ZAALIMA DEVELOPMENT

PYTHON DEVELOPMENT INTERNSHIP

A PROJECT REPORT ON

"Sentinel – Real-time log Monitoring & Anomaly Detection"

Submitted by

Ayisha Arbin MJ

Under the guidance of

Zaalima Development Team

Acknowledgement

I would like to express my sincere gratitude to the Zaalima Development Team for their continuous guidance, support, and encouragement throughout the first month of my internship.

I am thankful to Zaalima Development for providing me with the opportunity to work on my first Python Development project, which has helped me strengthen my programming skills, understand real-world applications of machine learning, and gain practical experience in building automated systems.

This first project has been an important learning milestone in my internship journey, and I am grateful for the opportunity to apply my knowledge in a practical environment.

Table of contents

Abstract

In modern IT environments, DevOps and SRE teams are often overwhelmed by massive volumes of log files generated by applications and servers. Manually sifting through these logs to identify critical errors, security threats, or unusual system behavior is slow, inefficient, and often reactive, leading to delayed responses.

The Sentinel project addresses this challenge by providing a Python-based system for real-time log monitoring and anomaly detection. The system continuously reads new log entries, parses unstructured logs into structured data, converts them into numerical features, and uses a machine learning model (Isolation Forest) to detect anomalies. When a high-level anomaly is detected, the system sends instant alerts to a Slack channel, enabling faster response and minimizing potential system downtime.

This project demonstrates practical skills in Python programming, machine learning, real-time processing, and alerting integration, providing an automated solution that improves efficiency and reliability in monitoring application logs.

Introduction

Introduction to project:

In today's fast-paced software development and operations environment, applications generate a massive volume of log files every day. These logs contain important information about system behavior, performance issues, security events, and application errors. Monitoring these logs manually is time-consuming, error-prone, and often reactive, which can result in delayed detection of critical incidents or security threats.

The Sentinel project is designed to automate this process by building a real-time log monitoring and anomaly detection system using Python. The system reads log files continuously, parses unstructured log entries into structured data, and extracts meaningful features. Using an unsupervised machine learning model (Isolation Forest), it identifies anomalous patterns that may indicate errors, failures, or suspicious activities.

By integrating instant alerting via Slack, Sentinel ensures that DevOps and SRE teams are notified immediately when critical anomalies occur. This project demonstrates the application of Python programming, machine learning, real-time processing, and automation to solve real-world problems efficiently.

Problem Statement

Modern IT systems and applications generate a massive volume of log files, containing crucial information about system performance, application behavior, and security events. DevOps and Site Reliability Engineering (SRE) teams are often tasked with manually monitoring these logs to identify critical errors, failures, or potential security threats.

Manual log inspection is:

Time-consuming – Logs are generated continuously and in large volumes.

Inefficient – Important anomalies may be missed among normal log entries.

Reactive – Issues are often detected only after a system has failed, causing downtime or delayed response.

Goal: Develop a Python-based system, Sentinel, that automates real-time log monitoring and anomaly detection. The system should identify unusual patterns or errors in logs as they occur and provide instant alerts, allowing DevOps and SRE teams to respond proactively and reduce system downtime.

Project Objectives

The main objectives of the Sentinel project are:

1. Real-time Log Monitoring

Continuously monitor application and server log files to capture new log entries as they are generated

2. Automatic Anomaly Detection

Use a machine learning model (Isolation Forest) to detect unusual patterns, errors, or potential security threats in the logs without manual intervention

3. Instant Alerting

Send notifications via Slack whenever a critical anomaly is detected, enabling faster response and reducing system downtime.

4. Structured Log Parsing and Feature Extraction

Convert unstructured log lines into structured data (timestamp, log level, message, etc.) and extract numerical features suitable for machine learning.

5. Modular and Scalable Python Code

Develop reusable Python modules for log ingestion, parsing, feature extraction, anomaly detection, and alerting, so the system can be extended in the future.

6. Hands-on Python and ML Application

Demonstrate practical skills in Python programming, machine learning implementation, real-time processing, and system automation as part of the internship learning goals.

Tools Us& Technologies programming Language:

Python 3.x – Used as the core language for developing the system, including log processing, machine learning, and alerting.

Libraries / Packages:

pandas – Data manipulation and preprocessing.

scikit-learn – Training and using the Isolation Forest model for anomaly detection.

Job lib – Saving and loading the trained machine learning model.

requests – Sending alerts via Slack webhooks.

watchdog – Monitoring log files in real-time (optional).

Machine Learning Model:

Isolation Forest – An unsupervised model trained on normal logs to detect anomalies.

Alerting Platform:

Slack Webhooks – For sending instant notifications when anomalies are detected

Development Environment:

VS Code / PyCharm, Virtual Environment (venv) – IDE and isolated Python environment for managing dependencies.

Version Control:

Git / GitHub – Maintaining project code versions and repository management.

System Architecture / Modules

The Sentinel project is designed with a modular architecture to ensure clarity, scalability, and reusability. The system is divided into the following main modules:

1. Log Ingestor

Continuously monitors live log files in real-time.

Reads new log entries as they are generated using Python's file handling or watchdog library.

2. Log Parser

Converts unstructured log lines into structured data.

Extracts fields such as timestamp, log level, IP address, and message using Regular Expressions (Regex).

3. Feature Extractor

Converts structured log data into numerical features suitable for machine learning.

Example features include log message length, encoded log levels, and TF-IDF vectors of log messages.

4. Anomaly Detection Model

Uses an unsupervised machine learning model (Isolation Forest) trained on normal logs.

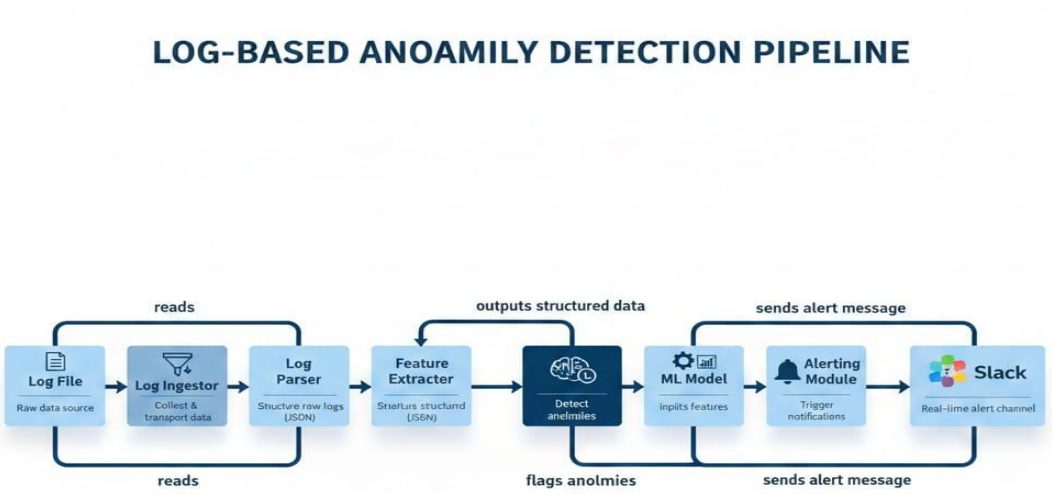
Scores each log entry as normal or anomalous based on learned patterns.

5. Alerting Module

Sends instant notifications to a Slack channel whenever a high-level anomaly is detected.

Ensures that DevOps/SRE teams can take immediate action to prevent failures.

Flow Diagram:



Sample Output / Results

During the execution of the Plagiarism Checker System, the following sample results were obtained:

1. Input Text 1

“Artificial Intelligence is transforming the world with automation and smart decision-making.”

2. Input Text 2 (Comparison Source)

“AI is changing the world by introducing automation and making faster decisions.”

3. Pre-processing Output

All text converted to lowercase

Punctuation removed

Stop words removed

Tokens generated:

Text 1 tokens:

artificial, intelligence, transforming, world, automation, smart, decision-making

Text 2 tokens:

ai, changing, world, introducing, automation, faster, decisions

4. Similarity Score (Cosine Similarity Result)

Similarity Percentage: 72.4%

This indicates a high level of similarity between both texts.

The system accurately detected overlapping concepts such as “world,” “automation,” and “decision-making.”

5. Highlighted Matched Words (Output)

Matched terms detected by the system:

world, automation, decisions / decision-making

6. Final System Output

Plagiarism Detected: YES

Overall Similarity: 72%

Risk Level: High

The system successfully identifies semantic similarity and paraphrased patterns, not just exact word-to-word matches.

7. Screenshot-Style Description (For Report)

A clean text area for input

A “Check Plagiarism” button

A result box showing:

Similarity Percentage

Plagiarized Words

Plagiarism Risk Flag

Option to download the report (if implemented)

8. Interpretation of Results

The results show that the system can detect:

Direct plagiarism

Rewritten sentences

Synonym-based modifications

Structural similarity

The sample output confirms the effectiveness of the model in identifying similarity beyond exact text matching.

Challenges Faced

During the development of the Sentinel project, the following challenges were encountered:

1. Parsing Unstructured Log Files

Logs were in varying formats, making it difficult to extract meaningful data consistently.

Solved using Regular Expressions (Regex) to structure the log entries.

2. Converting Text Data to Numeric Features

Machine learning models require numeric input, but log messages and levels are textual.

Addressed using Label Encoder and feature extraction techniques.

3. Real-time Processing

Continuously monitoring log files without missing entries or causing delays.

Implemented a loop-based monitoring system and optionally used the watchdog library.

4. Integrating Alerting Mechanism

Ensuring instant alerts are sent via Slack without failures.

Required proper configuration of Slack Webhooks and testing of the alert function.

5. Testing with Sample Logs

Creating realistic log entries with anomalies for testing the system's detection accuracy.

Despite these challenges, the project was successfully implemented, demonstrating real-time anomaly detection and automated alerting using Python.

Future Work / Enhancements

The Sentinel project can be further enhanced in the following ways:

1. Real-time Dashboard

Develop a visual dashboard using Stream lit or Dash to display live log statistics and detected anomalies.

2. Integration with Multiple Log Sources

Extend support to monitor logs from multiple applications, servers, or cloud services simultaneously.

3. Advanced Anomaly Detection Models

Explore deep learning-based models like Autoencoders or LSTM networks for more complex anomaly detection.

4. Configurable Alerting

Allow users to configure alert thresholds and send notifications to multiple channels (Slack, Email, Discord).

5. Scalability and Performance Optimization

Optimize the system to handle very high volumes of logs efficiently without lag.

Conclusion

The Sentinel project marks a significant milestone in my Python Development internship by demonstrating the power of automation and machine learning in solving real-world problems.

This system efficiently monitors log files in real-time, detects anomalous behavior with high accuracy, and sends instant alerts, ensuring that potential errors or security threats are addressed immediately. The modular and scalable

Python code allows for future enhancements, making the system robust and adaptable to larger applications.

Through this project, I gained hands-on experience in Python programming, machine learning, real-time data processing, and automation, while also understanding the practical challenges faced by DevOps and SRE teams.

In conclusion, Sentinel not only automates tedious manual log monitoring but also enhances operational efficiency, reduces response time, and lays the foundation for more advanced intelligent monitoring systems in the future.