Log Management

CONTENTS

WRITTEN BY JOHN VESTER
SR. ARCHITECT/PRACTICE LEAD AT CLEANSLATE TECHNOLOGY GROUP, FREELANCE WRITER, AND ZONE LEADER AT DZONE

The landscape for developing applications has continued to evolve, as technology matures into providing dedicated segments focused on handling aspects of a system's required processing. The notion of an all-encompassing monolith continues to become less of a reality as a result of this progression.

With the popularity of microservices, a dedicated application program interface (API) is built to handle the heavy-lifting needs of a focused set of functionalities. These APIs often utilize integrations that can include an enterprise service bus (ESB), file-system based resources, cloud solutions, or even legacy systems.

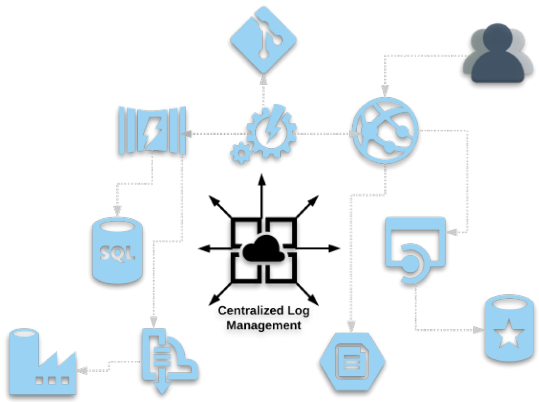
On the client-facing side, a collection of JavaScript-based frameworks is often employed to build out the presentation layer and handle interactions with any APIs needed.

The concept of "* as code" allows a majority of these underlying services to rely on a git-based repository and a build mechanism to transform the source code into functional systems.

This is the reality of a modern application design. When properly architected, these elements can work together quite well to present a fully functional application. However, when an unexpected issue occurs somewhere amidst these components, the ability to find the root cause can be difficult.

Each of these aspects has the ability to log events as they participate in some form of application service delivery. However, the format and structure of these logs vary from one system to the next. This is where the concept of centralized log management can provide a great deal of assistance.

In the diagram below, a centralized log management solution can ingest the log events from all components of an application in order to provide a single source to review and analyze.



graylog

TROUBLESHOOT WITH A CLEAR VIEW.

Easy access to all the log data from your entire app stack.

DOWNLOAD NOW

07:29:22
07:28:41
07:13:47
07:41:55
07:53:22
08:02:41 08:02:41 filterlog: 5,,,1000000103,em0,match,block,in,4,0x50,,53,0,0,DF,1,icmp
08:13:47 08:02:41 filterlog: 7,,,1000000105,igb0,match,block,in,6,0x00,0x00000,1,UDP,17
08:41:55 08:41:55 dhcpd: DHCPREQUEST for 10.0.0.187 (10.0.0.1) from fc:69:47:35:ee:87 (

When the logs are combined and organized properly, the production support engineer assigned to the situation can easily walk through the event without having to toggle between log files contained within disparate and proprietary systems.

Who Uses Centralized Log Management?

DEVOPS

Using the modern application design described in the introduction, consider the position a DevOps engineer may wind up in when an unexpected situation begins to occur. Source logs from all the components in the application landscape are buried deep within a file system, most likely using proprietary logging formats. Even when all the logs can be gathered, stepping through each log chronologically to determine the cause of the issue is a tedious task, at best.

By contrast, if a centralized log management system is in place, the logs are not only consolidated, but standardized and organized in a manner that will allow the DevOps engineer to play back the scenario under investigation. In doing so, the wider view provides a far greater opportunity to identify the root cause.

In most cases, this benefit justifies the use of centralized log management by a DevOps team. However, three other areas can become beneficiaries of a centralized log management solution.

SECURITY

A centralized log management solution can aid security teams by providing assistance when the team is scanning for unauthorized access to a given application or service. This can benefit anonymous external entities, as well as internal accounts.

Reports within the solution can be created within the centralized log management system to match logged events which may be an indicator of suspect activity. Consider the following errors being logged by an API under management by the solution:

```
2019-02-14 03:07:15.824 ERROR 36209 --- [ main]
AccessServiceImpl
: User id (someUserId) does not have access to perform
this operation

2019-02-14 05:27:07.212 ERROR 36209 --- [ main]
OrderServiceImpl :
User id (null) does not have access to review order
(1001001)

2019-02-14 10:17:37.542 ERROR 36209 --- [ main]
AccessServiceImpl
: Attempt to access API without a proper token on IP
127.0.0.1
```

Once enriched and ingested into the centralized log management solution, the report can be setup to keep off error type (36209), showing the date/time information, plus the message that was logged.

COMPLIANCE

As a part of the periodic compliance/audit tasks, a centralized log management solution can assist in compliance efforts by making sure the application or its users are in compliance with the expectations that have been established.

For applications required to comply to a regulatory guideline (SOX, FDA, HIPAA, etc.), employment of a centralized logging solution can provide a one-stop source for analysis and certification.

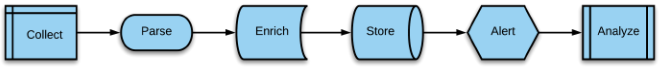
IT OPERATIONS

Monitoring and understanding the complexities of an IT infrastructure can become less intrusive when a centralized log management solution is adopted.

IT Operations staff can use the tool to gain an understanding as to how systems interact with each other, thus providing a tool to help make decisions when routine tasks (like system maintenance outages) are scheduled.

The Basic Process

A centralized log management solution utilizes a flow similar to what is noted below:



- **Collect:** The process of establishing a connection to a source system and ingesting the logs as they are natively created. A determination can be made regarding the log levels that are routed to the centralized log management solution, if necessary.
- **Parse:** Provides the ability to transform source log messages into a format that is standardized within the centralized log management solution. This is an important aspect since logs produced by an API using an “extended” Apache format will be different than a log from a database server (as an example).

Correctly parsed, the following log events:

```
1550149377 INFO Userid (someUserId) successfully logged
in from IP
127.0.0.1
02/14/2019 03:07:15.824 ERROR 36209 --- [ main]
AccessServiceImpl
: User id (someUserId) does not have access to perform
this operation
```

Could be updated to appear as shown below:

```
> 2019-02-14 08:02:57.000 (GMT) INFO Userid
(someUserId) successfully
> logged in from IP 127.0.0.1
> 2019-02-14 08:07:15.824 (GMT) ERROR 36209 --
AccessServiceImpl - User
> id (someUserId) does not have access to perform this
operation
```

The resulting parsed messages provide a standardized appearance for all messages, allowing the analyst to process the results of the logs more efficiently.

- **Enrich:** Introduces the ability to further define the log event. As an example, enrichment could functionally perform the necessary logic to analyze a logged IP address to make it easier for the analyst to understand the system or service that is being referenced. Application- or service-specific constants can also be transcribed here to limit the need for cross-reference logged information.

The following event:

```
1550149377 ERROR 90215 -- ServiceProviderCallOut --
Could not access
service on 127.0.1.1 due to an internal error code
10017.
```

Could be enriched as shown below:

```
**2019-02-14 08:02:57.000 (GMT)** -- ERROR 90215 --
> SeviceProviderCallOut -- Could not access
> **DocumentGenerationProcessor 4C** (127.0.1.1) due to
an internal
> **Socket Failure** (error code 10017).
```

With the above enrichment, the timestamp was converted to a GMT date, the IP address was translated to include the host system, and the error code was looked up to return additional information.

- **Store:** Persists the collected, parsed, and enriched logs into a data store utilized by the centralized log management solution. At this point, indexes and filters can be leveraged to provide greater insight into the native logs in the source system.
- **Alert:** With the necessary information stored in the centralized log management solution, alerts can be configured to catch events before they escalate to a higher severity level. At this point, centralized log management solutions can often send events to other systems to provide instant notification and escalation.
- **Analyze:** Using an interface into the centralized log management solution, an analyst can search, filter, and review all the events related to a given situation, without the need to review logs directly from the source system.

What To Log (and What Not to Log)

The time required to process and analyze logged information is important. As a result, a period of exploration should be taken in order to determine what information will be logged and what information will not be logged.

In the example noted in the introduction, the DevOps engineer needed to review the consolidated events for an application that encountered an incident. In most cases, the situation needs to be resolved as soon as possible. This effort could easily include logs from the microservices, databases, client application frameworks, and the security layer.

If the logs included aspects which were not pertinent for analysis, additional time would be required to review and discard this type of information.

Consider the following example of logs which are ingested from the authentication/authorization service participating in the centralized log management strategy:

```
1550149377 INFO Userid (someUserId) successfully logged
in from IP
127.0.0.1

1550149382 INFO SSID for someUserId updated to reflect
last login

1550149385 WARN Password for someUserId will expire in
26 hours

1550149415 INFO UserId (someUserId) granted access to
SomeApplication
via token #tokenGoesHere
```

While the information being logged is important information, it might be best to filter out all of the events, except for the following message:

```
> 1550149377 INFO Userid (someUserId) successfully
logged in from IP
> 127.0.0.1
```

In doing so, the number of log messages that would need to be reviewed during a crisis is minimized — especially in cases where hundreds (or thousands) of users are accessing the application.

SENSITIVE INFORMATION

Another aspect to consider is making sure that sensitive information (access tokens, database connection strings, encryption keys, account information, user information, etc.) is not stored in the centralized log management solution.

In the log example above, the tokenGoesHere log message should be suppressed from ingestion into the centralized log management solution since that token could be considered sensitive information. If the event is required, the message should be enriched to only ingest the following information:

```
> 1550149415 INFO UserId (someUserId) granted access to
SomeApplication
```

ESTABLISH GUIDELINES

The key is to establish guidelines that meet the need of the entire

user community that is utilizing the solution. This is no different than how any other application is architected — understanding the limits that are introduced with both not enough log events and too many log events. Once established, this information should be shared with teams that have the ability to create the events that are being captured by the centralized log management solution.

Centralized Log Management Checklist

When the decision is made to evaluate centralized log management solutions, the number of available solutions will certainly appear daunting. As a result, it is important to understand which features and functionalities are important for your implementation.

Below are some high-level aspects to consider:

- **Ease of data exploration:** How easy is it to take advantage of the solution to use and locate data for the type of user utilizing the system?

  Remember, the user is often more of a data explorer and not a data scientist. Any built-in reports or filters will lead to a better user experience when extracting results from the system.

- **Analyst efficiency:** How quickly does the system respond, including the ability to create complex searches or filters? Once data is returned, how easy are the results to comprehend and utilize?

  As noted above, time is often the driving component when trying to retrieve information from a centralized log management solution. Again, filters and reporting can help improve end-user efficiency.

- **Scalability**: How well does the solution work within your organization? Can all systems function within one centralized log management solution or would multiple instances be required? How about five years from now?

  It is important to understand how the centralized log management solution scales as more systems are introduced to the technology. While a degradation in performance is expected, anticipated response time is a good metric to use during side-by-side comparisons the larger the log data storage pool becomes.

- **Storage:** What are the storage expectations, where does the storage live, and what are the associated costs of a target implementation?

  It is also important to have an understanding on any boundaries around storage, especially with respect to system performance. Perhaps employment of second-tier storage can be utilized for data that is maintained for historical purposes over analytical purposes.

- **Log completeness:** Does the information retained include everything necessary or is extra effort required to retrieve data from an additional source?

  If you find yourself having to retrieve data from outside the centralized log management solution, there might be a gap in functionality with respect to your requirements.

- **Data enrichment functionality:** Does enrichment functionality exist? If so, how easy is it to utilize and maintain?

  It might be a good idea to review current log sources and understand more of the edge cases that could require data enrichment over typical enrichment usage patterns.

- **Open/closed source:** Does the solution utilize an open-source approach? How does this approach line up with other solutions that your entity employs?

- **Log collectors:** How are the log collectors defined? Are they proprietary or have they been created by a third-party/vendor? Can these log collectors be centrally managed by the solution or do they have to be independently managed at the log source level?

  Typically, proprietarily developed connectors lag behind those created by a third-party/vendor. If the connectors can be managed and configured by the centralized log management solution, there is less of a need for configuration to be maintained on the log source.

- **Configuration as code:** Does the solution employ a "* as code" approach, allowing the configuration of the centralized log management solution to be stored in a code repository?

  If your organization is embracing the concept of "* as code," centralized log management solutions that adopt this philosophy will be able to build and configure instances programmatically.

- **Class-specific functionality:** Does the solution contain features that help a particular class of user (DevOps, security, compliance, IT Ops) obtain common results? Do reports exist to locate regulatory exceptions, e.g. HIPAA?

  Having functionality built-in to assist specific use cases will lessen the time required for such groups to get up to speed and recognize value in the centralized log management solution.

- **API functionality:** Is there a public API available for the solution?

  Gaining a familiarity with any underlying application program interface (API) for the centralized log management solution could further justify or leverage the value returned from implementation.

- **Anticipated costs:** How is the product licensed?

  Gaining an understanding of the cost model will allow for product comparisons as time progresses and more components embrace centralized log management within your organization.

- **CLM vs. SEIM:** Is the target solution a centralized log management (CLM) solution or is it a security information and event management (SEIM) product? Do your current needs require one solution over the other, or perhaps both solutions?

  The requirements approved for the product will be a guide to understand what type of solution should be considered. It is important, however, to understand the differences between CLM and SEIM.

Conclusion

The value of a centralized log management solution can be justified by DevOps, IT Ops, and security/compliance groups within an entity's Information Technology division. With proper guidelines and a successful implementation, the benefits from utilizing centralized log management can result in not only being able to identify the root cause for an event, but to be able to receive notifications prior to issues being escalated to a higher state.

Implementation of a centralized log management solution should be treated no differently than any other solution being introduced into an organization. Requirements and understanding should drive what components are logged with the right level of logging being ingested into the solution. Any sensitive information should be omitted from being ingested into the centralized log management solution, always adhering to any regulatory guidelines.

A major success factor to a centralized log management solution is the ability for the analyst to find the information needed and have this take as little time as possible, without having to go outside the solution itself. Centralized log management solutions, which include pre-designed reports and functionality to assist all classes of users, can provide a faster turnaround with performing routine requests.

Time should be taken to understand whether a centralized log management solution is the right fit for your needs or if your requirements fall into a security information and event management solution set. While some solutions may operate well as either, understanding your target end-state is helpful in providing the best solution to your corporation.

Written by **John Vester,** *Sr. Architect/Practice Lead at CleanSlate Technology Group and Zone Leader at DZone*

John Vester is Information Technology professional with 25+ years expertise in application development, project management, enterprise integration, and team management. Currently focusing on enterprise architecture/ application design/continuous delivery utilizing object-oriented programming languages and frameworks. Experience in building Java-based APIs against React and Angular client frameworks, integrating architecture and design, CRM design and customization, with additional experience using both C# (.NET Framework) and J2EE (Spring, plus various other frameworks).

DZone
A DEVADA MEDIA PROPERTY

DZone communities deliver over 6 million pages each month to more than 3.3 million software developers, architects, and decision makers. DZone offers something for everyone, including news, tutorials, cheat sheets, research guides, feature articles, source code, and more. "DZone is a developer's dream," says PC Magazine.

Devada, Inc.
600 Park Offices Drive
Suite 150
Research Triangle Park, NC

888.678.0399 919.678.0300