s

(PROJECT TITLE / RESEARCH TITLE)

Submitted by

Richa Kumari

For the award of the degree of

B.tech

Under the supervision of

Supervisior
(Name, Designation, Organization)

Faculty of Mathematics and Computing
Banasthali Vidyapith

Banasthali - 304022

Session: 2024

(It must be printed on respective Organization's Letter Head)

CERTIFICATE FORMAT

Certified that **"Name of the Student"** has carried out the project work titled **"Title of the Project"** from _____ (**date**) to _____ (**date**) for the award of the _____ **(Course Name)** from _____ (**Organization)** under my supervision.

Abstract

This project presents a real-time log monitoring system designed to enhance log management and analysis by filtering, processing, and visualizing logs. The system captures logs categorized as trace, warning, and info, and utilizes a filtering mechanism to exclude info logs from the input file. The filtered logs are then sent to Filebeat, which forwards them to Logstash for further processing and storage in Elasticsearch indices. The logs stored in Elasticsearch are visualized using Kibana dashboards, providing a comprehensive and intuitive interface for monitoring and analyzing log data.

The primary objective of this project is to streamline log monitoring by reducing the volume of less critical logs and focusing on more significant trace and warning logs. This approach improves the efficiency of log analysis, enabling quicker identification and resolution of potential issues.

Filebeat serves as the lightweight shipper, collecting logs from the designated input sources. Logstash then processes these logs, applying various transformations and enriching the data before indexing it into Elasticsearch. This robust pipeline ensures that only pertinent log data is stored, optimizing storage usage and search performance.

Kibana, the visualization tool, offers dynamic and customizable dashboards that allow users to monitor system performance, detect anomalies, and conduct in-depth analysis of log events. The integration of these tools provides a powerful solution for real-time log monitoring, enhancing operational visibility and decision-making capabilities.

This system was developed at 8 Bit Systems under the guidance of Senior DevOps Engineer Ankur Sharma. The project showcases a successful implementation of the ELK stack (Elasticsearch, Logstash, Kibana) with

Filebeat, demonstrating its effectiveness in real-time log monitoring and analysis in a professional DevOps environment.

Acknowledgements

I am profoundly grateful to Senior DevOps Engineer Ankur Sharma, whose invaluable guidance and unwavering support were pivotal throughout the development of this project. His deep expertise and insightful feedback significantly contributed to the successful implementation of this real-time log monitoring system. I am deeply appreciative of the time and effort he invested in mentoring me and providing direction at crucial junctures.

I would also like to extend my heartfelt thanks to the entire team at 8 Bit Systems. Their collaboration and assistance were crucial in overcoming various challenges during the project's development. The collective effort and team spirit exhibited by everyone involved were instrumental in bringing this project to fruition. I am thankful for their encouragement, support, and the shared knowledge that made this endeavor a success.

Table of Contents

o Database Design

o Data flow diagrams/Activity Diagrams

o Flowcharts/Sequence Diagrams

Coding

Only main Modules (if available)

(Detailed Coding is not required)

Testing

o Test cases

User Interfaces

Appendices

References

Requirement Analysis

Requirement Specification

The project involves developing a real-time log monitoring system capable of processing and visualizing logs categorized as trace, warning, and info. The system filters out info logs, processes the remaining logs, and visualizes them on a Kibana dashboard.

1. Software Requirements

1. Jenkins:

- Version: Latest LTS version.

- Plugins: Docker Pipeline, Git, Pipeline, Ansible, etc.

2. Docker:

- Version: Latest stable release.

- Containers: Elasticsearch, Logstash, Kibana.

3. Ansible:

- Version: Latest stable release.

- Playbooks: For ELK setup and Logstash configuration.

4. ELK Stack:

- Elasticsearch: Latest version compatible with the rest of the stack.

- Logstash: Latest version.

- Kibana: Latest version.

5. Operating System:

- Linux: Preferred (e.g., Ubuntu, CentOS).

- Windows: Supported but less common.

6. Version Control System:

- Git: For managing playbooks and Jenkins pipeline scripts.

2. Hardware Requirements

The hardware requirements depend on the scale of your deployment and data volume. Here are general guidelines:

1. Minimal Setup:

CPU: 4 cores

RAM: 8 GB

-Storage: 50 GB SSD

- Network: 1 Gbps

2. Medium Scale Setup**:

- CPU: 8 cores
- RAM: 16-32 GB
- Storage: 200 GB SSD (with provision for scaling)
- Network: 1 Gbps or higher

3. Large Scale Setup**:

- CPU: 16+ cores
- RAM: 64+ GB
- Storage: 500 GB+ SSD, with additional storage for backup and archiving
- Network: High-speed network, 10 Gbps or higher for distributed setups

3. Feasibility Study

Feasibility Study for Real-Time Log Monitoring System

1. Operational Feasibility

Operational feasibility assesses the practicality of the system in terms of user interaction, operational workflow, and maintenance.

User Interaction:

- Ease of Use: The system leverages Kibana for visualizing log data, providing an intuitive and user-friendly interface. Users can easily navigate dashboards, set up visualizations, and create alerts.

- Training Requirements: Minimal training is required for users to interact with Kibana dashboards. Basic tutorials and documentation will suffice to get users up to speed.

Operational Workflow:

- Log Collection: Filebeat efficiently collects logs from input files and forwards them to Logstash. This automated process minimizes manual intervention.

- Log Filtering and Processing: Logstash processes and filters logs based on predefined rules, ensuring only relevant logs are stored in Elasticsearch. This streamlines log management and reduces storage overhead.

- Real-time Monitoring: The system provides real-time log monitoring, enabling timely detection and response to issues. This enhances operational efficiency and reduces downtime.

Maintenance:

- Automation with Jenkins: Using Jenkins for automating ELK container deployment and management ensures consistent and repeatable processes, reducing the likelihood of human errors. Automated scripts can handle routine tasks such as updates, backups, and scaling.

- Scalability: The system can be scaled horizontally by adding more nodes to the Elasticsearch cluster, ensuring it can handle increasing log volumes without performance degradation.

2. Technical Feasibility

Technical feasibility evaluates the technical resources required to implement the system and whether they are available and adequate.

Technology Stack:

- ELK Stack: The system uses well-established and widely supported open-source tools—Elasticsearch, Logstash, and Kibana—along with

Filebeat. These tools are designed to work seamlessly together, ensuring a robust and reliable log management solution.

- Containerization: Using Docker for containerizing the ELK stack components ensures consistency across different environments and simplifies deployment and scaling.

Infrastructure Requirements:

- Servers: Adequate server resources are required to run the ELK stack components. This includes sufficient CPU, memory, and storage to handle log ingestion, processing, and querying.

- Network: Reliable network connectivity is crucial for seamless communication between Filebeat, Logstash, Elasticsearch, and Kibana.

Integration:

- Compatibility: The ELK stack supports integration with various log sources and can be customized to fit specific log formats. This ensures compatibility with existing infrastructure.

- Automation with Jenkins: Jenkins scripts and Ansible playbooks can automate the deployment and configuration of ELK stack components, reducing setup time and ensuring consistent configurations.

Technical Expertise:

- Skill Set: The project requires familiarity with the ELK stack, Docker, Jenkins, and scripting languages such as Bash or Python. Adequate documentation and community support are available for these technologies, facilitating the learning process for team members.

3. Economic Feasibility

Economic feasibility assesses the cost-effectiveness of the project, including initial setup costs and ongoing operational expenses.

Initial Setup Costs:

- Hardware: Initial costs involve procuring servers and storage solutions capable of handling the anticipated log volumes. Using existing infrastructure can reduce these costs.

- Software: The ELK stack and associated tools (Filebeat, Docker, Jenkins) are open-source and free to use, significantly lowering software costs.

Operational Costs:

- Maintenance: Regular maintenance tasks such as updating software, monitoring system performance, and managing storage can be automated using Jenkins and Ansible, reducing labor costs.

- Scalability: The system's scalability ensures that additional resources can be added incrementally, spreading out costs over time rather than requiring a large upfront investment.

Cost-Benefit Analysis:

- Efficiency Gains: By automating log collection, processing, and visualization, the system reduces manual effort and improves response times to operational issues, leading to cost savings in labor and downtime.

- Improved Decision Making: Real-time insights into system performance and potential issues enable proactive management, reducing the risk of costly outages and improving overall system reliability.

Return on Investment (ROI):

- Short-term ROI: Immediate benefits include streamlined log management, enhanced monitoring capabilities, and reduced manual intervention.

- Long-term ROI: Over time, the system's ability to scale and adapt to growing log volumes without significant additional costs ensures sustained value and cost savings.

Conclusion

The real-time log monitoring system is operationally, technically, and economically feasible. It leverages the robust ELK stack for log management, uses automation tools like Jenkins for efficient deployment and maintenance, and provides significant cost savings through open-source software and reduced manual effort. The system is designed to scale with growing log volumes, ensuring long-term viability and return on investment.

4. Product Functions

1. Jenkins Automation:

- Pipeline Script: Automate the deployment using Jenkins Pipeline (Declarative or Scripted).

- Continuous Integration/Continuous Deployment (CI/CD): Integrate with version control for automated builds and deployments.

2. Docker Containers:

- Containerization: Use Docker to containerize Elasticsearch, Logstash, and Kibana.

- Networking: Ensure proper networking between the containers.

3. Ansible Playbooks:

- ELK Deployment: Ansible playbooks to deploy Elasticsearch, Logstash, and Kibana.

- Logstash Configuration: Playbooks to configure Logstash filters and pipelines.

4. ELK Stack:

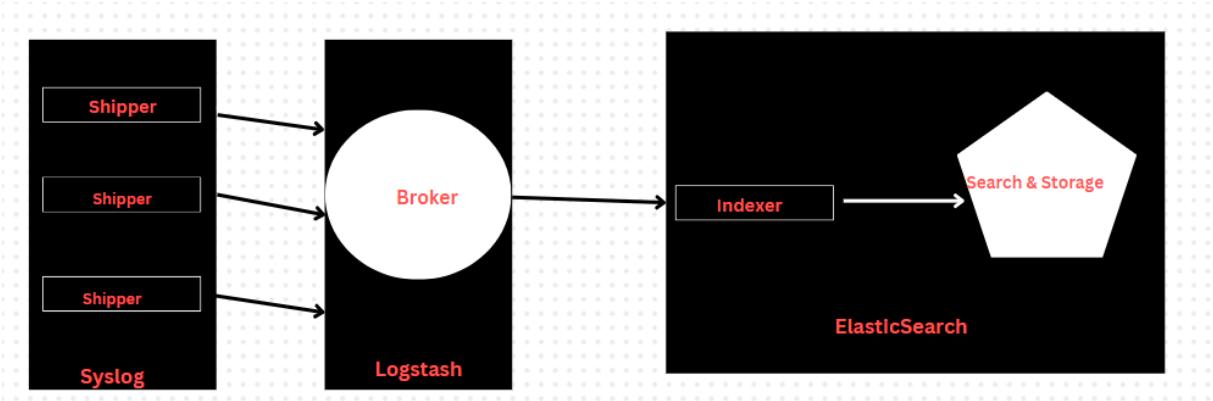
- Elasticsearch: Data indexing and search capabilities.

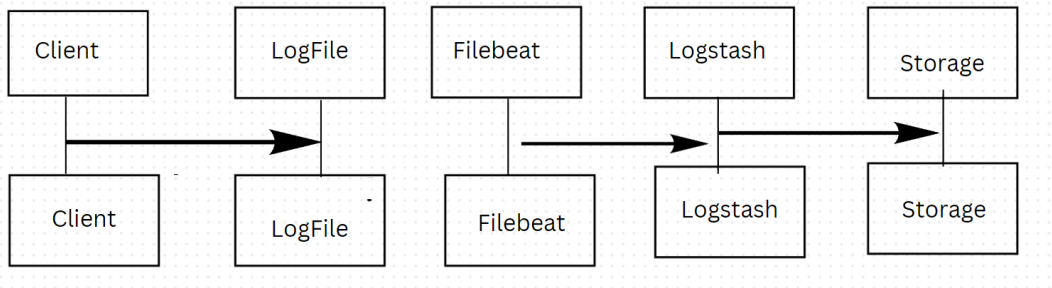
- Logstash: Data collection, transformation, and storage.

- Kibana: Data visualization and exploration.

Automating the ELK stack deployment using Jenkins, Docker, and Ansible ensures a robust and scalable log management solution. By following the outlined software and hardware requirements, feasibility study, and detailed product functions, you can efficiently manage and analyze large volumes of log data.

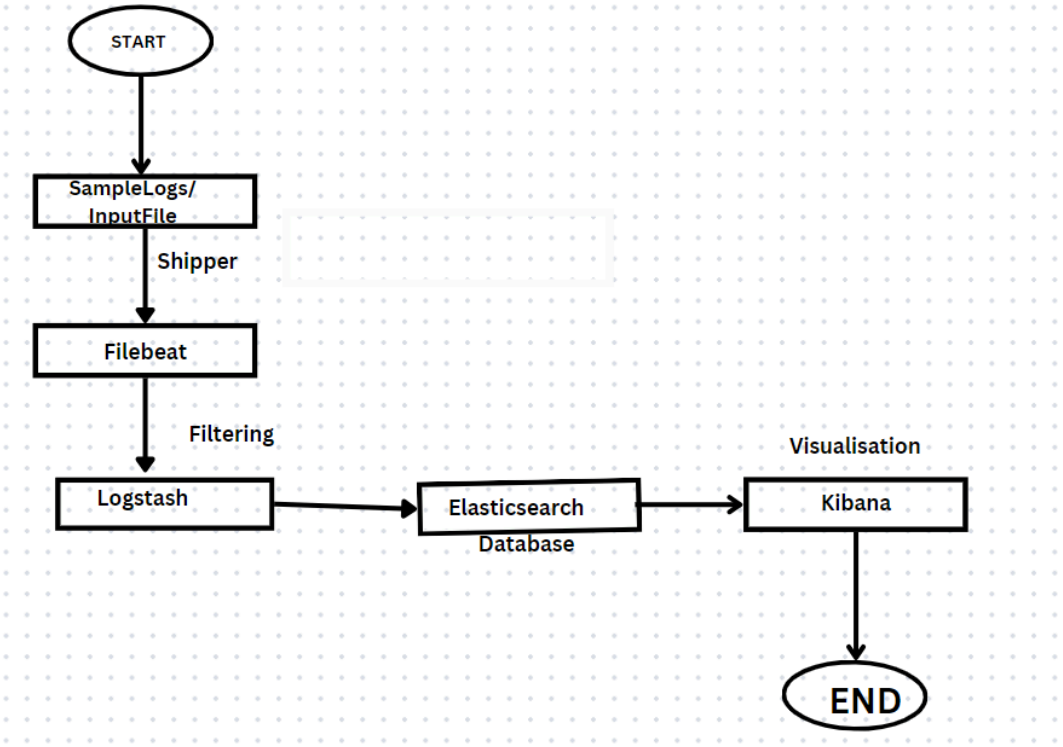
High level Diagram





Sequence diagram

FlowChart



Coding

Running Elastic Container

elastic.yml

```yaml
---
- name: Deploy Elasticsearch with Docker
  hosts: localhost
  become: true
  become_user: richa
  # become_method: sudo
  gather_facts: false
  roles:
    - elastic-docker
```

includes tasks -> main.yml

roles > elastic-docker > tasks > main.yml

```yaml
---
- name: Create Docker network
  docker_network:
    name: my-network

- name: ensure config dirctory present
  file:
    path: "/home/richa/ansible/elasticdata/elasticconfig"
    state: directory
    owner: 1000
    group: 1000
    mode: 0775
    force: yes

- name: ensure template file is present
  template:
    src: elasticsearch.yml
    dest: /home/richa/ansible/elasticdata/elasticconfig/elasticsearch.yml
    owner: 1000
    group: 1000
    mode: 0775
    force: yes

- name: pull elasticsearch docker image
  docker_image:
    name:  docker.elastic.co/elasticsearch/elasticsearch:7.17.0
    source: pull

```

roles > elastic-docker > tasks > main.yml

```yaml

- name: Install Elasticsearch using Docker
  docker_container:
    name: elasticsearch
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.0
    state: started
    ports:
      - "9200:9200"
        #env:
        #discovery.type: single-node
    volumes:
        - /home/richa/ansible/elasticdata/elasticconfig/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearc
    networks:
        - name: my-network

- name: wait for cluster to get ready
  uri:
    url: http://localhost:9200/_cluster/health?pretty
    method: GET
    status_code: [200]
    # user: elasticuser
    # passwrod: elasticpassword
    validate_certs: false
  register: result
  until: result.status == 200
  retries: 20
  delay: 10

```

Running filebeat container

myfilebeat.yml

```yaml
---
- name: Deploy Filebeat using Docker
  hosts: localhost
  become: true
  become_user: richa
  tasks:
    - name: Include docker-filebeat role
      include_role:
        name: docker-filebeat

```

Running logstash container

elastic.yml | mylogstash.yml

mylogstash.yml

```yaml
---
- name: Deploy logstash using Docker
  hosts: localhost
  become: true
  become_user: richa
  tasks:
    - name: Include logstash role
      include_role:
        name: docker-logstash
```

Running Kibana Container

mykibana.yml

```yaml
---
- name: Install and Configure Kibana with Docker
  hosts: localhost
  become: true
  become_user: richa
  gather_facts: false
  roles:
    - docker-kibana
```

Jenkins file

jenkinsfile

```
pipeline {
    agent any

    parameters {
        choice(name: 'CHOICE', choices: ['elasticsearch', 'kibana', 'filebeat', 'logstash', 'quit'], description: 'Se
    }

    stages {
        stage('Give Permission') {
            steps {
                // Give execution permission to execute_playbooks script
                sh 'chmod +x /home/richa/Ansible1/execute_playbooks.sh'
            }
        }
```

```
$ execute_playbooks.sh
  1  /bin/bash
  2
  3  choice="$1"
  4  case $choice in
  5          elasticsearch)
  6                  echo "elastic search runnng"
  7                  ansible-playbook ./Ansible1/elastic.yml
  8                  ;;
  9          kibana) echo "you select kibana"
 10                  ansible-playbook ./Ansible1/mykibana.yml
 11                  ;;
 12          filebeat) echo "you select filebeat"
 13                  ansible-playbook ./Ansible1/myfilebeat.yml
 14                  ;;
 15          logstash) echo "you select logstash"
 16                  ansible-playbook ./Ansible1/mylogstash.yml
 17                  ;;
 18          quit) echo "Quitting" exit
 19                  ;;
 20          *) echo "default selection";;
 21  esac
 22
```

Testing

Testing and Test Cases for Real-Time Log Monitoring System

Introduction to Testing

Testing a real-time log monitoring system involves verifying that each component (Filebeat, Logstash, and Elasticsearch) is correctly configured and functions as expected. The testing process ensures that logs are properly collected, filtered, processed, and stored in Elasticsearch indices, and that they can be visualized in Kibana. The following sections detail the testing strategy and specific test cases to validate the system.

Testing Strategy

1. Unit Testing: Verify individual components such as log filtering in Logstash and log shipping in Filebeat.

2. Integration Testing: Ensure seamless data flow between Filebeat, Logstash, and Elasticsearch.

3. System Testing: Validate the end-to-end process from log collection to visualization in Kibana.

4. Performance Testing: Assess the system's ability to handle high volumes of log data in real-time.

5. User Acceptance Testing (UAT): Confirm that the system meets user requirements and expectations.

Test Cases

1. Unit Testing

Test Case 1: Filebeat Log Collection

- Objective: Verify that Filebeat correctly collects logs from the specified input file.

- Preconditions: Input log file exists with sample log entries.

- Steps:

  1. Configure Filebeat to read from the input log file.

  2. Start the Filebeat service.

  3. Verify Filebeat log output to ensure it reads the input file.

- Expected Result: Filebeat successfully reads and forwards logs to Logstash.

Test Case 2: Logstash Filtering

- Objective: Ensure Logstash correctly filters out info logs.

- Preconditions: Logstash configuration file includes filter rules to drop info logs.

- Steps:

  1. Send sample logs with trace, warning, and info levels to Logstash.

  2. Start the Logstash service.

  3. Verify that Logstash only forwards trace and warning logs to Elasticsearch.

- Expected Result: Info logs are dropped, and only trace and warning logs are forwarded to Elasticsearch.

2. Integration Testing

Test Case 3: Filebeat to Logstash Integration**

- Objective: Validate that logs are correctly shipped from Filebeat to Logstash.

- Preconditions: Both Filebeat and Logstash services are running.

- Steps:

  1. Generate sample log entries in the input log file.

  2. Verify Filebeat log output to ensure it forwards logs to Logstash.

  3. Check Logstash logs for received log entries.

- Expected Result: Logs are successfully shipped from Filebeat to Logstash without loss.

Test Case 4: Logstash to Elasticsearch Integration

- Objective: Ensure that logs are correctly processed and stored in Elasticsearch.

- Preconditions: Logstash is configured to forward logs to Elasticsearch.

- Steps:

  1. Send sample logs to Logstash.

  2. Verify that Logstash processes and forwards logs to Elasticsearch.

  3. Query Elasticsearch indices to check for the presence of the logs.

- Expected Result: Logs are correctly stored in Elasticsearch indices.

3. System Testing

Test Case 5: End-to-End Log Flow

- Objective: Validate the complete log flow from input file to Kibana visualization.

- Preconditions: All components (Filebeat, Logstash, Elasticsearch, Kibana) are running and configured.

- Steps:

1. Generate sample log entries in the input log file.
2. Verify that Filebeat collects and forwards logs to Logstash.
3. Check Logstash logs for received and processed logs.
4. Query Elasticsearch indices for stored logs.
5. Visualize the logs in Kibana dashboard.

- Expected Result: Logs are successfully collected, filtered, processed, stored, and visualized.

4. Performance Testing

Test Case 6: High Volume Log Handling

- Objective: Assess the system's performance under high log volume.

- Preconditions: Log generation script to simulate high log volume.

- Steps:

1. Run the log generation script to produce a high volume of logs.
2. Monitor the performance of Filebeat, Logstash, and Elasticsearch.
3. Verify that the system handles the log volume without significant delays or loss.

- Expected Result: The system processes high volumes of logs efficiently.

5. User Acceptance Testing (UAT)

Test Case 7: User Validation of Kibana Dashboards

- Objective: Ensure that the Kibana dashboards meet user requirements.

- Preconditions: Kibana dashboards are configured and accessible.

- Steps:

1. Have users interact with the Kibana dashboards.
2. Collect feedback on dashboard usability and data accuracy.
3. Make adjustments based on user feedback if necessary.

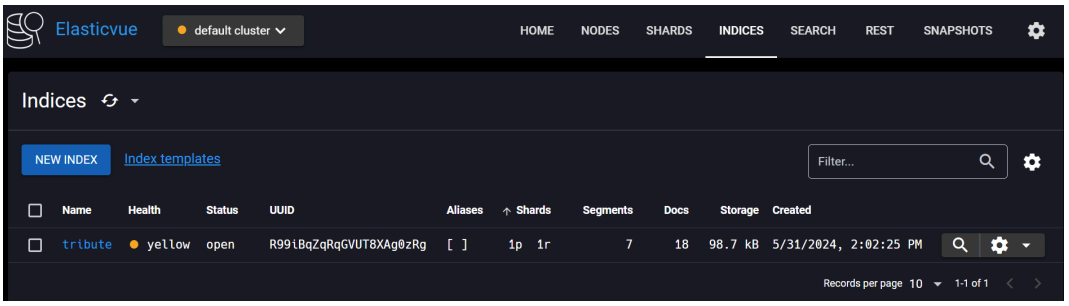
- Expected Result: Users are satisfied with the Kibana dashboards, and they find the data accurate and useful.

Conclusion

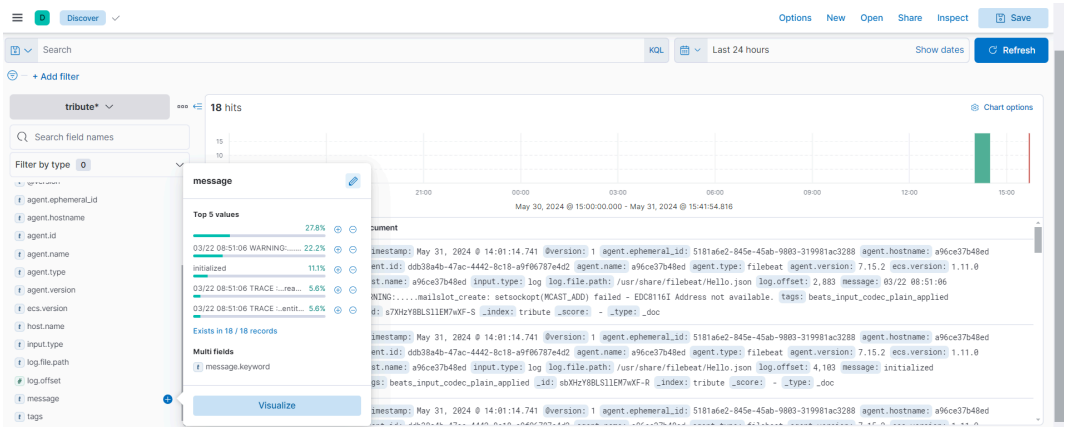
By following these detailed test cases, you can ensure that your real-time log monitoring system is robust, efficient, and meets the requirements. Regular testing and validation are crucial for maintaining the reliability and performance of the system in a production environment.

User Interface

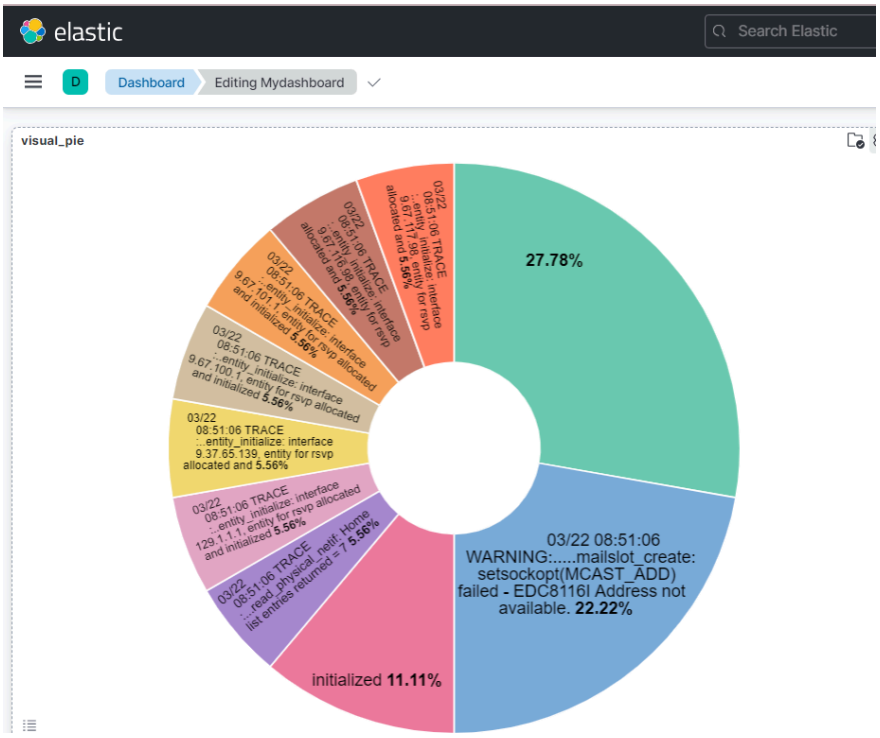
Elastic Index



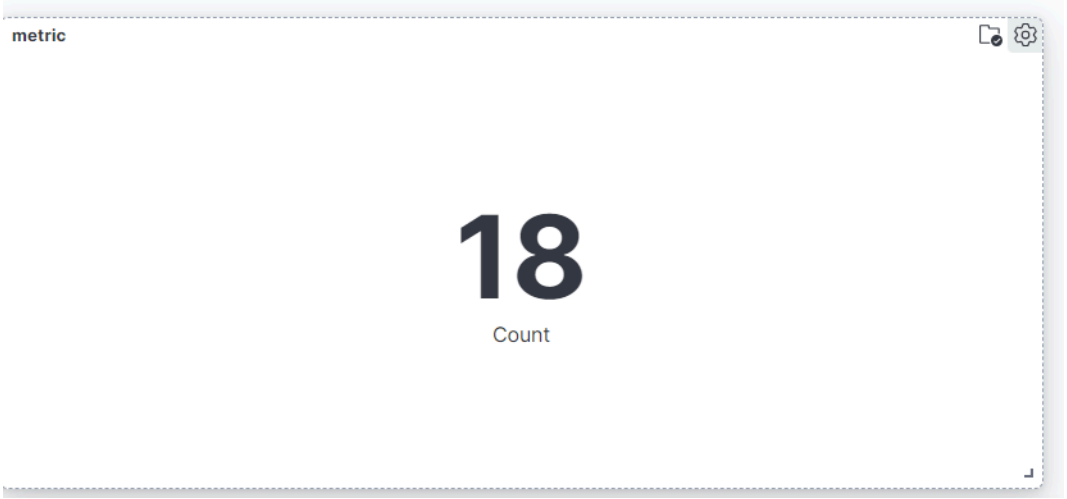
Index pattern consists of logs

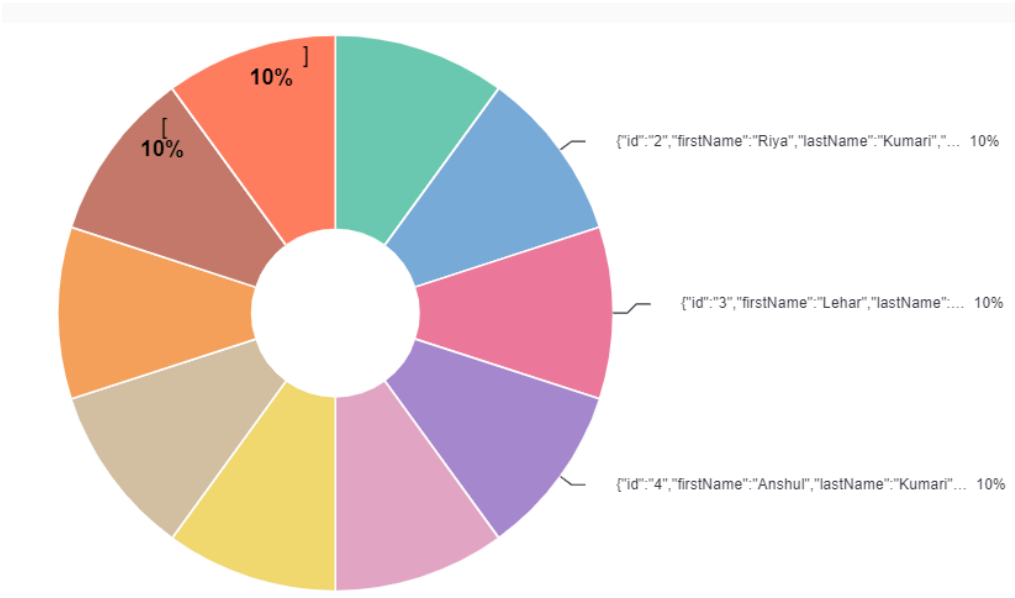


Kibana Dashboard



Total count of Logs




]
10%
[
10%
{"id":"2","firstName":"Riya","lastName":"Kumari","... 10%
{"id":"3","firstName":"Lehar","lastName":... 10%
{"id":"4","firstName":"Anshul","lastName":"Kumari"... 10%