PROJECT REPORT

Title: *Centralized Log Monitoring on Linux Using ELK Stack and Filebeat*

Created By: *Rumaisah Haroon*

Centralized Log Monitoring on Linux Using ELK Stack and Filebeat

Introduction

Logs are like a behind-the-scenes diary of everything that happens on a computer system. Whether it's someone logging in, an error occurring, or a process starting or stopping, it all gets recorded in logs. This project focuses on setting up a centralized logging system on a Linux machine using the ELK Stack (Elasticsearch, Logstash, Kibana) along with Filebeat. The goal is to collect and view logs in one place, making it easier to understand what's going on inside the system.

Purpose of the Setup

The purpose of this setup is to gather logs from the local system and send them to a centralized platform where they can be viewed and analyzed more easily. Instead of going through several log files manually, the ELK stack allows us to see everything through a clean, visual interface. Filebeat is used to monitor specific files like syslog and auth.log and ship those logs to Elasticsearch. This setup helps build a basic but powerful log management system that can be extended later to include other types of logs—like network activity or custom application logs.

Overview of Log Management

Every Linux system generates logs automatically. These logs are saved in files, usually under the /var/log/ directory. Two important examples are:

- /var/log/syslog: Contains general system messages, service startups, shutdowns, errors, etc.
- /var/log/auth.log: Contains authentication-related logs like login attempts, sudo usage, etc.

Managing these logs manually can get difficult, especially when there's a lot of data or multiple systems involved. That's why we use log management tools—to collect, organize, and search logs more efficiently.

Why Centralized Logging Matters

Having a centralized place to store and view logs is really useful. It saves time, helps with debugging, and improves overall system monitoring. Here are some reasons why centralized logging is important:

- It's easier to search and filter logs through Kibana's interface.
- You can quickly spot errors, failed logins, or suspicious activity.
- Visual dashboards give a better understanding of what's happening.
- Everything is in one place, so there's no need to check multiple files manually.

Overall, centralized logging gives better visibility into the system, which is especially important for security, troubleshooting, and keeping track of performance.

System Overview

The main focus of this project is to explore the use, understand the benefits, and get a clear picture of how the ELK Stack works as a whole. ELK—short for Elasticsearch, Logstash, and Kibana—is a powerful combination of tools that help in collecting, processing, storing, and visualizing log data. Through this project, I aimed to not just set up the ELK Stack, but also to understand how each component plays its part in creating a centralized logging system.

Components used:

The primary components used for this project is **Elastic Search, Logstash, Kibana** and **Filebeat.** Each component serves a different role in the system which is further explained ahead.

Elasticsearch

Elasticsearch is like the brain of the ELK Stack. It stores all the log data in a smart, searchable format. Once logs are in Elasticsearch, we can search for anything instantly, even across millions of log entries. it's built for speed and scale, so whether it's system logs, web traffic, or security events, Elasticsearch can handle it.

In this project, it is responsible for receiving logs (sent via Logstash) and indexing them so they can be searched and filtered quickly. It allows me to dig through large amounts of logs almost instantly, which is extremely helpful for spotting unusual patterns or checking system behavior.

Key points:

- Stores logs in structured, searchable format
- Supports fast querying and filtering
- Scales well with large amounts of data
- In this project: Stores and indexes logs from my local system

Logstash

Logstash acts like a middleman. It receives logs from different sources (like Filebeat), processes them, and then forwards them to Elasticsearch. Think of it as a filter where we can clean up or organize the logs before saving them.

In my setup, Logstash ensures that the logs are well-structured and clean before storage. It also allows customization in case I want to add tags, extract fields, or apply filters.

Key points:

- Acts as a pipeline between Filebeat and Elasticsearch
- Can filter, parse, and transform logs
- Supports multiple input and output plugins
- In this project: Processes logs from Filebeat before they go to Elasticsearch

Kibana

Kibana is the visual tool of the stack. It connects to Elasticsearch and lets us see the data through graphs, charts, and dashboards. Instead of scrolling through boring text logs, Kibana makes it easier to understand what's going on through visuals. It also lets us run queries, build custom dashboards, and even set up alerts for specific events.

With Kibana, I was able to create dashboards and graphs that helped me understand what was happening on my machine—from system activity to login attempts.

Key points:

- Provides a UI to view and explore logs
- Allows custom dashboards and visualizations
- Helps detect patterns, trends, or unusual behavior
- In this project: Lets me visualize and analyze the logs collected from my system

Filebeat

Filebeat is a lightweight log shipper developed by Elastic, designed to forward and centralize logs. Although it is **not** officially a part of the ELK Stack (Elasticsearch, Logstash, Kibana), it is commonly used alongside it to send log data efficiently. Filebeat runs on the client machine (in this project, my own system) and keeps an eye on specified log files. Whenever there's a new log entry, Filebeat quickly reads and forwards it to Logstash or directly to Elasticsearch.

In this project, Filebeat plays an essential role by collecting system logs—such as authentication events and system messages—and sending them into the ELK pipeline for analysis and visualization. It acts as the entry point of our log management system.

Key points:

- Not officially part of ELK but tightly integrated with it
- Monitors log files in real-time
- Sends logs to Logstash or Elasticsearch
- Uses very little system resources
- In this project: Collects logs from files like /var/log/syslog and /var/log/auth.log and forwards them to Logstash.

Tools used:

In the process of setting up this centralized logging system using the ELK Stack, several important tools and technologies were used to manage and streamline the deployment:

Docker

Docker is a platform that helps developers and system administrators create, deploy, and run applications using containers. A container is like a lightweight, standalone package that contains everything needed to run a piece of software—like code, libraries, system tools, and settings. Think of it as a mini-computer that runs only what you need and nothing more.

Unlike traditional virtual machines, containers don't need a full operating system inside each instance. This makes them faster, more efficient, and easier to manage.

Why use Docker?

Docker was used to containerize and run the ELK Stack components. It allowed me to quickly spin up Elasticsearch, Logstash, and Kibana in isolated environments without having to install each

service manually. Docker Compose made it easier to define and manage multi-container applications through a single configuration file.

- It makes complex software setups (like ELK) easier to install and run.
- It removes the “it works on my machine” problem—containers behave the same on every system.
- It saves time, especially when dealing with multiple services that need to work together.

systemd

systemd is the system and service manager for many modern Linux distributions. I used systemd to control services (like Filebeat), which was installed directly on the host machine. With simple commands like start, stop, and status, I was able to manage how and when services run on boot.

Example usage:

```
sudo systemctl start filebeat
sudo systemctl enable filebeat
sudo systemctl status filebeat
```

Architecture Diagram

Setup Process

Docker setup:

- To set up the docker, I had to first update and upgrade my linux machine (ubuntu)

```
sudo apt update
sudo apt upgrade
```

- After this we need to install docker.io

```
sudo apt install docker.io -y
```

- Now enable Docker to start on boot

```
sudo systemctl enable docker
sudo systemctl start docker
```

- Install Docker Compose

```
sudo apt install docker-compose -y
```

- To ensure docker is up and running, test Docker,

```
docker --version
docker-compose --version
```

ELK Setup

- Create folder for elk and change the directory to that of elk

```
Mkdir elk
cd elk
```

- Create a docker-compose.yml in nano and write the following (according to your need)

```
version: '3.7'
services:
 elasticsearch:
  image: docker.elastic.co/elasticsearch/elasticsearch:7.17.10
  container_name: elasticsearch
  environment:
   - discovery.type=single-node
   - xpack.security.enabled=false
  ports:
   - 9200:9200
  mem_limit: 1g

 logstash:
  image: docker.elastic.co/logstash/logstash:7.17.10
  container_name: logstash
  volumes:
   - ./logstash.conf:/usr/share/logstash/pipeline/logstash.conf
  ports:
   - 5044:5044
```

- Create a logstash.conf in nano and write the following (according to your need)

```
input {
 beats {
  port => 5044
 }
}
filter {
 grok {
  match => { "message" =>
"%{SYSLOGTIMESTAMP:timestamp} %{SYSLOGHOST:host} %{>
 }
}

output {
 elasticsearch {
  hosts => ["http://elasticsearch:9200"]
  index => "filebeat-%{+YYYY.MM.dd}"
 }
}
```

Purpose of such files:

Docker-compose:

This file automates the setup of multiple services (like Elasticsearch and Logstash) using Docker. It defines how containers should run, what ports to open, and how to link files like logstash.conf. It helps start the entire system with one command, making deployment faster, repeatable, and easier to manage.

Logstash.conf:

This file defines how Logstash should handle logs. It listens for logs on port 5044 (from Filebeat), uses a grok filter to extract fields like timestamp and host, and sends the processed logs to Elasticsearch under an index named filebeat-<date>.

- Start the ELK Stack:

```
sudo docker-compose up -d
```

- Check if the containers are running:

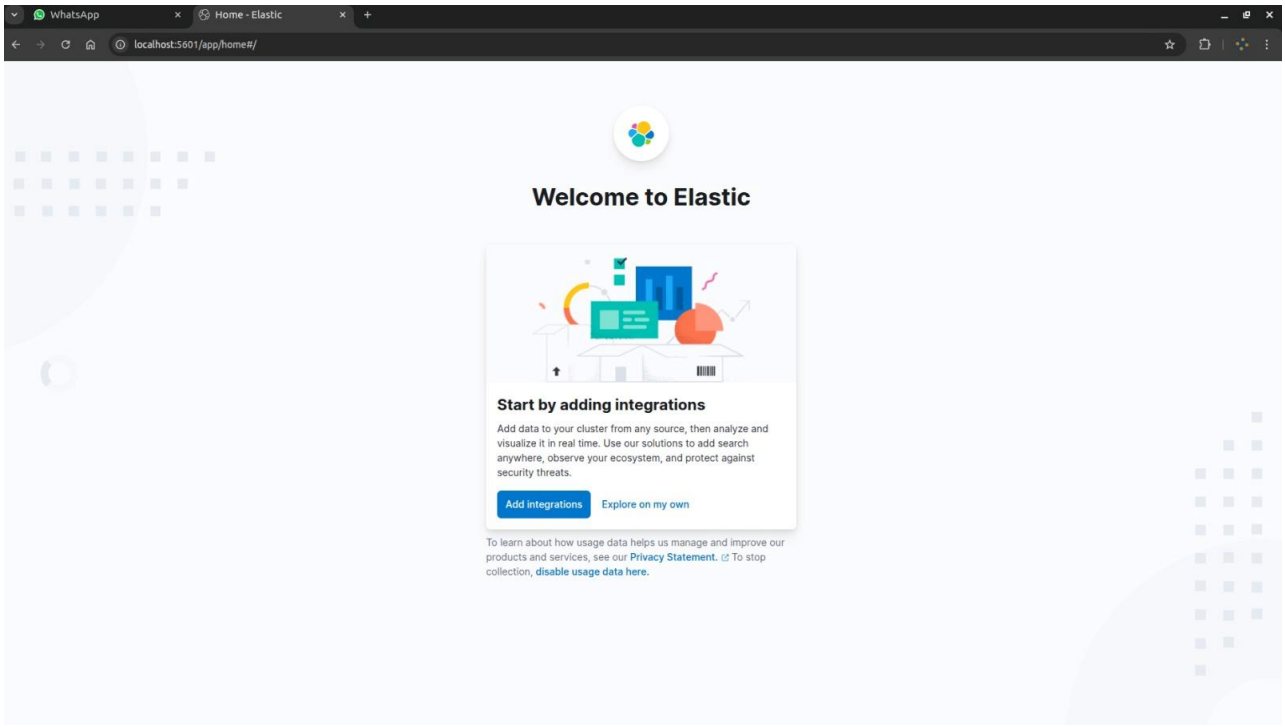
```
sudo docker ps
```

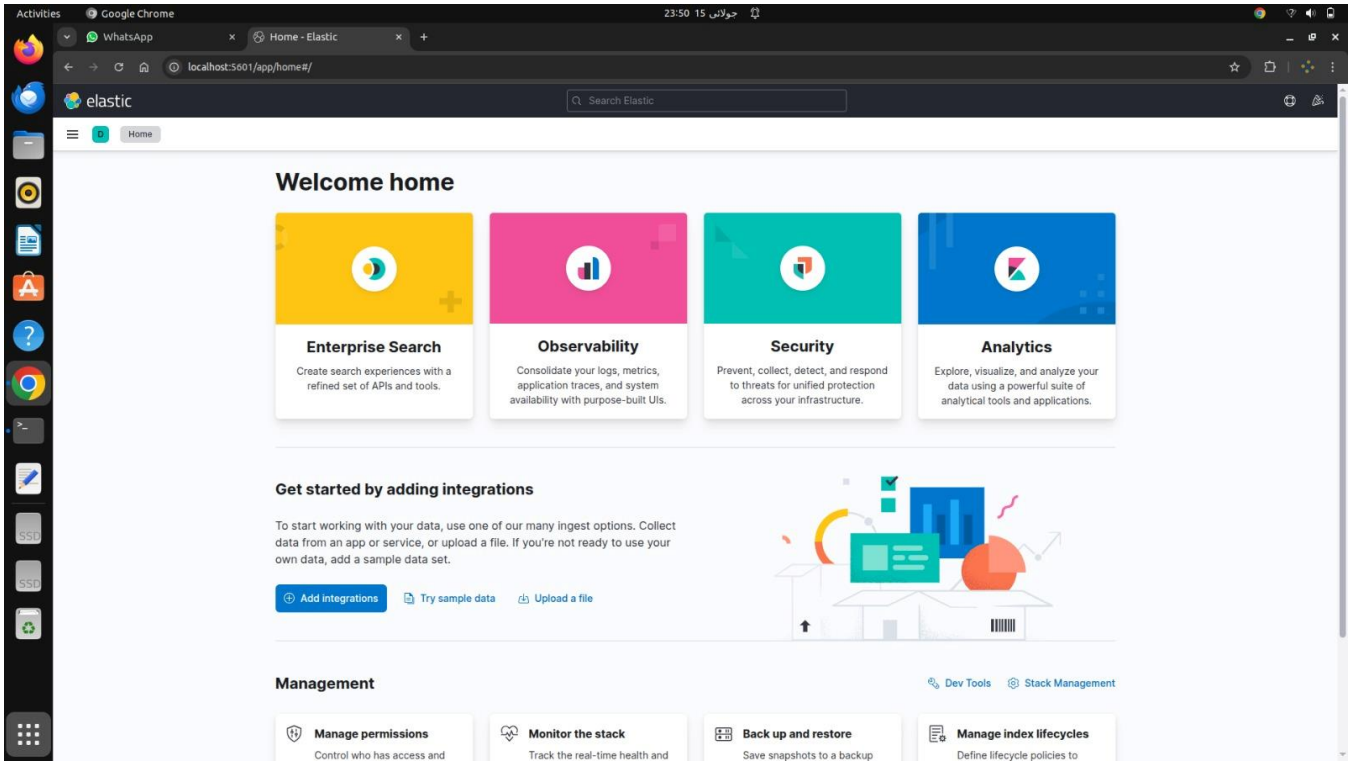
- Wait for 1–2 minutes to let all services start properly then open your browser and go to:

```
http://localhost:5601
```

You should see the Kibana UI if everything is running correctly.

You should be able to see the following screens:





Now that we have successfully launched ELK, we have to connect filebeat with logstash ti start collecting logs from filebeat.

Filebeat Setup:

- First, download all packages of filebeat to run it successfully. Run the following commands

```
curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-8.14.0-amd64.deb
sudo dpkg -i filebeat-8.14.0-amd64.deb
sudo apt-get install -f
filebeat version
```

- After checking if filebeat has been downloaded successfully, Edit the Filebeat config file:

```
sudo nano /etc/filebeat/filebeat.yml
```

- Update the output section:

```
# Comment out Elasticsearch output
#output.elasticsearch:
#  hosts: ["localhost:9200"]
# Enable Logstash output
output.logstash:
  hosts: ["localhost:5044"]
```

- Enable desired log inputs (e.g., system logs):

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/log/syslog
    - /var/log/auth.log
```

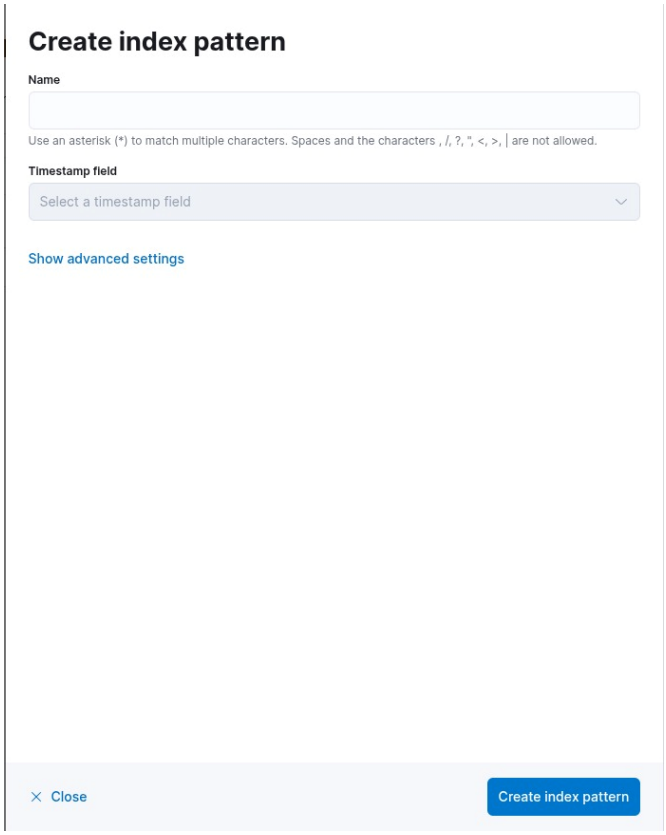
- Save and close the file.
- Restart Logstash container:

```
docker restart <logstash-container-name>
```

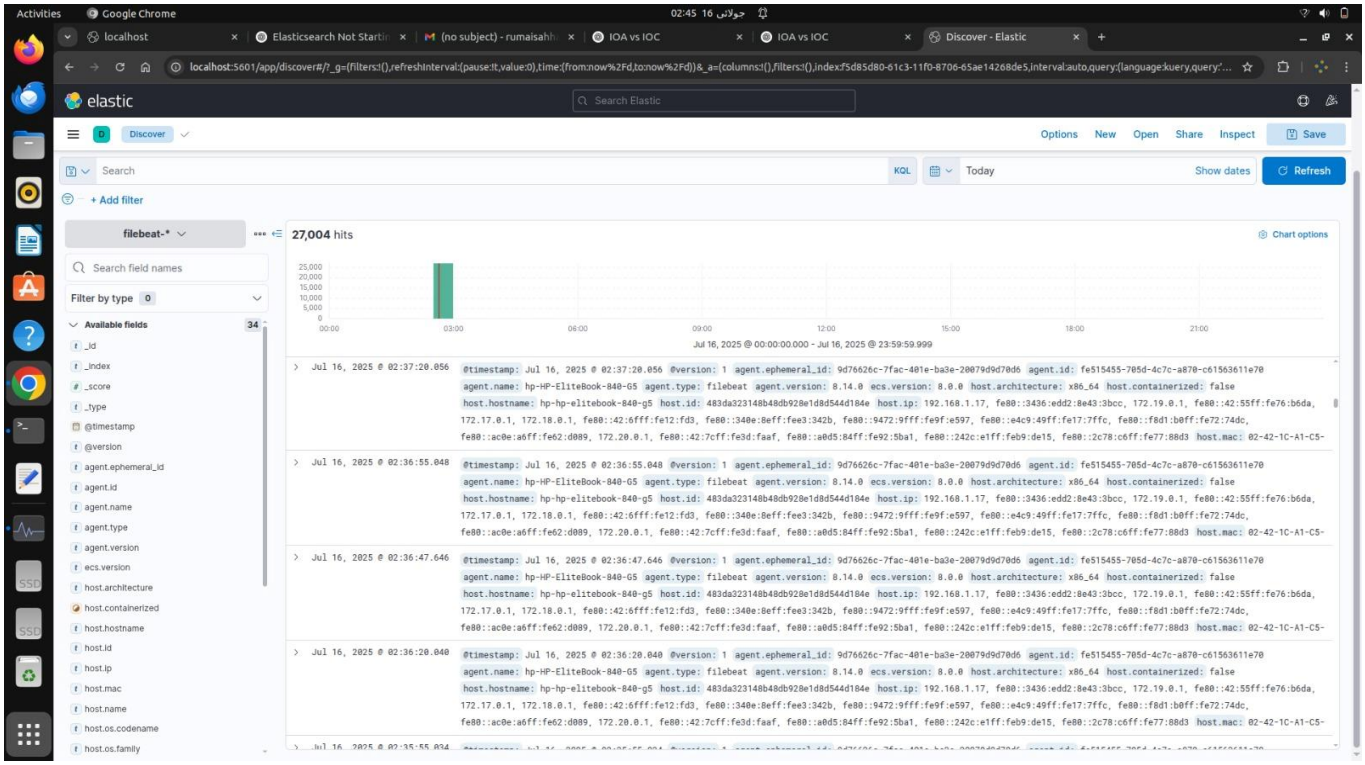
- Start Filebeat:

```
sudo systemctl start filebeat
```

- Go to Kibana: http://localhost:5601 and navigate to **Discover**. Then set the index pattern to: **filebeat-***. You should be able to see this:



- You should see system logs arriving after clicking **CREATE INDEX PATTERN** in such way:



Conclusion

This project allowed me to explore and implement a centralized log monitoring system using the ELK Stack along with Filebeat, all containerized through Docker and managed on a Linux host. The goal was to understand how system logs could be collected, parsed, stored, and visualized in a streamlined and scalable manner.

Throughout this process, each component of the stack contributed its unique function: Filebeat collected logs from sources like /var/log/syslog and /var/log/auth.log; Logstash parsed and forwarded the data; Elasticsearch indexed and stored it; and Kibana made it all visible through dashboards and graphs. The use of Docker simplified deployment, while systemd was instrumental in managing services like Filebeat on the host system.

Challenges

However, this project also presented several challenges:

- Installing and running Filebeat initially failed multiple times due to missing dependencies, incorrect configurations, and service startup issues.
- At one point, Filebeat reported “no outputs are defined,” which required revisiting the filebeat.yml file to correctly configure the output section for Logstash.
- I also faced issues with Docker container persistence—understanding the difference between stopping Docker and keeping it from restarting after a system reboot took trial and error.
- Testing and verifying the data flow between Filebeat and Logstash required careful analysis of logstash.conf and use of tools like curl and docker ps/logs.
- Visualizing data in Kibana was confusing initially, especially when it required creating an index pattern and understanding what data was actually being stored in Elasticsearch.

Despite these challenges, I was able to successfully complete the pipeline and visualize logs from my host system in Kibana. These hurdles helped me better understand the nuances of system logging, container orchestration, and pipeline debugging.

In future iterations, I would like to expand the system to collect more types of logs such as firewall data, user activity reports, and network traffic logs, creating a more comprehensive security monitoring environment.

Future Work

While this project successfully demonstrates centralized log monitoring using the ELK Stack with Filebeat, there are several important enhancements and expansions planned for the next phase of this project.

One key goal is to begin monitoring network activity on the Ubuntu host and visualizing that data through ELK. Currently, Filebeat is configured to collect log files like /var/log/syslog and /var/log/auth.log, which capture general system events and authentication events. However, for full network visibility, additional tools and configurations will be required.

Here are the next steps I plan to take:

- **Integrate Packetbeat:**
Packetbeat is another lightweight data shipper provided by Elastic, designed specifically to capture and ship network traffic. It acts as a network packet analyzer and can report on protocols like DNS, HTTP, MySQL, and more. By installing and configuring Packetbeat on my system, I will be able to collect real-time network data such as requests, responses, latencies, and failures.

- Configure Packetbeat Output to Logstash:
Just like Filebeat, Packetbeat will be set to forward its data to Logstash, where I can apply filters or enrich the data before sending it to Elasticsearch.

- Create Network Dashboards in Kibana:
Once network data is indexed in Elasticsearch, I will build custom dashboards in Kibana to monitor bandwidth usage, top IPs communicating with the system, protocol distribution, and possible suspicious activity.