Monitoring

These series of pages will describe in detail on the current implementation of the monitoring stack as used by the Mendix platform and the HCL monitoring teams. It is set as a replacement for Dynatrace with the primary objective to reduce cost.

Background and design decisions

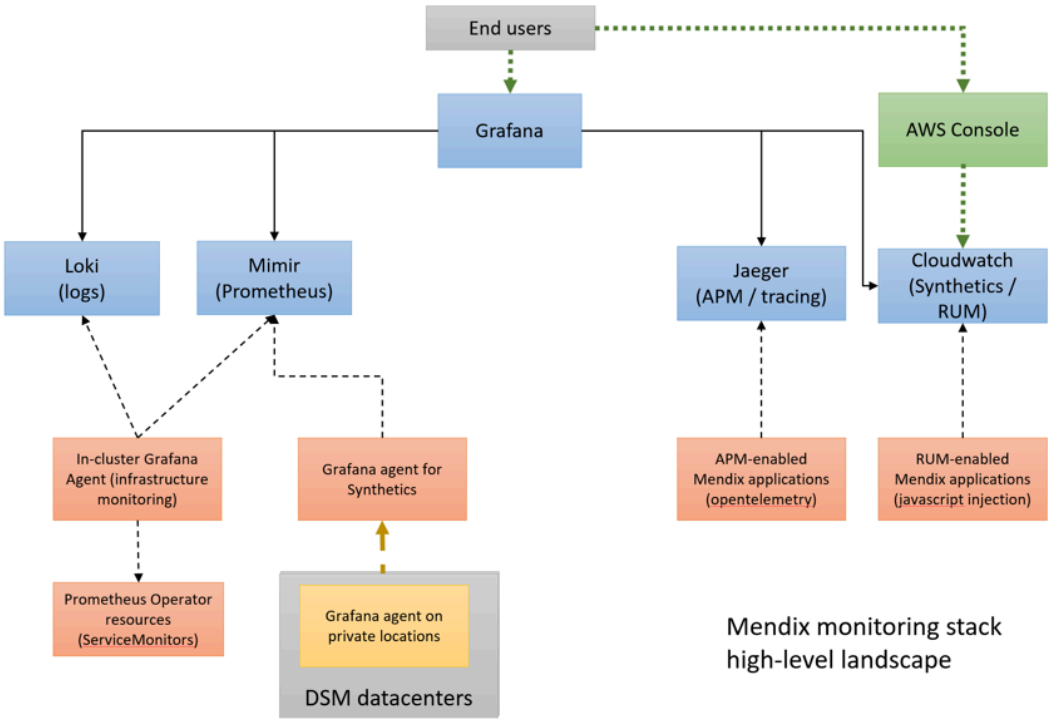
Somewhere around the late summer of 2023 there were conversations going about not extending the license of Dynatrace, which became stronger after the fall announcement of global cost reduction within DSM. Several POCs were done with both commercial and open-source tools to look for a potential replacement of Dynatrace, although there was a preference in the open-source world as that could be used to reduce costs of the monitoring footprint (manhour cost was considered out of scope). Eventually it was decided to divest Dynatrace with not a lot of time to move, so the project has been barebones and iterated over to add functionality. The nature of an open-source based system is that there are several components that integrate with each other one way or the other, which does not always provide all of the needed functionality. We also had to take the time constraint into account as we wanted to reduce the time that production was not going to be monitored as much as possible.

In the design decisions we balanced out what functionality is already by AWS and what are we going to run ourselves, keeping costs in mind. For example, AWS Managed Prometheus is expensive to run when comparing to running Mimir with S3 storage. Conversely, the Cloudwatch RUM is relatively cheap with $1 per 100.000 events, matches the use case needs of the teams and building it yourself would be a time consuming effort and not a lot of open source tools with similar functionality exists. Cloudwatch custom metrics are also relatively expensive, especially with a high number of metrics and a lot of Cloudwatch alarms executing queries. This was already a problem in the old deployment of the Mendix cluster.

Future iterations of the monitoring solution may see components changing, swapped or removed. The Elastic stack has been discussed on and off but, considering the time constraints at the time, we were further ahead with the Grafana stack and thus the decision was made to move forward with that. In the future, however, that may still change although that will not be in the short term.

High-level overview

We've settled on the following self-hosted components: Grafana, Mimir (metrics), Loki (logs), Jaeger (APM / tracing). The Grafana Agent is used extensively for various functionalities (it also runs more than one time with different configurations, most notably for HTTP based Synthetics). On the AWS side we're using AWS Synthetic for Browser based checks (also called Clickpath as per Dynatrace terminology), and Cloudwatch Real User Monitoring.



Mendix monitoring stack high-level landscape

As you can see, there are two interfaces the end-users can use; Grafana and the AWS console. This is the unfortunate side-effect of using CloudWatch for certain functionality, as the CloudWatch integration of Grafana only provides access to metrics and logs. RUM and Synthetic overviews must be viewed from there. You can also see that there are two types of Grafana Agents; one that is used as a daemonset that does the in-cluster monitoring and a separate one for the Grafana Agent based synthetic monitoring. Grafana Synthetic checks run from DSM's own datacenters as private locations, which in terms also run a Grafana Agent to execute the actual checks. For APM it was decided to use Jaeger because of the auto instrumentation of Java. We initially used the Grafana Agent Opentelemetry collector but that produced a new trace ID for every span, rendering it borderline useless. The Jaeger implementation does this correctly so we settled on this. For RUM, there is JavaScript injection happening on the nginx level of the Mendix application to inject the CloudWatch RUM JavaScript with the right parameters.

Loki and Mimir store all of their data on S3 and run entirely within the Mendix EKS cluster. Grafana has a separate RDS and also runs inside the Mendix EKS cluster. Jaeger runs inside of the EKS cluster and has an AWS OpenSearch Domain as a storage backend (this is one of the two supported by Jaeger, the other being Cassandra).

Repositories

The repositories referenced in the various pages are as follows:

- kubernetes-mendix-deployment - Repos (visualstudio.com)
- aws-synthetics - Repos (visualstudio.com)
- monitoring - Repos (visualstudio.com)

⚠ Please review the subpages of this page for additional information. All pages assume knowledge of the underlying tools. If you don't have that, review the further reading section of every page for documentation links.