Everything you wanted to know about Kafka Monitoring*

**But were too afraid to ask*

LENSES

Contents

Intro - why you should care

Apache Kafka is a popular and powerful component of many modern data platforms. But fitting it into a production environment can be pretty complicated.

Kafka often starts as a pilot supported by a few sharp developers to support a real-time data integration use case or for a new microservice-based application.

However, to grow platform adoption to more users, use cases and applications, the need to socialize enterprise-level visibility and monitoring across different teams becomes a must.

The struggle is real. Kafka doesn't just provide a technical challenge. To get enterprise-wide adoption of Kafka, you will need to get buy-in and win the hearts and minds of multiple teams which means integrating with their tools and giving them visibility.

tl;dr

That's why we are here sharing this, and doing what we do with Lenses.io.

Our tech writers and engineers know how hard your job is. So if your Kafka project has turned into multiple projects, or business expectations have exploded, this is your guide.

Here we cover observability and monitoring of Kafka. We discuss the data and what to make available to data platform teams and tenants (such as analytics and engineering teams) as well as external supporting teams (such as security) that need access.

We will also touch on a new process, one that removes the barriers to success inhibiting people, data and apps. Those in the know call it DataOps. This list includes many devs, ops people, pundits ... people like you.

Even the industry analysts are in on it.

Why modern data platforms need observability

You can't manage what you can't measure. It's funny that this "management consulting" concept used in performance management of people applies to modern data projects. Observability is a core component of every data project. But wait, there's more ... observability goes beyond monitoring metrics and traces.

Developers need to determine the performance characteristics of their apps, the impact on upstream and downstream services, and get notified when services are not operating as expected. When done effectively, observability reflects monitoring and collaboration, sometimes with as many as 5 different teams.

When IT (ops) and non-IT (security, audit, business, dev) share insights on the activity and health of their data platform and data - the outcomes are amazing and benefits far reaching - the gifts of a DataOps approach.

Benefits include:

- ✓ Fewer surprises!
- ✓ Less inbound requests from different teams
- ✓ Fewer late-night war rooms
- ✓ No more ticket pingpong or blame games
- ✓ Going to production faster
- ✓ Less risk of security breaches
- ✓ Higher standards of data ethics
- ✓ Cleaner data
- ✓ Happier, more collaborative colleagues

Visibility and collaboration across multiple teams (often a complex set of stakeholders) is essential for increased productivity, and to give the organization confidence in working with Open Source projects such as Kafka. DataOps is your road for this journey.

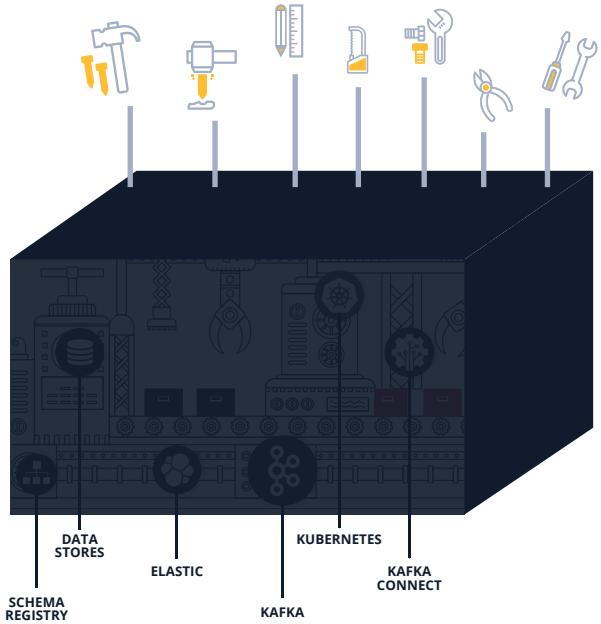
A modern data platform doesn't run on Kafka alone

Your Kafka environment does not live in isolation. It is typically held up by peripheral supporting infrastructure services: other data stores (such as Elasticsearch), workload orchestration such as Kafka Connect or Kubernetes and other services such as keystores. These different components make up a modern real-time data platform.

Then, to make monitoring more complicated there are the flows and microservices deployed on the platform.

The challenges are compounded by the high service levels and criticality of data often associated with projects on Kafka.

For healthy adoption of a data platform across a business, good enough just won't do. Tenants of the platform will demand visibility and high service levels with little room for downtime, or performance degradation. And it's kind of hard to make this happen when they live in a Black Box.



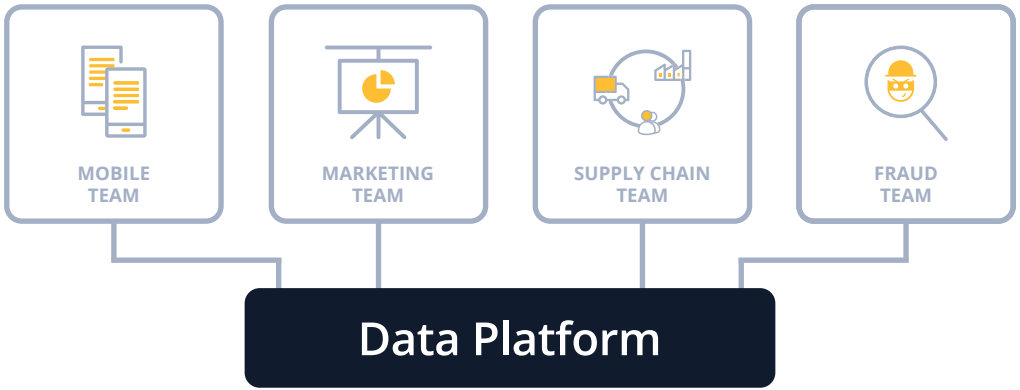
There's more than Kafka in this Open Source Software Black Box.

If only it was just data you needed to manage

You are dealing with a minefield. Sometimes you have a map, other times you are winging it. In addition to the data in your modern data platform, you are canvassing complex internal processes, teams and tools - all of which support strategic initiatives across many lines of business. Your platform and monitoring must integrate with a company's sometimes crazy organizational structures, tools and processes.

In large organizations, the platform is likely to touch dozens, maybe hundreds of internal tenants or clients, each using a different set of tools and different processes. Some of these are more demanding than others. However, each of the components needs to be productive, self-sufficient and compliant.

Integrations are key, and so is collaboration. In fact, over-communicating rather than under-communicating will cement the right friendships.



Tools, teams, processes, oh my!

Each internal customer will require a monitoring service to be delivered by the platform team. But they may also have their own teams and tools to monitor their business services and processes - from an operations, security and governance perspective.

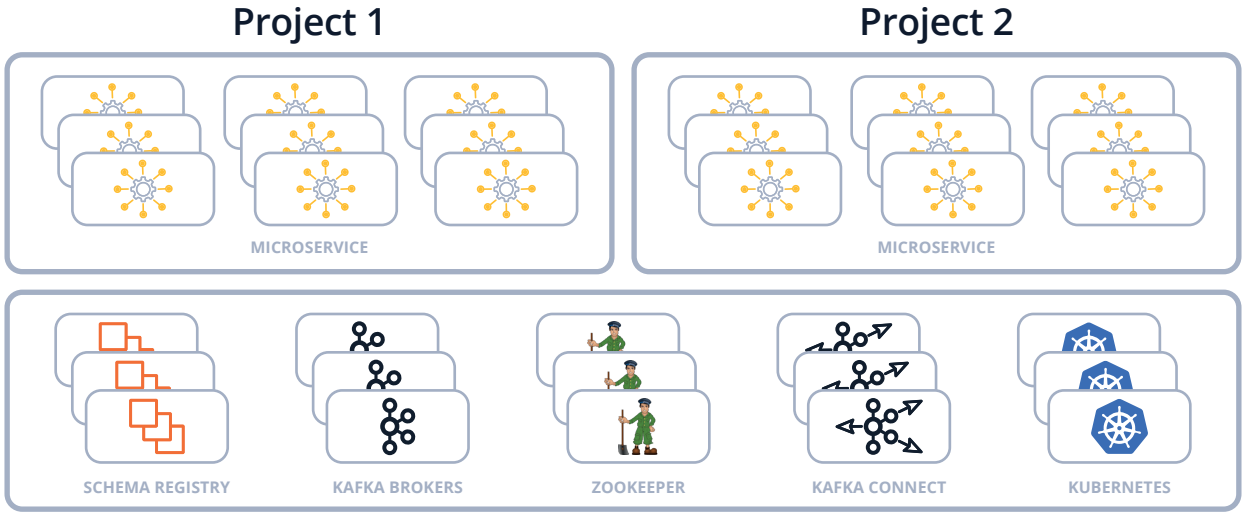
A successful platform must therefore have three things: visibility and monitoring for its own platform team delivering the service, socialization and visibility of this platform across different lines of business (LoBs), and the ability to integrate aspects across external tools and solutions.

Modern data platforms are highly complex

Distributed applications and systems are notoriously complex and it's impossible to predict what to monitor. This has driven IT teams to adopt observability practices and tools. Supporting such services is less about preventing problems and more about having data (lots of it and of different types) to be able to quickly respond to incidents when they occur. And they will occur.

As a core component of a modern data platform, Apache Kafka is comprised of multiple distributed solutions. Each solution represents another node, requiring instrumentation, monitoring and management.

The complexity and distributed nature of Kafka introduces an increased number of failure nodes as well as attack surfaces for security threats.



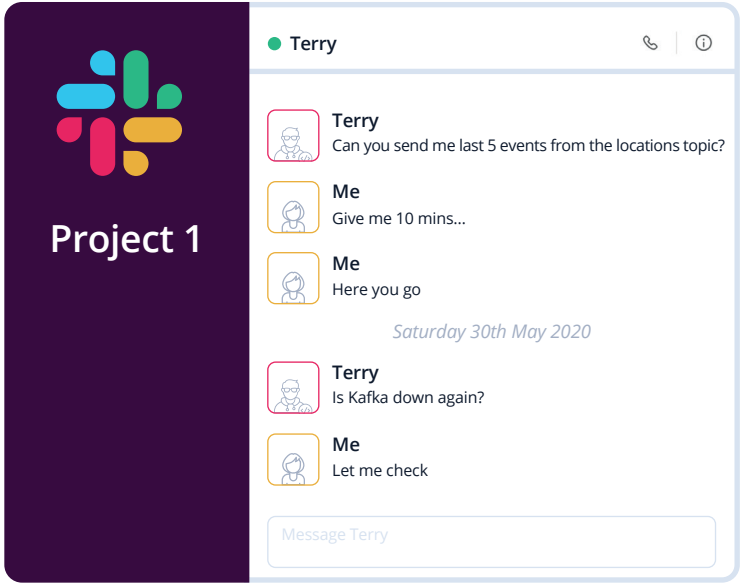
It’s deeper than IT operations monitoring

Once you start making your modern data platform work, everyone wants in.

Data platform Site Reliability Engineers’ (SREs), centralized ops and infrastructure teams as well as product DevOps teams will all require visibility into your data platform and deployed flows.

You are now the source of data for monitoring & observability, incident investigation/management, postmortem investigation/problem management as well as capacity management.

And your new platform comes with new expectations that need to be managed. Your monitoring posture must go deeper than ops and provide visibility to a range of different teams.



What about other engineers?

Your fellow engineers will require timely access to data relating to the platform and your flows to maintain productivity, service levels and code quality.

If a real-time application is deployed, they will want instant feedback to see it deployed (such as data flow topology) and see how it's operating (performance).

They are also likely to need direct access to the streaming data processed by the applications in order to investigate a production problem, for testing and development of new flows and applications.

Meet Terry

And don’t forget about your security team

We all know you can’t ignore the security team. Security operations and DevSecOps teams must ensure the continuous availability of systems as well as integrity and confidentiality of data. They do this by continuously monitoring the security posture, investigating and detecting threats and responding to security incidents if a threat is detected.

For a security professional, it’s all about having rapid access to data and collaborating with different teams. When threats are detected, they need to triage, isolate and then respond and finally remediate. This all requires lots of data for fast, forensic investigations.

Be prepared for them to use their own set of specialist security tools such as SIEMs. And just hope you don’t get a call in the middle of the night because there’s been a breach.

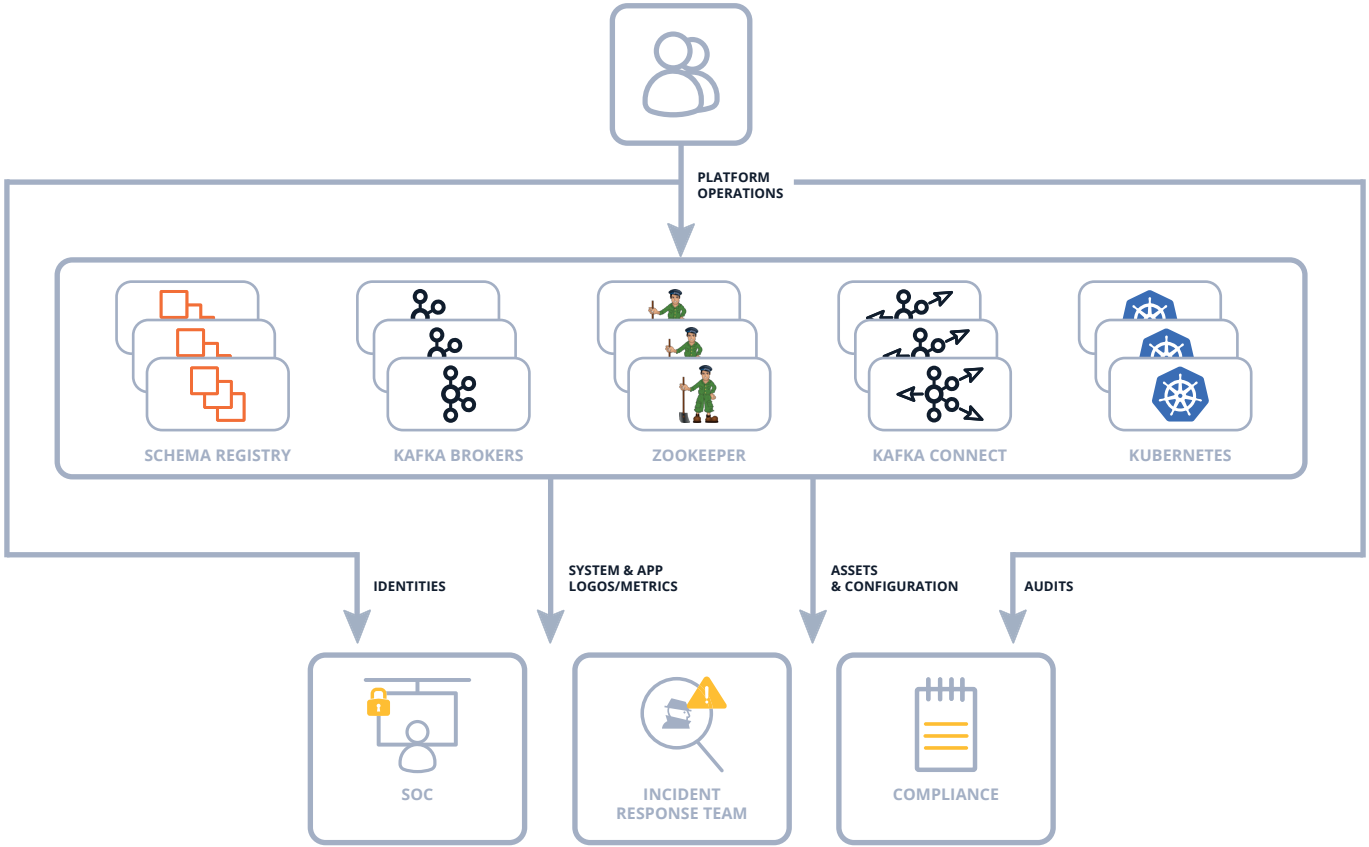
… or compliance

There’s risk associated with business applications running on a data platform. Risk levels can be lowered by ensuring security controls are in place. As an example, a business service owner must have guarantees that data within the data platform cannot be tampered with in any way to protect data integrity. Another example is controlling employee access to data from different applications and/or hashing sensitive strings of data to protect PII.

To ensure this the service owner and compliance officer must have real-time reports and alerts on user and system activity.

Some compliance may be internal compliance, some might be imposed by external regulators such as the SEC and ESMA in financial services.

Integrating with security & compliance teams



Last but not least, your internal customers

Where DevOps will already know a thing or two about streaming data microservices, marketers may well respond “Kafka, you mean the philosopher?”. These are the folks from the business units, sometimes IT, and security - and they all require real-time visibility.

They trust a data platform team to deliver a service, protect their data and notify them of what’s going on and who’s accessing it.

But this shouldn’t be through a ticketing system. If it is, run for the woods.

This is where another facet of DataOps comes through shining. DataOps principles promote self-service access, and real-time monitoring and alerting.

It’s not just the developer and analyst end users. Product owners and even business users will want to see how their data and applications are performing and that service levels are met.

Tenants might also require access to metadata (through a data catalog) as well as visibility into what applications they have deployed and other services consuming their data (topology).

All this insight must be delivered in a safe and compliant manner (of course).

Data

Dashboard: Mobile Team

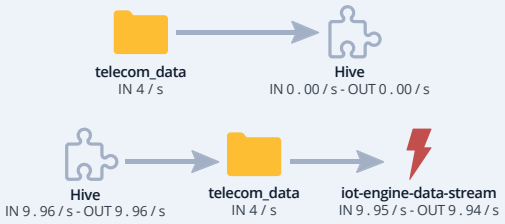
DATA CATALOG

Data Schema Monitor

```
1 SELECT Device, City, Country
2 FROM locations
```

Key	Device	City	Country
178287			
178288			
178289			
178290			

DATA FLOW TOPOLOGY



RECENT AUDITS

All resources | All actions | All users

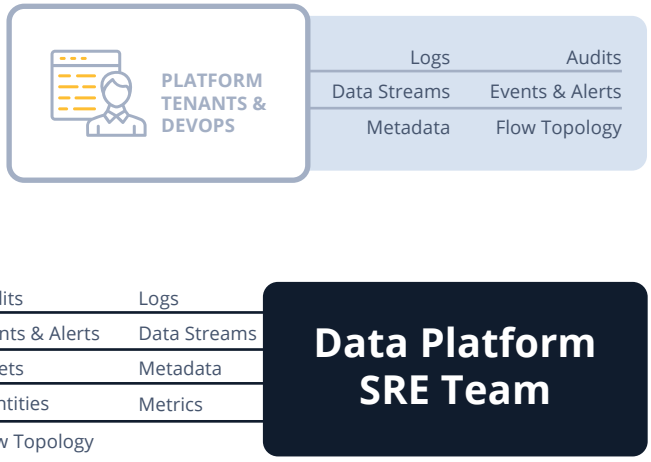
LOG	USER	TIMESTAMP
User Access logged in		
Locations Topic Queried		
Topic Created		

SERVICE LEVELS

Passed	Passed
Passed	Failed
Passed	Failed
Failed	Passed
Passed	Passed

Ok. This sounds doable. But what data do I actually need to get started?

Remember, it's more than just operations. The data you'll need goes beyond just that related to the infrastructure.



Modern data platforms start with streaming data

Streaming data is the lifeblood of an organization and represents the activity of your applications, transactions and business.

Almost all users across an organization may need access to this data for some reason or another. Developers need it to test a microservice. The Ops teams need it to trouble-shoot a problem in a flow. Data stewards to verify data quality. The Security teams need it for an incident response to an ongoing breach.

But giving data access to this many people can be a challenge. The volume of data can be huge. It can be serialized in proprietary format (AVRO, Protobuf etc.). And hold sensitive information, so not something you would want to expose even if you could.

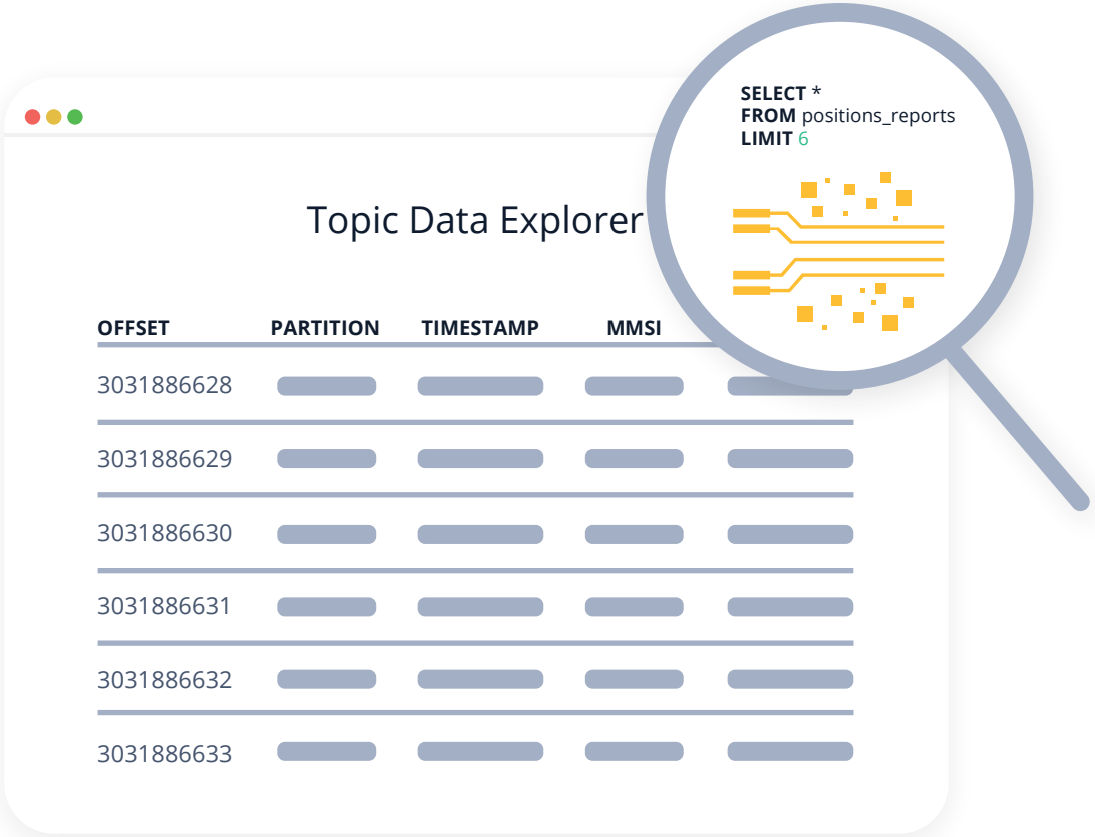
Sending all streaming data to a data store such as Elasticsearch to be queried could lead to high costs and isn't going to be pretty.

This data is best queried directly off the stream.

Since this data will interest almost all types of users across an organization, access should be protected with role-based security, field-level data masking and auditing.

Of course, giving access also means the data must be queryable in a way that is very familiar across the org (such as with SQL).

Access through a command line and using python just won't cut it. Those days are gone.


SELECT *
FROM positions_reports
LIMIT 6
Topic Data Explorer
OFFSET
PARTITION
TIMESTAMP
MMSI
3031886628
3031886629
3031886630
3031886631
3031886632
3031886633

Say hello to your metadata

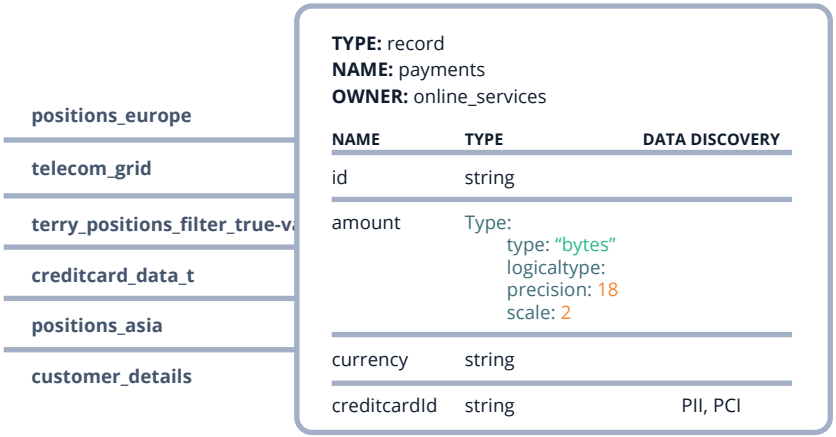
A successful modern data platform project will cover many different applications, use cases and streams.

If you incorporate DataOps principles they will lead to faster delivery of applications. But as more and more applications are developed, it will be harder to keep up with the data generated and apply data governance.

A lack of visibility into your metadata, having no data catalog, will impact the productivity of engineering, data science and data analytics teams. And you can wave goodbye to your good intentions with data governance.

The absence of a data catalog will make it difficult to onboard new teams and users. These users won't understand what data is being produced or who the owners are.

A data catalog is critical for any modern data platform and a key component of DataOps.



Don’t leave out logs

Logs are a crucial data source, particularly for the later stages of an investigation and post mortem phases of an incident.

Logs will be needed for both infrastructure (everything from system-level logs to Kafka components such as the garbage collection logs on the brokers, Kafka Connect worker logs etc) and application tiers of the data platform.

Logs should be accessible via a browser to all users and tenants.

...Or miss metrics

A key piece of the monitoring & observability of the application, flow and infrastructure layers, metrics are collected in real-time and then archived for capacity management.

As with logs, this should cover both infrastructure (such as CPU, Memory etc) as well as application tiers (such as JMX metrics from Kafka components).

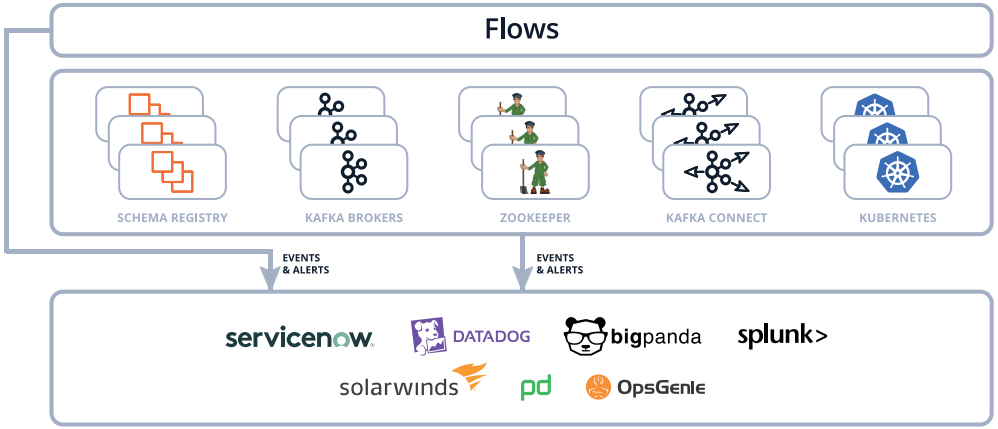
Typically stored in a metrics store or APM or time-series DB, metrics should also be archived for capacity management purposes. You’ll need plenty of metrics to hand for good observability.

Not to mention events & alerts

With multiple tools to monitor different parts of the stack, alerts represent state changes of different components. An example is a connector in Kafka Connect being stopped or a monitored component such as consumer lag exceeding a certain threshold.

Chances are, you'll need to route alerts to many different teams: the platform SRE team who are on call , tenant operations team of the impacted service as well as your centralised service desk, NOC/Ops bridge.

Events related to deployments from CI/CD tools also must be available for triage phases of an investigation so that if service levels of an application go down, they can be linked to a recent deployment.



Save tears with data flow & service topology

When onboarding multiple teams and projects onto Kafka, keeping track of data flow dependencies can make you dizzy.

By dependencies, we mean how data is consumed and processed by applications from one Kafka topic to another topic or from one topic to an external system.

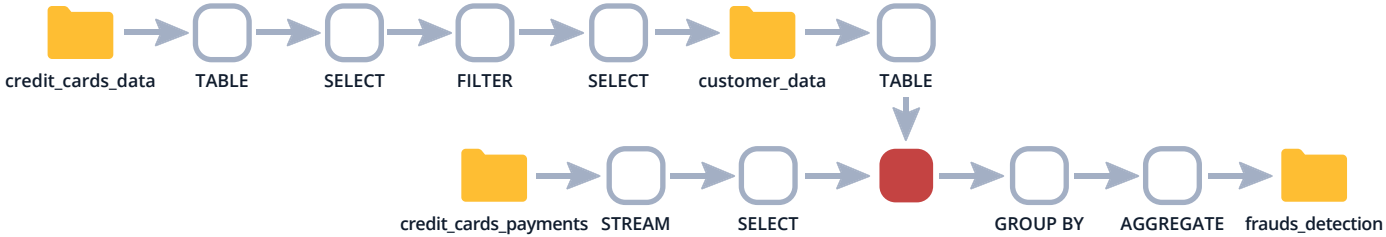
For operations, this visibility can be important when troubleshooting an availability or performance issue with an application and understanding what upstream services deliver data to the failing application.

The visibility is also crucial when identifying impacted services for a planned downtime. Turn your back for one minute and you'll find a critical service has been deployed that consumes data from non-critical services. That can only end in tears.

Data lineage and data provenance is also an important requirement to meet data compliance controls, including those imposed by external regulators such as with the GDPR.

Some compliance controls ask that you have an understanding and process to review logic within a particular application. MIFID II is an example of this. Visually representing the underlying logic of an application will be a great deal easier than reviewing Scala code.

More transparency over data flows and the logic of applications processing data will not only increase service levels and improve application quality, but will increase data ethics.



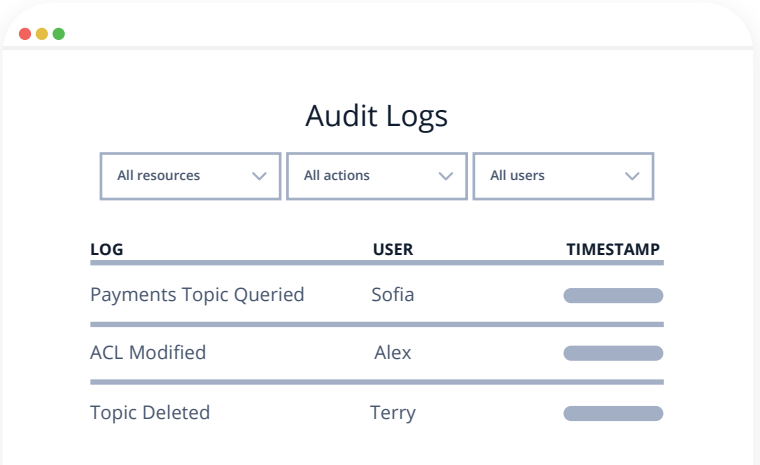
Avoid incident panic with audits

For operations, like logs and events, audits can be the missing clue for incident investigation and postmortem phases of an incident. They can help identify who or what changed the environment that may have led to a problem.

An example audit event would be: Terry from the back-end team deleted the dev_engines topic on Monday. No wonder the application went down!

But audits are also an important source of information to capture people with slightly less good-intentions than Terry. Security operations teams require audits for threat detection (including user behavior monitoring) and incident investigation. These are required by teams in real-time and should be exported out to a secure location as quickly as possible in order to avoid risk of tampering.

Compliance teams will also need to keep audits for real-time compliance reporting and ad-hoc auditing.



Help engineers see state & configuration

A Kafka data platform environment must be continuously tuned to cater for different workloads. Trust us when we say this can be a nightmare.

Service performance and data platform stability can be impacted through a poorly configured environment.

Developers need an understanding of the state & configuration of the platform and application landscape (such as topic partitioning and topic retention settings)

Configuration information of your environment must therefore be accessible to all those developing or maintaining flows: developers, QAs, SREs etc. Otherwise, brace yourself.

PARTITION	LEADER BROKER	TOTAL RECORDS	TOTAL BYTES	OFFSETS EARLIEST/LATEST	REPLICAS
0	2				3 | ✓
1	3				4 | ✓
2	4				5 | ✓
3	5				1 | ✗
4	1				2 | ✓
5	2				4 | ✓
6	3				5 | ✓

Avoid nightmares with asset discovery

Development, operations, compliance and security teams all require visibility into assets that compose the data platform.

This includes:

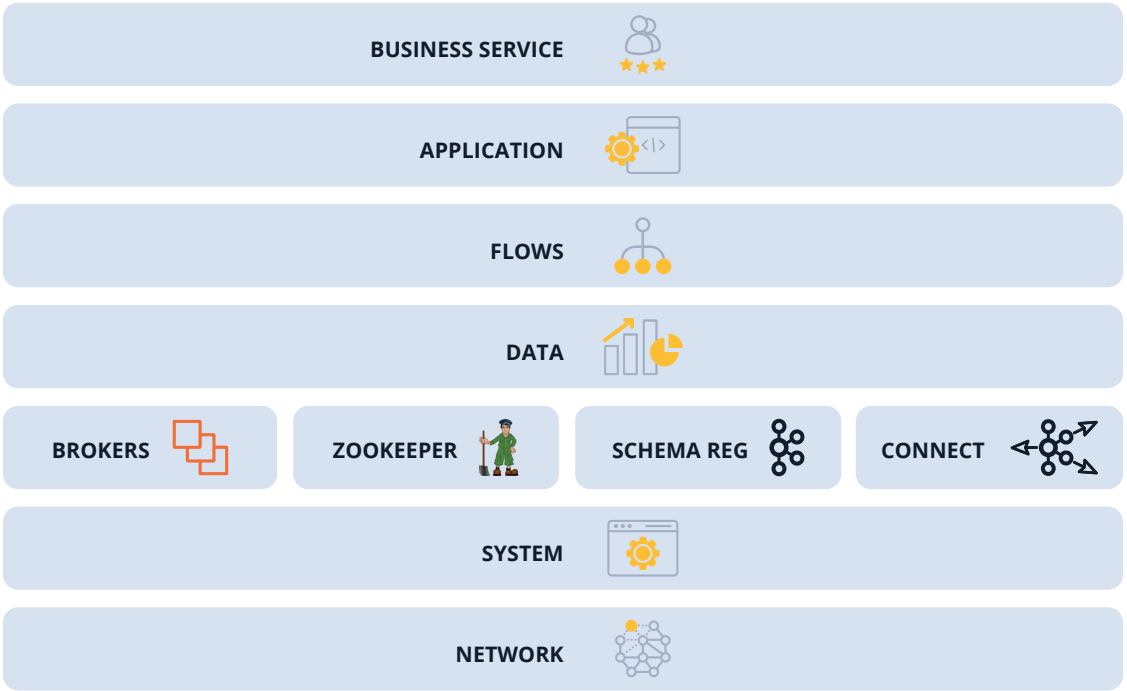
- ✓ Infrastructure (such us Apache Kafka Broker hosts)
- ✓ Flows and applications
- ✓ Identities
- ✓ ACLs
- ✓ Topics

This data should be accessible to relevant tenants and business units both as a secured UI and also via accessible API calls.

It is commonly held in application catalogs, asset tables, configuration management databases (CMDB) or as infrastructure-as-code within a source code repository such as Git.

Like with a data catalog, teams need fast access to information about assets to ensure best practices and compliance are maintained.

What components should I monitor?



Network matters

Despite many efforts, like many distributed systems, Apache Kafka isn't immune to poor network performance.

Large volumes of continuous data streams, potentially from tens of thousands of clients will flow. This data may then be further replicated across different Kafka brokers. Network therefore really matters.

Network latency in particular can destabilize the environment and cause Kafka to under replicated partitions. But it's not just between Brokers. Zookeeper must have good connection with the Brokers in order not to appear dead.

You'll want to keep on the Network Pool Usage for the network card of each Broker. If you exceed available threads in the pool you can expect latency in data being consumed/produced.

So do your System resources

For Brokers, memory will be the most important resource to monitor. CPU can be intensive in certain situations (such as if you're compressing data).

High time writing to disk & IOPS may impact data being consumed and produced. It goes without saying that you don't want to be caught with zero disk space.

For Zookeeper and Schema Registry, these are services that are relatively light on system resources. For Kafka Connect, memory and CPU will be important.

Kafka Services 

All Kafka components will expose metrics via JMX. What needs to be monitored most exactly depends on the component.

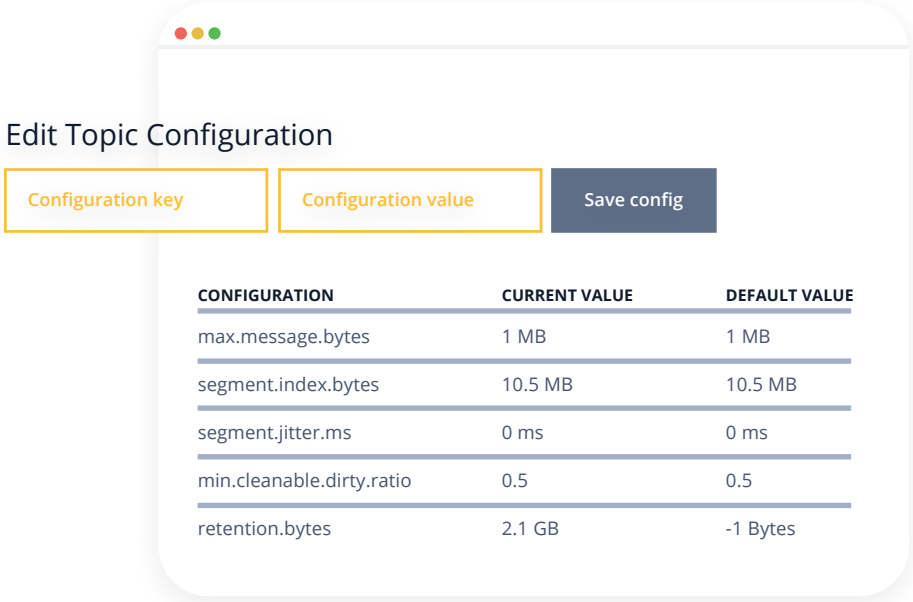
For Kafka Brokers and Zookeepers, the biggest symptom of a problem will be if there are any offline partitions or if for any reason there are two active controllers (there should always only be one).

Monitoring memory and garbage collection, particularly on the Brokers will also be very important. Ensure you're keeping an eye on the Broker logs to look for Out Of Memory dumps as well as the GC logs to view the GC latency.

If you're using Kafka Connect you'll want to monitor the workers (the JVMs running on your Connect nodes) are running as well as the state and throughput of each task.

Exploring the task and worker logs will show stack traces of non-running tasks is indispensable. But don't expect a tenant of the data platform to be able to grep these, they need to be easily accessible to everyone via a browser.

Kafka Connect will expose JMX metrics for Workers, Tasks and Connectors, these can be useful for observability. Kafka components are usually distributed so the configuration needs to be continuously tuned by platform teams and developers. So make sure conf is accessible to all parties concerned.


Edit Topic Configuration
Configuration key
Configuration value
Save config
CONFIGURATION
CURRENT VALUE
DEFAULT VALUE
max.message.bytes
1 MB
1 MB
segment.index.bytes
10.5 MB
10.5 MB
segment.jitter.ms
0 ms
0 ms
min.cleanable.dirty.ratio
0.5
0.5
retention.bytes
2.1 GB
-1 Bytes

Flows

Flows, also referred to as pipelines, underpin the flow of data across different systems, application components and infrastructure.

Monitoring flows includes monitoring the performance of producers & consumers, topics and partitions.

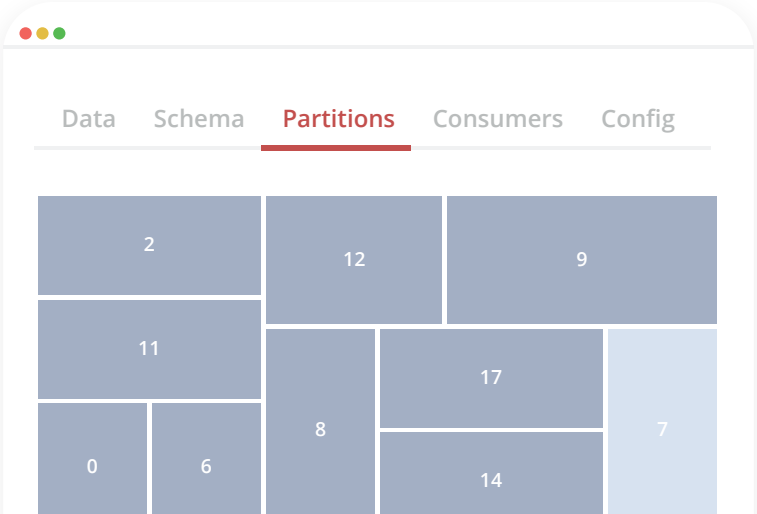
Consumer lag represents how far behind a consumer is from reading the last event in a partition. It's normal to have some lag. But what's not normal is if there is significantly more lag for one consumer reading from one partition in the same consumer group reading from other partitions for the same topic.

This is where keeping track of how your data is distributed across your different partitions will be important. They may depend on how you've keyed your partitions. There may be a large imbalance.

For partitions, monitor the health of under replicated partitions for each topic. For the topics, if you start seeing high lag, make sure you can track the records in and records out, it could be you have a flood more events coming in than you're used to.

Visualizing your pipelines in the form of a topology map: the deplency between topics and applications will be a god send when you're trying to troubleshoot.

Finally, remember that teams will always need to see what the configuration settings are of flows, this includes for example the configuration settings of topics.



Application

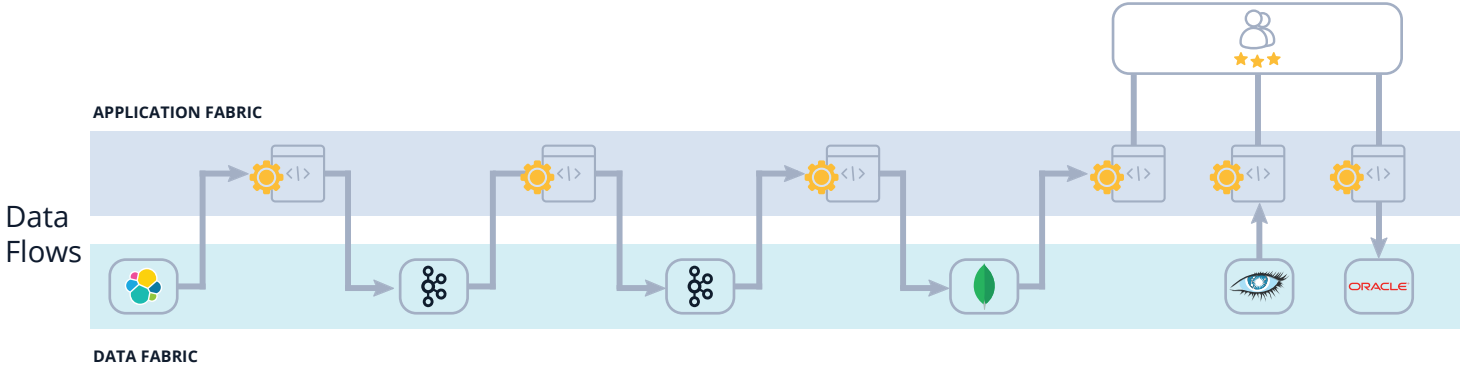
This is where having easy access to logs, metrics and traces will be important. If it's a microservice and distributed application, the failure modes may be too large to prevent failures so the key here is for teams troubleshooting applications to be able to drill down to understand the other flows deployed on the data platform.

Business Services

Business Service monitoring provides a highly abstracted real-time visibility into the health and performance of business services and processes.

The data is presented in the context of business processes.

Monitoring allows business teams to measure SLAs and quickly identify problems. That way, tenants of the data platform can drilldown from high level business service down to the application and flows. It's likely metrics from the flows themselves should be populated upwards into a business service dashboard - such as message throughput.



Data 

The final and most critical piece. You need to ensure a healthy flow of data between applications and systems. What we call the data fabric.

If you have flowing data, it's clean, chances are you have a healthy business and well running applications and processes.

Understanding how clean it is requires giving people who own and understand the data access to it with the power to explore it. Such as exploring the values or cardinality of a certain field.

This doesn't just mean access to the data stream, but also the metadata. If you're using schema registry, ensure teams can access schemas and that any changes are secured and audited of course. Or even better ensure teams have access to a data catalog that provides visibility of all data sources across your entire data platform.

Ultimately, everyone in the business should be able to view and understand data in a self service manner and with a common language such as SQL.

But you need to be able to put controls and auditing and masking in place. Do this correctly and it will spark a digital revolution like you've never seen.

Time to make data everyone's business

Fighting fires from a Kafka command line doesn't have to be a reality for your team. And even if *you* find it a compelling edge case - it won't fly with your friends in security operations or infrastructure; and it's a resounding 'no' from developers like Terry.

Lenses.io provides you with a portal into your real-time data platform, turning open-source technologies like Kafka from a black-box into a DataOps workspace for everyone from DevOps engineers to data analysts.

Even Terry will like it.

Get visibility into your data platform.

Start for free at **lenses.io/start**

LENSES