



Why Malt migrated from open source to SaaS-based monitoring



Hugo Lassiège
Co-Founder and CTO
Malt

ABOUT THE AUTHOR

Hugo Lassiège is the co-founder and CTO of Malt, a fast-growing marketplace with over 500,000 registered users that matches 150,000 digital-savvy freelancers with 30,000 corporate clients across Europe. He has over 20 years of experience in IT, including roles as a developer, QA engineer, project manager, architect, and agile coach.

Creating an excellent developer experience is a strategic imperative for companies to attract and retain talent, boost productivity, and ultimately release high-quality digital products. When I think about developer experience I think about two main things: (1) how quickly new team members are able to start adding value from their first day at the company and (2) removing pain points for all team members (both new and experienced). Because Malt is expanding rapidly across Europe (we currently operate in three countries), we're investing heavily in the developer experience as a key lever for scaling our team.

Two areas of developer experience have been of particular focus for us: deployment speed and performance monitoring. We've made great strides in both and I believe we've nearly solved the latter. In this article I'll tell the story of our journey to dramatically improve performance monitoring and how these improvements help our team scale.

2013-2019: Open source solutions

Malt was founded in 2013, and for six years we tried practically every open source monitoring tool imaginable: Telegraf, Jaeger, Grafana, the list goes on. The team wanted to build an open source monitoring solution that had all the capabilities of a commercial solution with none of the cost. This is an understandable impulse for any team of capable engineers, especially at an early-stage startup with limited funds. But we ultimately discovered that this view was naive. At every step of the process, our open source solutions generated complexity and cost:

- **Configuration:** Each open source tool we tried had its good points. But the problem was the fragmentation of the solution, the complexity it created, and the amount of time required to set up and maintain the solution. Open source tools have different query languages, interfaces, and methods to ingest data, and team members have to be knowledgeable about each one if they want to build working dashboards and meaningful alerts. This is enormously time-consuming and challenging to get right.
- **Alerting:** Then, once the tools are configured, are developers actually getting value out of it? In our case, we found the open source tooling generated a lot of noise. Because of the fragmentation of the solution with multiple open source tools, each tool pushed alerts to email or Slack, and it was not obvious whether there was a real problem or what the problem was. (And, often, the noise was due to a misconfiguration of a monitoring tool).
- **Incident management:** In turn, developers just stopped using the tools, which created a bad dynamic: developers asked their ops colleagues to translate and interpret the alerts and let them know if there was an important problem. This was not an efficient approach. We had inadvertently recreated the classic dev-ops divide in which devs deploy code and simply let ops teams worry about any problems it may cause.
- **Maintenance:** Finally, it's important to mention that open source isn't free. There is significant time and cost required to manage servers and maintain each tool. As a team, we gradually started to change our mindset about our monitoring strategy.

Early attempt with a commercial solution

I should mention that we did try a commercial solution at one point but it was short-lived. At that time, the pricing was not well-adapted to cloud infrastructure: the pricing model charged by host, not by container node. Nowadays, commercial solutions are more well-adapted to the cloud, but at the time it was not a good fit and it was fairly expensive. Unfortunately, this early false-start with a commercial monitoring solution reinforced our desire to build our own open source solution for "free."

2019: Bake off

By 2019 our mindset had completely changed: we recognized the hidden cost, complexity, and inefficiency of our open source monitoring solutions, and we wanted to buy a SaaS solution that centralized everything. We especially wanted the solution to be good at log management, which was a particular pain point. We had tried many, many log management tools, but still found that when we had an incident it was very difficult to find the right logs.

During 2019, we tried three different SaaS monitoring platforms. We chose to invest in Datadog. It helped that Datadog was especially helpful with our log management problem, our worst pain point.

Today: a single, centralized platform

Today Datadog is, more or less, our only monitoring tool. It's highly intuitive, which means every member of our engineering team can use it—and can learn how to use it fairly quickly. This is really powerful for us because it helps quickly spread knowledge across the whole team, which is necessary for scaling the team. With both deployment and monitoring—the two key developer experience pain points I described above—it's very inefficient to have a situation where only specific people know how to do things. By eliminating single points of failure and enabling any team member to deploy and monitor, we increase the rate at which our team can scale.

I'll mention a few of the things that we particularly value with our current performance monitoring solution:

- **Custom metrics:** We extensively use custom metrics to inform us if third-party services are experiencing issues. For example, we use third-party SaaS services to generate emails to our users and it's critical to know when there are problems.
- **Integrations:** PostgreSQL is our primary database, and thanks to Datadog's native integration we can directly collect a wealth of performance data and health metrics.
- **Distributed tracing:** We operate a microservices-based environment, which means it can be challenging to track requests end-to-end across the web of services. Also, we want to be able to determine what requests are taking too long and consuming excessive resources. All of this is easy to do with distributed tracing.
- **Infrastructure as Code (IaC) / Monitoring as Code:** We heavily use IaC to streamline our configuration and deployment process. We have extended this concept to monitoring: we are able to define our monitors, metrics, and dashboards with code using Terraform and deploy changes directly without having to touch the monitoring tool. Everything is versioned and deployed using pull requests.
- **Data pipelines:** We had a challenge on our data team too. Our data engineers also use Datadog to monitor the health of our Spark data pipelines running on Google Kubernetes Engine (GKE) and managed via Airflow.

Next steps

I'm proud that we solved our performance monitoring pain points and created a better developer experience over the last two years. I see a similar pattern playing out with our security tools and processes: a highly fragmented toolset which leads to inefficient processes and silos between teams. Datadog has introduced security monitoring, which is a logical extension of the platform. I'm excited to explore that in 2021. We've arrived at DevOps but I see the potential to arrive at DevSecOps this year.

Thanks to centralized and intuitive monitoring, our devs are now autonomous. They do not need to ask ops team members for help every time there's a problem. This has greatly improved our incident resolution times, and increased collaboration and productivity. Expertise about monitoring our environment is not restricted to one team or a small group of individuals, but is expanding rapidly across the team. This is very important for our business as we scale across Europe.