Conf42 Kube Native 2024 - Online

September 26 2024 - premiere 5PM GMT

Subscribe to watch

Watch this talk

Simplifying Multi-Cloud Observability

Video size:

Abstract

Managing Observability in multi-cloud is challenging. This talk emphasizes the need for standardized, vendor-agnostic approaches to ensure scalable and usable observability. Learn how standard schemas and open source tools can enhance visibility and resilience in multi-cloud deployments.

Summary

Transcript

This transcript was autogenerated. To make changes, submit a PR.

Hello, everyone. Welcome to my talk on simplifying multi cloud observability. In this presentation, we'll talk about multi cloud observability and why observability needs a different type of approach in the multi cloud world. My name is Sriverma Vegiraju, currently working as a software engineer at Microsoft. I'm a freelance contributor and a book reviewer as well. This is my LinkedIn and email. Please feel free to reach out to me if you have any suggestions or any questions regarding the talk or if you want to talk about software engineering in general. Let's get started with the agenda. First, we'll talk about what is multicloud and why is it gaining a lot of traction. Second, we'll look into what is observability. Third, we'll look at observability. Why is observability complex in the multi cloud architecture? Then we will look at how to simplify it and then we will conclude the talk. So what is multi cloud and why is it gaining a lot of traction? Initially, when cloud was just getting started, organization hosted their services on prem. Once cloud became mainstream, organization switched to using both On prem and cloud. This is what we call as hybrid cloud then companies Only started using cloud and if I as an organization use more than one Cloud provider to offer my services that is when i'm on a multi cloud strategy As part of a recent survey 98 percent of enterprise customers are either switching to or are already on a multi cloud strategy. And the main motivations, first, data sovereignty. If I have customers worldwide serving in different regions and different countries, I have to abide by the country's data privacy and data protection laws. For example, London has its own GDPR. European Union has its own GDPR. Similarly, United States, India, China, Japan, all of them have their own data privacy and protection laws. It's my responsibility to abide by these laws. And if I find that my cloud provider does not suit my needs, I might choose an additional cloud provider. This is one reason why I would switch to a take out strategy. Second, cloud vendor login concerns. Imagine I am using a particular service from a cloud provider, and somehow the provider decides to either increase the cost of the service, or change the service drastically, or deprecate the service. I want my business to be resilient to all these factors. This is the second reason why I would choose a multi cloud strategy. Third, cost optimizations. Every organization or every provider has their own strong offerings. Some offer free egress and ingress. Some offer cheap compute network or storage. If I am in a position to utilize all these strategies, I can reduce the cost of running my service on the cloud. So to summarize, data sovereignty, Cloud vendor lock ins and cost optimizations are the main reasons companies are switching towards multi cloud strategy So what is observability Put it simply it is the ability to measure the current state of my system Is my duration high? Is my availability low? Am I matching or am I? Reaching my SLAs, SLOs, SLIs Observability helps me answer all these questions. And the three main concepts in observability are metrics, logs, and traces. With metrics, it is a numerical measure of the current state of your system. For example, is my availability two nines, three nines, or four nines? What is the millisecond latency at which I'm serving my request? And what is the error rate? What is my CPU utilization? All these answers I can get from my tricks. Logs are nothing but your diagnostic information. For example, a request failed Then I have to look at the stack trace and additional metadata to debug the request This is what logs will give me. Traces on the other hand are becoming popular in the microservices world Let's say I have this request that has to pass through 10 different microservices And it fails in one of or one of the service You Traces will help me debug these kind of requests. Now, let's look at why observability needs a different approach in the multi cloud world and what are the changes that are mandating this mandating this. What you see here is a code snippet where I'm just using one cloud provider. What I have is a monitoring SDK that I'm using to emit metrics to a dashboard. Everything is simple because I'm just in a single cloud world. There is nothing complex here, right? Now, in, in this case here, I am using two or three different cloud providers. So I have three different SDKs and on the right I have two or three different kinds of dashboards Just looking at this we can understand the complexity that we are getting into. So let's look at that Because i'm using multiple clouds. I have multiple SDKs And because I have multiple SDKs I end up maintaining all of them. It is either versioning or deprecation or schematics I'll have to Keep tabs of all this information for the SDKs. Second, each cloud provider dashboard is different and the way they do aggregations is also different. Some sample at five minutes, some sample at one minute. Now, I have to document all this information so that my on calls understand how to debug a particular dashboard. Now, there's an interesting nuance there. The on calls In addition to be debugging the service outages are also debugging the dashboards now. By just hearing that we know things will get easily out of hand once there's a big fire. Last but not the least, the cost. Now we have to hire more experienced developers who don't come cheap and also train them, which is also not, which is also very expensive. So how do we get out of this problem? How do we make things complex for our developers is what we'll take a look at now. If I summarize the problem here, what I want to offer is one single pane of glass experience for my customers, for my developers. And if I have to offer one single pane of glass experience, I should also be cloud or provider agnostic. And this is where a lot of open source. Tooling has been helping us, and it is also helping us now. What we are going to take a look at is

OpenTelemetry, which is one such open source tool. It is under the Cloud Native Computing Foundation umbrella. And what it simply states is, it is a single open source standard to emit your observability data, your metrics, logs, and traces. Second, it is already supported by major cloud providers like AWS, GCP and Oracle already support this. And the more, and the best part, it is vendor agnostic. So let's look at how what are the critical components in this cloud native in OpenTelemetry. First, we have the specification. The specification comes into picture when you're implementing OpenTelemetry for a specific language. Because we don't do this daily, we will not get into details here. Next, we have semantics. Remember we discussed about one single pane of glass experience? For that, one important thing you need is a unified schema. With semantics, you can get that. Open telemetry SDK is what we will use to emit the metrics, logs, and traces. Before we were using a provider specific SDK, now we will use open telemetry SDK. By the way, it is available in many different languages, C sharp, Java, Go, Ruby, and so on. Fourth, what we have is the exporter. Exporter is responsible for translating the metrics that we Translating the observability data from OpenTelemetry SDK into a language that the provider understands. AWS has their own exporter. GCP, Microsoft, all of them have their own exporters. And what these exporters do is query SDK to get the observability data and emit to the backend, which is our last piece in the puzzle. The backend is nothing but the cloud provider themselves. AWS, Azure, GCP, or they can be Prometheus, Jaeger, also. To summarize, specification required mostly when you are implementing OpenTelemetry in a specific language. Semantics are the way you achieve one single schema, one single schema. OpenTelemetry SDK is what you use to emit the observability data. Exporter is what you use to used to translate the the OpenTelemetry observability data into vendor specific data. And then you have the packets. Let's look at architecture. So what on the left is on, what on the left here is before OpenTelemetry, we have our VM, our service, and the Azure Monitor SDK. Because we are tied to Azure. Azure backend, we are using their SDK to EMIT metrics, which we know is a problem on the right. What is the Open Telemetry one here. What ends up happening is instead of the Azure Monitor, SDK, use the open telemetry SDK to emit your observability data, and then you use the Azure telemetry exporter, which is readily available for us to consume. Just to clarify. The exporter is not a sidecar. It is just another package in C sharp or a dependency in Maven that you import into your service. And then the exporter does all the heavy lifting of translating the metrics or observability data into a way that the Azure backend can understand. And then send it to Azure. Now, what we have done is consolidated Now, what we have done is consolidated our code from multiple, using multiple different SDKs to one SDK to send the data. What you see here is a code snippet of how, when we are using the OTEL SDK. What we have here on the top is the namespace and then what, and then here we have the exporters and then on the last year. Is where we are sending, emitting the metrics. Now, if I want to add one more exporter I just add it to the set of extensions here and I don't change any of the names space the way I emit metrics. So the whole process has become simplified. This is an example of how we could, how how using open telemetry would look when in, when using multiple cloud providers. So here, none of the VM service or the SDK changes, they all remain the same. The only thing that changes is the exporter. Remember we also discussed single pane of class experience to see the dashboard. This is where we can use Grafana to export all the metrics data from different providers to the to Grafana so that we can offer developers once we can offer developers unified experience by while debugging through the dashboards. So this is how. We can simplify the whole observability stack for our service and make experience seamless for our developers As multi cloud architecture keeps growing it is very important that organizations make vendor agnostic strategy part of their architecture and part of their culture. This is how organizations can keep scaling without any bottlenecks and offer seamless experience to both developers and their clients. Apart from this, there are a ton of other open source tooling also available that have their own exporters and can help Query open telemetry sdk, for example with prometheus you are able to export Metrics to your own bucket With Loki, it is a log search experience that Grafana offers in addition to Metrix. Previously, they only had the Metrix, the dashboards only used to show Metrix, but now with Loki, you can also see logs. With Zepkin and Jaeger, you can export traces out of traces from the SDK to the Jaeger or Zepkin backends. With this, I conclude my talk. Thank you for taking the time to listen to the, to listen to this talk on simplifying multicloud observability and hope you enjoyed it.

Slides

Simplifying Mult... 1 / 19 76%

Download slides (PDF)

See all 32 talks at this event!

Siri Varma Vegiraju

Software Development Engineer @ Microsoft

Post

Share

Join the community!

Learn for free, join the best tech learning community for a price of a pumpkin latte.

Annual Monthly

NEWSLETTER

$0/mo

- ✓ Event notifications, weekly newsletter
- ✓ **Delayed access** to all content
- ✓ Immediate access to Keynotes & Panels

COMMUNITY

$8.34/mo

- ✓ Access to Circle community platform
- ✓ **Immediate access** to all content
- ✓ **Live events!**

- ✓ Regular office hours, Q&As, CV reviews
- ✓ Courses, quizes & certificates
- ✓ Community chats

Join the community (7 day free trial) →

Email address
First Name
Last Name
Company
Job Title
Phone Number
Country United States

☐ I consent to the following terms: Terms and Conditions & Code of Conduct

Subscribe to free newsletter →

Online tech events

EVENTS 2025

DevOps 2025

Python 2025

Chaos Engineering 2025

Cloud Native 2025

Large Language Models (LLMs) 2025

Golang 2025

Site Reliability Engineering (SRE) 2025

Machine Learning 2025

Observability 2025

Quantum Computing 2025

Rustlang 2025

Platform Engineering 2025

MLOps 2025

Incident Management 2025

Kube Native 2025

JavaScript 2025

Prompt Engineering 2025

Robotics 2025

DevSecOps 2025

Internet of Things (IoT) 2025

EVENTS 2024

DevOps 2024

Chaos Engineering 2024

Python 2024

Cloud Native 2024

Large Language Models (LLMs) 2024

Golang 2024

Site Reliability Engineering (SRE) 2024

Machine Learning 2024

Observability 2024

Quantum Computing 2024

Rustlang 2024

Platform Engineering
2024

Kube Native 2024

Incident Management
2024

JavaScript 2024

Prompt Engineering 2024

DevSecOps 2024

Internet of Things (IoT)
2024

EVENTS 2023

DevOps 2023

Chaos Engineering 2023

Python 2023

Cloud Native 2023

Golang 2023

Site Reliability Engineering 2023

Machine Learning 2023

Observability 2023

Quantum Computing 2023

Rustlang 2023

Platform Engineering 2023

Kube Native 2023

Incident Management 2023

JavaScript 2023

DevSecOps 2023

Internet of Things (IoT) 2023

EVENTS 2022

Python 2022

Mobile 2022

Chaos Engineering 2022

Golang 2022

Cloud Native 2022

Machine Learning 2022

Site Reliability Engineering 2022

Quantum Computing 2022

Rustlang 2022

Incident Management 2022

Kube Native 2022

JavaScript 2022

DevSecOps 2022

Web 3.0 2022

EVENTS 2021

Chaos Engineering 2021

Enterprise Software 2021

Cloud Native 2021

Python 2021

Golang 2021

Machine Learning 2021

Site Reliability Engineering 2021

JavaScript 2021

DevSecOps 2021

EVENTS 2020

Chaos Engineering 2020

Open Source Showcase 2020

Site Reliability Engineering 2020

JavaScript 2020

COMMUNITY

Support us

Speakers

Hall of fame

Discord

LEGAL

Code of Conduct

Terms and Conditions

Privacy policy

About the team

SPONSORS

Sponsorship

Request the Prospectus

Media kit