Marwadi University

Faculty of Engineering and Technology

Department of Information and Communication Technology

Subject: Course: Capstone Project

Academic Year: 2025-26

BEHAVIORAL-ANALYTICS AND USER ACCESS VISUALIZATION (IN SPLUNK)

Name: FAITH JACKSON NKUBA (92200133020)

System Design and Architecture

Introduction

The success of any ICT-based solution depends heavily on a well-structured system design and architecture that ensures robustness, maintainability, and scalability. For this project, the proposed system integrates **web-based interaction, backend data processing, database management, and intelligent monitoring with Splunk** , enabling efficient data handling, anomaly detection, and visualization. The architecture is deliberately modular to ensure that each component operates independently while contributing to the overall system objectives. By leveraging an appropriate technology stack and planning for scalability, the system can adapt to future growth in terms of data, users, and functional requirements.

<u>GENERAL LOOK</u>

Splunk follows an architecture which contains the following three tiers:

- Collection
- Indexing
- Searching

Splunk supports a wide range of data collection mechanisms that helps ingest data into Splunk easily, such that it can be indexed and made available to search. This tier is nothing but your heavy forwarder or universal forwarder.

You must install the add-on application on the heavy forwarder layer instead of the universal forwarder layer. Because, with few exceptions for well-structured data (such as, json, csv, tsv), the universal forwarder does not parse log sources into events, so it cannot perform any action that requires understanding of the format of the logs.

It also ships with a stripped down version of Python, which makes it incompatible with any modular input applications that require a full Splunk stack to function. The heavy forwarder is nothing but your collection tier.

The key difference between a universal forwarder and a heavy forwarder is that the heavy forwarder contains the full parsing pipeline, performing the identical functions an indexer performs without actually writing and indexing events on disk. This enables the heavy forwarder to understand and act on individual events such as masking data, filtering, and routing based on event data. Since the add-on application has a full Splunk Enterprise installation, it can host modular inputs that require a full Python stack for proper data collection, or act as an endpoint for the Splunk HTTP Event Collector (HEC).

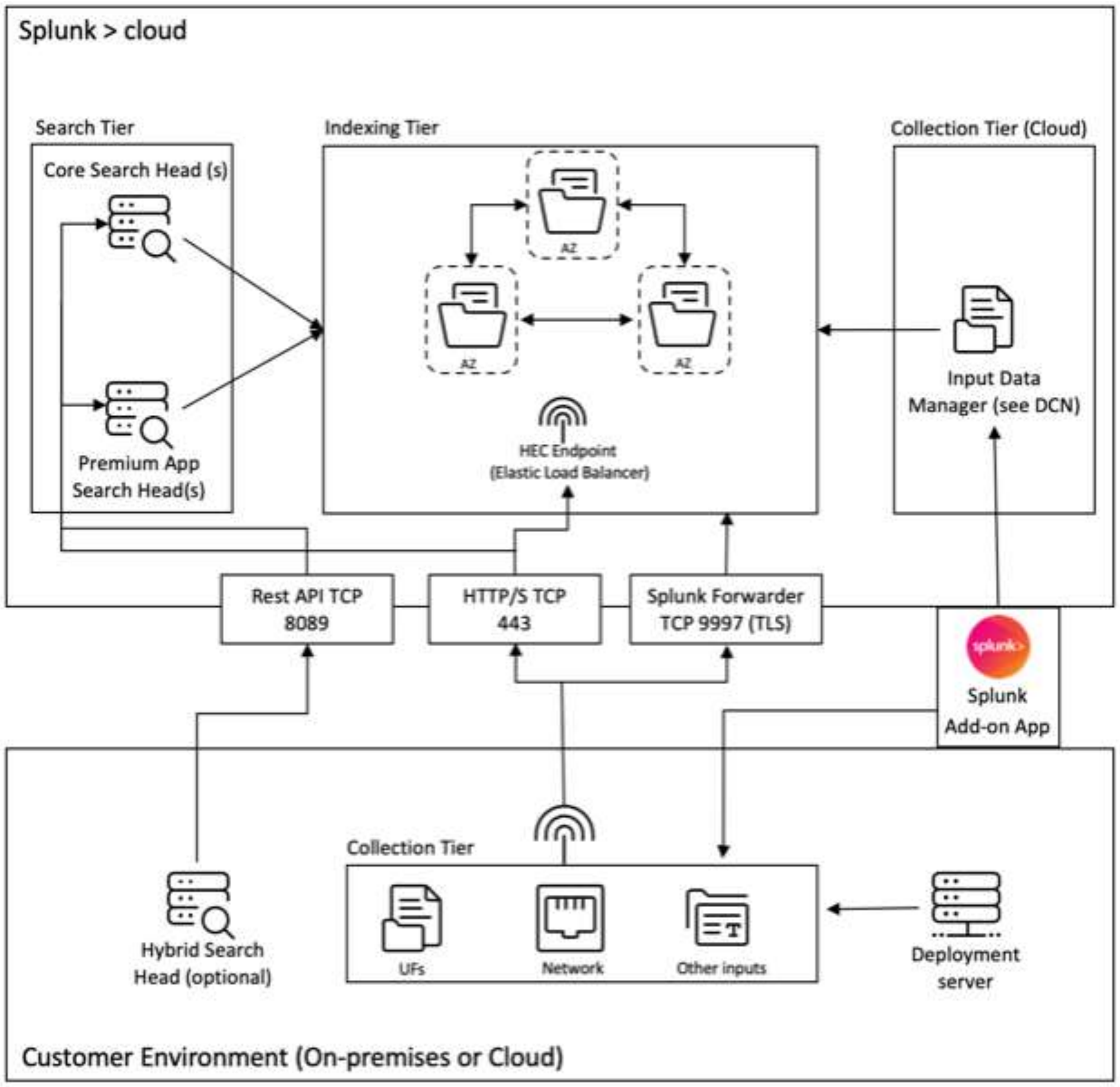
Once the data is collected, it is indexed or processed and stored in a way that makes it searchable.

The primary way for customers to explore their data is through search. A search can be saved as a report and used to power dashboard panels. Searches are the extract information from your data.

In general, the Splunk add-on application is deployed in the Collection tier (at Splunk enterprise level), whereas our dashboarding application is deployed on the search layer (at Splunk Cloud level). On a simple on-prem setup, you can have all these three tiers on a single Splunk host (known as single server deployment).

The collection tier is much better way to use the add-on application for Splunk. There are two ways to install the add-on application. Either you can install it at the collection tier under the customer environment or you can install it at the inputs data manager under the **Splunk Cloud instance**.

Refer the following diagram to understand the Splunk deployment architecture with our add-on application:



The Inputs Data Manager (IDM) shown in the aforementioned diagram is the Splunk Cloud-managed implementation of a Data Collection Node (DCN) that supports scripted and modular inputs only. For data collection needs beyond that, you can deploy and manage a DCN in your environment using a Splunk heavy forwarder.

Splunk allows to collect, index, and search data from various sources. One way to collect data is through APIs, which allows Splunk to access data stored in other systems or applications. These APIs can include REST, web services, JMS and/or JDBC as the query mechanism. Splunk and any third-party developers offer a range of applications that enable API interactions through the Splunk modular input framework. These applications typically require a full Splunk enterprise software installation to function properly.

To facilitate the collection of data through APIs, it is common to deploy a heavy forwarder as a DCN. Heavy forwarders are more powerful agents than universal forwarders, as they contain the full parsing pipeline and can understand and act on individual events. This enables them to collect data through APIs and process it before forwarding it to a Splunk instance for indexing.

To understand more about the high level architecture of a Splunk Cloud deployment, refer **Splunk Validated Architectures**.

Modular Design

The system follows a **modular architecture**, dividing functionality into independent but interlinked modules. This modular approach improves maintainability, facilitates upgrades, and enhances reusability across different ICT domains. The major modules are:

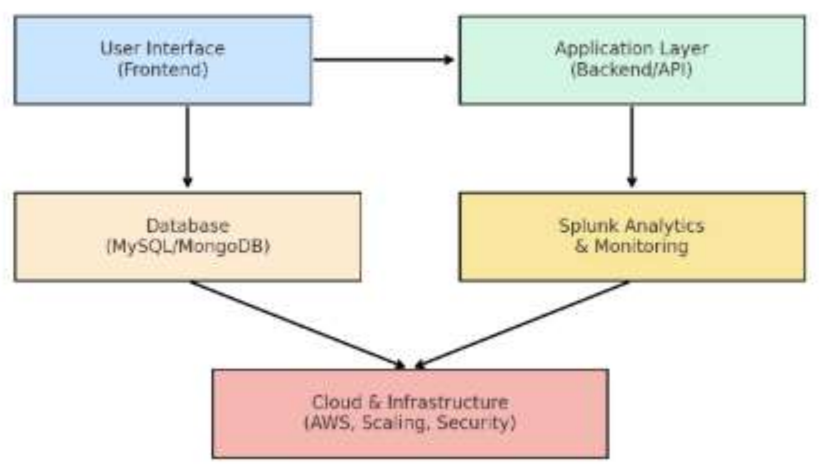
1. **User Interface (UI) Module**
    - Provides a responsive web-based interface for interaction.
    - Supports dashboards, user inputs, and report generation.
2. **Application Layer / Backend Module**
    - Implements business logic using APIs and middleware.
    - Handles authentication, authorization, and data preprocessing before storage.

3. **Database Management Module**
    - Centralized repository storing user profiles, logs, and system data.
    - Optimized for structured queries, indexing, and retrieval.

4. **Splunk Analytics and Monitoring Module**
    - Ingests raw system logs and user activity.
    - Uses **Search Processing Language (SPL)** to generate real-time analytics and visualizations.
    - Detects anomalies, login patterns, and unusual activity.
5. **Scalability and Infrastructure Module**
    - Cloud-based deployment enabling auto-scaling and redundancy.
    - Integrates load balancers to distribute requests.

System Architecture Diagram (Figure 1)

Figure 1: System Architecture Diagram


User Interface (Frontend)
Application Layer (Backend/API)
Database (MySQL/MongoDB)
Splunk Analytics & Monitoring
Cloud & Infrastructure (AWS, Scaling, Security)

Justification of Modularity:

- **Maintainability:** Each module can be modified without affecting the others.
- **Reusability:** Components (like authentication, Splunk analytics) can be reused in other ICT solutions.
- **Extensibility:** New features can be added without restructuring the entire system.

Technology Stack

Splunk integrates multiple components into its ecosystem:

- **Core Components**:
  - **Universal/Heavy Forwarders** for ingestion
  - **Indexers** for data storage and retrieval
  - **Search Heads** for queries and visualization
- **Storage Models**:
  - **Classic Storage** (local hot/warm/cold tiers)
  - **Smart Store** (decouples storage via S3-compatible cloud for scalability and cost efficiency)
- **Management Tools**:
  - Cluster Manager, License Manager, Deployment Server
- **Programming/Integration**:
  - SPL (Search Processing Language) for querying, stats, and anomaly detection
  - APIs for custom integrations

- **Deployment Options**:
  - On-premise (Single Site or Multi-Site clusters)
  - Cloud (Splunk Cloud, managed service with on-premise forwarders for ingestion)

SPL Example:

```
index=main sourcetype=access_combined status=404 | stats count by host
```

(Finds 404 errors per host and counts occurrences.)

Scalability Plan

The scalability strategy ensures the system can handle **increasing load, data volume, and user base**:

1. **Horizontal Scaling**
   - Additional application servers added behind load balancers.
   - AWS Elastic Load Balancer ensures traffic distribution.
2. **Database Scaling**
   - **Sharding** (splitting large datasets across multiple servers).
   - **Caching** with Redis/Memcached to reduce database load.
3. **Splunk Scaling**
   - Use **indexer clustering** to distribute log indexing across nodes.
   - **Search head clustering** to parallelize SPL queries.
4. **Bottleneck Management**
   - **Database Performance:** Indexing and caching mitigate slow queries.
   - **Network Latency:** Content Delivery Networks (CDNs) reduce delays.
   - **Processing Load:** Asynchronous job queues (RabbitMQ/Kafka) handle spikes in events.

5. **Cost, Performance, and Reliability Considerations**
    - **Cost:** AWS auto-scaling minimizes over-provisioning.
    - **Performance:** Caching, clustering, and distributed processing ensure low response time.
    - **Reliability:** Failover strategies and backups ensure system resilience.

Splunk and Search Processing Language (SPL)

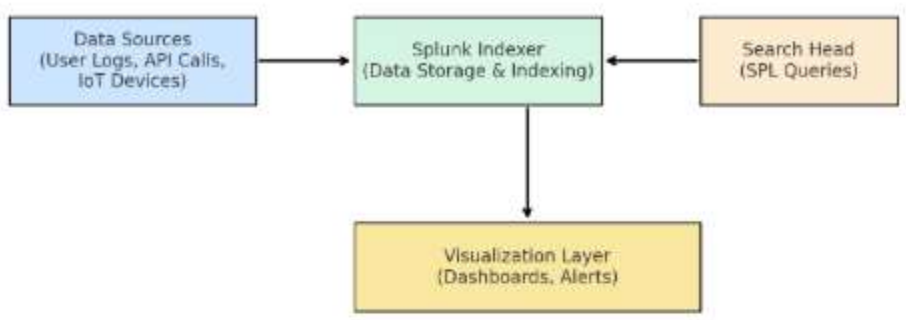
Splunk serves as the **analytics backbone** of the system. It ingests logs from the application, database, and user activity, then applies SPL queries to extract meaningful insights.

- **How Splunk Works:**
    1. **Data Ingestion:** Collects logs and metrics from sources (web server, database, sensors).
    2. **Indexing:** Stores data in indexes for fast retrieval.
    3. **Search with SPL:** SPL queries filter, aggregate, and visualize data.
    4. **Visualization:** Results displayed on dashboards with graphs, alerts, and anomaly scores.

- **Example SPL Queries:**
    - Detect failed logins:
    - index=main sourcetype=auth action=failure | stats count by user, ip
    - Monitor anomalies in login times:
    - index=main sourcetype=auth | timechart span=1h count by user
    - Track unusual data transfer:
    - index=network sourcetype=traffic bytes>50000 | stats sum(bytes) by ip

Data Flow with Splunk (Figure 2)

Figure 2: Data Flow with Splunk Using SPL


Data Sources (User Logs, API Calls, IoT Devices)
Splunk Indexer (Data Storage & Indexing)
Search Head (SPL Queries)
Visualization Layer (Dashboards, Alerts)

This integration ensures real-time monitoring, early detection of threats, and visual representation for decision-making.

Conclusion

Splunk's modular, tiered architecture enables flexible system design, supporting deployments from small single-instance servers to global multi-site clusters. With **forwarders for ingestion, indexers for storage, and search heads for analysis**, it ensures scalability, high availability, and real-time analytics. By leveraging Smart Store, clustering, and cloud integration, Splunk meets modern ICT requirements for reliability, cost efficiency, and performance, making it a robust choice for enterprise data analytics

Reference

1. **https://lipsonthomas.com/introduction-splunk/**
2. **https://www.conducivesi.com/about-splunk/splunk-architecture**
3. **https://cloudian.com/guides/splunk-big-data/splunk-architecture-data-flow-components-and-topologies/**
4. **https://www.splunk.com/en_us/pdfs/white-paper/splunk-validated-architectures.pdf**