# What is ELK:

ELK STACK

ELK (Elasticsearch, Logstash, Kibana) is a popular open-source stack developed by Elastic, a company at the core of its creation. Elasticsearch, the heart of the ELK stack, serves as a distributed search and analytics engine. The stack is primarily employed for log analysis, search, and visualization tasks.

ELK is free of cost but may pose a steeper learning curve, as users are required to set up the foundational components. The stack consists of three interconnected parts: Elasticsearch for storage and search, Logstash for data processing, and Kibana for visualizing and analyzing the data.

Various enterprises, ranging from Facebook to Netflix, utilize the ELK stack for diverse purposes, including log analysis, product and enterprise search, monitoring, and more. Elastic, the company behind ELK, was founded in Amsterdam in 2012.

# Components of ELK:

The ELK stack comprises three main components, each serving a specific purpose in log management and analytics:

## Elasticsearch:

Role: Elasticsearch is at the core of the ELK stack and functions as a distributed search and analytics engine.

Purpose: It is responsible for indexing, searching, and analyzing large volumes of data quickly and in near real-time. Elasticsearch provides powerful querying capabilities and is well-suited for log and event data.

## Logstash:

Role: Logstash is a server-side data processing pipeline.

Purpose: It ingests data from multiple sources simultaneously, transforms it according to user- defined configurations, and then sends it to a "stash" (often Elasticsearch). Logstash supports a wide variety of inputs, filters, and outputs, making it versatile for handling different types of data.

## Kibana:

Role: Kibana is a web-based visualization tool.

Purpose: It provides a user-friendly interface for exploring, analyzing, and visualizing data stored in Elasticsearch. Users can create dashboards, visualizations, and perform ad-hoc queries to gain insights into their data. Kibana is essential for real-time data exploration and log analysis**.**

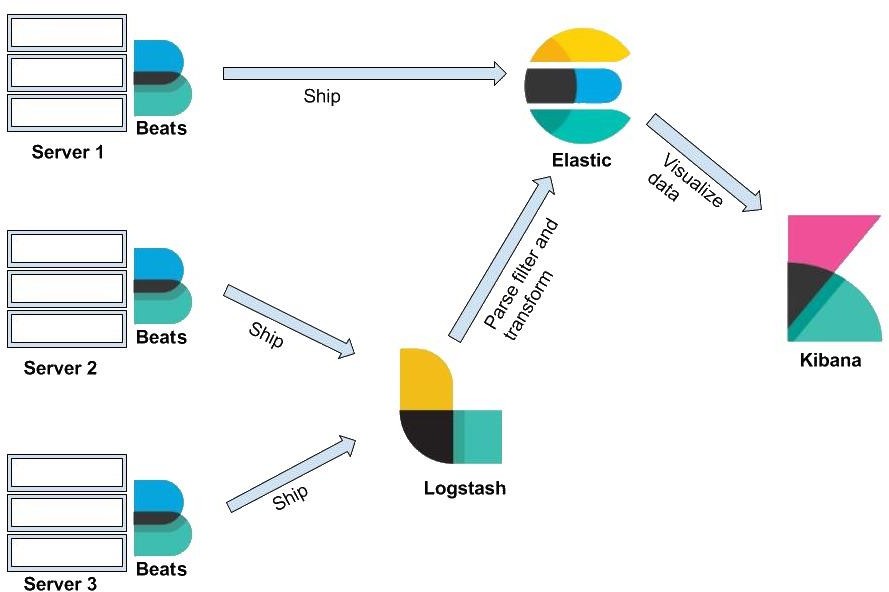
in addition to this there is another component called Beats

## Beats:

This is a new component which came in practice from the last 3-4 years. Beats is a lightweight data shipper that sits on different servers and send data to Elasticsearch or via Logstach.

Beats is a single application with no dependency

# ELK Flow:



For example:

production services are not working properly and once we get this message, we will log into servers where the application is running and we are going t0 trouble shoot logs itself.

Combining logs in a single location and searching a query on it gives very meaning full insights.

If the data we gathered is not in a usable format sometimes, then we will ship this data to the Logstash, after parsing and filtering the data logstash will ship this data to the Elastic Search and the Kibana will visualize this data.

If the data is in a usable format then it directly shipped to the Elasticsearch and followed by kibana visualization.

# Elasticsearch:

Elasticsearch is an open-source, distributed search and analytics engine designed for horizontal scalability, speed, and reliability. Key features and aspects include:

* Distributed and Scalable:

Elasticsearch is designed to be distributed, allowing it to scale horizontally by adding more nodes to a cluster. This design ensures both performance and fault tolerance.

* Full-Text Search:

It excels at full-text search, enabling users to quickly and efficiently search large volumes of data. Elasticsearch uses a scoring mechanism to rank search results based on relevance.

* Schema-Free JSON Documents:

Data in Elasticsearch is stored as JSON documents, which are schema-free. This flexibility allows for easy indexing and querying of diverse types of data.

* RESTful API:

Elasticsearch provides a RESTful API, allowing users to interact with it using HTTP methods. This makes it accessible and easy to integrate into various applications.

* Real-Time Analytics:

It offers real-time analytics capabilities, making it suitable for use cases such as log and event data analysis, monitoring, and business intelligence.

* Aggregations and Analysis:

Elasticsearch supports aggregations, allowing users to perform complex data analysis, such as calculating averages, sums, and more on large datasets.

* Indexing and Mapping:

Data is organized into indices, and mappings define the data types and structure within an index. Elasticsearch automatically indexes data for efficient search operations.

Use Cases:

Elasticsearch is commonly used for various applications, including website search engines, log and event data analysis, monitoring, and exploring large datasets for insights.

Elastic Stack (ELK):

Elasticsearch is often used in conjunction with Logstash and Kibana to create a powerful trio known as the ELK stack. Logstash is used for data processing, and Kibana is a visualization tool that complements Elasticsearch.

# Kibana:

Beginning with data exploration, the process involves thorough analysis through diverse metrics, leading to insightful visualizations using various chart types. Machine learning is then applied for anomaly detection and predicting future trends, enhancing data understanding. The inclusion of Application Performance Monitoring (APM) ensures vigilant tracking of application health. Managing users and roles ensures secure data access. The console for Elasticsearch expressions facilitates advanced querying, while the use of Timeline caters to effective manipulation of time-series data. Finally, monitoring the Elastic Stack guarantees a robust infrastructure. Altogether, these components create a powerful ecosystem for data exploration, analysis, and system management.

# Logstash:

Logstash stands as a pivotal component within the Elastic Stack, serving as a versatile data pipeline. Its primary function involves ingesting data from a myriad of sources, ranging from files and Kafka to databases. During this process, Logstash applies filters to refine the incoming data before directing it to designated destinations. One of the principal destinations is Elasticsearch, a robust search and analytics engine. The integration between Logstash and Elasticsearch is seamless, enabling efficient storage and retrieval of data.

In the broader context of Elastic Stack, Logstash plays a vital role in facilitating centralized logging—a fundamental practice for managing and analyzing logs across different components of a system or application. This becomes particularly crucial in microservices architectures, where numerous distributed services generate logs requiring cohesive aggregation and analysis.

The snippet from the ELK tutorial for beginners emphasizes Logstash's significance in centralizing logging within a microservices environment. This centralization is essential for effective monitoring and troubleshooting, ensuring logs from diverse sources are aggregated and processed cohesively.

Logstash's adaptability is evident in its ability to handle various data types—structured or unstructured—originating from different places, including data retrieved from the internet. Serving as a data ingestion and transformation tool, Logstash enables the collection, processing, and preparation of data for subsequent analysis or visualization in Elasticsearch.

In summary, Logstash plays a critical role as a conduit for data integration within the Elastic Stack. It ensures a smooth flow of data from diverse sources to Elasticsearch, where it can undergo comprehensive analysis and be visualized using tools like Kibana.

# Beats:

Beats, part of the ELK (Elasticsearch, Logstash, Kibana) stack, are lightweight, purpose- specific data shippers. Installed as agents on servers, they excel in collecting and transmitting diverse data types, including system metrics and log files.

Upon installation, Beats operate seamlessly, efficiently gathering information from the host machine. They play a crucial role in the data pipeline by sending this information to either Logstash or Elasticsearch for further processing and analysis.

The versatility of Beats lies in their ability to adapt to different use cases. Whether it's shipping logs, metrics, or other types of data, Beats streamline the process. Logstash can be employed for parsing or transforming the collected data before it is dispatched to Elasticsearch. Alternatively, Beats can directly route data to Elasticsearch, simplifying the architecture.

The swift installation and minimal configuration requirements make Beats a pragmatic choice for organizations seeking a rapid and effective means of collecting and transmitting data across their infrastructure.

## Family of beats:

The Beats family encompasses a set of purpose-built, lightweight data shippers developed by Elastic. Each Beat is designed for specific use cases, facilitating efficient data collection and transmission within the ELK (Elasticsearch, Logstash, Kibana) stack. Some notable members of the Beats family include:

1. **Filebeat**: Specialized in forwarding log files, Filebeat simplifies the process of collecting and forwarding log data from various sources.
2. **Metricbeat**: Focused on gathering system metrics, Metricbeat monitors and ships information about CPU, memory, disk, and network usage, providing valuable insights into system performance.
3. **Packetbeat**: Designed for network data analysis, Packetbeat captures and analyzes network packet data to monitor application-level protocols and detect issues.
4. **Heartbeat**: Primarily used for monitoring uptime, Heartbeat checks the availability of services and systems by sending periodic requests and tracking response times.
5. **Winlogbeat**: Tailored for Windows environments, Winlogbeat collects and ships Windows event logs to centralize monitoring systems.
6. **Auditbeat**: A Beat dedicated to collecting Linux audit framework data, Auditbeat assists in monitoring security-related events on Linux systems.
7. **Functionbeat**: Geared towards serverless environments, Functionbeat facilitates the shipping of events from cloud services like AWS Lambda to Elasticsearch.

# XPACK:

X-Pack is an Elastic Stack extension that provides security, alerting, monitoring, reporting, machine learning, and many other capabilities. By default, when you install Elasticsearch, X- Pack is installed.

# Elasticsearch Architecture:

Elasticsearch is a real-time distributed search and analytics engine that is built on top of Apache Lucene. It is designed for full-text search and is commonly used for log and event data analysis, as well as for other types of structured and unstructured data. Here is an overview of Elasticsearch architecture and its key components:

Elasticsearch is designed to operate in a distributed environment, allowing it to scale horizontally by adding more nodes to the cluster.Data is distributed across multiple nodes for improved performance, fault tolerance, and scalability.Elasticsearch is known for its powerful full-text search capabilities, making it suitable for searching and analyzing large volumes of textual data. Elasticsearch is built on top of Apache Lucene, a high-performance, full-featured text search engine library written in Java.

Elasticsearch is schema-free, meaning that it can index and search data without requiring a predefined schema. This flexibility is beneficial for handling diverse and evolving data. While Elasticsearch is excellent for searching and analyzing data, it is not typically used as a primary data source. Instead, it is often used in conjunction with other data storage solutions.Elasticsearch provides a RESTful API that allows users to interact with the system using HTTP requests. The responses are typically in JSON format, making it easy to integrate with various programming languages and applications.

While it's possible to install Elasticsearch on a single server for testing or small-scale setups, it's not recommended for production environments due to scalability and fault-tolerance considerations. In a production environment, Elasticsearch is usually installed on multiple nodes to distribute the workload, provide fault tolerance, and ensure scalability.

## Elasticsearch Node Types:

* Coordinating Node: Handles incoming requests, distributes the requests to the relevant data nodes, and aggregates the results.
* Data Node: Stores and indexes the data, performs search and retrieval operations.
* Master Node: Manages the cluster, including index and node management, and coordinates cluster-wide activities.
* Ingest Node: Allows data transformation and enrichment before indexing.

# Data in the context of Elasticsearch:

## Fields:

Fields represent individual units of information in Elasticsearch.

Each field has a defined data type, and it can also be complex, supporting complex data types.

Elasticsearch allows the use of multifields, enabling multiple representations of a single value.

## Index:

In Elasticsearch, an index is similar to a database in relational databases. It is a way to organize and store data.

Indices can contain multiple types of documents with a similar structure.

## Document:

A document is the smallest unit of data in Elasticsearch. Each piece of data within an index is called a document. Documents in Elasticsearch are represented as JSON objects.

A document can be compared to a row in a table in a traditional relational database.

# Mapping:

Definition: Mapping in Elasticsearch is the process of defining how documents and their fields are stored and indexed.

## Types:

Static Mapping: Fields are explicitly defined with their data types. This is suitable when the structure of the data is known in advance.

Dynamic Mapping: Elasticsearch automatically detects the data type of fields based on the provided data. This is useful when dealing with dynamic or evolving schemas.

## Properties:

Each field in a document has properties like data type, analyzer, and index options specified in the mapping.

## Sharding:

Definition: Sharding is the process of breaking down the index into smaller, more manageable units called shards.

## Purpose:

Allows horizontal scaling by distributing data across multiple nodes in a cluster. Improves performance by parallelizing search and indexing operations.

## Number of Shards:

The number of primary shards is set at the time of index creation and cannot be changed afterward.

Elasticsearch distributes these primary shards across the available nodes in the cluster.

# Replica:

Let us understand replicas through below example:

Replicas are copies of indexe’s shards Nodes:

NodeA:

Shard a: Primary shard residing on NodeA. Shard b: Primary shard residing on NodeA.

Replica c: Synchronized replica of Shard a, hosted on NodeB for fault tolerance. Replica d: Synchronized replica of Shard b, hosted on NodeB for fault tolerance. NodeB:

Replica a: Synchronized replica of Shard a, hosted on NodeA for fault tolerance. Replica b: Synchronized replica of Shard b, hosted on NodeA for fault tolerance. Shard c: Primary shard residing on NodeB.

Shard d: Primary shard residing on NodeB.

## Explanation:

Primary Shards (a, b, c, and d):

Primary shards contain the main data for an index and are distributed across nodes for scalability.

Replicas (c, d, a, and b):

Replicas are synchronized copies of their respective primary shards.

Replicas ensure fault tolerance by allowing data retrieval from other nodes if the primary shard's node is unavailable.

## Synchronization:

Elasticsearch ensures synchronization between primary shards and their replicas in real-time.

Any changes made to a primary shard are replicated to its associated replicas, maintaining consistency across nodes.

## Fault Tolerance Example:

If NodeA goes down, Elasticsearch can still retrieve data from NodeB, as it hosts replicas of the primary shards from NodeA (c and d).

Similarly, if NodeB goes down, data can still be retrieved from NodeA due to the replicas of primary shards (a and b) hosted on NodeB.

# Search Query Execution:

Simply put, when a client sends a search query cluster, it hits a node. This node, which has received the request from the client, is called a coordinating Node, which means that now this node is responsible for sending out the query to other nodes, assembling the results, and responding to the client

Now, as the coordinating node itself contains the shard upon which the search query is to be performed, it first performs the search query first on itself, then sends out the query to every other shard or replica in the index.

## Download Elasticsearch:

Go to the official Elasticsearch download page: Elasticsearch Downloads.

Choose the desired version and download the Elasticsearch ZIP file tailored for Windows.

## Extract the ZIP File:

After the download completes, unzip the contents to a directory of your choice on your machine.

## Configure Elasticsearch (Optional):

Navigate to the config directory within the Elasticsearch installation folder. Edit the elasticsearch.yml file using a text editor like Notepad.

Adjust settings such as cluster name, node name, network configurations, etc., as per your requirements.

## Install Java:

Elasticsearch relies on Java. Ensure that Java is installed on your computer. Download Java from the official Oracle website or consider using OpenJDK. **Start Elasticsearch:**

Open a command prompt in the Elasticsearch installation directory. Execute the following command to launch Elasticsearch:

## Command:

bin\elasticsearch.bat

Wait for Elasticsearch to initialize. Check the command prompt for log messages indicating a successful startup.

## Access Elasticsearch:

Open a web browser and go to http://localhost:9200/.

If Elasticsearch started correctly, you'll receive a JSON response displaying information such as the cluster name, version, and more.

## Stop Elasticsearch:

halt Elasticsearch, return to the command prompt where Elasticsearch is running and press Ctrl+C. Confirm the termination.

# Postman App Installation and Usage:

Postman is a powerful tool for API development and testing, allowing you to send HTTP requests, view responses, and manage your APIs in a user-friendly interface. Here's a guide on how to use Postman for working with HTTP requests:

## Installation and Setup:

Follow the steps in the previous response to install and launch Postman on your machine.

## Creating a New Request:

Open Postman, and you'll be greeted with the Postman workspace. Click on the "New" button in the top left corner to create a new request. Give your request a name and save it in a collection if desired.

## Choosing the HTTP Method:

Select the HTTP method for your request (GET, POST, PUT, DELETE, etc.) from the dropdown menu next to the URL field.

## Adding Request URL:

Enter the URL of the API endpoint you want to interact with in the URL field.

## Headers:

Navigate to the "Headers" tab to add any required headers for your request. Common headers include Content-Type, Authorization, etc.

## Request Body (if applicable):

If your request requires a request body (for POST, PUT, etc.), go to the "Body" tab.

Select the data format (raw, form-data, x-www-form-urlencoded, etc.) and enter the necessary data.

## Params (if applicable):

In the "Params" tab, you can add query parameters for your request.

## Sending the Request:

Click the "Send" button to send the request.

View the response in the "Body" tab. You'll see the raw response, headers, and other details.

## Saving Responses:

If you want to save a response for future reference, you can click on the "Save Response" button.