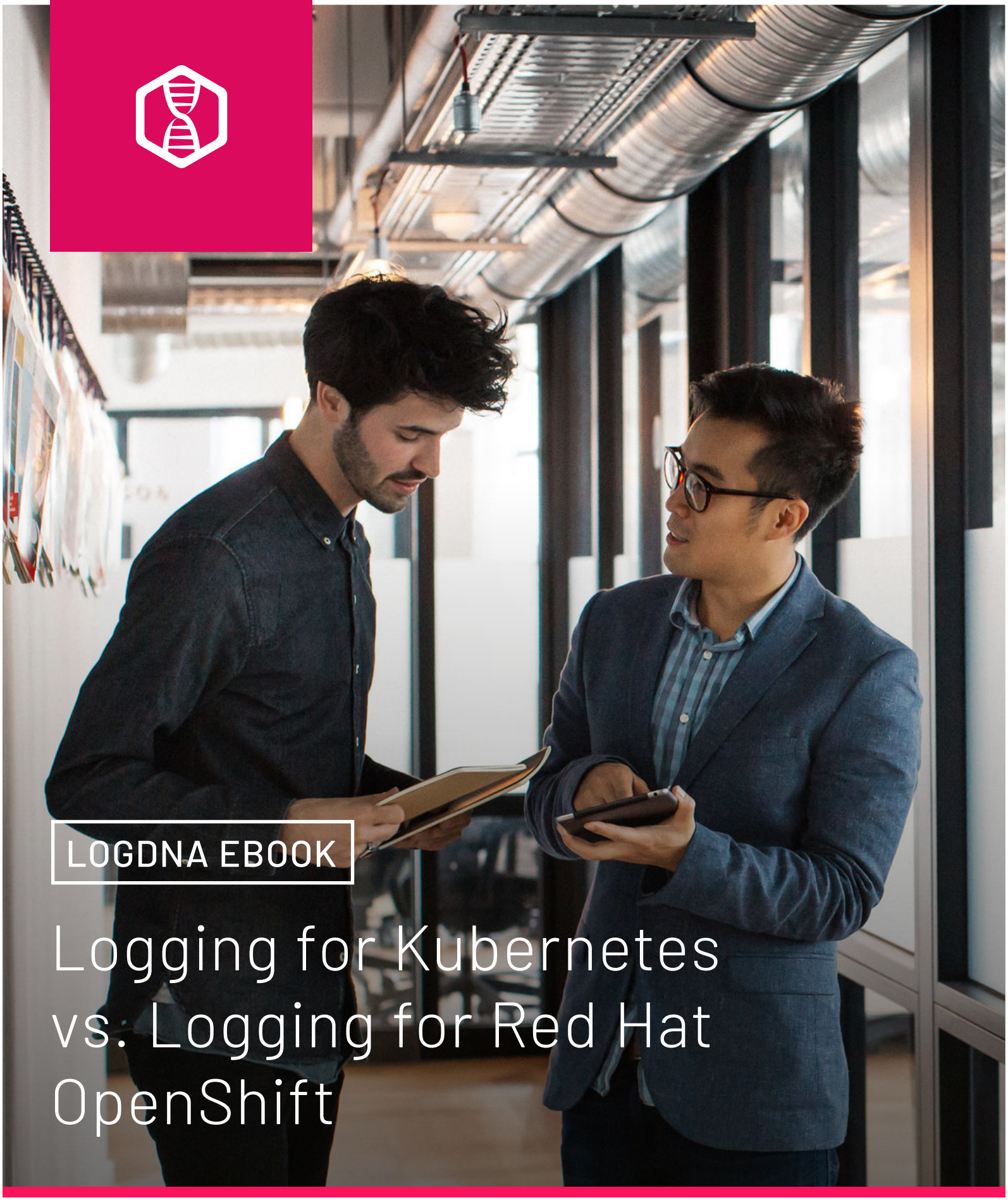


LOGDNA EBOOK

Logging for Kubernetes vs. Logging for Red Hat OpenShift

Kubernetes and OpenShift are broadly similar platforms. In some respects, they are the same platform, given that OpenShift is based largely on Kubernetes. Yet there are notable differences between them, not least when it comes to logging.

logdna

EXECUTIVE SUMMARY

Kubernetes and OpenShift are broadly similar platforms. In some respects, they are the same platform, given that OpenShift is based largely on Kubernetes. Yet there are notable differences between them, not least when it comes to logging.

To provide guidance for teams deploying Kubernetes and OpenShift — or those in the process of deciding which platform is the best fit for their needs — LogDNA has prepared this eBook, which explains how logging works in both Kubernetes and OpenShift. The chapters below discuss how Kubernetes and OpenShift relate to each other, which data each platform makes available to log, how to collect logs on the respective platforms and how their approaches to logging are similar and different.

As the eBook explains, there are a variety of deployment options available for both Kubernetes and OpenShift. The logging tools and methods available to you will vary, in many cases, depending on where you deploy the platforms — on your own infrastructure or using a managed cloud service — as well as which Kubernetes distribution or OpenShift version you use. For that reason, the following chapters don't exhaustively cover every logging strategy available for every type of Kubernetes or OpenShift deployment.

Instead, we'll discuss the main generic logging methods available for standard Kubernetes and OpenShift environments. We'll also contextualize the discussion by explaining how logging works when deploying these technologies on one specific platform — IBM Cloud — which integrates with LogDNA to simplify and centralize log management for both Kubernetes and OpenShift.

TABLE OF CONTENTS

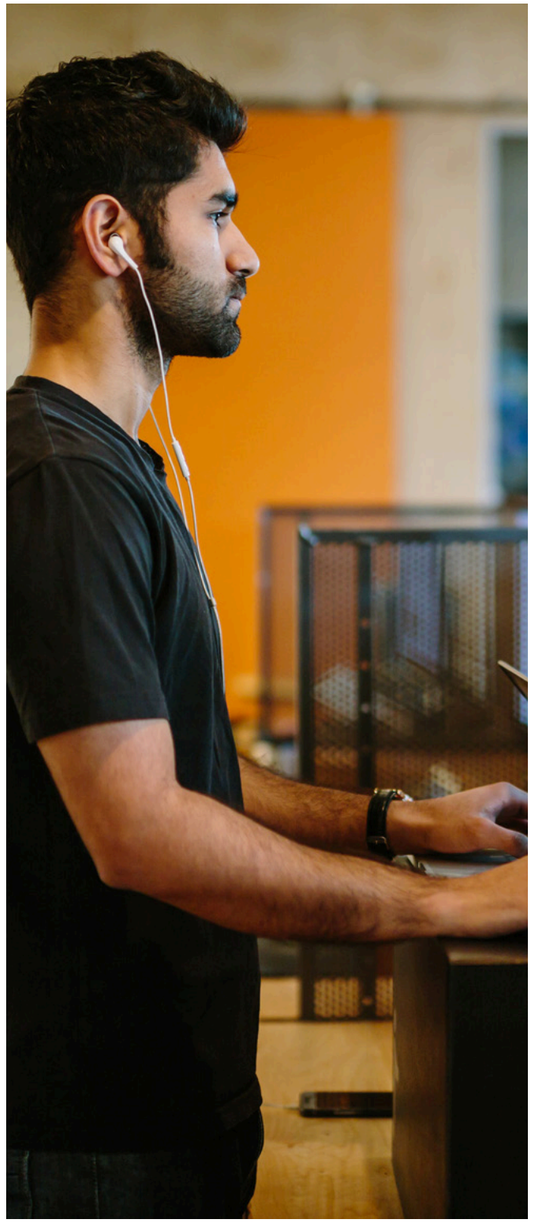
CHAPTER 1: UNDERSTANDING KUBERNETES & OPENSHIFT

To understand how to manage logs for Kubernetes and OpenShift, you must understand what each platform does, and how logs contribute to achieving the intended results that the platforms can deliver.

What is Kubernetes?

Kubernetes is an application management platform that offers several key types of functionality:

- **Application hosting:** Kubernetes's first and foremost feature is hosting applications. Typically, those applications are hosted in containers, although it's possible to run other types of workloads, such as virtual machines, with Kubernetes.
- **Load balancing:** Kubernetes automatically distributes traffic between different application instances to optimize performance and availability.
- **Storage management:** Kubernetes can manage access to storage pools that applications use to store stateful data.
- **Self-healing:** When something goes wrong, such as an application failure, Kubernetes attempts to fix it automatically. (It doesn't always succeed, however, which is one reason why Kubernetes logging is so important.)



Kubernetes offers a variety of other features, too, but these are the core areas of functionality it offers.

As you'll note, logging and monitoring are not among Kubernetes's core features; for more on that, see the following chapter.

What is OpenShift?

OpenShift is also a platform for managing containerized applications running on Red Hat Enterprise Linux servers. (OpenShift does not support other operating systems.) It offers the same core types of functionality as those described above for Kubernetes — which is no surprise because, again, OpenShift is based on Kubernetes.

However, one difference between Kubernetes and OpenShift is that the latter is developed by Red Hat, which offers multiple deployment options:

- **OpenShift Container Platform**, which you deploy on your own infrastructure. This is a commercial platform that you pay for. Red Hat provides support.
- **OKD (formerly known as OpenShift Origin)**, a fully open source and completely free platform. Origin is community-supported.
- **OpenShift Online**, an SaaS implementation of OpenShift that is hosted on infrastructure maintained by Red Hat. This is also a commercial platform and is supported by Red Hat.
- **OpenShift dedicated**, a fully managed offering from Red Hat. This is a commercial offering as well, and offers the most extensive level of support from Red Hat.

Red Hat OpenShift on IBM Cloud

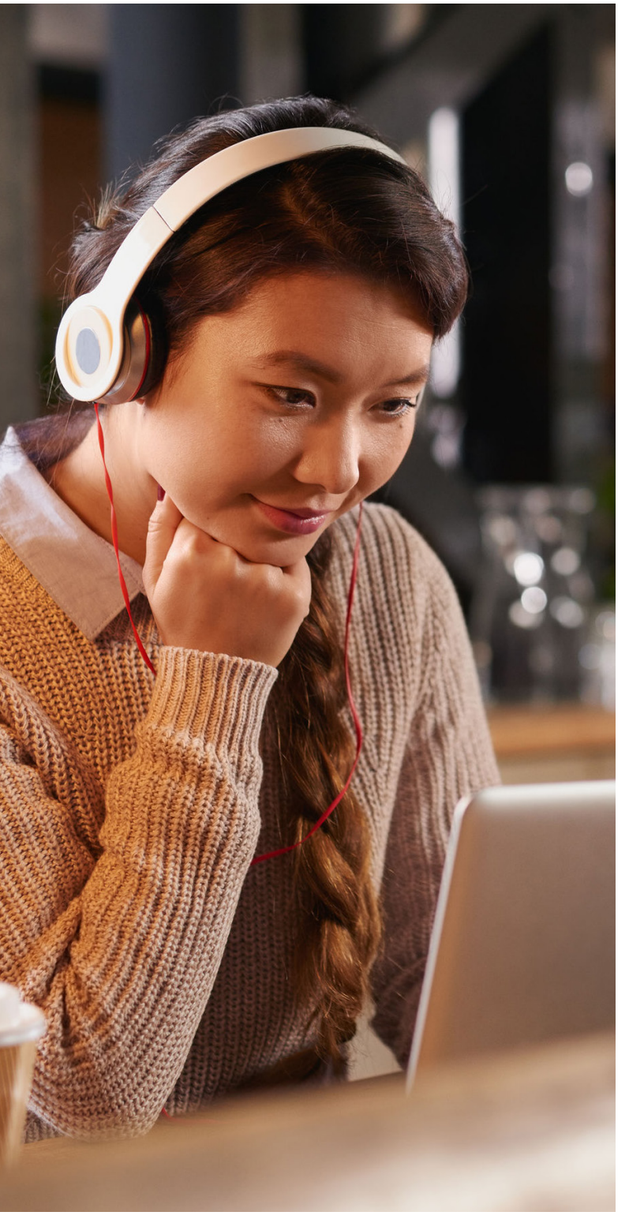
In addition to the preceding iterations of OpenShift, which are available directly from Red Hat, several public cloud vendors offer (in partnership with Red Hat) fully managed OpenShift services hosted on their clouds. This includes OpenShift on IBM Cloud, a service that allows users to spin up OpenShift clusters with just a few clicks on the infrastructure provided and managed by IBM.

OpenShift on IBM Cloud provides access to most of OpenShift's native functionality. As we'll explore later in this article, however, it also offers certain add-on features, including a more user-friendly OpenShift log management solution than OpenShift's built-in log tooling.

OpenShift vs. Kubernetes

OpenShift is a certified Kubernetes distribution that is fully compatible with Kubernetes's native tooling. That means you can use tools like kubectl on OpenShift if you want.

This does not mean, however, that OpenShift is just a Kubernetes distribution. It differs from Kubernetes in key ways. For one, there are OpenShift-specific tools (like oc, which provides many of the same features as kubectl, as well as a built-in container registry) that admins should generally use instead of relying on generic Kubernetes tools. For another, although the technology behind OpenShift is open source, the platform is a commercial product developed by Red Hat. Processes like upgrades and log management (again, keep reading for more on that) also work somewhat differently in OpenShift than in Kubernetes.



CHAPTER 2: KUBERNETES LOGGING ESSENTIALS

The previous chapter discussed what Kubernetes does. You may have noticed that logging and monitoring weren't on the list of core Kubernetes features.

That's not because Kubernetes doesn't offer any kind of logging and monitoring functionality. It does, but it's complicated.

On the one hand, Kubernetes offers some basic functions through kubectl for checking on the status of objects in a cluster, which we'll discuss below. It also creates logs for certain types of data, and it exposes other types of data in ways that make it available for collection through third-party logging tools.

On the other hand, Kubernetes offers no full-fledged, native logging solution. Unlike, say, AWS, which has a built-in logging solution in the form of CloudWatch, or OpenStack, which has its own comprehensive logging solution, stock Kubernetes doesn't have a complete native logging service, or even a preferred third-party logging method. Instead, it expects you to use external tools to collect and interpret log data.

That said, certain Kubernetes distributions do come with built-in logging extensions based on third-party tooling, or at least a preferred logging method that they support. For example, as we'll see below, Kubernetes on IBM Cloud integrates with LogDNA to collect Kubernetes log data and enable real-time analysis and log management using LogDNA.

In most cases, it's possible to use an alternative logging method even on a Kubernetes distribution that has a preferred or natively integrated logging solution. However, the vendor-supported approach is usually simpler to implement.

What to log in Kubernetes

No matter which logging option you choose, there are several log data types that you can collect in Kubernetes.

Application logs

First and foremost are the logs from the applications that run on Kubernetes. The data stored in these logs consists of whichever information your applications output as they run. Typically, this data is written to stdout inside the container where the application runs.

Below, we'll look at how to access this data in the "Viewing application logs" section.

Kubernetes cluster logs

Several of the components that form Kubernetes itself generate their own logs:

- Kube-apiserver.
- Kube-scheduler.
- Etcd.
- Kube-proxy.
- Kubelet.

These logs are usually stored in files under the /var/log directory of the server on which the service runs. For most services, that server is the Kubernetes master node. Kubelet, however, runs on worker nodes.

If you're experiencing a cluster-level problem (as opposed to one that impacts just a certain container or pod), these logs are a good place to look for insight. For example, if your applications are having trouble accessing configuration data, you could look at Etcd logs to see if the problem lies with Etcd. If a worker node is failing to come online as expected, its Kubelet log could provide insights.

Kubernetes events

Kubernetes keeps track of what it calls "events," which can be normal changes to the state of an object in a cluster (such as a container being created or starting) or errors (such as the exhaustion of resources).

Events provide only limited context and visibility. They tell you that something happened, but not much about why it happened. They are still a useful way of getting quick information about the state of various objects within your cluster.

Kubernetes audit logs

Kubernetes can be configured to log requests to the Kube-apiserver. These include requests made by humans (such as requesting a list of running pods) and Kubernetes resources (such as a container requesting access to storage).

Audit logs record who or what issued the request, what the request was for, and the result. If you need to troubleshoot a problem related to an API request, audit logs provide a great deal of visibility. They are also useful for detecting unusual behavior by looking for requests that are out of the ordinary, like repeated failed attempts by a user to access different resources in the cluster, which could signal attempted abuse by someone looking for improperly secured resources. (It could also reflect a problem with your authentication configuration or certificates.)

CHAPTER 3: OPENSHIFT LOGGING ESSENTIALS

Overall, OpenShift is quite similar to Kubernetes in terms of the data available for logging. But, there are some nuanced differences that make it worth walking through different log data categories in OpenShift. The data categories discussed below apply no matter which OpenShift deployment option you choose.

OpenShift events

In OpenShift, an event is one of the dozens of different actions that may occur within your cluster. Some of them are routine occurrences, like the creation of a container, that generally don't require your attention. Others signal undesirable conditions, like a storage volume that failed to mount or an out-of-memory situation, which you may want to investigate further. A full list of OpenShift event types is available here.

The information included in events data is basic. OpenShift tells you that the event occurred, but doesn't provide detail about why it occurred or, if something failed, what the scope of the failure was. OpenShift also doesn't map interrelated events together; it's up to you to figure out how one event relates to another. For these reasons, events provide only limited visibility.

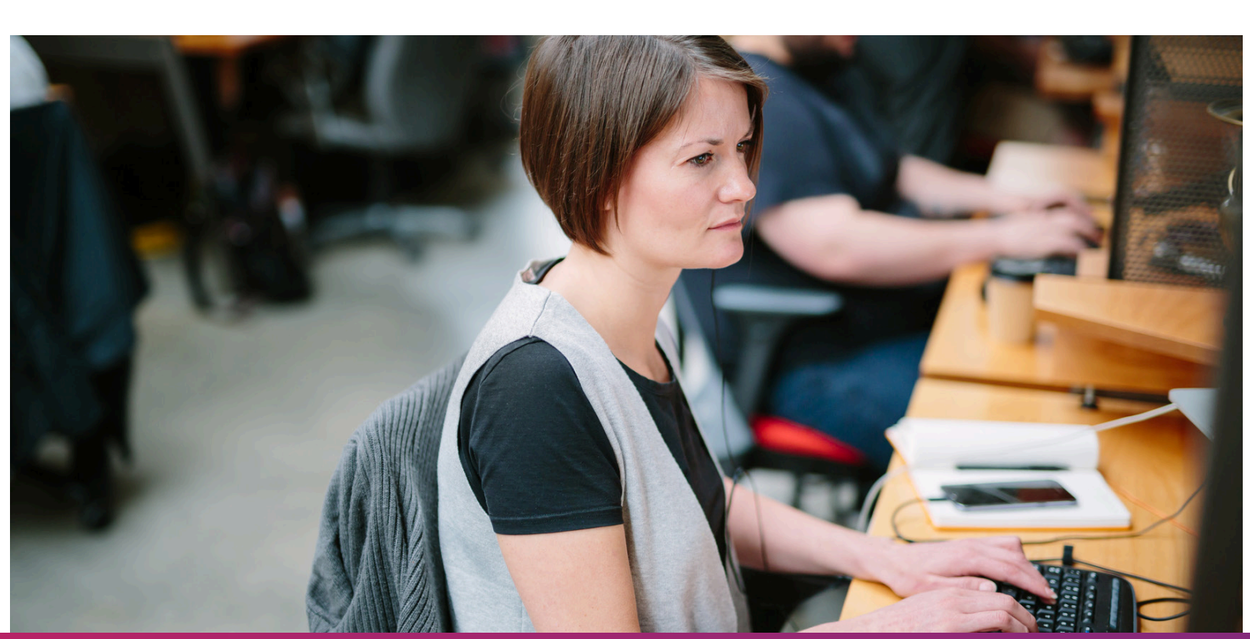
Nonetheless, events offer a quick way of gaining a basic understanding of the state of your cluster and identifying common problems. Historical event data is also useful for researching the root cause of failures.

OpenShift API audit logs

OpenShift provides support for logging API requests issued by users and administrators and other components of the cluster. This data, which is known as API audit logs, provides a deeper level of visibility into actions performed within a cluster than do events. That is because the audit logs provide full context for each request: Where it originated, which namespace it impacted (if relevant), what the response was, and more.

Infrastructure logs

If you're responsible for managing the infrastructure that hosts your OpenShift cluster, keeping track of the underlying servers' health is important. You can do this by looking at the standard Linux log files in the /var/log directory of each node in your OpenShift cluster.



CHAPTER 4: COLLECTING KUBERNETES LOGS

Now, let's get into the meat of this eBook: How to collect log data from Kubernetes and OpenShift (starting with Kubernetes, the focus of this chapter).

The various types of log data described above can be accessed in different ways.

Viewing application logs

There are two main ways to interact with application log data. The first is to run a command like:

```
kubectl logs pod-name
```

Where pod-name is the name of the pod that hosts the application whose logs you want to access.

The kubectl method is useful for a quick look at log data. But if you want to store logs persistently and analyze them systematically, you're better served by using an external logging tool, like LogDNA, to collect and interpret the logs. On IBM Cloud Kubernetes Service, or IKS, you can set up a LogDNA instance to perform this function for application logs (as well as for logs associated with Kubernetes itself) by following a few steps in the IKS Web Console or from the command line. For full instructions, check out the IBM Cloud documentation.

Viewing cluster logs

There are multiple ways of viewing cluster logs. You can simply log into the server that hosts the log you want to view (as noted above, that's the Kubernetes master node server in most cases) and opening the individual log files directly in a text editor, less, cat or whatever command-line tool you prefer. Or, you can use journalctl to retrieve and display logs of a given type for you.

The most user-friendly solution, however, is again to use an external logging tool, like LogDNA. As noted above, IBM Cloud's integration with LogDNA makes it easy to collect Kubernetes cluster logs as well as application logs and analyze them through a centralized interface, without having to worry about the tedious process of collecting individual logs from each of your nodes through the command line.

Viewing events

You can view Kubernetes event data through kubectl with a command like:

```
kubectl get events -n default
```

Where the -n flag specifies the namespace whose events you want to view (default in the example above). The command:

```
kubectl describe my-pod
```

Will show you events data for a specific pod.

Because the context of events data is limited, you may not find it very useful to log all events. However, you can always redirect the CLI output from kubectl into a log file and then analyze it with a log analysis tool.

Viewing audit logs

To a greater extent than is the case for other types of Kubernetes log data, the way you view and manage audit logs varies significantly depending on which Kubernetes distribution you use and which log collector you want to use to collect these logs. There is no generic and straightforward way to collect audit logs directly from kubectl.

On IKS, audit events are routed to a webhook URL. From there, you can collect logging data with IKS's native LogDNA integration by following these instructions.

CHAPTER 5: COLLECTING OPENSHIFT LOGS

Compared to standard Kubernetes, log management in OpenShift is a bit more convenient overall, thanks to the robust support for accessing and interpreting log data that is built into OpenShift's native tools.

Accessing OpenShift event data

Event data can be accessed in several ways. One is to use the OpenShift Web Console by navigating to Browse > Events. This is convenient if you need to check event data quickly, but you can't access it programmatically in this way.

If you want more control over which event data you are looking at, or you want to pass it to external tools, you can use the CLI utility oc with a command such as:

```
oc get events
```

You can pass a few parameters to control output, such as specifying a certain namespace using the -n flag. However, the extent to which you can focus on specific events using oc is limited, so it may be necessary to pipe the oc output into external tools, like grep, to home in on events related to a certain node, events of a certain type and so on.

A third approach is to use an external log management tool to read and analyze event data. For example, if you use OpenShift on IBM Cloud, you can take advantage of the native LogDNA integration to set up log analysis in a few simple steps. LogDNA enables real-time access to log data, alerting based

on log contents and the ability to store log data as long as you want. The latter feature may be particularly important because OpenShift itself deletes log data permanently if you delete a namespace. But if you import log data into LogDNA, you can store it for as long as you need, even if the original data source disappears.

To use this option, you must first create a LogDNA service instance within your OpenShift cluster. You can do this graphically through the IBM Cloud UI by following a simple set of configuration steps, or from the CLI with a command like:

```
ibmcloud resource service-
instance-create NAME logdna
SERVICE_PLAN_NAME LOCATION
```

For full documentation of the command, refer to the IBM Cloud documentation.

You will then need to deploy LogDNA agents to connect to your OpenShift cluster and forward logs to IBM Log Analysis.

LogDNA agents can be deployed in OpenShift using the oc utility. To do this, first create a new namespace in your cluster to host the agents with a command such as (here, we'll create a namespace called ibm-observe):

```
oc adm new-project --node-selector='' 
ibm-observe
```

Next, create a service account for the LogDNA agent:

```
oc create serviceaccount logdna-agent -n
ibm-observe
```

Next, configure privileges so that the logdna-agent service account can create privileged LogDNA pods:

```
oc adm policy add-scc-to-user privileged \
system:serviceaccount:ibm-
observe:logdna-agent
```

Next, add a secret for the ingestion key that the LogDNA agent will use to send logs:

```
oc create secret generic logdna-agent-key
--from-literal=logdna-agent-
key=INGESTION_KEY -n PROJECT
```

Finally, deploy LogDNA agents to nodes using kubectl. Preconfigured YAML files are available for different public IBM cloud endpoints, so you can use a command such as the following to deploy an agent:

```
kubectl create -f https://assets.eu-de.
logging.cloud.ibm.com/clients/logdna-
agent-ds-os.yaml -n ibm-observe
```

For a full list of available public endpoints, as well as additional context on deploying LogDNA agents on IBM cloud, refer to this documentation.

Once fully configured, the LogDNA service instance and agents provide a LogDNA dashboard within your OpenShift console. There, you have full access to LogDNA's feature set for viewing and managing OpenShift events and other log data.

Viewing OpenShift API audit logs

To view API audit logs in a generic OpenShift installation, you must first enable audit logging by adding an auditConfig stanza to the /etc/origin/master/master-config.yaml on your master node, such as the following:

```
auditConfig:
 auditFilePath: /var/lib/origin/log/ocp-
audit.log
 enabled: true
 maximumFileRetentionDays: 5
 maximumFileSizeMegabytes: 20
 maximumRetainedFiles: 20
```

The enabled: true parameter turns audit logging on; other fields define log settings.

Once audit logs are enabled, audit log data will be managed via systemd and can be accessed by running journalctl on the node where log data is stored. Typically, that is your OpenShift master node. You could also, if you wish, access the audit log file directly with a command like:

```
less /var/lib/origin/log/ocp-audit.log
```

In OpenShift version 4.0 and later, you can also access audit logs using the oc utility:

```
oc adm node-logs <node-name>
--path=openshift-apiserver/<log-name>
```

If you use OpenShift on IBM Cloud, OpenShift's native audit logging utilities are replaced by IBM Log Analysis with LogDNA. Using the same LogDNA integration process described in the preceding section of this article, you can configure LogDNA to collect and analyze audit log data.

This approach offers several advantages over relying on native OpenShift tooling. LogDNA provides advanced log analysis features and a graphical interface that makes it more convenient to access audit log data. It also aggregates audit logs alongside other types of log data, providing admins with the ability to review all logs from a central location. It makes it possible to store log data for as long as needed, even if it disappears from OpenShift itself.

CONCLUSION

Kubernetes Logging vs. OpenShift Logging

At a high level, there are no fundamental differences between logging in Kubernetes and logging in OpenShift. Both platforms make the same types of log data available, and both support a range of methods for collecting and interacting with it.

Yet there are more nuanced differences when it comes to logging implementation. Kubernetes and OpenShift use different tools -- kubectl and oc, respectively -- for making log data accessible from the CLI. The infrastructure logs associated with Kubernetes nodes may vary more widely than those for OpenShift nodes. The latter platform runs only on Red Hat Enterprise Linux servers, and is, therefore, more uniform. Likewise, there is more uniformity in the way logging is configured by default in OpenShift, a single platform maintained by a single vendor, than in stock Kubernetes, which is available by many vendors often change default settings or prefer certain tools over another.

Whether you wish to use OpenShift or any vanilla Kubernetes, you can simplify your logging strategy by relying on LogDNA to collect logs for you. LogDNA can collect multiple types of Kubernetes and OpenShift logs -- including application logs, infrastructure logs, and audit logs -- and make them available in a central location for analysis. It also eliminates the need to work with clunky CLI tools, or master the differences between kubectl and oc, to interact with log data; instead, LogDNA provides a rich graphical interface for managing logs.

In addition, deploying the LogDNA agent is very simple, especially on platforms like IBM Cloud, which offers native integration. With just a few commands or clicks of the mouse, you can deploy LogDNA in IBM Cloud to manage all of your Kubernetes and OpenShift logs.

In short, logging for Kubernetes and OpenShift can be complicated -- especially if you rely on simple methods, such as viewing logs on the command line, that were never designed to be scalable or to allow you to compare data across logs. But logging on these platforms doesn't need to be hard. By adopting a comprehensive log management tool like LogDNA, you can meet all of your logging needs for Kubernetes and OpenShift -- not to mention virtually any other modern platform -- in a scalable, simple way.

ABOUT LOGDNA

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

LogDNA 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 LogDNA 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.

logdna

Thank You

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