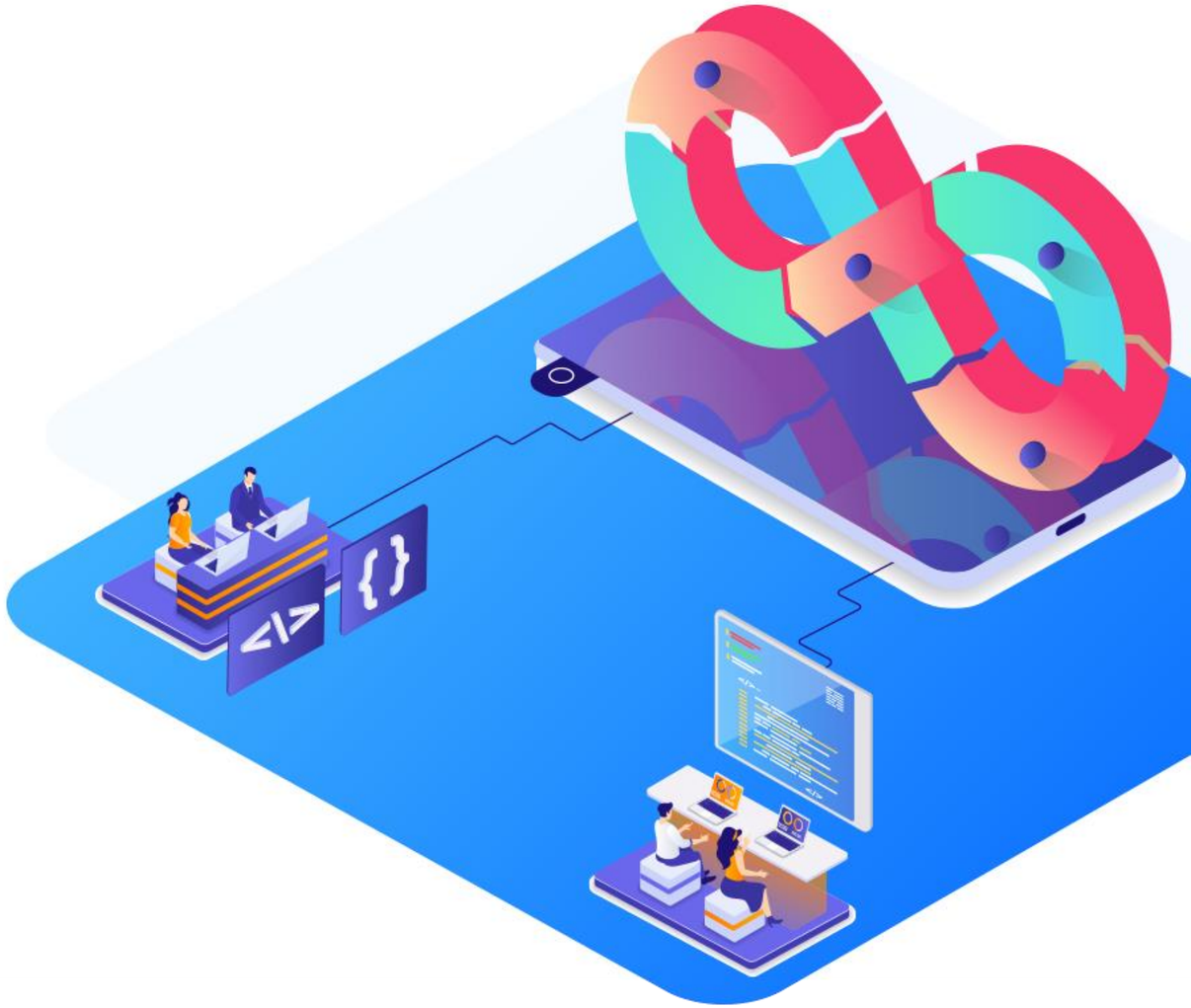
Monitoring and Logging in DevOps



Centralized Logging and Monitoring

Engage and Think

You are responsible for maintaining a large-scale, distributed application that serves thousands of users daily. The application generates massive amounts of log data across multiple servers and services. Ensuring uptime, quickly identifying issues, and continuously improving system performance are critical to the project's success. However, sorting through countless logs from different sources to find relevant information can be overwhelming.

How could implementing a centralized logging system like the ELK Stack streamline troubleshooting, improve system monitoring, and help your team maintain peak performance under pressure?

Learning Objectives

By the end of this lesson, you will be able to:

- Apply the ELK Stack (Elasticsearch, Logstash, Kibana) to configure and manage centralized logging for distributed applications in a DevOps environment
- Implement structured and unstructured log formats along with appropriate log levels to optimize system monitoring and troubleshooting
- Configure Elasticsearch for log storage, retrieval, and indexing to support efficient log management
- Create visualizations and dashboards in Kibana to monitor system health, analyze logs, detect anomalies, and set up real-time alerts based on log data

Learning Objectives

By the end of this lesson, you will be able to:

- Configure Filebeat to collect and manage logs from systems and applications using advanced log collection techniques in an ELK Stack environment
- Set up an automated log processing pipeline using Apache Kafka and the ELK Stack to support integration with CI/CD pipelines

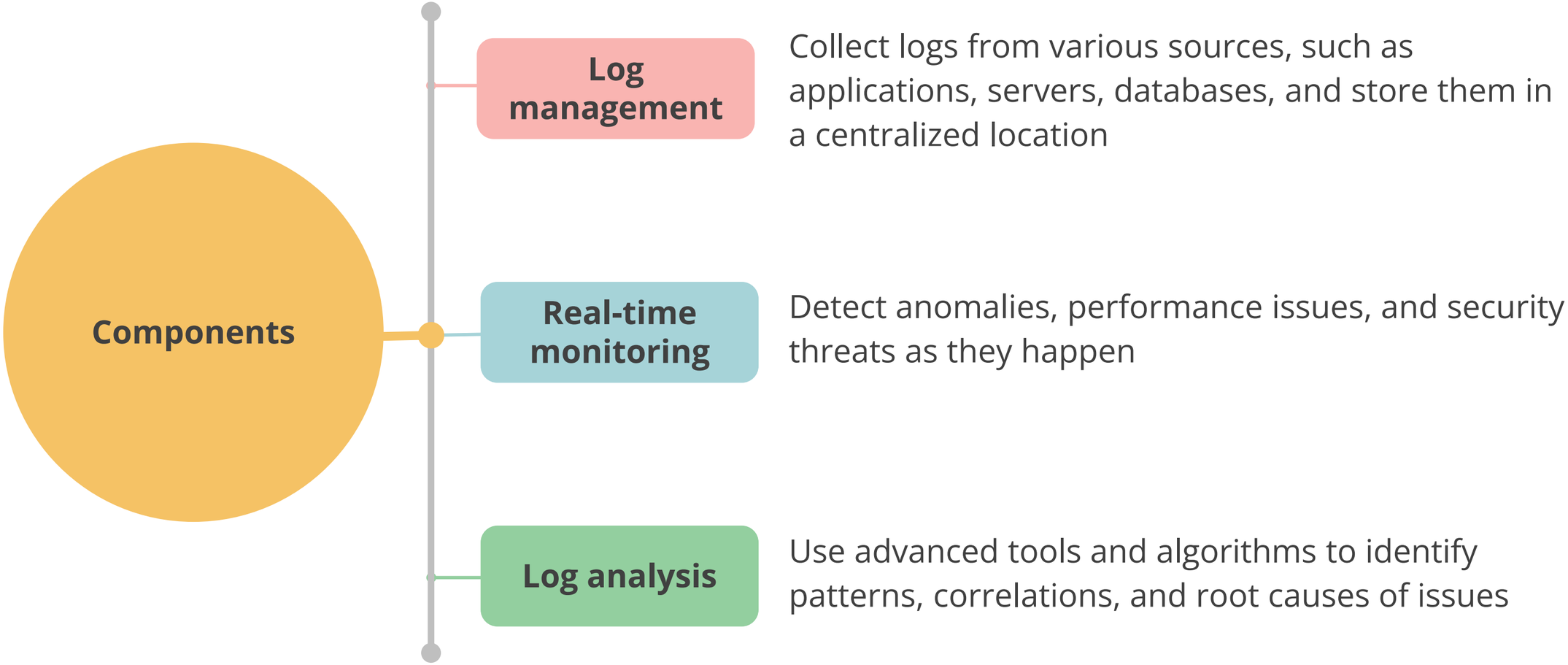
Introduction to Logging and ELK Stack

What Is Logging?

In DevOps, logging tracks and stores data about an application's or system's performance, access, changes, and how those changes affect the system.

Logging is crucial for maintaining application and infrastructure health, ensuring security, and troubleshooting issues.

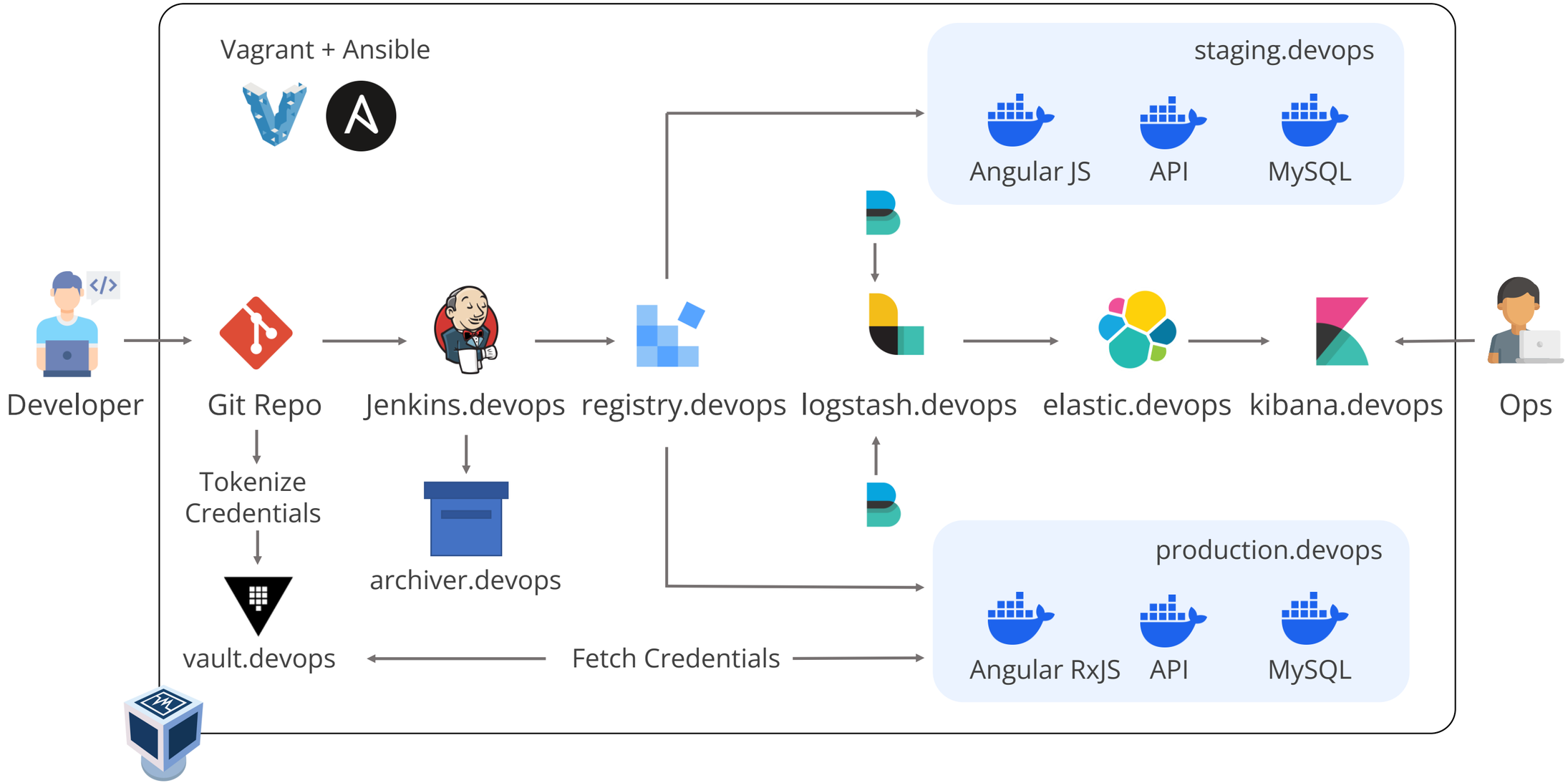
Key Components of Logging



Importance of Logging in DevOps

Provides visibility	Offers insights into application and system performance to optimize and identify issues
Enables troubleshooting	Facilitates quick diagnosis and resolution by identifying root causes
Optimizes performance	Analyzes user behavior and resource usage to drive data-informed improvements
Ensures security and compliance	Tracks access and changes, detecting unusual activities for security and regulatory needs
Automates processes	Streamlines log collection and analysis, allowing teams to focus on strategic tasks

How Logging Works in DevOps



Types of Logs

Logs are of different types, each serving a specific purpose.

Event logs	Track login attempts, app events, and system errors
Server logs	Document server activities like HTTP requests
System logs (Syslogs)	Record operating system events like startup and system messages
Authorization and access logs	Monitor access attempts and logins
Application logs	Record performance data, errors, and user activities

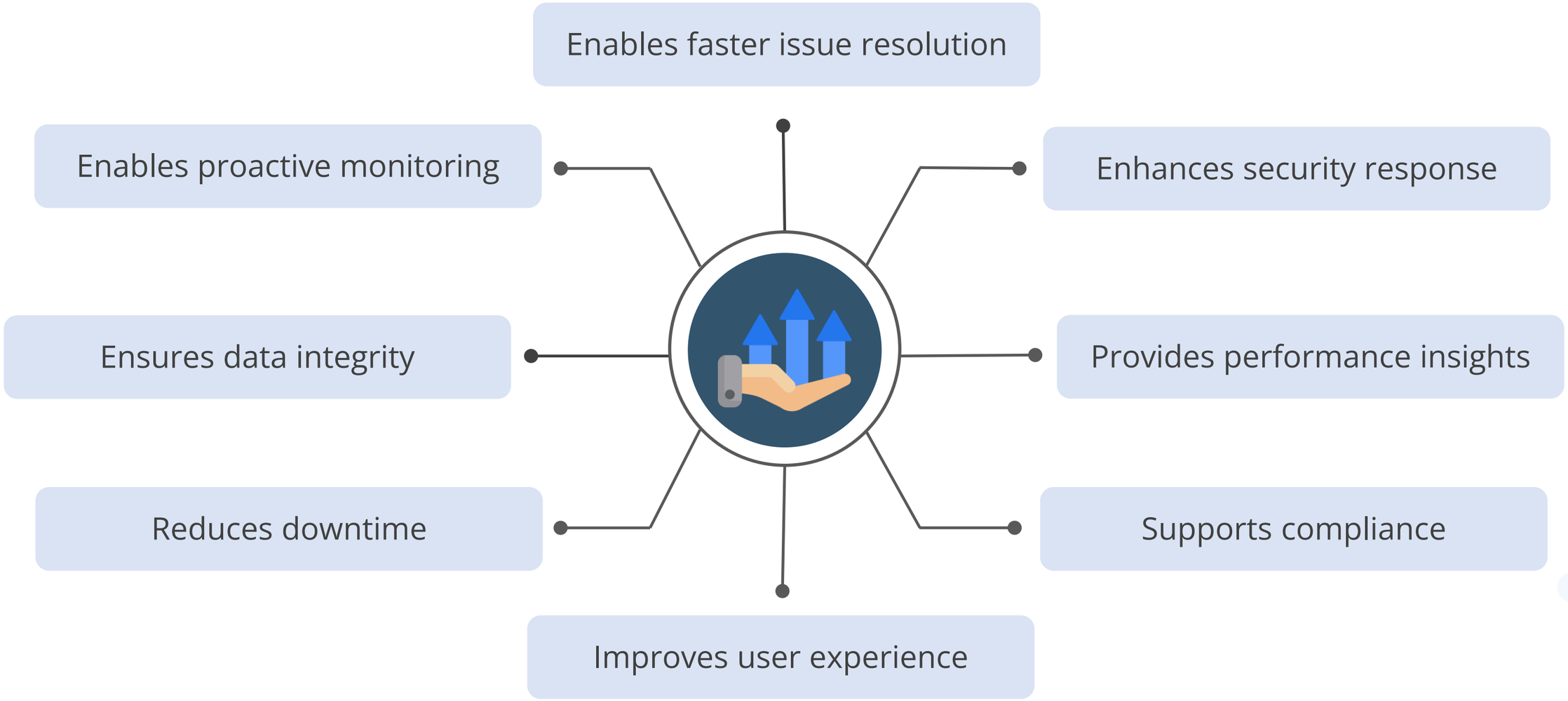
Types of Logs

Logs are of different types, each serving a specific purpose.

Security logs	Track security events like unauthorized access
Audit logs	Record system changes and who made them
Error logs	Capture error messages for troubleshooting
Transaction logs	Record system transactions for data integrity
Debug logs	Capture execution details to help developers debug apps

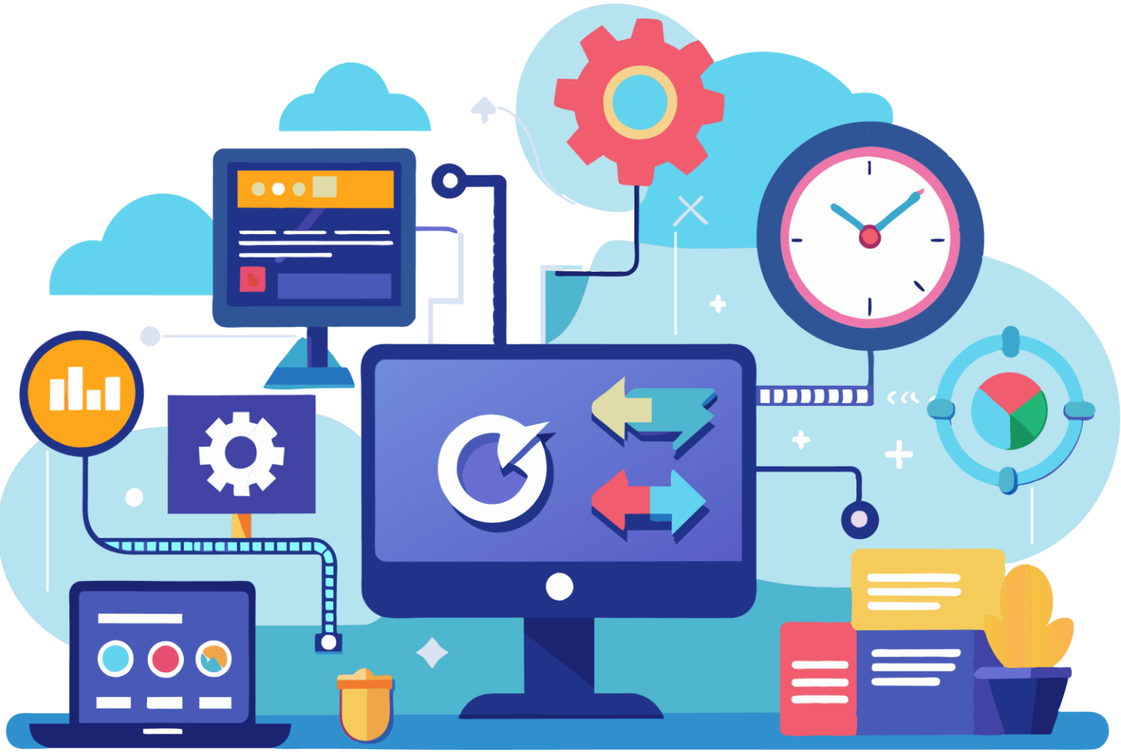
Benefits of Effective Logging

The benefits of effective logging are as follows:



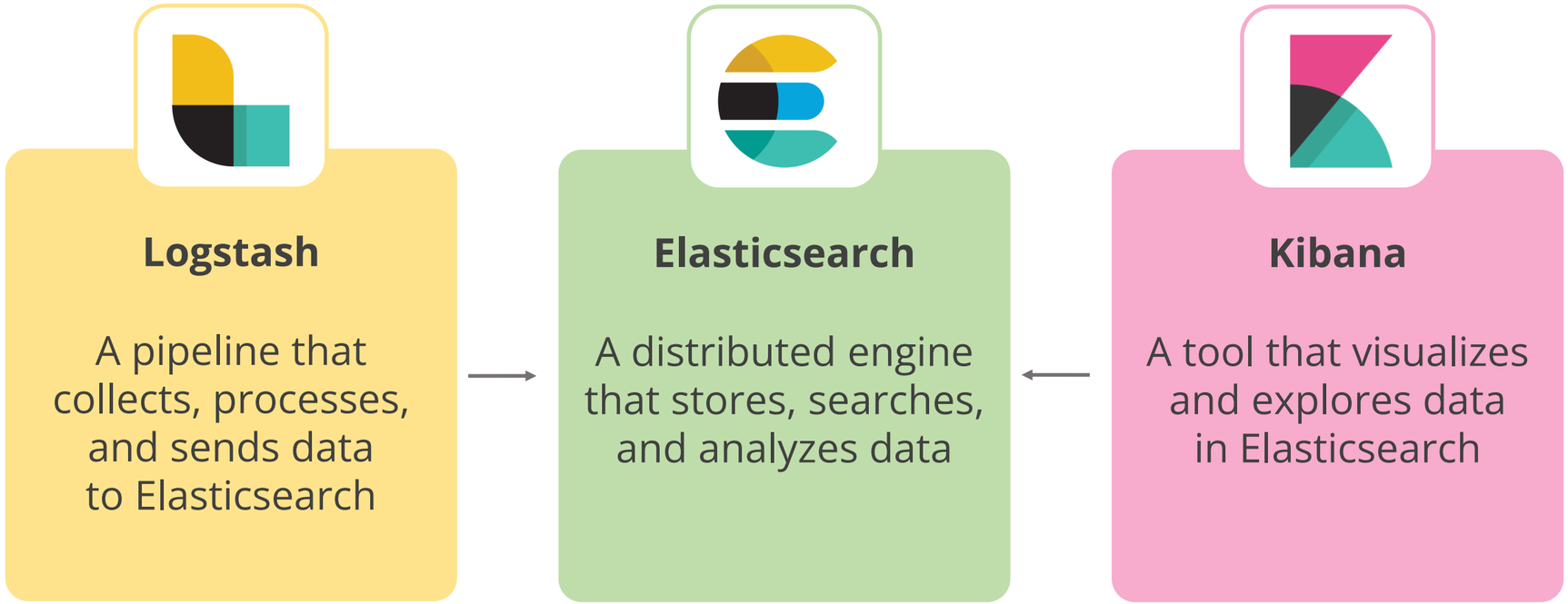
What Is ELK Stack?

ELK, short for **Elasticsearch**, **Logstash**, and **Kibana,** is a robust suite of open-source tools that search, analyze, and visualize log data in real time.



ELK Stack: Components

There are three primary components of ELK stack:

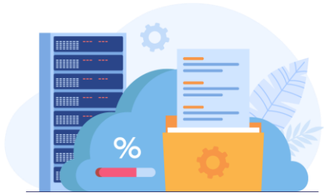


ELK Stack: Components

ELK uses lightweight **Beats** agents to ship logs and other data directly to the stack for real-time analysis.

Data collection

Data aggregation and processing



Indexing and storage

Analysis and visualization

beats

beats

Filebeat agents installed on source machines from where the logs can be shipped

Logstash

On a Dedicated machine

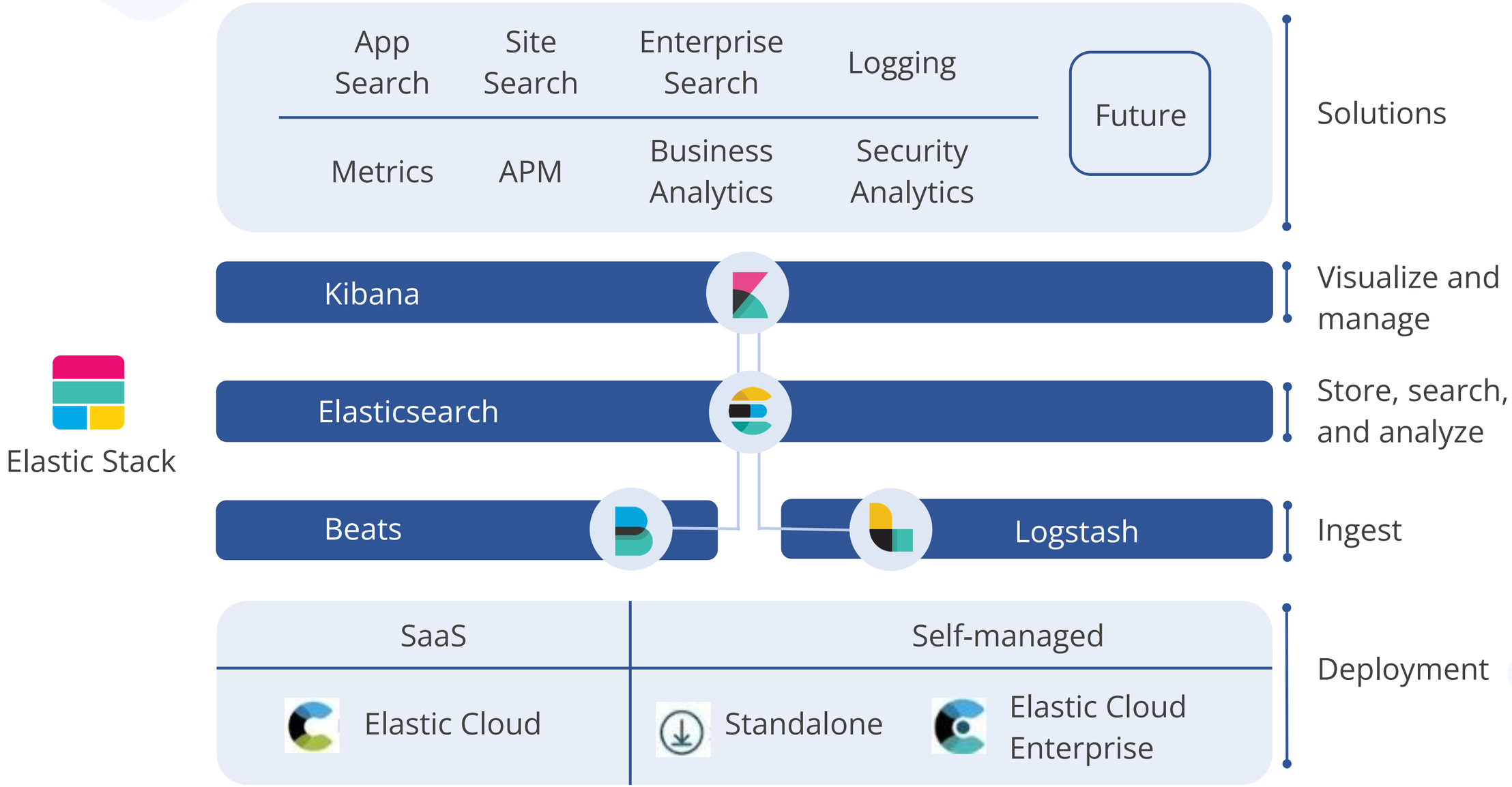
Elasticsearch

On a dedicated machine

Kibana

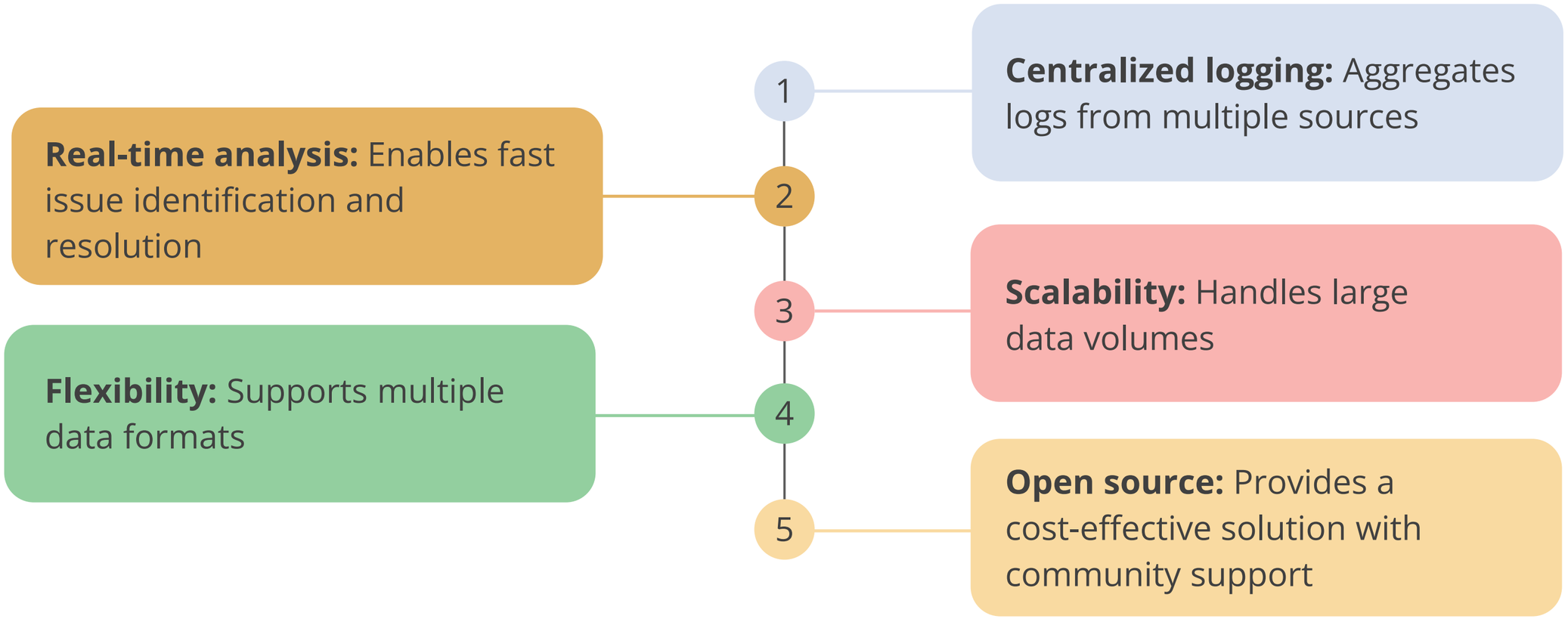
On a dedicated machine

ELK Logging Architecture

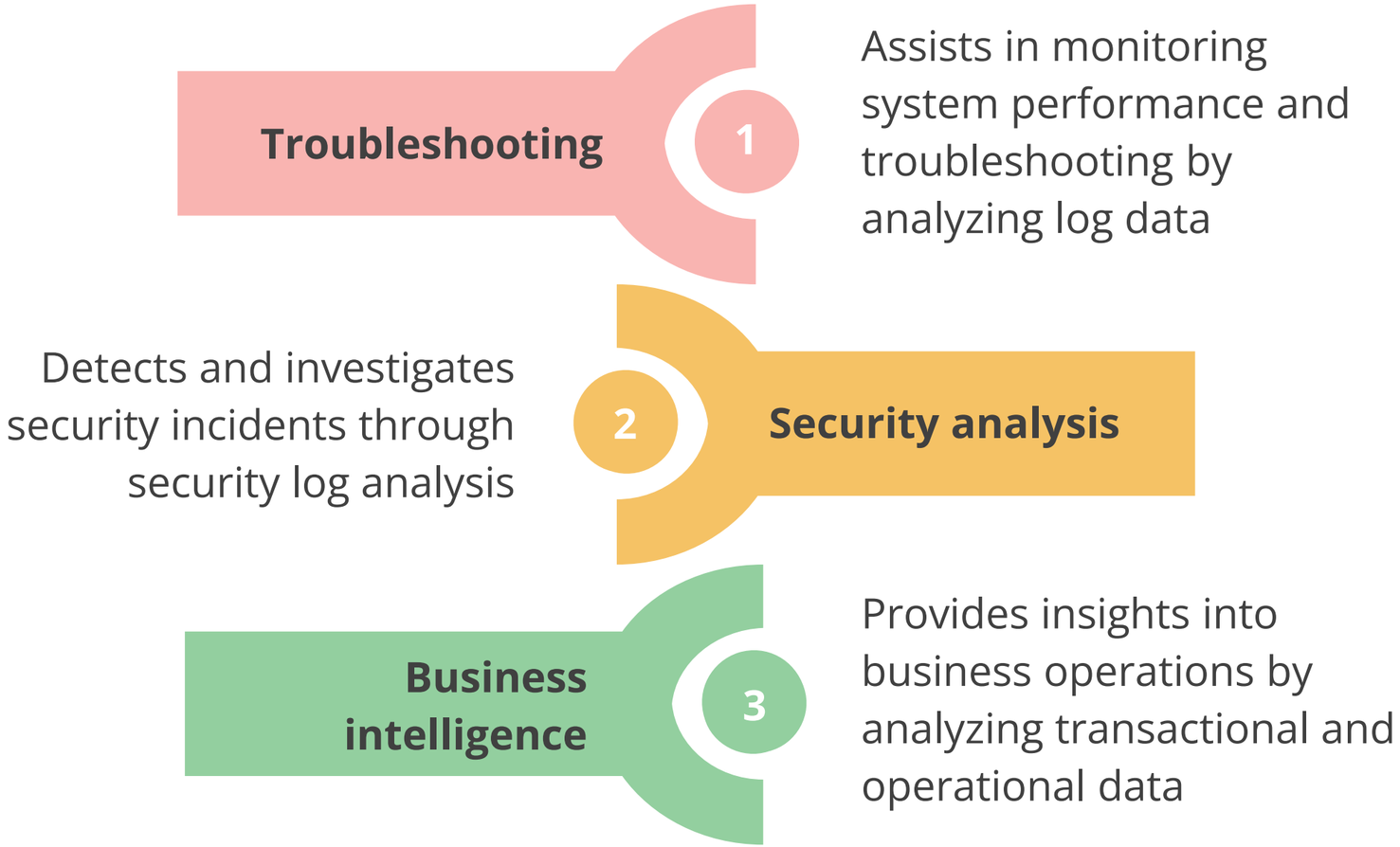


Benefits of the ELK Stack

Key advantages include:



Common Use Cases of ELK Stack

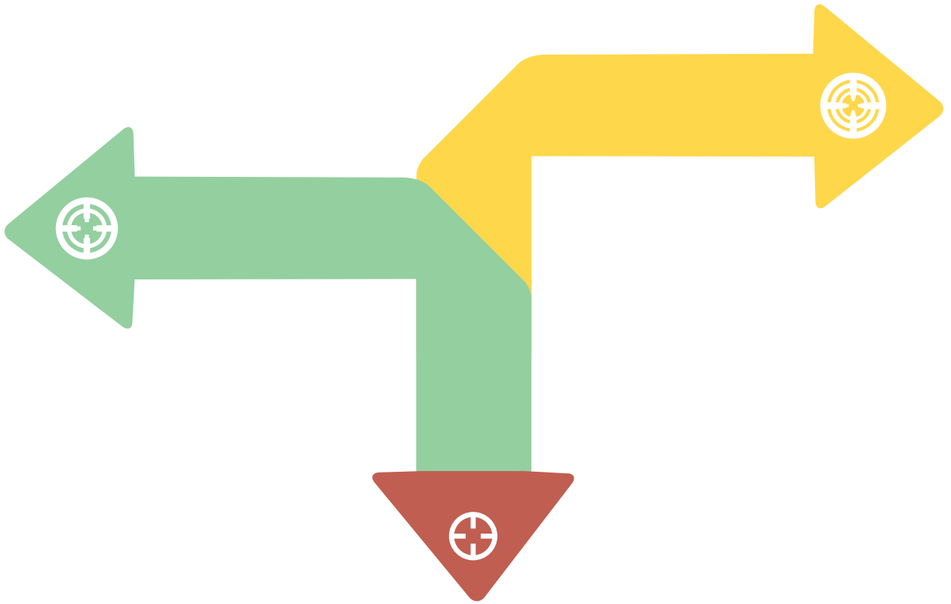


Custom Log Message Formats for ELK Stack

Creating custom log message formats for the ELK Stack involves defining how logs are structured and processed before being indexed in Elasticsearch, improving the following aspects of log data:

Benefits of Custom Log Message Formats

Key advantages include:



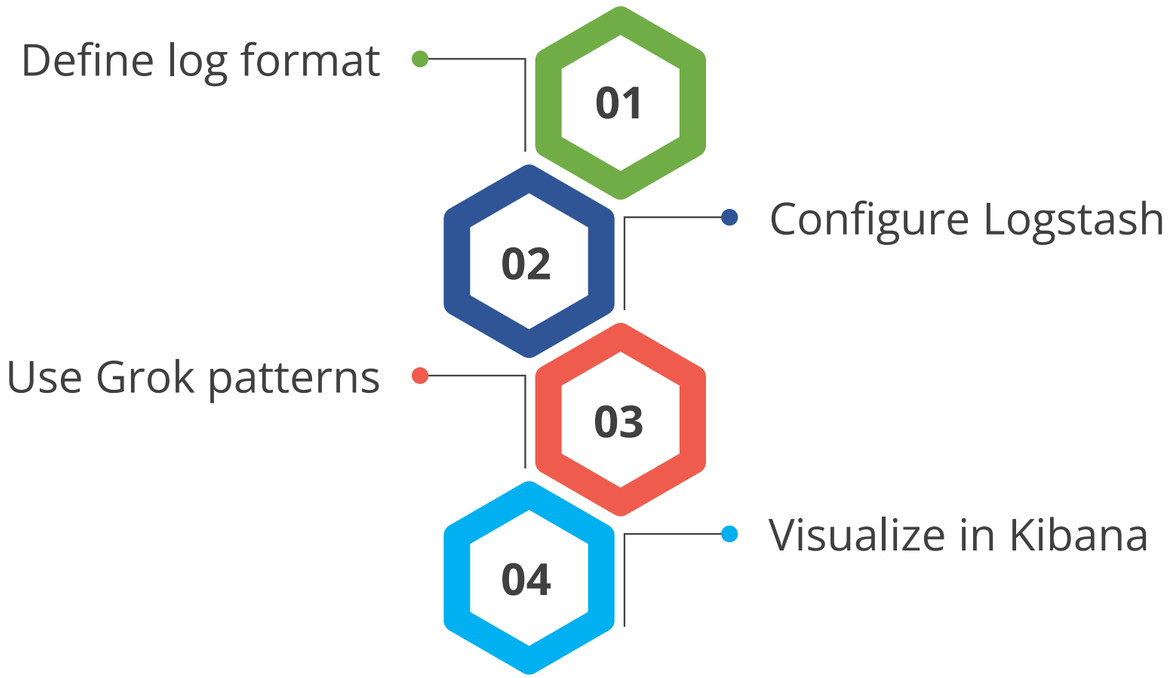
Enhanced searchability with custom fields

Improved analysis with custom formats

Better data organization

Creating Custom Log Message Formats

Here is an overview of the steps:



Creating Custom Log Message Formats: Step 1

Steps to create custom log message formats using the ELK Stack:

1. Define log format

Decide on the structure of the log messages
A common choice is the JSON format, as it offers flexibility and is easy to parse.

Example of a custom log format in JSON

```
{
 "timestamp": "2024-08-23T10:00:00Z",
 "level": "INFO",
 "message": "User login successful",
 "user": {
  "id": "12345",
  "name": "John Doe"
},
"context": {
 "ip": "192.168.1.1",
 "session_id": "abc123"
 }
}
```

Creating Custom Log Message Formats: Step 2

Steps to create custom log message formats using the ELK Stack:

2. Configure Logstash

Configure Logstash to parse and transform custom log messages

Example of a Logstash configuration file (logstash.conf)

```
input {
 file {
  path => "/path/to/your/logfile.log"
  start_position => "beginning"
  sincedb_path => "/dev/null"
 }
}

filter {
 json {
  source => "message"
 }
 date {
  match => ["timestamp", "ISO8601"]
 }
}

output {
 elasticsearch {
  hosts => ["http://localhost:9200"]
  index => "custom-logs-%{+YYYY.MM.dd}"
 }
 stdout { codec => rubydebug }
}
```

Creating Custom Log Message Formats: Step 3

Steps to create custom log message formats using the ELK Stack:

3. Use Grok patterns

Use Grok patterns to parse the logs for non-JSON log formats

Example of a Grok pattern for Apache logs

```
filter {
 grok {
  match => { "message" => "%{COMBINEDAPACHELOG}" }
}
date {
 match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
 }
}
```

Creating Custom Log Message Formats: Step 4

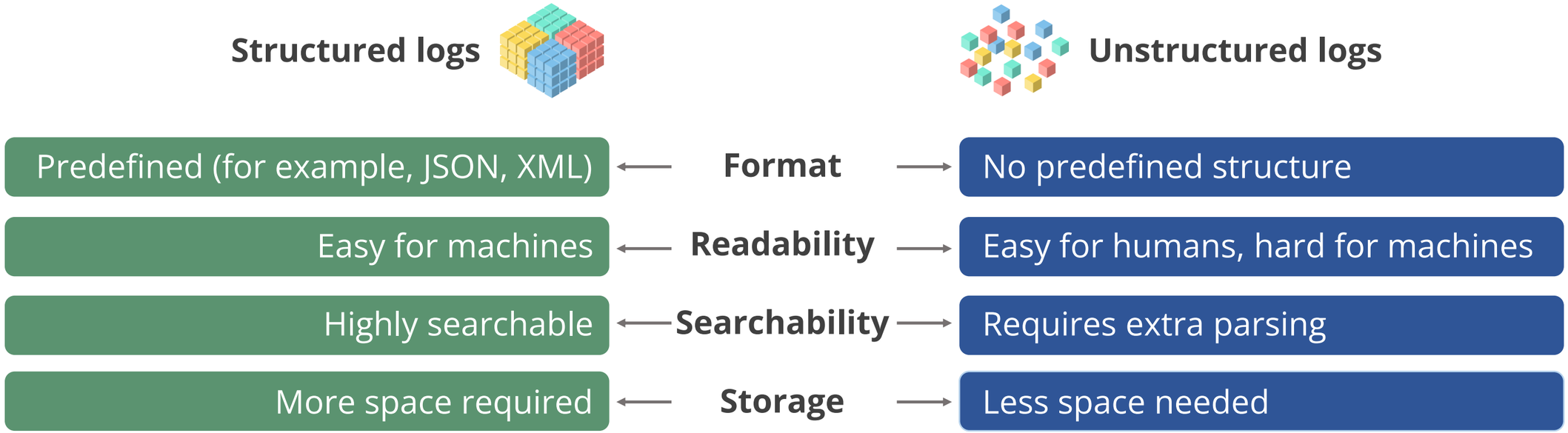
Steps to create custom log message formats using the ELK Stack:

4. Visualize in Kibana
Once the logs are indexed in Elasticsearch, leverage Kibana to build dynamic visualizations and dashboards

Tailor the Kibana index patterns to align with the fields in the custom log format for more effective analysis.

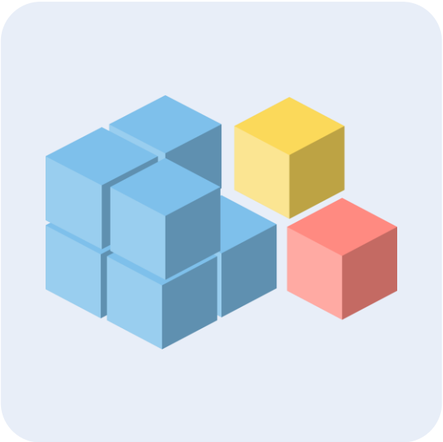
Structured vs. Unstructured Logs

Logs in the ELK Stack fall into two types:



Semi-structured Logs

These logs can consist of a mix of structured and unstructured types.



Format	Mix of structured and unstructured data; variable formatting
Readability	Human readable
Searchability	Limited; requires complex parsing
Storage	Less efficient; inconsistent for automation

Example of Structured Logs

Example

```
{
 "timestamp": "2024-08-23T10:00:00Z",
 "level": "INFO",
 "message": "User login successful",
 "user": {
  "id": "12345",
  "name": "John Doe"
},
"context": {
  "ip": "192.168.1.1",
  "session_id": "abc123"
 }
}
```

Explanation

- **timestamp:** Time of log entry
- **level:** Severity of the log
- **message:** Description of the event
- **context:** Additional details

Example of Unstructured Logs

Example

```
2024-08-23 10:00:00 INFO User login successful - UserID: 12345,
UserName: John Doe, IP: 192.168.1.1, SessionID: abc123
```

Explanation

- **2024-08-23 10:00:00:** Time of the log entry
- **INFO:** Type of the event
- **UserID, UserName, IP, SessionID:** Details about the user and session

Use Cases for Structured and Unstructured Logs

Here are the best-suited scenarios:

Structured logs

Ideal for automated log analysis, monitoring, and alerting

Unstructured logs

Ideal for simple logging needs and manual inspection

Impact on ELK Stack

Impact of log types on ELK Stack processing:

Structured logs

The ELK Stack efficiently indexes and analyzes structured logs, simplifying the creation of visualizations and dashboards in Kibana.

Unstructured logs

The ELK Stack processes unstructured logs by requiring additional steps, such as using Grok patterns in Logstash to extract meaningful data.

Benefits of Structured and Unstructured Logs

Structured logs

- Search enhancement
- Improved data organization
- Visualization analysis

Unstructured logs

- Simplicity in log generation
- Flexibility to log any data type

Centralized vs. Distributed Logging

The two key logging approaches are:

Centralized logging	Aggregates log data from multiple sources into a single, centralized location
Distributed logging	Distributes log data across multiple, decentralized locations

Centralized Logging

Benefits

- Simplified management with centralized logs
- Enhanced analytics using Kibana
- Improved security and compliance
- Efficient troubleshooting

Challenges

- Scalability issues with increasing data
- Increased risk of single point of failure (SPOF)

Distributed Logging

Benefits

- Scalability with each component handling its own logs
- Resilience through reduced risk of total system failure
- Flexibility with tailored logging for different components

Challenges

- Complex management of multiple sources
- Inconsistent data across locations

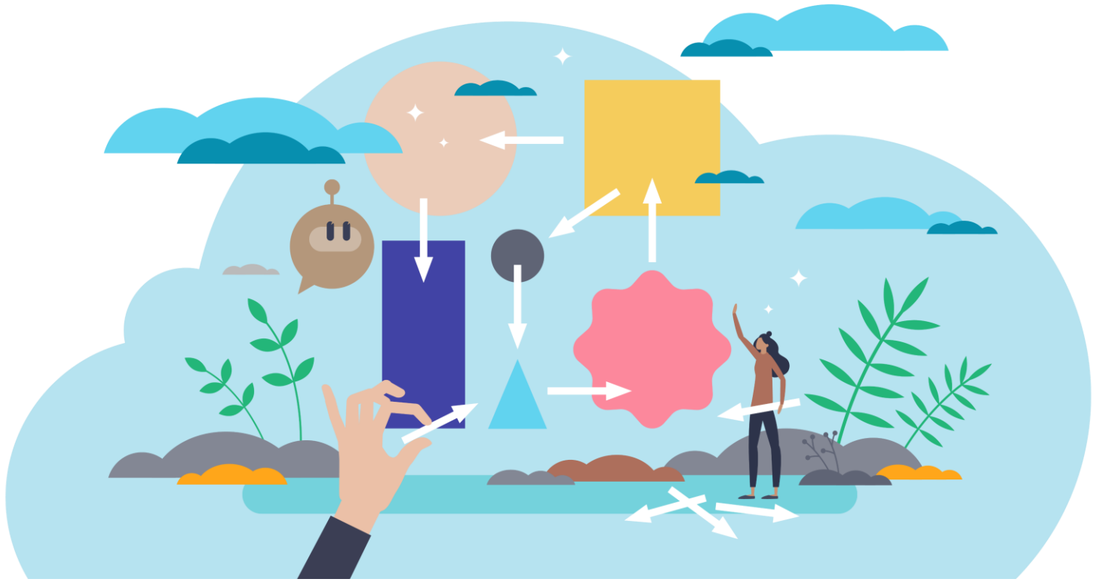
Impact of Centralized Logging on ELK Stack

The ELK Stack is well-suited for centralized logging, enabling efficient indexing, searching, and visualization of log data.

Centralized logging helps identify and resolve issues by simplifying monitoring and troubleshooting.

Impact of Distributed Logging on ELK Stack

The ELK Stack supports distributed logging but often requires additional configuration to aggregate logs from multiple sources.



It adds complexity but enhances resilience and scalability.

How to Choose the Right Approach?

Selecting the optimal logging strategy depends on the system’s needs and scalability.

Centralized logging

Ideal for environments where ease of management, security, and comprehensive analysis are top priorities

Distributed logging

Ideal for large-scale, complex environments where scalability and resilience are critical

Quick Check

You are a DevOps engineer at a healthcare company managing sensitive data. To ensure efficient log management and security, you are setting up the ELK Stack. Which logging strategy and log format would best meet the system's scalability and compliance needs?

A. Unstructured logs with distributed logging for scalability and flexibility

B. Structured logs with centralized logging and custom formats for analysis

C. Unstructured logs with centralized logging and strict log levels for filtering

D. Structured logs with distributed logging and custom log levels for control

Components of ELK Stack

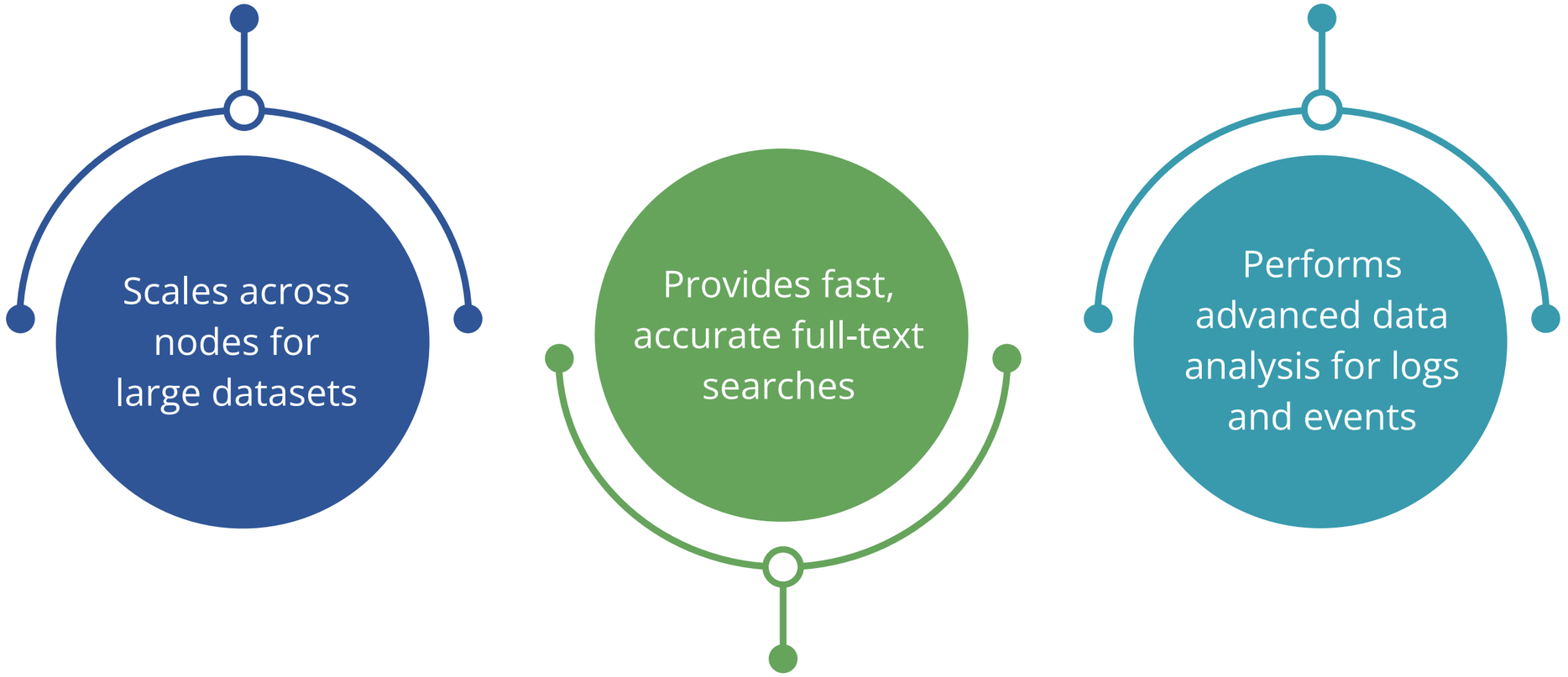
Introduction to Elasticsearch

It is a fast, distributed engine for searching and analyzing data, built on Apache Lucene.

It handles large data and provides near real-time search results.

Elasticsearch: Features

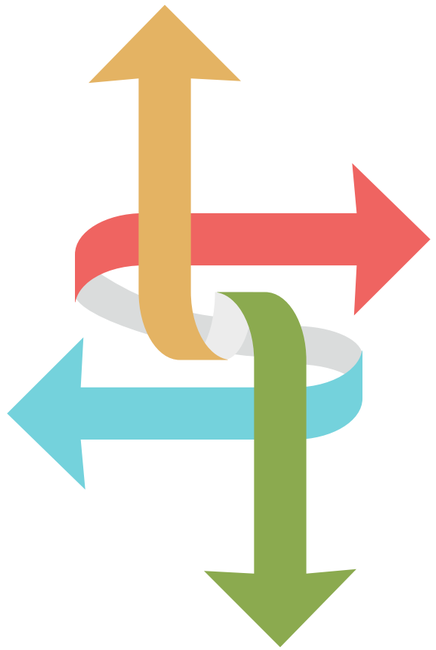
It offers several key features that make it effective for managing large datasets and performing advanced searches:



Fundamental Concepts in Elasticsearch

It operates on several core concepts:

Index: A collection of documents serving as a logical namespace



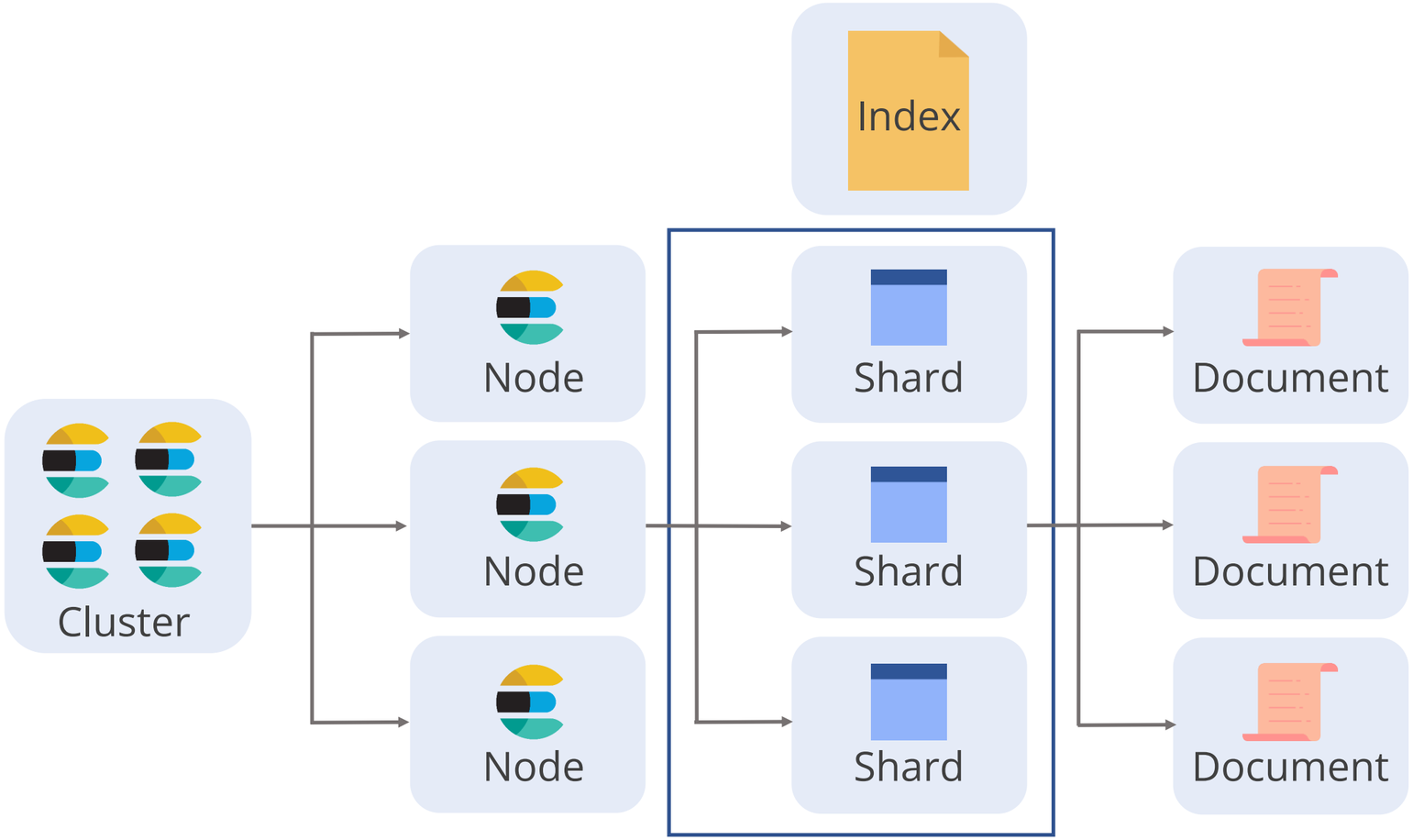
Document: The smallest unit of data, stored in JSON format

Node: A single instance of Elasticsearch that works within a cluster

Shard: A partition of an index, allowing data to be distributed across nodes

Elasticsearch Distributed Architecture

It distributes data across nodes in the form of shards. This allows it to efficiently manage growing datasets and maintain performance under heavy workloads.



Common Use Cases of Elasticsearch

Application search

Adding search functionality to applications or websites

Log and event data analysis

Collecting and analyzing log data from various sources

Business intelligence

Aggregating and analyzing business data to gain insights

Elasticsearch Functionality for Log Storage and Retrieval

It offers a variety of robust features:

Real-time data ingestion	Ingests and indexes log data in real time, making it immediately searchable
Advanced search capabilities	Provides powerful full-text search, enabling efficient searches with a range of query types
Scalability	Scales horizontally by adding nodes, allowing the system to handle growing data volumes and search loads
Distributed architecture	Distributes data across multiple nodes, ensuring high availability and fault tolerance with replicated data
Security features	Offers robust security measures (role-based access control, encryption, and auditing) to protect log data

Elasticsearch Functionality for Log Storage and Retrieval

It offers a variety of robust features:

Aggregation and analytics	Supports complex data analysis and insight generation through advanced aggregation features
Integration with ELK Stack	Integrates with Logstash for data processing and Kibana for data visualization, forming the ELK Stack
RESTful API	Provides a RESTful API for easy integration and interaction with other applications and services
Machine learning	Detects anomalies and patterns in log data using machine learning for proactive monitoring and issue detection
Storage	Uses an inverted index structure for efficient data storage and fast retrieval of log data

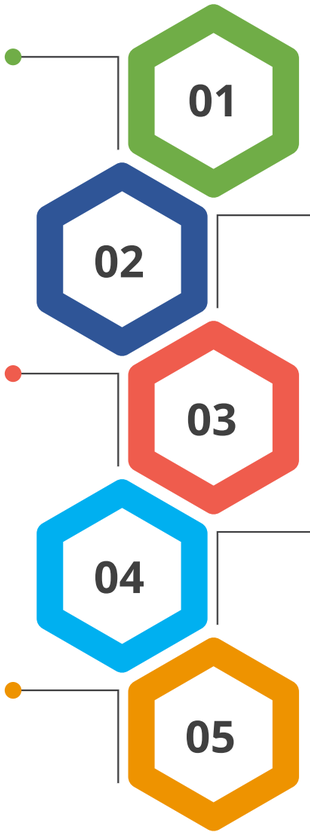
The Indexing Process

This process in Elasticsearch follows several key steps to store and retrieve data efficiently, including:

Document creation: Stores data as JSON documents with key-value pairs that represent fields and values

Mapping definition: Specifies how documents and fields are stored, including data types like text, keyword, or geo_point

Inverted index creation: Maps unique words to the documents they appear in, enabling fast searches



Index creation: Creates a logical namespace that holds documents defined using a PUT request and mappings

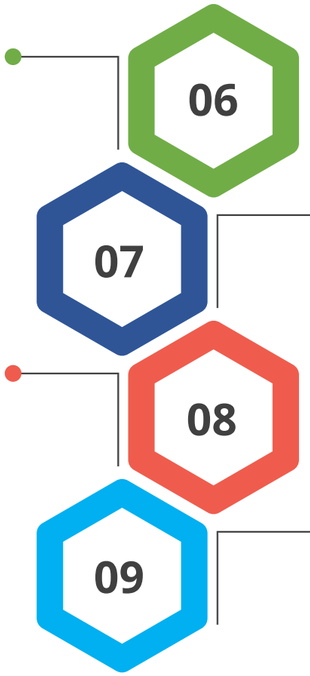
Data ingestion: Ingests documents using the RESTful API, client libraries, or tools like Logstash

The Indexing Process

This process in Elasticsearch follows several key steps to store and retrieve data efficiently, including:

Dynamic mapping: Automatically detects and indexes new fields during data ingestion

Search and retrieval: Retrieves documents quickly and ranks results based on relevance

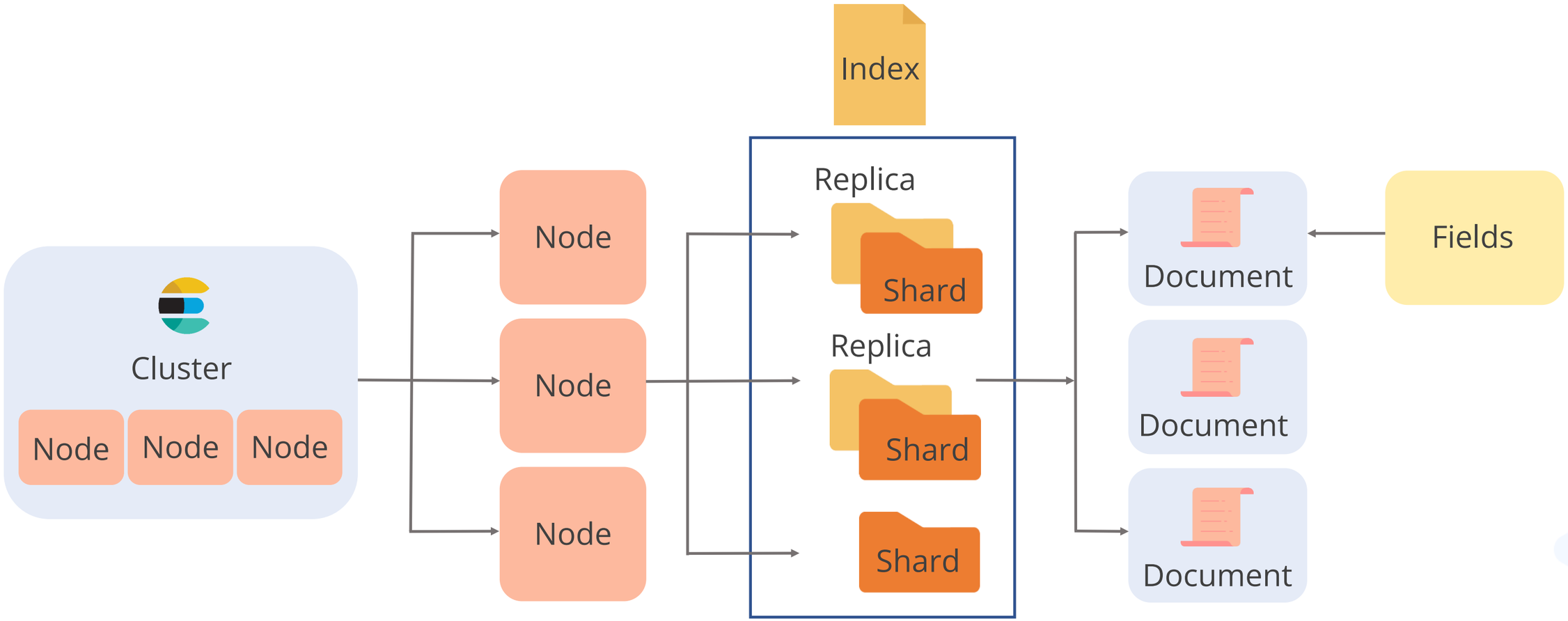


Replication: Replicates documents across nodes for high availability and fault tolerance

Index management: Provides tools for monitoring, optimizing, and managing the index lifecycle

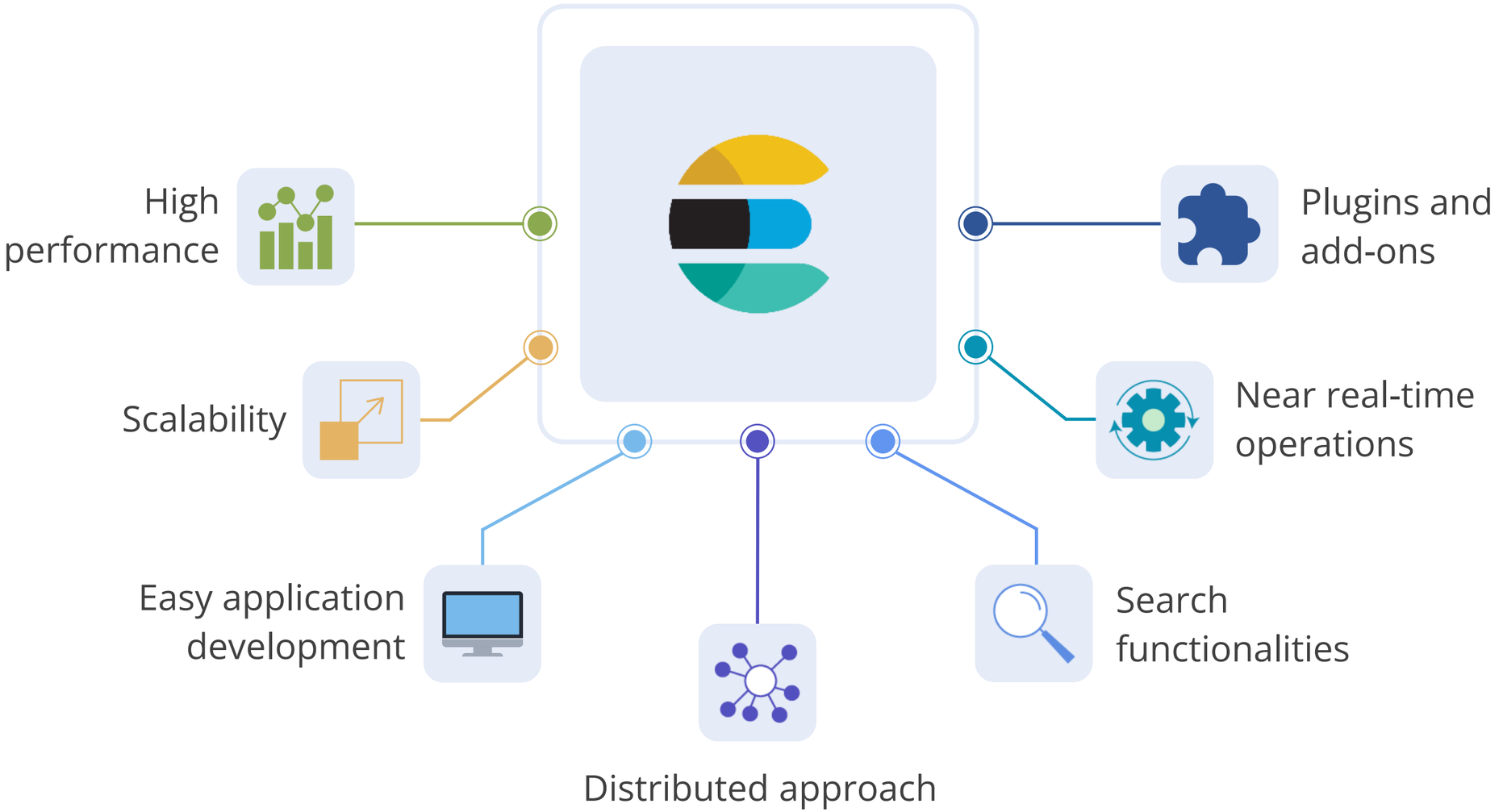
Indexing Process in Elasticsearch

It distributes documents across nodes and shards, replicates individual shard to prevent data loss even during failures, and ensures efficient handling of large datasets.



Benefits of Using Elasticsearch for Log Analytics

It offers a wide range of benefits that make it a popular choice for search and analytics.



Assisted Practice

Setting up and Configuring Elasticsearch for Log Storage **Duration: 15 Min.**

Problem statement:
You have been assigned a task to implement Elasticsearch as a monitoring tool for storing application logs, enhancing log management, and improving observability with real-time performance insights.

Outcome:
By the end of this demo, you will be able to successfully set up and configure Elasticsearch as a monitoring tool for storing application logs, enhancing log management, and providing real-time performance insights.

Note: Refer to the demo document for detailed steps
01_Setting_up_and_Configuring_Elasticsearch_for_Log_Storage

Assisted Practice: Guidelines

Steps to be followed:

1. Configure Elasticsearch for log storage

Introduction to Logstash

It is an open-source server-side pipeline that collects, processes, and sends data to destinations like Elasticsearch for indexing and analysis.

Key Features of Logstash

Collects data from multiple sources, including logs and databases

Processes and analyzes data in real time for immediate insights

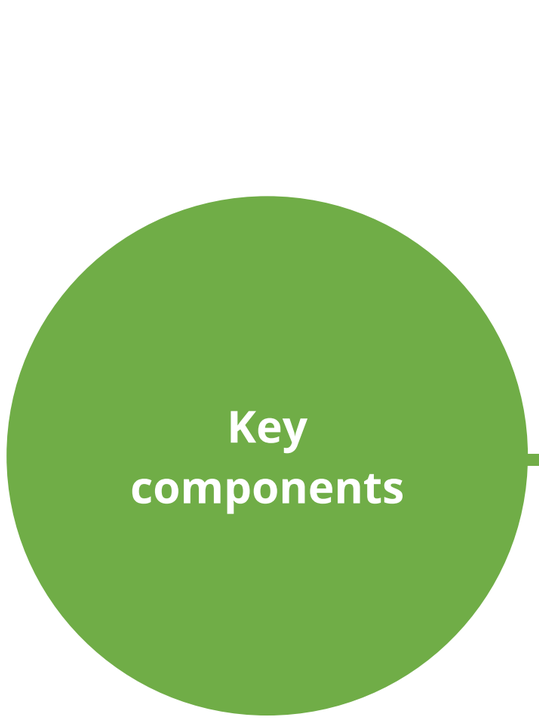
Transforms and enriches data to make it more useful

Integrates seamlessly with Elasticsearch and Kibana

Adapts to diverse needs through its flexible plugin-based architecture

Architecture of Logstash

The key components of Logstash are:



Inputs

Collects data from sources like log files, databases, and message queues via input plugins

Filters

Processes data to modify, enrich, and transform using filters like Grok, mutate, and date

Outputs

Sends processed data to destinations such as databases, Elasticsearch, or files through output plugins

Input Plugins for Collecting Logs

These are the different types of input plugins to collect logs:

File: Reads logs from files, like the **tail -f** command in Unix

TCP/UDP: Collects logs over TCP or UDP protocols

Syslog: Gathers logs from Syslog servers

HTTP: Captures logs via HTTP requests

Beats: Collects and forwards data from various platforms

Input Plugins for Collecting Logs

These are the different types of input plugins to collect logs:

- **JDBC:** Fetches data from relational databases using JDBC
- **Kafka:** Consumes messages from Kafka topics
- **S3:** Reads logs stored in Amazon S3 buckets
- **Stdin:** Collects logs from standard input, useful for testing and debugging
- **Twitter:** Fetches data from Twitter streams

Filters for Manipulating and Transforming Log Data

Logstash offers a range of filters to efficiently manipulate and transform log data.

Grok

Parses unstructured data into structured data using regular expressions

Mutate

Modifies fields, including renaming, replacing, and changing data types

Date

Parses date fields and sets the Logstash timestamp

GeoIP

Adds geographic data to logs based on IP addresses

KV

Extracts key-value pairs from fields

JSON

Parses logs formatted in JSON

Output Plugins for Sending Logs

Logstash output plugins that route logs to various destinations include:

Elasticsearch	Stores logs in Elasticsearch to make them searchable and analyzable
File	Writes events to files on disk for archiving or further processing
HTTP	Sends events to an HTTP or HTTPS endpoint for integration with web services
Kafka	Writes events to a Kafka topic, supporting real-time data processing

Output Plugins for Sending Logs

Logstash output plugins that route logs to various destinations include:

Amazon S3	Uploads log events to Amazon S3, providing scalable storage
Email	Sends email notifications based on log events for alerting purposes
Google Cloud Storage	Uploads events to Google Cloud Storage for another scalable storage option

Assisted Practice

Building Logstash Pipelines for Centralized Collection **Duration: 15 Min.**

Problem statement:
You have been assigned the task to implement Logstash to store logs in Elasticsearch for efficient management, analysis, and real-time monitoring of application logs.

Outcome:
By completing this demo, you will be able to successfully build Logstash pipelines for centralized log collection, enabling efficient management, analysis, and real-time monitoring of application logs.

Note: Refer to the demo document for detailed steps
02_Building_Logstash_Pipelines_for_Centralized_Log_Collection

Assisted Practice: Guidelines

Steps to be followed:

1. Configure the Logstash pipeline

Introduction to Kibana

It is a versatile, web-based tool designed for visualizing, exploring, and analyzing data.
It is a key component of the ELK Stack along with Elasticsearch and Logstash.

Kibana was developed in 2013 by Rashid Khan.

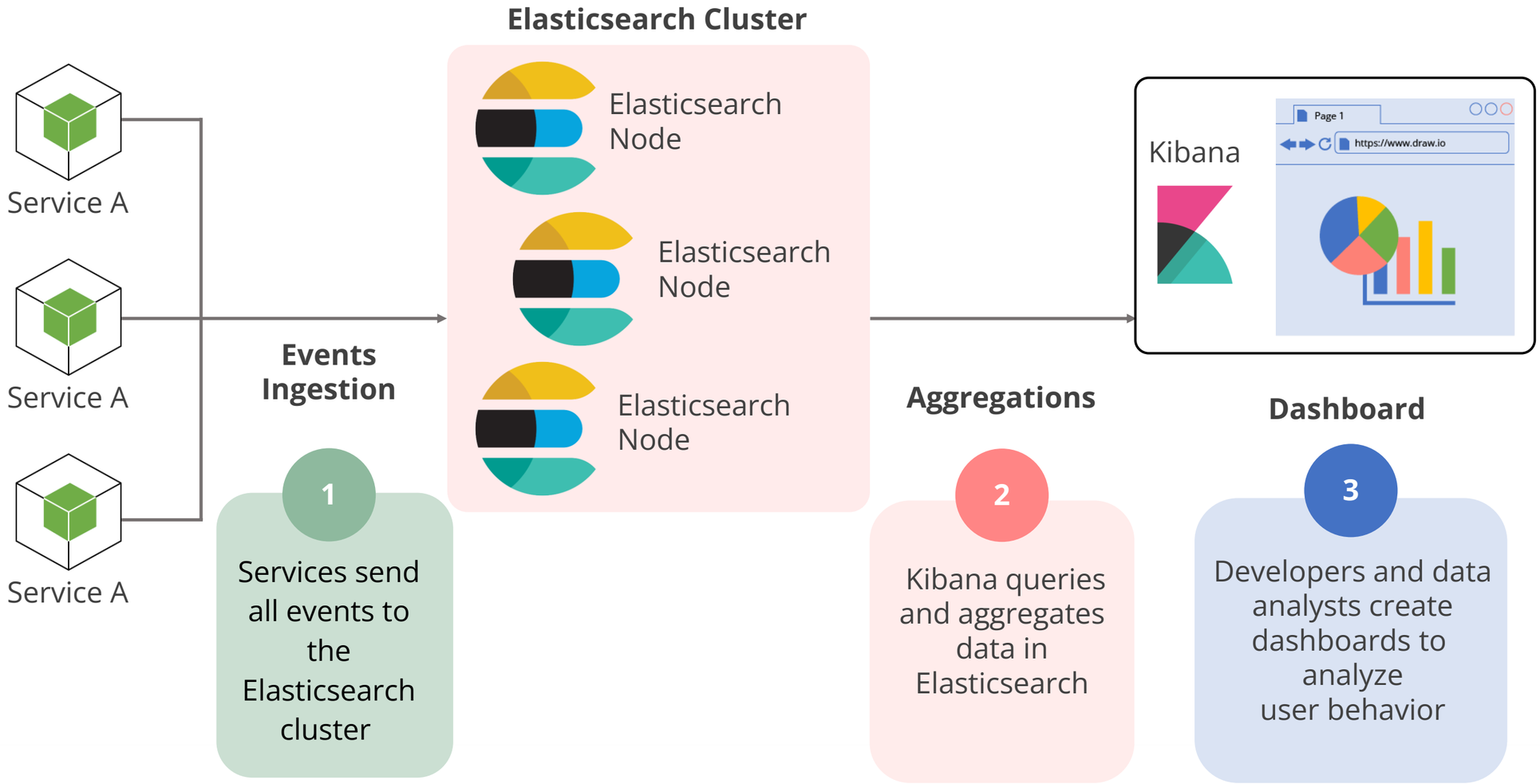
Introduction to Kibana

Here are the key functions of Kibana:

- It enables users to create visualizations, reports, and dashboards from various data sources.
- It integrates seamlessly with Elasticsearch, allowing users to search, view, and interact with data stored in Elasticsearch indices.

Kibana and Elasticsearch

This diagram shows a data flow diagram for event ingestion into Elasticsearch and analysis via Kibana to create dashboards.



Key Features of Kibana

These are Kibana’s core capabilities:

Data visualization

Provides a variety of visualization options

Dashboards

Allows the combination of multiple visualizations into interactive dashboards

Security and monitoring

Offers tools for managing, monitoring, and securing the Elastic Stack

Data exploration

Enables searching through large datasets, applying filters, and exploring fields and values

Key Features of Kibana

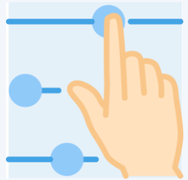
These are Kibana’s core capabilities:

Search and query capabilities

Supports intuitive free-text and field-based searches on data indexed in Elasticsearch

Real-time data updates

Ensures visualizations and dashboards stay up-to-date with the latest log data



Customizable visualizations

Allows tailoring of visualizations to meet specific needs

Plugin support

Extends functionality through various supported plugins

Steps for Creating Dashboards for Log Data Exploration

1. Identify your data

Determine which log data you want to visualize

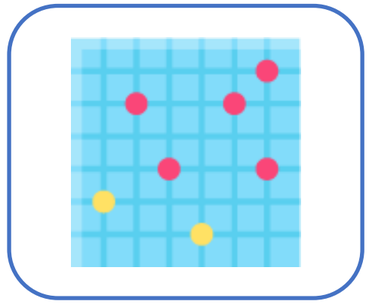
Steps for Creating Dashboards for Log Data Exploration

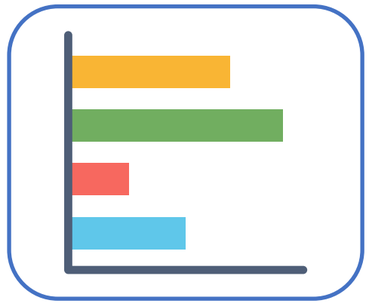
2. Create visualizations

Visualize using graphs, bar charts, pie charts, and heatmaps

Steps to create a visualization:

1. Go to the **Visualize Library** in Kibana
2. Click on **Create Visualization**
3. Select the type of visualization you want to create
4. Choose the index pattern that matches your log data
5. Configure the visualization by selecting the appropriate fields and metrics





Steps for Creating Dashboards for Log Data Exploration

3. Build dashboards

Enable multiple visualizations to be combined into a single view

Steps to create a dashboard:

1. Navigate to the **Dashboard** section in Kibana
2. Click on **Create Dashboard**
3. Add your visualizations by clicking on **Add** and selecting the visualizations you created
4. Arrange and resize the visualizations as needed
5. Save the dashboard with a descriptive name

Steps for Creating Dashboards for Log Data Exploration

4. Explore and analyze

Use the **Discover** feature to search through the data

You can apply filters, create queries, and explore fields to gain insights, and identify patterns and anomalies in the logs.

Uses of Kibana

It is an essential tool for these tasks:

Real-time log monitoring

Setting up alerts

Real-Time Log Monitoring with Kibana

These are Kibana’s key monitoring capabilities:

- Visualizing data with custom dashboards and the Discover feature to spot trends and anomalies
- Handling data ingestion with the help of index patterns and the automatic fresh feature
- Searching and filtering with the help of the query syntax and the time range selector

Setting up Alerts with Kibana

These are Kibana’s key alerting capabilities:

- Alert system with the ability to set notification rules based on conditions and other options
- Pre-configured alerts that detect anomalies and threshold triggers
- Features to manage alerts through alert history and action triggers

Assisted Practice

Building Dashboards and Visualizations in Kibana **Duration: 15 Min.**

Problem statement:
You have been assigned the task to build dashboards and visualizations in Kibana for monitoring application performance and system health, which aids in quickly identifying issues.

Outcome:
By the end of this demo, you will be able to successfully create dashboards and visualizations in Kibana to effectively display data collected by Logstash from Elasticsearch.

Note: Refer to the demo document for detailed steps:
03_Building_Dashboards_and_Visualizations_in_Kibana

Assisted Practice: Guidelines

Steps to be followed:

1. Configure Kibana dashboard
2. Connect with Elasticsearch on lab

ASSISTED PRACTICE

Quick Check

You are a system administrator managing centralized log collection for an online service. Logstash is already configured to ingest logs, and now you need to build Kibana dashboards that display real-time user activity and server errors. What steps would complete your setup?

A. Create **Kibana visualizations** for errors and activity, send logs to **Elasticsearch**, and configure **index patterns** in Kibana

B. Use **grok filters** in Logstash to structure logs, send logs directly to **Kibana**, and create dashboards

C. Apply **geoip** to logs, set **Kibana alerts** for error spikes, and use **heatmaps** for real-time visualization

D. Set up **Kafka** to store logs, use **mutate** filters in Logstash, and create **pie charts** in Kibana for monitoring

Quick Check

You have recently joined a tech consulting firm as a DevOps engineer, and one of your first tasks is to deploy an Elasticsearch cluster for a new client. As you set up the environment, you must ensure you have configured the correct output plugin in Logstash, where the logs will be finalized and stored. Which code snippet would make sense to store logs in Elasticsearch?

A.
```
output {
      elasticsearch {
         hosts =>
"http://elasticsearch:9050"
         index => "logstash-logs"
      }
}
```

B.
```
output {
 elastic-search {
    hosts =>
"http://elasticsearch:9050"
 index => "logstash-logs"
 }
}
```

C.
```
output {
 elastic {
   hosts =>
"http://elasticsearch:9050"
}
```

D.
```
output {
 elastic_search {
   hosts =>
"http://elasticsearch:9050"
 index => "logstash-logs"
 }
}
```

Advanced Log Collection with ELK Stack

Advanced Log Collection with ELK Stack

Advanced log collection with the ELK Stack involves several sophisticated techniques to ensure efficient, scalable, and reliable log management.

Advanced Strategies for Log Management

To enhance log management and data processing, the following advanced strategies can be used:

Using Beats such as Filebeat for data collection

Logstash pipelines

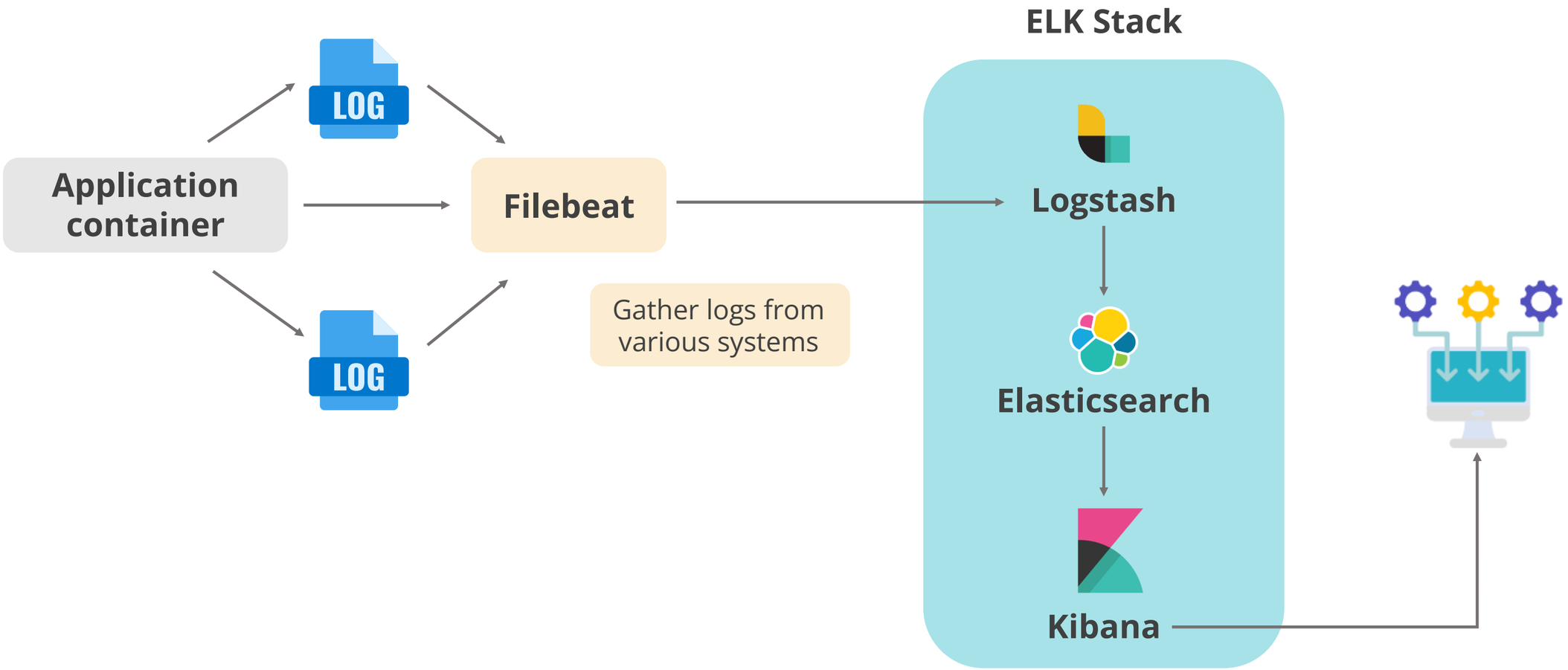
Data enrichment

Buffering with Kafka

Index lifecycle management

Filebeat as Component of ELK Stack

Filebeat collects and transmits log data from various sources, including application and system logs, to a central logging system like Elasticsearch or Logstash.



Key Features of Filebeat

Notable features of Filebeat are as follows:

01

Configuration details

It uses YAML files and predefined modules.

02

Resource-efficient

It is engineered for efficient log collection with minimal system impact.

03

Log collection capabilities

It monitors logs in real time and handles multiline entries.

Key Features of Filebeat

Notable features of Filebeat are as follows:

04

Security measures

It uses TLS for encryption and supports various authentication methods.

05

Data forwarding options

It sends logs directly to Elasticsearch or through Logstash for further processing.

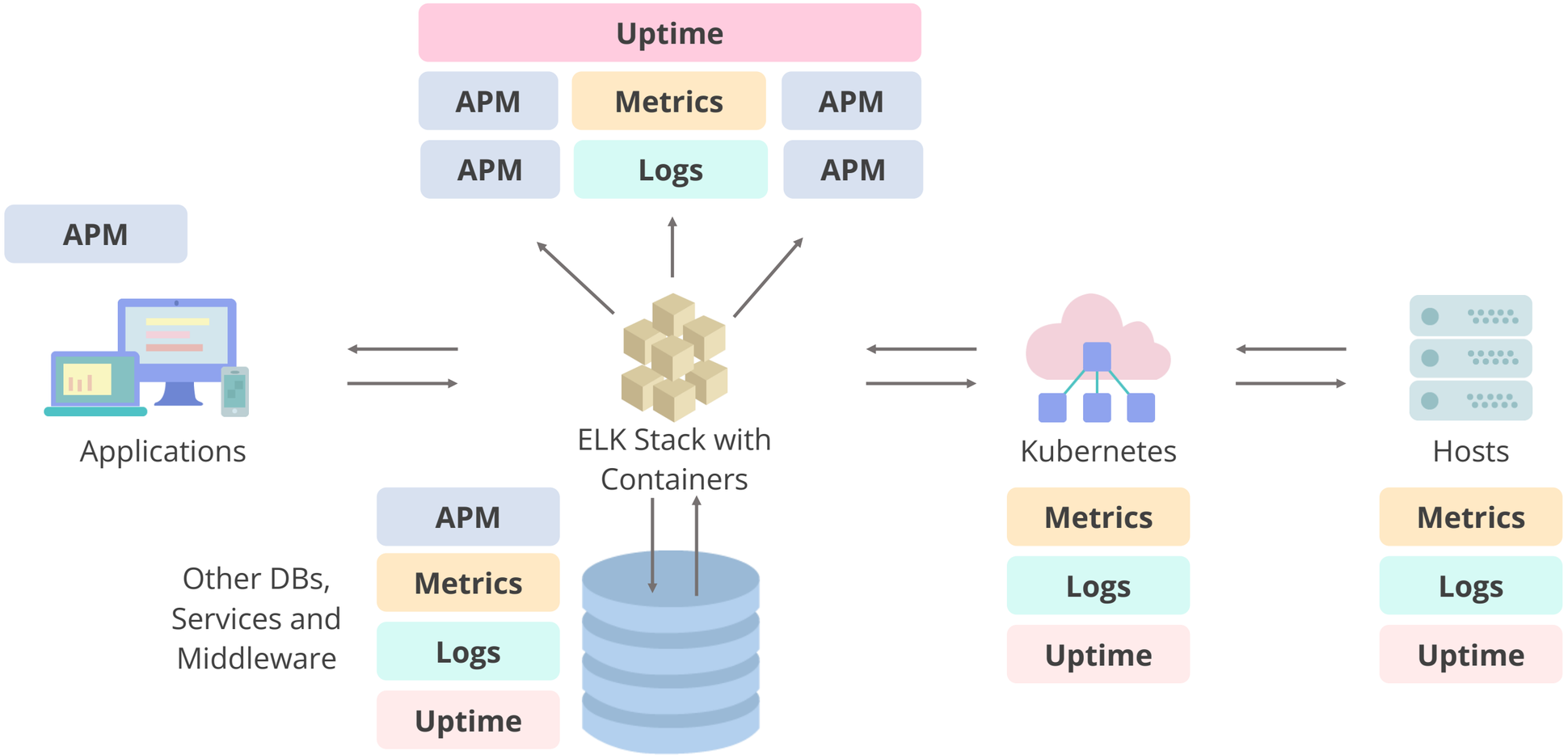
06

Data enrichment features options

It adds metadata to logs and supports buffering and compression for reliable transmission.

Integrating ELK Stack with Containers and Microservices

The diagram illustrates the integration of the ELK Stack with various data sources and monitoring tools.



Ref: https://www.elastic.co/blog/monitoring-infrastructure-and-microservices-with-elastic-observability

Integrating ELK Stack with Containers and Microservices

The steps to integrate the ELK Stack with containers and microservices are as follows:

01 Install ELK Stack

02 Containerize ELK Stack

03 Configure Logstash

04 Integrate with microservices

05 Visualize logs with Kibana

Integrating ELK Stack with Containers and Microservices

01 Install ELK Stack

This step involves setting up and installing the following:

- **Elasticsearch:** Set up Elasticsearch, a NoSQL database that uses the Lucene search engine for storing and searching log data
- **Logstash:** Install Logstash to collect, process, and forward log data to Elasticsearch
- **Kibana:** Install Kibana to visualize the log data stored in Elasticsearch

Integrating ELK Stack with Containers and Microservices

02 Containerize ELK Stack

Use Docker to create containers for Elasticsearch, Logstash, and Kibana. Docker Hub provides images for each component.

Define and manage your ELK stack services using a **docker-compose.yml** file

Integrating ELK Stack with Containers and Microservices

03 Configure Logstash

This is essential for collecting, transforming, and forwarding logs from various microservices to Elasticsearch.

Integrating ELK Stack with Containers and Microservices

04 Integrate with microservices

This includes the following tasks:

- **Logging:** Ensure your microservices generate logs in a format compatible with Logstash, such as JSON for structured logging

- **Shipping logs:** Deploy Filebeat on each microservice container to ship logs to Logstash. Filebeat is lightweight and designed to forward and centralize log data.

Integrating ELK Stack with Containers and Microservices

05 Visualize logs with Kibana

This includes performing the following steps:

- Access Kibana through your browser and create an index pattern to visualize the logs stored in Elasticsearch

- Use Kibana's **Discover, Visualize, and Dashboard** features to analyze and monitor your logs

Why Use Advanced Log Collection Techniques?

Employing advanced log collection techniques can significantly improve these areas:

Log management

Analysis

Performance

Advanced Log Collection Techniques

Several advanced techniques are designed to optimize log collection and processing with the ELK Stack. These include:

1. Scalable and distributed log collection	Logstash facilitates log aggregation and clustered log shippers facilitate handling of larger volumes of data.
2. Log collection in containerized environments	Sidecar containers improve management and scalability. DaemonSets ensure full log collection in Kubernetes.
3. Structured and enhanced logging	Structured formats like JSON simplify Logstash parsing, while added context improves log clarity.

Advanced Log Collection Techniques

Several advanced techniques are designed to optimize log collection and processing with the ELK Stack. These include:

4. Real-time log processing

Logstash filters and enriches logs, while Apache Kafka manages high-throughput streams.

5. Advanced log parsing

Logstash uses custom filters or parsing scripts to accurately extract and organize log information.

Assisted Practice

Configuring Filebeat to Collect Logs from Applications and Systems **Duration: 30 Min.**

Problem statement:
You have been assigned the task to implement Filebeat for collecting server metrics and storing them in Elasticsearch for enhanced monitoring and analysis.

Outcome:
By the end of this demo, you will be able to successfully configure Filebeat to collect logs from applications and systems and store them in Elasticsearch for efficient monitoring and analysis.

Note: Refer to the demo document for detailed steps:
04_Configuring_Filebeat_to_Collect_Logs_from_Applications_and_Systems

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed:

1. Configure Filebeat to collect logs

Quick Check

You are a DevOps engineer at a tech consulting firm. One of your tasks is to set up Filebeat on virtual machines to collect logs and send them to Logstash for further processing. You need to ensure that the Filebeat configuration file is correctly set up so the logs are directed to the appropriate Logstash instance. In the filebeat.yml configuration, where would you specify the destination for sending logs to Logstash?

A. filebeat.inputs

B. output.logstash

C. output.elasticsearch

D. processors.add_host_metadata

Quick Check

You've recently joined a firm as a DevOps engineer, and one of your first tasks is to deploy an ELK cluster for a new client. After setting up the ELK stack, the client requests that you start monitoring Linux VMs for metrics and logs. Your task is to configure Filebeat on the VMs to collect logs and visualize them in Kibana. What sequence of tasks would you follow?

A. Launch VM > Configure Filebeat > Fetch logs in Kibana

B. Launch VM > Fetch logs in Kibana > Configure Filebeat

C. Launch VM > Configure Filebeat > Fetch logs in Elasticsearch

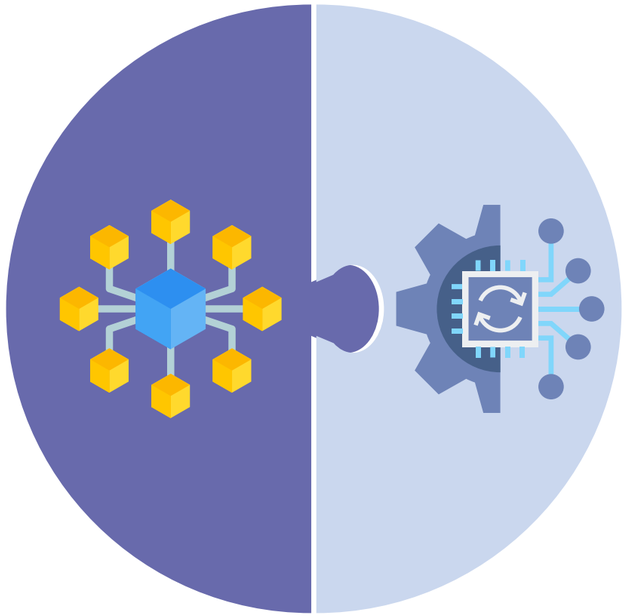
D. Launch VM > Fetch logs in Elasticsearch > Configure Filebeat

ELK Stack Integrations and Automation

Introduction to Integrating ELK Stack and Automation

ELK Stack offers the following features:

Flexible integration with various data sources, cloud services, and third-party tools for analytics



Automation capabilities such as alerting, scripting, CI/CD pipelines, and configuration management

Advantages of Integrating ELK Stack and Automation

Integrating the ELK Stack with CI/CD (continuous integration/continuous deployment) pipelines has several benefits, like:

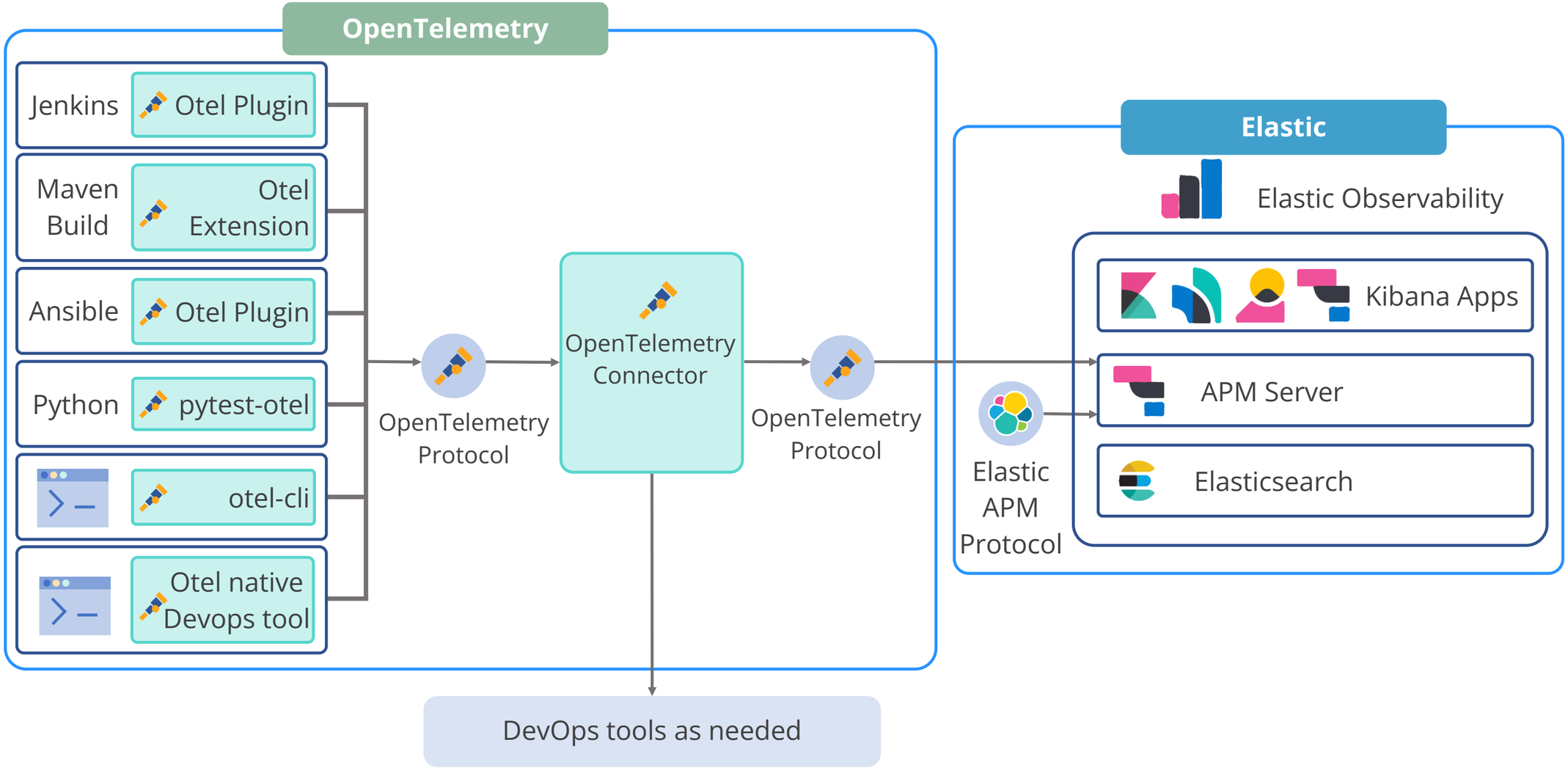
The integration enhances the visibility and management of development and deployment processes.

The ability to monitor and analyze logs in real-time is enhanced by such integration.

The integration also provides valuable insights and helps to detect issues early.

Integration of Elk Stack with CI/CD Pipeline

Effectively integrate ELK stack with your CI/CD pipelines for enhanced observability



Integration of Elk Stack with CI/CD Pipeline

After integrating ELK stack, become familiar with the processes to undertake these tasks:

Integration of Elk Stack with CI/CD Pipeline

Collecting logs: This is the first step in understanding log collection. The high-level process for integrating ELK stack and collecting logs are as follows:

1. Collecting logs from CI/CD tool logs

Gather logs from CI/CD systems (like Jenkins), which include information on build statuses, test results, and deployments

Example for Jenkins:
Configure Jenkins to output logs or use Jenkins plugins

2. Collecting logs from application logs

Collect logs generated during the build and deployment phases

Example for Docker logs:
Use Docker's logging drivers for containerized applications. This helps to forward logs to a centralized collector.

Integration of Elk Stack with CI/CD Pipeline

Configuring log collection: The next step is understanding how to configure logs after they are collected. The steps to complete the Filebeat setup are as follows:

1. Deployment

- Install Filebeat on CI/CD servers or as a sidecar container in a Kubernetes environment

2. Configuration

- Set up Filebeat to monitor specific log file paths or endpoints
- Configure it for the required log formats or use Filebeat modules designed for specific tools

Integration of Elk Stack with CI/CD Pipeline

Processing log collection: The third aspect is understanding how to process logs after configuration. The steps to configure Logstash to handle pipelines are as follows:

1. Log processing

- Configure Logstash to handle and process logs
- Set up pipelines to parse, filter, and enrich logs

2. Integration

- Ensure Logstash is configured to manage logs from different sources consistently

Integration of Elk Stack with CI/CD Pipeline

Enhancing and parsing logs: After collecting and processing logs, you can also enhance and parse them. The steps for each of these tasks are as follows:

1. Add metadata

Enhance logs with extra information. Additional context helps in identifying and correlating log entries more effectively.

Example:

Append commit hashes or usernames for improved traceability

2. Parse logs

Use Logstash filters (for example, Grok, Date, Mutate) to extract and structure log data

Example:

Extract IP addresses from logs and make it more searchable and useful in Kibana

Integration of Elk Stack with CI/CD Pipeline

Facilitating visualization and analysis: The final aspect to understand is how to effectively visualize and analyze data. The steps to do it are as follows:

1. Create dashboards in Kibana

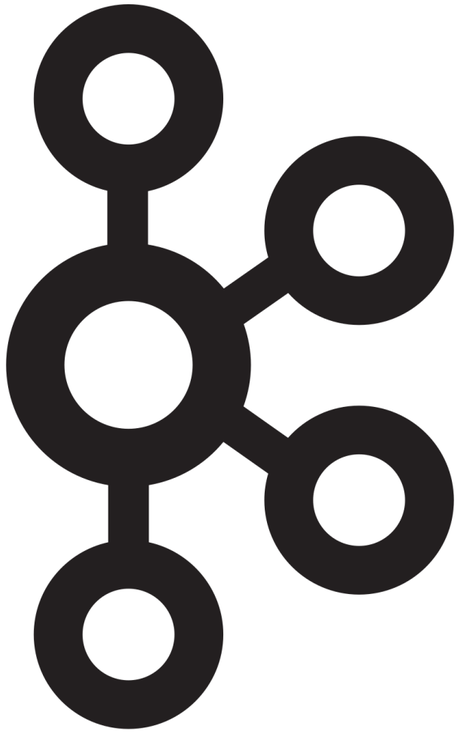
- **Metrics visualization:** Create dashboards to display metrics from CI/CD pipelines
- **Log analysis:** Build dashboards to visualize log data trends, monitor system performance, and troubleshoot issues

2. Set up alerting protocol

- **Custom alerts:** Alerts can be set up to provide notification of critical events or thresholds.
- **Integration:** Configure alerts to work with communication tools like email or Slack for real-time notifications

Apache Kafka

It is a distributed streaming platform used for building real-time data pipelines.



In DevOps monitoring and logging, it is essential for processing large volumes of log data, metrics, and events from various systems and applications.

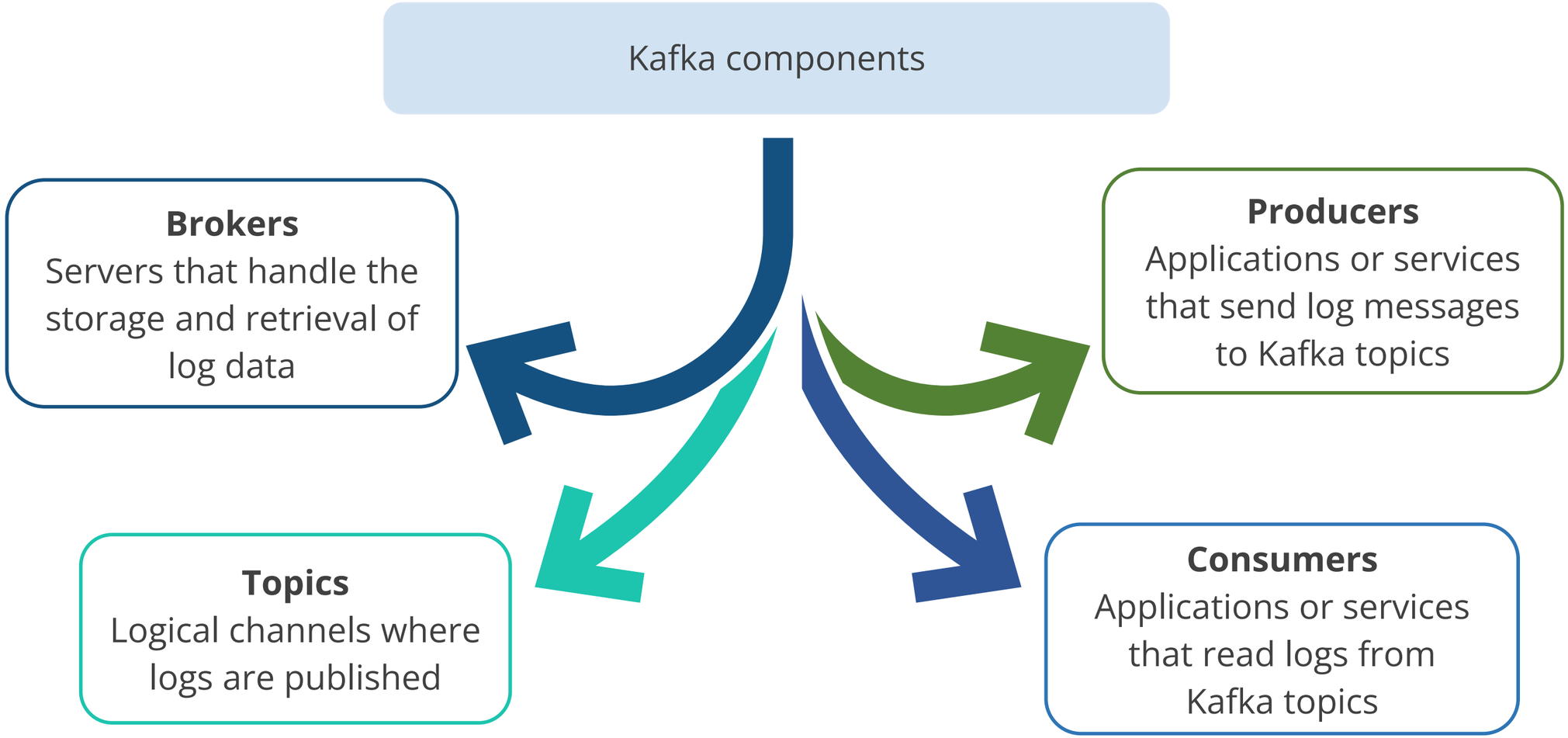
Streamlining Log Processing with Apache Kafka and ELK Stack

Kafka can greatly enhance log management and analysis in significant ways.

This integration supports scalable, efficient log processing and enables real-time insights into log data.

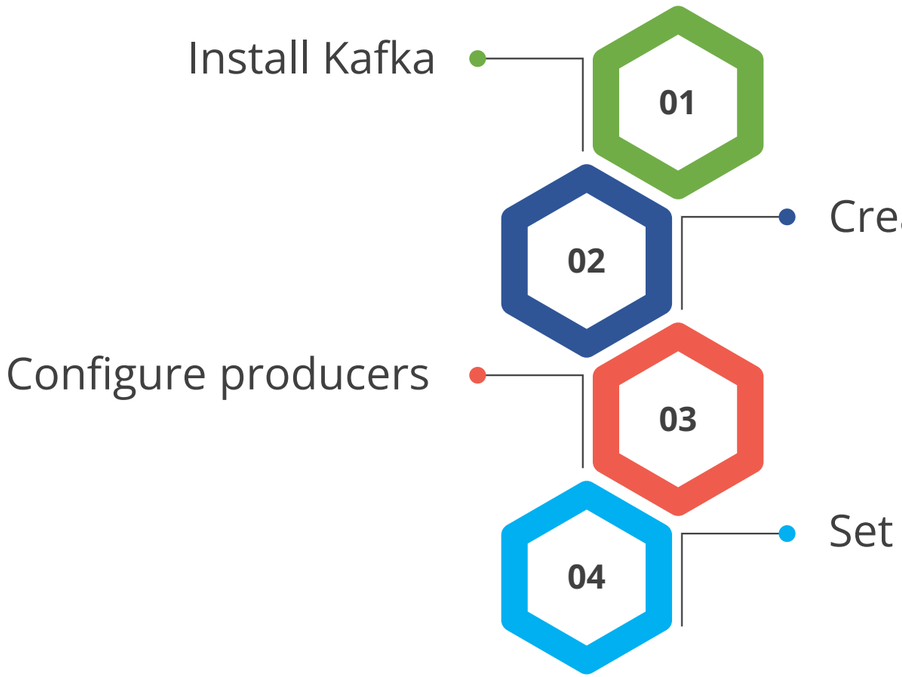
Understanding Kafka Components

These are core Kafka components that one must be familiar with when setting it up.



Steps to Integrate Kafka with Logstash

Here is an overview of the steps involved in configuring Kafka:



Steps to Integrate Kafka with Logstash

Kafka is typically integrated first because it's responsible for collecting and transmitting large streams of data in real time.

1. Install Logstash

Ensure that Logstash is installed on the system to modify the Logstash configuration

Example

```
root@ip-172-31-36-251:~# apt install logstash
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  logstash
0 upgraded, 1 newly installed, 0 to remove and 256 not upgraded.
Need to get 421 MB of archives.
After this operation, 698 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 logstash amd64 1:8.15.0-1 [421 MB]
9% [1 logstash 46.1 MB/421 MB 11%]
```

Steps to Integrate Kafka with Logstash

Kafka is typically integrated first because it's responsible for collecting and transmitting large streams of data in real time.

2. Configure Kafka input plugin
Set up the Kafka input plugin in Logstash to pull log data from Kafka topics

Example of a configuration

```
input {
 kafka {
  bootstrap_servers => "kafka-broker1:9092,kafka-broker2:9092"
  topics => ["logs-topic"]
  group_id => "logstash-group"
  codec => "json"
 }
}
```

Steps to Integrate Kafka with Logstash

Kafka is typically integrated first because it's responsible for collecting and transmitting large streams of data in real time.

3. Apply filters in Logstash
Use filters to parse, enrich, and transform logs

Example of a filter configuration

```
filter {
 grok {
  match => { "message" => "%{TIMESTAMP_ISO8601:timestamp}
%{LOGLEVEL:loglevel} %{GREEDYDATA:message}" }
}
date {
 match => [ "timestamp", "ISO8601" ]
 }
}
```

Steps to Configure Kafka

Kafka is typically integrated first because it's responsible for collecting and transmitting large streams of data in real time.

4. Forward to Elasticsearch

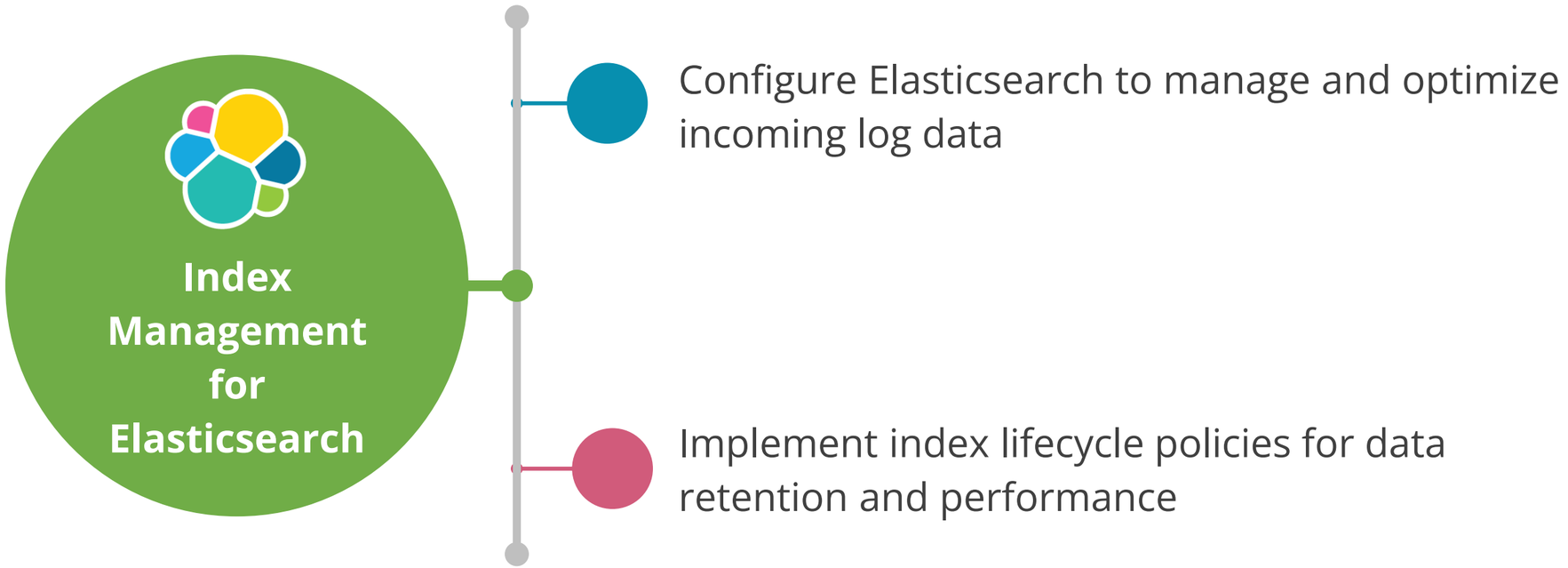
Use the Elasticsearch output plugin in Logstash to send the processed logs to Elasticsearch

Code for Elasticsearch output login

```
output {
  elasticsearch {
    hosts => ["http://elasticsearch:9200"]
    index => "logs-%{+YYYY.MM.dd}"
  }
}
```

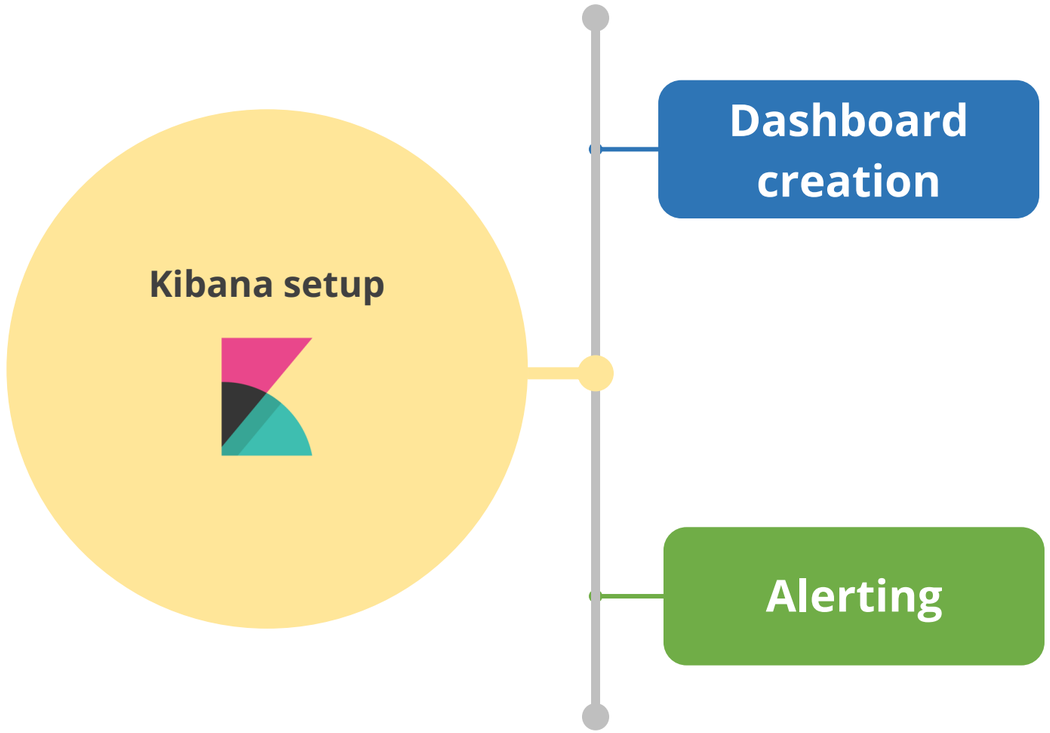
Steps to Configure Elasticsearch

After the data is ingested and processed, it needs to be stored and indexed, which is where Elasticsearch comes in. The steps to configure Elasticsearch are as follows:



Steps to Configure Kibana

Finally, Kibana is configured to visualize the data stored in Elasticsearch. The steps to configure Kibana are as follows:



- Use Kibana to build dashboards that visualize log data from Elasticsearch
- Create visualizations for metrics such as error rates, build statuses, and system performance

- Set up alerts in Kibana for critical log events or anomalies
- Address issues promptly in response to timely alerts

Alerts and Notification Based on Log Data

The steps to set up alerts and notifications based on log data are:

1. Define alerting rules

Specify the conditions under which an alert should be triggered

Example:
Trigger an alert if error logs exceed a certain threshold

Example of a configuration

```
name: High Error Rate Alert
type: threshold
index: "logs-*"
query: 'status:"error"'
threshold: 100
time_window: "5m"
```

2. Create an alert

Navigate to the **Alerting** section on Kibana and define the alert criteria

Alerts and Notification Based on Log Data

The steps to set up alerts and notifications based on log data are:

3. Set up actions

Configure actions to be taken when an alert is triggered

Example:
Configure email settings to send alerts to a distribution list

4. Test and deploy

Ensure alerts are firing as expected and monitor their performance

Assisted Practice

Building an Automated Log Processing Pipeline with Kafka and ELK Stack **Duration: 30 Min.**

Problem statement:
You have been assigned the task to integrate Kafka with the ELK Stack to buffer and process log surges, protecting Logstash and Elasticsearch during high-volume events for improved reliability and scalable log management.

Outcome:
By the end of this demo, you will be able to successfully integrate Kafka with the ELK Stack to buffer and process log surges, ensuring improved reliability and scalable log management for Logstash and Elasticsearch during high-volume events.

Note: Refer to the demo document for detailed steps:
05_Building_an_Automated_Log_Processing_Pipeline_with_Kafka_and_ELK_Stack

Assisted Practice: Guidelines

Steps to be followed:

1. Set up the system and install Java Runtime Environment (JRE)
2. Set up the Apache Kafka message broker
3. Create a topic for the Apache logs

Best Practices for Effective DevOps Logging with ELK Stack

Key practices followed to achieve effective DevOps are:

01

Define clear learning objectives

02

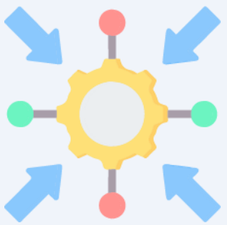
Standardize log formats

03

Implement structured logging

04

Centralize log collection



Best Practices for Effective DevOps Logging with ELK Stack

Key practices followed to achieve effective DevOps are:

05

Optimize log ingestion and storage

06

Enhance log parsing and enrichment

07

Develop useful visualizations

08

Set up alerting and notifications

Best Practices for Effective DevOps Logging with ELK Stack

Key practices followed to achieve effective DevOps are:

09

Ensure security and compliance

10

Automate and scale

Quick Check

You are setting up a logging and monitoring system for a large-scale e-commerce platform. To manage log data, you've integrated the ELK Stack with Apache Kafka. Now, you need to configure Logstash to process and forward the logs to Elasticsearch. What step would you take next in the Logstash configuration to ensure log data is pulled from Kafka?

A. Install Logstash on the Kafka broker

B. Set up the Kafka input in Logstash to pull logs from Kafka topics

C. Install Kibana to visualize log data from Kafka

D. Configure Elasticsearch to index logs directly from Kafka

Quick Check

Your team is automating the logging process for the GitLab CI/CD pipeline. To capture logs in real time and index them for analysis, you need to set up the ELK Stack for log shipping and visualization. What steps would you follow to configure Filebeat to ship logs from GitLab CI to Elasticsearch?

A. Install Filebeat on the GitLab CI server, configure the correct log paths in **filebeat.yml**, and set the output to Elasticsearch

B. Install Logstash on the GitLab CI server, configure Logstash to send logs to Elasticsearch, and visualize them in Kibana

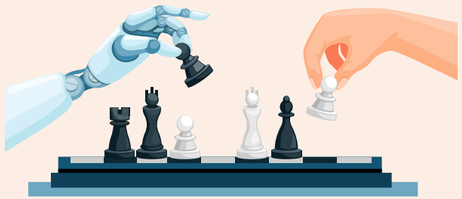
C. Configure GitLab CI to send logs directly to Kibana for visualization

D. Install Filebeat on the GitLab CI server and configure it to send logs directly to Kibana

Centralized Logging and Monitoring: Case Studies

Case Study: Netflix

ELK Stack for Log Management



Challenges

- Netflix, a global leader in streaming services, needed a system to manage and analyze extensive log data generated by its complex infrastructure.
- The system had to scale for real-time data processing from multiple sources.
- Better visibility of operations was needed to obtain useful insights.

Case Study: Netflix

ELK Stack for Log Management

Solution

- Utilizing Elasticsearch, Netflix achieved effective real-time indexing and querying of log data, significantly enhancing the speed and accuracy of their data analysis.
- The combination of Logstash and Kibana streamlined the data management process and empowered Netflix to monitor system health accurately. Custom dashboards helped improve the overall operational efficiency.

Case Study: Netflix

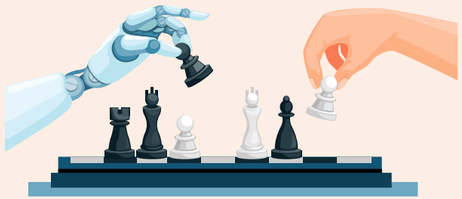
ELK Stack for Log Management

Outcome

- Netflix enhanced operational visibility and problem resolution through real-time log analysis, enabling proactive system management and quicker troubleshooting.
- The ELK Stack efficiently handled Netflix’s expanding data needs. Kibana’s visualizations and Logstash’s data processing equipped it to swiftly identify and resolve potential issues, maintaining consistent service stability.

Case Study: GitHub

Scalability and Performance Enhancement with the ELK Stack



Challenges

- GitHub, a leading platform for version control and collaborative software development, generates vast amounts of log data from its extensive infrastructure, including web servers, application servers, and user activities.
- Rapid identification and resolution of system issues were hindered by the need for immediate data processing.
- It was difficult to extract actionable insights from performance trends and user behavior as the data was complex and vast.

Case Study: GitHub

Scalability and Performance Enhancement with the ELK Stack

Solution

- GitHub utilized Elasticsearch to manage and query vast log datasets efficiently, providing scalability and significantly speeding up data access.
- It enhanced data processing with Logstash and used Kibana's dashboards to quickly address issues, ensuring seamless operations for developer collaboration and code management.

Case Study: GitHub

Scalability and Performance Enhancement with the ELK Stack

Outcome

- GitHub enhanced monitoring capabilities using real-time log processing, which enabled faster identification and resolution of issues, crucial for managing the high volume of user activities and system performance demands.
- The flexibility of the ELK Stack allowed GitHub to scale operations effectively, essential for supporting continuous integration and deployment in software development environments.

Key Takeaways

- Effective logging in DevOps ensures application and infrastructure health, security, troubleshooting, and performance optimization.
- The ELK Stack is an open-source suite that uses Elasticsearch for data search, Logstash for data collection, and Kibana for visualization.
- In DevOps, logs from events, servers, systems, and applications are crucial for monitoring behavior, performance, security, and aiding in debugging.
- Key features of Elasticsearch include distributed architecture, full-text search, and data analytics.

Key Takeaways

- Advanced log collection techniques include scalable and distributed collection, containerized logging, structured logging, real-time processing, and advanced parsing.
- Integrating the ELK Stack with CI/CD pipelines enhances log monitoring, improving issue detection and supporting smoother deployments
- For effective DevOps logging with the ELK Stack, standardize formats, use structured logging, centralize logs, optimize storage, enhance parsing, create dashboards, and set alerts for issue detection.

Implementing Jenkins Logging Using ELK Stack

Duration: 45 Min.

Project agenda: To create Jenkins logging integration with ELK Stack

Description: You have been given the task of onboarding the Jenkins application to implement CI/CD pipelines. Once the pipelines are created, you need to ensure that Jenkins is working as expected. The integration of Jenkins logging with the ELK Stack will allow developers to monitor Jenkins logs without needing root access.

Tools required: Jenkins, ELK Stack

Prerequisites: None

LESSON-END PROJECT

Implementing Jenkins Logging Using ELK Stack

Duration: 45 Min.

Perform the following:

1. Log in to Simplilearn Lab
2. Install Elasticsearch on Lab for log storage
3. Configure Filebeat to collect Jenkins logs
4. Configure the Kibana visualization tool to check Jenkins logs

Expected deliverables: Integrating Jenkins logs to ELK Stack

LESSON-END PROJECT

Thank You