OverOps

Enrich the Value of Splunk with Code-Aware Insight Into The Quality of Your Applications and Services with OverOps

Splunk and your log files

Most companies and organizations know of, and probably use, Splunk. It helps them understand where things go wrong in applications, how to optimize the customer experience and to identify the fingerprints of fraud. Splunk does this by investigating machine data—the digital exhaust created by the systems, technologies and infrastructure powering modern businesses.

Log files help identify when or where something has gone wrong, but in order to troubleshoot an issue, there is still a lot of work to get to the "why". There are four main challenges with log files in this context:

1. *Log statements are manual and shallow*
   Developers manually choose where log statements are used and decide what information will be sent to the log file. This information is typically shallow, providing only a few variables that are often just representations of values or shorthand of what needs to be communicated.

2. *Searching through log files and classification of entries is tricky*
   Log files lack structure, and in order to sift through them you need to use grep, grok, regex, etc. or other more advanced tools. Regardless of how you process them, classification and de-duplication of log file entries is more than a simple challenge. Further, it is difficult to correlate a log statement with a specific version of code.

3. *Log files provide limited visibility*
   Related to the manual nature of a log file, they are not comprehensive across every error and exception. You only get visibility into what you choose to log. Log files give you no visibility into uncaught or swallowed exceptions.

4. *Tracking down and tracing errors in microservices is impossible*
   The transition to microservices introduces further complexity. We are now facing an enormous amount of application exhaust across multiple, disconnected distributed services that need to be aggregated and then searched through to gain insight into where our systems are failing. Some frameworks have been introduced to trace events across services, but they are good for performance tracking only.

Log files have been with us for ages and are of huge value, but is there another way to collect deeper, more complete information?

Another way: Code-aware machine data with OverOps

At OverOps, we deliver a whole new approach for gaining insight into the quality of our apps and services. Using both static and dynamic code analysis, we capture complete, code-aware insight into every known and unknown error and exception at the moment they occur.

With a log file you might determine which function failed and capture a few variables, but OverOps provides:

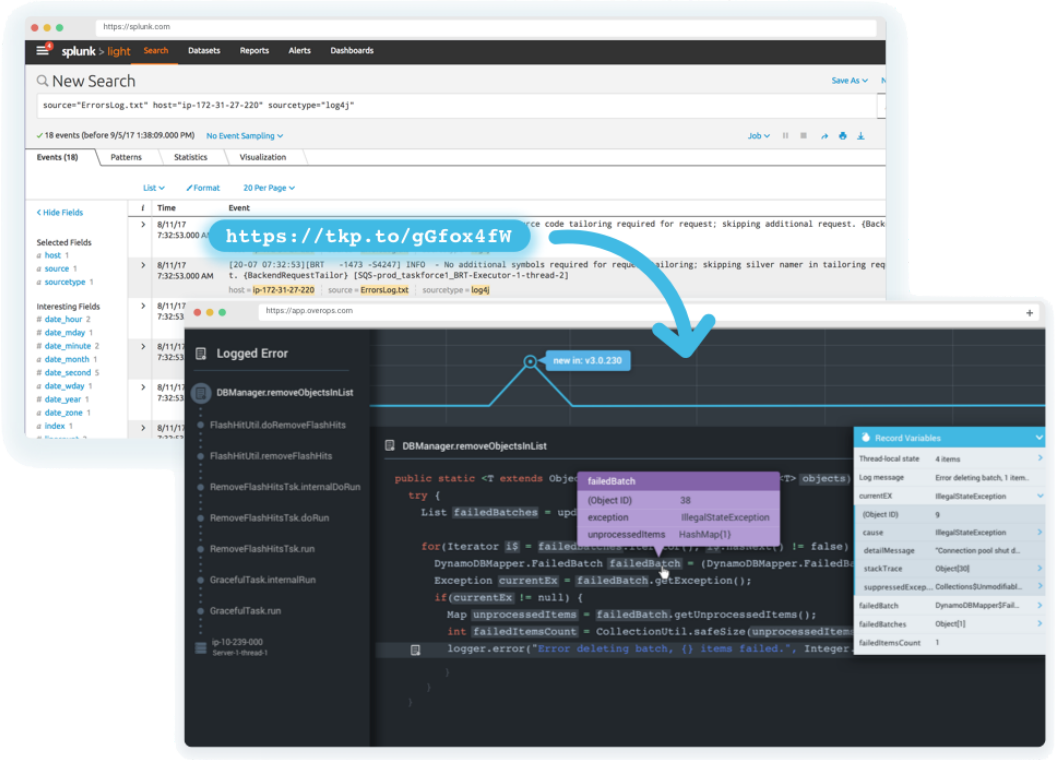
- the value of every single variable across the entire execution stack,
- the complete state of the JVM, including heap and garbage collection,
- the value of every environment variable
- the last 250 log statements (all environments, including debug/trace in prod)
- the frequency and failure rate of an error
- the classification of new and reintroduced
- the release or build number associated with the event
- the frequency and failure rate for each error
- and, with integration into your code repository, we can even map an error back to a developer or team.

Enriching Splunk with machine data from OverOps

There are three ways in which OverOps integrates directly with Splunk. We can help developers troubleshoot more quickly, give metrics and insight to DevOps to gauge overall quality of software and fuel some new AIOps initiatives. Let's explore each.

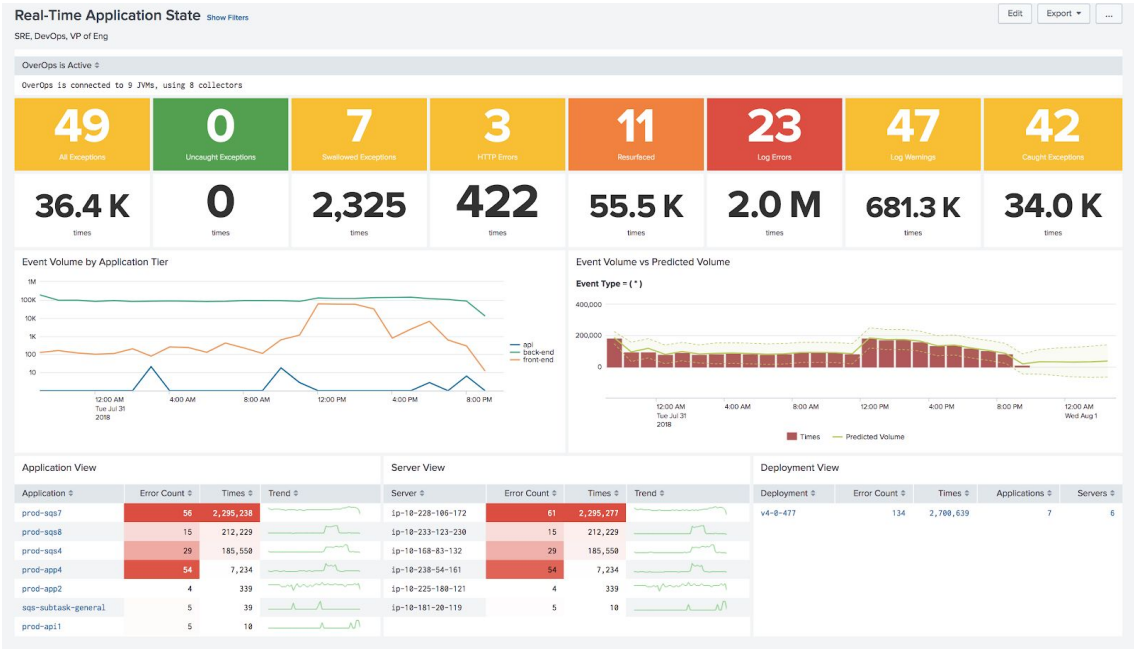
OverOps inserts links into your current Splunk log files

With the insertion of the OverOps links into a log file, you can find where the error occurred using Splunk and then easily link to OverOps to access complete insight into what happened. A Splunk user can now see these links and connect directly to the powerful OverOps ARC UI to troubleshoot issues with this complete information. This can both help developer productivity and have a huge impact on the QA to dev conversation as your team is armed with exact information about every event.



OverOps data within your Splunk Metrics Dashboard

OverOps also exposes all the data we collect through direct integration into our platform. We publish an API and have an option to deliver the information via StatsD. With this integration you can profile the data collected through the Splunk dashboard and see in a glance the applications, deployments, servers, and methods that were affected by different issues. It provides granular information about application level events and errors classified by "New Today", "Resurfaced", "Network", "Database", and more. You can analyze and monitor your application's health, which includes the number of new errors that were detected, the number of errors that resurfaced, and other information that is beneficial for your team and product.



IT Service Intelligence (ITSI) integration and how great data makes AIOps a differentiator

A key emerging capability from Splunk is the IT Service Intelligence (ITSI) product which brings artificial intelligence to events so you can get visibility across IT and business services. It enables you to use AI to go from reactive to predictive IT. OverOps understands the importance of data in this new effort and our granular set of contextual insights can provide an incredibly useful baseline for deep, effective AI. Our API and extracts can be used within ITSI.

While log files are still extremely valuable, this net new data can provide additional value and also help you accelerate the delivery of more reliable software. Further, using OverOps you can gain insight into the quality of your apps and services in lower level environments such as dev, test and staging without increasing the size of your Splunk implementation.

The benefits of being "code-aware"

To capture this data OverOps uses static and dynamic analysis. A key element of this approach is a **Code Graph**, which is an index of all possible execution paths within your application code and is unique to every build or release. It's a map of where and why the code may log or encounter exceptions (i.e. break). The code graph allows OverOps to deliver a lot of net new insight, but one of the most valuable benefits is it allows us to **identify uncaught and swallowed exceptions**. The code graph acts to inform OverOps when it should capture an event that a developer never thought to capture. These are issues that were once impossible to identify and are often the most painful production issues to deal with. They simply aren't in log files so you might never chase them down until after a customer complains or a business indicator slips.

Being code-aware also allows OverOps to treat the code pipeline execution data as code. It **provides schema and structure** to investigate what is happening so we can deliver this as rich, structured output to your developers and to other tools that can then use the OverOps data. Conversely, events in a log file are usually formatted by a logging framework and are a schema-less string representation of an event, with limited information stored in plain text.

Finally, with an understanding of the code before it is executed, OverOps delivers all this value with only a **minimal impact on application performance**. The code graph allows the platform to proactively know an error or exception is going to be encountered so it acts without having to inform the SVM and slow it down. For more information, a demo or to try this yourself, please visit us at **www.overops.com/splunk**

About OverOps

OverOps provides net new machine data from applications and services to help organizations effectively evaluate the reliability of their software and implement a culture of accountability. OverOps combines static and dynamic analysis to collect complete contextual data for every error and exception thrown –both caught and uncaught– in any environment, including production with minimal performance impact, securely and without any requirement to modify code. It is code-aware and delivers net new and structured machine data that provides granular detail about every error, its related application and the environment in which it was found. This deep visibility into the functional quality of applications and services not only helps developers more effectively troubleshoot, but also empowers DevOps to build metrics dashboards, implement continuous reliability and enhance the entire software delivery supply chain. As more organizations aim to innovate faster and deliver a seamless digital experience for their customers, OverOps helps avoid costly downtime that can lead to lost revenue and brand degradation.