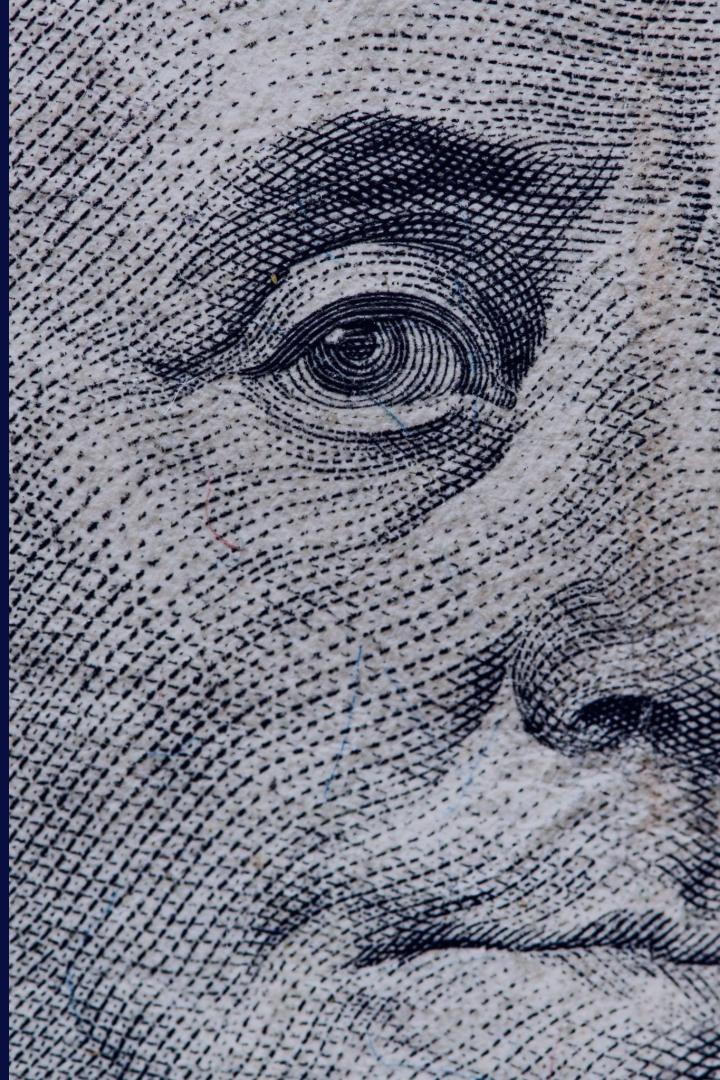


riskified tech;

Let's Make Your CFO Happy;
A Practical Guide for
**Kafka Cost
Reduction**



The Elephant In The Room



What are we paying for?

Running a self-hosted Kafka deployment



Where and how can we cut costs?

Tips, tricks, and KIPs



Develop an economic mindset

It's part of our role



Elad Leev

Data Engineer at Riskified

#DistributedSystems #DataStreams

#Scalability

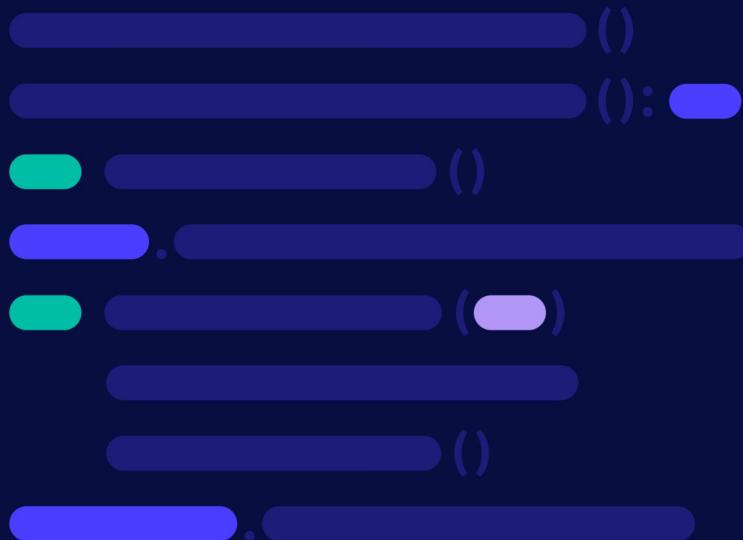
#Kafka #ConfluentCommunityCatalyst



@eladleev

What We Do

Riskified



Riskified by the Numbers

750+

Global team, nearly 50%
in **engineering & analytics**

180+

Countries across
the globe

50+

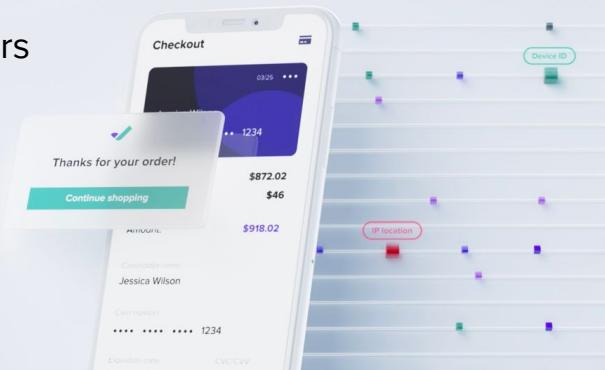
Publicly held companies
among our clients

\$89B

Online volume (GMV)
reviewed in 2021

98%+

Client retention*
for the past 2 years



*Annual dollar retention

As of March 2022

ticketmaster

macys

PRADA

wish

wayfair

lastminute.com

REVOLVE

FINISH LINE

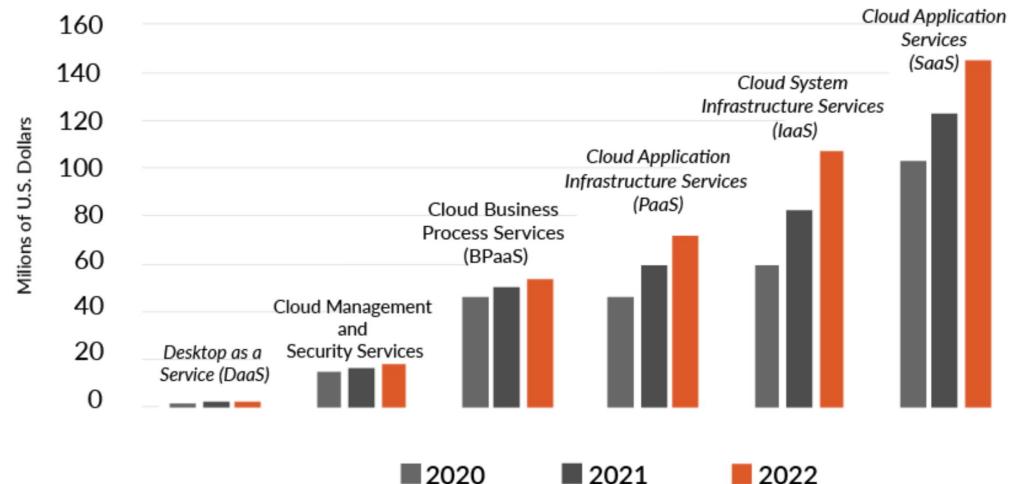


Let's dive in!

According to Gartner Forecasts

The worldwide end-user spending on public cloud services is forecast to grow by **20%** in 2022 to a total of **\$397 billion**.

■ Worldwide Public Cloud Services End-User Spending Forecast

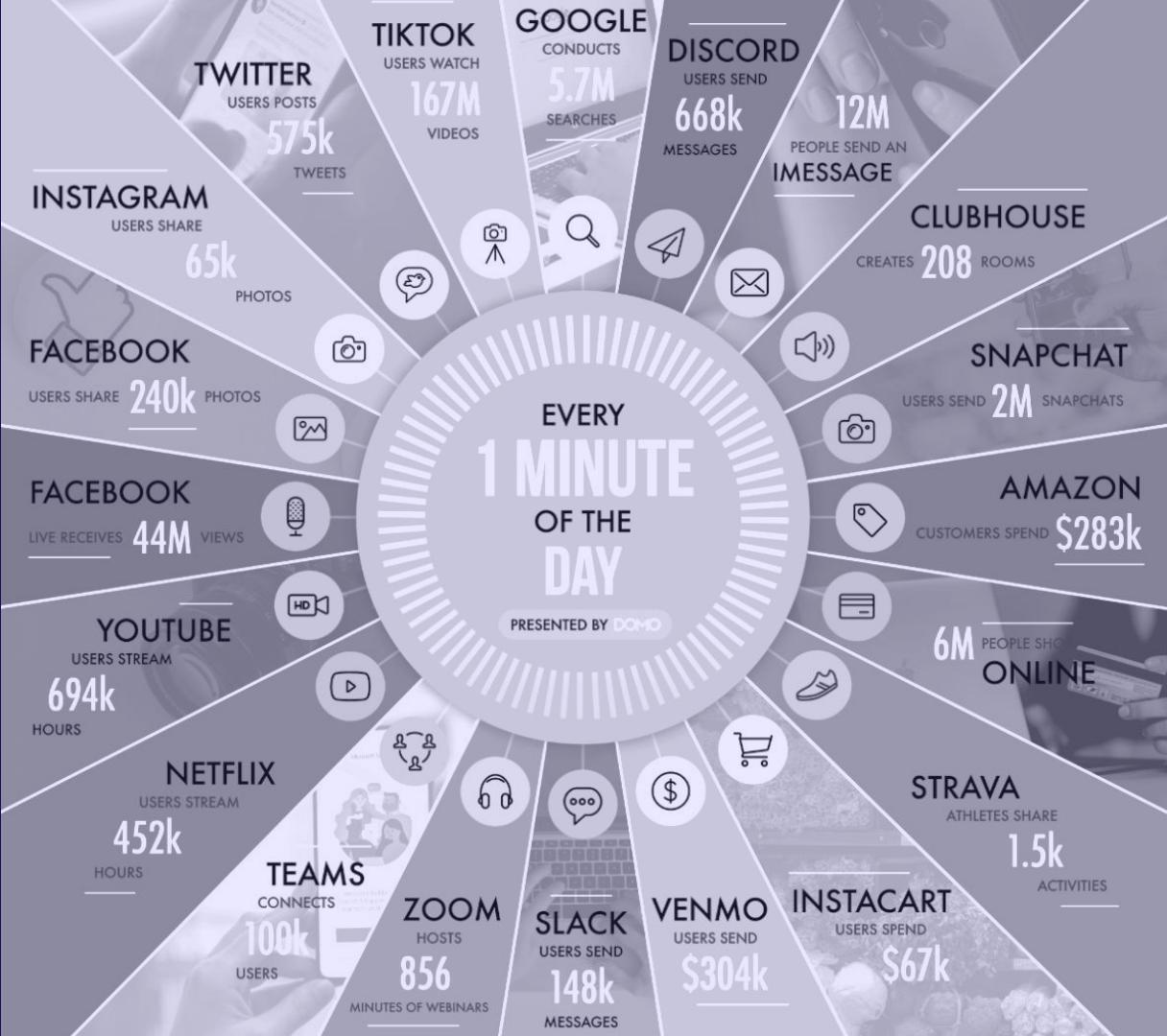


Source: Gartner (April 2021)



Data Never Sleeps

According to Statista, the total amount of data consumed globally in 2021 was **79 Zettabytes**.



More than
80%

of all Fortune 100 companies
trust, and use Kafka.



**10 OUT
OF 10**

MANUFACTURING



**7 OUT
OF 10**

BANKS



**10 OUT
OF 10**

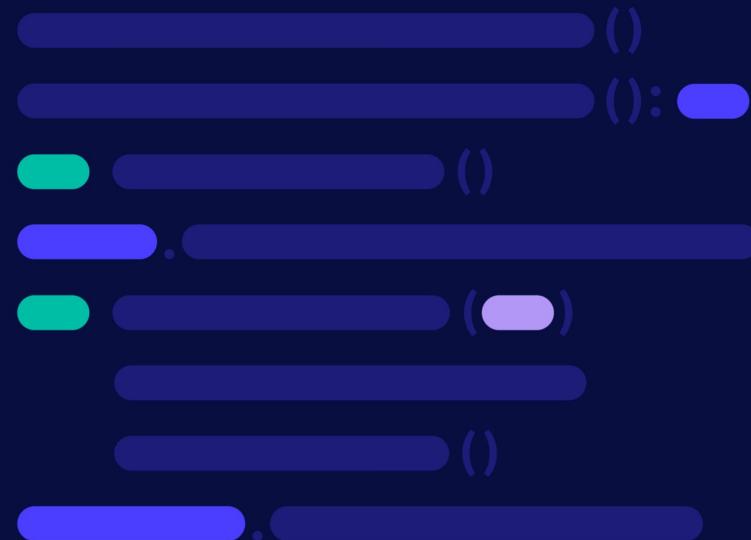
INSURANCE



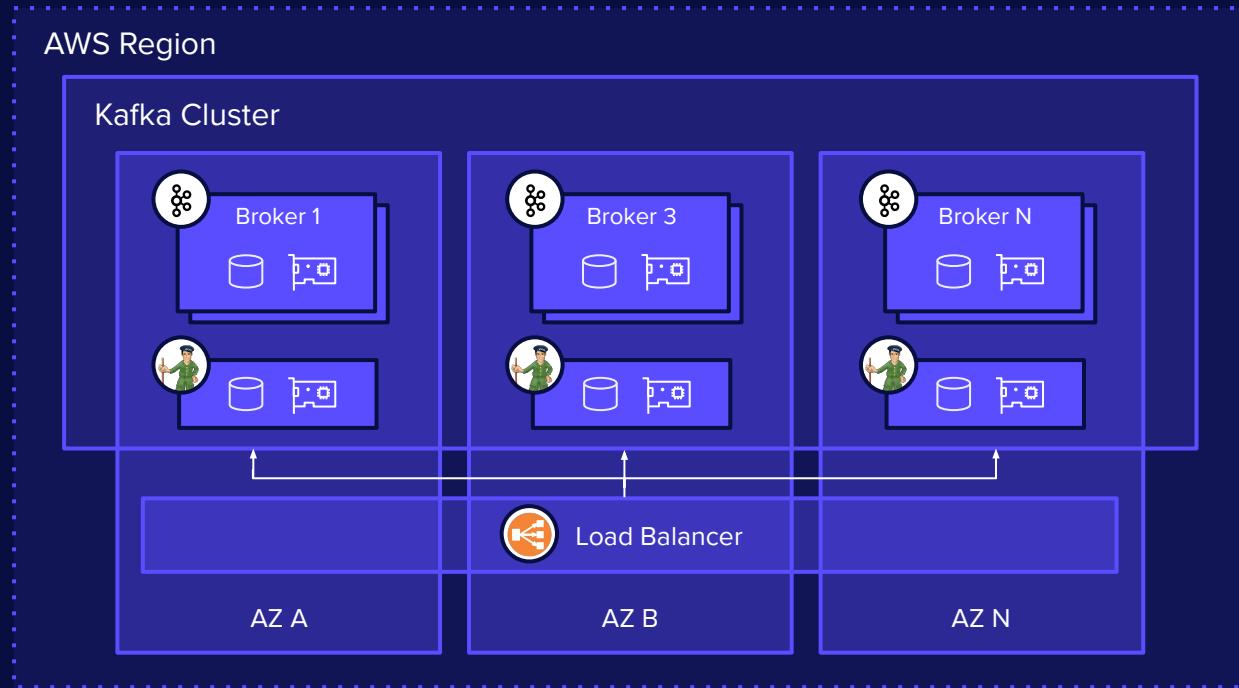
**8 OUT
OF 10**

TELECOM

What are we paying for?



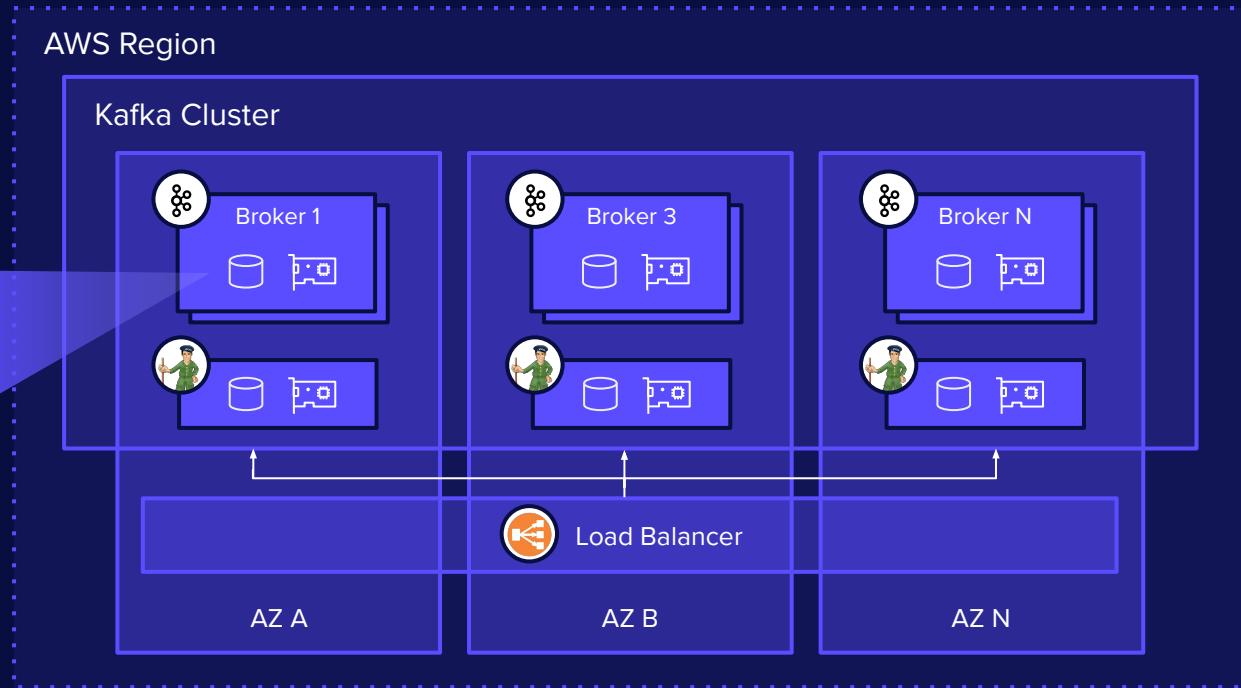
Kafka Deployment



Kafka Deployment



EC2 Machines



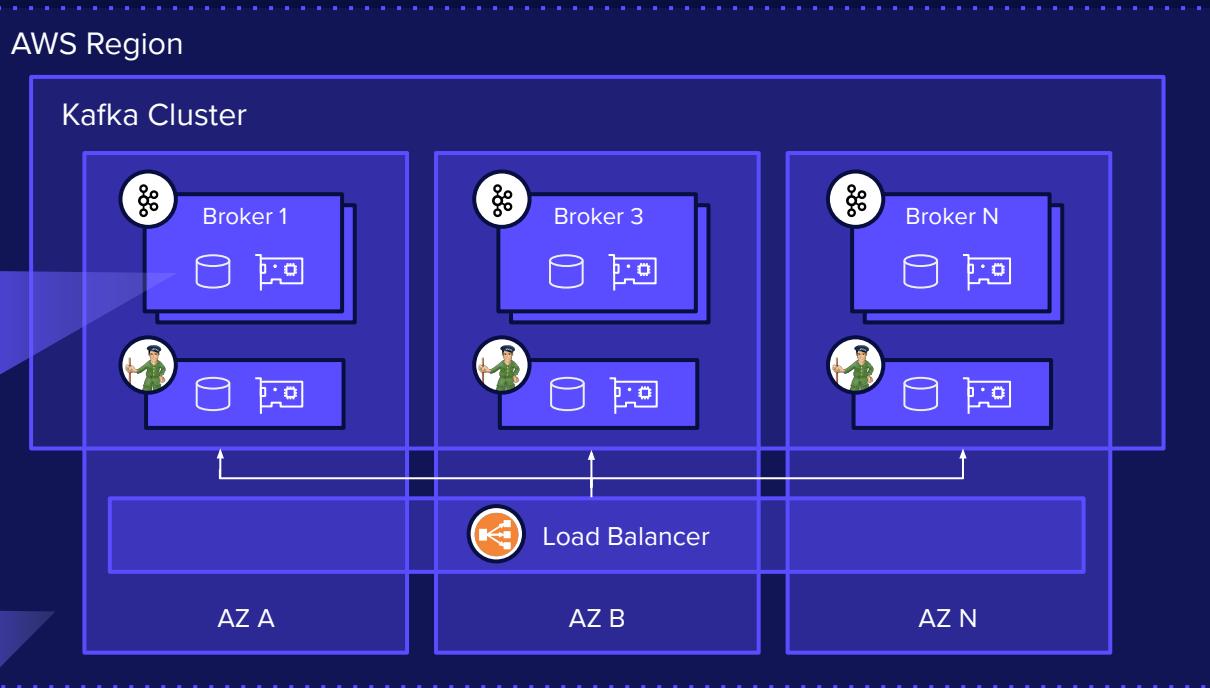
Kafka Deployment



EC2 Machines



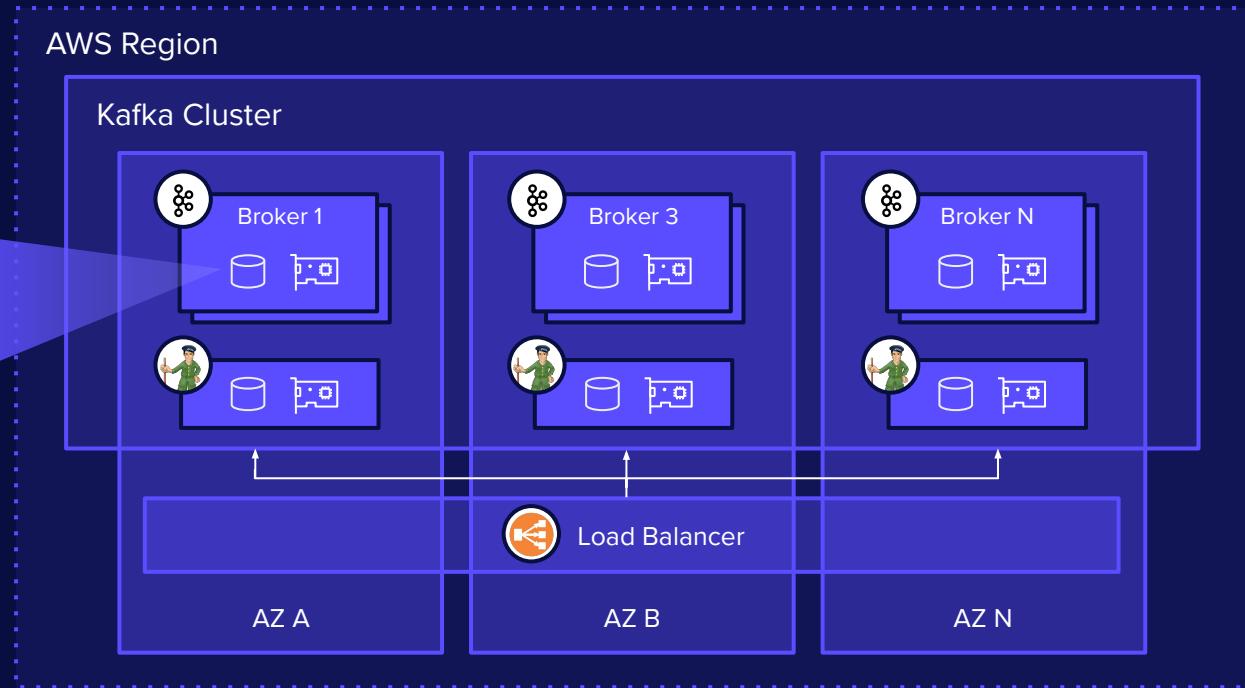
EKS



Kafka Deployment



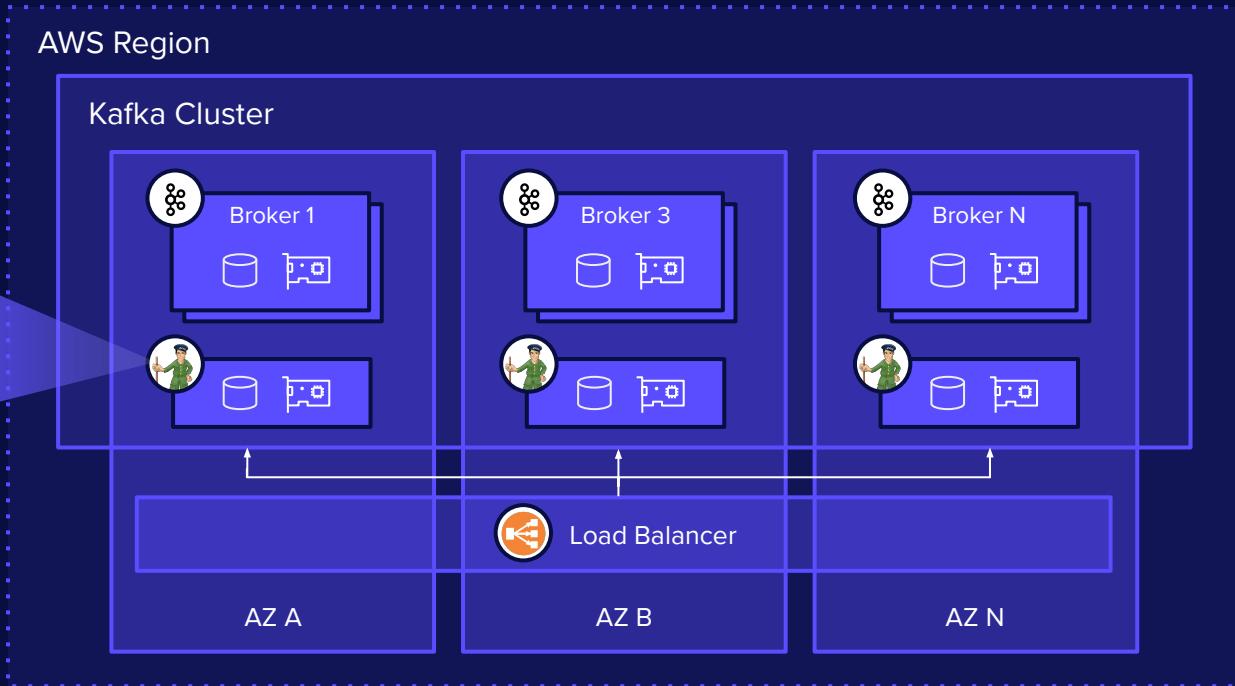
EBS Drives



Kafka Deployment



ZooKeepers



Kafka Deployment



AWS Region

Kafka Cluster



Broker 1



Broker 3



Broker N



AZ A

AZ B

AZ N



Load Balancer

\$

\$

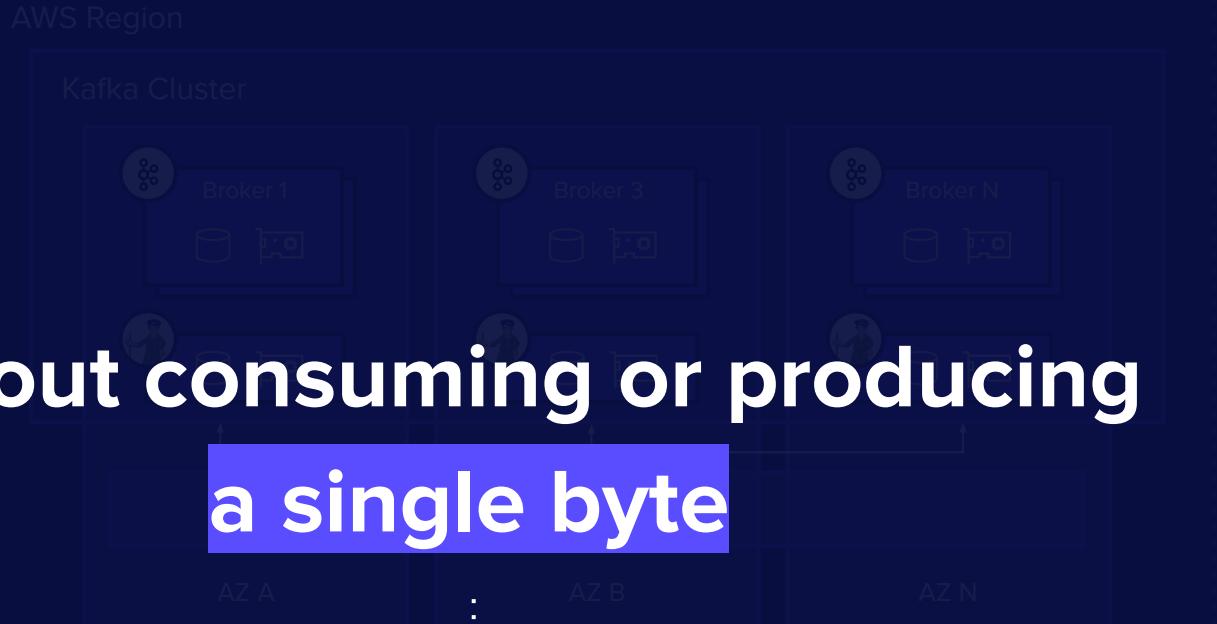
\$

\$

\$

\$

Kafka Deployment

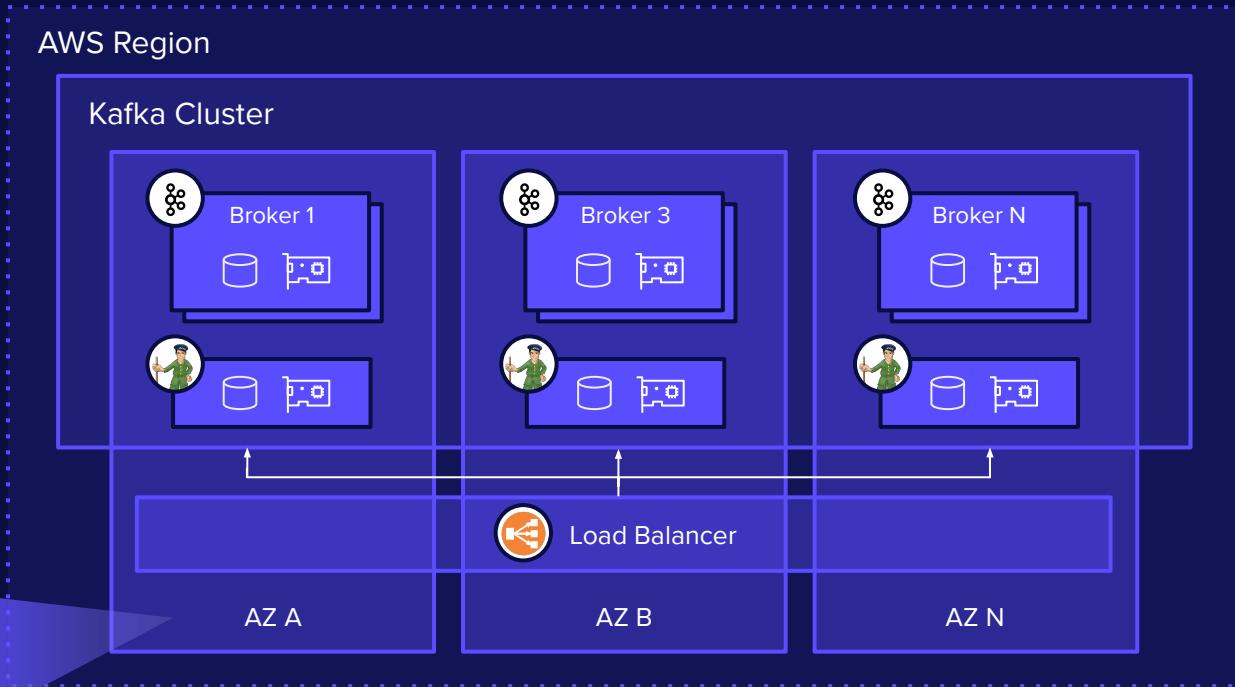


Without consuming or producing
a single byte

Kafka Deployment



Traffic



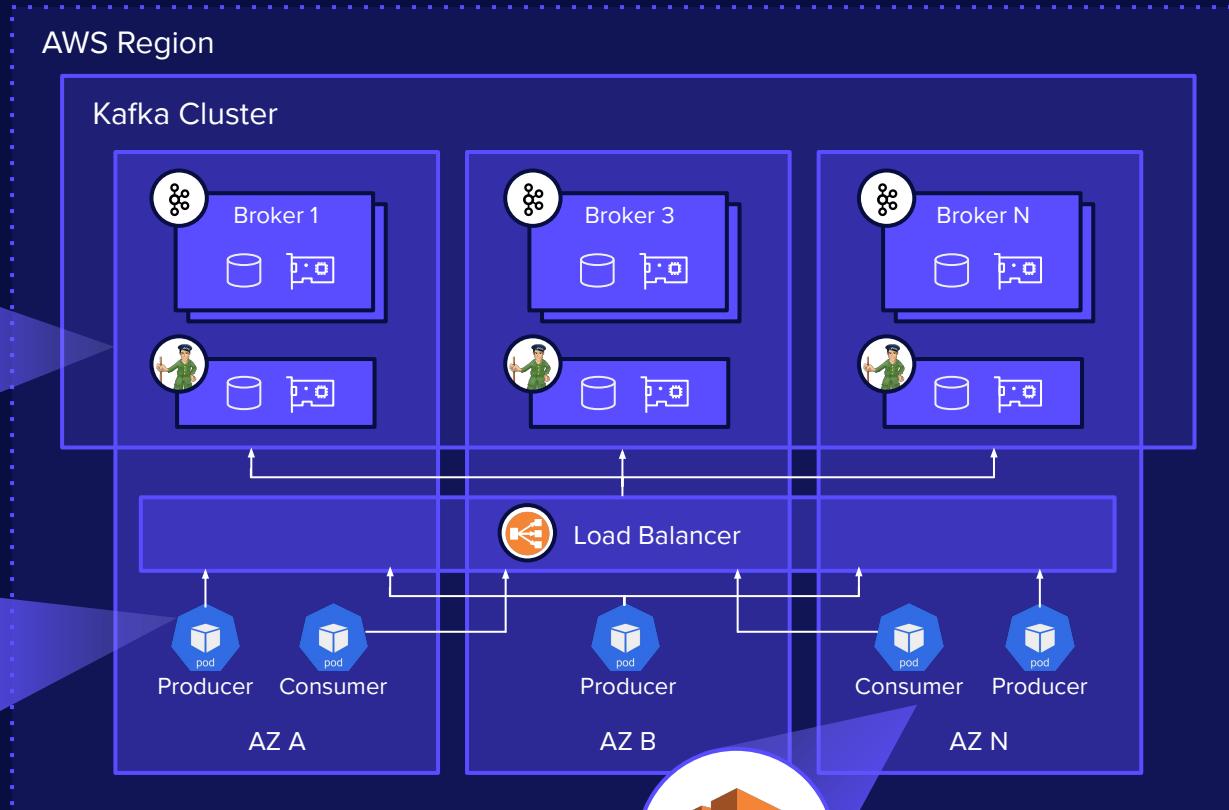
Kafka Deployment



Traffic



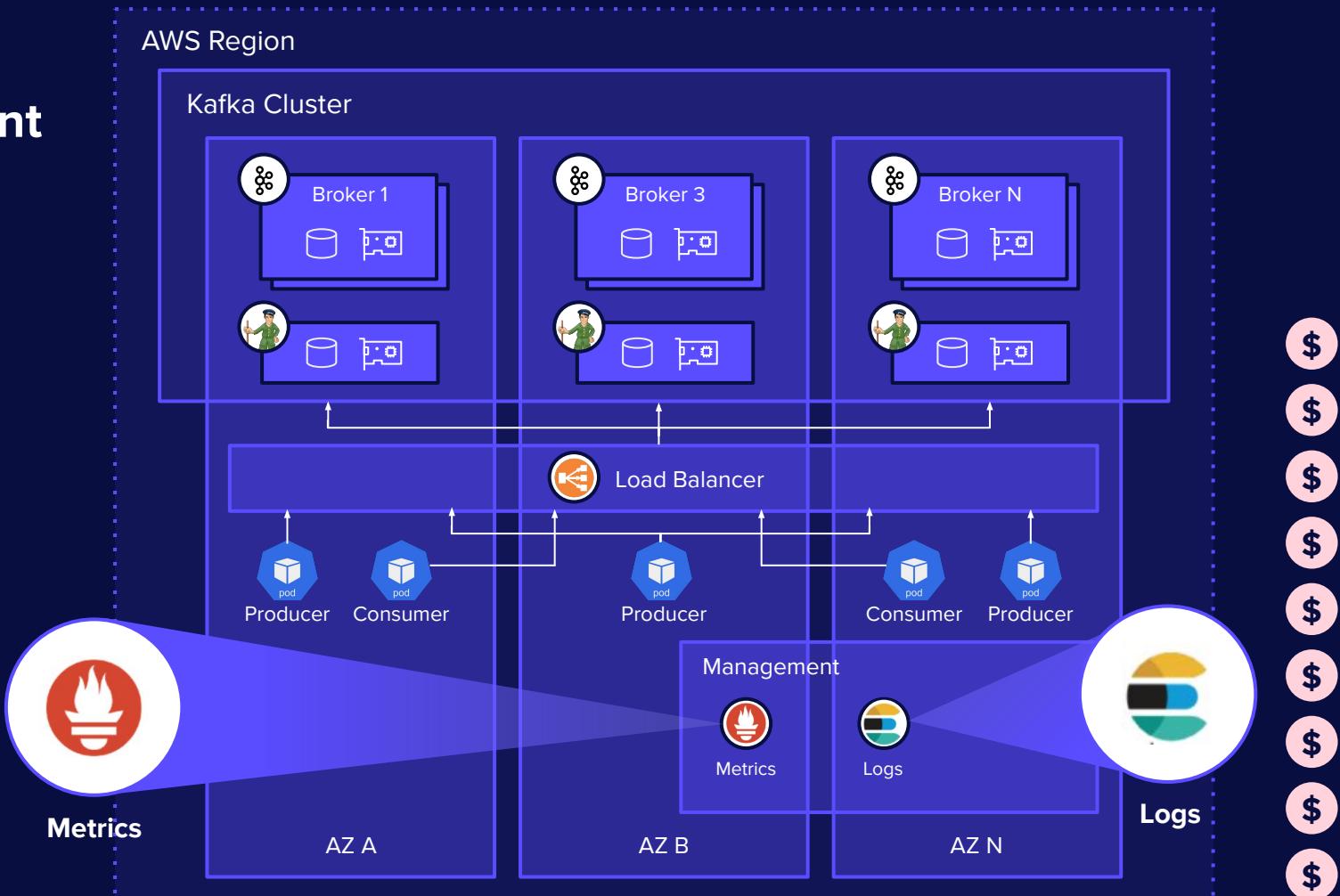
Traffic



Traffic



Kafka Deployment



Kafka Deployment



Schema Registry



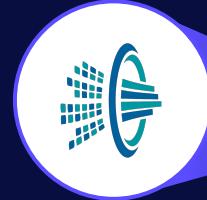
Kafka Connect



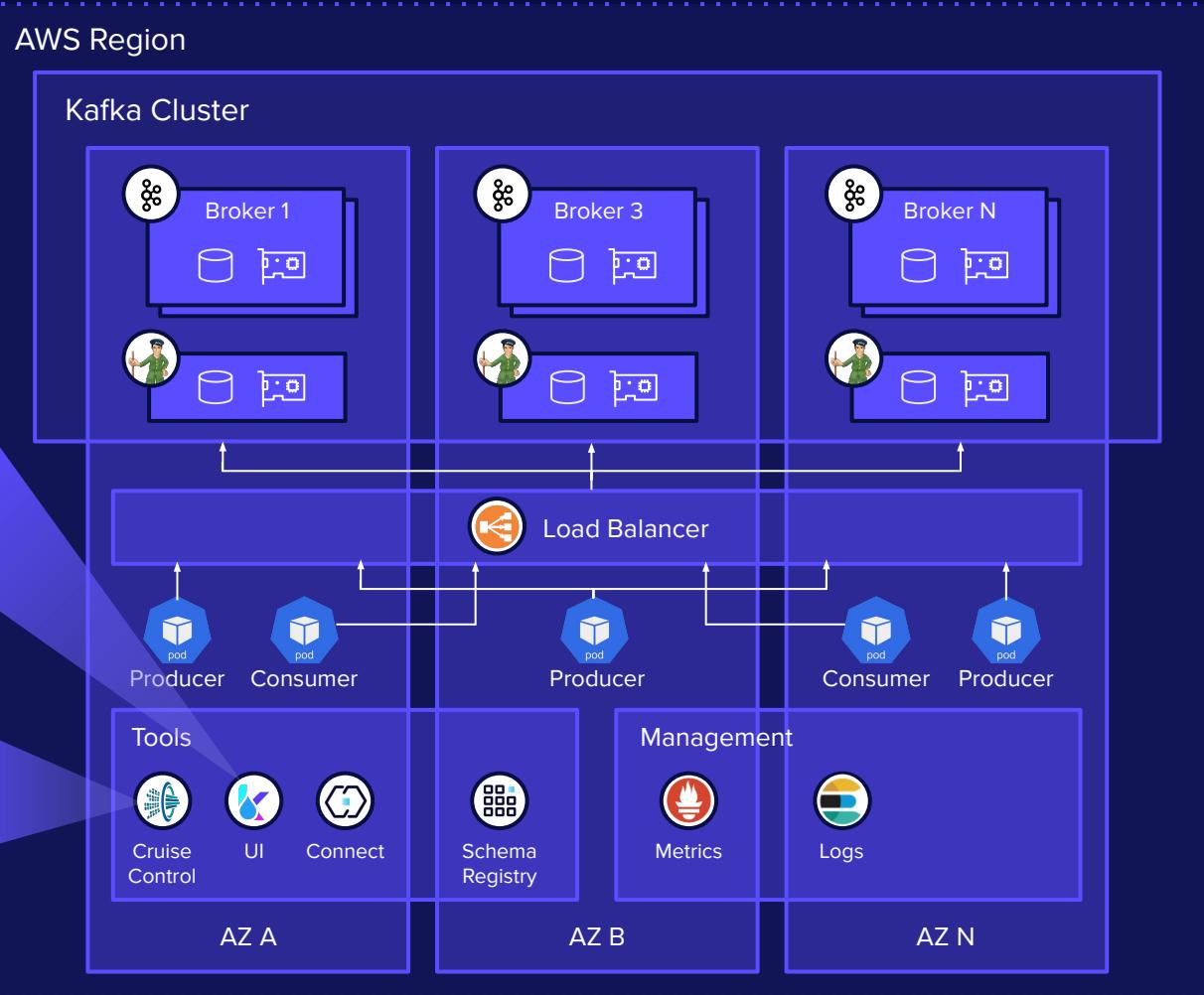
Kafka Deployment



Kafka
UI



Cruise
Control



Kafka Deployment





LOONEY
TUNES

Cost Factors

Machines



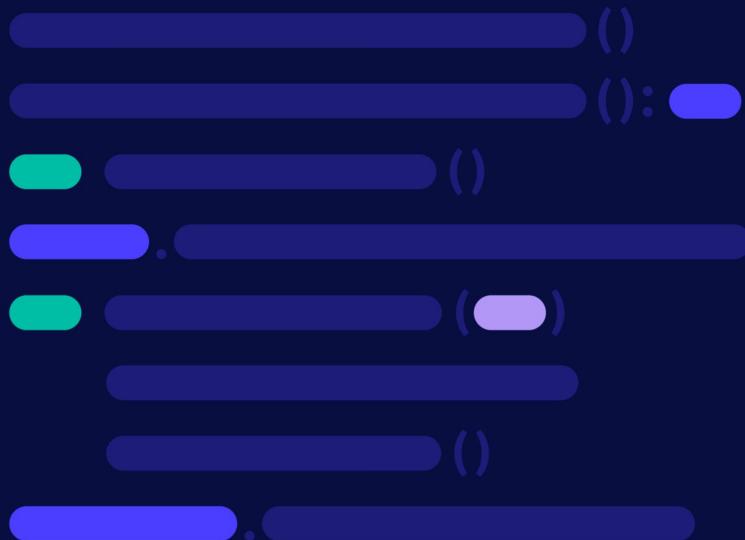
Network



Storage



What can we do?



01 **Instance type**



02 **Compression**

03 **Fetch from replica**

04 **Cluster balance**

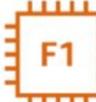
05 **Client configuration fine tune**

06 **Tiered storage**



Are you using the right instance type?

A quick Google search will suggest the **R5, D2, C5** combined with **GP2/3** or **IO2** storage as the broad recommendation.

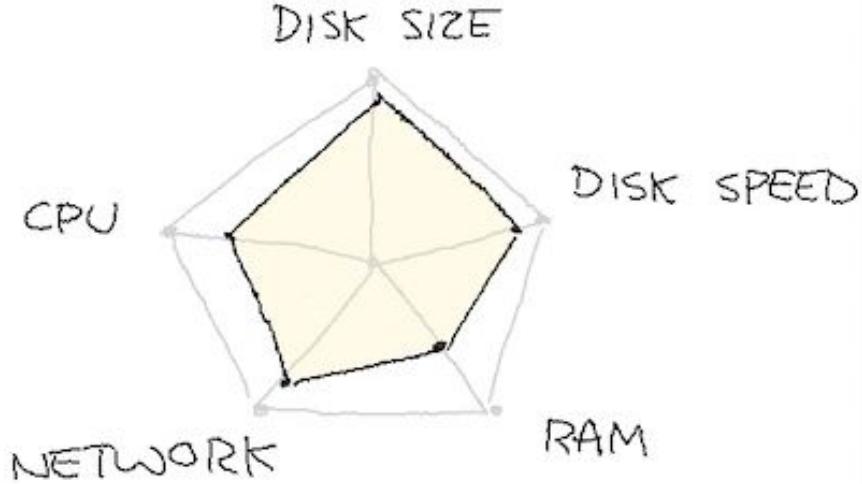
General Purpose	Compute Optimised	Memory Optimised	Accelerated Computing	Storage Optimised
 A1 ARM based core and custom silicon	 C5 Compute - CPU intensive apps and DBs	 R5 RAM - Memory intensive apps and DB's	 P2 Processing optimised-Machine Learning	 H1 High Disk Throughput - Big data clusters
 T2 Tiny - Web servers and small DBs		 X1 Xtreme RAM - For SAP/Spark	 G3 Graphics Intensive - Video and streaming	 I3 IOPS - NoSQL DBs
 M4 Main - App servers and general purpose		 z1d High Compute and High Memory - Gaming	 F1 Field Programmable - Hardware acceleration	 D2 Dense Storage - Data Warehousing



Are you using the right instance type?

Everything is a trade-off.

Choose wisely based on your needs: Time-to-recover, Storage-to-dollar ratio, network throughput, EBS downfalls.

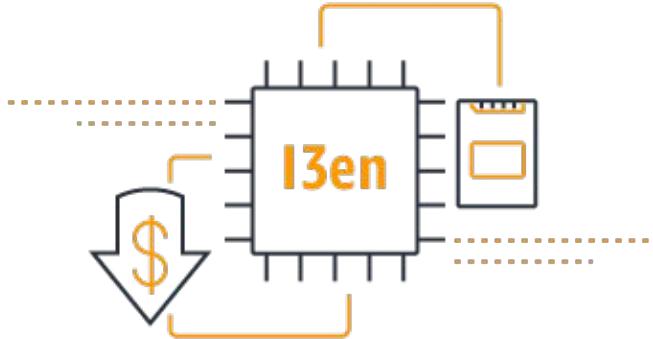
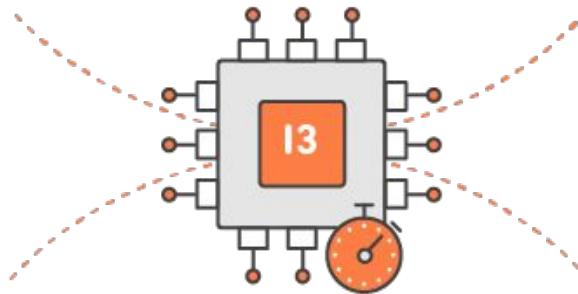


Source: Liz Fong-Jones, Honeycomb.io



i3 and i3en for High-performance Kafka Deployments

Despite the operational overhead of ephemeral drives.



i3en provides great storage-to-dollar ratio

Suitable for I/O intensive deployments in which the limiting factor is storage capacity.

	Storage size per core	price per core	price per TB of data
i3.16xlarge	480GB	\$ 880.50 / year	\$ 1,857.96 / year
i3.metal	426GB	\$ 773.78 / year	\$ 1,857.06 / year
i3en.24xlarge	1.2TB	\$ 1,258.43 / year	\$ 1,006.75 / year

Benchmark	i3.16xlarge (ops)	i3en.24xlarge (ops)	difference
cache	1.59	2.25	41.51%
cpu	221.21	302.40	36.70%
dentry	23,578.10	29,544.91	25.31%
icache	399.97	449.88	12.48%
matrix	4,219.04	4,133.63	-2.02%
memcpy	567.87	381.34	-32.85%
qsort	8.82	10.63	20.52%
timer	119,784.59	136,156.85	13.67%
tsc	847,677.39	983,544.69	16.03%

Source: ScyllaDB



Are you using the right instance types?

**Is your cluster saturated?
On what conditions?**

**What is the
limiting factor?**

**Are EBS drives the right
decision?**

**Do the cluster needs
changed over time?**

01 **Instance type**



02 **Compression**

03 **Fetch from replica**

04 **Cluster balance**

05 **Client configuration fine tune**

06 **Tiered storage**



Change compression type

You can choose between GZIP, LZ4, Snappy, and since KIP-110 - **ZSTD**.

compression.type

The compression type for all data generated by the producer. The default is none (i.e. no compression). Valid values are `none`, `gzip`, `snappy`, `lz4`, or `zstd`. Compression is of full batches of data, so the efficacy of batching will also impact the compression ratio (more batching means better compression).

Type: string

Default: none

Valid Values:

Importance: high

[Pages](#) / [Index](#) / [Kafka Improvement Proposals](#)

KIP-110: Add Codec for ZStandard Compression

Created by Dongjin Lee, last modified on Nov 18, 2018

- Status
- Motivation
- Proposed Changes
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives
 - A. Support ZStandard to the old clients which can understand v0, v1 messages only.
 - B. Bump the API version and support only v2-available clients
 - B.1. INVALID_REQUEST (42)
 - B.2. CORRUPT_MESSAGE (2)
 - B.3 UNSUPPORTED_FOR_MESSAGE_FORMAT (43)
- Related issues
 - Whether to use existing library or not
 - License



Zstandard

Compression algorithm
by Facebook^{Meta}
It aims for a smaller and
faster data compression.

Compressor name	Ratio	Compression	Decompress.
zstd 1.4.5 -1	2.884	500 MB/s	1660 MB/s
zlib 1.2.11 -1	2.743	90 MB/s	400 MB/s
brotli 1.0.7 -0	2.703	400 MB/s	450 MB/s
zstd 1.4.5 --fast=1	2.434	570 MB/s	2200 MB/s
zstd 1.4.5 --fast=3	2.312	640 MB/s	2300 MB/s
quicklz 1.5.0 -1	2.238	560 MB/s	710 MB/s
zstd 1.4.5 --fast=5	2.178	700 MB/s	2420 MB/s
lzo1x 2.10 -1	2.106	690 MB/s	820 MB/s
lz4 1.9.2	2.101	740 MB/s	4530 MB/s
lzf 3.6 -1	2.077	410 MB/s	860 MB/s
snappy 1.1.8	2.073	560 MB/s	1790 MB/s



Real world examples

Using Zstd, Shopify was able to get a 4.28x compression ratio compared to Snappy's 2.5x.



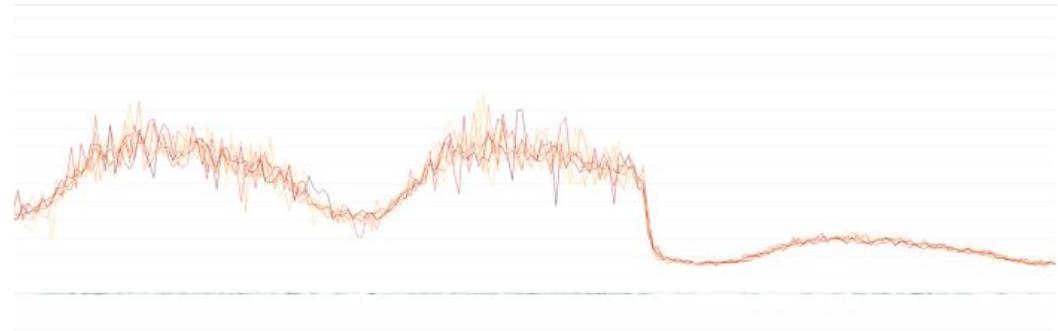
“

After switching to Zstandard,
our bandwidth usage
decreased **by 3x!**

This saves us tens of
thousands of dollars per
month in data transfer costs
just from the processing
pipeline alone.

With these great results we
immediately began deploying
Zstandard to other systems
in our architecture for further
cost savings.

”



Bandwidth usage



KIP-390: Support Compression Level

Zstd Level 1 produces **32.7% more** messages per second than Zstd Level 3.

Gzip Level 1 produces **56.4% more** than Gzip Level 6.

Status: **IN PROGRESS**

Resolution: Unresolved

Fix Version/s: 3.2.0

codec	level	produced message / sec	latency (ms)	size (bytes)	description
none		2,739.50	205.34	5,659,454,754	
gzip	1	1,122.96	1,230.22	1,787,505,238	min. level
gzip	6	717.71	2,041.24	1,644,280,629	default level
gzip	9	608.54	2,413.66	1,643,517,758	max. level
lz4	1	1,694.69	603.46	2,211,346,795	min. level
lz4	9	1,199.93	878.85	2,184,022,257	default level
lz4	17	495.34	2,110.55	2,178,643,665	max. level
zstd	-5	7,653.45	156.88	1,997,500,892	experimental level
zstd	1	6,317.52	68.55	1,521,783,958	
zstd	3	4,760.54	286.79	1,494,620,615	default level
zstd	12	988.95	863.89	1,458,150,768	
zstd	18	85.20	2,017.92	1,492,015,424	

29218 JSON files with an average size of 55.25kb

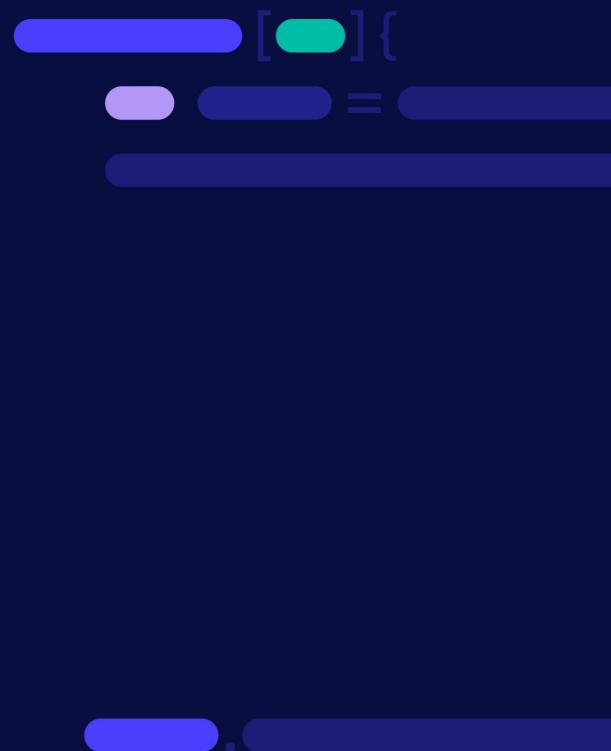


```
compression.type=zstd  
compression.level=1
```

NEW: Compression level to be used.

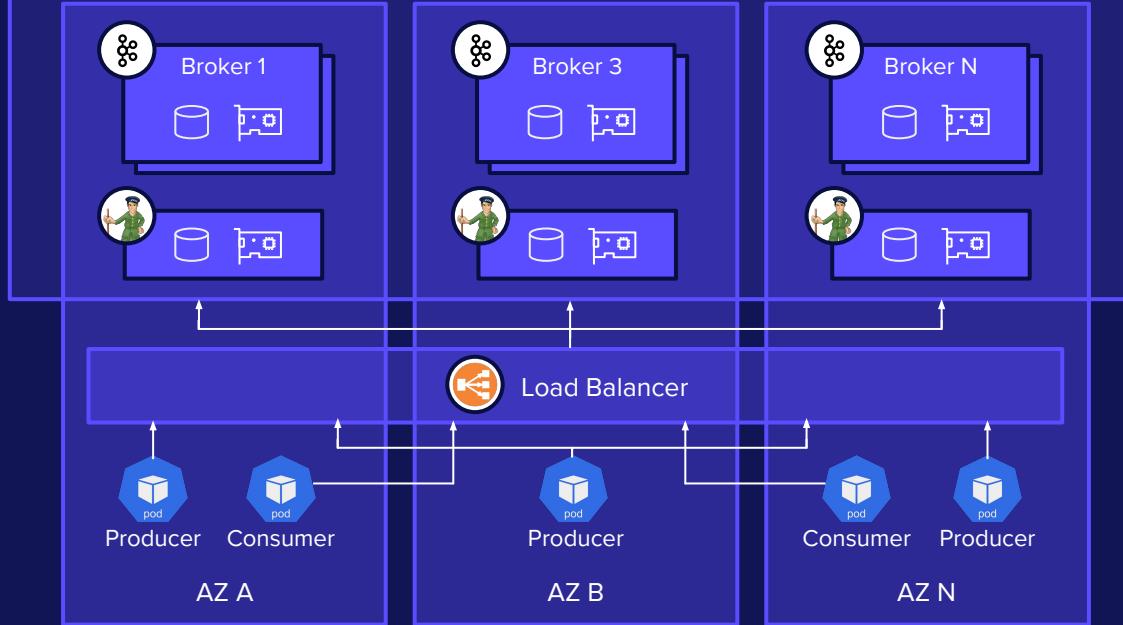


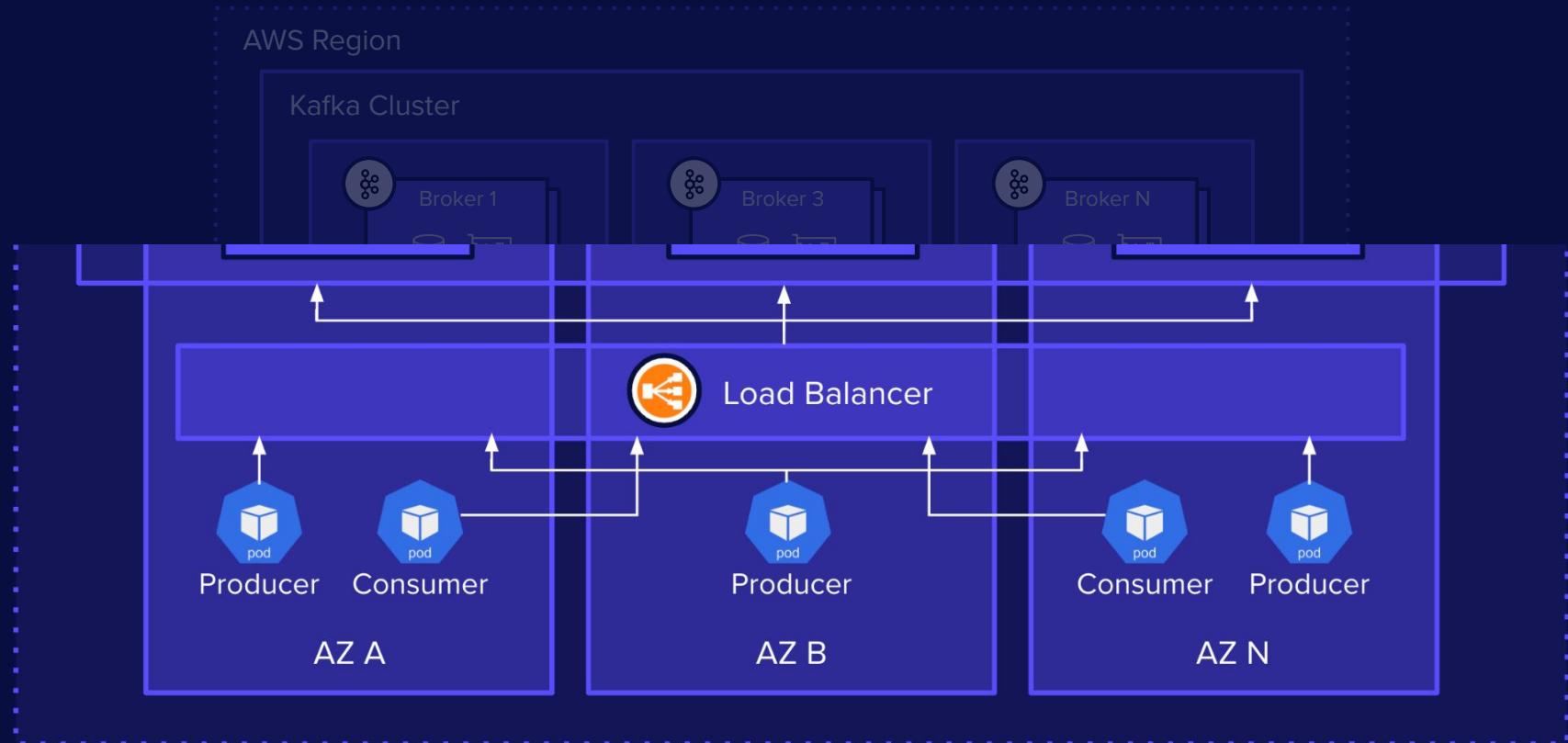
- 01 **Instance type**
- 02 **Compression**
- 03 **Fetch from replica**
- 04 **Cluster balance**
- 05 **Client configuration fine tune**
- 06 **Tiered storage**



AWS Region

Kafka Cluster





KIP-392: Fetch from closest replica

Leverage locality in order to reduce expensive cross- AZ / cross-DC traffic costs.

KIP-392: Allow consumers to fetch from closest replica

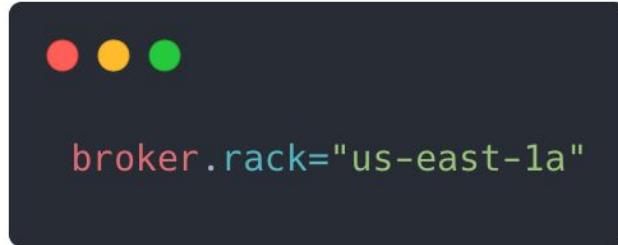
Created by Jason Gustafson, last modified by Matthias J. Sax on Nov 05, 2019

- Status
- Motivation
- Proposed Changes
 - Follower Fetching
 - High watermark propagation
 - Out of range handling
 - Finding the preferred follower
 - Public Interfaces
 - Consumer API
 - Broker API
 - Protocol Changes
 - Compatibility, Deprecation, and Migration Plan
 - Rejected Alternatives



KIP-392: Fetch from closest replica

Rack awareness needs to be set. Support from the client is needed as well.



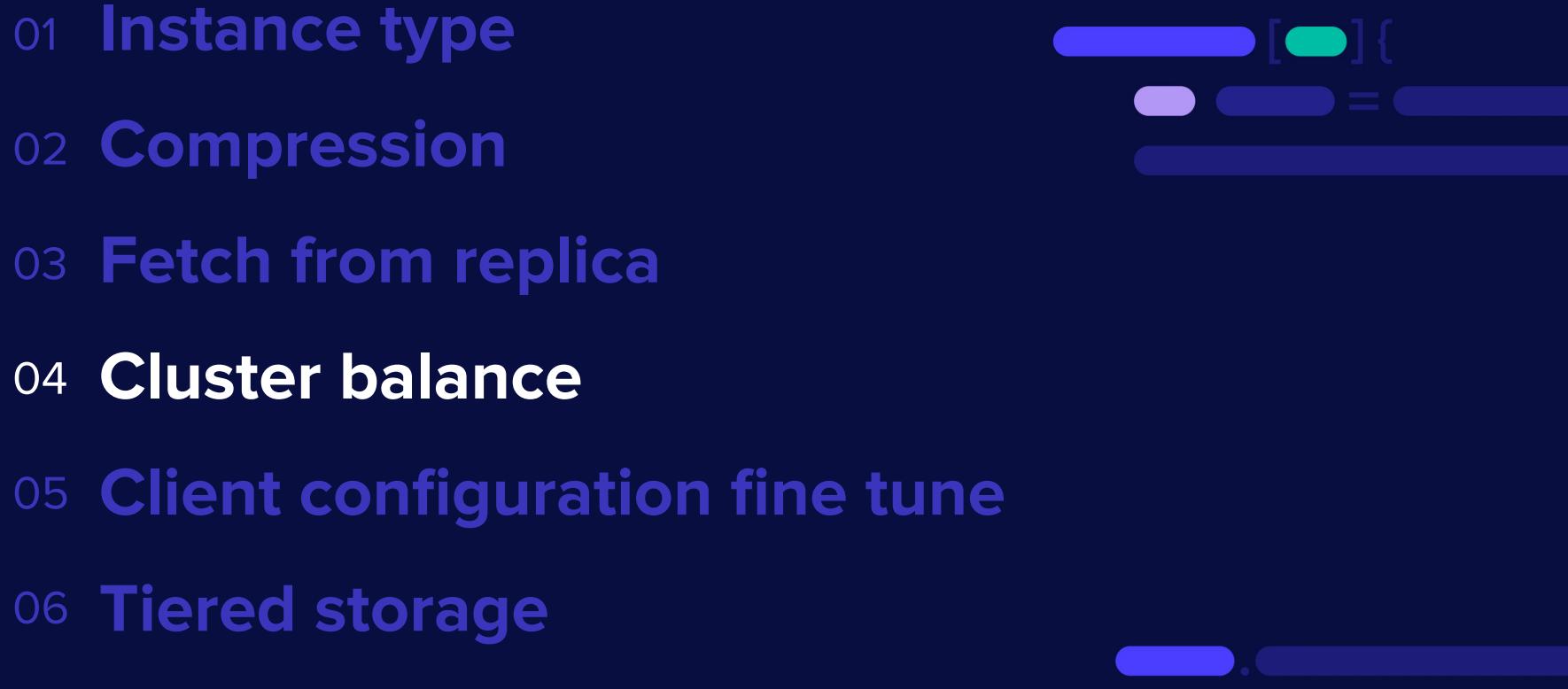
A screenshot of a GitHub pull request page for KIP-392. The repository is 'Shopify / sarama'. The pull request has 12 reviews and was merged by dnwe on Nov 6, 2020. The commit message says 'Merged dnwe merged 4 commits into Shopify:master from danp:dnearalla-kip-392'.

Shopify / sarama (Public)

Code Issues 220 Pull requests 12 Actions Projects Wiki Security Insights

KIP-392: Allow consumers to fetch from closest replica #1822

Merged dnwe merged 4 commits into Shopify:master from danp:dnearalla-kip-392 on Nov 6, 2020

- 
- 01 **Instance type**
 - 02 **Compression**
 - 03 **Fetch from replica**
 - 04 **Cluster balance**
 - 05 **Client configuration fine tune**
 - 06 **Tiered storage**

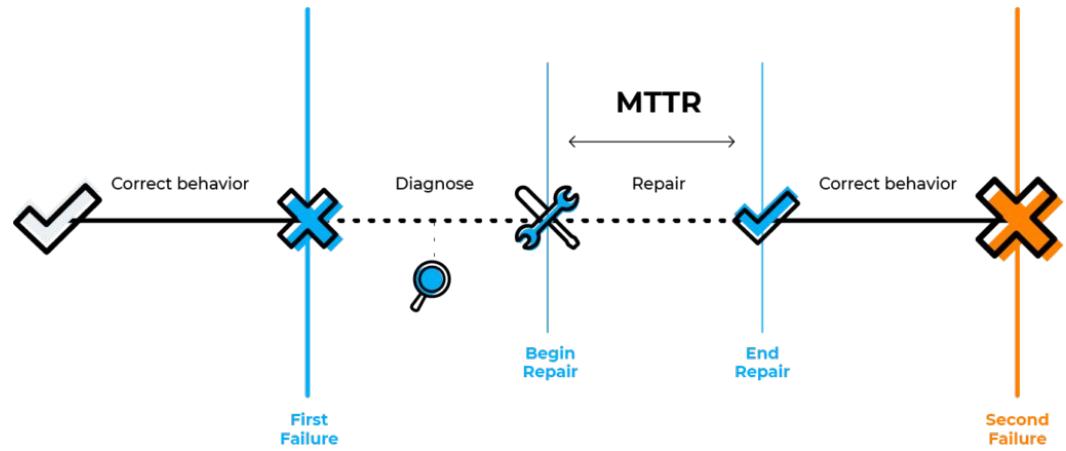
Cluster imbalance

Might hurt cluster performance, lead to brokers skew, resource saturation, and as consequence - unnecessary capacity addition.



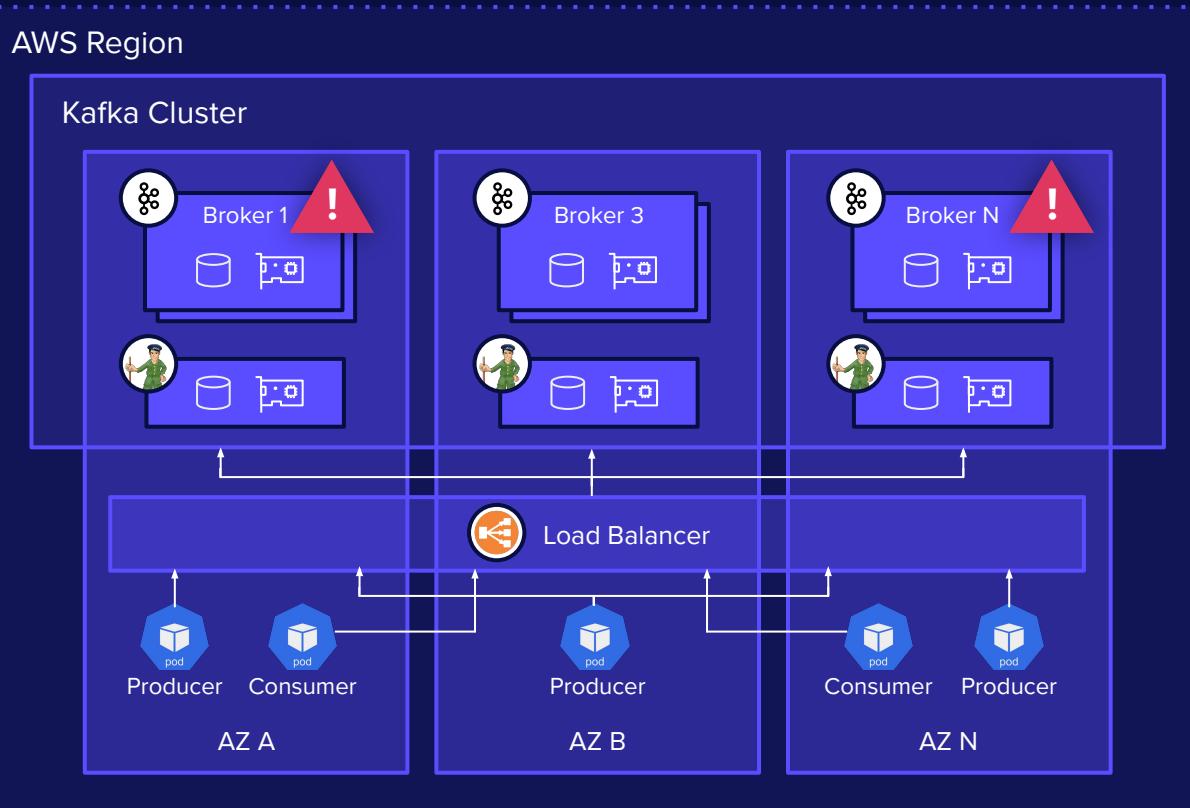
Cluster imbalance

Overloaded brokers (more partitions, more segments) will suffer from higher MTTR.



Cluster imbalance

Long startup time
exposes you to a higher
risk of data loss in case
of cascading failures.



Cluster imbalance

Relatively easy-to-do task
using-
Cruise Control, CMAK,
Kafka-Kit, and more.

linkedin/cruise-control

Cruise-control is the first of its kind to fully automate the dynamic workload rebalance and self-healing of a Kafka cluster. It provides great value to Kafka users by simplifying the operation of Kafka clusters.



11
Contributors

2
Issues

2k
Stars

49
Forks

yahoo/CMAK

CMAK is a tool for managing Apache Kafka clusters



71
Contributors

463
Issues

11k
Stars

2k
Forks

DataDog/kafka-kit

Kafka storage rebalancing, automated replication throttle, cluster API and more



10
Contributors

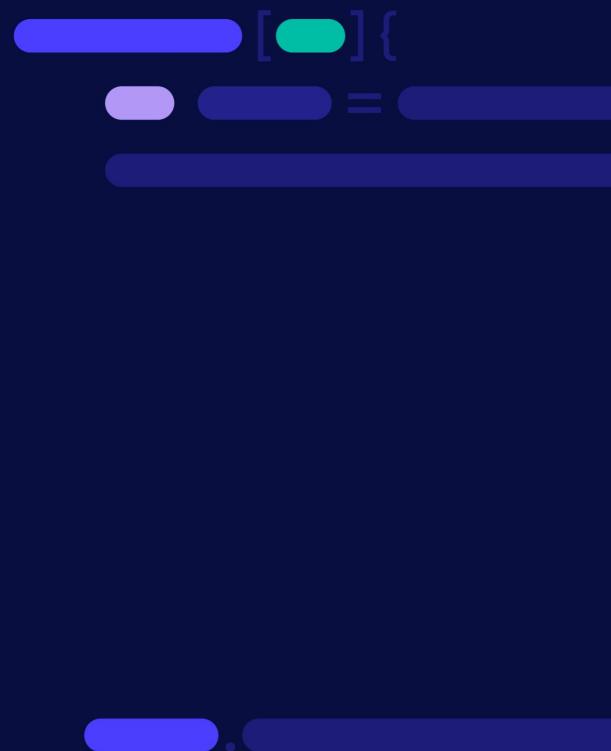
1
Used by

398
Stars

42
Forks



- 01 **Instance type**
- 02 **Compression**
- 03 **Fetch from replica**
- 04 **Cluster balance**
- 05 **Client configuration fine tune**
- 06 **Tiered storage**



Client Configurations

Deep dive into the client configurations knobs can dramatically affect the way your cluster works



GOT TO PICK A BUTTON.



EENY, MEENY, MINEY...

Clients Configurations

Kafka just works out of the box. However to unlock its full potential - invest time in learning.



Optimizing Kafka broker configuration

June 08, 2021 by Paul Mellor

Following our popular and evergreen post on Kafka brokers, we’re back with another post on Kafka brokers.

In terms of the options you can use, there’s a lot to consider.

And when we say brokers, we allude to the Strimzi Kafka brokers.

Basic broker configuration

A basic broker configuration might look like this:

```
# ...
num.partitions=1
default.replication.factor=1
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.max.size.bytes=104857600
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=60000
num.network.threads=3
num.io.threads=8
num.recovery.threads.peripheries=1
socket.receive.buffer.bytes=131072
socket.request.max.bytes=1048576
group.initial.rebalance.delay.ms=0
zookeeper.connection.timeout.ms=6000
# ...
```

In Strimzi, you configure these settings in the `strimzi.yaml` file.



Optimizing Kafka consumers

January 07, 2021 by Paul Mellor

We recently gave a few pointers on how to tune your Kafka consumer.

It is worthwhile considering both ends of the spectrum when you’ve made. As we shall see in this post, there are many ways to tune your consumer.

How is your consumer performing?

When looking to optimize your consumer, it’s important to understand the level of throughput you expect. As a general rule, the more partitions you have, the higher the throughput.

Basic consumer configuration

A basic consumer configuration must have the `bootstrap.servers` source for requests in logs and metrics.

```
bootstrap.servers=localhost:9092
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer
value.deserializer=org.apache.kafka.common.serialization.StringDeserializer
client.id=my-client
```



Optimizing Kafka producers

October 15, 2020 by Paul Mellor

You can fine-tune Kafka producers using configuration properties to optimize the streaming of data to consumers. Get the tuning right and your producers operate.

In this post we’ll discuss typical tuning considerations for Kafka producers.

Optimizing Kafka producers

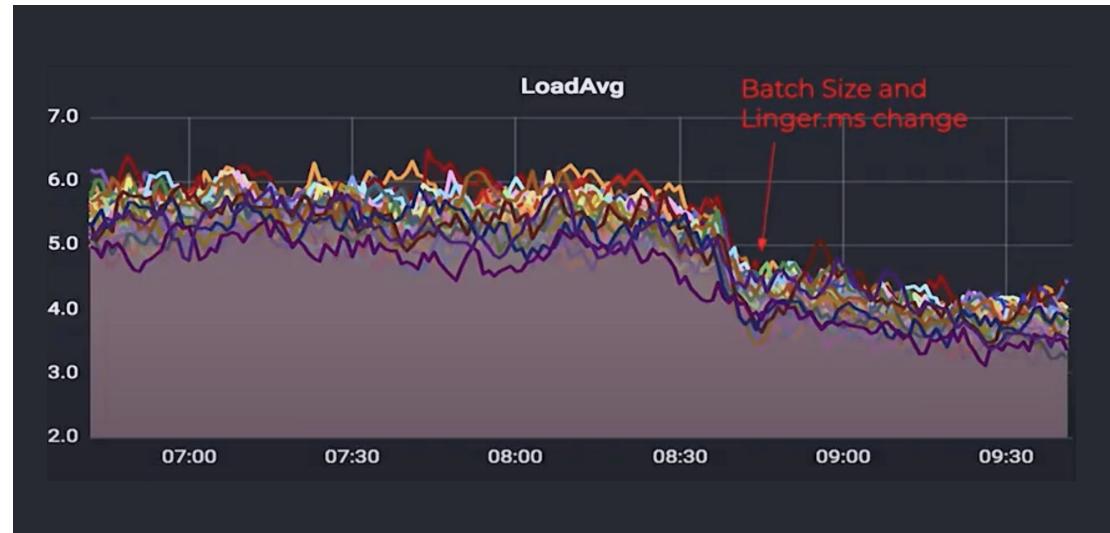
Obviously, we want our producers to deliver data to Kafka topics as efficiently as possible. But what do we mean by this, and how do we achieve it? We’ll also increase latency.

Be prepared to make adjustments to your adjustments.

How is your producer performing?

Clients Configurations

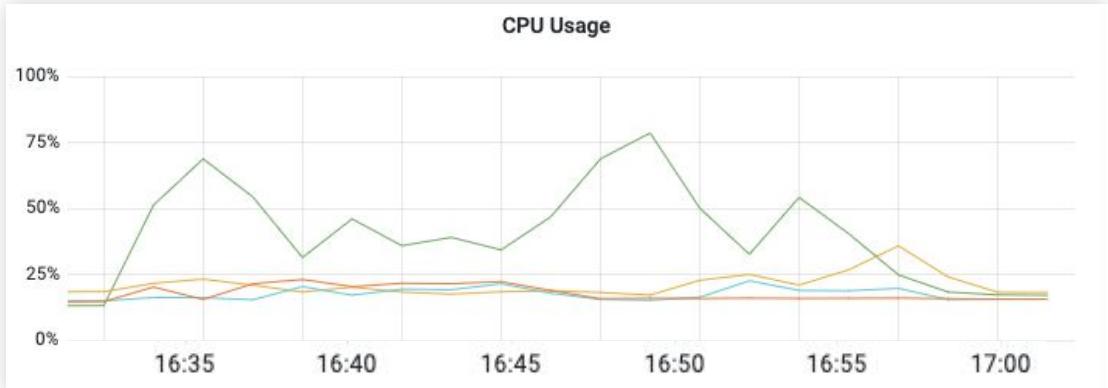
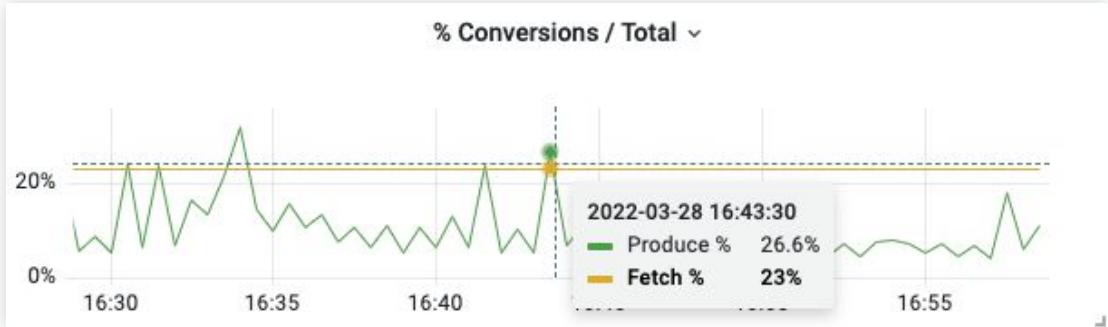
For example, changing your clients **batch.size** and **linger.ms** configurations can reduce your cluster resource utilization.



Message Conversions

Message conversion introduces a processing overhead.

Upgrading clients can free up resources that were used unnecessarily for this task.



- 
- 01 **Instance type**
 - 02 **Compression**
 - 03 **Fetch from replica**
 - 04 **Cluster balance**
 - 05 **Client configuration fine tune**
 - 06 **Tiered storage**
- 

The Cluster Storage

Kafka became the main entry point of all of the data in organizations, allowing clients to consume not only the recent events, but also older data based on the topic retention.



Upstream Clusters

A common pattern is to split your cluster into smaller clusters based on business domains.

Using this method you can set a higher retention period on the upstream cluster for data recovery in case of a failure.



Capacity addition

Both cases require you to add disk capacity to support growth.

High retention, justified by business needs, may lead to needless resource addition (memory and CPUs) to support storage capacity.

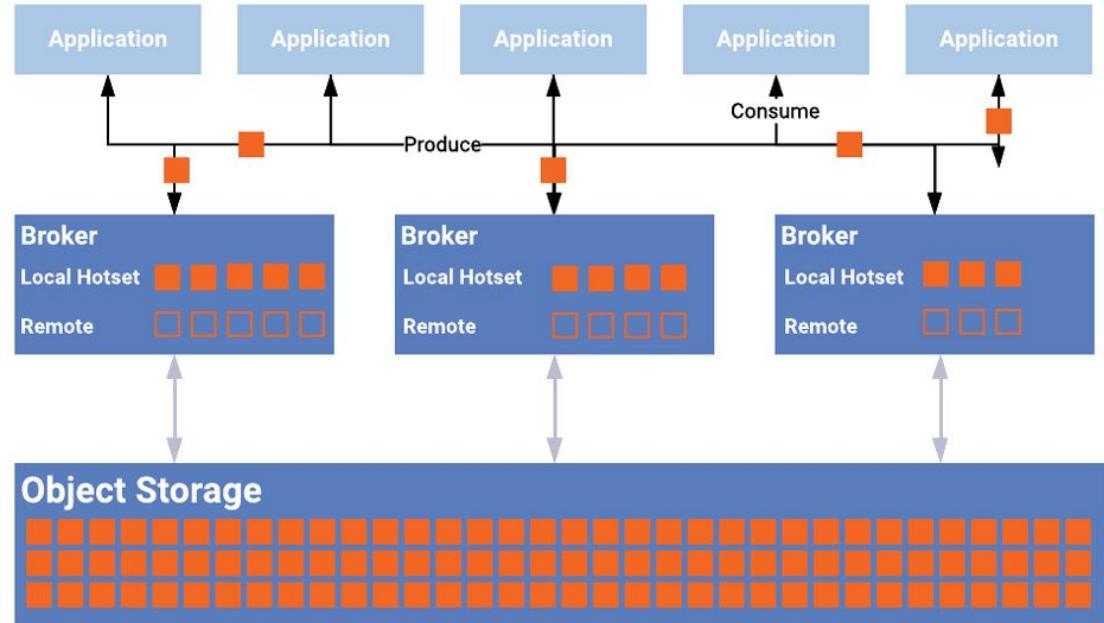


KIP-405:

Kafka Tiered Storage

Using Tiered Storage, Kafka clusters are configured with two tiers of storage: local and remote.

The new remote tier uses external object storage layers such as AWS S3 or HDFS.



Kafka Tiered Storage

Tiered Storage is already available on Confluent Platform 6.0.0, and will be added to the Kafka 3.2 release.

01 Storage cost saving

~\$0.08 per GB on gp3 Vs.
~\$0.023 on S3

02 Accurate capacity planning

Mostly based on computing power, and not storage

03 Scaling storage

Independently from memory and CPUs

04 Remove unnecessary capacity

Remove capacity that has been added to support long-term retention

05 Faster broker startup

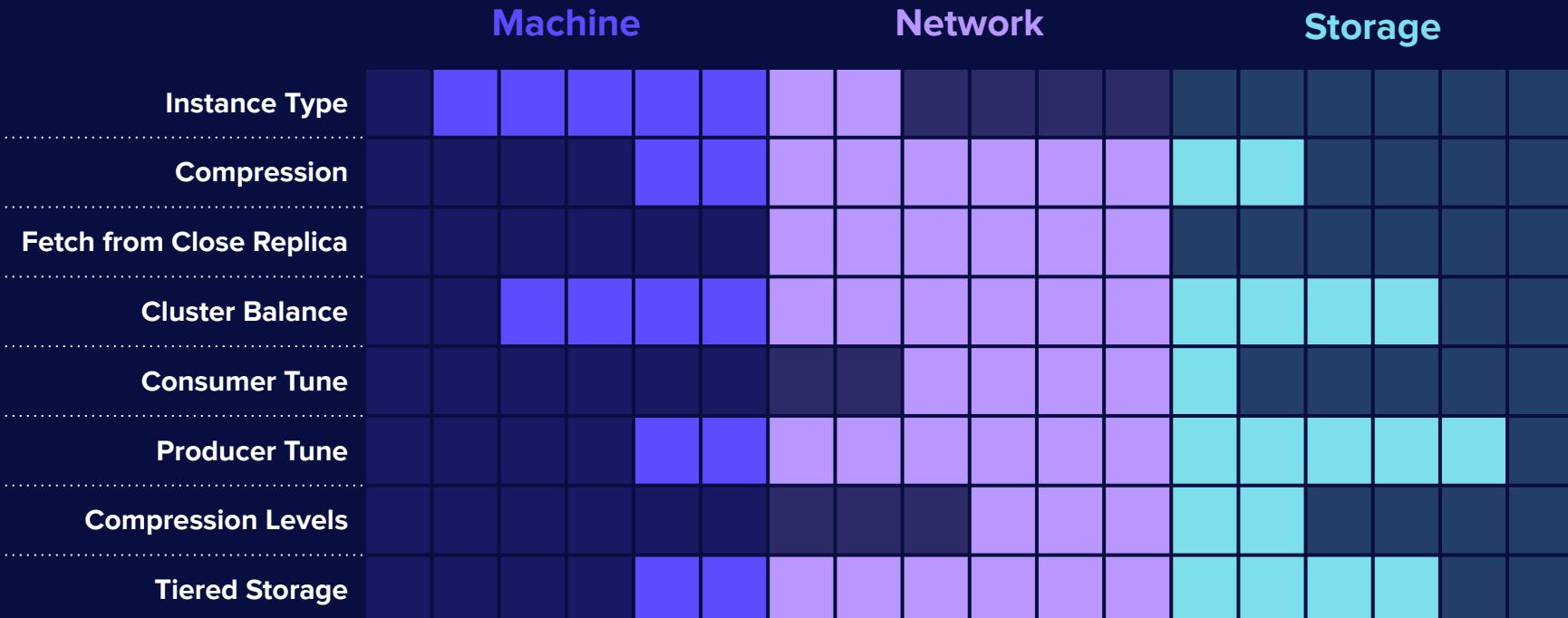
Loading local (and fewer) segments on broker startup





The **TL;DL** (Too Long ; Didn't Listen)

Cost Reduction Areas





Thank You For Your Time!

🌐 <https://leevs.dev>

🐦 @eladleev

linkedin.com/in/elad-leev