2024

Digibee

Monitor usability enhancements

View on monitoring

Our objective - Enhance Monitor’s overall experience

Why are we doing it?

After interviewing several **users, stakeholders and going over a vast desk research and competitive analysis**, we understood that we’re lacking behind several aspects that are hindering **Monitor’s full potential**, they are:

- **Objective troubleshooting**
- **Data autonomy**
- **Observability**
- **Proactive Monitoring**

Methodology

In order to reach our current state, we walked through a long path. Here are some of our steps and how we went by each one of them:

1. Desk Research

By digging into Mulesoft's documentation regarding all things Monitor, we sought to understand how some of the known issues we face here at Digibee are taken care of by our competitor. Not only that but we also went after other competitors so we could trace a line of what is, apparently, "effective".

2. Side by side competitive analysis & Benchmarks

Although we wish to explore and engage into new features, we had first to work over what we already have and improve upon that before heading into something brand new. From this line of thought, we listed everything we do at Monitor and brought at the side, what Mulesoft does correspondingly at their side. It not only gave us some hints on what to improve but also gave us some confidence because although Mule is more robust than Digibee, it's interface and usability is lacking behind in some aspects.

3. Interviews with Key users and Stakeholders

After covering what we could gather by studying the material at our grasp we went after those who were more in contact with the main pain points and issues that needed to be addressed at Monitor. At the end of the interviews we could also link our new findings to the previous one we had and then, trace a more reliable path.

4.Jobs to be done

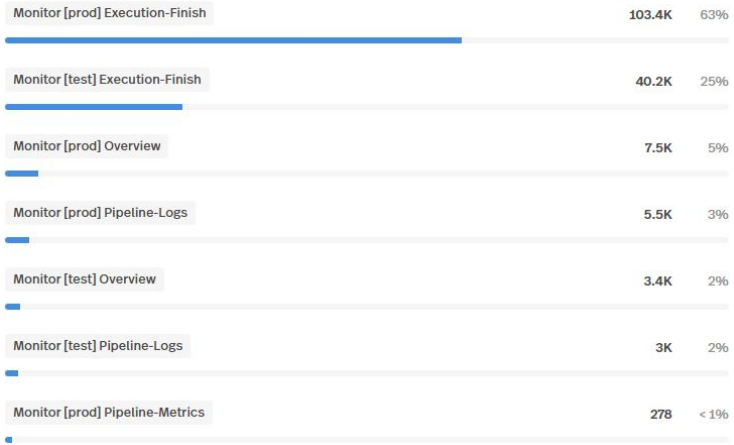
After delving into Mulesoft, and reassessing Digibee's main gaps at Monitor we structured a simple, but firm perspective of what had to be done in order to achieve the main points discovered.

Why improve Monitor's Experience?

Completed executions	Pipeline logs
18 minutes	24 minutes

The average time spent on the most utilized pages at Monitor elapses at a total of 42 minutes

Mulesoft's target regarding troubleshooting is **30 minutes.**



Conversion Rate

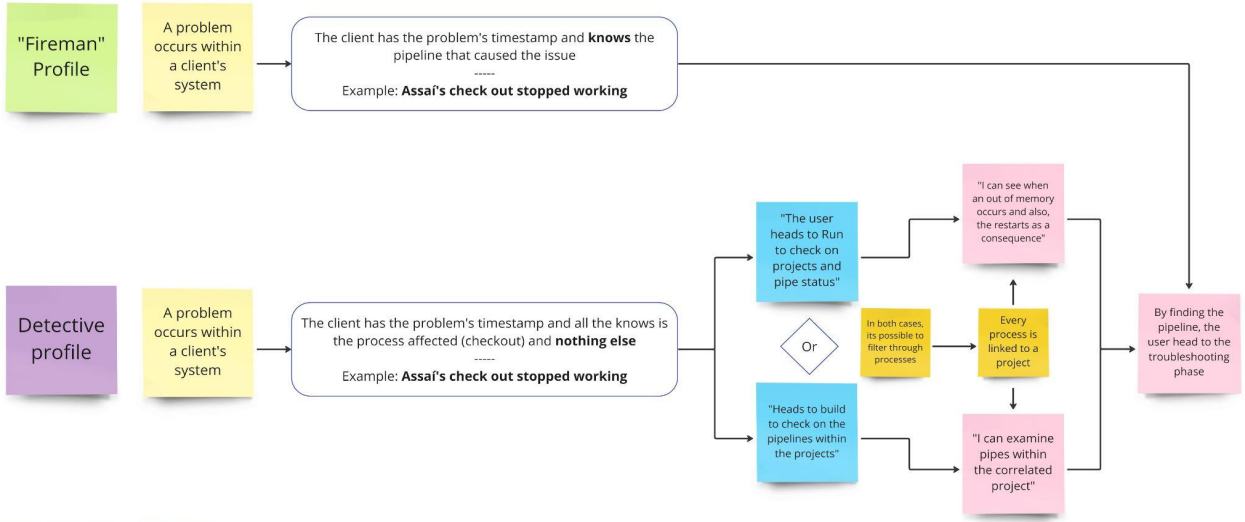
41.92%

Users	Steps
637	5

Source:
https://app.fullstory.com/ui/191BCH/metrics/details/KqEnmK4BAPQ6?fromDashboard=6407864933535744&fromCard=5019802572304384&completeSessions=false

Why improve Monitor's Experience?

We understand that users **want** to only **find a problem, quickly and easily**, but also, have exactly the **necessary tools and steps** at their disposal. The whole process must be intuitive and objective.



Currently, users can't have a **whole picture of the integration**. Therefore, they have to go out of their path to find what they need.

- Filters does not **persist** when users change **menus**
- It's not possible to see the **timestamp** being worked on
- There's no **pagination**
- There's no **observability**
- It's not possible to grasp the integration's **health** (Our pipeline metrics page can only show 1 pipeline at a time)

Delving into the research

After a brief analysis its possible to understand that the **14 missing features** are, generally speaking, built in order to prove user **autonomy and simplicity** over the troubleshooting process as a whole.

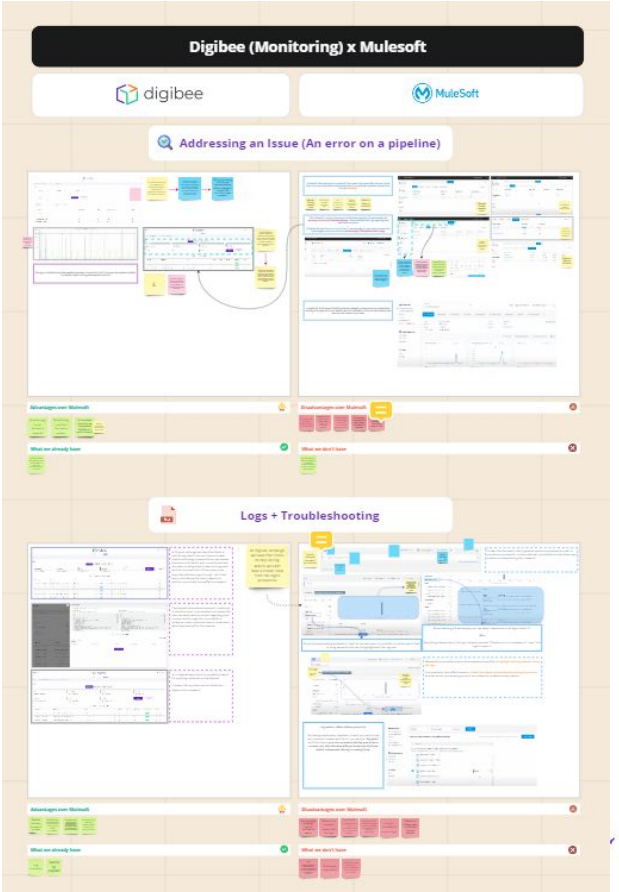
	Features	Mulesoft	Digibee
Environment overview	Displaying information about performance and general behavior of all pipelines	x	x
	Metrics displayed	8 metrics	7 metrics
	Export data as a CSV	x	
Dashoards	Pre-built dashboards	x	x
	Charts covering metrics	80 metrics	7 metrics
	Export data from chart to a CSV File	x	
	Save a Pre-build Dashboard as a Custom Dashboard	x	
	Custom dashboards	x	
Metrics	Choose metrics to be displayed	x	
	View performance metrics for supported connectors	x	
	Export metrics to external tool	x	x
Alerts	Configure alerts	x	x
	Metrics to be alerted	9 metrics	7 metrics
	Alerts for Graphs in Custom Dashboards	x	
	Event-Driven Alerts	x	
Log management	Search and filter logs	x	x
	Share log	x	x
	Save and View Saved Log Searches	x	
	Generate logs for apps and APIs in real-time and without writing code	x	
	Download archived log data (raw data) collected	x	
	Export Log to external analytics tools	x	
	Display transactions, events and metadata in one place	x	
	Real time graphical representation of how all pipelines connect/integrate	x	

Data source:
https://miro.com/app/board/uXjVNV-4gyE=/?moveToWidget=3458764568362779661&cot=14

Delving into the research

After Besides listing what we had missing, we managed to structure a side-by-side comparison in order to understand how Mulesoft deals with recurrent scenarios common to both platforms.

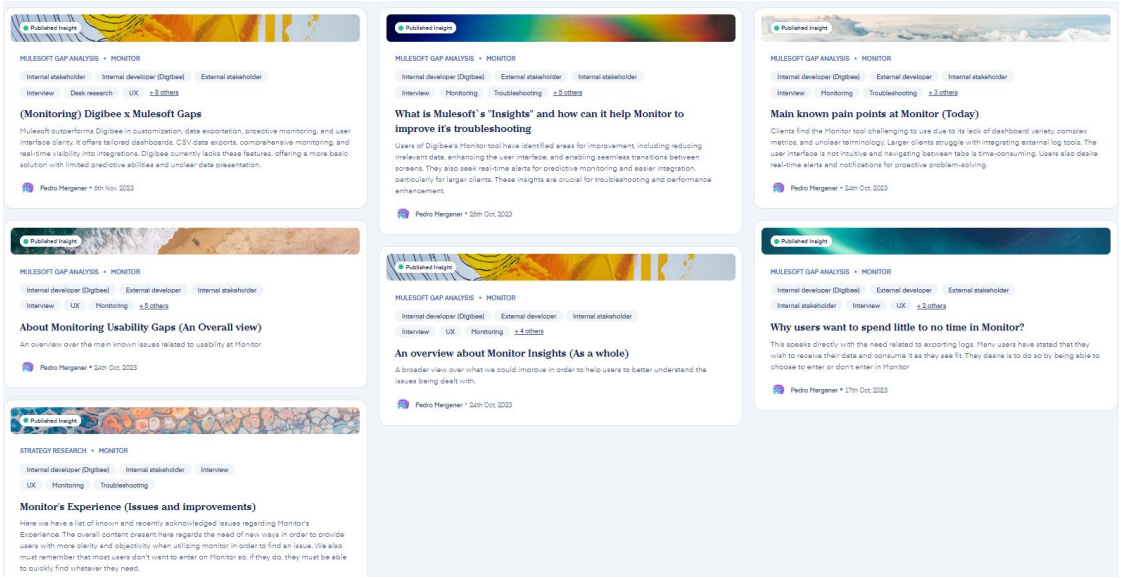
This study helped us to understand what we're currently being effective and where we're not. Also where we must invest more time and effort.

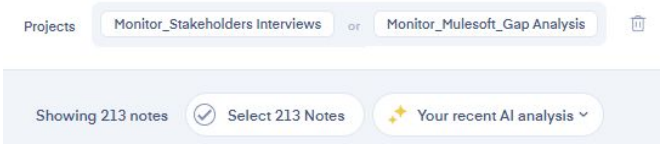


Data source:
https://miro.com/app/board/uXjVMqCaetw=/

Delving into the research

Still watching over the data gathered from more than **25 interviews**, we could then, converge everything we managed to find into **Insights** through the **Marvin** tool. This step allowed us not only to centralized our knowledge but also, to “translate” our users and clients needs into something more tangible.





Data source: https://app.heymarvin.com/projects/5243/insights/

Where are we coming from?

From the topics stated at the last slide regarding the studies conducted by the team, we managed to identify **key points** to work over in order to improve Monitor's overall experience:

"Access MY logs at anytime, anyhow and anywhere"

Since clients have paid for their logs, they want to access them when they want, as they want and wherever they need. Currently, logs are restricted by a restricted timeline and a non-optimize visualization.

If clients wish, for example, to consume their logs outside Digibee, they must be able to do so.

Data autonomy

"Isolate the pipeline's problem within my integration"

Although its possible to get from point A to point B its not only complicated but non linear and also inconsistent.

Currently, if users have to search for a problem they most of the time, have to go out of their way to simply find the pipeline.

The troubleshooting phase but me consistent, simple and objective.

At last but not least, we can't rely on observability which is a critical aspect regarding monitoring market-wide. Observability not only allows an overall panoram but also, gives users a broaders view of the integration health and what is not working according to plan.

Observability

"If a problem is about to happen I want the solution handled to me"

A perfect analogy is this case is "Health Insurance" - I don't to have to rely on it but if I do, it better be good - In other words, users want to know exactly what needs to be done once a problem shows up.

Objective troubleshooting

"I just want to deploy my pipes. But if I have a problem, I don't want to access monitor"

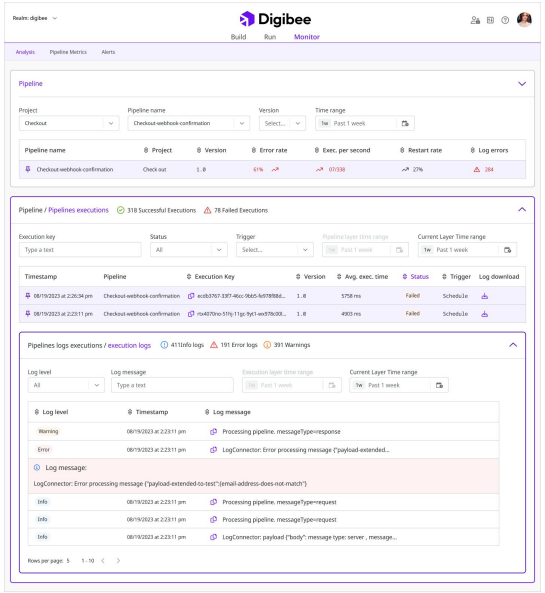
A perfect analogy is this case is "Health Insurance" - I don't to have to rely on it but if I do, it better be good - In other words, users want to know exactly what needs to be done once a problem shows up.

Proactive Monitoring

We will allow users and developers to be self-reliant on identifying and treating issues.

Objective troubleshooting

Objective on our context relates directly to consistency. A considerable amount of the time spent on Monitor relies on the alternation among tabs and the “refilling” of filters and lack of visibility over the timestamp being worked at. Plus, there's the fact that its necessary to re-enter the pipeline name every time the user needs to go back at one of the menus. These topics among others, contribute to the lack of objectivity and consistency during the troubleshooting phase



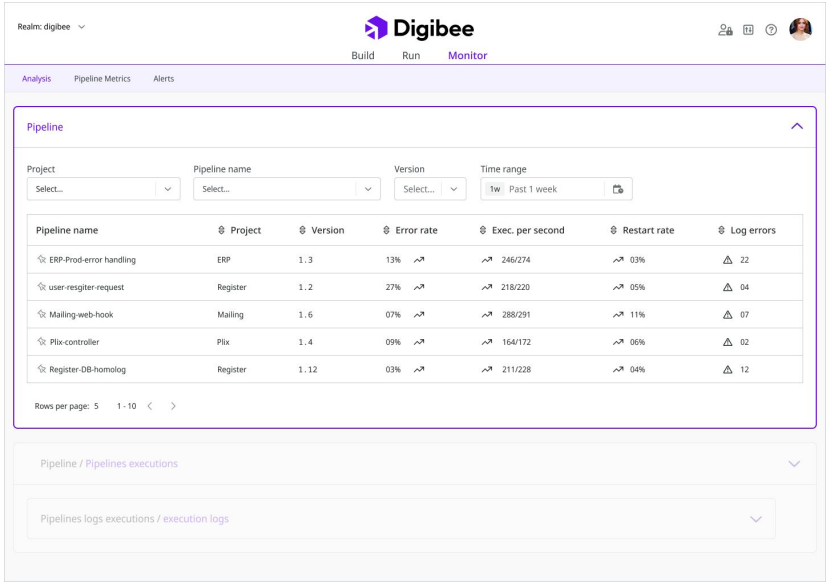
“Concentrate at the object of focus”

In our context, object stands for “pipeline”. By maintaining the object at focus we not only reduce the amount of steps but we also provide consistency, clarity and linearity.

The concept of “Layers”

Monitor has three layers: Pipeline, Executions and Logs

As state before, we will provide consistency and objectivity as our first initiative. How?



Once our troubleshooting becomes optimal, then, we can head into a broader approach in a **near future**

Why invest on observability?

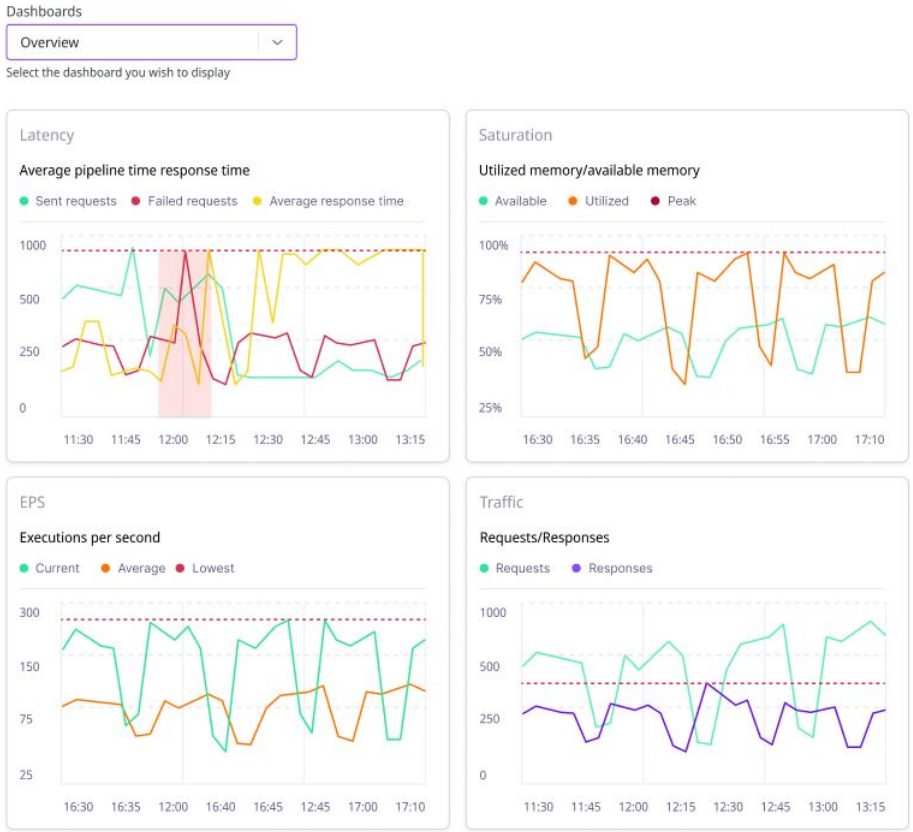
By being able to cover **critical metrics** on which the integration's overall status is reflected, its possible to quickly and objectively head to the problem since now, we would be working with "**health**" indicators.

What problems are we solving:

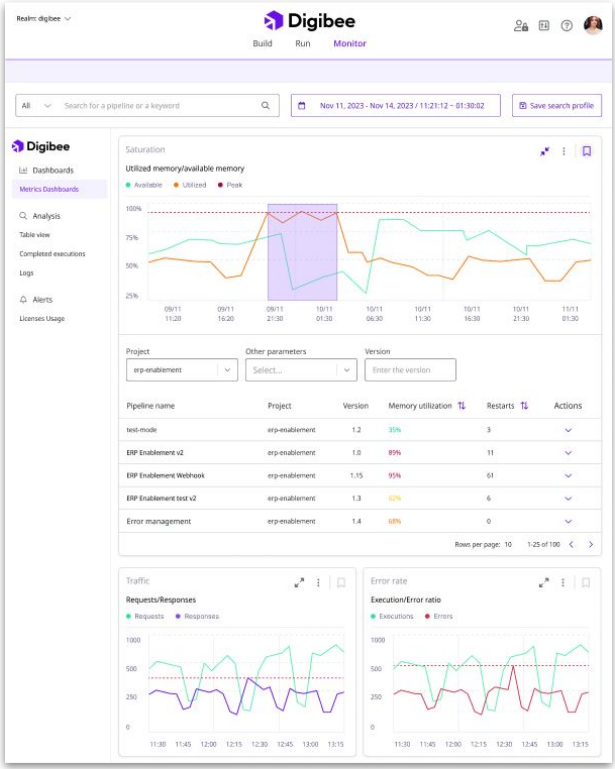
The ability to **uncover** an ongoing issue even when the pipeline is not known.

Isolating problems becomes a lot easier, independently if the **root cause is known or not.** Due to the fact that it's possible to compare **base**, **average** and **peak** values.

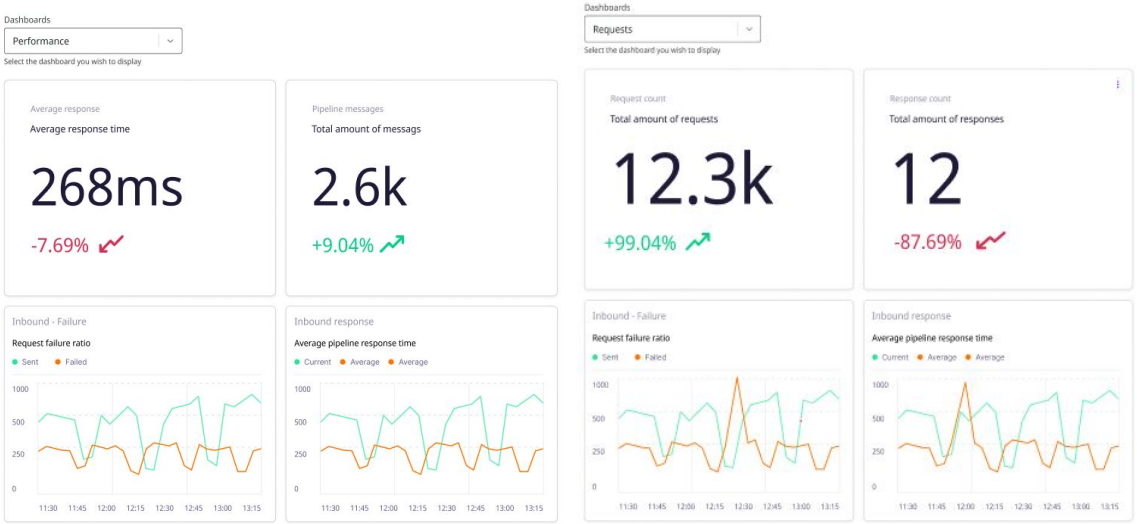
And last but not least, it reduces the time needed to search for the issue at hand.



Why invest on observability?



Still on the Integration's health subject, once we manage to provide a clear understanding of the integration's overall status and it's other indicators, users will be able to find issues regardless of knowing or not knowing where the root problem is.



Proactive Monitoring

Being proactive, means provide means for users to deal with issues on a active instead of reactive manner.

In another words, giving users, data to work with. For example:

Problem diagnostics.

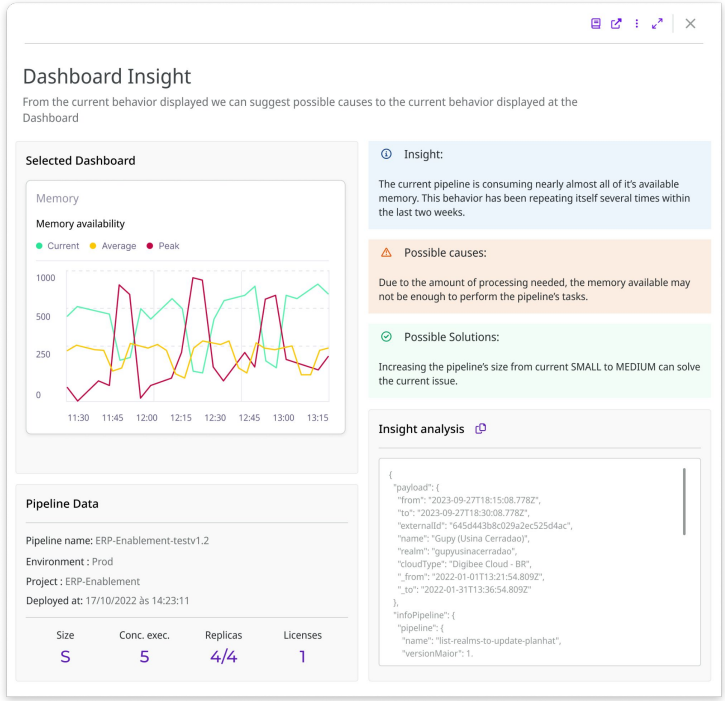
Noise reduction (By displaying only relevant data).

Being able to understand and act upon the pipeline's health

(Take us back to observability)

Having a search profile.

Safe measures to mitigate problems



It's still a concept

Data autonomy

We will provide users with the option to consume their data where, how, and when they see fit. One of the most pain points found regarding logs has to do with the limited action users have when dealing with their logs.

Data autonomy is also something that **Mulesoft as well as other competitors** see a great value since clients tend to utilize logs for several reasons going from audition up to tracking the integration's history from where its possible to tackle important decisions regarding strategy or performance.

We're currently allowing users to download the logs from the pipelines executions but we're already studying on how to provide the whole batch of logs from a pipeline.

What made us choose our current path?

Although we know the power that comes with observability we're also aware of it's complexity. Therefore, we intend to work on greatly improve our troubleshooting by utilizing the layer concept, mainly because it's easier and faster do develop and once it's tangible, it's also easier to test and understand what works and what doesn't. Meanwhile, we can utilize what we learned to tackle the next step and kickstart the next step, revolving observability.

Thanks,

[obrigado, obrigada, gracias]

Monitor Team