What Should We Expect from Centralized Logging?

In this lesson, we will see what expectations do we have from centralized logging.

WE'LL COVER THE FOLLOWING

Which product to use for centralized logging?

We explored several products that can be used to centralize logging. As you saw, all are very similar, and we can assume that most of the other solutions follow the same principles. We need to collect logs across the cluster. We used **Fluentd** for that, which is the most widely accepted solution that you will likely use no matter which database receives those logs (Azure being an exception). Log entries collected with **Fluentd** are shipped to a database which, in our case, is **Papertrail**, *Elasticsearch*, or one of the solutions provided by hosting vendors. Finally, all solutions offer a UI that allows us to explore the logs.

I usually provide a single solution for a problem but, in this case, there are quite a few candidates for centralized logging. Which one should you choose? Will it be **Papertrail**, **Elasticsearch-Fluentd-Kibana stack (EFK)**, **AWS CloudWatch**, **GCP Stackdriver**, **Azure Log Analytics**, or something else?

When possible and practical, I prefer a centralized logging solution provided as a service, instead of running it inside my clusters. Many things are easier when others are making sure that everything works. If we use Helm to install **EFK**, it might seem like an easy setup. However, maintenance is far from trivial. **Elasticsearch** requires a lot of resources. For smaller clusters, compute required to run **Elasticsearch** alone is likely higher than the price of **Papertrail** or similar solutions. If I can get a service managed by others for the same price as running the alternative inside my own cluster, service wins most of the time. But, there are a few exceptions.

Retaining control of core components

I do not want to lock my business into a service provider. Or, to be more precise, I think it's crucial that core components are controlled by me, while as much of the rest is given to others. A good example is VMs. I do not honestly care who creates them, as long as the price is competitive and the service is reliable. I can easily move my VMs from on-prem to AWS, and from there to, let's say, Azure. I can even go back to on-prem. There's not much logic in the creation and maintenance of VMs. Or, at least, there shouldn't be.

Switch from one provider to another

What I genuinely care about are my applications. As long as they are running, fault-tolerant, highly available, and their maintenance is not costly, it doesn't matter where they run. But, I need to make sure that the system is done in a way that allows me to switch from one provider to another, without spending months in refactoring. That's one of the big reasons why Kubernetes is so widely adopted. It abstracts everything below it, thus allowing us to run our applications in (almost) any Kubernetes cluster. I believe the same can be applied to logs. We need to be clear what we expect, and any solution that meets our requirements is as good as any other. So, what do we need from a logging solution?

> 🔍 We need logs centralized in a single location so that we can explore logs from any part of the system.

> 🔍 We need a query language that will allow us to filter the results.

> 🔍 We need the solution to be fast.

Simple operations with logs

All of the solutions we explored meet those requirements. **Papertrail**, **EFK**, **AWS CloudWatch**, **GCP Stackdriver**, and **Azure Log Analytics** all fulfill those requirements. **Kibana** might be a bit prettier, and **Elasticsearch's** query language might be a bit richer than those provided by the other solutions. The importance of prettiness is up to you to establish. As for **Elasticsearch's** query language being more powerful... It does not really matter. Most of the time, we need simple operations with logs. Find me all the entries that have specific keywords. Return all logs from that application. Limit the result to the last thirty minutes.

> 🔍 When possible and practical, logging-as-a-service provided by a third party like Papertrail, AWS, GCP, or Azure is a better option than to host it inside our clusters.

Using a Third-party something-as-a-service solution

With a service, we accomplish the same goals, while getting rid of one of the things we need to worry about. The reasoning behind that statement is similar to the logic that makes me believe that managed Kubernetes services (e.g., EKS, AKS, GKE) are a better choice than Kubernetes maintained by us. Nevertheless, there might be many reasons why using a third-party something-as-a-service solution is not possible. Regulations might not allow us to go outside the internal network. Latency might be too big. Decision-makers are stubborn. No matter the reasons, when we can not use something-as-a-service, we have to host that something ourselves. In such a case, **EFK** is likely the best solution, excluding enterprise offerings that are out of the scope of this course.

If **EFK** is likely one of the best solutions for self-hosted centralized logging, which one should be our choice when we can use logging-as-a-service? Is **Papertrail** a good choice?

Solutions offered by Cloud Providers

If our cluster is running inside one of the Cloud providers, there is likely already a good solution offered by it. For example, EKS has **AWS CloudWatch**, GKE has **GCP Stackdriver**, and AKS has **Azure Log Analytics**. Using one of those makes perfect sense. It's already there, it is likely already integrated with the cluster you are running, and all you have to do is say yes. When a cluster is running with one of the Cloud providers, the only reason to choose some other solution could be the price.

> 🔍 Use a service provided by your Cloud provider, unless it is more expensive than alternatives. If your cluster is on-prem, use a third-party service like **Papertrail**, unless there are rules that prevent you from sending logs outside your internal network. If everything else fails, use **EFK**.

At this point, you might be wondering why I suggest using a service for logs, while I proposed that our metrics should be hosted inside our clusters. Isn't that contradictory? Following that logic, shouldn't we use metrics-as-a-service as well?

Our system does not need to interact with our logs storage. The system needs to ship logs, but it does not need to retrieve them. As an example, there is no need for `HorizontalPodAutoscaler` to hook into **Elasticsearch** and use logs to decide whether to scale the number of Pods. If the system does not need logs to make decisions, can we say the same for humans? What do we need logs for? We need logs for debugging. We need them to find the cause of a problem. What we do NOT need are alerts based on logs. Logs do not help us discover that there is an issue, but to find the cause of a problem detected through alerts based on metrics.

Wait a minute! Shouldn't we create an alert when the number of log entries with the word *ERROR* goes over a certain threshold? The answer is no. We can (and should) accomplish the same objective through metrics. We already explored how to fetch errors from exporters as well as through instrumentation.

When to explore logs

When to explore logs

What happens when we detect that there is an issue through a metrics-based notification? Is that the moment we should start exploring logs? Most of the time, the first steps towards finding the cause of a problem does not lie in exploring logs, but in querying metrics. Is the application down? Does it have a memory leak? Is there a problem with networking? Is there a high number of error responses? Those and countless other questions are answered through metrics. Sometimes metrics reveal the cause of the problem, and in other cases, they help us narrow it down to a specific part of the system. Logs are becoming useful only in the latter case.

> 🔍 We should start exploring logs only when metrics reveal the culprit, but not the cause of the issue.

Features from a logging solution

If we do have comprehensive metrics, and they do reveal most (if not all) of the information we need to solve an issue, we do not need much from a logging solution. We need logs to be centralized so that we can find them all in one place, to filter them by application or a specific replica, to narrow the scope to a particular time frame, and to search for specific keywords. That's all we need. As it happens, almost all solutions offer those features. As such, the choice should be based on simplicity and the cost of ownership.

Solutions simple to use and manage

Whatever you choose, do not fall into the trap of getting impressed with shiny features that you are not going to use. I prefer solutions that are simple to use and manage. **Papertrail** fulfills all the requirements, and it's cheap. It's the perfect choice for both on-prem and cloud clusters. The same can be said for **CloudWatch (AWS)**, **Stackdriver (GCP)**, and **Log Analytics (Azure)**. Even though I have a slight preference towards **Papertrail**, those three do, more or less, the same job, and they are already part of the offer.

If you are not allowed to store your data outside your cluster, or you have some other impediment towards one of those solutions, **EFK** is a good choice. Just be aware that it'll eat your resources for breakfast, and still complain that

it's hungry. **Elasticsearch** alone requires a few GB of RAM as a minimum, and you will likely need much more. That, of course, is not that important if you're already using **Elasticsearch** for other purposes. If that's the case, **EFK** is a no-brainer. It's already there, so use it.

Q

We should create an alert when the number of log entries with the word *ERROR* goes over a certain threshold.

COMPLETED 0%

1 of 1

In the next lesson, we will revise and test the concepts of this chapter through a short quiz.