Aggregated Containerized Logging Solution with Fluentd, Elasticsearch and Kibana

Kaichuang Yang
University of International Relations
Beijing, China

ABSTRACT
In the operation and maintenance of cloud production lines, access to logs from containers in order to perform day-to-day investigation and debugging is needed. Logs must be accessible beyond the life of a container. Logs from across the cluster need to be aggregated to allow log searching, backup, and storage. As a solution, this paper present a diagram combined with Fluentd, Elasticsearch and Kibana, that enables the deployment and management of multiple containers log reviewing with a creatively way. The solution perform well efficiently on log collection and indexing. Even though the prototype has been effectively used to deploy and manage containers logging data, serial risks are still needed put into consideration as important issues.

General Terms
Data and Information Systems, Software Testing

Keywords
Logging solution, Fluentd, Elasticsearch, Kibana, Proxy, Performance testing.

1. INTRODUCTION
Log availability is currently limited to a user using the name of a currently-existing container to retrieve the logs. Once the container is deleted, logs are no longer accessible; there is no mechanism for aggregating or querying logs from containers distributed across the cluster. The current log functionality also provides no support to retrieve historical logs once the current node logging infrastructure adds log rotation.

Therefore, the logging system must function in a multinational environment where users are restricted to seeing logs for which they are granted access by an administrator.

With the following motivations, we put forward a solution attempting to meet the developer's needs.

- Developers should be able to review any log to which they have access.
- Developers should be able to follow and see logs with minimal latency (e.g. < 5s).
- Developers should be able to review logs even after the container from which it originated is deleted.

2. SOLUTION DIAGRAM
Aggregated logging updates the logging API while adding the following main components to the system: a node agent, a log aggregator, a log aggregator proxy, and visualization interface. These components work together to transfer logs from a volatile store on individual nodes in the cluster to a central location for backup and storage. Each component is configured and designed to operate in the cluster's multi-tenant environment. [1]

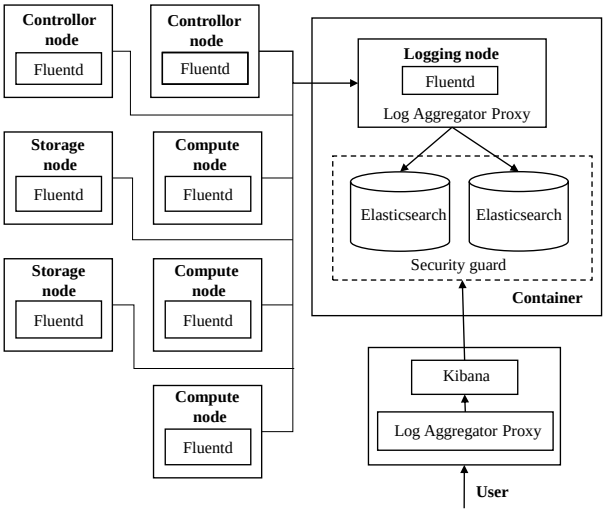


Fig 1: Containerized logging component diagram

2.1 Cluster Node
Containers running on a node will be configured to write logs to disk and rotate them based on size, limited to a fixed number of rotated files. These features should be available with Docker. Docker document site does not seem to have caught up but source docs show new options for rotating with the json-file logging driver.

2.2 Node Agent - Fluentd
Fluentd is similar in that it has inputs and outputs and a matching mechanism to route log data between destinations. Internally, log messages are converted to JSON which provides structure to an unstructured log message. Messages can be tagged and then routed to different outputs.

An agent will be installed as a static container on each node that is responsible for associating metadata with the container logs and transferring them to the log aggregator. The agent will utilize the log file name to retrieve additional container information from the master for use as metadata. Container log file names follow a naming convention that consists of: the container's name and the container's pod.

The agent will add the metadata to a cache in order to minimize the number of calls to the master. The cache will expire entries based on a configurable time. The agent will be configured with mutual TLS between agent and log aggregator proxy to ensure only the agent can write logs.

2.3 Log Authentication Proxy
The log proxy is a multi-use component that provides authorization and request transformation services. Its primary responsibility is securing other components and controlling access and authorization based on a user's authorization

token. It may be required to transform queries and responses if other components cannot be made to scope their requests by user as we desire. The proxy will only be responsible for controlling access to Kibana and making a user's authorization token available for subsequent requests to Elasticsearch. It is unnecessary to provide any transformation services.

2.4 Log Aggregator - Elasticsearch

The log aggregator is deployed as a pod with multiple containers that is accessed from the logging service. One container is the Elasticsearch log aggregator. The second container is the proxy mentioned previously. Multiple replicas of the pod should be deployed as a cluster behind a kubernetes service, which can remain internal-only for access by the master.

Securing Elasticsearch will be accomplished using the proxy to front the Elasticsearch log aggregator. The proxy will be configured with mutual TLS between it and clients (e.g. node agent) looking to write log data to the aggregator. This will ensure only authorized agents may write logs.

2.5 Log Visualization Tool - Kibana

A log visualization tool (e.g. Kibana) could be accessed from the web console by a link that navigates to the tool. A user will return to the web console via a link in the tool that navigates them back to the web console.

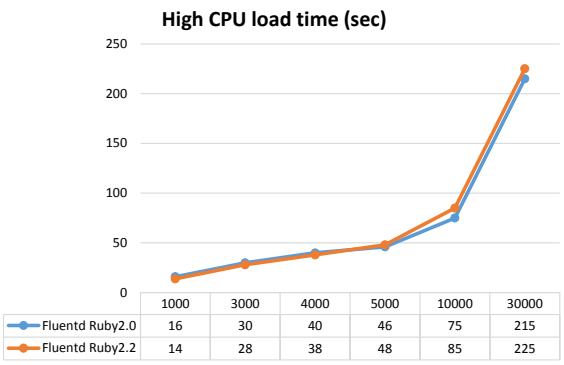
Log visualization is deployed separately, linked from the web console, and configured to retrieve data from the logging service. Kibana will need modifications to accept a token from the web UI, persist it in a session appropriately, correctly indicate an expired token, and provide a link back to the web UI. It will also need modifications to remove configuration items from its UI that aren't appropriate for a single user in a multitenant tool.

Kibana has no concept of user at this time and does not have a mechanism for scoping the Kibana profile which including saved queries, visualizations, dashboards to the user.

3. PERFORMANCE TESTING

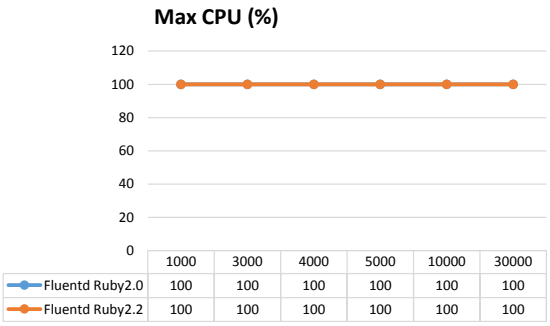
In order to create performance tests to verify and measure assumption about the solution's resource intensive, we push 10.000 messages via Fluentd to Elasticsearch [2]. There are the CPU consumptions per PID. The Fluentd process is picking up 10.000 messages waiting in the queue and pushing it into Elasticsearch.

Table 1. High CPU load time (sec) performance



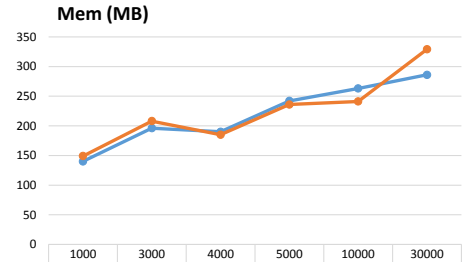
	1000	3000	4000	5000	10000	30000
Fluentd Ruby2.0	16	30	40	46	75	215
Fluentd Ruby2.2	14	28	38	48	85	225

Table 2.Max CPU (%) performance



	1000	3000	4000	5000	10000	30000
Fluentd Ruby2.0	100	100	100	100	100	100
Fluentd Ruby2.2	100	100	100	100	100	100

Table 3. Memory performance



	1000	3000	4000	5000	10000	30000
Fluentd Ruby2.0	140	196	190	242	263	286
Fluentd Ruby2.2	149	208	185	236	241	329

4. RISKS

There are serial risks currently faced by logging solutions in the following areas [3].

4.1 Docker logs

Docker can probably only log to a single destination. If we want to support use cases requiring local logs as a backup to aggregated logs, we probably need to write logs locally and have the agent use that as a source, which is the current design.

4.2 Securing Elasticsearch and Kibana

Currently it is uncertain how to secure the Kibana integration since the current design would require it to pass along a user's token in order to retrieve information from a secure Elasticsearch. Investigation into Kibana plugins has shown the project advises against writing them. Kibana appears to support basic authority for use with Shield; there may be some way to shoehorn our token in there.

4.3 Latency

It is not known how long it will take for logs to be retrievable from Elasticsearch once generated. A few seconds is probably acceptable, a few minutes not so much. If latency is too long, then having local logs as a backup becomes far more important.

5. CONCLUSION

This solution could bring developer's much more convenience on containerized environment, but more research and development still needed. If the aggregator client can be added to Kibana by way of plugin or fork, or whether it can be simulated with a transformation by the proxy in front of Elasticsearch. Clients that wish to retrieve data from the aggregator will present an authorization token to the proxy along with their search query. The proxy will utilize the token to determine the client's read access to containers. The plugin

could provide secure communication, via mutual TLS, between Elasticsearch instances that are part of a cluster. Therefore, the solution's performance would make an obvious promotion. In the future research, serial risks are still needed put into consideration as important issues.

6. REFERENCES

[1] T. Prakash, M. Kakkar and K. Patel, "Geo-identification of web users through logs using ELK stack," 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), Noida, 2016, pp. 606-610.

[2] P. P. I. Langi, Widyawan, W. Najib and T. B. Aji, "An evaluation of Twitter river and Logstash performances as elasticsearch inputs for social media analysis of Twitter," Information & Communication Technology and Systems (ICTS), 2015 International Conference on, Surabaya, 2015, pp. 181-186.

[3] W. Wongthai, F. Rocha and A. v. Moorsel, "Logging Solutions to Mitigate Risks Associated with Threats in Infrastructure as a Service Cloud," Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on, Fuzhou, 2013, pp. 163-170.

[4] Logging Evaluation of the Ordovician Carbonate Reservoir Beds in the Lungudong Region, Tarim Basin [J]. Acta Geologica Sinica (English Edition), 2010, 05:1141-1156.

[5] Effects of electric-acoustic and acoustic-electric conversions of transducers on acoustic logging signal [J]. Chinese Science Bulletin, 2012, 11:1246-1260.