loggly

e book series

The Field Guide for Log Management

We all know log files.

We all use log data. At a minimum, every admin and developer knows how to fire up `tail -f` and use an arsenal of command line tools to dig into a system's log files. But the days where those practices would suffice for operational troubleshooting are long gone. Today, you need a solid log strategy.

Log data has become big data, and it plays a bigger role in your success than ever before. Not being able to manage it and make meaningful use of it can, in the worst case, kill your business.

You might have implemented good application and performance monitoring, but that only tells you that something is happening, not why. The information needed to understand the why, and the ability to predict and prevent it, is in your log data. And log data has exploded in volume and complexity.

In this eBook, we'll discuss strategies for dealing with your log data in the cloud era.

Log Data Is No Longer the SysAdmin's Problem

Log data volume has exploded. Why? The same reason that innovation has exploded in the tech industry over recent years: the commoditization of cloud technology.

Cloud services like Amazon's AWS, Microsoft's Azure, or Rackspace have made it affordable even for small- and medium-sized businesses to run complex applications on elastic virtual server farms. Containers running microservices are the next step in this move toward distributed and modularized systems.

The downside is that complexity is multiplied in those environments. Running tens or hundreds of machines with many different application components increases the risk that one of them will start malfunctioning. And finding the problem is far more difficult than SSHing into a single server.

Log Data Is Scattered

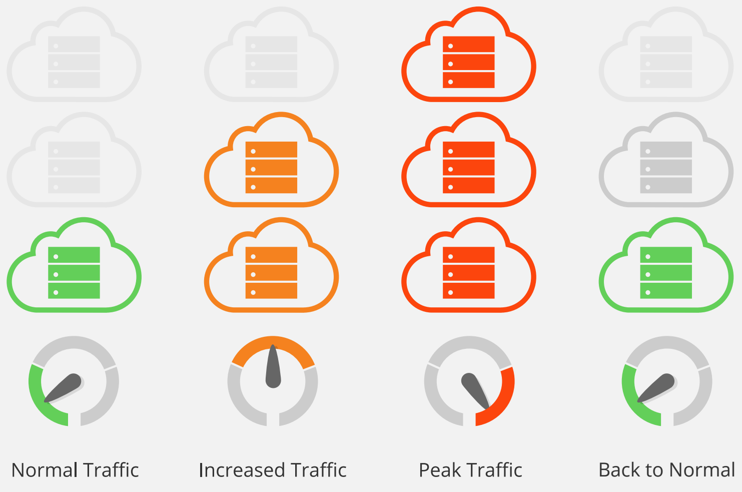
To allow for troubleshooting, each of these many components typically (and hopefully) writes log data. Not only do you have to deal with a staggering number of large log files, but they're also scattered all over your network(s).

"As soon as we exit the boundary of the Microservice we enter a wild ocean of non-determinism—the world of distributed systems—where systems fail in the most spectacular and intricate ways, where information gets lost, reordered, garbled, and where failure detection is a guessing game."

Jonas Bonér, Founder & CTO, Lightbend

Log Data Is Sometimes Ephemeral

To make things a bit more interesting, some components, like VMs or containers, are ephemeral. They're launched on demand, and take their log data with them once they're terminated. Maybe the root cause slowing down or crashing your web store was visible in exactly one of those lost log files.



In an elastic cloud, compute resources are automatically added and terminated based on needs. Unless precaution is taken, this means that log data from terminated instances can get lost.

Log Data Contains a Variety of Technologies

If that's still not complex enough, add in that people mix different technologies—for example, hybrid clouds that keep some systems on-premise or in a colo. You might run containers inside of VMs or use a container to deploy a hypervisor. Or you also might need to collect data from mobile applications and IoT devices.

Log data is the single common denominator across diverse application environments.

The New Norm for Log Data: Centralized Log Management

A log management solution aggregates all of your log data, indexes it, parses it, generates real-time metrics, and provides unique tools to navigate and analyze your logs to make problem solving much more productive. **It leaves no log behind—including the ones from ephemeral systems.**

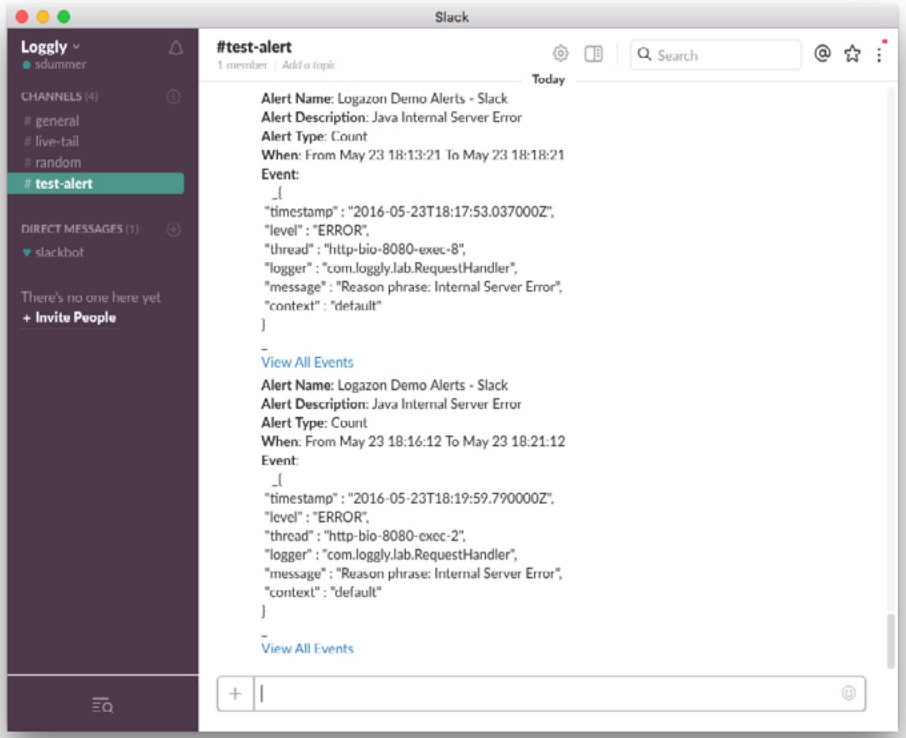
Some log management solutions require installing agents to accomplish this, while others are agentless and use de-facto standards like syslog, which are an integral part of many systems that allow sending logs over the network. If your solution uses agents, it's vital to make sure they are available for all operating systems, devices, and other components. You'll also need a strategy to keep the agents updated and patched.

Log Management Is an Integral Part of DevOps

Because log data is the universal element across every component and every layer, centralizing all your log data is integral to:

- **Troubleshooting problems faster** and bringing in the right people as early as possible.
- **Monitoring for issues at all levels in the stack.** There are a ton of monitoring tools out there, but many of them actually create silos of users instead of breaking those silos down. Having your teams looking at (and learning) multiple tools and trying to compare their output will not reduce, but increase MTTR.
- **Continuous code deployment with confidence.** Log management should be part of your CI test cycle. The more cohesive your test scenarios are, the more components you need to log.

Making the fruits of this analysis accessible to everyone who's working on a problem–for example, by sharing relevant searches or metrics through HipChat or Slack workspaces where developers and ops people collaborate–goes a long way towards fostering a DevOps mindset. But the value of log data doesn't end when a burning operational issue is solved. Log data also supports a pro-active approach to preventing problems, which is also a hallmark of a successful DevOps culture.



```
Alert Name: Logazon Demo Alerts - Slack
Alert Description: Java Internal Server Error
Alert Type: Count
When: From May 23 18:13:21 To May 23 18:18:21
Event:
  _{
 "timestamp" : "2016-05-23T18:17:53.037000Z",
 "level" : "ERROR",
 "thread" : "http-bio-8080-exec-8",
 "logger" : "com.loggly.lab.RequestHandler",
 "message" : "Reason phrase: Internal Server Error",
 "context" : "default"
}
_
View All Events
Alert Name: Logazon Demo Alerts - Slack
Alert Description: Java Internal Server Error
Alert Type: Count
When: From May 23 18:16:12 To May 23 18:21:12
Event:
  _{
 "timestamp" : "2016-05-23T18:19:59.790000Z",
 "level" : "ERROR",
 "thread" : "http-bio-8080-exec-2",
 "logger" : "com.loggly.lab.RequestHandler",
 "message" : "Reason phrase: Internal Server Error",
 "context" : "default"
}
_
View All Events
```

Log data gets everyone on the same page when it comes to troubleshooting operational issues.

You Need Log Management If...

- Your application runs on more than a couple of servers.
- You run in a dynamically scaling environment where servers spin up and down, taking your logs with them.
- You release code more frequently than once a quarter.
- Your application processes transactions that are mission critical for your customers or users.
- The availability of your application depends on third-party services.
- You struggle to isolate problems with grep/awk/sed or other local text search utilities.

Don't Panic. Log Management Is Here to Help.

Here are some use cases where log management has driven value and made life easier for Loggly customers.

Root Cause Analysis

"Why did our application go down? Why are transactions failing? Why is performance slow?"

The answers to these questions aren't always readily available. Any of the issues could be caused by system crashes, bugs in your code, timeout of a third-party service, network problems, or more.

The Log Management Solution

Look at aggregated log metrics to see if any problems "jump out." For example, you might spot a spike in error rates or a spike in traffic from a few IP addresses.

1. If you have a customer ID or other unique identifier, use it to trace the transaction through your stack.
2. When you have some ideas of what to look for, you can search your logs to pinpoint the root cause.

Learn more >>

Loggly customer xsolla reports that it saves three person-hours per day because of Loggly.

Error and Exception Reporting

"How can I see the top errors or exceptions affecting my service? How will I know when there is a big spike in errors?"

Errors and exceptions do happen in the normal course of business. But if you're seeing more than usual, something is probably wrong with your application. You'll want to know about spikes immediately so you can start taking action.

The Log Management Solution

1. Search all of your logs and filter the data for errors and exceptions. For example, you could filter on 4xx and 5xx status codes from your web server.
2. Create saved searches that aggregate errors for the key components of your stack. Track these searches by building a custom dashboard.
3. Monitor your error and exception dashboards every time you push new code.
4. Create alerts to let you know of major spikes in errors. Your alerting threshold may vary based on the "normal" number of errors and the impact that the error has on user success. For example, an error in the checkout process is critical to e-commerce sites.

Learn more >>

Segment integrates with more than 100 web-based services and billions of API calls every month. If the DevOps team sees an uptick in errors relating to a specific partner, it may reach out to that partner and/or buffer messages until the issue is resolved.

Performance Monitoring

"How is my technology stack performing? Is my site any slower than expected? Are transactions completing successfully?"

With the right logging, you have all the data you need to monitor performance and pinpoint problems at all levels in your stack. Of course, you need to act quickly to resolve problems in order to keep your users happy and your business KPIs on track.

The Log Management Solution

1. Centralize data that characterizes the performance of your application, such as web server logs, database slow query logs, application-specific performance metrics logged as JSON, response times for third-party services, and more.

2. Visualize performance by charting this data with a solution like Loggly.

3. Determine parameters for unacceptable performance in key components of your application and set alerts to trigger when these parameters are exceeded.

Learn more >>

"If we didn't have Loggly, there would be days where our revenue would be affected by as much as 70 percent."

***Albert Ho**, Executive Producer and Product Manager of Platform at Rumble Entertainment*

Transaction and Request Tracing

"Why did a high-value transaction fail? Which step of the transaction had a problem?"

If your logs include a unique user identifier (such as a Globally Unique Identifier (GUID), a unique number, an API key, or even a session ID), you'll be able to view the progress of a user through your entire application. This type of analysis can be very useful in spotting elusive technical problems.

The Log Management Solution

1. Incorporate a unique user identifier into all of your log events.
2. Search for the unique identifier to trace logs across your stack.
3. Follow your user through the transaction process.

Learn more >>

Numerous studies show that operational issues cause customers to take their business elsewhere very quickly. Minimizing the impact of these issues on your customers protects your revenue while helping you earn customer loyalty and repeat business.

Trend Analysis and Planning

"What are our biggest opportunities to improve the success of our application? How is our capacity usage changing as our business scales? Are we getting the most out of our investments in AWS? How should we prioritize our development efforts?"

Taken as a whole, log data can provide you with a bird's eye view of what's happening with your application now, and these insights can inform you on where to go next. Loggly customers are using log management to optimize their application's performance, prioritize development projects, and understand future capacity needs.

Case in Point: Monitoring for E-Commerce Sites

There are lots of components and potential events to watch out for when you're developing or hosting an e-commerce website, all with direct impact on revenue. Where do you start monitoring?

1. **Checkout:** Log every step in the checkout process for errors and set alerts to detect failures.
2. **Shopping cart:** Log all add-to-cart failures.
3. **Online catalog/product page:** Look for issues with specific product lines, markets, or other logical groups of products, especially with legacy software integrations.
4. **Email signup:** Look for both client-side and server-side issues.
5. **Login and registration:** In addition to form submission and validation, focus on authentication and authorization logic as a whole. Log social media login errors, authentication and authorization cookies that may be out of sync, and errors from additional authentication checks.

The Log Management Solution

1. Build a metrics API for every single component in your technology stack. This API provides the stats for the component in your service. For example, it could provide input bytes received per second (bps), processed bytes per second, transactions per second, or a list of the top 10, 20, or 50 customers based on received bps, etc. Of course, these are just examples: The metrics that your API delivers should reflect your application's functionality.

Learn more >>

2. Log these metrics and send the data to a centralized log management system.
3. Analyze historical behavior to understand how your application is scaling and what changes will have the most impact on your application.

See how >>

Sendhub has Loggly dashboards that show aggregated metrics such as average response times for its most important endpoints or threads. It uses the data to drive its service level and development project goals for the upcoming quarter.

See an example: Benchmarking Elasticsearch >>

Tracking Unusual Activity

"Is this pattern normal?"

Logs are where login attempts and other system activities are being recorded, so they are where suspicious events can be tracked. Proactive, automated detection of unusual activity is a must-do. You might not be able to know every potential attack pattern in advance, so this is not an easy task. But if you don't analyze your logs to see what's going on, you'll never be able to detect suspicious activity. It's also important to try to anticipate what normal activity on your system might be used to fly under the radar. For example, multiple failed login attempts by a user might be considered normal, but hundreds or thousands of failed login attempts might point to a brute force or dictionary attack.

The Log Management Solution

Centralized log data helps you detect unusual activity on your sites. For example, you could detect excessive traffic from a single IP address, bot activity, unusual logins to your AWS services, or behavior patterns that users wouldn't exhibit in real life. Log management solution features like alerting and anomaly detection take the manual effort out of these types of analyses. Loggly Anomaly Detection, for example, uses machine learning to determine what "normal" values and frequencies in your log data are and can alert you of any deviation.

What Logs Should You Send?

If you're embracing the wonderful world of DevOps, you know that one of the most important principles is getting everyone in development and operations on the same page. To get on the same page, everyone needs a holistic view of what's happening across every component in the stack. So it pays to log at multiple levels.

Yet, many customers tell us that their application logs are the only ones that really matter, and those are the only ones they collect and centralize in a log management solution like Loggly. At the end of the day, the application is what powers their business: It's what their customers see, and a quick look at their support statistics shows that the majority of problems occur within application code.

The question is, though: **Where does your application start and where does it end?** Is it, say, the Java code written by your developers? Most people will likely say that the Java runtime environment is also a crucial part, and typically we see people collect those logs as well.

The web server? Yes. Databases used by the application? Maybe. Here we already see users who don't include the database logs into what they send to Loggly. Operating system, hypervisor, components of the cloud infrastructure? Storage backend? That's where it starts to get spotty. Load balancers, switches, routers? A lot of people seem to assume that

Components of the Application Stack

- Client devices
- Network
- Content delivery networks (CDN)
- Load balancers
- Web server
- Application server
- Databases
- Operating system
- Hypervisor
- Cloud infrastructure

Monolithic Architecture
One system, few components to log

Cloud-based Architecture
Distributed systems, SOA many components that log

the more low-level it gets, the less need they have to centrally collect and manage log data.

Reasons given for abandoning such log sources:

- These components are perceived to be (for the most part) reliable.
- They typically come with their own log monitoring solution.

Thus, sending their log data to a central log management solution like Loggly is not considered necessary. Some users even see them as adding "noise." They monitor their application logs in Loggly, and if they need to look at the logs of their router or AWS Linux instances, they go to the router's web front end or check in AWS CloudWatch.

Instead of making all log data easily accessible to every member of a DevOps organization, there might be a network team that is the only one that has access to (and is familiar with) the proprietary monitoring solution of the network switches or load balancers. The storage team knows how to access the monitoring tool of the RAID systems. All logs in one place, easy to access by all groups and individuals so that all eye-balls can look at a problem if needed? Not really.

The biggest problem with this approach is that **as long as you maintain these "log silos," you will not be able to get a cohesive, in-context view of everything that makes your application run**. And the assumption that these more low-level components "usually work" can be fatal to your business. What happens if you get hit by a rare corner case defect or a well-disguised malware or hacker attack? Chances for that might be (or seem) low, but IF they hit you, chances that they do significant damage are exceptionally high. It's a bit like that fire insurance that you buy even though you never had a fire in your home.

Aren't You Promoting Log Overkill?

No. Consolidating and centrally managing all your logs is different from logging each and every event. The question of what events to record and how much you log is an entirely different problem, which needs consideration as to what the right level of information is or if intensive logging will have performance implications. These questions typically need to be answered for every component of your system. For many (if not most) components, you will probably stick with their defaults.

In a nutshell, *you don't need to log everything, but everything that you do log should be centrally collected and managed*.

Thinking about what to log?

Read the Pragmatic Logging Guide >>

Log Management Must-Haves

Here are the major elements of a good log strategy. This list will serve you well as a checklist of what to look for in a log management solution and what to think about as you build log management into your DevOps processes.

Collect, Aggregate, and Retain

It's crucial to think about your data retention needs and the costs associated with storing them. How long do you need to keep the logs? Do you need them just for troubleshooting, or also for business intelligence type of analysis? Are there regulatory or audit requirements that require you to keep the logs for a certain period of time?

Your daily log volume might already be large, but keep in mind that it doesn't take much to multiply the volume temporarily. A component failure and the resulting log messages in a complex system could easily quadruple the amount of log messages that your application generates. An external event could have the same effect. For example, if you run an online store, Black Friday might balloon your sales as well as your log volumes. If your log aggregation doesn't scale, you could lose your main troubleshooting foundation when you need it most.

Handle Log Diversity

Log files come in a variety of formats, with some following standards and conventions and others using completely custom formats. Your log solution should be able to parse and present the data in a compre-hensive form in near real time, and it should allow to define custom parsing rules. A desirable feature is the ability to add metadata.

Reveal What Matters

Just having a search tool is not enough. To make sense of your log data and the correlation between different data points, you need real-time indexing, parsing, and grouping, along with powerful analytics, customizable dashboards, and data visualization. Your log analytics solution should provide a "treasure map" to the contents of your logs, not just a "metal detector" that requires you to scan indiscriminately (and get lucky) in order to spot a problem.

Detect Anomalies

Given the volume and complexity of log data, you can't rely on searching for problems. Too often, things you never anticipated happening are the type of problems that hurt the most. A good log analytics solution should be able to learn what is "normal" in your log data, and automatically identify and highlight any deviations from norms.

Make Your Own Apps Log

If you write your own code, your log management solution must be able to parse and analyze it. Consider using a well-established data format like JSON (our recommendation) or XML. Whatever you choose, make sure it's plain text format (not binary), human-readable, and easy to parse. Your log solution should be able to easily receive the logs from your application and allow you to set up custom parsing rules if needed.

Be Alert(ed)

Just like every good monitoring application, every good log management solution should be able to send you and your teams alerts based on defined events like error messages. It should be possible to send these alerts through common third-party collaboration tools.

Don’t Break the Bank

Cloud technologies made running distributed systems and elastic compute farms affordable for SMBs. The bill for the troubleshooting tools should be affordable, too. There are fully cloud-based SaaS solutions out there, as well as on-premise products and hybrids, which typically come at higher costs (including those for hardware and datacenter footprint).

Key criteria to decide if SaaS or on-premise solutions are right for you are the sensitivity and volume of your data. Security or privacy concerns or regulatory requirements may keep you from transferring data across public networks. Similarly, the sheer data volume could make this impossible or too expensive.

Get the Log Management Buyer’s Guide >>

You may be trying to decide whether to build your own solution or deploy a cloud-based service like Loggly. You’re not alone; our survey data tells us that one-third of our customers started trying to build their own log management solutions before deploying Loggly. There’s no single right answer to the build vs. buy question, but we think there are five factors that you should consider.

Read our perspective >>

Summary

In the world of distributed systems, complex microservices, and elastic cloud capacity, log management is more important than ever. Your log data is the single common thread that reveals how your users interact with your application and how your application is performing. Good intelligence can mean the difference between successful innovation and a world of pain.

We hope that you have walked away from this eBook with some new ideas about how to solve problems and how to prevent these problems from occurring in the first place. But there's no better way to see the value of log management than to start applying it to your own business. We encourage you to take a free test drive of Loggly today.

Start Free Trial >>