Kubernetes Observability: Deployment&conifguration of ELK vs EFK(fluentbit) stack on EKS with Terraform & eksctl

Check GitHub for helpful DevOps tools:

Michael Robotics
Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics

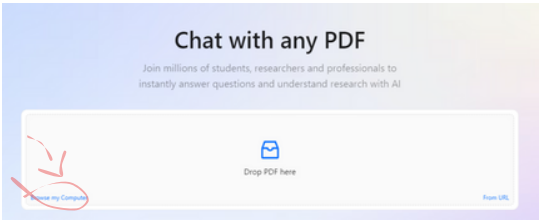
Ask Personal AI Document assistant to learn interactively (FASTER)!

1. Download PDF
2. Go to website
3. Browse file
4. Chat with Document

1 https://github.com/MichaelRobotics/DevOpsTools/blob/main/KubernetesEKSObserv.pdf

Click there to go to ChatPdf website 2

3



Ask questions about document! 4

Complety new to Linux and Networking?

Essential for this PDF is a thorough knowledge of networking. I highly recommend the HTB platform's networking module, which offers extensive information to help build a comprehensive understanding.

HTB - Your Cyber Performance Center

We provide a human-first platform creating and maintaining high performing cybersecurity individuals and organizations.

https://www.hackthebox.com/

What is Kubernetes?

Kubernetes is an open-source platform that automates the deployment, scaling, and management of containerized applications. It helps manage clusters of nodes running containers, ensuring efficient and reliable operation.

How Kubernetes clusters are made?

Kubernetes clusters consist of a control plane and multiple worker nodes. The control plane manages cluster operations, while worker nodes run the actual container workloads.

Why and When use Kubernetes

Kubernetes is ideal for deploying scalable, resilient, and automated containerized applications. It is used when managing multiple containers across different environments is necessary.

Example: Running a microservices-based e-commerce platform that scales up during peak hours.

System Requirements

- RAM: 2 GB per node (1 GB can work for testing but may lead to limited performance)
- 10 GB free storage
- Ubuntu

Kubernetes: Main components & packages

- **kube-apiserver:** Central management component that exposes the Kubernetes API; acts as the front-end for the cluster.
- **etcd:** Distributed key-value store for storing all cluster data, ensuring data consistency across nodes.
- **kube-scheduler:** Assigns pods to available nodes based on resource requirements and policies.
- **kube-controller-manager:** Manages core controllers that handle various functions like node status, replication, and endpoints.
- **kubelet:** Agent that runs on each node, responsible for managing pods and their containers.
- **kube-proxy:** Manages networking on each node, ensuring communication between pods and services within the cluster.

Kubernetes Observability: Intro

1) What is Observability

Observability helps you understand what's happening inside a system, like a website or an app. It has three main parts:

Metrics

Thoser are Numbers that show how well a system is working—like how busy the system is (CPU usage), how many requests it's handling, or how often errors happen.
Metrics are like a health check. They help you see if something's wrong, track changes over time, and set up alerts if things go off track.

Logs

Its detailed diary of events, recording what happened, when it happened, and where. This could include error messages or records of what users did.
Logs are super helpful when something breaks. They help you figure out what went wrong, when it happened, and why.

Traces

A step-by-step map that shows the path a request (like clicking a button on a website) takes through different parts of the system.
Traces help you spot slowdowns or problems in the system, especially when it's made up of many connected parts.

2) Why use observability stacks

The biggest reason to use observability stacks is to quickly identify and fix issues, reducing downtime. For example, if an e-commerce site like Amazon’s checkout slows down, observability tools can pinpoint whether it's a server load spike (metrics), a payment error (logs), or a slow API call (traces).

Without them, you're left guessing and manually sifting through logs, which takes much longer and increases the chances of missing the root cause.

6) EFK(F-FluebtBit) vs ELK(L-Logstash) stack

Fluent Bit runs as a DaemonSet on all nodes, collecting logs from various sources. It acts as a lightweight log forwarder, efficiently shipping logs to Elasticsearch for storage and indexing. Kibana serves as the visualization layer, providing dashboards and generating queries to retrieve logs from Elasticsearch.

Fluent Bit forwards logs, while Logstash aggregates and processes them with tasks like filtering and parsing. However, in most cases, the EFK stack (Elasticsearch, Fluent Bit, Kibana) is enough, as Fluent Bit's lightweight design minimizes resource usage, making Logstash unnecessary unless advanced processing is required.

Kubernetes Observability: EFK deployment&configuration on EKS

1) Create EKS cluster:

Firstly set your cluster name and zones

```
eksctl create cluster --name=observability \
                --region=us-east-1 \
                --zones=us-east-1a,us-east-1b \
                --without-nodegroup
```

configure the EKS cluster to allow the use of IAM roles with Kubernetes service accounts

```
eksctl utils associate-iam-oidc-provider \
  --region us-east-1 \
  --cluster observability \
  --approve
```

Setup eks nodes specification and proper k8s context in your terminal

```
eksctl create nodegroup --cluster=observability \
                --region=us-east-1 \
                --name=observability-ng-private \
                --node-type=t3.medium \
                --nodes-min=2 \
                --nodes-max=3 \
                --node-volume-size=20 \
                --managed \
                --asg-access \
                --external-dns-access \
                --full-ecr-access \
                --appmesh-access \
                --alb-ingress-access \
                --node-private-networking
```

```
aws eks update-kubeconfig --name observability --region us-east-1
```

2) Deploy observability stack

This command creates an IAM role for the EBS CSI controller, allowing it to interact with AWS resources to manage EBS volumes in the Kubernetes cluster, and will be attached to a service account.

```
eksctl create iamserviceaccount \
  --name ebs-csi-controller-sa \
  --namespace kube-system \
  --cluster observability \
  --region us-east-1 \
  --role-name AmazonEKS_EBS_CSI_DriverRole \
  --role-only \
  --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
  --approve
```

This command deploys the AWS EBS CSI driver as an addon to your Kubernetes cluster, using the previously created IAM service account role to securely manage EBS volumes.

```
ARN=$(aws iam get-role --role-name AmazonEKS_EBS_CSI_DriverRole --query 'Role.Arn'
--output text)
```

```
eksctl create addon --cluster observability --region=us-east-1 --name aws-ebs-csi-driver
--version latest \
  --service-account-role-arn $ARN --force
```

Let's create a namespace to be used when the stack is deployed.

```
kubectl create namespace logging
```

This command installs Elasticsearch in the logging namespace, configuring the number of replicas, specifying the storage class, and enabling persistence labels to ensure data is stored on persistent volumes.

```
helm repo add elastic https://helm.elastic.co

helm install elasticsearch \
 --set replicas=1 \
 --set volumeClaimTemplate.storageClassName=gp2 \
 --set persistence.labels.enabled=true elastic/elasticsearch -n logging
```

Check if everything works:

```
laptopdev@laptopdev2:~/Kubernetes$ kubectl get pods -n logging
NAME                     READY   STATUS    RESTARTS   AGE
elasticsearch-master-0   0/1     Running   0          72s
laptopdev@laptopdev2:~/Kubernetes$
laptopdev@laptopdev2:~/Kubernetes$ kubectl get pods -n logging
NAME                     READY   STATUS    RESTARTS   AGE
elasticsearch-master-0   1/1     Running   0          3m39s
laptopdev@laptopdev2:~/Kubernetes$
```

Kibana provides a user-friendly interface for exploring and visualizing data stored in Elasticsearch and is exposed as a LoadBalancer service for external accessibility.

```
helm install kibana --set service.type=LoadBalancer elastic/kibana -n logging
```

This command retrieves the base64-encoded password for the Elasticsearch cluster's master credentials from the Kubernetes secret, which needs to be decoded before use

```
# for username
kubectl get secrets --namespace=logging elasticsearch-master-credentials -
ojsonpath='{.data.username}' | base64 -d
# for password
kubectl get secrets --namespace=logging elasticsearch-master-credentials -
ojsonpath='{.data.password}' | base64 -d
```

Download repo from github:

```
git clone https://github.com/MichaelRobotics/Kubernetes.git
cd Kubernetes/EFK
```

Modify Fluentbit configuration to access Elasticsearch in the EFK stack using the fluentbit-values.yaml file. Add the Elasticsearch username and password, which should be retrieved from the secrets created earlier."

```
EFK > ! fluentbit-values.yaml
389   config:
419     filters: |

427
428     [FILTER]
429         Name lua
430         Match kube.*
431         script /fluent-bit/scripts/setIndex.lua
432         call set_index
433
434   ## https://docs.fluentbit.io/manual/pipeline/outputs
435   outputs: |
436     [OUTPUT]
437         Name es
438         Match kube.*
439         Type  _doc
440         Host elasticsearch-master
441         Port 9200
442         HTTP_User elastic
443         HTTP_Passwd cbTQj1qxRIPNF5uc
```

Install fluentbit

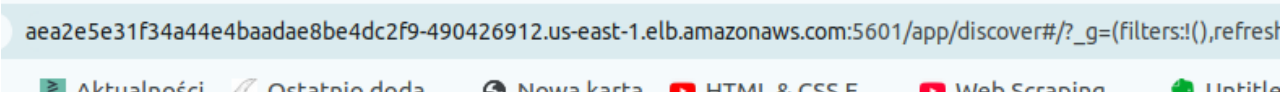
```
helm repo add fluent https://fluent.github.io/helm-charts
helm install fluent-bit fluent/fluent-bit -f
<path_to_cloned_repo>/Kubernetes/EFK/fluentbit-values.yaml -n logging
```

Retrieve the LoadBalancer address and pass the LoadBalancer external IP along with its port to access Elasticsearch.

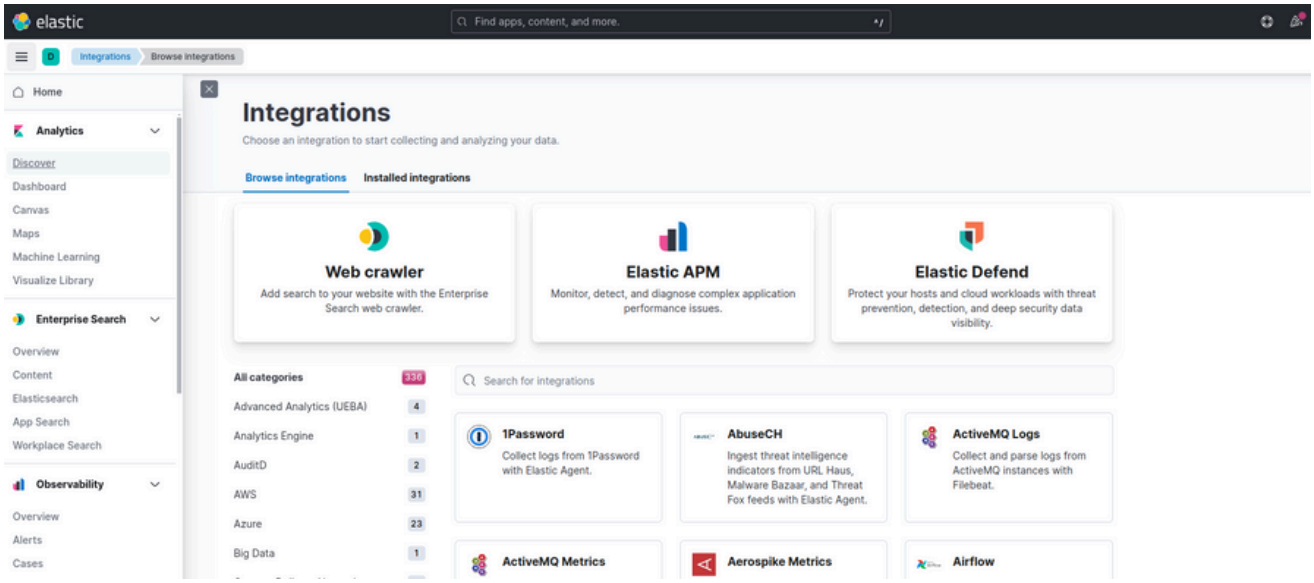
```
kubectl get svc -n logging
```

```
laptopdev@laptopdev2:~/Kubernetes/EFK$ kubectl get svc -n logging
NAME                          TYPE           CLUSTER-IP       EXTERNAL-IP                                                                  PORT(S)             AGE
elasticsearch-master          ClusterIP      10.100.191.93    <none>                                                                       9200/TCP,9300/TCP   26m
elasticsearch-master-headless ClusterIP      None             <none>                                                                       9200/TCP,9300/TCP   26m
fluent-bit                    ClusterIP      10.100.235.205   <none>                                                                       2020/TCP            12m
kibana-kibana                 LoadBalancer   10.100.214.251   aea2e5e31f34a44e4baadae8be4dc2f9-490426912.us-east-1.elb.amazonaws.com   5601:32228/TCP      19m
laptopdev@laptopdev2:~/Kubernetes/EFK$
```

Paste url into your web browser



Log into Elasticsearch using the credentials retrieved from secrets, then navigate to the Discover bar. Since no app is deployed, Elasticsearch will not find any data yet.



3) Deploy Application

```
kubectl create ns dev

<path_to_cloned_repo> kubectl  apply -k .
```

```
laptopdev@laptopdev2:~/Kubernetes/EFK/app$ kubectl create ns dev
namespace/dev created
laptopdev@laptopdev2:~/Kubernetes/EFK/app$ kubectl apply -k .
service/a-service created
service/b-service created
deployment.apps/service-a-deployment created
deployment.apps/service-b-deployment created
```

Check if pods are running and check if they generate any logs

```
kubectl get pods -n dev

kubectl logs pod/<pod_name> -n dev
```

```
laptopdev@laptopdev2:~/Kubernetes/EFK/app$ kubectl get pods -n dev
NAME                                  READY   STATUS    RESTARTS   AGE
service-a-deployment-565d5c86d5-s297h   1/1     Running   0          6m16s
service-b-deployment-7b466747cf-98xc5   1/1     Running   0          6m16s
laptopdev@laptopdev2:~/Kubernetes/EFK/app$ kubectl logs pod/service-a-deployment-565d5c86d5-s297h -n dev
Tracing initialized
Service A is running on port 3001
::ffff:192.168.115.127 - - [09/Feb/2025:15:07:53 +0000] "GET / HTTP/1.1" 200 26
```

Check if fluentbit captures any logs:

```
kubectl logs pod/<fluentbit_pod> -n logging
```

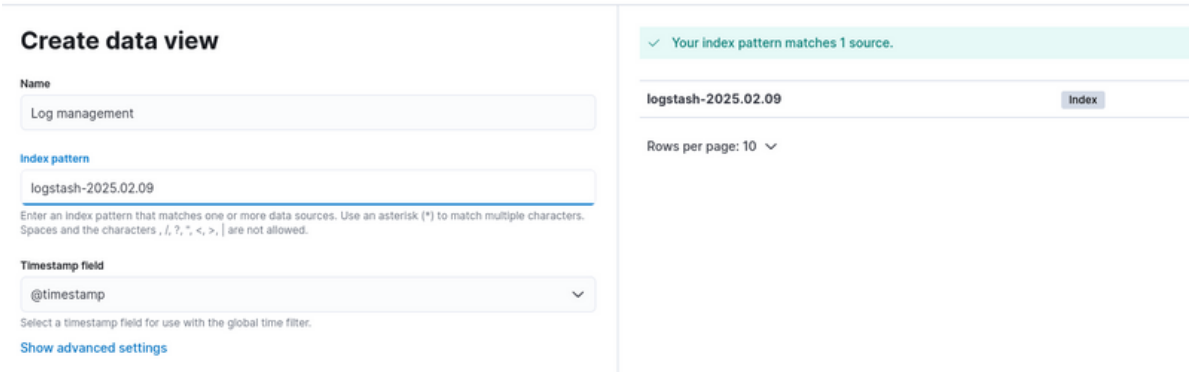
```
laptopdev@laptopdev2:~/Kubernetes/EFK/app$ kubectl logs pod/fluent-bit-2j8d2 -n logging
Fluent Bit v3.2.4
* Copyright (C) 2015-2024 The Fluent Bit Authors
* Fluent Bit is a CNCF sub-project under the umbrella of Fluentd
* https://fluentbit.io
```

Search for info message with path to logs inside any pod of deployed application:

```
[2025/02/09 15:05:23] [ warn] [engine] failed to flush chunk '1-1739113471.508903373.flb', retry in 17 seconds: task_id=3, input=systemd.1 > output=es.1 (out_id=1)
[2025/02/09 15:05:25] [ info] [filter:kubernetes:kubernetes.0]  token updated
[2025/02/09 15:05:25] [ info] [input:tail:tail.0] inotify_fs_add(): inode=29559363 watch_fd=17 name=/var/log/containers/service-a-deployment-565d5c86d5-s297h_dev_serv
ice-a-4dd86b43bbbf762f4bf32a5a3ddd14d6af7ab77b793eab7f6cce812962e99708.log
[2025/02/09 15:05:27] [error] [output:es:es.1] HTTP status=401 URI=/_bulk, response:
{"error":{"root_cause":[{"type":"security_exception","reason":"unable to authenticate user [elastic] for REST request [/_bulk]","header":{"WWW-Authenticate":["Basic r
```

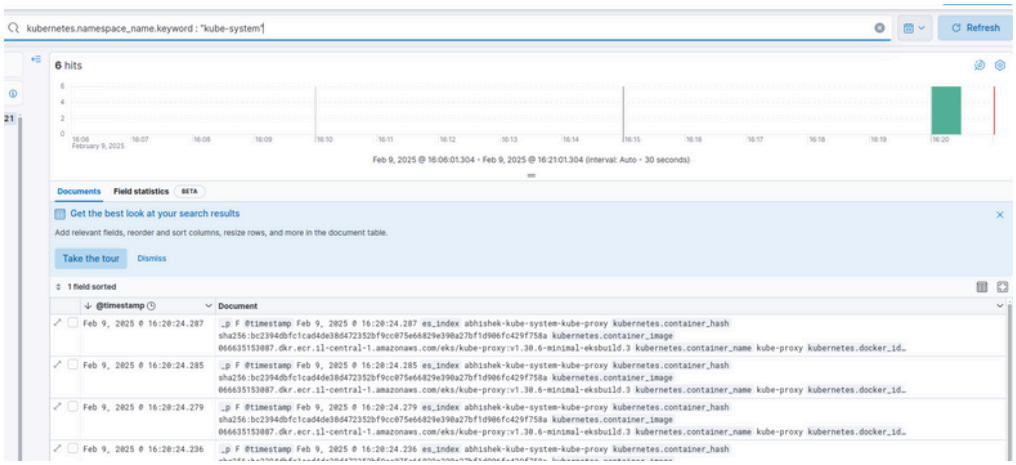
4) Connect Application

Back to elasticsearch. Create new data view, name it as you wish and in “index patterns” write any index pattern which elasticsearch found inside captured logs:



Apply configuration. Kibana dashboard should become visible. By default grafana presentes timestamped logs graph and all logs messages sorted in predefined order:

Kibana have its own query language. For example we queried only logs from namespace “kube-system”:



5) Customize Fluentbit

Fluentbit is vendor-neutral because it uses Lua, allowing users to easily switch to other platforms like Splunk or Logstash without being locked into a specific vendor's ecosystem. This flexibility makes Fluentbit a versatile choice for log management and data processing.

There are 4 most important parts of fluentbit configuration:

Service

The Service section in Fluentbit defines global settings like logging behavior and runtime options, ensuring it operates under the desired conditions. It configures the service environment and specifies the configuration file location.

```
config:
  service: |
    [SERVICE]
        Daemon Off
        Flush {{ .Values.flush }}
        Log_Level {{ .Values.logLevel }}
        Parsers_File /fluent-bit/etc/parsers.conf
        Parsers_File /fluent-bit/etc/conf/custom_parsers.conf
        HTTP_Server On
        HTTP_Listen 0.0.0.0
        HTTP_Port {{ .Values.metricsPort }}
        Health_Check On
```

Input

The Input section specifies the data sources that Fluentbit will collect logs or metrics from, such as files, system logs, or network ports. It configures how and where Fluentbit retrieves the log data before processing.

```
  ## https://docs.fluentbit.io/manual/pipeline/inputs
  inputs: |
    [INPUT]
        Name tail
        Path /var/log/containers/*.log
        multiline.parser docker, cri
        Tag kube.*
        Mem_Buf_Limit 5MB
        Skip_Long_Lines On

    [INPUT]
        Name systemd
        Tag host.*
        Systemd_Filter _SYSTEMD_UNIT=kubelet.service
        Read_From_Tail On
```

Output

The Output section defines the destinations where Fluentbit will send the processed log data, such as cloud platforms, databases, or other log collectors like Elasticsearch or Splunk. This section determines where your logs will be forwarded after they are parsed and filtered.

```
outputs: |
  [OUTPUT]
      Name es
      Match kube.*
      Type  _doc
      Host elasticsearch-master
      Port 9200
      HTTP_User elastic
      HTTP_Passwd wihUATeKeOPujkNn
      tls On
      tls.verify Off
      Logstash_Format On
      Logstash_Prefix logstash
      Retry_Limit False
      Suppress_Type_Name On

  [OUTPUT]
      Name es
      Match host.*
      Type  _doc
      Host elasticsearch-master
      Port 9200
      HTTP_User elastic
      HTTP_Passwd cbTQj1qxRIPNF5uc
      tls On
      tls.verify Off
      Logstash_Format On
```

Filters

The Filters section allows Fluentbit to modify or enrich logs before they are sent to the output. Filters can be used to add metadata, change the log format, or remove unnecessary data, ensuring the output meets specific requirements or standards.

```
## https://docs.fluentbit.io/manual/pipeline/filters
filters: |
  [FILTER]
      Name kubernetes
      Match kube.*
      Merge_Log On
      Keep_Log Off
      K8S-Logging.Parser On
      K8S-Logging.Exclude On

  [FILTER]
      Name lua
      Match kube.*
      script /fluent-bit/scripts/setIndex.lua
      call set_index
```

Fluentbit can execute Lua scripts during log processing, allowing for tailored log handling. In this case, the setIndex.lua script sets the es_index field based on the Kubernetes namespace and container name, and skips logs from the "logging" namespace.

```
luaScripts:
  setIndex.lua: |
    function set_index(tag, timestamp, record)
        index = "abhishek-"
        if record["kubernetes"] ~= nil then
            if record["kubernetes"]["namespace_name"] == "logging" then
                return -1, timestamp, record  -- Skip logs from the logging namespace
            end
            if record["kubernetes"]["namespace_name"] ~= nil then
                if record["kubernetes"]["container_name"] ~= nil then
                    record["es_index"] = index
                        .. record["kubernetes"]["namespace_name"]
                        .. "-"
                        .. record["kubernetes"]["container_name"]
                    return 1, timestamp, record
                end
                record["es_index"] = index
                    .. record["kubernetes"]["namespace_name"]
                return 1, timestamp, record
            end
        end
        return 1, timestamp, record
    end
```

Kubernetes Observability: ELK deployment&configuration on EKS

1) Intro

The ELK Stack consists of Elasticsearch (for storing and searching data), Logstash (for processing and transforming logs), and Kibana (for visualizing data). Filebeat acts as a lightweight log shipper, scraping log files and sending the data to Logstash for further processing or directly to Elasticsearch. Logstash processes and enriches the data before forwarding it to Elasticsearch, where Kibana retrieves and visualizes the stored information.

2) VPC and EKS terraform files

Download git repo

```
git clone https://github.com/MichaelRobotics/Kubernetes.git
cd Kubernetes/ELK
```

Setup is compsed from provider.tf where we can choose AWS region for our cluster an main.tf with eks & vpc configuration.
You can customize parameters like cluster_name, endpoint_public_access, instance_types, cluster size, and node instance types. Remember to change key_pair to existing key pair in us-west-1 region in your AWS acoount.

```
module "eks" {
  source                  = "./modules/eks"
  aws_public_subnet       = module.vpc.aws_public_subnet
  vpc_id                  = module.vpc.vpc_id
  cluster_name            = "module-eks-${random_string.suffix.result}"
  endpoint_public_access  = true
  endpoint_private_access = false
  public_access_cidrs     = ["0.0.0.0/0"]
  node_group_name         = "michaelrobotics"
  scaling_desired_size    = 1
  scaling_max_size        = 1
  scaling_min_size        = 1
  instance_types          = ["t3.large"]
  key_pair                = "TestKeyPair"
}
```

Customizable VPC parameters include vpc_cidr (VPC IP range), public_cidrs (subnet IP ranges), access_ip (allowed IPs for access), public_sn_count (number of public subnets), and map_public_ip_on_launch (controls public IP assignment for instances).

```
module "vpc" {
  source                  = "./modules/vpc"
  tags                    = "michaelrobotics"
  instance_tenancy        = "default"
  vpc_cidr                = "10.0.0.0/16"
  access_ip               = "0.0.0.0/0"
  public_sn_count         = 2
  public_cidrs            = ["10.0.1.0/24", "10.0.2.0/24"]
  map_public_ip_on_launch = true
  rt_route_cidr_block     = "0.0.0.0/0"

}
  (Alt+C) Duo Quick Chat
```

in directory ELK, initialize and apply configuration.

```
terraform init
terraform apply
```

3) ELK stack yaml files

Each stack component is deployed as a deployment.

Elasticsearch 1 replica deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: elasticsearch
  namespace: elk
spec:
  replicas: 1
  selector:
```

Filebeat 1 replica deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: filebeat
  namespace: elk
  labels:
    app: filebeat
spec:
  replicas: 1
```

Kibana 1 replica deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kibana
  namespace: elk
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kibana
```

Logstash as 1 replica deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: logstash
  namespace: elk
spec:
  replicas: 1
  selector:
    matchLabels:
      app: logstash
```

It's important to note that Filebeat is deployed as a Deployment, meaning it won't run on every node and won't scrape logs from all pods. Filebeat configurations are set via a ConfigMap, collecting logs from /var/log/*.log on the node and forwarding them to the Logstash service on port 5044.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: filebeat-config
  namespace: elk
data:
  filebeat.yml: |
    filebeat.inputs:
    - type: log
      enabled: true
      paths:
        - /var/log/*.log

    output.logstash:
      hosts: ["logstash:5044"]
```

Logstash configurations are mounted as a ConfigMap. It listens on port 5044 for logs from Filebeat agents and forwards them to the Elasticsearch service on port 9200. Logs are indexed based on the pattern specified in the "index" setting.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: logstash-config
  namespace: elk
data:
  logstash.conf: |
    input {
      beats {
        port => 5044
      }
    }
    output {
      elasticsearch {
        hosts => ["elasticsearch:9200"]
        index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
      }
    }
```

The configurations for both Filebeat and Logstash are fairly basic. To modify them or deploy Filebeat as a DaemonSet, refer to the documentation and customize the project to fit your needs.

now create elk namespace

```
kubectl create ns elk
```

```
laptopdev@laptopdev2:~/Kubernetes/ELK/modules$ kubectl create ns elk
namespace/elk created
```

Depploy application and check if everything work just fine.

```
kubectl apply -f elk/
```

```
laptopdev@laptopdev2:~/Kubernetes/ELK/modules$ kubectl apply -f elk/
deployment.apps/elasticsearch created
service/elasticsearch created
clusterrolebinding.rbac.authorization.k8s.io/filebeat-cluster-role-binding unchanged
clusterrole.rbac.authorization.k8s.io/filebeat-cluster-role unchanged
configmap/filebeat-config created
deployment.apps/filebeat created
serviceaccount/filebeat created
deployment.apps/kibana created
service/kibana created
configmap/logstash-config created
deployment.apps/logstash created
service/logstash created
```

After a few seconds all pods were created.

```
kubectl get pods -n elk
```

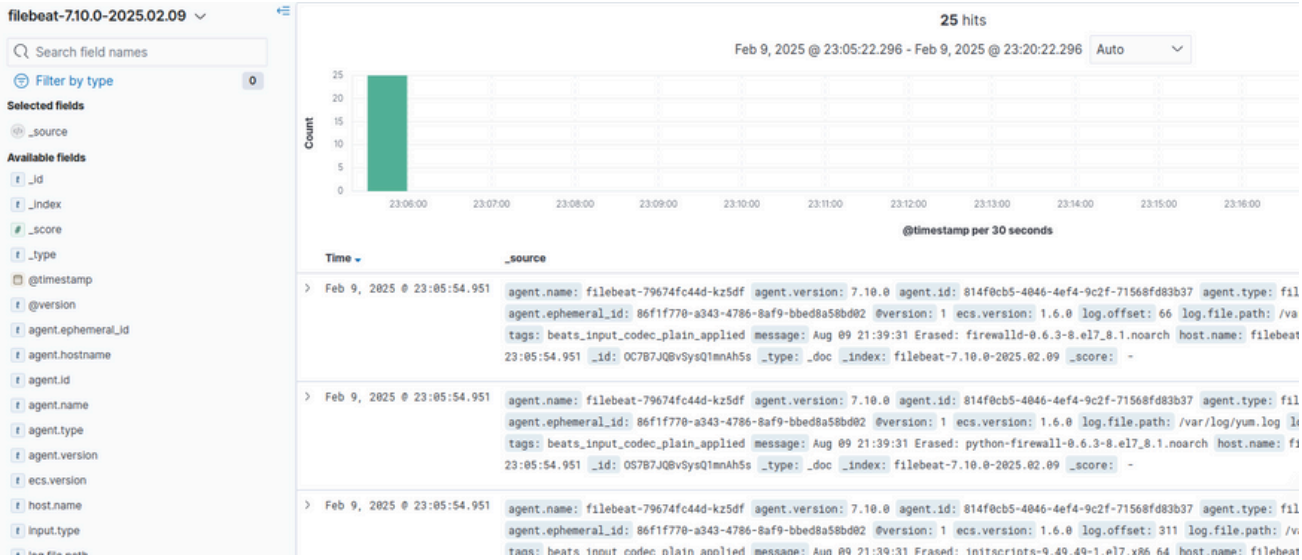
```
laptopdev@laptopdev2:~/Kubernetes/ELK/modules$ kubectl get pods -n elk
NAME                              READY   STATUS              RESTARTS   AGE
elasticsearch-5495ddc97c-qxtv2    0/1     ContainerCreating   0          14s
filebeat-79674fc44d-kz5df         0/1     ContainerCreating   0          12s
kibana-7dd6fd6fcc-dqgtk           0/1     ContainerCreating   0          11s
logstash-5fd545c6c8-rn5zn         0/1     ContainerCreating   0          9s
laptopdev@laptopdev2:~/Kubernetes/ELK/modules$ kubectl get pods -n elk
NAME                              READY   STATUS    RESTARTS   AGE
elasticsearch-5495ddc97c-qxtv2    1/1     Running   0          111s
filebeat-79674fc44d-kz5df         1/1     Running   0          109s
kibana-7dd6fd6fcc-dqgtk           1/1     Running   0          108s
logstash-5fd545c6c8-rn5zn         1/1     Running   0          106s
```

Now get kibana service and navigate towards its GUI in web browser

```
kubectl get svc -n elk
```

```
laptopdev@laptopdev2:~/Kubernetes/ELK/modules$ kubectl get svc -n elk
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP                                                                    PORT(S)          AGE
elasticsearch   LoadBalancer   172.20.49.83     a30a3ec8d88d6476f87e0d15e6c67836-634110686.us-west-2.elb.amazonaws.com         9200:32425/TCP   2m35s
kibana          LoadBalancer   172.20.152.208   ab9dcc1a071634fb2abd3b2ab7a3f03e-1973390940.us-west-2.elb.amazonaws.com        5601:30618/TCP   2m31s
logstash        ClusterIP      172.20.85.120    <none>                                                                         5044/TCP         2m30s
```

Voila! Kibana now displays graphs and logs scraped by Filebeat. Run your queries to search for logs containing useful information.



common troubleshooting

1) Logs Not Appearing in Kibana (ELK Stack)

Cause: Filebeat is not forwarding logs to Logstash or Logstash isn't connecting to Elasticsearch.
Solution: Check Filebeat logs with kubectl logs <filebeat-pod> -n <namespace>, verify Logstash is listening on port 5044, and ensure Elasticsearch is reachable at port 9200

2) Fluent Bit Not Forwarding Logs (EFK Stack)

Cause: Incorrect output configuration or network issues between Fluent Bit and Elasticsearch.
Solution: Validate Fluent Bit configuration (output section), check logs with kubectl logs <fluent-bit-pod>, and ensure the Elasticsearch service is accessible.

3) Elasticsearch Pods CrashLoopBackOff

Cause: Insufficient memory or disk space leading to resource exhaustion.
Solution: Check pod events with kubectl describe pod <es-pod>, increase resource limits in the deployment, and monitor disk usage with df -h inside the pod.

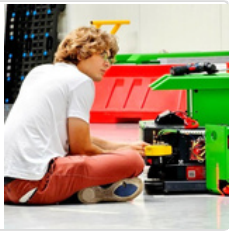
4) Kibana Fails to Connect to Elasticsearch

Cause: Incorrect Elasticsearch endpoint in Kibana config or Elasticsearch service is down.
Solution: Verify Kibana config (elasticsearch.hosts setting), check service status with kubectl get svc -n <namespace>, and ensure Elasticsearch pods are healthy.

5) Check my Kubernetes Troubleshooting series:

Michael Robotics
Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics



Learn more about Kubernetes

Check Kubernetes and piyushsachdeva - great docs!

Setup a Multi Node Kubernetes Cluster

kubeadm is a tool to bootstrap the Kubernetes cluster

https://github.com/piyushsachdeva/CKA-2024/tree/main/Resources/Day27

Kubernetes Documentation

This section lists the different ways to set up and run Kubernetes

https://kubernetes.io/docs/setup/

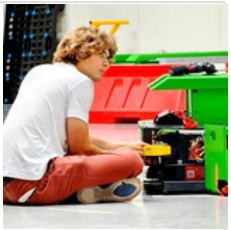
Share, comment, DM and check GitHub for scripts & playbooks created to automate process.

Check my GitHub

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics



PS.

If you need a playbook or bash script to manage KVM on a specific Linux distribution, feel free to ask me in the comments or send a direct message!