Demonstration of an Observability Framework for Cloud Native Microservices

Nicolas Marie-Magdelaine[1,2], Toufik Ahmed[1], Gauthier Astruc-Amato[2]
[1)] Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400 Talence, France
[2)] Lectra SA, 23 Chemin de Marticot, 33610 Cestas, France

nicolas.marie-magdelaine@labri.fr, tad@labri.fr, g.astruc-amato@lectra.fr

***Abstract*— The advent of Cloud Native Microservices [1] has brought new challenges. Among them, the need to understand and monitor the complex and dynamic distributed systems that have become web applications. This demonstration present presents observability framework for microservices orchestration. The developed observability framework provides the means to understand the internal behavior of microservices at different layers, lifetime and abstraction levels. This demonstration utilizes a cloud-based infrastructure for observability framework and need an only a web-browser for a front-end.**

Keywords—observability, cloud computing, monitoring, cloud native applications, microservices.

I. INTRODUCTION

Many organisations and companies are progressively adopting cloud native applications along with associated microservices software design, containerization, orchestration and DevOps practices. One of the latest survey on the state of microservices adoption show the growing interest and adoption rate. [2]

This demonstration will introduce a practical implementation of our Observability concept in real cloud native orchestrated microservices environment. Our contribution in this demonstration rest in demonstrating the feasibility of an observability driven monitoring using a suit of customized tools. We will also demonstrate how to reproduce our results by showing how to implement a similar but simpler architecture based open-source and free software.

II. PROPOSED APPROACH

Observability is an emergent concept that extends the idea of monitoring. While monitoring mainly reside in connecting up/down checks to data extracted from servers and websites[3][4], Observability hopes to bring understanding to internals of applications and infrastructures. This need has become a necessity because of the increased complexity of microservices based applications. Those inner workings are understandable by correlating all the inputs and output retrieved by monitoring tools.

Traditionally, monitoring tools were just checking and alerting system and others functionalities were devoted to more complicated APMs application performance management systems. In modern cloud native application environment where resources are scaled on demand, microservices performances are now part of the application health.

Our observability framework can then be displayed in four different logical layers responsible respectively for:

- Collection and retrieval
- Raw data storage
- Processing and correlation
- Visualisation and alerting

Observability can be considered as a superset of monitoring by extending its concept and applying data analytics techniques on the monitoring data.

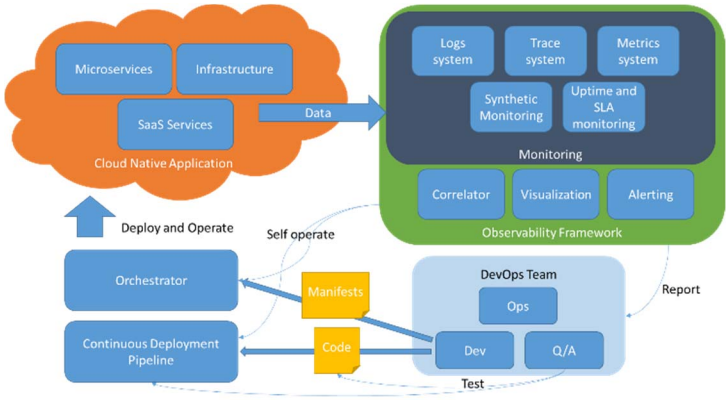

Figure 1: Elements and interaction in our Observability Framework for Cloud Native Microservices

Figure 1 describe the proposed architecture. In this architecture an observability layer is extracting data from the cloud native applications and use those data to both inform DevOps teams but also to operate on the general orchestrator which then can apply changes to the application. This control loop enabled by observability information is called “decisional observability”. Teams not only get better, clearer information about the state of their system, they can also automate their application to self-mitigate themselves for knows issues. This process is the base of Netflix’s principle chaos engineering [5].

978-3-903176-15-7 © 2019 IFIP

III. Demonstration Environment

We implemented an observability framework first in an industrial production environment within Lectra SA (EPA:LSS)[6], which is an industry-leading technology company specialized in software and soft fabric cutting equipment for fashion, technical textiles, furniture and automotive. Being an important player in the 4.0 industry, this production environment offers many SaaS cloud services. As a constraint, all the microservices created rely on the Microsoft Azure cloud infrastructure.

This demonstration is entirely based on our cloud infrastructures and tools built for Lectra's SaaS offer. However a list of alternative components will also be on display with free, open-source and on-premise software enabling those who wish to create a similar test-bed.

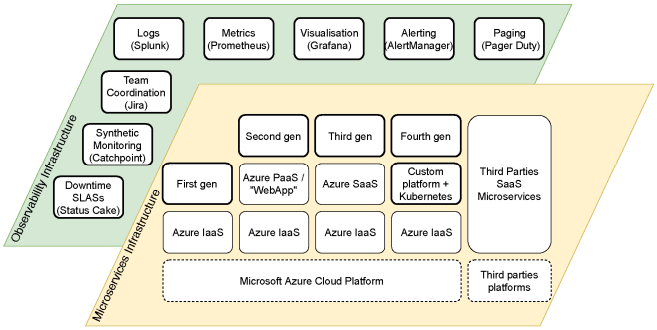

Figure 2: Our industrial heterogeneous cloud environment. Dotted elements are fully managed by providers, slim lines indicate partially configurable or observable elements, bold lines indicate full control over the elements.

Figure 2 shows both microservice infrastructure and observability framework. The observability framework provides a number of functionalities and build on highly customizable tools such as:

- Logs recovery, storage and management (Splunk[7])
- Metrics recovery, storage and management (Prometheus[9])
- Metrics visualization and correlation (Grafana[10])
- Alerting (AlertManager[11])
- Paging (Pager Duty[12])
- Team Coordination (Jira[13])
- Synthetic monitoring (Catchpoint[14], Status Cake[15])
- Service level agreement evaluation (Status Cake)

Our cloud native microservices based applications will be manipulated in order to demonstrate the features enabled by our observability framework.

A. *Custom health checks based on Observability data*

In cloud native environment, a microservices health cannot be deduced by checking basic values such as checking if a process is running. Many dependencies and connectivity factors must be considered to get insights a microservices health. In this demo, we will displayed how our observability framework can solve this issue.

B. *Observability based auto-scaling*

The dynamic nature of resources size and quantity in cloud native applications is highly dependent on multiple factors. Scaling needs may be extremely volatile with sporadic spikes that require very fast reaction timing in order to be transparent to final users. Observability enable us to detect those scaling spikes in a proactive manner. Indeed, scaling on the capacity needed instead of mitigating the saturation is a better design.

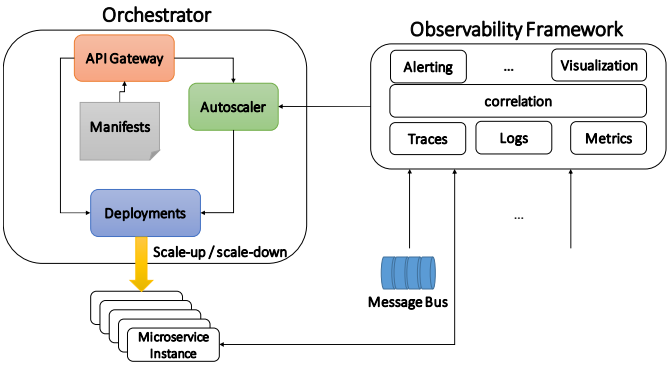

Figure 3: Elements of the observability driven auto-scaling orchestrator

Microservices are deployed by the orchestrator using a manifest file containing all the necessary parameters to deploy and run an instance. This manifest also specifies the autoscaler behavior and which data is used in the decision algorithm.

C. *Anomaly detection based alerting*

Operational engineers used to check for abnormal pattern by looking at dashboard. This type of monitoring was using human to perform pattern recognition. Nowadays statistical tools and even machine learning can be applied to the monitoring data centralized in our observability framework and enable Ops teams to relies on automated alerts provided by those systems.

D. *Metametric insights*

As all the collected data are stored in our observability framework it become possible to correlate them and process them using formulas to compose new metrics with new units that can describe better microservices' behaviors and states.

IV. Conclusion

This demonstration paper give some insights on the concepts, features and prototypes around observability and cloud native applications. Our production environment hopes to demonstrate the feasibility of those objectives and the benefits to any organization willing to follow the same path.

V. REFERENCE

[1] "FAQ - Cloud Native Computing Foundation." Cloud Native Computing Foundation, https://www.cncf.io/about/faq/.

[2] Ford, *The State of Microservices Maturity*. https://www.oreilly.com/programming/free/the-state-of-microservices-maturity.csp

[3] Aceto, Giuseppe, et al. "Cloud Monitoring: A Survey." Computer Networks, vol. 57, no. 9, June 2013, pp. 2093–115, doi:10.1016/j.comnet.2013.04.001.

[4] Gutierrez-Aguado, Juan, et al. "IaaSMon: Monitoring Architecture for Public Cloud Computing Data Centers." Journal of Grid Computing, vol. 14, no. 2, Springer Netherlands, June 2016, pp. 283–97, doi:10.1007/s10723-015-9357-4.

[5] « Principles of Chaos Engineering ». https://principlesofchaos.org/.

[6] "About Lectra." Lectra, 11 Dec. 2014, https://www.lectra.com/en/about-lectra.

[7] "SIEM, AIOps, Application Management, Log Management, Machine Learning, and Compliance | Splunk." Splunk, https://www.splunk.com/

[8] Google - Site Reliability Engineering. https://landing.google.com/sre/book/chapters/monitoring-distributed-systems.html.

[9] Prometheus - Monitoring System & Time Series Database. https://prometheus.io/.

[10] "Grafana - The Open Platform for Analytics and Monitoring." Grafana Labs, https://grafana.com/.

[11] Prometheus. Alertmanager | Prometheus. https://prometheus.io/docs/alerting/alertmanager/.

[12] "PagerDuty | Digital Operations Management Platform." PagerDuty, https://www.pagerduty.com/.

[13] Atlassian. "Jira | Logiciel de Suivi Des Tickets et Des Projets | Atlassian." Atlassian, https://fr.atlassian.com/software/jira. Accessed 8 Sept. 2018.

[14] "Leading Digital Experience Intelligence | Catchpoint." Catchpoint Systems, http://www.catchpoint.com/.

[15] Website Monitoring & Downtime Updates | StatusCake. https://www.statuscake.com/.