Logging Reporting Assignment

1. List some logging and visualization tools available in the market with the preferred scenario to use one over other.

Logs are crucial data about users, traffic, performance and security related aspects of a distributed environment. Visualizing these logs and interpreting it in an understandable way is as much important. Some of the tools that are available now for logging and visualization are listed below

- ELK Stack
- Graylog
- Splunk
- Sumo Logic
- Loggly

ELK Stack

ELK stack is an open-source project made up of many different tools for application data analysis and visualization. **Logstash**, specifically, was made for the collection and management of log files. Beyond log aggregation, it includes **ElasticSearch** for indexing and searching through data and **Kibana** for charting and visualizing data. Together they form a powerful log management solution.

When to use it: If you want an open-source tool. If you're interested in implementing the entire Elastic stack or at least see value separately in using ElasticSearch or Kibana and want the interactive benefits that come from combining these tools.

GrayLog

Graylog is an open-source log analyzer backed by **MongoDB** as well as **ElasticSearch** (similar to Logstash) for storing and searching log errors. It's mainly focused on helping developers detect and fix errors in their apps, but they've also released an official enterprise-ready platform.

When to use it: Graylog is more targeted towards developers than other open-source log management tools. Plus, if you want a log management tool that aims to be both enterprise-ready and is open source, Graylog definitely deserves to be in the mix with Elastic/Logstash.

Splunk

Splunk is the biggest tool in the log management space. It's well-established, full-featured, and enterprise-class. It's unique in this space as an on-premises tool (although they have come out with a Cloud version as well).

When to use it: Enterprise companies with lots of feature needs and a variety of data that needs analyzing.

Sumo Logic

Sumo Logic was founded as a SaaS version of Splunk, going so far as to imitate some of Splunk's features and visuals early on. Since then, Sumo Logic has developed into a full-fledged enterprise-class log management solution in its own right. Sumo Logic is the most enterprise-focused of the cloud-native log analyzers.

When to use it: If you're an enterprise-type company but are willing to sacrifice some features for the benefits of SaaS, Sumo Logic is worth exploring. It's also good if you have a strong focus on security. It's not just developer-oriented as a tool either, with benefits for security teams and business purposes.

Loggly

Loggly is a robust log analyzer, focusing on simplicity and ease of use. It's targeted for developers and DevOps – making it less enterprise-focused.

When to use it: Primary use cases are for troubleshooting and customer support scenarios. It's a good tool for a DevOps team.

PaperTrail

PaperTrail is a simple way to look and search through logs from multiple machines, in one consolidated easy-to-use interface. It's a SaaS tool designed to enhance the logs you already collect or generate.

When to use it: If you want a simple and straightforward tool without lots of extra bells and whistles. If you want a stripped down and basic log analyzer that is good for looking at log files in aggregation and doesn't try to be anything more.

2. Mention 10 best practices when logging. Why is log formatting necessary?

Some Best Practices when logging is listed below.

- Log at the Proper Level
  Not all events are equally important for troubleshooting or network monitoring, and being able to differentiate severe ones from irregular or normal logs is critical. Logging levels should be a part of all logs for all devices, including statements such as FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ALL, or OFF.

- Find the Middle Ground in Detail
  When configuring what kind of information your logs provide, you need to find the middle ground in the amount of detail your log messages show.

- Logging Standard
  Logs needs to be consistent so using a logging standard is preferrable. Ensure all logs show the timestamp and the names of the host and logger. Other valuable information can include event or user ID, application version, and metrics.

- Log in Machine Parse able Format
  Log entries are really good for humans but very poor for machines. Sometimes it is not enough to manually read log files, you need to perform some automated processing. *This is also the reason why log formatting is necessary.*

- Meaningful Log Messages
  As logs are the first thing to look into in the time of emergency, it needs to be simple and meaningful which in turn helps understand what happened.

- Add Context
  Without proper context, those messages are only noise, they don't add value and consume space that could have been useful during troubleshooting.

- NOT logging Sensitive information
  Sensitive information like passwords, credit card information and social security numbers must not be logged. It may expose certain vulnerabilities to the system.

- Anticipate common scenarios
  Logging with some common scenarios in mind ensures the logs provide direct value to your organization. Logs aren't just for incident response. Logs can help with other parts of your business, such as performance profiling or gathering statistics.

- Log In English
  English means your messages will be in logged with ASCII characters. If your message uses a special charset or even UTF-8, it might not render correctly at the end, but worst it could be corrupted in transit and become unreadable.

3. Create a file in your system. Whenever a someone performs some action(read, write, execute) on that file, the event should be logged somewhere.

To track a file in the system we can use Auditd, which is a monitoring tool used for auditing activity like authentications, failed cryptographic operations, abnormal terminations, program execution, and SELinux modifications.

CentOS ships with Auditd, but Ubuntu does not so to install it we use the command:

```
~ sudo apt install auditd
```

Now to monitor a file at the location **/home/prajesh/checkme/watching.txt** we edit the auditd config file at **location /etc/audit/rules.d/audit.rules** to add the rules for *File Access monitoring:*

```
-w /home/Prajesh/checkme/watching.txt -p rwx -k watch_file
```

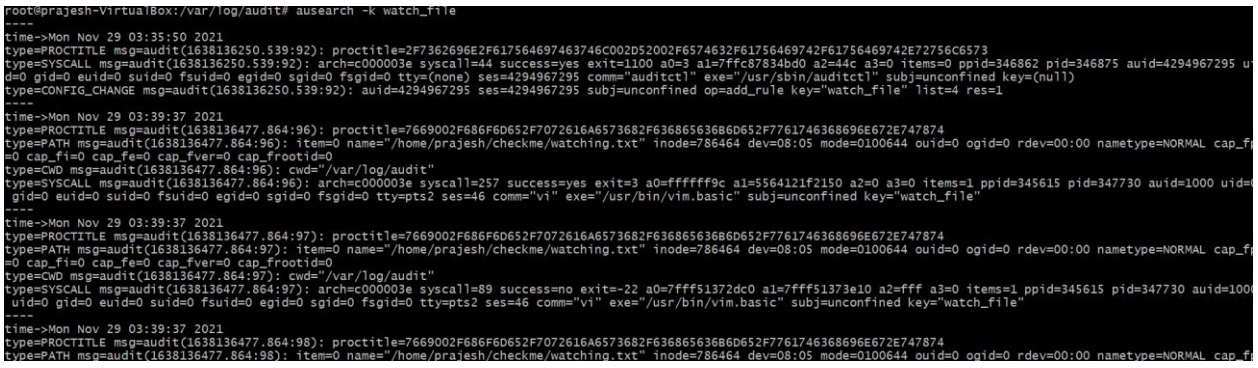
Here the format of the rule for File Access monitoring is

```
-w <path/to/file/location> -p <permissions/to/monitor> -k <key_name>
```

So now any changes in the file will be logged in the logs file in the location **/var/log/audit/audit.log.**

We can search through the log file using **ausearch** tool as such:

```
~ ausearch -k watch_file
```



4. Install Logstash in your system. download a sample nginx log from https://github.com/elastic/examples/blob/master/Common%20Data%20Formats/nginx_logs/nginx_logs, parse the logs using Logstash. The parsed output must contain the geographical information like country, state etc. that the request is originating from. save the parsed output to a file in your system.

So first to install Logstash use the following commands:

```
~ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch |
  sudo apt-key add –
~ sudo apt-get install apt-transport-https
~ echo "deb https://artifacts.elastic.co/packages/7.x/apt stable
  main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
~ sudo apt-get update && sudo apt-get install logstash
```

Logstash works with 3 step processes:

- Input
- Filter
- Output

So, to configure these we will need to create a configuration file for parsing the Nginx log sample as given below

```
input {
      file {
            path => ["/home/prajesh/logstash/nginx_logs"]
            start_position => "beginning"
      }
}
filter {
      grok {
            match => {
                  "message" => [ "%{IPORHOST:remote_ip} -
                        %{DATA:user_name}
                        \[%{HTTPDATE:access_time}\]
                        \"%{WORD:http_method} %{DATA:url}
                        HTTP/%{NUMBER:http_version}\"
                        %{NUMBER:response_code}
                        %{NUMBER:body_sent_bytes}
                        \"%{DATA:referrer}\" \"%{DATA:agent}\""] }
      }
      geoip {
            source => "remote_ip"
```

```
        }
}
output {
        stdout { codec => rubydebug }
        file { path => "/home/prajesh/logstash/output.json"}
}
```

Input is used to get the log file

Filter is used to apply some operation to the logs, here we used **grok** and **geoip** plugin to parse and get the geographical information from the IP address.

Output is used to get the output of the parsing from Logstash.

In this way the sample log file is parsed into a Json file with geographical information as given in the screenshot