**Overview of EFK Stack and Its Benefits in Kubernetes**

**What is the EFK Stack?**

The **EFK stack** is a **powerful log management solution** used in Kubernetes to collect, store, and visualize logs. It consists of three main components:

**1. Elasticsearch (E)**

* A **distributed search and analytics engine**
* Stores logs and allows fast searching and filtering
* Supports **scalability** and **high availability**

**2. Fluentd (F)**

* A **lightweight log collector** that gathers logs from nodes, pods, and containers
* Parses and **forwards logs to Elasticsearch**
* Supports **various log sources** and **multiple output formats**

**3. Kibana (K)**

* A **web-based visualization tool** for Elasticsearch data
* Provides dashboards, charts, and search capabilities
* Helps in **log analysis, troubleshooting, and monitoring**

**Why Use EFK in Kubernetes?**

Kubernetes generates a **large volume of logs** from different components, making log management challenging. The **EFK stack** provides the following benefits:

**1. Centralized Log Management**

* Collects logs from all nodes and pods
* Stores logs in Elasticsearch for easy retrieval
* Enables **real-time monitoring** of logs

**2. Easy Log Searching and Filtering**

* Logs can be **queried and filtered** using Kibana
* Helps in troubleshooting **container crashes, errors, and failed deployments**

**3. Improved Observability and Debugging**

* Detect and analyze issues **faster**
* Helps in **tracking API requests, performance bottlenecks, and security threats**

**4. Scalability and High Availability**

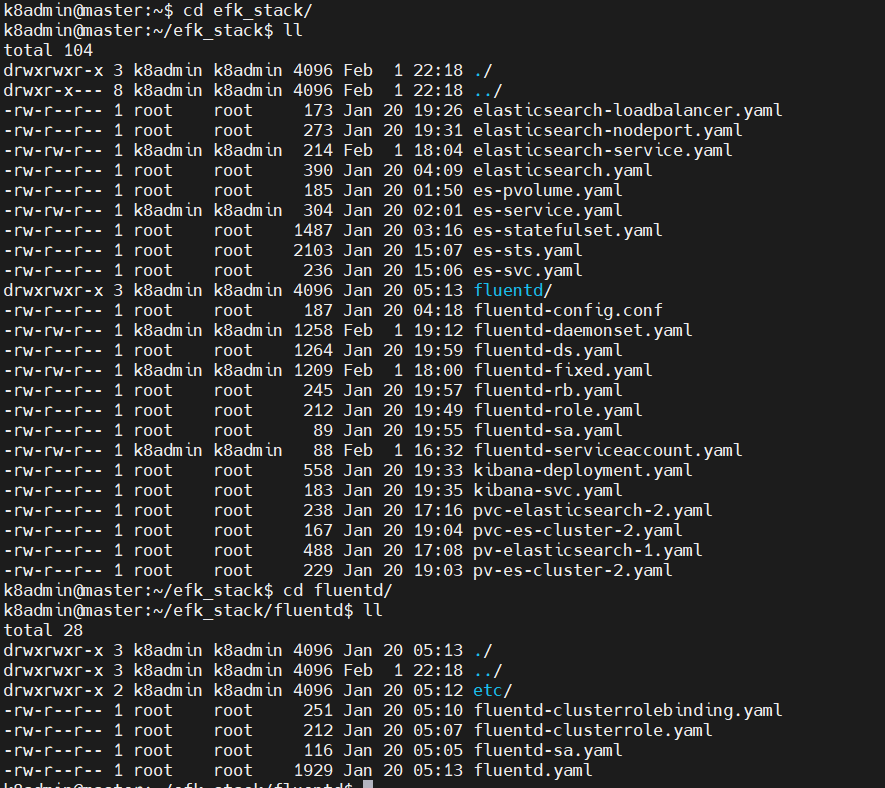
* Elasticsearch can **scale horizontally** as log volume grows
* Fluentd and Kibana support **multi-node deployments**

**5. Customizable and Extensible**

* Fluentd supports **various log sources** (JSON, syslog, application logs, etc.)
* Kibana allows **custom dashboards and alerts**

**Deploying EFK Stack on Kubernetes**

**Here is the list of files I used for the EFK Stack deployment:**

****

**Step 1: Deploy Elasticsearch**

**1.1 Created Persistent Volumes (PVs) for Elasticsearch**

I created the following PV and PVC files for Elasticsearch:

* pv-elasticsearch-1.yaml
* pvc-elasticsearch-2.yaml
* pv-es-cluster-2.yaml
* pvc-es-cluster-2.yaml
* pv.yaml

I applied them using:

kubectl apply -f pv-elasticsearch-1.yaml

kubectl apply -f pvc-elasticsearch-2.yaml

kubectl apply -f pv-es-cluster-2.yaml

kubectl apply -f pvc-es-cluster-2.yaml

kubectl apply -f pv.yaml

**1.2 Deployed Elasticsearch StatefulSet**

I deployed the Elasticsearch StatefulSet using the following files:

* es-statefulset.yaml
* es-sts.yaml

I applied them using:

kubectl apply -f es-statefulset.yaml

kubectl apply -f es-sts.yaml

**1.3 Exposed Elasticsearch Service**

To expose Elasticsearch, I created multiple service configurations:

* es-service.yaml
* elasticsearch-service.yaml
* es-svc.yaml
* elasticsearch-loadbalancer.yaml
* elasticsearch-nodeport.yaml

I applied them using:

kubectl apply -f es-service.yaml

kubectl apply -f elasticsearch-service.yaml

kubectl apply -f es-svc.yaml

kubectl apply -f elasticsearch-loadbalancer.yaml

kubectl apply -f elasticsearch-nodeport.yaml

**Step 2: Deploy Fluentd**

**2.1 Created ServiceAccount and RBAC for Fluentd**

For Fluentd to have proper permissions, I created the following RBAC files:

* fluentd-serviceaccount.yaml
* fluentd-role.yaml
* fluentd-rb.yaml
* fluentd-sa.yaml

I applied them using:

kubectl apply -f fluentd-serviceaccount.yaml

kubectl apply -f fluentd-role.yaml

kubectl apply -f fluentd-rb.yaml

kubectl apply -f fluentd-sa.yaml

**2.2 Deployed Fluentd as DaemonSet**

I created Fluentd DaemonSet configurations using:

* fluentd-daemonset.yaml
* fluentd-ds.yaml
* fluentd-fixed.yaml

I applied them using:

kubectl apply -f fluentd-daemonset.yaml

kubectl apply -f fluentd-ds.yaml

kubectl apply -f fluentd-fixed.yaml

**2.3 Configured Fluentd**

I created and verified the Fluentd configuration file:

* fluent.conf
* fluentd-config.conf

I ensured Fluentd was correctly parsing and forwarding logs to Elasticsearch.

**Step 3: Deploy Kibana**

**3.1 Deployed Kibana Deployment**

I deployed Kibana using:

* kibana-deployment.yaml

I applied it using:

kubectl apply -f kibana-deployment.yaml

**3.2 Exposed Kibana Service**

I created a service for Kibana using:

* kibana-svc.yaml

I applied it using:

kubectl apply -f kibana-svc.yaml

**Step 4: Verify the EFK Stack Deployment**

**4.1 Checked if Pods are Running**

I verified that all Elasticsearch, Fluentd, and Kibana pods were running:

kubectl get pods -n kube-system

**4.2 Verified Logs in Elasticsearch**

To check if Fluentd was sending logs to Elasticsearch, I forwarded the Elasticsearch service and queried indices:

kubectl port-forward svc/elasticsearch 9200:9200

curl -X GET <http://localhost:9200/_cat/indices?v>

**4.3 Accessed Kibana**

I checked the Kibana service:

kubectl get svc | grep kibana

Using **NodePort**, I accessed Kibana at:

http://192.168.239.134:30001/

Once inside Kibana:

* I navigated to **Management > Stack Management > Index Patterns**
* Created an index pattern for logstash-\*
* Started visualizing logs

**conclusion**

I successfully deployed the **EFK (Elasticsearch, Fluentd, Kibana) stack** in Kubernetes using my YAML files. Now, logs are being collected from my Kubernetes cluster and visualized in Kibana.