**Objective:** Implement a robust logging solution for EKS.

**Tasks:**

1. Set up Fluentd within EKS.

2. Configure AWS Elasticsearch.

3. Visualize logs using Kibana.

**Prerequisites:**

* An active AWS account.
* An EKS cluster up and running.
* kubectl configured to interact with your EKS cluster.
* aws CLI installed and configured for your environment.
* Proper IAM roles and policies attached for EKS to interact with other AWS services.

**Documentation:**

- Importance of centralized logging.

- Integrating Fluentd and Elasticsearch.

- Log analysis with Kibana.

**Tasks:**

1. Set up Fluentd within EKS.

Enabling IAM roles for service accounts on your cluster.

To use IAM roles for service accounts in your cluster, we will first create an OIDC identity provider.

| eksctl utils associate-iam-oidc-provider \  --cluster eks \  --approve |
| --- |

If **envsubst** isn't installed, you can usually install it as part of the **gettext** package on **Ubuntu** with:

| sudo apt-get update && sudo apt-get install -y gettext-base |
| --- |

Now open this editor: **nano fluent-bit-policy.json**

| {  "Version": "2012-10-17",  "Statement": [  {  "Action": [  "es:ESHttp\*"  ],  "Resource": "arn:aws:es:${AWS\_REGION}:${ACCOUNT\_ID}:domain/${ES\_DOMAIN\_NAME}",  "Effect": "Allow"  }  ] } |
| --- |

| envsubst < fluent-bit-policy.json > temp-fluent-bit-policy.json |
| --- |

| mv fluent-bit-policy.json alnafi-temp-fluent-bit-policy.json mv temp-fluent-bit-policy.json fluent-bit-policy.json |
| --- |

| aws iam create-policy \  --policy-name fluent-bit-policy \  --policy-document file://fluent-bit-policy.json |
| --- |

**Create an IAM role**

Finally, create an IAM role for the fluent-bit Service Account in the **logging namespace**.

| kubectl create namespace logging  eksctl create iamserviceaccount \  --name fluent-bit \  --namespace logging \  --cluster eks \  --attach-policy-arn "arn:aws:iam::${ACCOUNT\_ID}:policy/fluent-bit-policy" \  --approve \  --override-existing-serviceaccounts |
| --- |

First let’s create some variables

| # name of our Amazon OpenSearch cluster export ES\_DOMAIN\_NAME="eks-opensearch" export AWS\_REGION='us-east-1'  export ACCOUNT\_ID="587027614473" # Elasticsearch version #export ES\_VERSION="OpenSearch\_1.0" export ES\_VERSION="OpenSearch\_2.9" # OpenSearch Dashboards admin user export ES\_DOMAIN\_USER="root" # OpenSearch Dashboards admin password export ES\_DOMAIN\_PASSWORD="Alnafi@123" # Set AWS OpenSearch domain endpoint  export OPENSEARCH\_ENDPOINT="https://search-eks-opensearch-mgb5vhfyucwz4mmca4vp24jsoe.us-east-1.es.amazonaws.com" |
| --- |

Now we will create **aws-opensearch ( Elastic-Search)**

| curl -L -o es\_domain.json https://www.eksworkshop.com/intermediate/230\_logging/deploy.files/es\_domain.json |
| --- |

| **cat** **es\_domain**.json |
| --- |

# Create the AWS Openseach cluster ( Elastic Search)

| aws opensearch create-domain \  --cli-input-json file://es\_domain.json |
| --- |

| envsubst < es\_domain.json > es\_domain\_filled.json aws opensearch create-domain --cli-input-json file://es\_domain\_filled.json |
| --- |

**Mapping Roles to Users**

| # We need to retrieve the Fluent Bit Role ARN export FLUENTBIT\_ROLE=$(eksctl get iamserviceaccount --cluster eks --namespace logging -o json | jq '.[].status.roleARN' -r)  # Get the Amazon OpenSearch Endpoint export ES\_ENDPOINT=$(aws opensearch describe-domain --domain-name ${ES\_DOMAIN\_NAME} --output text --query "DomainStatus.Endpoint")  # Update the Elasticsearch internal database curl -sS -u "${ES\_DOMAIN\_USER}:${ES\_DOMAIN\_PASSWORD}" \  -X PATCH \  https://${ES\_ENDPOINT}/\_opendistro/\_security/api/rolesmapping/all\_access?pretty \  -H 'Content-Type: application/json' \  -d' [  {  "op": "add", "path": "/backend\_roles", "value": ["'${FLUENTBIT\_ROLE}'"]  } ] ' |
| --- |

**DEPLOY FLUENT BIT**

Let’s start by downloading the **fluentbit.yaml** deployment file and **replace** some **variables**.

| # get the Amazon OpenSearch Endpoint export ES\_ENDPOINT=$(aws es describe-elasticsearch-domain --domain-name ${ES\_DOMAIN\_NAME} --output text --query "DomainStatus.Endpoint")  curl -Ss https://www.eksworkshop.com/intermediate/230\_logging/deploy.files/fluentbit.yaml \  | envsubst > fluentbit.yaml |
| --- |

| curl -L -Ss https://www.eksworkshop.com/intermediate/230\_logging/deploy.files/fluentbit.yaml | envsubst > fluentbit.yaml |
| --- |

| kubectl apply -f fluentbit.yaml kubectl --namespace=logging get pods |
| --- |

**Documentation:**

Importance of Centralized Logging:

Centralized logging in a cloud environment, especially in Kubernetes, is critical for the following reasons:

* **Troubleshooting:** Centralized logs help in troubleshooting issues across multiple pods and services.
* **Monitoring:** It allows monitoring application performance and system health in real time.
* **Security:** Logs are vital for security audits and anomaly detection.
* **Compliance:** Maintaining logs is often required for regulatory compliance.

**Integrating Fluentd and Elasticsearch:**

Fluentd is used as the data collector that streams logs to Elasticsearch, which is a search and analytics engine. Integration involves configuring Fluentd with the appropriate output plugin for Elasticsearch and ensuring Fluentd runs on all nodes in the cluster to collect logs.

**Log Analysis with Kibana:**

Once logs are stored in Elasticsearch, Kibana provides powerful analysis and visualization capabilities. Users can create dashboards to track certain metrics or logs, search through log data, and set up alerts based on certain events or log patterns.