	newbie unit-test
	KAFKA-1155	Kafka server can miss zookeeper watches during long zkclient callbacks	Neha Narkhede	Neha Narkhede		OPEN	02/Dec/13	21/Feb/15	newbie++
	KAFKA-4320	Log compaction docs update	*Unassigned*	Dustin Cote		OPEN	19/Oct/16	31/Jan/17	newbie
	KAFKA-2939	Make AbstractConfig.logUnused() tunable for clients	Guozhang Wang	Guozhang Wang		OPEN	02/Dec/15	23/Jan/17	newbie
	KAFKA-4560	Min / Max Partitions Fetch Records params	*Unassigned*	Stephane Maarek		OPEN	19/Dec/16	21/Dec/16	features newbie
	KAFKA-1676	Transient Failure	*Unassigned*	Fangmin Lv		OPEN	25/Jul/15	23/Jan/17	newbie transient-unit-test-failure

KAFKA-3923: Make KafkaMetric not final, update JmxReporter and unit tests #1579

Open **stepio** wants to merge 4 commits into `apache:trunk` from `stepio:metrics_reporter`

Conversation 102 | Commits 4 | Files changed 8 | +227 −157

stepio commented on Jul 1, 2016 • edited — Contributor

This is a patch for the previously highlighted issue:
https://issues.apache.org/jira/browse/KAFKA-3923

I've minimised the scope of changes, so that now Metrics class is not affected and generics are introduced on the method level instead of class level.

stepio commented on Jul 1, 2016 — Contributor

It's funny that there are both FakeMetricsReporter and MockMetricsReporter - maybe the first one may be dropped.

stepio commented on Jul 5, 2016 — Contributor

Hello, **@guozhangwang** ,

May I ease the process of reviewing for you by any chance?

Reviewers

 ijuma

 guozhangwang

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Notifications

Unsubscribe

Take-aways

- **Build evolvable systems**
- **What gets measured gets fixed**
- **APIs stay forever**
- **(Multi-tenant) Services need gatekeepers**
- **Ecosystems are the key**

THANKS!

Kafka Summit 2017 @ NYC & SF

Guozhang Wang | guozhang@confluent.io | @guozhangwang