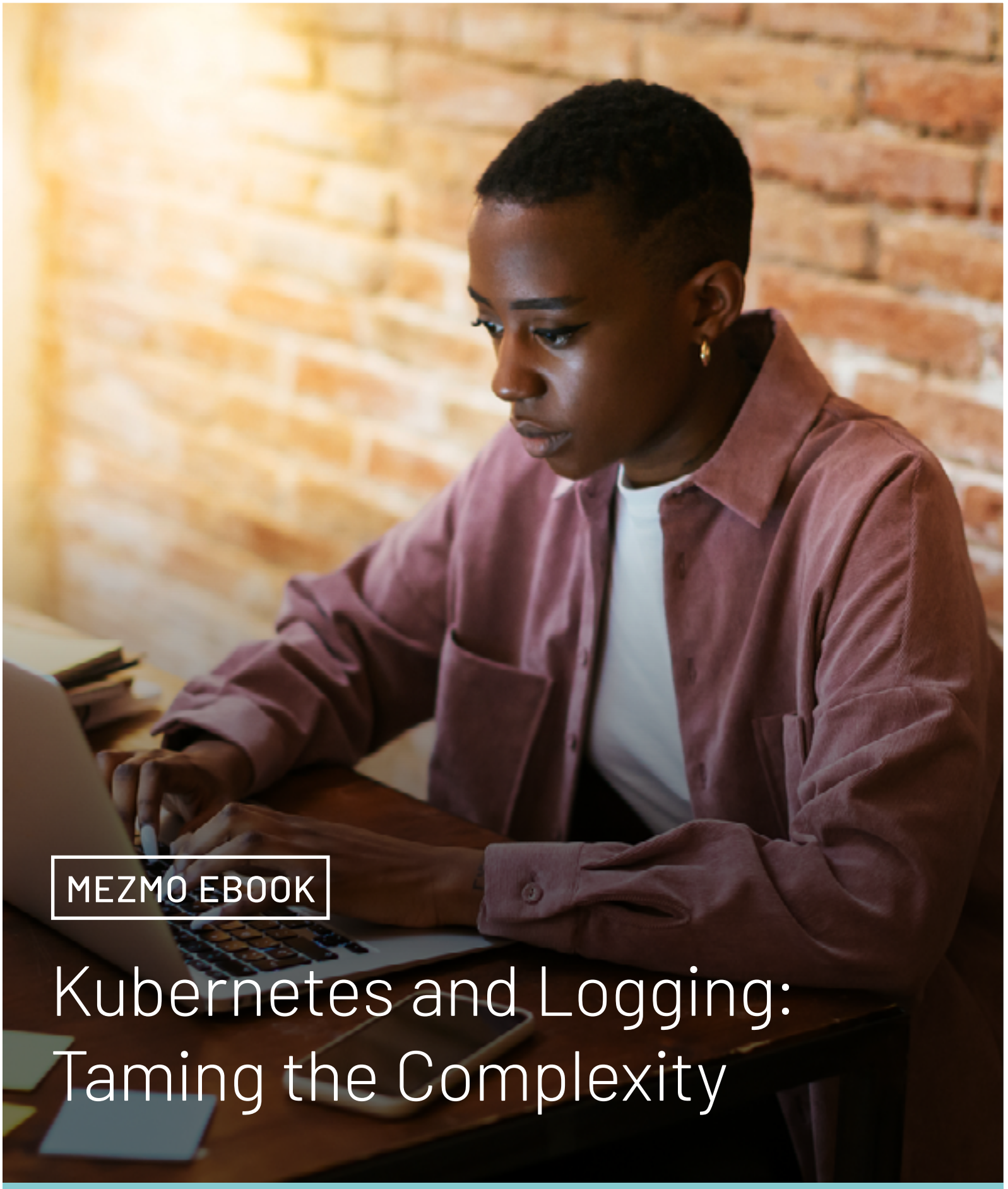

MEZMO EBOOK

Kubernetes and Logging: Taming the Complexity

Log management for Kubernetes doesn't have to be painful. Instead of learning a totally new set of methods and tools, you can adapt and extend your existing log management strategy to fit Kubernetes.

mezmo

TABLE OF CONTENTS

INTRODUCTION

We'll admit it: Even though we here at Mezmo are all about logging, we know that log management and analytics are not on the top of most IT professionals' lists of fun tasks. For most teams, log management is just something you have to do, not something you look forward to doing.

That doesn't mean log management has to be a pain, though. With the right techniques and technology, you can make logging less of a chore and more of a streamlined part of your workflow. What's more, when done properly, log management makes other parts of your job easier by helping you to work more efficiently with the rest of your technology stack.

To prove these points and provide some real-world context on what effective log management looks like for a modern team, this eBook offers guidance on logging for Kubernetes. It begins with an overview of Kubernetes logging basics, then explains how and why to build a log management strategy tailored to Kubernetes.

As we'll see, even though Kubernetes is a fundamentally new type of technology for most IT pros today, log management for Kubernetes doesn't have to be painful. Instead of learning a totally new set of methods and tools, you can adapt and extend your existing log management strategy to fit Kubernetes. In doing so, you'll gain critical visibility into your Kubernetes clusters, making the monumental task of Kubernetes administration much easier.

CHAPTER 1: KUBERNETES LOGGING ESSENTIALS

Kubernetes is a container orchestration tool, but its functionality extends far beyond just orchestrating containers in a narrow sense. It offers a range of additional features that—to a limited extent—address needs such as load balancing, access control, security policy enforcement, and even logging and monitoring. Indeed, Kubernetes's broad functionality has led some folks to call it an "operating system" in its own right.

That said, many of the extra features that Kubernetes provides are not full-fledged solutions. On the security front, for example, Kubernetes provides some tools to prevent abuse. Still, it's hardly a sufficient solution on its own to address every security aspect of a given workload. For load-balancing, Kubernetes manages the way traffic is distributed to workloads within a cluster, but it's not as if it will load-balance your entire network.

The same type of limitation applies to Kubernetes's logging and monitoring features: While Kubernetes offers some basic logging and monitoring facilities, it's a far cry from a complete logging and monitoring solution.

Because of these limitations, understanding what Kubernetes can do natively, and when it requires help from external tools to address a particular need, is critical for deploying Kubernetes successfully.

With that reality in mind, let's take a look at Kubernetes's built-in logging and monitoring functionality and what's missing out-of-the-box on the logging and monitoring front in Kubernetes.

Native Kubernetes Logging and Monitoring Features

The built-in monitoring and logging tooling in Kubernetes is basic but effective for certain types of needs. Essentially, it boils down to two types of functionality: log access and log storage.

Using Kubectl to Access the "Logs" of Each Kubernetes Pod

Using a command like **`kubectl logs [pod name] -c [container name]`**, you can read the "logs" of every container running within a Kubernetes cluster.

The caveat here (and the reason "logs" is in scare quotes) is that the "logs" you can access this way are not actually log files in the traditional sense, but rather the stdout and stderr messages generated by containers as they run. Kubernetes collects this data and stores it in a file that you can access with kubectl.

Log Data Stored on Kubernetes Nodes

Kubernetes also logs data from various components of Kubernetes itself to files that you can access by logging into Kubernetes nodes directly.

Specifically, the Kubernetes master node (or nodes, if you have multiple masters) offers log data at /var/log/kube-apiserver.log, /var/log/kube-scheduler.log, and /var/log/kube-controller-manager.log, and each worker node has /var/log/kubelet.log and /var/log/kube-proxy.log files.

When Kubernetes's Native Logging and Monitoring Features Are Useful

The two types of logging facilities described above come in handy if you need to check information quickly or research a one-time event that occurred within your Kubernetes cluster. They're kind of akin to the information you could get by running `dmesg | tail` in a Bash shell on a Linux server, in that they are a quick and easy way of accessing small amounts of information, especially if you already know what kind of information you are looking for.

Missing Logging and Monitoring Features in Kubernetes

When it comes to more complex logging and monitoring needs, however, Kubernetes alone doesn't cut it. Kubernetes lacks native features for the following critical tasks:

Log Rotation

Although Kubernetes creates logs for each container and for Kubernetes itself, it doesn't automatically rotate or archive this data. On the contrary, it expects you to handle log rotation, and if you don't, you risk having your log files eat up all of the storage space on your nodes.

(For the record, we should point out that most Kubernetes distributions do set up log rotation facilities for you when you install them. However, Kubernetes itself doesn't handle log rotation, and if your distribution doesn't provide a solution for this task automatically, you need to implement one manually.)

Log Aggregation and Centralization

Likewise, Kubernetes doesn't offer any tools for aggregating log data in a single location or merging similar types of logs together. It lets you view logs for containers and nodes on an individual, one-off basis, which is useful if you need to pull some quick information about a particular container or node.

But, what if you want to monitor all of your containers at once, or trace monitoring data related to a particular event across multiple containers or nodes? The only way to do that natively in Kubernetes would be to access each log manually, which is not practical to do at scale.

"The built-in tooling in Kubernetes is basic for certain types of needs. It boils down to two types of functionality: log access and log storage."

Log Analysis

Kubernetes will show you log data, but it does nothing to help you read or interpret it.
It doesn't offer visualization features, or even alerts or notifications about monitoring events that could signal a problem.

Scalable Logging

In most Kubernetes distributions, the container logs available from kubectl are limited to a mere 10 megabytes in size. Kubernetes automatically deletes older data if the logs exceed this limit.

This may not be much of an issue if you only have a few containers running and generating log data. But if you have dozens, your log file won't be of much use because it won't be large enough to accommodate all of your containers.

Access Historical Monitoring Data

For similar reasons, accessing log data through kubectl is not very helpful if you need to access information about a historical event. Kubernetes may have deleted that data in order to keep the log file under 10 megabytes.

Chapter Conclusion: Filling In Kubernetes Logging and Monitoring Gaps

In short, Kubernetes offers enough built-in logging and monitoring functionality to allow you to monitor workloads on a small scale or research one-off events that occurred in the recent past.

However, Kubernetes on its own falls far short of offering a full-fledged logging and monitoring solution. To fill the gaps, you need to pair Kubernetes with external tools that can handle log rotation and aggregation, store historical log data over the long term, and provide you with the analytics features you need to achieve true monitoring visibility.

There are different ways to implement this, with the most common being to run a "sidecar" container in each pod that interfaces between the pod and an external log manager. Setting up this type of solution requires a little extra work. No matter how you ensure you gather data for all of your stack to fill the gaps that Kuberentes has, it's critical to do so if you want to be able to monitor and provide logging for your Kubernetes workloads at scale.

CHAPTER 2: HOW TO EVOLVE YOUR EXISTING LOGGING STRATEGY FOR KUBERNETES

It's one thing to build a Kubernetes log management strategy that only needs to support Kubernetes. But most organizations don't have that luxury. They have log management practices already in place for other types of platforms or infrastructure, and they need to extend them to support Kubernetes.

How can you do that in an efficient way? Keep reading for tips on integrating Kubernetes logging data into your existing log management workflow without rebuilding from the ground up.

How Kubernetes Logging Is Different

The first step in devising a strategy for supporting Kubernetes through your existing logging workflow is to understand how Kubernetes log data is similar to and different from conventional logs.

Like any other type of modern software platform, Kubernetes creates logs. Specifically, it creates two main types of logs:

1. Logs of stdout and stderr messages for each running container. These can help you monitor applications hosted on Kubernetes.
2. Logs for the main Kubernetes services (like the API server and Kubelet), which are useful for monitoring Kubernetes itself and the nodes that host it.

These logs are not fundamentally different in type or scope from the logs you already manage for other parts of your infrastructure, such as logs for non-containerized applications and conventional operating systems.

That said, the way Kubernetes approaches logs is different in several key respects:

1. Lack of centralization: Kubernetes doesn't attempt to centralize logs for you or even centralize all log data in the same logfile. Monitoring information is spread across multiple files on Kubernetes master and worker nodes.

2. Log access: Kubernetes expects you to access application logs using the kubectl utility. Because of this, unlike on a conventional operating system, you can't use conventional text-manipulation tools (like grep and awk) to interact with Kubernetes log data unless you access the logs from outside the Kubernetes interface.

3. Log rotation: By default, most Kubernetes distributions delete old data in application logs once the size of the log file exceeds 10 megabytes, which is really not a lot if you have lots of applications and therefore lots of log data. Thus, you can't count on Kubernetes itself to keep your log data around as long as you need it; you must devise your own strategy for exporting the logs and rotating them in a way that aligns with your needs.

4. Log structure: Unlike, say, a Linux server, Kubernetes doesn't care about trying to keep your log data formatted or structured in a neat and consistent way, at least when it comes to application logs. It just records whatever your containers dump to stdout or stderr. Whether that data is standardized and easy to work with depends on the way your containers are configured, not the way Kubernetes collects data from them.

Each of these differences introduces challenges for integrating Kubernetes into existing logging workflows.

"The first step in devising a strategy for supporting Kubernetes through your existing logging workflow is to understand how Kubernetes log data is similar to and different from conventional logs."

Managing Kubernetes Logs Efficiently

Fortunately, those challenges can be solved. The following are some tips for managing Kubernetes log data effectively using your existing log management strategy without the need to run a separate log manager just for Kubernetes.

Don't Use Kubectl to Manage Logs

Although kubectl lets you view log data, you shouldn't treat it as your main log management tool. Think of it instead as a quick way to grab recent monitoring data for an individual application, just as you would with a command like head or tail on Linux.

When you need a more insightful and comprehensive way to analyze log data from all of the applications that you have running in Kubernetes, you'll need to connect log data to a third-party analytics and visualization tool.

Don't Settle for Stdout and Stderr

Another reason not to depend on kubectl as the foundation of Kubernetes log analytics is that the log data you can access through kubectl is limited, as noted above, to stdout and stderr. Sometimes, you may run an application that has no stdout and stderr for whatever reason. Maybe it was designed to expose monitoring data in another way, in which case Kubernetes won't capture it. Or maybe the application is not configured to be verbose enough to generate meaningful messages to stdout or stderr.

One common approach that lets you avoid these limitations is to run a logging agent. You can deploy an agent as a node-level system, with a DaemonSet, or as a sidecar container (or containers). The agent collects log data from the application in whichever form the application exposes it. This strategy not only allows you potentially to collect more logging data but also makes it easy to run the same logging agent inside your Kubernetes cluster that you use for the rest of your environment.

Standardize Logs

Because Kubernetes itself doesn't attempt to structure log data in a consistent way, you can end up with a mess if you attempt to analyze logs from a Kubernetes environment alongside logs from other systems without tools that can interpret the Kubernetes logs effectively.

That's why standardizing your Kubernetes logs is so critical. You can do this by exporting log data to a log manager that supports all common logging formats and then querying logs from the log manager. This is much more efficient than attempting to use kubectl to interact with log data that may not be structured consistently.

Simplify Kubernetes Log Agent Deployment

Reading the above, you might be thinking that an enormous amount of manual effort is required to set up a log management solution for Kubernetes that also works for other types of systems. And it is, if you attempt to create logging sidecars yourself to host a logging agent system or run a node-level agent.

It's much easier if you use a solution like Mezmo that offers prebuilt Kubernetes agents that you can deploy with a few simple commands instead of sidecars. This way, you can easily use the same log management tooling and workflow for Kubernetes that you use for other parts of your stack.

Chapter Summary

Kubernetes is complicated enough without having to develop a bespoke log management strategy for it. To help reduce your Kubernetes management overhead, adopt log management tools and strategies that are based on those you already have in place, instead of ones that require you to develop a separate logging operation just for Kubernetes.

CHAPTER 3: ENHANCING THE DEVOPS EXPERIENCE ON KUBERNETES WITH LOGGING

Although setting up an effective log management solution for Kubernetes requires some effort, your investment of labor will be repaid many times through the visibility that you gain—an especially critical asset in the context of Kubernetes, which in some respects introduces an unprecedented level of complexity to software stacks. Log-driven visibility is critical for taming that complexity.

Indeed, keeping track of what's going on in Kubernetes isn't easy. It's an environment where things move quickly, individual containers come and go, and a large number of independent processes involving separate users may all be happening at the same time. Container-based systems are by their nature optimized for rapid, efficient response to a heavy load of requests from multiple users in a highly abstracted environment and not for high-visibility, real-time monitoring.

The Need for Insight

However, the fact remains that you need to know what's happening inside your Kubernetes clusters; the cost of not knowing is simply too high. You need to be able to track bugs, to identify bottlenecks and other performance issues, and to detect security problems—and you may need to do these things in at least close-to-real time. So how can you gain the necessary insight into your Kubernetes system?

Logs Are the Answer—And the Problem

If you search around, you may see a lot of answers to that question, but the truth is that they pretty much all come down to this: Look at the logs. Logs are the number one resource for staying on top of application development and deployment, and that is as true for Kubernetes as it is for any other platform.

But all too often, logs themselves aren't that easy to find, let alone manage or read. Kubernetes container logs last only as long as the pod itself, and every platform, tool, and application has its own logging system. How do you organize and keep track of them all? And, when you find them and read them, how do you pick your way through the contents when each log may have its own syntax and method of organization, and any or all of them may be excessively verbose, or cryptic, or both?

Log Management

To make effective use of the logs that are available, you need to automate the process of finding log data, then gathering and processing that data, presenting it in clearly understandable formats, and turning high-priority items into alerts or other calls to action. You need a comprehensive, powerful tool to handle the complex and demanding tasks required for truly useful log management.

Mezmo

That's where Mezmo comes in. Mezmo does the heavy lifting when it comes to managing logs and log data. It can pull in log data from an extraordinarily wide variety of sources (including Docker, Heroku, CloudFoundry, AWS, and IBM, plus those here-one-millisecond-gone-the-next Kubernetes container logs), as well as parse, manage, and organize the data; display log contents along with metrics in a visual format; archive and export log data; and generate alerts via your favorite alerting system.

Mezmo and DevOps

What can Mezmo do for your DevOps team? We've already seen that Mezmo provides access to a wealth of log data, so this may be a better way to ask the question: What can the information contained in application, infrastructure, and platform logs do for your team at key points in your DevOps delivery chain? Let's take a quick look:

Log Data for Developers

For developers, logging at the code-library level can provide key insights into application behavior, including cumulative effects, patterns of action over

time, and anomalous behavior that may not generate an error. Mezmo includes official integrations with Node.js, Python, Go, and Ruby among others, as well as unofficial integrations with Java, and other major languages and libraries.

Mezmo also makes it easy to monitor the development process using GitHub event integration. You can monitor team activities for individual repositories including push, pull, commit, fork, create, and comment events.

At the deployment end, operating system-specific or platform-specific integrations can provide additional insight into individual apps, as well as allow you to monitor application interactions with other apps and the deployment infrastructure.

QA and Logging

For quality assurance, application, platform, and infrastructure logs can all provide important information. For performance monitoring, for example, log data from both the deployment and host infrastructure can give you valuable insight into time and resource use at the level of individual containers and microservices; it can also help you uncover resource-use conflicts and other potential bottlenecks.

Operations-Level Logging

At the level of operations, you need quick, centralized access to key log data items not only from your deployment and hosting platforms but also from infrastructure-related management and resource services. Real-time and near real-time log monitoring makes it possible to anticipate such things as shifting load requirements over time and changing patterns of user access, as well as to make the necessary adjustments before they become a problem. Mezmo integrates with most major alerting services in addition to providing generic email alerts and webhooks. You can set up alerts based on the number or frequency of specific log events, then use the alert service's features to notify the appropriate team members based on the time, nature, and severity of the alert.

Logs and Security

Security is no longer an option in DevOps—it's a necessity. Log aggregation, monitoring, and analysis can and should play a key role in your security operations. Platform and infrastructure-level service provider logs are typically the best (and often the only) way to track user access; to detect unusual load patterns, anomalous behavior, and suspicious incidents (such as repeated login errors); and to trace the actions of intruders. Your alert-system integration should include any security-related events and enable quick security-team access to all relevant log data.

Kubernetes Logs in Context

No matter who in your organization is using Mezmo, our platform provides the most complete view of log data from every source. With a centralized view of all logs, users are able to view Kubernetes events alongside application logs to gain holistic insights into the health of their application. No need to context switch and hunt down logs from various sources, everything is available from one interface, eliminating the need for any specialized knowledge of Kubernetes.

Combined with features to help make data more actionable, including automatic parsing and real-time alerting, so everyone on your team feels empowered with the log data they need to optimize every stage of the development lifecycle.

Chapter Summary: Logs as a Resource

What's the bottom line? Maybe it's this: Your application, service, and infrastructure logs are a valuable resource, and if you don't make good use of them, you are depriving your operation of important—and in many ways vital—tools. Mezmo makes it easy to harness the full power of your system's logs and put it to work for your DevOps team.

CONCLUSION

You may be tempted to think of Kubernetes log management as a boring, tedious, and challenging task. After all, the complexity of Kubernetes makes it more difficult to implement log collection and analysis than it would be in other types of environments, and mastering the nuances of log centralization in Kubernetes is not most people' s idea of a fun way to spend the weekend.

But the reality is that Kubernetes log management doesn't have to be difficult or complicated. Using a log management solution like Mezmo, you can collect logs from all parts of your Kubernetes cluster, regardless of which Kubernetes distribution you use or whether your clusters run in the cloud, on-premises, or via a hybrid architecture. With Mezmo, three simple kubectl commands allow you to deploy a logging agent that will give you total visibility into your cluster. From there, view Kubernetes events in context with application logs, and easily set up alerts and views so teams understand the health of their applications. Mezmo helps modern DevOps teams truly unlock the power of all of their logs, including Kubernetes.

To learn more about how Mezmo simplifies Kubernetes log management, sign up for our 14-day free trial.

ABOUT MEZMO

Mezmo is a centralized log management solution that gives DevOps teams control of their data and allows them to gain valuable insights from their logs.

Mezmo was brought to life by three-time co-founders Chris Nguyen and Lee Liu and included in the Winter 2015 batch of Y Combinator. In 2018 Mezmo partnered with tech giant, IBM, to become the sole logging provider for IBM Cloud.

This past year, the company was named to the Enterprise Tech 30 list, the Forbes Cloud 100 Rising Stars list, the Top 25 Enterprise Software Startups to Watch in 2020 list, and the CRN 10 Hottest Cloud Startups of 2020 list, and received the 2020 IBM Cloud Embed Excellence Award.

mezmo

Thank You

Sales Contact:	outreach@Mezmo.com
Support Contact:	support@Mezmo.com
Media Inquiries:	press@Mezmo.com