Log Management Systems

Why Do We Log Data?

- Record user activity such as log times and application access. [1]
- Record request data, URLs, file access, system command usage, etc. for security purposes. [1]
- Record errors and warnings for debugging. [1]

Challenges of Logging:

- Logs from different applications may have different formats and may be distributed at different levels and transports, so difficult to analyze for humans. [1]
- Logs written to logging databases is another option. However, this involves using a library to connect that application's logger to the database and configuring the proper format and minding performance. [1]
- Also, there may be logs from applications which may not be under our control. [1]

A log management system allows us to handle these logs in a unified way. Such handling may include generating alerts from these logs, analysing and predicting application errors and also archiving parts of / the entirety of these logs. Some LMSs are discussed below:

ELK Stack (Currently Elastic Stack):
It consists of 3 parts: Elasticsearch, Logstash and Kibana. [2]

Elasticsearch is a **NoSQL** database that helps store inputs and logs. [2]

- It is based on the **Lucene** search engine. [2]

Logstash is a log pipeline tool that accepts inputs from various sources and exports the data to various targets. [2]

- Here, data is handled as events. [2]
- It can handle different kinds of data. Eg - logs, chat messages, ecommerce orders, etc. [2]
- A logstash pipeline consists of inputs, filters and outputs. [2]
- The input plugins take data (Eg - reading events from a log file), the filter plugins filter data (Eg - parsing CSV, JSON or XML or lookup of data) and output plugins, called **stashes,** are where the data is sent. [2]
- It can read files. [2]
- If it receives any unstructured data, it has to structure that data. [2]

Kibana is a visualization UI layer that helps a developer to monitor logs. [2]

X-Pack is a set of features of Kibana and ElasticSearch.

- It adds authentication and authorization info to Elasticsearch & Kibana. [2]
- **Kibana** can integrate with LDAP, Microsoft Active Directory for authentication and RBAC. [2]
- It allows us to monitor the CPU and RAM usage of ElasticSearch. [2]
- It can set up alerting for various things. [2]
- It can generate frequent reports, detect anomalies in log data and predict future trends using Machine Learning. Eg - Predicting web traffic, number of errors. [2]
- It has **Graph**, a utility that can find relations between different types of data. [2]

Beats is a collection of data shippers.

- They are lightweight agents, each with a single purpose that are installed on servers. [2]
- They send data to Logstash or ElasticSearch. [2]
- There are many different kinds of data shippers on Beats and they all collect different types of data. Eg - **FileBeat** is a beat that collects data from log files while **MetricBeat** is a beat that collects metrics such as CPU, Memory and Disk Usage. [2]

Beats & **Logstash** send data to **ElasticSearch** which stores and analyzes it while **Kibana** helps visualise it. [2]

Grafana Loki:

- Log collection is docs by **PromTail** or some other ingester. It sources logs from various sources such as **files**, **systemd journals** and **K8S pods**. [3]
- Log collection can also be done using other tools such as **FluentD** or FluentBit. [3]
- It is also possible to send logs to **Loki's** HTTP API. [3]
- Loki supports multi-tenancy, i.e., logs from multiple users / environments within a single running instance. Each tenant's logs are logically separated. [3]
- Logs are indexed by labels, allowing Loki to index large number of logs efficiently. A label can represent environment, service name, pod, or any custom metadata. [3]
- **LogQL** is the language that Loki uses to filter, parse and analyze logs. Value pairs can be extracted by using **JSON, Regex** / **LogFMT**. Many metrics can be generated from log data. Eg - (Log Occurrences, Line Graphs, Histograms, etc.) [3]
- Integration with **Grafana** allows the following: dashboards to be created, log-based alerts and data exploration and insights. [3]
- Logs are split and stored as chunks and indexed by metadata, speeding up queries. [3]
- Horizontal scaling is supported for large deployments. [3]

- Loki supports storing logs on S3, MinIO, Cassandra, DynamoDB, localStorage, etc. [3]
- Supports the log retention policies. [3]

Fluentd:

- It is a fully free and open-source log collector. [1]
- It can accept logs from many systems such as Servers, SysLog, Mobile Apps, Databases, etc. and converts all of them into a universal format and, sometimes, adds additional data to them (Eg - **pod_name**, **name_space**, **container_name**, etc). [1]
- It is also possible to group logs together using some metadata, i.e., **pod_name**, **name_space**, **container_name**, etc. [1]
- It can also send the logs to several destinations. Eg - Elastic, Kafka, MongoDB, etc. It is possible to route logs from each source to a particular destination. [1]
- Fluentd isn't tied to any backend. [1]
- No Vendor-Lock-In while using Fluentd. [1]

Installation and Use:

- Fluentd needs to be installed on Kubernetes through DaemonSet. [1]
- DaemonSet is a component that runs on each Kubernetes node. This means that if there are N Kubernetes Nodes, then there will be a Fluentd pod running on each Kubernetes node. [1]
- We have to configure Fluentd using the fluentd config file. [1]
- To use Fluentd we must configure its plugins. The first plugins to be configured are the data sources (From which data are collected), then we configure how the data is to be processed (by parsers), then we can use record transformers to enrich the data. [1]
- When Fluentd processes the data, it stores it on a hard drive. This way, if the fluentd pod restarts or if the server restarts, the data will still be there and thus, additional storage such as Redis Database isn't necessary. [1]
- If the backend isn't available, then fluentd can keep sending logs to it till it becomes available. [1]

Some Use Cases:

- It is possible to use a combination of Fluentd, ElasticSearch and Kibana to create a log analysis and visualisation engine similar to splunk. [5]
- It is possible to create a tool to send alerts to an email based on HTTP 500 error code detected in logs using the **grepcounter** plugin and **mail** plugins of fluentd. [5]
- It is possible to build a data collection and analysis system using the **fluentd-td** plugin and **treasure-data** (A customer data platform). [5]

Differences Between Elastic Stack and Grafana Loki

Topic	Elastic Stack	Grafana Loki
Indexing and Storage	It indexes the entire contents of the logs thus providing more granular search capabilities but also higher storage requirements. [4]	It indexes only the metadata of the logs and not all the logs. This makes it more efficient with log processing. [4]
Scalability	More resource intensive. Elastic Stack clusters can be scaled horizontally and logstash clusters can be scaled by distributing processing across multiple instances. [4]	It can handle large volumes of logs with low infrastructure costs but struggles with specific types of logs such as unstructured logs or logs with many unique labels. [4]
Ease of Use	More advanced and steeper learning curve. Need to understand each component of the system. [4]	Has straightforward setup and configuration and less complexity in system management due to focus on metadata. [4]
Use Cases	It is useful when detailed log analysis, flexible log management setup & in-depth log analysis features are needed. [4]	It is useful for cloud-native projects since it is lightweight, and since it uses fewer resources, it si suitable for budget-constrained projects. [4]
Cost Efficiency	It provides more granular search capabilities but at the cost of higher compute and storage needs and thus has higher operational cost. [4]	Since it indexes only log metadata, so it has lower infrastructure and compute needs and so results in lower operational costs. [4]

Comparison of Log Management System Features:
Note: All of them have a dashboard.

Features	**Splunk**	**ELK**	**Grafana Loki**
Real-Time Log Monitoring	*Timeline* -> View All Your logs by time; *Logtable* -> Scan all logs and filter logs by word; *Content Control* -> Search logs by field; *Related Content* -> See the metrics, traces, and infrastructure related to a specific log.	Done by Logstash. It can be configured to watch specific log files / directories for any changes. It uses input plugins to read info from a variety of sources such as text files, TCP / UDP ports or message queues.	Less compared to other systems as only log labels are indexed.
Alerting Mechanisms	Creating Alerts using LogEvents to index and search metadata. Other types of alerts can be created based on specific data patterns / thresholds. Alerts can also be delivered by SMS / email.	Possible to create alerts based on log data, thresholds and events.	Threshold-based alerts and log-based pattern matching alerts.
Log Data Analysis	Search Processing Language (SPL) supports retrieving events from index, allows PERL regex and log correlation, etc.	Extraction and parsing of unstructured data using **grok,** parsing of JSON and XML data using filters. Creating Queries through DSL query language on Elasticsearch.	Logs can be queried using **LogQL**.. Supports filtering logs by labels, content search, and combining queries with metrics.
Winston Integration	Use **winston-splunk-httplogger** as transport.	Use the **winston-elasticsearch** as transport.	Use **winston-loki** as transport.

REFERENCES

[1] https://www.youtube.com/watch?v=Hqn5p67uev4 [ELK Stack Overview]
[2] https://www.youtube.com/watch?v=5ofsNyHZwWE [FluentD Simply Explained]
[3] https://grafana.com/docs/loki/latest/get-started/overview/
[4]https://opsverse.io/2024/07/26/grafana-loki-vs-elk-stack-for-logging-a-comprehensive-comparison/
[5] https://docs.fluentd.org/quickstart