EPISODE 1088

[INTRODUCTION]

[00:00:00] JM: Grafana is an open source visualization and monitoring tool that is used for creating dashboards and charting time series data. Grafana is used by thousands of companies to monitor their infrastructure. It's a popular component in monitoring stacks and it's often used together with Prometheus, Elasticsearch, MySQL and other data sources.

The engineering complexities around building Grafana involve the large number of integrations, the highly configurable ReactJS frontend and the ability to query and display large datasets. Grafana must also be deployable to cloud and on-prem environments.

Torkel Ödegaard is a cofounder of Grafana Labs and he joins the show to talk about his work on the open source project and the company that he's building around it.

If you want to 30,000 unique engineers every day, consider sponsoring Software Engineering Daily. Whether you are hiring engineers or selling a product to engineers, Software Engineering Daily is a great place to reach talented engineers and you can send me an email, jeff@softwareengineeringdaily.com if you're curious about sponsoring the podcast or forward it to your marketing team. We are also looking for writers and a videographer. If you're interested in working with us, you can also send me an email, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[00:01:28] JM: This episode of Software Engineering Daily is brought to you by Datadog, a full stack monitoring platform that integrates with over 350 technologies like Gremlin, PagerDuty, AWS Lambda, Spinnaker and more. With the rich visualizations and algorithmic alerts, Datadog can help you monitor the effects of chaos experiments. It can also identify weaknesses and improve the reliability of your systems. Visit softwareengineeringdaily.com/datadog to start a free 14-day trial and receive one of Datadog's famously cozy T-shirts. That softwareengineeringdaily.com/datadog.

Thank you to Datadog for being a long-running sponsor of Software Engineering Daily.

[INTERVIEW]

[00:02:22] JM: Torkel, welcome to the show.

[00:02:23] TO: Thank you. Great to be here.

[00:02:25] JM: Yeah. You work on Grafana. Grafana is a visualization and monitoring tool. Describe the use case for Grafana.

[00:02:33] TO: Yeah. The use case varies greatly, because it's very open-ended into what the use is for. But mainly it's used for monitoring IT systems, so applications as services and infrastructure. But it is a very sort of agnostic tool in terms of what actual applications you put it to. Some use it for monitoring an industrial process through sensors that sort of transmit networks, or you use it to monitor your home kind of energy usage or if you have solar panels on your roof. You might use Grafana to monitor your energy usage. It's very much everything from sort of hobby usage, to big power plants that might use Grafana.

But the main, the big use case that we focus more on maybe than the others is the DevOps kind of IT operations, monitoring, and that's how I got started in being kind of an architect and a developer. I wanted to know what was going in production with these applications that I was writing and the metric time series databases and the tools to actually build dashboards to show what the applications and services and users were doing in real-time. That's kind of how I got into this space [inaudible 00:03:47] with metrics and instrumentation.

[00:03:51] JM: When you first created Grafana, what was the state of the art for monitoring and visualization products?

[00:03:59] TO: Yeah, it was a very different space back then. The biggest product or open source project at the time was called Graphite, and this is how I got into this space. It was actually a quite old open source project. I think it was created in 2011 and it was sort of like maybe some of you that are listening are familiar with RRDtool I suppose. Mainly the monitoring

tools back then started kind of monitoring infrastructure, monitoring networking. Having PNG graphs on a very basic dashboarding tools to monitor infrastructure metrics, APU networking.

The thing that kind of changed around 2011, 2012 was Graphite and Statsd. They kind of made metrics more accessible and easy to use. They made most issues to instrument replication. You send application metrics, not just kind of CPU metrics or network metrics. That made it possible to sort of visualize what the application was doing, the performance and behavior and business metric, kind of adding user behavior metrics.

I think a key turning point for me at least was the introduction of the open source project, Statsd, which is now more like a protocol, a bunch of different implementations. This is what you write in your application to send metrics to a server. That made it super easy to – Or sending application. Graphite was one of the first popular database system for them. But the actual experience of building dashboards of showing these metrics was primitive and hard to use, and that's kind of where I struggled, kind of getting my team to start using this tool was kind of – They struggled building the dashboards and creating these graphs. That's why I started looking at creating Grafana, is to help make that process easier and more kind of visually appealing to really make something that you want to put on a TV on the wall.

[00:05:51] JM: Graphite is the database for storing the metrics?

[00:05:56] TO: Graphite at the time was both. It was a database. It was a graph-rendering. Is a software that rendered PNGs and they had a UI to build dashboards as well. At the time, the client-side rendering wasn't that kind of great as well. Graphite was kind of a complete package. But many struggled with its UI, its dashboard tool. So that's kind of where Grafana replaced the full dashboard for Graphite.

Later on, this kind of space of time series database became hugely popular and more open source project try to build an improve on what Grahpite was doing. Databases like InfluxDB, Prometheus and others came on to sort of also do time series databases, but it's slightly different focused. Grafana kind of took all those communities and said, "Okay. Well, Grafana can be the UI for all of these." Grafana has a very open architecture for where the data is coming from. You can visualize data from many different types of databases.

[00:07:01] JM: At this point, we can start to get into the engineering of a monitoring and visualization product. I think about needing to handle inputs from a variety of different sources. There's really fast, high-volume streaming data that you could need handle and there might be buffering issues involved with that. Maybe you have really large batch imports that you have to handle as well. Tell me about the handling of large inputs of various speeds.

[00:07:36] TO: Grafana do some streaming, but mainly it's about the query processing and sort of dispatching queries to different data sources. A single dashboard could issue queries against multiple different data sources. Most of the processing is done by each database layer. Grafana doesn't actually do that much processing itself. It issues queries and expects kind of each database backend to do most of the heavy lifting on kind of query processing.

There are some processing and translation that Grafana does in terms of kind of translating the response to a unified format that Grafana understands. Yeah, there is not that much buffering going on. Grafana always kind of fetches the whole time range and visualize that. We are investing more in streaming and we have some streaming use cases, but most of the scenarios and databases we work with now is kind of – Grafana sends a query. All the processing is done by the actual database. There's not a lot of offering problem. There're caching problems and issues, but most of the heavy lifting is not done by Grafana, those problems.

[00:08:51] JM: How much data do you cache on the Grafana side?

[00:08:55] TO: It depends. It's not something we do right now. We cache depends on which data source you're using, but we don't really cache at all mostly because when we're looking at these time series databases, they have their own caching built-in in the backend and Grafana also is usually [inaudible 00:09:16] ask queries and queries and low time ranges. So you're looking at maybe last hour, last day, and the caching is mostly done by the database system.

[00:09:28] JM: The querying process, fetching queries from various data sources, could you take me through the lifecycle of a query to a typical data source from Grafana?

[00:09:40] TO: Yeah, sure. This is actually where I spend a lot of time initially, is if you start with the lifecycle of actually writing the query, because that's where a lot of the dashboarding tools at the time did not do any sort of – Provided any help to the user. A big thing that I focused early on and continued to do is make it easier for users to actually write and understand the queries, and that was something that I wanted to place a heavy focus on. To make it easier for everyone on the team to actually start using this tool and not have to learn some new syntax or query language. To be able to sort of get up and running by themselves.

The journey for a query starts really in the query builder and it's like a UI query builder that kind of helps the user write a query, and maybe this is different for every data source. It's kind of tailored-made UI to build a query. Then it gets then to – When Grafana actually issues a query, it gets sent through Grafana backend that proxies the query, and this is usually to work around course issues and security issues, because of the authentication to the actual database happens in them. The lifecycle there is – Yes, a query goes through some translation layer and the result gets transformed to a format that Grafana can process. Because the biggest [inaudible 00:11:02] it's multi-data source or I think it's up to 50 different data sources now or more. All the data sources kind of have to translate into the standard format.

[00:11:13] JM: What is the standard format for Grafana they all have to translate to?

[00:11:17] TO: It's based on Arrow, Apache Arrow. I'm not sure if you're –

[00:11:20] JM: Oh, yeah. Sure. Sure. Sure.

[00:11:21] TO: Touch on that. This is a recent kind of investment we have done to use Apache Arrow mostly as the wire protocol, but also the structure we work with in-memory is also very sort of similar and can be serialized to an Apache Arrow columnar data format.

[00:11:39] JM: Right. When I first did a show about Apache Arrrow, it was a data interchange format for Python and Java. You have these people who are writing Spark queries and wanted to have the Spark data, those loaded in the JVM to easily be portable to some shared Python process. But as I understand, the interchange format has really evolved to become more flexible

and used in a wider variety of use cases. Can you tell me more about how Apache Arrow is used?

[00:12:15] TO: Yeah. I mean, I'm actually not an expert on Apache Arrow, but the main kind of innovations in Apache Arrow is efficient in-memory structures for columnar data to be able to do really sort of fast processing on really large data structure and also be able to have efficient wire protocols for the same kind of binary data structures. Apache Arrow is a binary data structure to really able to do in-processing without having serialization and also be able to do wire transfers without serialization costs.

I mean, the interesting things with Apache Arrow, it's kind of a standardized protocol. You can have a way to interrupt with other sources as well. The interesting thing with Grafana, I think, with our backend plugin model is that it's based on GRPC as well as the wire format for the data is Apache Arrow, but the plugin protocol is GRPC. You can actually write data source plugins in Grafana in almost any language that supports GRPC. We have some plugins – Most plugins are written in Go, because our backend plugin is Go or written in Go. But we have some plugins in Node and we're looking through expanding that to Python and Java as well.

[00:13:35] JM: To better understand the use case of Grafana and the execution of somebody building dashboards in Grafana. I set it up and I've got my different pieces of my infrastructure that I'm monitoring. Let's say I'm a ridesharing company. I've got all kinds of things I want to monitor throughout the infrastructure. Do I set up these queries and then the queries just stand there and then trigger every epoch and retrieve the data? What's the workflow for the continuous querying?

[00:14:11] TO: Yeah. Let's take one of the biggest kind of banded use cases right now. You might have a Docker – You might be running some application in Docker and you want to have some Docker stats, say. One of the most popular databases right now is Prometheus. Prometheus has some ready integrations for monitoring and storing data from Docker and it also has some excellent application libraries as well for writing custom metrics that get stored in Prometheus.

What you then do is you can either then set up a Prometheus data source in Grafana. You just point out, “Here is my Prometheus database server,” and then building dashboards yourself, or you can find our already built dashboard by the community. There are tons of kind of Docker-based monitoring dashboards that you can find on grafana.com or on GitHub. People share these kind of prebuilt dashboards. These dashboards are specific then to that combination of kind of, “Okay. You have a Prometheus database and you want to monitor, visualize Docker metric.”

In those cases, you don’t actually have to build the dashboard yourself. You can find really awesome, already pre-built dashboard. If you have custom application metrics and sort of you have actually written code to increment. So maybe you want to have metrics on when people log in or you want to measure the performance of certain API calls, then you have to have sort of write code to instrument your application. Then you have to actually build your manual dashboard, build a new dashboard, add a panel and write a query in the Prometheus query language, and this Prometheus query language is very domain-focused. It’s not like any other query language or it’s not very similar to SQL, say. It’s very specific to this is the metric and these are the key values of kind of what I want to query.

You have to learn a new language. Still something that is even though you have a query UI or a builder that helps you a bit, you’ll have to learn some new semantics of writing a metric query. Those are the most common use case, is either monitoring an application like MySQL or some kind of prebuilt where the metrics are well-defined and you can find an already built dashboard for it or you’re building custom dashboard or your own application.

[00:16:32] JM: You mentioned Prometheus. That’s a metric server for – Well, it’s a distributed metrics server, and I’ve done a number of shows on Prometheus scalability, because often times you’re wanting to store these metrics from Prometheus and you’re getting lots and lots of data and you need some scalable way of storing and querying that data. What’s the state of the art for a scalable Prometheus and what have the people that are using Grafana tended to architect around Prometheus infrastructure?

[00:17:10] TO: Yeah. There are really three different solutions to scaling Prometheus. The first kind of out of the box experience of scaling Prometheus is sort of having many Prometheus, and

this is pretty common in Grafana, that you just have a lot of Prometheus servers that monitor different parts of your kind of infrastructure or applications. The problem then is that you don't have a central place to query all of your applications or all your infrastructure.

If you want to have a truly centralized query API that can query across many, many Prometheus servers, there are two big contenders. There's Cortex Project, which is a horizontal scalable Prometheus implementation with truly clustered data store, and this is what we have invested heavily at Grafana Labs and what we run our Prometheus, kind of Grafana Cloud Prometheus service on using the Cortex open source project.

There's also another project called Thanos that tries to do something very similar, providing a horizontally scalable distributed Prometheus solution. They both share some similar characteristics, sort of similar traits, but they're also very different in how you operate them.

[00:18:28] JM: You have a hosted Cortex product?

[00:18:30] TO: Yes. Our Grafana Cloud sort of Prometheus monitoring solution or data store is run using Cortex.

[00:18:39] JM: The use case there is somebody has distributed applications that they want to monitor and they want to create dashboards for those. They set up a Cortex instance on your cloud and then that makes it easy to also use your Grafana instances.

[00:18:57] TO: Exactly. IN those instances, it's usually when you are reaching a point of sort of scaling Prometheus. You have a lot of Prometheus servers and you want a single source of truth across your whole infrastructure and across your whole kind of application services. That's kind of where Grafana Cloud Prometheus really shines, is when you want to have a centralized place to query and to visualize all that data from all your Prometheus servers. Yeah, that's the main use case when you really at a large scale as well.

We do have sort of small tiers as well when you just don't want to run Prometheus at all. We have a small agent, a Grafana Cloud agent that is only like data collection side of the Prometheus server. There are use cases as well for a smaller user.

[SPONSOR MESSAGE]

[00:19:52] JM: When the New Yorker magazine asked Mark Zuckerberg how he gets his news. He said the one news source he definitively follows is Techmeme. For more than two years and nearly 700 episodes, the Techmeme Ride Home Podcast has been one of Silicon Valley's favorites. The Techmeme Ride Home Podcast is daily. It's only 15 to 20 minutes long and it's every day. By 5 PM Eastern, it has all the latest tech news, but it's more than just headlines. You could get a robot to read you the headlines.

The Techmeme Ride Home Podcast is all about the context around the latest news of the day. It's top stories, the top posts and tweets and conversations about those stories as well as behind-the-scenes analysis. Techmeme Ride Home is like TL DR as a service. The folks at Techmeme are online all day reading everything so they can catch you up. Search your podcast player today for Ride Home and subscribe to the Techmeme Ride Home Podcast.

[INTERVIEW CONTINUED]

[00:20:54] JM: If we come back to the core conversation around monitoring. If you're talking about monitoring, latency can be really important. If you have a dashboard, you want that dashboard to be up to date. Are there any issues with dealing with the latency in querying?

[00:21:13] TO: There's usually not a problem of latency. There can be a problem of sort of ingestion time maybe, sort of the problem of – In Prometheus, for example, you set up a scrape interval, a period of sort of – When Prometheus will ask all the applications for their metrics and for all the servers for their metrics. There could be a lag or when new data arrives and when it's available for being queries as well. There's also of course sort of latency in query execution. If you have a very expensive query, it can take a long time. Depending on what type of queries you do, that's rarely a huge problem with time series databases, because you usually have a specific query involved and they're usually pretty fast. I mean, it's not like you're connect querying and going for a coffee. Queries for time series databases are usually sort of fast, because you want to be able to interact with the data. You want to filter, zoom in. The latency there is pretty good.

The thing that is interesting and tricky is – I'm not sure if this was what you're getting at as well, but it is a different problem to measure latency, you want to visualize latency. That's a different problem that Prometheus also address, and this is where you kind of get into sort of technical terms, like percentiles and kind of distribution or measurements as well and it's incredibly important topic if you're interested in performance and actual – What your users are experiencing. If you're looking at, say, an API latency that you're measuring. Here, there're really a lot of good tools out there in these instrument libraries that can measure your performance that you add to your code. They can record percentiles or they basically record a distribution. You can say with pretty good certainty that's 90% of users get a response under a second. But 10%, they do have a latency above a second or it can get 99% of all users have a certain experience. That's the real power of looking at latency data in a distribution instead of just looking at an average or min or max. That can also be very misleading. That's super common use case for Grafana, looking at latency.

[00:23:29] JM: Grafana could be used to inspect metrics and logs. Can you tell me about the difference in what kinds of infrastructure you have to build around log querying versus metrics querying.

[00:23:45] TO: Good question. It's something that at least when I got started into this, the concept of metrics and time series databases was really quite novel and I had to explain to people what is a time series database, and people were familiar with logging, because log files has existed for quite a number of years. But metrics is quite still maybe not super familiar to everyone.

The big difference is volume, basically. You can send a huge amount. You can record a huge amount of metrics because the format is so specialized and the databases are so specialized just recording measurements of a time. You don't have to have at all the same kind of level scrutiny of what you're measuring as you have with logs. Logs tend to be piled up and be quite expensive and the data volume becomes so large.

Grafana started, yes, with a focus on metrics. But we have last few years or so been focusing more and more on logs and really kind of tying them together. You can sort of explore and

search for logs and also kind of jump from metrics to logs. That's one of the reasons we launched logs dedicated project, a database I recall, Loki, which where we tried to do something. The tagline for Loki is like Prometheus, but for logs. It has a very similar kind of tagging, labeling system. The logs have a similar naming scheme to your metrics, and this allows you to actually then go from a metrics query that looks at, say, if you're running Kubernetes, you can look at a metrics from a specific pod in Kubernetes, say, and then jump directly to the logs for that pod. Navigations like that are really something that we focused heavily on and quite a unique within Grafana.

[00:25:35] JM: That use case of being able to label logs, the Prometheus-like experience of a logging system, why is that useful?

[00:25:47] TO: It's useful because it's important when you want to sort of filter your logs or start your logs. You want to be able to define the same kind of filters as for your metrics. When you search your logs or filter down your search query, you can use the same key names, label names and label keys as you do when you write your metrics queries, and this enables kind of as consistent navigation and filtering as well as building dashboards as well that have – Where you can use the same filter keys in both your metrics and your logs query.

This is something that also kind of magic with Loki in particular, is that some of these label names that kind of gets be part of the log screen is defined automatically for using Kubernetes and has this automatic way of kind of determining which pod and cluster is running in.

[00:26:43] JM: If we talk about the actual design of the dashboarding layer, it's built in React, and have you had much involvement in the frontend development?

[00:26:53] TO: Yeah, that's pretty much what I'm mostly involve din, pretty much involved in the frontend and UX design as well. One of my passions are is UX experience of using Grafana. We're doing a very tricky migration as well from AngularJS. Grafana started over 6 years ago now, and at the time, React didn't exist and the most popular frontend framework six years ago was AngularJS, the first one, the first version that Google launched. Horrible library, but it was popular at the time.

Google deprecated it and created a new framework called Angular 2, which is a completely different frontend library, or at least technically it's very different the migration. Almost like you have to rewrite your application. I wasn't that impressed with Angular 2. I think they made a lot of the same mistake as Angular 1. I was much more impressed with React, especially kind of for defining a complex markup. I think React is an amazing library and gives us powerful tools to mix code and markup. I really like React, at least before they created – Added hooks.

[00:28:06] JM: You're not a fan of hooks.

[00:28:08] TO: Both. I mean, I think large functional components with a bunch of use effect and use call back, they become quite messy in my view and harder to reason about, unless you kind of move all the state logic out. Yeah, there are pros and cons to hooks, I think. But anyway, I'm a huge fan of React, and we started a process of migrating AngularJS, and this is something we've been working on for now more than two years, which has been interesting, because we have – One of the big brains of Grafana is a plug-in system, where you can write – Plug in data sources, plug in panels, and it's been a challenge trying to move this plug-in platform and dashboarding platform to React while being somewhat backward compatible with all the existing plugins out there. I've likened it to replacing kind of all the pieces of a car while it's moving. It's gone surprisingly well. There are lots of good ways we have been able to sort of render Angular within React and render React within Angular. Thanks to those approaches, we've been able to sort of migrate piece by piece.

The other things that have been fun to do on the frontend side is that we're really investing in Grafana as a platform. We have NPM packages for all our UI components at Grafana/ui, and we have tutorials and docs now for something like a CLI tool that helps you create plugins as well so you can get through a template going. We are almost like sort of think of bootstrap, but for Grafana plugins. Something we're investing heavily in. Plugin platform, component libraries and docs. We really make – Creating a new panel, a new visualization or a new data source [inaudible 00:29:50].

[00:29:52] JM: Can you say a little bit more about the frustrations with hooks? I mean, I talked to a lot of people who are fans of hooks and say it improves a lot of the React concurrency issues.

[00:30:03] TO: It's mainly I think a mental problem I have. I see a render function, like functional component as it's a render function. It produces markup. Seeing all that, if you then mix up a bunch of [inaudible 00:30:15] and use callbacks in that render function, it just kind of messes with my mind in terms of what's just in the render path and what's actually not in the render path? That makes it at least for me harder to reason about the code, because it mixes up all these callbacks and state logic within the render function.

Unlike the small, the hooks, when the component is really small. If you're talking about a functional component with hundreds lines of code and a bunch of [inaudible 00:30:51] and use callbacks, and the use callbacks is also – I guess, the fact that you have to – If you want to memoise the callback, you have to pass in a parameter list of kind reuse this function unless these property is changed. Just add so much complexity, in my view, to have all these memoisation and things you have to think about. Okay, cache this function unless this property is changed. To me, it adds a lot of things that could go wrong and complexities as well in terms of kind of caching functions and generating functions and dependency lists. But those are the things that I find problematic with Hooks.

[00:31:32] JM: How big is the frontend team?

[00:31:34] TO: Let's see. The front team I would say is – We're split. We have two mainly frontend teams and one enterprise team I have a mix. Maybe 15, 16 people that are mainly frontend engineers, plus [inaudible 00:31:49] two, three. We're growing a lot. My numbers might be off, but it's around there. We have one platform team that is mainly as focused on these components and the dashboarding experience. We have another team that is mainly focused on a use case, on the SRE, DevOps troubleshooting use case well. We have sort of both focused teams that are focusing on a specific use case and then we have sort of platforms teams that have broader focus.

[00:32:21] JM: Can you help me understand the architecture of the overall organization? What are the different teams and what is the interaction patterns around sharing information between those different teams and organizing plans and missions?

[00:32:36] TO: It's a bit different, but mainly we try as much as possible kind of have a shared platform and components and all the data source abstractions and query editors are the same. Heavy focused on a platform and sharing as much as possible. Grafana's frontend is all Typescript, and a huge shout out to Typescript. If you're building a large frontend application with lots of kind of shared components and shared types and shared interfaces, use typing, use Typescript, use Flow or whatever. I could not foresee building Grafana with sort of plane un-typed JavaScript. It's been a huge advantage to us as invested more and more into batch script over the last 2 and a half years.

[00:33:25] JM: How do you feel Grafana fits into the market? You've got a wide variety of monitoring platforms, Datadog for example. It seems like there's new monitoring and logging companies that come out every day. What's Grafana's place in the market?

[00:33:40] TO: Grafana's place in the market is quite unique still, and it's been I think very unique for six years now, and that is that we allow you to visualize data from different types of data sources in a single dashboard. You can even mix different data sources within a single graph. The main difference compared to some of the big SaaS vendors is that they require their data you've visualized in Datadog, for example, must live inside Datadog. You have to use some of their agents and some of the ingestion APIs if you want to visualize data in Datadog's UI. You have to send the data to them.

The biggest kind of difference with Grafana has been that its data source is very live. Grafana itself doesn't store any data. It stores some events. You can comment on Graphs. You can have what we call annotations in Grafana [inaudible 00:34:31], or for events, like deployments or you just want to comment on a graph. Those are the only things basically we store in Grafana besides dashboard.

Data itself, the metrics or the logs, Grafana doesn't store them. It just fetches them live directly from the different data sources you can hook up to Grafana. I think that dashboarding solution with lots of support for different open source and commercial kind of services and databases is one of the unique things about it, as well as having alerting on all those data sources. That's also a key part of Grafana, is that you can also define alert rules. Not only visualize the data.

[00:35:12] JM: What are the engineering problems that you've had as the company has scaled that you didn't have in the beginning?

[00:35:18] TO: One of the biggest problems that we've had is kind of dealing with the open source and the adaption rate. Grafana has become one of the biggest open source projects, and seeing that adaption and seeing all the data sources also adds a ton of support [inaudible 00:35:36]. We get a ton of sort of issues and feature request and plugs reports and problems that – Dealing with the scale has been kind of a problem that continues to challenge us in terms of finding better solutions for issue triage and for better ways to solve community support and help users be effective. Those are I think the problems that I didn't have in the beginning. It was just Graphite and InfluxDB in the beginning and it was quite easy to sort of help users with those two data sources. Now, we support so many different data sources and so many more use cases. That's a challenge in terms of being able to really provide solid sort of help to users that use data source that you don't have access to or some data source plugin that you don't really know how it works. That's one challenge I would say.

The other one would be onboarding your developers and get community involved, because it's such a complex domain. That's another challenge, is getting people to be familiar with developing Grafana and engaging, being kind of a frequent contributor is something that we struggled with a big and try to improve kind of developer guides in the last year or so. Been focusing more on kind of develop advocacy and getting more people involved.

[00:36:56] JM: As you build out like connections to more and more data sources, is the management of all of the different integrations, is that a difficult maintenance problem?

[00:37:08] TO: It's also something that we are kind of investing heavily in and sort of helping us. Maybe we're doing more in terms of integration testing and having more ways to automate. It is the cost of all these kind of interoperability and being a platform, is that the feature matrix is quite big in terms if you want to test with all different setups. We're still trying to sort of come up with better solutions there and get maybe more community involved and helping support different community data sources.

[SPONSOR MESSAGE]

[00:37:46] JM: Today's show is sponsored by StrongDM. Managing your remote team as they work from home can be difficult. You might be managing a gazillion SSH keys and database passwords and Kubernetes certs. So meet StrongDM. Manage and audit access to servers, databases and Kubernetes clusters no matter where your employees are. With StrongDM, you can easily extend your identity provider to manage infrastructure access. Automate onboarding, off-boarding and moving people within roles. These are annoying problems. You can grant temporary access that automatically expires to your on-call teams. Admins get full auditability into anything anyone does. When they connect, what queries they run, what commands are typed? It's full visibility into everything. For SSH and RDP and Kubernetes, that means video replays. For databases, it's a single unified query log across all database management systems. StrongDM is used by companies like Hurst, Peloton, Betterment, Greenhouse and SoFi to manage access. It's more control and less hassle. StrongDM allows you to manage and audit remote access to infrastructure. Start your free 14-day trial today at strongdm.com/sedaily. That's strongdm.com/sedaily to start your free 14-day trial.

[INTERVIEW CONTINUED]

[00:39:15] JM: When you talk to SRE teams or DevOps teams, do you get any sense of ways that DevOps workflows are changing? Any new insights about how these teams are organized and the problems that they're dealing with?

[00:39:32] TO: I think there's more to explore I think around kind of prebuilt dashboards, best of practice kind of alert rules and more ways we can, as a community, kind of define processes so we don't have to sort of reinvent the wheel in every kind of company for different types of alerting and monitoring scenarios. I think there are more things we can do there. One step in that direction has been what we call Prometheus mix-ins or Grafana mix-ins. I can't remember the name now, but that kind of defined dashboards and Prometheus alert rules. Everything kind of in one package, I think there's more along that path that we can explore.

The other thing that we see as well is kind of as a challenge users adapting Grafana is kind of – Because of the dashboard sprawl. Everyone is creating copies of dashboards and figuring out sort of what dashboard is authoritative or which dashboard is not being used. We're looking at

ways to sort of help those things as well. I think the thing that we want to invest more in is kind of dashboards as code and find better ways to kind of define your infrastructure and application set up more as GitOps and through code.

People use [inaudible 00:40:40] Grafana as well. There's a lot of popular tool as to provision Grafana with dashboards that you're defining through files and checked-in through GitHub. But there's more – That's kind of one of the advanced use cases of Grafana as kind of dashboards as code and where you fully provision Grafana from a git repo, say. But there are more things we can do there to help improve that workflow so you can actually edit and improve the dashboard and the UI, but then kind of have that be maybe open a pull request to the GitHub repo or somehow make that flow go back. Not just have your dashboard and monitoring experience defined from the code, but also defined in the UI. That's something that we want to invest more in. Have a two-way story there. Still, provide GitOps experience, but don't lose the UI. Still being able to have that be user-friendly and be powered by the UI.

[00:41:36] JM: What kinds of management issues have you had to get better at in your journey towards becoming a company leader?

[00:41:46] TO: Oh! Transition into sort of a management position at Grafana labs as we have grown, and I'm a manager of managers, because we're now so many people that that has been required and just recently been – Before, I tried to delegate all of the management and people side of things to other people so I can stay involved in the technical side of things and staying involved in the product management.

I haven't sort of delved super deep into the people management and the management side of things. But tricky thing has been kind of scaling the product management side of things as well, and that's something we're exploring more now. On the people side, I think been a fun journey kind of exploring how the one-on-ones and kind of the process around OKRs and setting goals. I'm still evolving as a manager. I'm still kind of a reluctant manager and an engineer at heart.

[00:42:37] JM: What's the interaction with the open source community? How does the open source community feed into the roadmap at Grafana?

[00:42:45] TO: I think we definitely checked the GitHub issues and popularity there, and that controls [inaudible 00:42:52] into the roadmap and priorities. The biggest that have impacted their priorities right now, the path from work that I've been involved in is sort of the React migration and getting platform that can last for a long time. Getting a really modern data visualization architecture and plug-in architecture that can stand the test of time. That is something that's taken a huge amount of time and huge amount of people. I'm looking forward to sort of have that sort of – We're super close to now being kind of somewhat done with this new panel or this new React-based architecture and new data visualization architecture.

Then we can actually start addressing more of the popular community requests, because they have been mostly on second place in priorities because we have been dealing with this complex migration and trying to get a new platform in place. But yeah, I think it's super important to look at the open source feedback. We do tons of user interviews where we ask the source users and customers through Twitter and other ways on our public Slack as well. Give us feedback on what we should do and give us feedback on early mock-ups. We try to engage a lot mainly through GitHub and our public Slack.

But on that point, we actually recently introduced a governance model and a public mailing list as well. Kind of how important decisions are made on a much more public. So we have sort of our traditional open source governance model for the whole project.

[00:44:23] JM: We should have talked earlier about this, but tell me about the deployment model for Grafana. Obviously, it's going to vary if somebody is self-hosting versus hosting on your infrastructure, but I just love to know how those deployments typically are architected.

[00:44:38] TO: Grafana itself, because Grafana doesn't do that much data processing, it just stores dashboards and users and proxies queries to other databases. Grafana itself can be run very sort of slim. It requires not a lot of resources itself. It can run it just on 100 or 200 megs of RAM and it runs by default with an embedded SQLite database. There're kind of zero dependencies. If you just want to get started with Grafana, you can just download it, run it on multiple packages for different OSS and distributions as well as Docker, of course. It's super easy to get started with.

Then, of course, if you have a lot of users, the most common kind of thing to do if you kind of cross maybe 100 users or 50 users. I don't know exactly what the threshold is. You can upgrade to MySQL or Postgres database. That's just for kind of dealing log-in sessions, dashboards, users, application tokens and other things. That's what Grafana uses its database for as well as a few other things, like annotation events. That's the most common set of hosting Grafana and then, of course, it varies based on what data source.

[00:45:53] JM: People these days, they're typically using – What? InfluxDB or, like you said, Cortex, or Prometheus data. What are the other data sources that people are using?

[00:46:04] TO: The most popular one by far is Prometheus and InfluxDB is also very popular, and Elasticsearch is also very popular. Then we have kind of the non-core use cases in terms of SQL databases. We do support Postgres out of the box and MySQL, and both of those are also really popular. Then we see – The other really popular one are Cloud Watch and we support out of the box all the three major cloud vendors. If you have running services on and use Cloud Watch, you can use Grafana. And if you use Google Stack Driver to monitor your infrastructure applications there, you can use that, visualize that also in Grafana, and same for Azure. Then we have support for Datadog and New Relic and all these other kind of SaaS vendors as well through plugins.

[00:46:57] JM: Cool. Well, Torkel, thanks for coming on the show. It's been really great talking to you. Do you have any closing thoughts on where Grafana is going and the future of the company?

[00:47:06] TO: Yeah, sure. I mean, we just launched Grafana 7.0 just a week ago, which is a huge landmark for this kind of new visualization platform that we've been building. I'm really excited about seeing kind of what the community will build with this new platform. It's just a whole new level of kind of ease of use to build a new visualization within Grafana and you get so much more for free now when you build a visualization with this new platform. You will get access to Grafana thresholds, override rules and all kind of things that are provided by the platform now. We have some cool tutorials for how to do this. I'm really excited about seeing kind of that just in the next couple of months and years on what the community will do and as well as what we, Grafana Labs, the company, will build using this platform. Because the thing

that excites me is kind of Grafana itself is very open-ended tool. It's like a toolbox where you can build your dashboard, visualizing the data.

What I'm excited about kind of in the future is building more solution-based applications using those tools. Not everyone has to build kind of custom solutions for every use case. That's something I'm also excited about, kind of using the toolbox that we have assembled and building more solution-based products on top of it.

[00:48:22] JM: Okay, Torkel. Thanks for coming on the show. It's been great talking.

[00:48:24] TO: Yeah. Great! Thanks.

[END OF INTERVIEW]

[00:48:35] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you

want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[END]

