

# **Secure and Scalable Log Monitoring System using Docker & Splunk 3and Jenkins deployment**

A Project report

Submitted in partial fulfilment of the requirements for the award of a degree of Bachelor of  
Technology

(Computer Science and Engineering)

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB



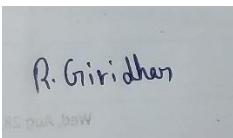
**L**OVELY  
**P**ROFESSIONAL  
**U**NIVERSITY

Submitted by

Name of the Student

Ruppa Giridhar

Registration Number: 12207968



**Signature of Student**

Name of the Supervisor

Divya Thakur

**Signature of Supervisor**

**DECLARATION BY STUDENT**

We To whom so ever it may concern.

I Ruppa Giridhar (12207968), hereby declare that the work done by me on “Secure and Scalable Log Monitoring System using Docker & Splunk and Jenkins deployment” under the supervision of (Divya Thakur-Assistant Professor), Lovely professional University, Phagwara, Punjab, is a record of original work for the partial fulfilment of the requirements for the award of the degree, in Computer Science Engineering from Lovely Professional University, Phagwara.

**Project Group Number:**

Name of Student 1: Ruppa Giridhar

Registration Number: 12207968

---

Signature of Student 1  
Date: 03-05-2

---

## ACKNOWLEDGEMENT

Heartfelt thanks are due to those who supported and guided me in many ways throughout the project.

I would like to express my deepest gratitude to my guide and mentor, **Ms. Divya Thakur**, whose encouragement, suggestions, and support were constant throughout the project. Her guidance was instrumental in defining the course of this work and helped transform some raw ideas into a practically viable and implementable solution. I will always appreciate having the opportunity to carry out the project under her guidance.

I also appreciate the constructive feedback, collaboration, and motivation from my peers and friends. Their contributions created such an energetic and bright atmosphere while moving the project, making the process enriching and fulfilling.

Finally, my gratitude goes to my university for providing this opportunity and resources to undertake this project to give me an avenue to apply theoretical knowledge in addressing real-world problems.

**Ruppa Giridhar**  
Bachelor of Technology  
(Computer Science and Engineering)

---

## LIST OF CONTENT

S. No.	Section Title	Description
1	<b>Abstract</b>	Brief overview of the project's purpose, goals, and outcomes
2	<b>Acknowledgement</b>	Gratitude to mentors, friends, and contributors
3	<b>Introduction</b>	Project background, objectives, and scope
4	<b>Problem Identification</b>	Challenges in traditional log monitoring systems
5	<b>Proposed Solution</b>	Overview of the system, how it solves the identified problems
6	<b>System Architecture</b>	Architecture diagram and explanation of key components
7	<b>Implementation Details</b>	Step-by-step details on tools, setup, and integration
8	<b>Data Flow &amp; Log Collection</b>	Description of how logs are collected, processed, and stored
9	<b>Alerting Mechanism</b>	Setup of email alerts in Splunk (e.g., wrong password scenario)
10	<b>Security Measures</b>	Security features including RBAC and secure communications
11	<b>Scalability and Performance</b>	How the system can scale, performance tuning
12	<b>Testing &amp; Results</b>	Functional testing reports and log sample screenshots
13	<b>Final Thoughts &amp; Conclusion</b>	Summary, learning, challenges, and future enhancements
14	<b>References</b>	Books, tools, articles, and other sources used
15	<b>Appendix</b>	Code snippets, Jenkins file, Docker Compose, config files

## **ABSTRACT:**

Real-time log monitoring is indispensable in current digital infrastructures concerning security, reliability, and performance. Conventional log management solutions usually suffer from scalability, real-time analysis, and centralized monitoring, especially under distributed environments. This project presents a secure and scalable log-monitoring system built on Docker and Splunk to address these issues.

This system implements containerized deployment through Docker to guarantee platform independence, ease of deployment, and consistent environment replication. Splunk Enterprise is primarily used for log collection, indexing, and visualization, while Splunk Universal Forwarders residing in remote systems provide assured reliable ingestion of logs from various sources. CI/CD automation with Jenkins and GitHub provides enhanced support for efficient and repeatable deployments.

Security features include role-based access control (RBAC), encrypted communication between components, and anomaly detection strategies. Emailing mechanisms for alerting were also integrated into the system for real-time notifications, e.g., detection and alerting when a wrong password is entered in authentication-based services.

The result is a flexible, efficient, and secure log-monitoring framework that can scale from a small to an enterprise-level application. The project enhances visibility into the behavior of systems and improves incident response and operational transparency.

## **INTRODUCTION:**

The Today's rapidly changing digital ecosystem, where data security and operational efficiency drive concerns, has made it extremely difficult for organizations to identify and manage the amount of log data produced by their systems. Logs serve as a vital source of information revealing system health, performance metrics, user activities, security breaches, and potential system vulnerabilities. It is no longer a best practice but a necessity for organizations aiming to maintain strong cybersecurity posture and comply with regulations.

Many traditional log monitoring systems struggle with challenges such as centralization, scalability, real-time processing, and security across a multitude of platforms or infrastructures. Increasing numbers of endpoints and systems can result in inefficiencies in these legacy solutions, as well as error proneness, high costs, and intensely burdensome maintenance requirements for such systems. Moreover, manual monitoring can result in longer times between incident detection and response, meaning that an attack can potentially cause damage before it is detected.

To elaborate on these inadequacies, this work proposes the Secure and Scalable Log Monitoring System, working with Docker and Splunk. The objective of this project is real-time collection, analysis, and visualization of logs from multiple sources using a containerized setup. Docker encapsulates Splunk components in easily deployable and portable containers. While Splunk Enterprise is more for indexing, searching, and visualizing log data, Splunk Universal Forwarder is used for efficient and secure log forwarding from any remote machines.

Furthermore, CI/CD automation is implemented through Jenkins and GitHub for streamlining deployments and updates, enabling the infrastructure to dynamically accommodate the organizational needs. The security features are reinforced by encrypted communications, role-based access controls, and alert mechanisms (for example, email alerts on failed login attempts).

Merged with the power of monitoring tools, the modern-day DevOps will give IT administrators and security teams an overarching view of the system behavior, fast detection and response to any threat, and constant availability and performance of their services.

## Objective of the Project

The goal of the project "Secure and Scalable Log Monitoring System Using Docker and Splunk" is to design and implement a well-defined, strong, real-time, and secure infrastructure for monitoring logs from distributed systems. In this day and age, organizations produce massive amounts of log data from their applications, services, cloud platforms, and security devices for consideration in the context of current-day cybersecurity. However, the secure and efficient collection, management, and analysis of these logs surmount challenges that are quite significant in their magnitude. This project will accommodate the proceedings by developing an automated, containerized logging framework that would scale up, be robust, and be easy to deploy.

The project is based on modern technology which uses Docker for containerization, Splunk Enterprise for log collection and analysis, Splunk Universal Forwarder for remote log transmission, and Jenkins for CI/CD-based automation. It also comes with alerting mechanisms in the form of email alerts from the system to inform the administrator of suspicious or anomalous behavior, such as unauthorized login attempts, failed authentications, or service failures, in a proactive manner.

Derived key objectives are:

- To design a Docker-based logging infrastructure that can be deployed in cloud environments or on-premises sites.
- To enable secure log transport across remote systems, e.g. Kali Linux VMs, with Splunk Universal Forwarders.
- To automate deployment and update implementation with Jenkins pipelines to support consistency, reduce manual errors, and save time.
- To develop user-defined dashboards and reports in Splunk that visualise and provide insights into critical security events and key metrics of performance.
- To set up email alerts for instant notification to the security teams about critical issues like unauthorized login attempts or failed services.
- To make a system scalable and modular, it is required as it allows growing with the organization without entirely changing the architecture.
- Improving overall security posture by providing real-time visibility into logs and events thus facilitating real-time faster detection of threats and quicker response action.

This gives practical exposure to implementing real-life security monitoring systems as well as keeping up with all current practices under the umbrella of DevSecOps where security, automation, and scalability converge in any of the cases.

## **Scope of the Project:**

### **Project Scope**

The scope of the Project includes the design, development, and deployment of a secure and scalable log monitoring system using Docker containers and Splunk Enterprise. The solution attempts to automate the ingestion, analysis, and alerting of security-pertaining logs from distributed systems to ensure enhanced visibility, incident detection, and operational efficiency.

### **The focus of the project is on:**

- **Centralized log monitoring:** Collecting logs from multiple sources such as web servers, applications, and authentication services into a single instance of Splunk.
- **Docker-based deployment:** Portability, scalability, and consistency in the environment are achieved by containerizing Splunk Enterprise and Splunk Universal Forwarders.
- **Security and alerting:** Configuring real-time alerting for suspicious events such as failed logins, wrong password attempts, and unauthorized access.
- **Scalability:** The system can thus be scaled horizontally by deployment of more forwarders or Splunk indexers as required.
- **CI/CD integration:** The deployment and upgrading mechanism shall be automated through Jenkins pipelines for reliability and consistency.
- **Cloud integration (optional):** Supporting future extension to AWS for log storage (S3), security (IAM), and long-term analysis.

The system aims to be a true, enterprise-grade solution for real-time log visibility, threat detection, and operational insights across dynamic infrastructures.

# **Problem Identification**

## **Existing Issues in Log Monitoring**

In modern IT systems, logs are a primary backbone of system monitoring, troubleshooting, and security auditing. Log analysis bypassing these restrictions in performance, scaling, and reliability. Some issues pertaining to existing log environments include the following:

### **1)Lack of Centralization**

Logs are spread over different machines and environments, preventing access to, correlation with, and analysis of data through a single interface. This requires time-consuming manual efforts and runs the risk of missing critical events.

### **2)Delayed Detection and Response**

Traditional systems do not provide real-time log analysis and alerting. Accordingly, incidents such as unsuccessful logins, suspicious activity, or application errors may not be detected until significant damage has occurred, allowing for only ineffective response efforts.

### **3)Highly Complex Setup and Configuration**

Most of the classical log-monitoring tools have complex setups and configuration requirements, making them difficult to use in organizations with small teams with limited technical resources.

### **4)Scalability Issues**

With an increasing volume of data due to services and infrastructures getting larger, many traditional solutions now find it difficult to scale without fail. This might result in dropped logs, performance issues, or higher expenses.

### **5)Security Issues**

If there are no strong encryption services in place and a sound access control policy, logs can expose sensitive information to any unauthorized person. Moreover, weak monitoring systems may fail to capture the malicious activities or attempts of login properly.

### **6)Inadequate Alert Mechanisms**

Such systems may miss some critical incidents or might send too many unwanted alerts which could lead to alert fatigue and even missing anomalies.

## **Motive for the Project**

Many more cyber threats are becoming motivated to exploit growing complexities in the digital infrastructure, for which a secure, scalable, and automated solution is required for log monitoring. This project-**"Secure and Scalable Log Monitoring System using Docker & Splunk"** has been up in the air to prove these points practically to face these issues through modern DevOps and cybersecurity practices.

The scope of our motivation was to create a solution that:

**Brings Synergy Between the Power of Docker and Splunk:** While using Docker as the containerization platform itself ensures high portability and scalability, Splunk is the most powerful engine for live logging and analysis and visualization of these logs.

**Enables Real-Time Alerting:** Events like multiple failed attempts or failure during authentication should be able to trigger instant alerts through emails or dashboards so proactive action can be taken.

**Automate the Setup:** All deployment and configuration procedures automated by CD pipelines (Jenkins or GitHub Action) to decrease human errors and save time.

**Supports Distributed Architecture:** With Splunk Universal Forwarder on a separate VM (Kali Linux) and Splunk Enterprise on Azure container, this system simulates real-world distributed environments.

**Strengthens Security Monitoring:** This system allows the detection of anomalies or unauthorized activities for secure log forwarding and user access control.

In essence, the intent was to produce a log-and-activity monitoring solution that would go into **production, be easy to deploy, and secure**, not just to remedy the current issues but also to be a scalable model for future enterprise-class systems.

## 4. Proposed Solution

### 4.1 System Overview

This distributed architecture ensures robust log aggregation, efficient monitoring, and scalability across multiple environments. We propose here a **secure and scalable log monitoring system** using Docker and Splunk to prevent bickering of manual log monitoring, real-time alerting, and decentralized log sources. This includes:

- **Splunk Enterprise:** Hosted in Docker container with Azure VM deployments to process and visualize logs.
- **Splunk Universal Forwarder:** Installed in another container (on a Kali Linux VM) simply forwarding logs securely.
- **Jenkins CI/CD Pipeline:** Automates the deployment and realizing update workflows through GitHub integration.
- **Real-time alerts** based on critical log events such as failed login attempts or suspicious access patterns arise through Email Alerting.
- **Optional AWS Integration:** Logs can be stored or backed up to AWS S3 as a redundancy and for more analysis.

This architecture is distributed and thus assures considerable log aggregation, monitoring with efficient scale across many environments.

---

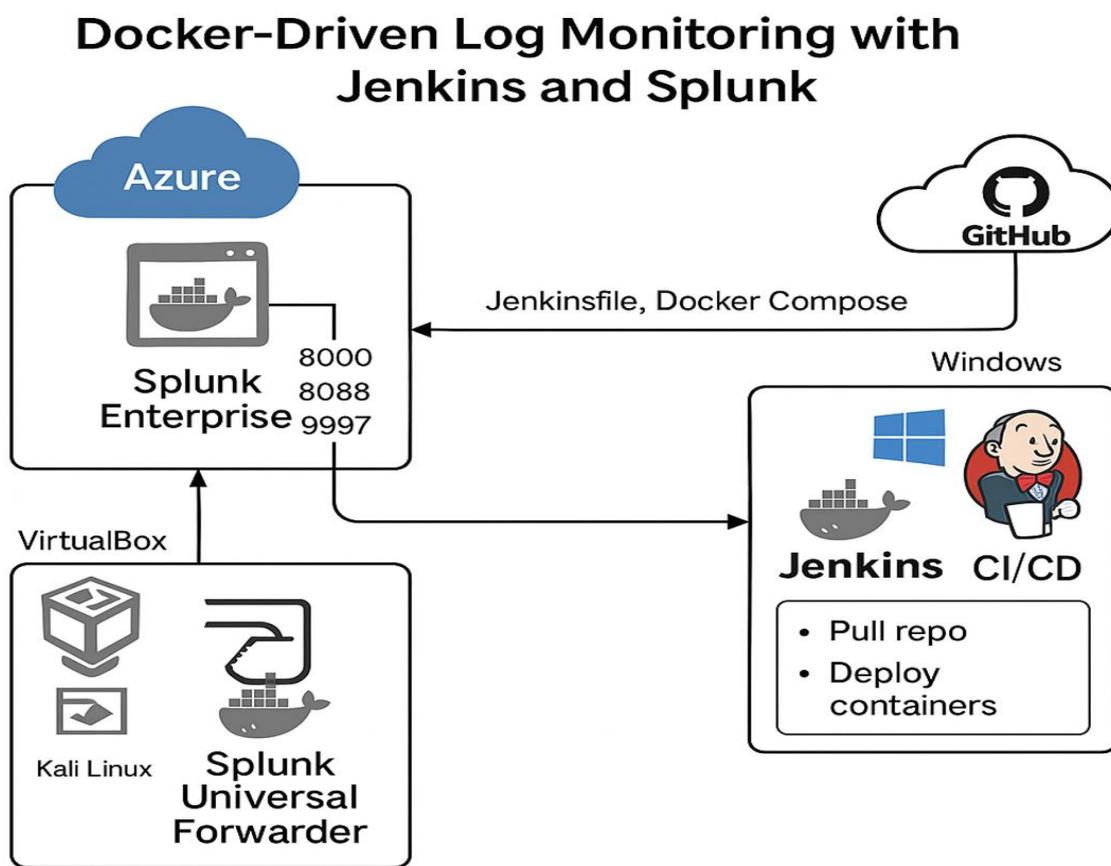
### 4.2 Key Features

Feature	Description
<b>Containerized Setup</b>	Using a Docker-based setup where Splunk components are deployed in isolated and scalable deployment.
<b>Log Aggregation</b>	Central collection of logs from distributed systems via Splunk Forwarder
<b>Real-Time Alerting</b>	Email alerts based on specific events such as failed login and all auth failures.
<b>CI/CD Pipeline</b>	Automates build and deployment via Jenkins and GitHub
<b>Security</b>	Role-based access, transported secure logging, and cloud security integration
<b>Scalability</b>	Expanded flexibility for additional forwarders or storage integrators.
<b>Monitoring Dashboards</b>	Custom Splunk dashboards for visual analytics and tracking
<b>Cross-Platform Compatibility</b>	Working on Windows, Linux, and in the cloud.

## 6. System Architecture

### 6.1 Architecture Diagram

Architecture is designed for ultimate **secure scalable automated log monitoring** over Splunk and Docker. It is realized as several components across multiple environments that integrate into CI/CD practices for doing it all more efficiently.



### 6.2 Components Involved

Component	Role
<b>Splunk Universal Forwarder</b>	Installed inside Kali Linux VM. Forwards logs to Splunk Enterprise
<b>Splunk Enterprise</b>	Installed via Docker on Azure VM. Receives and analyzes logs
<b>Jenkins</b>	Runs on Windows. Manages CI/CD pipeline for deploying Docker containers
<b>GitHub</b>	Hosts source code, Dockerfiles, and Jenkinsfile
<b>Docker</b>	Used to containerize Splunk Enterprise and Universal Forwarder
<b>Email Alert System</b>	Configured in Splunk for specific log triggers (e.g., failed login)

## Implementation Details

### 1. Tools and Technologies Used

To create a secure and scalable log-monitoring platform, the following tools and technologies were used:

- **Docker:** For the containerization of the Splunk components and services themselves.
- **Splunk Enterprise:** For log ingestion, searching, and alerting.
- **Splunk Universal Forwarder:** Lightweight agent used to forward logs from the endpoint systems.
- **Jenkins:** For CI/CD automation.
- **GitHub:** Version control and repository for Jenkins integration.
- **Linux (Ubuntu/Kali) and Windows:** OS environments for Splunk and Forwarders applications.
- **Docker Compose:** For orchestration of multi-containers.

### 2. Docker Configuration

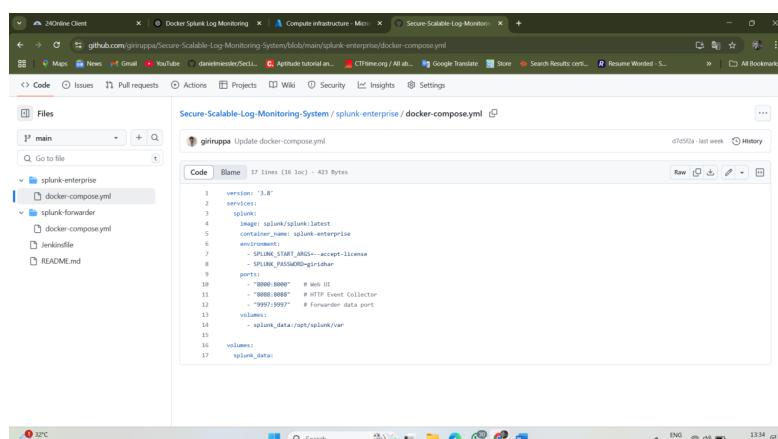
With Splunk components containerized in Docker, this made them able to be deployed in isolation and therefore in reproducible manner:

- **Splunk Enterprise Container:**

- Docker image pulled from [Splunk Docker Hub repo](#)
- Set up with ports 8000, 8089, and mounted volumes for persistence.
- Credentials and license acceptance were set through environment variables

- **Docker Compose File:**

- Manage service lifecycle (up/down).
- Used to bring up splunk-enterprise service by a single command.



The screenshot shows a GitHub repository page for "Secure-Scaleable-Log-Monitoring-System". The "splunk-enterprise" branch is selected. The "docker-compose.yml" file is open in the code editor. The file content is as follows:

```
version: "3.8"
services:
  splunk:
    image: splunk/splunk:latest
    container_name: splunk_enterprise
    environment:
      - SPLUNK_START_ARGS=--accept-license
      - SPLUNK_PASSWORD=gitadmin
    ports:
      - "8000:8000" # Web UI
      - "8088:8088" # HTTP Event Collector
      - "9997:9997" # Forwarder data port
    volumes:
      - splunk_data:/opt/splunk/var
    volumes:
      splunk_data:
```

---

### 3. Splunk Setup

- Open the **Splunk Enterprise Web Interface**, by hitting `http://<host-ip>:8000`.
  - Configured indexes and data inputs for log collection.
  - Installed and configured **Splunk Universal Forwarder** on a remote VM:
    - Used `/opt/splunkforwarder/bin/splunk add forward-server <IP>:9997`.
    - Added file monitoring inputs:  
`./splunk add monitor /var/log/auth.log`.
- 

### 4. Jenkins CI/CD Integration

- **Jenkins Pipeline** created with a `Jenkinsfile` hosted in GitHub.
- Stages contained:
  - GitHub Checkout
  - Repository Cloning
  - Splunk Deployment using Docker-Compose
- Jenkins monitored the Git repo for changes and redeployed automatically on push.

stage:

```
stage('Start Splunk Enterprise on Azure') {
    steps {
        echo "Skipping automatic start of Splunk on Azure."
        sshagent(['azure-ssh-credentials']) {
            sh 'ssh -o StrictHostKeyChecking=no $AZURE_USER@$AZURE_IP "cd /home/azureuser/docker-log-monitoring && docker-compose up -d"'
        }
    }
}

stage('Start Universal Forwarder on Kali') {
    steps {
        echo "Skipping automatic start of Universal Forwarder on Kali."
        sshagent(['kali-ssh-credentials']) {
            sh 'ssh -o StrictHostKeyChecking=no $KALI_USER@$KALI_IP "cd /home/giridhar/devops/SplunkUniversal && docker-compose up -d"'
        }
    }
}
```

---

## 8. Data Flow & Log Collection

Confidently maintaining a superior and pragmatic log-monitoring system is dependent on the seamless integration of data flow from multiple sources to a central repository for indexation, searching, and analysis. The project implemented a secure and scalable pipeline for log data into the **Splunk Universal Forwarder** and **Splunk Enterprise** to ensure high availability and accuracy of logs for both security monitoring and operational analysis.

---

## 8.1 Log Sources

Log sources are the beginning of event and activity data generation. This includes systems, applications, servers, or network devices. From our side, we have chosen quite a few types of log-generating sources such as:

- **Authentication logs** for Linux systems (e.g., /var/log/auth.log)
- **System logs** from /var/log/syslog or /var/log/messages
- **Web application logs** (e.g., Apache, NGINX access logs)
- **Logs that capture security incidents** such as failed login attempts or unauthorized access
- **Logs from Docker containers** where each service container generates stdout/stderr logs

The variety of sources ensures that the monitoring system can offer great visibility into the behaviour and security events taking place within the system.

---

## 8.2 Splunk Universal Forwarder

The **Splunk Universal Forwarder (UF)** is a lightweight data-collecting agent that is installed mainly on edge machines (in this case, a Kali Linux VM within VirtualBox). The main duty of the Universal Forwarder is to securely collect raw logs from local sources and then forward them centrally to be indexed and analyzed from that **Splunk Enterprise** instance.

### Such Key Features and configurations:

- Installed as a Docker container inside the Kali VM
- inputs.conf file editing to specify the paths of the monitored logs
- Strictly secured over port 9997 (which is the default Splunk TCP port)
- Deployment server model which manages multiple forwarders (optional)
- Real-time log forwarding with minimum resource utilization

Thus, this important component ensures the sensitive logs at various remote endpoints remain captured and forwarded efficiently.

---

## 8.3 Indexing and Searching (Splunk Enterprise)

**Splunk Enterprise** on the receiving end (installed on a Docker container) constitutes **the centralized indexing and search engine**.

### Process breakdown:

- **Receiving logs** via TCP input from the Universal Forwarder (either set in inputs.conf or in the Splunk UI)
- **Log parsing into specific indexes** (e.g. auth\_index, web\_index) and log indexing
- Structuring data into **events with timestamps, source types, and host associated**
- **Search Processing Language (SPL)** is used by the end-users for querying and analyzing the logs
- Dashboards and visualization provide insights into security trends, failed logins, and anomalies

#### **Example SPL search:**

```
index=auth_index sourcetype=linux_secure "Failed password"
```

Retrieving failed SSH login attempts to trigger alerts.

---

## **8.4 Summary of Log Flow**

- 1. Log Generation:** Logs generated-on OS, application, and container level
  - 2. Collection:** Universal Forwarder-read and forwards logs in near real time
  - 3. Transfer:** Data will now be sent over the secure network to the Splunk server
  - 4. Indexing:** Splunk Enterprise parses and stores logs on specific indexes
  - 5. Searching:** Users analyse logs with the help of SPL and set alerts/dashboards
- 

## **9. Alerting Mechanism**

This was among the principal tasks undertaken by this project: establishing real-time monitoring and proactive alerting on system log entries where users have performed suspicious or noteworthy activities. Alerts are among essential characteristics of every log monitoring system that will notify security teams about anomalies—especially events with the potential of unauthorized access attempts, policy violations, or system errors. In this case, we have created email alerts using Splunk Enterprise as the technology platform for real-time threat detection and notification purposes.

### **9.1 Setting up Email Alerts in Splunk Enterprise**

Splunk Enterprise offers a built-in method to configure alerts and notifications through many different actions such as sending an email or executing a script in conjunction with other third-party services like Slack, PagerDuty, and ServiceNow.

Here are the steps for configuring **email alerts in Splunk Enterprise**:

### **Step 1: Enable Email Settings in Splunk**

1. Go to **Settings** → **Server settings** → **Email settings** in Splunk.
2. Fill in the SMTP details of your email service provider. For example:
  - a. **Mail Host:** smtp.gmail.com:587
  - b. **Authentication:** Yes
  - c. **Username:** [girirupppa964@gmail.com](mailto:girirupppa964@gmail.com)
  - d. **Password:** App Password (for Gmail with 2FA)
  - e. **TLS/SSL:** Enable STARTTLS
3. Save the configuration and test the email receipt.

### **Step 2: Write a Search Query**

Build a SPL query for Splunk that specifies what condition should be included for the alert. To, for example, detect failed log-on attempts, the following would be a frequently used query:

*index=forwarded\_logs sourcetype=auth\_logs "authentication failure" OR "failed password"*

This query checks logs forwarded by the **Splunk Universal Forwarder** for keywords that indicate failed authentication attempts.

### **Step 3: Creating and Scheduling the Alert**

1. Visit **Search & Reporting**.
2. Run your SPL query to check that it gives relevant results.
3. Click **Save As** → **Alert**.
4. Add relevant details:
  - a. **Alert Title:** Detected a Failed Password Attempt
  - b. **Alert Type:** Scheduled or Real-Time
  - c. **Condition to Trigger:** If results are greater than 0
  - d. **Time Range:** Last 5 minutes (can be adjusted)
5. Under Trigger Actions, choose Send email.
  - a. Add email recipient addresses.
  - b. Customize subject line and message body (result tokens if needed).
6. Save alert.

## 9.2 Sample Use Case: Wrong Password Detection

Now, let's review an implementation for a situation in which an erroneous password in a Linux system is entered multiple times, triggering an email notification for such an event.

### Scenario:

A malevolent intruder or just an incompetent user is unknowingly attempting to gain entrance into a protected website or SSH terminal with an invalid password. We want to make sure this action is logged by the Splunk Universal Forwarder in Splunk Enterprise's records.

### Step-by-Step Flow:

#### 1. Log Generation:

- o A log of the authentication failure should be generated on the monitored machine (Kali Linux) in the `/var/log/auth.log` file.
- o Failed password for invalid user admin from `192.168.1.100 port 2222 ssh2`

#### 2. Log Forwarding:

- o Configure the Splunk Universal Forwarder installed on the machine to monitor `/var/log/auth.log` and forward events to the Splunk Enterprise instance in Azure.

#### 3. Log Indexing and Searching:

- o This log is then indexed by Splunk Enterprise, which will use a custom index, such as `index=forwarded_logs`.

#### 4. Detection Query in Splunk:

- o The SPL query scans for words like "`Failed password`" and "`authentication failure`", or similar patterns in the logs:
- o `index=forwarded_logs sourcetype=linux_secure "Failed password"`

#### 5. Triggering the Alert:

- o An event-based alert is incorporated into the workings.
- o Set the alert to poll every five minutes and mail if at least one matching event is found.

#### 6. Email Notification:

- o Takes place when an email is sent to the system administrator, with a subject as follows:
- o `Alert: Failed Login Detected from 192.168.1.100`
- o The body of the email contains the actual log entry, along with the timestamp and IP address of the source.

### Results:

The alert enables the security or operations team to react in real-time to initiate prompt corrective action—whether blocking an IP, disabling an account, or sweeping in for the forensic audit.

# Security Measures

Securing log data and ensuring the integrity of the monitoring system have become strategic features of this project, especially considering that logs are sensitive by nature. Logs contain authentication attempts, system errors, access logs, and user activity, all of which are considered sensitive information and must not be corrupted. To solve these problems, our solution has two major security strategies:

**Role-Based Access Control (RBAC)** and **Secure Communication Channels**.

---

## 1. Role-Based Access Control (RBAC)

RBAC is a primary mechanism to restrict system access only to authorized users based on their organizational roles. Thus, a user has access to information and can perform tasks only according to his or her responsibilities. This reduces the chance of an insider threat or accidental misconfiguration.

### Key Implementations:

- **User Segmentation:** Users are grouped by roles such as Admin, Developer, Security Analyst, and Viewer. Permissions for each of these roles are specific to Splunk and Jenkins.
- **Custom Role Permissions in Splunk:** Within Splunk, a role is configured to restrict users against searching or viewing or modifying certain information. For example, a Viewer may be limited to the ability to view dashboards, whereas a Security Analyst could create alerts and review logs.
- **Access to Jenkins:** In Jenkins, a matrix-based security approach defines job-level and global-level permissions. Only the Jenkins administrator has full access to pipeline definitions; all other users are restricted to triggering builds and accessing logs.
- **Audit trails:** All user activity in Splunk and Jenkins is logged for the sake of accountability. Any unauthorized attempts are tracked and flagged.

RBAC minimizes security risks and guarantees compliance with industry best practices by enforcing the *least privilege*.

---

## 2. Secure Communication Channels

All communication channels put in our architecture are secured by encryption and secure protocols to ensure the confidential and integrity of data while it is being transferred across different systems.

Key Implementations:

- **TLS/SSL Encryption:** The communication between Splunk Universal Forwarder (running in the Kali Linux VM) and Splunk Enterprise (hosted on the Azure VM) is secured with SSL certificates. This means log data is encrypted while in transit and is not exposed to man-in-the-middle other types of attacks.
- **Secure GitHub Integration:** The Jenkins pipeline integrates securely with the GitHub repository leveraging Personal Access Tokens (PATs) and webhook secrets to restrict unauthorized code pushes and access.
- **Secured Jenkins Deployment:** As an HTTPS, Jenkins has been configured using self-signed or trusted CA certificates and stores user credentials securely using Jenkins' credential plugin with encryption.
- **Firewall and IP Restrictions:** Firewall rules secure Azure VM and VirtualBox environments to allow only access to needed ports, such as 8000 for Splunk and 8080 for Jenkins or specific IP addresses.
- **Log Obfuscation:** This is the making of sensitive data like passwords and tokens in the logs obfuscating by filtering it before indexing in Splunk so that they do not leak.

Combining RBAC with secure communication practices ensures absolute that only the right people will have access to the right information and that data transfer will be free from unauthorized interception. Thus, it fortifies and strengthens the integrity of log monitoring systems and adds trust when implemented in enterprise environments.

---

## Scalability and Performance

### 1. Docker Compose for Container Management

Unfortunately, our system should remain modular. Therefore, we use **Docker Compose** to manage and orchestrate all relevant containers like Splunk Enterprise and Universal Forwarder. Docker Compose takes the definition and execution of multi-container docker applications with a single YAML configuration file. With the help of Docker Compose:

- **Easily duplicate any environment** under development testing, or production.
- **Deploy containers as isolated** for the parts of components so that dependency is not affected.
- **Horizontally scale services** by adjusting the number of container instances through the scale command or replicas in Docker Swarm mode.
- Manage and restart the services intact using docker-compose up/down commands with few manual interactions.

This makes our entire log monitoring infrastructure portable, consistent, and highly manageable-all essential traits of a large-scale system.

---

## 2. Resource Utilization Optimization

To deploy higher number of containers with higher performance and scalability, it **does not meet any** of these about the **efficiency of one being used resources**. So, in our own case, we optimize resource consumption in the following ways:

- **Limit at container Level:** Setting the cpu and memory consumption in docker-compose.yml ensured that not any one of the hosted containers abuses resources.

Example snippet:

```
deploy:  
  resources:  
    limits:  
      cpus: "1.0"  
      memory: 1G
```

- **Light base images:** like official, slim or alpine-based images, fast, official base images help in saving memory and startup time from containers.
- **Logging and Filtering:** by Splunk and the Universal Forwarder for log filtering and rotating the old logs, we have avoided bloating disks and have enhanced I/O performance as well.
- **Good use of persistent storage:** In this case, for example, indexing data and configuration files for Splunk stores in Docker volumes that are brought back to default state of data reuse on rescheduling and restarts of containers.

---

## 3. Scalability Considerations

Our consideration for future scalability is built into the very architecture of the system. In case log volume rises or if more systems are to be monitored, the architecture will support:

- **Adding more Universal Forwarders:** The new systems are easily deployable to collect logs and send them to a central Splunk instance.
- **Load Balancing:** Docker allows us to put Splunk behind a reverse proxy or load balancer should we add multiple indexers or search heads in the future.
- **Cloud Integration:** This setup can work on cloud platforms like AWS or Azure. Everything is containerized, thereby deploying the setup onto cloud VMs or integrated with managed Kubernetes services for elastic scalability.

The **Secure and Scalable Log Monitoring System** remains performance-driven with Docker Compose, thus holding under load, scalable for larger infrastructures, and resource-efficient for enterprise readiness and quickly adaptable to real-world deployment environments.

## 13. Testing & Results

This chapter addresses the most essential aspect of any project lifecycle: testing, which is supposed to substantiate whether or not the developed system has been realized in practice, for all expected scenarios. While for the above-mentioned project "Secure and Scalable Log Monitoring System using Docker & Splunk", a thorough functional testing study, scenario-based validation, and visual log verification were performed for the reliability, performance, and security testing of the system.

---

### 13.1 Functional Testing

Functional testing involved checks on the core components and processes of the system. The tests verified that the integration of Splunk Enterprise, Splunk Universal Forwarder, and Docker had gone smoothly to achieve the purpose of log monitoring and alerting. The main functionalities tested were:

- **Log Ingestion:** This checked to see whether logs were being transmitted from the source system (Kali Linux VM) towards the Splunk Enterprise instance forwarder (running as an Azure container) successful.
- **Indexing & Storage:** This variable examined whether logs already received on Splunk Enterprise were properly indexed and stored for later retrievable access.
- **Search Capability:** SPL queries were executed to provide evidence that the logs were searchable and filterable by time, source type, and content.
- **Email Alert Triggering:** Wrong password attempts on a dummy web interface were simulated and the delivery of the alert and email was confirmed.
- **Uptime of Docker Containers:** Confirmed successful launch and expected uptime of the Splunk Enterprise container after starting it with docker-compose.
- **CI/CD:** A pipeline was tested on Jenkins to automate pulling and deployment of updated Splunk configurations from GitHub repos.

Valid and invalid test inputs were used to evaluate the robustness of each functionality and also the error handling systems.

---

### 13.2 Log Sample Output Screenshots

Here are some sample screenshots with a short description of the output collected during the testing phase:

---

#### 1. Splunk Dashboard View:

- Description:** A screenshot to show the Splunk dashboard with the forwarded logs during capture.
- Details Captured:** Log source, timestamp, severity level, host machine, and message contents.

Time	Event
4/26/25 6:50:24.654 AM	<pre> &lt;!-- SELECTED FIELDS @ host 1 @ source 7 @ sourcetype 5  INTERESTING FIELDS # avg_age 9 # bytes 16 # clientip 2 # component 3 # date_hour 1 # date_mday 1 # date_minute 1 # date_second 7 # date_vday 1 # date_year 1 # date_zone 2 --&gt; <i>        code: 200 durationMS: 4.776 expectedCode: 200 hostname: 79af5a993575 level: INFO location: splunkmgmtclient/client.go:186 message: request to splunk server succeeded operation: GetConfStanza requestURL: https://127.0.0.1:8089/services/configs/conf-server/teleport_supervisor?output_mode=json service: identity time: 2025-04-26T06:50:24.654Z </i> ) Show as raw text host = 79af5a993575   source = /opt/splunk/var/log/splunk/sup-pkg-identity-stdout.log   sourcetype = sup-pkg-identity-stdout-2 </pre>
4/26/25 6:50:24.652 AM	<pre> 127.0.0.1 - splunk-system-user [26/Apr/2025:06:50:24.652 +0000] "GET /services/configs/conf-server/teleport_supervisor?output_mode=json HTTP/1.1" 200 1391 "-" "Go-http-client/1.1" "-" 2ms host = 79af5a993575   source = /opt/splunk/var/log/splunk/solnkd_access.log   sourcetype = solnkd_access </pre>

*Example:*

The log entry from */var/log/auth.log* on Kali Linux showing the failed login attempt due to an incorrect password.

## 2. Search Query (SPL) Result:

- Description:** SPL query result for unsuccessful authentication attempts.
- Query Used:**
- index=main sourcetype=linux\_secure "Failed password"**
- Result:** Provides a list of failed login attempts with the accompanying timestamp and IP address information.

## 3. Triggered Alert Screenshot (Email):

- Description:** An email alert screenshot displaying the state triggered (e.g., "Failed password" log event).
- Subject:** "Alert: Unauthorized Login Attempt Detected".
- Body:** Comprises time with source IP and log context.

#### **4. Docker Container Logs:**

- **Description:** Output from docker ps and docker logs commands.
  - **Reason:** In order to verify health and running Splunk services within the container.
- 

#### **5. Jenkins Console Output:**

- **Description:** Output of a successful pipeline run on Jenkins.
  - **Stages Shown:** Checkout from GitHub, Docker Compose Build and Up, deployment confirmation.
- 

### **Summary of Test Outcomes**

Test Case	Expected Outcome	Actual Outcome	Status
Universal Forwarder sends logs	Logs appear in Splunk dashboard	Success	<input checked="" type="checkbox"/> Passed
Alert on failed login (wrong password)	Email triggered within seconds of detection	Success	<input checked="" type="checkbox"/> Passed
Docker container uptime check	Splunk container is running & accessible	Running and accessible	<input checked="" type="checkbox"/> Passed
Jenkins pipeline deployment	Project is deployed automatically with each push	Success	<input checked="" type="checkbox"/> Passed

---

---

### **Final Thoughts & Conclusion**

#### **Summary**

It was with heavy and great efforts that we made this project entitled "**Secure and Scalable Log Monitoring System using Docker & Splunk**". It solves almost every problem that an environmental log monitoring integration has-a few highlighted among those being scalability, alerting, and integration. Using containerized deployment leveraging Docker and analysis from Splunk, we built real-time flexible log monitoring architecture to intake logs from multiple systems.

The solution consists of a Splunk Universal Forwarder installed on a remote system (e.g., Kali Linux VM) and integrated with a central Splunk Enterprise instance running inside a Docker container on an Azure VM. Jenkins automates deployment to version control in GitHub and enables smooth CI/CD implementation. Other interesting features included in the system are: email alerting for failed logins.

Thus, this project increases operational security and portrays what modern DevSecOps could do; it offers an efficient and modular log analysis in an organization, which can be scaled according to need.

---

## Challenges Faced

These were some of the real-time challenges put forward in this project during its development and deployment phases.

- 1. Cross-Platform Integration:** Inter-Network and firewall configuration present serious issues during cross-communications concerning a host Windows operating system, its Kali Linux VM, and Azure-hosted Docker containers.
  - 2. Splunk Authentication & Licensing:** Splunk Authz and forwarding configurations and enterprise feature licenses have made it very difficult for proper implementation and testing without a thorough reading of documentation.
  - 3. Coordinates Jenkins and Docker:** Automating docker compose tasks through Jenkins in an environment that does not directly host the containers includes complex management of path and permission issues.
  - 4. Email Alert Configuration:** To set up reliable email alerts for specific events, for example, incorrect password entries, proper event filtering, using SPL (Search Processing Language) and SMTP server configuration, is needed.
  - 5. Data Consistency & Volume:** Spending time to generate realistic-looking logs along with testing within variable load conditions was necessary to ensure that Splunk behaved well and efficiently under realistic workloads.
- 

## Future Enhancements

Succeeding the completion of the project objectives, some areas enhance and scale down for future use:

- 1. AI/ML-Based Anomaly Detection:** Making the gross system smarter and autonomously run by creating anomaly detection using ML on logs in Python for detecting anomalies like brute-force attacks or insider threats.
- 2. Support for Multi-Cloud and Microservices:** Extend the architecture to multi-cloud deployments and log aggregation from microservices managed by Kubernetes.

**3. Dashboards & Analytics:** Advanced dashboards are created within Splunk to impart visual insights regarding network behavior, system health, and incident trends to security and operations teams.

**4. Threat Intelligence Integration:** Integrating external threat feeds and allowing Splunk to automatically tag or alert on IOCs.

**5. Audit and Compliance Reporting:** Incessantly building templates for regulatory reporting (i.e., from log data to help compliance teams with GDPR, ISO 27001).

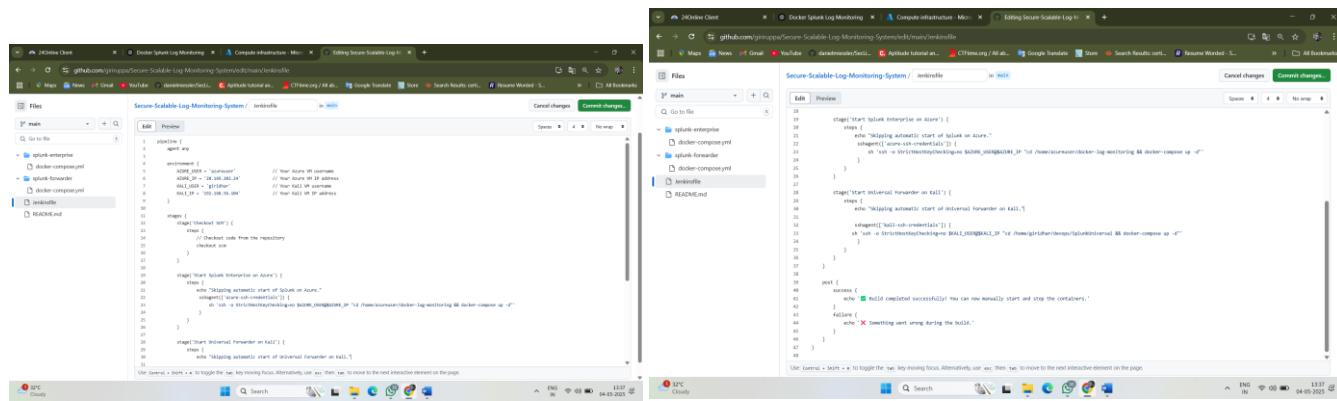
**6. Auto-Healing and Remediation Scripts:** Configuration of Jenkins or Ansible will trigger remediation scripts (block IP, and restart service) upon detection of critical log events.

## References

1. Docker Documentation – <https://docs.docker.com/>
2. Splunk Enterprise Documentation – <https://docs.splunk.com/Documentation/Splunk>
3. Jenkins Documentation – <https://www.jenkins.io/doc/>
4. Splunk Universal Forwarder Setup – <https://docs.splunk.com/Documentation/Forwarder>
5. AWS EC2 Instance User Guide – <https://docs.aws.amazon.com/ec2/>
6. DevOps Best Practices – <https://www.atlassian.com/devops>
7. Secure Logging Architecture Whitepapers – [Various Online Sources]
8. GitHub Repository for Project – <https://github.com/girirappa/log-monitoring-docker-splunk>

## Appendix

### A. Jenkinsfile



The screenshot shows a dual-monitor setup. The left monitor displays a Jenkinsfile titled 'Jenkinsfile' with the following content:

```
pipeline {
    agent any
    stages {
        stage('Alerts') {
            steps {
                script {
                    // Alert on specific log entries
                    if (sh(script: "grep -q 'error' /var/log/splunkd.log")) {
                        slackSend(color: "#FF0000", message: "Splunk Log Monitoring System has detected an error!")
                    }
                }
            }
        }
        stage('Checkouts') {
            steps {
                checkout scm
            }
        }
        stage('Logs') {
            steps {
                sh("tail -f /var/log/splunkd.log")
            }
        }
    }
}
```

The right monitor displays another Jenkinsfile titled 'Jenkinsfile' with the following content:

```
stage('Start Splunk on Docker') {
    steps {
        dockerComposeUp()
    }
}
stage('Wait for Splunk to start') {
    steps {
        sleep 10
    }
}
stage('Start Universal Forwarder on Kali') {
    steps {
        dockerComposeUp()
    }
}
stage('Wait for Universal Forwarder to start') {
    steps {
        sleep 10
    }
}
stage('Start Splunk Forwarder on Kali') {
    steps {
        dockerComposeUp()
    }
}
stage('Wait for Splunk Forwarder to start') {
    steps {
        sleep 10
    }
}
```

## B. Docker Compose File – `splunk-enterprise/docker-compose.yml`

The screenshot shows a GitHub repository page for the "Secure-Scaleable-Log-Monitoring-System". The repository path is "splunk-enterprise". The current file being viewed is "docker-compose.yml". The code editor displays the following configuration:

```
version: '3.8'
services:
  splunk:
    image: splunk/splunk:latest
    container_name: splunk-enterprise
    environment:
      - SPLUNK_START_ARGS---accept-license
      - SPLUNK_PASSWORD=giridhar
    ports:
      - "8000:8000" # Web UI
      - "8088:8088" # HTTP Event Collector
      - "9997:9997" # Forwarder data port
    volumes:
      - splunk_data:/opt/splunk/var
volumes:
  splunk_data:
```

## C. Docker Compose File – `universal-forwarder/docker-compose.yml`

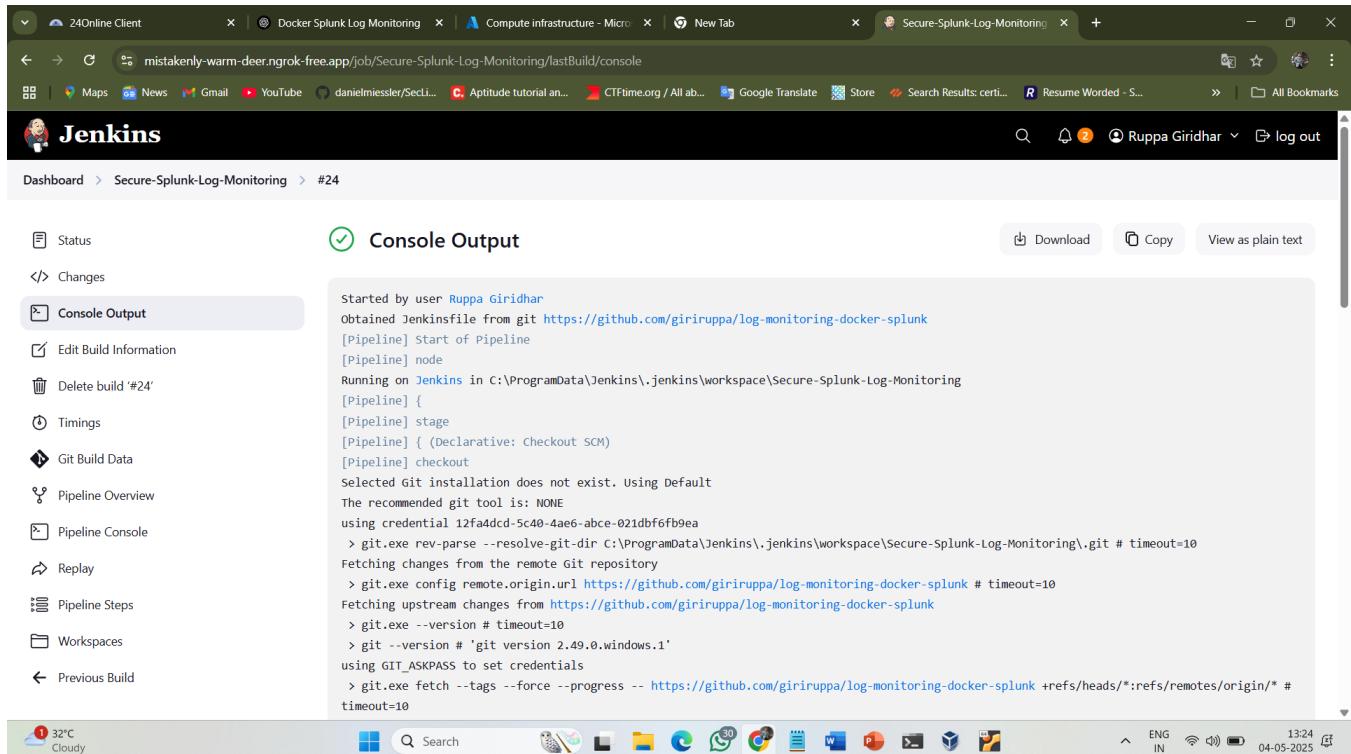
The screenshot shows a GitHub repository page for the "Secure-Scaleable-Log-Monitoring-System". The repository path is "main/splunk-forwarder". The current file being viewed is "docker-compose.yml". The code editor displays the following configuration:

```
version: '3.8'
services:
  forwarder:
    image: splunk/universalforwarder:latest
    container_name: splunk-forwarder
    environment:
      - SPLUNK_START_ARGS---accept-license
      - SPLUNK_PASSWORD=giridhar
      - SPLUNK_ADD-monitor /var/log
      - SPLUNK_FORWARD_SERVER=20.169.202.24:9997
    volumes:
      - /var/log:/var/log
```

## D. Splunk Configuration Snippet – `inputs.conf` (For Forwarder)

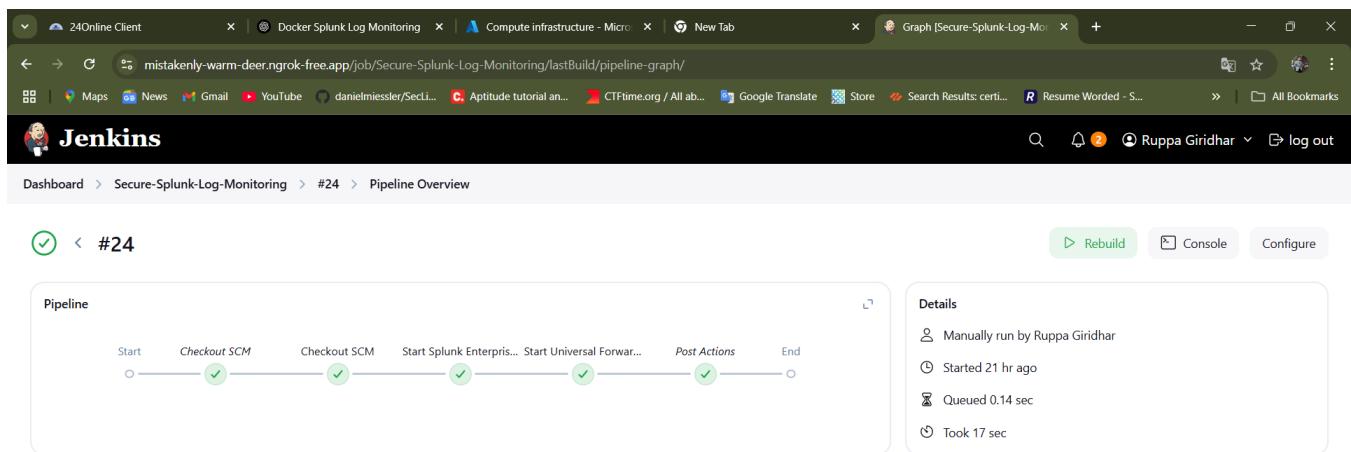
```
[monitor:///var/log/auth.log]
disabled = false
index = main
sourcetype = linux_secure
```

### Build Output



The screenshot shows the Jenkins console output for build #24. The left sidebar contains navigation links like Status, Changes, Console Output (which is selected), Edit Build Information, Delete build '#24', Timings, Git Build Data, Pipeline Overview, Pipeline Console, Replay, Pipeline Steps, Workspaces, and Previous Build. The right pane displays the build logs:

```
Started by user Ruppa Giridhar
Obtained Jenkinsfile from git https://github.com/giriruppa/log-monitoring-docker-splunk
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\jenkins\workspace\Secure-Splunk-Log-Monitoring
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential 12fa4dcd-5c40-4ae6-abce-021dbf6fb9ea
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\Secure-Splunk-Log-Monitoring\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/giriruppa/log-monitoring-docker-splunk # timeout=10
Fetching upstream changes from https://github.com/giriruppa/log-monitoring-docker-splunk
> git.exe --version # timeout=10
> git --version # 'git version 2.49.0.windows.1'
using GIT_ASKPASS to set credentials
> git.exe fetch --tags --force --progress -- https://github.com/giriruppa/log-monitoring-docker-splunk +refs/heads/*:refs/remotes/origin/* #
timeout=10
```



```
Memory usage: 35%                               Saved to this PC
Swap usage:  0%
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.
https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

6 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

3 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Sat Apr 26 04:41:01 2025 from 152.59.118.88
azureuser@MyDockerVM:~$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS        PORTS
79af5a993575   splunk/splunk:latest  "/sbin/entrypoint.sh..."  35 hours ago  Up About an hour (healthy)  0.0.0.0:8000-
>8000/tcp, [::]:8000->8000/tcp, 8065/tcp, 8089/tcp, 8191/tcp, 0.0.0.0:8088->8088/tcp, [::]:8088->8088/tcp, 0.0.0.0:9997-
>9997/tcp, [::]:9997->9997/tcp, 9887/tcp   splunk-enterprise
azureuser@MyDockerVM:~$
```

```
KALI LINUX (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
(giridhar@giridhar:[~/devops/SplunkUniversal]
$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS        PORTS
5b916be6cad6   splunk/universalforwarder:latest  "/sbin/entrypoint.sh..."  35 hours ago  Up 18 seconds (health: starting)  8088-8089/tcp, 9997/tcp   NAMES
splunk-forwarder
(giridhar@giridhar:[~/devops/SplunkUniversal]
$
```