ELK Stack & OpenSearch for DevOps

ELK Stack
ELK Stack Project for DevOps

OpenSearch

Opensearch Project for DevOps
OpenSearch vs. ELK Stack

ELK Stack

🎬 **Imagine you're a movie director**
With the **ELK Stack**, it's like you're the director of a movie set—making sure everything runs smoothly behind the scenes. And if something goes wrong, you can quickly spot the issue and fix it, just like a pro production crew handling problems on set.

🎥 **A DevOps Example in Movie Production:**

> Imagine your **movie shoot** is your app's deployment, and something goes wrong with the filming.
>
> - **Logstash** collects all the logs from different departments—camera team (containers), script writers (API

requests), and more.

- **Elasticsearch** pinpoints that the **lighting team (server)** had a malfunction, which caused the shot to fail.

- **Kibana** shows you a timeline of events—when the lighting failure occurred, which shots were affected, and what adjustments need to be made.

Now you can **fix the lighting issue**, and your movie shoot continues on schedule! 🎬🚀

🔍 What is the ELK Stack?

The ELK Stack is a combination of three tools—**Elasticsearch**, **Logstash**, and **Kibana**—used to collect, process, and display log data from your applications and systems.

You can use the basic features of the ELK Stack for free, but some advanced features may require a paid subscription. It's commonly used to monitor your system, spot issues, and analyze data in real-time.

📦 What's Inside the ELK Stack?

E = Elasticsearch

Elasticsearch is like a super-fast librarian. It stores all your data and helps you search through it instantly.

- It works with JSON (a simple format for data).

- It's fast and can handle large amounts of data.

- Great for searching logs or any text data quickly.

🔒 *Note:* Newer versions of Elasticsearch aren't fully open-source anymore. If you prefer open-source tools, try **OpenSearch**, which is a free alternative.

L = Logstash
Logstash is like your data organizer. It collects messy data from different places, cleans it up, and sends it to Elasticsearch.

- Works with both structured and unstructured data.
- Has 200+ plugins to connect with many sources.
- Helps clean, filter, and reformat log files, system data, and more.

✨ *Example:* You can use Logstash to pick up log files from your app, add a timestamp, and send them to Elasticsearch for storage.

K = Kibana
Kibana is your visual dashboard. It helps you **see** what's going on using graphs, charts, and maps.

- Supports many types of visualizations: line graphs, pie charts, maps, etc.
- Easy to use with filters and search.
- Dashboards update in real time to show live data.

🔒 *Note:* Like Elasticsearch, Kibana is no longer fully open-source after version 7.10.2. You can use **OpenSearch Dashboards** as an alternative.

⚙️ How Does It All Work Together?

Here's how the pieces connect:

1. **Logstash** collects and cleans up data (like a data janitor).
2. **Elasticsearch** stores and lets you search that data quickly.
3. **Kibana** shows the data in charts and dashboards (like a reporting tool).

🧠 In Simple Terms:

With ELK Stack, you're watching over your apps and systems like a director watches a film shoot—spotting problems early, fixing them fast, and keeping everything running like a well-oiled machine.

🔧 Setting Up ELK Stack for Log Management

Install Elasticsearch

For **Debian/Ubuntu**:

wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.x.x-amd64.deb
sudo dpkg -i elasticsearch-7.x.x-amd64.deb

For **RPM-based systems**:

wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.x.x-x86_64.rpm
sudo rpm -ivh elasticsearch-7.x.x-x86_64.rpm

Start Elasticsearch:

sudo service elasticsearch start

Edit config file:

yaml

```
# /etc/elasticsearch/elasticsearch.yml
network.host: localhost
http.port: 9200
cluster.initial_master_nodes: ["node-1"]
```

Verify:

curl -X GET "localhost:9200/"

Create an index:

curl -X PUT "localhost:9200/logs/"

Configure Logstash

Create a configuration file with three sections: input, filter, and output.

ruby

```
input {
  beats {
    port => 5044
  }
}
```

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

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

To get logs from a file:

ruby

```
input {
  file {
    path => "/var/log/myapp/*.log"
    start_position => "beginning"
  }
}
```

Run Logstash with your config file:

bin/logstash -f logstash.conf

📊 ELK in DevOps

The ELK Stack is a core part of observability in DevOps. It enables:

- Centralized logging from all microservices and servers
- Quick detection of failures and performance bottlenecks
- Integration with CI/CD for real-time monitoring and alerts

You can also use hosted solutions like **Amazon OpenSearch Service** for a managed ELK experience that takes care of scaling, patching, and backups.

✅ Why ELK Stack Matters

- **Real-time log insights** to speed up troubleshooting
- **Cost-effective** alternative to commercial logging tools
- **Custom dashboards and alerts** for operations and security teams
- **Open-source flexibility** with community support and plugins

Elasticsearch Commands:

Check cluster health:
curl -X GET "localhost:9200/_cluster/health?pretty"

Get Elasticsearch version:
curl -X GET "localhost:9200"

Index a document:
curl -X POST "localhost:9200/my_index/_doc/1" -H 'Content-Type: application/json' -d'

{

"name": "John Doe",

"age": 30

}

'

Search in an index:
curl -X GET "localhost:9200/my_index/_search?q=name:John"

Delete a document:
curl -X DELETE "localhost:9200/my_index/_doc/1"

Get all indices:
curl -X GET "localhost:9200/_cat/indices?v"

Delete an index:
curl -X DELETE "localhost:9200/my_index"

Get document count:
curl -X GET "localhost:9200/my_index/_count"

Logstash Commands:

Start Logstash with a config file:
sudo service logstash start

Test Logstash configuration:
sudo /usr/share/logstash/bin/logstash -f /path/to/your-config-file.conf --config.test_and_exit

Start Logstash with inline configuration:
echo "Hello World" | /usr/share/logstash/bin/logstash -e 'input { stdin { } } output { stdout {} }'

View Logstash logs:
tail -f /var/log/logstash/logstash-plain.log

Kibana Commands:

Start Kibana:
sudo service kibana start

- **Access Kibana dashboard**: Open your browser and navigate to: http://localhost:5601

Check Kibana logs:
tail -f /var/log/kibana/kibana.log

- **Get Kibana version**:
  curl -X GET "localhost:5601/api/status"

ELK Stack Project for DevOps

ELK Stack: Real-time Log Monitoring and Visualization Project with Fluentd

Objective:
Set up an ELK Stack (Elasticsearch, Logstash, Kibana) for centralized log management using Fluentd for log collection, processing, and visualization. The setup includes real-time log monitoring, Dockerization, and the use of Kubernetes (via kind) for deployment.

Step 1: Install Elasticsearch

Follow the same steps as the original to install Elasticsearch:

sudo apt-get update
sudo apt-get install openjdk-11-jdk
wget
https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.10.0-amd64.deb
sudo dpkg -i elasticsearch-7.10.0-amd64.deb
sudo service elasticsearch start

Verify Elasticsearch by visiting http://localhost:9200.

Step 2: Install Logstash

Install Logstash as in the original:

wget https://artifacts.elastic.co/downloads/logstash/logstash-7.10.0.deb
sudo dpkg -i logstash-7.10.0.deb

Configure Logstash to collect logs from Fluentd:

```
# Edit /etc/logstash/conf.d/fluend-logs.conf
sudo nano /etc/logstash/conf.d/fluend-logs.conf
```

Example configuration for receiving logs from Fluentd:

plaintext

```
input {
  tcp {
    port => 5044
  }
}

filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }
}

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

Start Logstash:

```
sudo service logstash start
```

Step 3: Install Kibana

Install Kibana as before:

wget https://artifacts.elastic.co/downloads/kibana/kibana-7.10.0-amd64.deb
sudo dpkg -i kibana-7.10.0-amd64.deb
sudo service kibana start

Access Kibana at http://localhost:5601.

Step 4: Visualize Logs in Kibana

Create an index pattern in Kibana for Fluentd logs:

- Go to **Management > Index Patterns** and create a new index pattern for fluentd-logs-*.

Explore logs in **Discover** and use Kibana's tools to search, filter, and analyze logs.

Step 5: Set Up Fluentd for Log Collection

5.1 Install Fluentd on Your Servers
Install Fluentd on your server to collect logs:

 # Install Fluentd (Debian/Ubuntu)
curl -L https://toolbelt.treasuredata.com/sh/install-debian.sh | sh

5.2 Configure Fluentd to Send Logs to Logstash
Create or modify the Fluentd configuration file to send logs to Logstash:

sudo nano /etc/fluentd/fluentd.conf

Example Fluentd configuration:

xml

```
<source>
  @type tail
  path /var/log/apache2/access.log
  pos_file /var/log/td-agent/access.log.pos
  tag apache.access
  format apache2
</source>

<match apache.access>
  @type forward
  <server>
    host your-logstash-server-ip
    port 5044
  </server>
</match>
```

Start Fluentd:

sudo service td-agent start

Step 6: Set Up Alerts in Kibana

Configure Kibana alerting as in the original setup:

- Go to **Stack Management > Rules and Connectors** in Kibana.
- Create a rule to alert when certain conditions are met (e.g., more than 50 HTTP 500 errors).

Step 7: Enhance Logstash with GeoIP and JSON Parsing

As in the original, you can enhance your Logstash configuration to parse JSON and GeoIP data:

- **Install the GeoIP plugin:**
  sudo bin/logstash-plugin install logstash-filter-geoip

Update the Logstash configuration (fluend-logs.conf):

plaintext

```
filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }
  geoip {
    source => "clientip"
    target => "geoip"
  }
}
```

Restart Logstash:

sudo service logstash restart

To parse JSON logs, add this filter to your Logstash configuration:

plaintext

```
filter {
  json {
    source => "message"
  }
```

}

Step 8: Dockerize the ELK Stack with Kubernetes (kind)

You will deploy the ELK Stack using Kubernetes in Docker (kind). First, install kind if you haven't already.

Create a docker-compose.yml for Elasticsearch, Logstash, and Kibana:

yaml

```
version: '3'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.10.0
    environment:
      - discovery.type=single-node
    ports:
      - "9200:9200"
    volumes:
      - elasticsearch_data:/usr/share/elasticsearch/data

  logstash:
    image: docker.elastic.co/logstash/logstash:7.10.0
    ports:
      - "5044:5044"
    volumes:
      - ./logstash.conf:/etc/logstash/conf.d/logstash.conf
    depends_on:
      - elasticsearch

  kibana:
    image: docker.elastic.co/kibana/kibana:7.10.0
    ports:
```

```
    - "5601:5601"
  depends_on:
    - elasticsearch

volumes:
  elasticsearch_data:
```

Create a Kubernetes deployment using kind for the ELK stack. This will set up the Elasticsearch, Logstash, and Kibana pods in a Kubernetes cluster running inside Docker.

Step 9: Add More Log Sources

To expand monitoring, add more log sources (e.g., MySQL or PostgreSQL) to Fluentd:

9.1 Collect Database Logs

For MySQL logs, update the Fluentd configuration (fluentd.conf):

xml

```
<source>
  @type tail
  path /var/log/mysql/mysql.log
  pos_file /var/log/td-agent/mysql.log.pos
  tag mysql.log
  format json
</source>

<match mysql.log>
  @type forward
  <server>
    host your-logstash-server-ip
    port 5044
```

```
  </server>
</match>
```

Restart Fluentd:

sudo service td-agent restart

Conclusion

This project achieves real-time log monitoring and visualization with the ELK stack using Fluentd :

- Fluentd for log collection and shipping.
- Use of Kubernetes (via kind) for container orchestration.
- Dockerization of the entire ELK Stack.
- Enhanced log analysis with GeoIP and JSON parsing.

This setup provides a scalable solution for monitoring, visualizing, and alerting based on logs from multiple servers and services.

Opensearch

OpenSearch is like giving your data a **magnifying glass, a jetpack, and a crystal ball** — all at once.

🔍 **Magnifying glass** – because it lets you zoom in on the tiniest detail in massive logs and datasets.

🚀 **Jetpack** – because it searches lightning-fast across oceans of information.

🔮 **Crystal ball** – because it reveals trends, patterns, and anomalies before they turn into real problems.

It's your **open-source sidekick** for search, analytics, and visualization — born from Elasticsearch, but free to explore, expand, and evolve the way you want.

Whether you're a developer, DevOps engineer, or data geek — OpenSearch helps you **search smarter, troubleshoot faster, and visualize beautifully**.

OpenSearch is a free, open-source tool used to **search, monitor, and analyze data** in real-time. It's built on **Apache Lucene** and is great for handling large data, like logs and user activity. It includes features like **full-text search**, **anomaly detection**, and **data dashboards** for easy visualization.

💡 Why Use OpenSearch?

OpenSearch is a **free and open-source search and analytics tool**. It helps you:

- **Ingest** (collect), **secure**, **search**, and **analyze** your data
- **Visualize** your data using easy dashboards
- Work with different types of data like **logs**, **metrics**, and **application traces**

✨ Key Benefits

- **Completely Free & Open Source** – You can use, change, and share it without cost.
- **Community Support** – Developers and companies around the world contribute to it.

- **Growing Ecosystem** – Includes plugins, managed services, and continuous improvements.

🛠️ Why Was OpenSearch Created?

In 2021, the creators of Elasticsearch made it **no longer fully open-source**. To keep a free option available, **OpenSearch** was created by forking (copying and modifying) the last open-source version (Elasticsearch 7.10).

OpenSearch is now governed by the **Apache 2.0 license** – meaning it's truly open-source and safe to use.

☁️ What is Amazon OpenSearch Service?

It’s a **managed version** of OpenSearch offered by **AWS**.

With it, you don’t need to handle:

- Server setup
- Scaling
- Monitoring

It also includes cool features like **cross-cluster replication** and **trace analytics**.

🔄 Will OpenSearch Stay Compatible with Elasticsearch?

OpenSearch started from Elasticsearch, but it now grows **independently**. It focuses on features the **community values most**, especially in:

- Search
- Observability
- Analytics

🔐 Key Features of OpenSearch

- **Security**: Built-in encryption, user roles, and access controls
- **Search**: Full-text search, autocomplete, fuzzy search
- **SQL & PPL Support**: Use SQL-like queries
- **Dashboards**: View data with graphs and charts
- **Machine Learning**: Detect unusual patterns
- **Alerting**: Get notified in real time
- **Cross-Cluster Replication**: Share data across systems

🌍 Who Maintains OpenSearch?

OpenSearch is developed by:

- The open-source community
- Companies like **AWS**, **Red Hat**, and **SAP**

Anyone can contribute!

🔮 The Future of OpenSearch

Since 2021, OpenSearch has added features for:

- **Security**
- **Performance monitoring**
- **Machine learning**

New updates are based on what **users want**.

📜 Licensing

OpenSearch uses the **Apache License 2.0**, which means:

- Free to use
- Free to modify
- Encourages contributions

🖥️ How to Install OpenSearch on Linux

1. Download OpenSearch:

wget
https://artifacts.opensearch.org/releases/bundle/opensearch/2.0.0/opensearch-2.0.0-linux-x64.tar.gz

2. Extract and Install:

tar -zxvf opensearch-2.0.0-linux-x64.tar.gz
cd opensearch-2.0.0

3. Start OpenSearch:

./bin/opensearch

4. Check if it's running:

curl -X GET "localhost:9200/"

📊 OpenSearch Dashboards

Download & Start:

wget
https://artifacts.opensearch.org/releases/bundle/opensearch-dashboards/2.0.0/opensearch-dashboards-2.0.0-linux-x64.tar.gz

tar -zxvf opensearch-dashboards-2.0.0-linux-x64.tar.gz
cd opensearch-dashboards-2.0.0
./bin/opensearch-dashboards

👉 Access at: http://localhost:5601

🔎 Basic OpenSearch Queries

Check Cluster Health:

curl -X GET "localhost:9200/_cluster/health?pretty"

Create an Index:

curl -X PUT "localhost:9200/my-index?pretty"

Index a Document:

```
curl -X POST "localhost:9200/my-index/_doc/1?pretty" \
-H 'Content-Type: application/json' \
-d '{
  "title": "OpenSearch Introduction",
  "content": "OpenSearch is a scalable, open-source search and analytics engine."
}'
```

Opensearch Project for DevOps

Project Title: OpenSearch Setup and Integration in DevOps Pipeline

---

Objective:

To learn how to deploy and manage OpenSearch in a DevOps pipeline, integrate it with a simple application, and automate the deployment using Docker, Kubernetes, and CI/CD tools like Jenkins or GitHub Actions.

Steps for the Project:

1. Understanding OpenSearch

- **What is OpenSearch?**

    - OpenSearch is an open-source search and analytics suite derived from Elasticsearch, designed for scalability, security, and versatility.

    - It provides features like full-text search, data analytics, and log monitoring.

- **Why use OpenSearch?**

    - It helps in storing and analyzing large volumes of structured and unstructured data.

    - You can use OpenSearch for log analysis, real-time analytics, and visualizing metrics.

2. Setting Up OpenSearch Locally

- Install OpenSearch on your local machine or use Docker for a local setup.

Run OpenSearch as a Docker container:

docker pull opensearchproject/opensearch:latest
docker run -d --name opensearch -p 9200:9200 -p 9600:9600 opensearchproject/opensearch:latest

- Verify the OpenSearch is running by navigating to http://localhost:9200.

3. Setting Up OpenSearch in Kubernetes

- **Why Kubernetes?**

    - Kubernetes allows you to easily deploy and manage containerized applications like OpenSearch.

- Create Kubernetes resources for OpenSearch:
    - **Deployment.yaml**: Define the deployment configuration for OpenSearch.
    - **Service.yaml**: Create a service to expose OpenSearch externally.

Example **deployment.yaml** for OpenSearch:

yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: opensearch
spec:
  replicas: 1
  selector:
    matchLabels:
      app: opensearch
  template:
    metadata:
      labels:
        app: opensearch
    spec:
      containers:
        - name: opensearch
          image: opensearchproject/opensearch:latest
          ports:
            - containerPort: 9200
            - containerPort: 9600
```

Example **service.yaml** for OpenSearch:

yaml

```
apiVersion: v1
kind: Service
metadata:
  name: opensearch-service
spec:
  selector:
    app: opensearch
  ports:
    - protocol: TCP
      port: 9200
      targetPort: 9200
  type: NodePort
```

Deploy OpenSearch in Kubernetes:

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```

4. CI/CD Integration with Jenkins or GitHub Actions

- **Jenkins Setup**:
    - Use Jenkins pipelines to automate the deployment of OpenSearch in Kubernetes or Docker.
- **Sample Jenkinsfile** to deploy OpenSearch in Kubernetes:

groovy

```
pipeline {
    agent any
```

```
    stages {
        stage('Clone Repository') {
            steps {
                git 'https://github.com/your-repo.git'
            }
        }
        stage('Deploy to Kubernetes') {
            steps {
                script {
                    sh 'kubectl apply -f deployment.yaml'
                    sh 'kubectl apply -f service.yaml'
                }
            }
        }
    }
}
```

- **GitHub Actions Setup**:
    - Use GitHub Actions workflows to automate the build and deployment of OpenSearch containers to Docker Hub or Kubernetes.

Example GitHub Actions Workflow (.github/workflows/deploy.yml):

yaml

```
name: Deploy OpenSearch to Kubernetes
on:
  push:
    branches:
      - main

jobs:
  deploy:
```

```
runs-on: ubuntu-latest
steps:
  - name: Checkout repository
    uses: actions/checkout@v2

  - name: Set up kubectl
    uses: azure/setup-kubectl@v1

  - name: Deploy OpenSearch to Kubernetes
    run: |
      kubectl apply -f deployment.yaml
      kubectl apply -f service.yaml
```

5. Integrating OpenSearch with Application Logs

- **Log Collection**:
    - Integrate OpenSearch with your application for log collection and visualization.
    - Use Filebeat or Fluentd to ship logs from your application to OpenSearch.
- **Example**:
    - Install Filebeat on your application server to collect logs and send them to OpenSearch:

```
filebeat modules enable system
filebeat setup
filebeat -e
```

6. Visualizing Data with OpenSearch Dashboards

- **Install OpenSearch Dashboards**:
    - OpenSearch Dashboards is used to visualize data stored in OpenSearch. It's similar to Kibana.

To deploy OpenSearch Dashboards in Kubernetes:

yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: opensearch-dashboards
spec:
  replicas: 1
  selector:
    matchLabels:
      app: opensearch-dashboards
  template:
    metadata:
      labels:
        app: opensearch-dashboards
    spec:
      containers:
        - name: opensearch-dashboards
          image: opensearchproject/opensearch-dashboards:latest
          ports:
            - containerPort: 5601
```

- Expose the service externally and access it through the browser at http://localhost:5601.

7. Monitoring and Alerting with OpenSearch

- **Monitoring**:
    - Use OpenSearch to monitor your application logs, performance metrics, and infrastructure health.
- **Alerting**:
    - Set up alerts based on specific log patterns or performance thresholds.
- Example: Alert if error logs are greater than a specific threshold.

Skills Learned:

- Setting up and deploying OpenSearch in Docker and Kubernetes.
- Integrating OpenSearch with CI/CD pipelines (Jenkins/GitHub Actions).
- Using OpenSearch Dashboards for log visualization.
- Automating deployments using Jenkins or GitHub Actions.
- Collecting logs and monitoring infrastructure using OpenSearch.

This project provides a hands-on approach to learning DevOps practices while getting familiar with OpenSearch. It will give you a practical understanding of how logging, monitoring, and analytics fit into DevOps workflows.

🔁 OpenSearch vs. ELK Stack

Feature	ELK Stack	OpenSearch
Origin	Created by Elastic	Forked from Elasticsearch 7.10 (AWS)
License	SSPL (not fully open-source)	Apache 2.0 (fully open-source)
Security	Paid only	Built-in and free
Dashboards	Kibana	OpenSearch Dashboards
Machine Learning	Paid tier	Free plugins available
Support	Commercial (Elastic)	Community + AWS options
Plugins	Some are paid	All are open-source
Popular In	Enterprises	AWS, open-source users

✅ Which Should You Choose?

- **Choose OpenSearch** if you want a **fully open-source**, community-backed tool with no hidden costs. Great choice for AWS users.
- **Choose ELK Stack** if you need **commercial support**, enterprise features, and are okay with paid plans.
