# EPISODE 496

[INTRODUCTION]

**[0:00:00.2] JM:** A backend application can have hundreds of services written in different programming frameworks and languages. Across these different languages, log messages are produced in different formats. Some logging is produced in XML, some is produced in JSON, some is in other formats, and these logs need to be unified into a common format and centralized for any developer who wants to debug.

The popularity of Kubernetes is making it easier for companies to build this kind of distributed application where different services of different languages are communicating over a network with a variety of log message types. FluentD is a tool for solving this problem of log collection and unification.

In today's episode, Eduardo Silva joins the show to describe how FluentD is deployed to Kubernetes and the role of FluentD in a Kubernetes logging pipeline. We also discussed the company where Eduardo works, Treasure Data. The story of Treasure Data is unusual. The team started out doing log management but has found itself moving up the stack into marketing analytics, sales analytics and customer data management.

This story might be useful for anyone who is an open-source developer thinking about how to evolve your project into a business. I certainly found it useful. I hope you like this episode, and we'll continue to do Kubernetes related episodes for the next couple of weeks.

[SPONSOR MESSAGE]

**[0:01:37.2] JM:** Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes. You can quickly provision clusters to be up and running in no time while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked into any one vendor or resource. You can continue to work with the tools that you already know, such as Helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications off-line. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

Check out the Azure Container Service at aka.ms/acs. That's aka.ms/acs, and the link is in the show notes. Thank you to Azure Container Service for being a sponsor of Software Engineering Daily.

[INTERVIEW]

**[0:03:04.3] JM:** Eduardo Silva is an engineer with Treasure Data. Eduardo, welcome to Software Engineering Daily.

**[0:03:09.1] ES:**Thanks for the invitation. Happy to be here.

**[0:03:10.9] JM:** Today we're going to talk about FluentD. We're going to talk about Treasure Data. Let's start with FluentD. It's an open-source data collector. What is a data collector? What does that term mean?

**[0:03:21.0] ES:**Data collector, I will say that the description is — I will to start, because what's the problem that FluentD solves. When you have a lot of data being generated by applications, do we like to have some tools to unify this data and try to centralize the data? Because your aim, when you have many applications, you will like to do some kind of troubleshooting, monitoring or data analysis. If you have multiple applications, generating data from multiple sources with different formats, that is complex.

FluentD is a solution that allows you to take all these information from different place and centralize it in one database or multiple cloud services or any kind of a storage service that you're using.

**[0:04:04.8] JM:** Are we only talking about logging data?

**[0:04:06.4] ES:**Yeah, logging data.

**[0:04:07.5] JM:** Okay. Logging data. The error logs or the logs of user session that went well or any kinds of logs?

**[0:04:15.5] ES:**Actually, it depends. It depends on the user, because many people — You won't believe this, but they use logging for metrics. At some point you are collecting logs, but the insight from that data, there are metrics. So imagine that somebody has created a mobile application, they created the backend service and that backend service is generating data — I don't know, to the log files saying, "Okay. This player is buying this stuff, is looking at these items, is moving around this area of the map," and that information goes to the log files.

If you look carefully, okay, you are consuming log files, but sometimes inside the log files, the insights of that data is metrics. It can be a little bit of everything.

**[0:04:58.2] JM:** Right. By metrics, you're saying this could arguably be business data. This is not necessarily just logging data in the sense that I'm using these logs to debug or put out fires or make sure that my servers are overheating. This is like how many customers are making their way through the checkout line, for example. You could use that, aggregate that into metrics. That's a business level metric.

**[0:05:22.3] ES:**Yeah, because if you think about operations, from operations perspective, what you care about is about a — If you have any error, warnings, is the application crashing? You need to get this information to troubleshoot and see what's going on.

From a higher level of management, also you can get really insights from the applications. So I will say that it's a backbone for logging and the data is quite agnostic of the data. Of course, that data can be for troubleshooting to take business decisions or any kind of thing.

**[0:05:53.3] JM:** Yeah. Where in my infrastructure does data collection typically occur?

**[0:06:01.1] ES:**Typically occur locally. So because for safety, what people does is to write the log messages to the file system in the local server. So for that point of view, you need to run some data collector or agent where the files are located. On this case, FluentD.

But there are other kind of logging information. For example, think about firewalls or Cisco devices which are able to push the metrics or log information over the network, because they are vendors, the system is closed, but they are able to ship the logs. What do you do in that place is just, for example, from a FluentD perspective, start FluentD, listening for a TCP or a UDP port, lead your devices, talk to it.

**[0:06:46.6] JM:** Let's say I've got a Kubernetes deployment, I've got a bunch of containers. All of those containers are logging certain information. Am I going to run a collector? The FluentD collector on each of those containers? Sorry. On each of those pods I should say.

**[0:07:04.7] ES:**Yeah. Actually, in Kubernetes, it's like a special case, because [inaudible 0:07:09.0] Kubernetes, it's also that one application can be replicated, and replicated means that a same copy or you can have some kind of micro-service distributed in one node or maybe multiple nodes. How do you correlate the data? How do you capture that data?

In Kubernetes specifically, what we do is to deploy FluentD as a daemon set, and a daemon set is just a pod that runs on every node of the cluster, and with a node, I can refer to a bare metal machine, to a virtual machine. In Kubernetes specifically, you deployment FluentD on each node and of course FluentD will access to the Docker log files if you are using Docker, of course, because the log files of every container, of every pod, it's in the local file system.

Having the agent on that specific area allows you to collect the data, and what FluentD does, once it collects the information, it goes to the API server from Kubernetes to try to get some metadata, because when you deploy application on Kubernetes, you also have some information, like labels, annotations that allows you to identify an application. When you collect the logs, the application that is running inside the container, inside a pod is not aware about the context where its running.

From a logging perspective, you need to take out the logs and add that kind of metadata, which is the identification of that application. Meaning these application, this log information belongs to this application, to this container, to this pod name, to this node, and these are the labels and these are the annotations. Once you get all that information together, then FluentD can ship the logs outside of the node to your central database, can be Elasticsearch, Hadoop or anything.

**[0:08:54.8] JM:** You're describing the outline of a metrics pipeline, but I think you described mostly in terms of the node instance or the pod instance. On Kubernetes we have nodes, we have pods that are running on those nods and we have containers that are running within those pods and you're saying that we might — I'm sorry. Did you say that you might have a FluentD collector container that's running in each pod?

**[0:09:24.2] ES:**Yeah, exactly.

**[0:09:24.4] JM:** Okay. A sidecar.

**[0:09:25.8] ES:**No. No, that's a sidecar, because a sidecar belongs to the same pod. What we do is to deploy FluentD as a pod.

**[0:09:34.2] JM:** As its own pod.

**[0:09:35.6] ES:**Yeah. It's just like a separate pod in the same node, but the difference that this pod, it's running on every node but also is getting access to the local file system. Because on that physical space, you have the containers running and each container is generating their own log files. What you do is to let FluentD consume those log files and then ship the log files outside of that node.

**[0:09:58.5] JM:** Yes. The FluentD pod on each node needs to share a mounted volume with the application pod.

**[0:10:10.8] ES:**yeah, exactly. When you deploy FluentD as a daemon set, we have all these kind of state and configurations in the documentation, but actually when you deploy with a YAML file, what it does besides to deploy just FluentD, it ingests the right configuration and also mounts the volumes, and these kind of volumes are like the main Docker engine log files, so FluentD can get access to it.

**[0:10:33.8] JM:** Okay. Understood. Each of my nodes, let's say I've got a hundred node cluster and each of those nodes has a bunch of different pods and then on each node there's also a FluentD pod, and information from those different FluentD pods gets aggregated in some centralized place. What is that centralized place?

**[0:10:58.4] ES:**That is up to the user, but most of people use Elasticsearch, but there are other people using, for example, InfluxDB or Amazon S3. Having that also FluentD as a low collector, as an ecosystem, we have more than 700 plugins between inputs and outputs. From an output perspective, I mean what kind of destinations we can ship the logs. I always say that a huge advantage of FluentD that is quite flexible for different use cases and different backend storage services.

**[0:11:30.3] JM:** Okay. I have a lot of data that is being created. It's being logged on all of these different application — Being logged in all of these different FluentD pods, on all of my different nodes, and that data needs to be centralized somewhere. I'd just like to get a better idea of like a prototypical deployment where — Even if those all get put into Elasticsearch somewhere, there is some process of buffering all that data together, aggregating it and then shipping it out to Elasticsearch. I'd just like to better understand the pipeline between the FluentD pods that are running on all of my nodes and the centralized, perhaps Elasticsearch cluster with all of my log data.

**[0:12:11.4] ES:**Okay. Yeah. If we talk about the logging pipeline, it works like this — So you have so many faces inside FluentD. For example, you have the input, which means from where I'm going to collect the data. On this case, it will be the files, log files on the file system. Then you have the parts or stage when you say, "Okay. These log files are coming," for example in Docker context, "are coming in JSON." So I'm going to interpret and convert that JSON representation to my internal representation. At that moment, we're pretty fine.

Then we go to the filtering phase, and filtering means, "Okay. I'm going to filter this data," meaning I'm going to drop some messages, because maybe I don't care about them or maybe

I'm going to match these messages that I want. But also in Kubernetes, for us, in Kubernetes, a filter plugin can [inaudible 0:13:05.4] metadata. When we collect the data in Kubernetes, we collect the data, we parse this data, but in the filtering phase, what we do is to go to the API server, get the labels, their annotations and aggregate that information into the log line.

After that, we go to the buffering phase, because when you're dealing with logging, sometimes your backend services or where you are going to ship the logs, you face some problem. Service can be down. You can face some network outage or things go wrong. From a FluentD perspective, we always think, "Okay. Everything can fail," but how do we prepare for that? Everything can fail. The thing is how do you behave when that happens?

From a buffering perspective, FluentD supports a buffering in-memory or either in the file system. Most of people who say I cannot lose data, they go with buffering the file system. It's a slower option, but it's safer. But if you want something more performance, you will go with memory. Also, if you get some back pressure when you try to ship the logs, you can get some problems, because your memory will go up, your buffers go up.

**[0:14:15.3] JM:** Yes.

**[0:14:17.3] ES:**In terms of buffering, you can choose between a memory, the file system, but also you can define your own limits. Say, I cannot buffer more than two gigabytes of data. You can also decide your strategy about I'm going to drop the oldest records if I'm gaining new ones. It's very flexible to decide, "Okay. If I'm going to face some bad situation, I'm going to take this approach, and this approach means store until this limit of data, and if I reach that limit, [inaudible 0:14:47.2] start dropping the old data, or maybe [inaudible 0:14:49.8] stop consuming, a pause for a time." Then keep retrying and don't lose the data. Just keep retrying and try to deliver the data. Also, you can just say please retry — I don't know, every 10 minutes or by seconds. That's totally up to you.

**[0:15:05.3] JM:** We're talking about retries. You're talking about there are buffers of data that is going to be shipped from those pods that are on nodes, those pods that are logging information from your application pods.

**[0:15:21.7] ES:**Yeah, exactly.

**[0:15:22.8] JM:** Okay. They get buffered, because let's say I have a micro-service pod that processes payments, for example, and all of the data, all of the logging data that comes off of that payment processing pod gets written to a file system. That file system is shared with a pod that is running FluentD. FluentD is buffering that data, or you could have a shared memory volume, I guess, and that's where you're saying you could have — But that's less durable.

Overtime, the FluentD pod is going to say, "Okay. We've buffered enough messages. We're going to send that —" Send that where exactly?

**[0:16:01.5] ES:**Yeah, the destination. This is like an input and output, but we have many phases in [inaudible 0:16:05.7]. With output means, "Okay. I have this data, which was collected [inaudible 0:16:09.3], buffering," and now where this data is going to be sent? For example, that's up to the user. Some people say I'm going to use Elasticsearch, or some of them say I want to ship this data to Elasticsearch and also to Hadoop and ship the same data to both multiple places for different purposes.

**[0:16:29.4] JM:** Right.

**[0:16:31.2] ES:**For example, also we have some users who uses Splunk and they say we will like to filter the data that we're sending to Splunk, because we would not want to send the whole data, because we're going to pay more money, because that is the business model, and that is fine. But with FluentD, you can filter the data and send some of the relevant data to Splunk and maybe other data to Elasticsearch or a different backend service.

**[0:16:55.6] JM:** Okay, and the filters are defined at the node level, like basically on the client level.

**[0:17:03.8] ES:**Yeah. When you deploy FluentD, you deploy with a configuration, and this configuration is exactly as the login pipeline for where do you collect the information, how do you parse information, how do you filter, how do you buffer and where do you send this data.

**[0:17:17.7] JM:** There's no middleware that's sitting between my FluentD deployed pods and elastic search or HDFS.

**[0:17:27.3] ES:**Actually, it depends on the pattern that you want to follow. Most of people ships logs directly to the database over the network, but others sometimes decide, "Okay. This kind of namespace in Kubernetes, I'm going to collect the logs with FluentD from each node, but the destination will be a central FluentD, which is a main aggregator. From that main aggregator, I'm going to make my own decisions. Why? Because the teams that works on these specific namespace maybe cannot have access to Elasticsearch or Hadoop. They need a central place to aggregate the logs. From there, a different team takes care of that data. It depends on what's [inaudible 0:18:10.6] architecture, and every user has different needs. The thing is that FluentD is quite flexible even to collect, process the data or aggregate the data.

[SPONSOR MESSAGE]

**[0:18:28.3] JM:** Your company needs to build a new app, but you don't have the spare engineering resources. There are some technical people in your company who have time to build apps, but they're not engineers. They don't know JavaScript or iOS or android, that's where OutSystems comes in. OutSystems is a platform for building low code apps. As an enterprise grows, it needs more and more apps to support different types of customers and internal employee use cases.

Do you need to build an app for inventory management? Does your bank need a simple mobile app for mobile banking transactions? Do you need an app for visualizing your customer data? OutSystems has everything that you need to build, release and update your apps without needing an expert engineer. If you are an engineer, you will be massively productive with OutSystems.

Find out how to get started with low code apps today at outsystems.com/sedaily. There are videos showing how to use the OutSystems development platform and testimonials from enterprises like FICO, Mercedes Benz and Safeway.

I love to see new people exposed to software engineering. That's exactly what OutSystems does. OutSystems enables you to quickly build web and mobile applications whether you are an engineer or not.

Check out how to build low code apps by going to outsystems.com/sedaily. Thank you to OutSystems for being a new sponsor of Software Engineering Daily, and you're building something that's really cool and very much needed in the world. Thank you, OutSystems.

[INTERVIEW CONTINUED]

**[0:20:20.7] JM:** data can be in many different formats when it hits FluentD. You could have maybe a Go service that logs data in one type of format. Maybe you've got a Spring service somewhere else in your Kubernetes deployment, it's logging data in some XML kind of thing. You might want to unify these formats so that you can have all of your logging in one unified schema. My understanding is that FluentD is pretty good at unifying those schemas. Can you describe what FluentD does to bring those all into a consistent schema?

**[0:21:01.9] ES:**Yeah. In the input section of a logging pipeline, for example, when you are collecting data, you can define certain parsers. Sometimes the data comes in JSON. Before to jump into that, we have to say that internally, FluentD, manage the whole structure in a binary format, which is called msgpack. Msgpack is a spec that was created by the creator of FluentD before FluentD to manage a data serialization in an optimized way. FluentD internally use msgpack, but in order to convert that data that is coming from maybe a log file to msgpack, we need to have a process. That's your question.

For example, if I'm reading a log file, which is Apache log files, you as a human, you know that has a structure; IP address, time thumb, HTTP method and so on. But for a computer, that is just a byte of strings. So how do we convert that to a structure in msgpack? So that's where we apply some kind of regular expression with [inaudible 0:22:05.5] capture. We can define some kind of key values for this string that is coming in. You can define many sources for different formats. Everything that say, "Okay, from this path that belongs to Apache, I'm going to apply the Apache parses. Everything that comes from nginx, nginx parses. Everything from MySQL," and so on.

You can define many sources and assign it, write regular expressions or set up, because we have many built in regular expression for you. You can decide this as Apache, this is a MySQL, or this I a Java stack trace.

**[0:22:38.9] JM:** Yeah. This is where the community is pretty important, because the community probably is writing all those parsers. Somebody in the community says, “I want to use FluentD and there’s no nginx parses. Well, I’ll write it.”

**[0:22:49.6] ES:**Yeah, and that happens the whole day. Every day we get new changes and new plugins for FluentD in order to interpret or parse different kind of data. I think that without the community, FluentD will not be here as is today with 700 plugins, more than 500 contributors to the plugin’s ecosystem, because FluentD, as a core, is quite small, but the huge plus is it’s quite flexible. If you look at flexibility, it’s pretty much what Kubernetes is looking for. I don’t know if you were in the keynote this morning, but they are trying to promote the plugins, try to extend the core of Kubernetes through plugins, and FluentD did that from the beginning, and that’s why we have a huge community.

**[0:23:33.9] JM:** Yes. This model of basically, you deploy FluentD collectors and those FluentD collectors can parse the data, put it in the right format and just ship it straight to your centralized log database, I guess, the Elasticsearch or MySQL, wherever you’re keeping all your centralized logs.

The model of not needing a middleware, bug chunky middleware thing to throw all your logs into and have that big monolithic middleware taking care of things, that seems like it’s very in line with the Kubernetes design ethos.

Just to reinforce the architecture one more time. I’ve got my application. It’s deployed to Kubernetes. Kubernetes is sitting on a number of different nodes. Each of those nodes is running a number of different pods. Pods are at Kubernetes notion. Within each of those pods, you have one more containers, and you might have an application pod that is doing something, the Java micro-service, like let’s say processing payments or serving WordPress sites or whatever. Let’s say it’s processing payments, and you’ve also got a FluentD pod that is collocated in the same node, in the same server, and that FluentD pod is collecting the log data off of your application logs through a shared volume, and FluentD is shipping those to Elasticsearch.

Now, that I've restated that, I want to ask a little bit more about that process of buffering and chunking and queuing up that data, because if I've got a high volume transaction system of a banking system and I've got tons and tons of data that's coming through it. If I just send individual log messages, every single individual log message to my Elasticsearch elsewhere in my Kubernetes deployment, that's going to be  a ton of network overhead, and I would much rather chunk it into maybe all the logs that got collected over 30 seconds or something.

When data hits FluentD, it's put into a buffer in the FluentD pod, and the buffered events are chunked and put into a queue, and that queue is also on the FluentD pod?

**[0:25:56.7] ES:**Yeah. FluentD manages the queue. Yeah. It's not like another solution where you have the queues outside. We have built-in reliability for that. Yeah, of course, we don't send one message. We don't ship one by one. We do data chunking. Otherwise it would be a performance killer. It's pretty much what every — I don't know, high performance type it does. At Apache Kafka, that's the same. It tries to chunk data. You're not going to ship one or send records. Maybe you can do like — I don't know, a thousand or 2,000, but it's something that you can define.

Also, you can say, "Please [inaudible 0:26:32.6] the logs every 10 seconds, every 20 seconds." Then the buffer gets fills, right? Then you take the chunks from that buffer and you send the data out.

**[0:26:45.3] JM:** Is that a user-defined specification how often you're chunking?

**[0:26:48.9] ES:**Yeah. There are the defaults for each output plugin, but also you can say I want to ship every one minute. There are some people, some users that ship the data after two hours, for example, because they don't need to analyze the data in between.

**[0:27:04.6] JM:** On-the-fly.

**[0:27:04.6] ES:**Yeah.

**[0:27:05.6] JM:** What's an output plugin? What is an example of an output plugin?

**[0:27:09.8] ES:**Elasticsearch, Splunk, Kafka.

**[0:27:12.3] JM:** Okay. This is the recipient that I'm sending the data.

**[0:27:14.8] ES:**Yeah, the destination.

**[0:27:16.0] JM:** Okay. How are those chunks defined? Let's say if I'm running a banking application, maybe I've got some exceptions that get written to the logs. I've got some credits. I've got some debits. I've got all kinds of different logs that are happening on the same node. Is there a way of — I think you said there's filtering. What level does the filtering happen and how does it get turned into different —

**[0:27:44.0] ES:**Filtering happen after the parsing phase. You collect the data, you parse the data and you have the filters.

**[0:27:51.2] JM:** Is there a different queue for each filter?

**[0:27:54.4] ES:**Yeah. Each filter get the whole context of the data that goes out of the parser.

**[0:27:59.7] JM:** Okay. I see. There are multiple queues on one FluentD pod.

**[0:28:06.0] ES:**Yeah, because every time that you get — For example, if you define two input sources of data, like TCP or maybe log files, there's like two queues from a memory perspective when the data [inaudible 0:28:06.0] flow in.

**[0:28:17.9] JM:** Yeah. Actually, let me ask you this. What's the failover model for your FluentD pod?

**[0:28:25.6] ES:**Okay. For example, you can define HA. Okay, the first one if you are going to send a data, and for some reason the output plugin, like for Elasticsearch or for Kafka, it's failing. The strategy is that it's going to retry. But also you can look at some kind of load balancing, which you can say, "This is my primary destination. If it fails, please go to this second one as a backup." Maybe you can do some kind of round robin between multiple destinations for the same data. Multiple destination can say Elasticsearch one, Elasticseach two or Elasticsearch three.

**[0:29:01.2] JM:** You're talking about the failover for — If the output source fails.

**[0:29:04.0] ES:**Yeah.

**[0:29:05.3] JM:** Okay.

**[0:29:05.6] ES:**Yeah, you always do that kind of — Okay, if you want to have some failure mechanism to send the data, we have that in place because of the buffer queues and the [inaudible 0:29:15.3] mechanism. If you care about the things goes wrong while FluentD is working, that's why it's buffering. Always, we suggest that you use file system buffering, because if something happens and you need to restart FluentD [inaudible 0:29:30.0] resume later from those buffers.

**[0:29:31.6] JM:** Yeah.

**[0:29:32.8] ES:**And you lose your data.

**[0:29:34.5] JM:** Yeah. It now works fine, because just sharing a file system, the FluentD node goes down. Well, it just figures out where in the file system it should reboot from and can replay — Not really replay, but interesting.

Is FluentD reading — If I have my application server, it's writing to log files on disc and FluentD is pulling those log files into memory on the FluentD pod and then chunking them in memory and then it's sending them over the wire essentially.

**[0:30:07.5] ES:**Optionally, because when you say buffering, you can specify in the configuration, “I want to use memory or the file system.” Because of the nature of Kubernetes, we always suggest go to buffering the file system, because otherwise your pod can consume too much memory if you have some back pressure, because you cannot ship the logs, because of a network outage or because of the services are not fast enough. You get that pressure and your queues goes up. It’s better if your file system goes up than your memory, because that will — It can end up that your container can be killed because of the memory limits.

**[0:30:46.1] JM:** Okay. If my FluentD pod is going to send a chunk of logs from the file system on the application pod — Sorry. From the file system that is shared by the FluentD and the application pod. If I’m going to send a chunk of logging data, FluentD is able to just read that directly from the file system and ship it over in a network call? It doesn’t need to pull it into memory in an intermediary step?

**[0:31:14.5] ES:**No. Of course, when FluentD runs, the first thing it asks from a low level perspective is you open a file, a file descriptor. Do you read from one position? Do you read — I don’t know, a chunk of one megabyte or 50 kilobytes. I don’t know. It’s up to you. that is already memory. When you get that information in memory [inaudible 0:31:33.2] when parsing and filtering happens. We always open the files in read only mode. We don’t touch the files. That’s that idea.

**[0:31:41.2] JM:** Of course.

**[0:31:41.9] ES:**Once we modify those kind of a piece of data in memory, that’s when we do buffering in the file system. Meaning that this data was already processed and it’s ready to be out of FluentD or to be shipped to a database or a cloud service.

**[0:31:57.8] JM:** Yup. Okay. That makes sense. Describe more some of the patterns that people are using for — Where they put this log data. You mentioned I can send this log data to Kafka. I can send this log data to Elasticsearch. I can send this log data to Splunk. Let’s take the Kafka example. Why am I going to ship my data from FluentD to Kafka?

**[0:32:22.1] ES:**I think that Kafka is one of the great experience in terms of data persistency, because, okay if you want to have data persistency, you will like to have maybe a database. But maybe a database is too complex if you want to have the data and save it for a long time of period and just consume fraction of that data.

What Kafka allows you is to store the data in a persistent way and also let the other third-party vendors, like consumers, consume these data based on topics of specific channels, but your data persists. Because in FluentD, we don't care about data persistency, just in the buffering side. But Kafka is mostly about streaming data. Meaning, for example, if I'm a consumer, "Okay. I want to get the whole logs that is coming from these Kubernetes namespace, and this is a topic. They subscribe to this topic in Kafka, and while FluentD ingests this data into Kafka, Kafka saves this information to the file system, then consumers can get this data from Kafka.

Also, you can say, "Please, Kafka, show me only — I don't know, the last 20 messages or send me the messages from the beginning." Something that FluentD does not have, because it took the logs and just ship the logs. But Kafka has that kind of persistency, which is a really good complement for many people.

[SPONSOR MESSAGE]

**[0:33:55.2] JM:** The octopus, a sea creature known for its intelligence and flexibility. Octopus Deploy, a friendly deployment automation tool for deploying applications like .NET apps, Java apps and more. Ask any developer and they'll tell you that it's never fun pushing code at 5 p.m. on a Friday and then crossing your fingers hoping for the best. We've all been there. We've all done that, and that's where Octopus Deploy comes into the picture.

Octopus Deploy is a friendly deployment automation tool taking over where your build or CI server ends. Use Octopus to promote releases on prem or to the cloud. Octopus integrates with your existing build pipeline, TFS and VSTS, Bamboo, Team City and Jenkins. It integrates with AWS, Azure and on-prem environments. You can reliably and repeatedly deploy your .NET and Java apps and more. If you can package it, Octopus can deploy it.

It’s quick and easy to install and you can just go to octopus.com to trial Octopus free for 45 days. That’s octopus.com, O-C-T-O-P-U-S.com.

[INTERVIEW CONTINUED]

**[0:35:27.3] JM:** Talking more about failover, if the network gets really flaky and some of my FluentD nodes or some of my FluentD pods cannot push data to their output destinations because of flaky networks, is there — I guess the penalty is not super severe, because assuming that we’re reading from a shared file system, it’s not problematic, right?

**[0:35:57.8] ES:**Yeah. It is not.

**[0:36:01.1] JM:** Okay. If I’m FluentD and I’m sharing a memory volume with the application service and the network gets flaky for like an hour or two hours, then what are the consequences there?

**[0:36:17.6] ES:**Okay. Actually, in FluentD, we don’t have a memory volume concept. Yes, FluentD runs both [inaudible 0:36:24.4] in memory and it takes advantage of their memory space. For example, if you cannot ship the logs for two hours, something is happening. If you are reading more data, your queues or your buffers is going up and maybe you’re going to hit a limit. You define in the configuration, when the hit is limit, I’m going to drop the oldest chunk or just drop everything and start from zero. You can define your strategy, but it’s something that is up to the user. We are flexible enough to say, “Okay. If you face this network outage problem, we’re going to retry. Okay. Let us know how many times you want to retry, or let’s keep retrying forever.” But if you retry forever, your buffers fills up. What will be your strategy for that special failure scenario, and by default you’re going to drop the old data. The thing is that you’re not going to consume all your hard disc. You’re not going to consume all your memory available. The thing is how to keep running in a safety way.

**[0:37:28.5] JM:** Understood. The scalability model. As my banking transaction application pod scales up, is there a predefined way for how my FluentD pods are going to scale up?

**[0:37:40.9] ES:**What do you mean? Scaling up on different nodes?

**[0:37:43.1] JM:** Sorry. If my application pods are scaling up, that probably means the log data that they're putting off is scaling up. Do I need to scale FluentD pods along with that?

**[0:37:52.9] ES:**Actually, when you define FluentD as a daemon set, you always try to prepare it for the worst case. If an application might scale a lot, you prepare for that. For example, in the latest version of FluentD, like 1.0, we have the multi-process support so you can start many workers to prepare to that specific scenario.

**[0:38:14.7] JM:** Interesting. I think we've covered FluentD pretty thoroughly at this point. What are some other patterns that people are using together with FluentD in — We're here at Cube-Con. You're talking to people how they're using Kubernetes and how they're using FluentD. What are some other patterns you're seeing?

**[0:38:33.4] ES:**We have patterns like — Well, I have seen like, "Oh, okay. You know what? I'm using FluentD. I ship my logs to Kafka, and then I conceal my Kafka logs with FluentD again."

**[0:38:45.5] JM:** Why?

**[0:38:46.6] ES:** That's my question, because if you ship the data to Kafka, the data was processed, but maybe you need to process the data again, and they use FluentD to consume the logs back and maybe [inaudible 0:38:45.5] metadata or any special stuff [inaudible 0:39:03.1] processing on both sides, and that's interesting.

**[0:39:07.3] JM:** Yeah, you could add geo-data, for example.

**[0:39:10.5] ES:**Exactly.

**[0:39:10.8] JM:** Interesting.

**[0:39:12.4] ES:**Yeah. On that specific use case, you say that, "Okay. Kafka is for persistency and data streaming," but on both side, because FluentD also can read data from Kafka. You have a producer, FluentD producer and you have a FluentD consumer, and both of them are doing data processing.

**[0:39:28.4] JM:** Right. Yeah, Kafka is often used for this event-sourcing model where you have multiple consumers, multiple databases that are writing their materialized views of events that come off of Karka. Maybe you log application level events, like a banking transaction, and that banking transaction includes an IP address, and you send those logs to Kafka. Now you've got well-formed log messages sitting in Kafka and maybe your Elasticsearch cluster wants to read those. But maybe also you've got some other stats database that is logging all of the users that came from Uzbekistan, and so you can take the IP addresses of those logs and enrich it with metadata, like geo-tags and say, "Okay. Now, I've got an IP address. I want to transform that and find out if that IP address was in Uzbekistan, and if it's in Uzbekistan, write it to whatever materialized — The Uzbekistan materialized view."

**[0:40:37.3] ES:**Yeah. I think that's a really interesting case. The most — Taking that same example, sometimes FluentD, from a consumer perspective, take the data in from Kafka. Do the processing and generate new topics and ingest the data back into Kafka.

**[0:40:54.6] JM:** Oh! What's an example?

**[0:40:57.1] ES:**Let's say geo information.

**[0:40:59.4] JM:** Okay. Now you would have a topic, the geo information topic.

**[0:41:03.2] ES:**Exactly. Yeah. The same logging information. Yeah. We have seen cases like crazy.

**[0:41:08.9] JM:** Really?

**[0:41:09.1] ES:**Yeah, but that is good, because we learn about the different use cases, because everybody has a different need. Data processing is something that everybody needs, and persistency is something that everybody needs. I think that FluentD with Kafka is a really good complement for that.

**[0:41:26.6] JM:** Yeah. Go a little deeper there. Talking about Kafka specifically, like we've done a lot of shows about Kafka event-sourcing, but you're creating a vision where Kafka is really this place where we're storing multiple copies of the same piece of data. It's like you're presenting a vision where it's like the database — Well, maybe a database of record for everything. I don't know. Give me some more color. What are some other interesting Kafka use cases you've seen?

**[0:42:00.5] ES:**I think that the positive thing about Kafka from a general design perspective is it allow many users to subscribe to certain information. From that perspective, it makes sense that the data that you have there needs to be persistent. Also, you can process the data and ingest that data back.

Most of the cases that I hear a FluentD perspective, I'm not a Kafka expert, but it's just what I hear from FluentD users. They use it for persistency and data processing and insert that information back into Kafka for other consumers.

**[0:42:34.6] JM:** Wow! Interesting. Your company, Treasure Data, has an enterprise FluentD platform. But then you also have these higher level services. Treasure Data has marketing analytics, sales operations analytics, a data warehousing solution. This is kind of unique. It's not very often that you see a company that has a big open-source focus. They are working — FluentD — I'm going to these talks here at Cube-Con and the standard way that people are doing logging seems to be FluentD — Oh, god! What is it? FluentD, Elasticsearch and Kibana. Yeah, that's right. Okay. You couldn't expect that technology to also come from a company that's doing these like kind of higher level analytics tools. Why is that? How did your company turn out that way?

**[0:43:30.3] ES:**It's like a story. When the company started, it started like a service to sort data, agnostic data. But before to start the data, you need to collect the data. That was FluentD. FluentD before the cloud service existed, existed FluentD. Since the CTO, the CEO and all the cofounders has a really strong background in open-source. They decided that FluentD will be open-source and will be open-source the core DNA of the company. You know that a company also has the close product, but if we are going to make something open-source, that will not take out value of our service. That is fine. Because it's going to improve other kind of technologies and other people, and that is fine.

I think that that was the best decision since FluentD was created, because if you look carefully, FluentD started to being used for different use cases.

**[0:44:29.1] JM:** Yeah.

**[0:44:30.7] ES:**That's why you have 700 plugins around. More than 500 people has contributed to it. Then you can say that in the DNA of the company, there's a strong background about how to manage data and also an open-source.

**[0:44:46.9] JM:** Did you say you have a FluentD, like an enterprise version also?

**[0:44:52.9] ES:**Yeah. Actually,  all these type of FluentD, Kubernetes, you won't believe it, but many big companies, because some names I cannot mention here. They are using FluentD heavily internally and there are some point they say, "Okay. Who's behind FluentD, or who are the primary sponsors?" We are. Many people don't know that Treasure Data is behind FluentD.

**[0:45:16.1] JM:** It's a good sales model.

**[0:45:16.9] ES:**Yeah. It's like — Okay, they came to us and say, "You know what? We are deploying this." But we need, for example — We need a special connector for Splunk," and we ask why. Because the Splunk forwarder send the whole data to Splunk and I need to pay more money, and FluentD offers filtering, so I could cut my bills.

**[0:45:35.5] JM:** Wow! They filter it in order to send less data to Splunk.

**[0:45:41.0] ES:**Yeah. I think the end is not to send less data. it's just to send the data that you care about and not pay for data that you don't care about. Yeah, I don't want to make for [inaudible 0:45:51.9] but I'm saying how this works. Others say, "You know what? I'm using Kafka, but I need some special security with [inaudible 0:45:51.9] on special TLS changes."

Also others say, “We need support. We need professional support, because this is running, because sometimes we get some problems and we don’t want to have our operations people taking care of these, because we have many things running.”

Okay. We started the FluentD enterprise program this year as a support service, and then we started creating the enterprise FluentD product, which has an enhanced connectivity for Splunk certified with security for Kafka. We’re building a new UI to manage FluentD remotely, a new FluentD who has an API so you can manage it remotely and that kind of things that you will expect that an enterprise company needs.

**[0:46:43.4] JM:** Then the stuff like marketing analytics and sales operations analytics, this is a totally different division of the company that then the dev ops/Kubernetes deployment person — It’s just not obvious to me what is the — How you get the upmarket? Because I think typical — What I imagine is maybe you could say, “Okay. We’ve got this FluentD data collector here. We could throw some other stuff in that data collector and have it doing more transforms on marketing data.” Yeah, I guess give me a perspective for how you decided to move up market from the data collection and filtering business to the marketing analytics business.

**[0:47:25.5] ES:**Actually, the company is on the CDP business, which is customer data platform. If you think about marketing — Well, if you think about data, data is data and it does not work until you get some insights from it or you provide the right interface to take advantage of that data. Sometimes data is just a backup, but sometimes data for you, it’s relevant to take a business decision.

If you think about your customer data platform, you think about the marketing people. They have different kind of concepts, like segments, personas and so on. I’m not expert on marketing. From a company perspective, we started to get a lot of traction for the marketing division for each company, saying that we need  a platform to unify our data, such as FluentD, but at a bigger scale where they say, “[inaudible 0:48:15.0] all my data from my Twitter user, from Facebook or for different social media, but I need to unify this data and correlate this data.” I cannot do this at home, but you guys, you have a service to store data. We said, “Okay. We’re going to build the whole interfaces for marketing.” So we provide you the whole concepts, abstracted concepts of segments, personas using your own data.

The cool thing is that the data belongs to you. If you want to export the data for visualization, for example, to Tableau, you can do it. You can export to Amazon Red Shift. It's very much the same concept of taking data in and taking data out, but also providing a lot of tools that allows you to process the data and also get the right insights that are right for your business.

**[0:49:04.2] JM:** And it runs on FluentD? Is it built —

**[0:49:06.6] ES:**No. FluentD, we use as a data collector.

**[0:49:08.3] JM:** Wow! This is like totally agnostic of FluentD, but it has —

**[0:49:12.0] ES:**Exactly.

**[0:49:12.2] JM:** Yeah.

**[0:49:12.6] ES:**Also, you don't need FluentD. You can ingest using your own code, your own scripts. We have SDKs for mobile phones. We have customers who are from healthcare, from gaming, from security companies. There are many areas, because at the end, data is data. The thing is you need to be reliable. So as soon as you get the data, to store the data and offer the right interface to query this data and export this data.

**[0:49:37.8] JM:** Is there any overall mission statement of the company? Because I mean what you're describing makes complete sense, but it's just funny that you ended up with marketing server here that does not even leverage the technology of FluentD data collector service over here.

**[0:49:54.0] ES:**Yeah. Having that the open-source division started to grow, and we [inaudible 0:49:57.3] in the last years with all these about Docker containers, Kubernetes. If you look carefully, FluentD has been used by the whole [inaudible 0:50:06.2] Kubernetes distributions, like Tectonic, Open Shift and so on.

Yeah. I think that from a business perspective sometimes can generate some confusion, because you say, “Okay. Our statement right now, we care about data, customer first, and we try to solve complex problems.” We solve the problem for our customers. So they don’t need to focus on this. We just give the results.

**[0:50:30.4] JM:** Yeah.

**[0:50:31.8] ES:**Also, open-source is in our DNA, and you can see that Treasure Data has an open-source section, where we don’t have just FluentD. We have FluentBit, which is like a C language version of FluentD. We have Bulk, which is like a FluentD, but in order to move big amount of data, like a [inaudible 0:50:48.0] data. [inaudible 0:50:49.8], a streaming data platform service. We have many things as an open-source company.

It’s hard to say an open-source company, because we have a core business. We have private products, but we try to keep the balance and always be a good citizen with the open-source community.

**[0:51:06.7] JM:** Eduardo Silva, thanks for coming on Software Engineering Daily.

**[0:51:09.0] ES:**Okay. Thank you so much.

**[0:51:10.2] JM:** Wonderful.

[END OF INTERVIEW]

**[0:51:15.2] JM:** If you enjoy Software Engineering Daily, consider becoming a paid subscriber. Subscribers get access to premium episodes as well as ad free content. The premium episodes will be released roughly once a month and they’ll be similar to our recent episode, The Gravity of Kubernetes. If you like that format, subscribe to hear more in the future. We’ve also taken all 650+ of our episodes. We’ve copied them and removed the ads, so paid subscribers will not hear advertisements if they listen to the podcast using the Software Engineering Daily app.

To support the show through your subscription, go to softwaredaily.com and click subscribe. Softwaredaily.com is our new platform that the community has been building in the open. We'd love for you to check it out even if you're not subscribing, but if you are subscribing, you can also listen to premium content and the ad free episodes on the website if you're a paid subscriber. So you can use softwaredaily.com for that.

Whether you pay to subscribe or not, just by listening to the show, you are supporting us. You could also tweet about us or write about us on Facebook or something. That's also additional ways to support us. Again, the minimal amount of supporting us by just listening is just fine with us. Thank you.

[END]