EFK Stack Deployment Documentation

Overview

This document provides a comprehensive guide to deploying the EFK (Elasticsearch, Fluentd, Kibana) stack on Kubernetes with S3 backups and log archiving.

Architecture Diagram

Application Pods → Fluentd (Log Collector) → Elasticsearch (Storage) → Kibana (Visualization)

↓

S3 Bucket

├── info/ (INFO logs)

└── error/ (ERROR logs)

Prerequisites

- Kubernetes cluster (k3s used in this setup)
- AWS account with S3 bucket
- AWS IAM credentials with S3 access
- kubectl configured

Step-by-Step Deployment

1. Namespace Setup

```
kubectl create namespace logging
```

2. Elasticsearch Deployment

File: `elasticsearch-deployment.yml`

```yaml
# elasticsearch-deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: elasticsearch
 namespace: default
spec:
 replicas: 1
 selector:
   matchLabels:
     app: elasticsearch
 template:
   metadata:
     labels:
       app: elasticsearch
   spec:
     containers:
     - name: elasticsearch
       image: docker.elastic.co/elasticsearch/elasticsearch:8.11.0
       ports:
       - containerPort: 9200
       - containerPort: 9300
       env:
       - name: ES_JAVA_OPTS
         value: "-Xms512m -Xmx512m"
       - name: discovery.type
         value: single-node
       - name: xpack.security.enabled
         value: "false"
       volumeMounts:
       - name: elasticsearch-config
         mountPath: /usr/share/elasticsearch/config/elasticsearch.yml
         subPath: elasticsearch.yml
       - name: data
         mountPath: /usr/share/elasticsearch/data
     volumes:
     - name: elasticsearch-config
       configMap:
```

```
          name: elasticsearch-config
      - name: data
        emptyDir: {}
```

File: `elasticsearch-service.yml`

```
# elasticsearch-service.yml
apiVersion: v1
kind: Service
metadata:
 name: elasticsearch
 namespace: default
spec:
 selector:
   app: elasticsearch
 ports:
 - name: http
   port: 9200
   targetPort: 9200
 - name: transport
   port: 9300
   targetPort: 9300
 type: ClusterIP
```

Apply:

```
kubectl apply -f elasticsearch-deployment.yml
kubectl apply -f elasticsearch-service.yml
```

3. Fluentd Configuration

File: `fluentd-config.yml`

```
# fluentd-config.yml
apiVersion: v1
kind: ConfigMap
metadata:
 name: fluentd-config
 namespace: default
data:
 fluent.conf: |
   <system>
    log_level info
  </system>

  <source>
    @type tail
    path /var/log/containers/*.log
    pos_file /var/log/pos/fluentd-containers.log.pos
    tag kubernetes.*
    read_from_head true
    <parse>
      @type regexp
      expression /^(?<time>.+) (?<stream>stdout|stderr) [^ ]*
(?<log>.*)$/
      time_format %Y-%m-%dT%H:%M:%S.%NZ
      keep_time_key true
    </parse>
  </source>

  <filter kubernetes.**>
    @type kubernetes_metadata
  </filter>

  <match kubernetes.**>
    @type elasticsearch
    host elasticsearch
    port 9200
    scheme http
    ssl_verify false
    logstash_format true
    logstash_prefix fluentd
```

```
    include_tag_key true
    type_name _doc
    flush_interval 5s

    <buffer>
      @type memory
      flush_interval 5s
      retry_forever true
    </buffer>
  </match>

  <match **>
    @type stdout
  </match>
```

Apply:

```
kubectl apply -f fluentd-config.yml
```

4. Fluentd DaemonSet

File: `fluentd-daemonset.yml`

```
# fluentd-daemonset-fixed-mounts.yml
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: fluentd
 namespace: default
 labels:
   app: fluentd
   logging: "true"
spec:
 selector:
   matchLabels:
     app: fluentd
 template:
   metadata:
     labels:
```

```
      app: fluentd
      logging: "true"
  spec:
    serviceAccountName: fluentd
    tolerations:
    - key: node-role.kubernetes.io/control-plane
      operator: Exists
      effect: NoSchedule
    - key: node-role.kubernetes.io/master
      operator: Exists
      effect: NoSchedule
    containers:
    - name: fluentd
      image:
fluent/fluentd-kubernetes-daemonset:v1.16.2-debian-elasticsearch8-1.0
      env:
      - name: FLUENT_ELASTICSEARCH_HOST
        value: "elasticsearch"
      - name: FLUENT_ELASTICSEARCH_PORT
        value: "9200"
      - name: FLUENT_ELASTICSEARCH_SCHEME
        value: "http"
      - name: FLUENT_ELASTICSEARCH_SSL_VERIFY
        value: "false"
      - name: K8S_NODE_NAME
        valueFrom:
          fieldRef:
            fieldPath: spec.nodeName
      - name: FLUENT_UID
        value: "0"
      resources:
        limits:
          memory: 512Mi
        requests:
          cpu: 100m
          memory: 200Mi
      volumeMounts:
      - name: varlog
        mountPath: /var/log
      - name: varlibdockercontainers
        mountPath: /var/lib/docker/containers
        readOnly: true
      - name: dockercontainers
```

```
        mountPath: /var/log/containers
        readOnly: true
      - name: fluentd-config
        mountPath: /fluentd/etc/fluent.conf
        subPath: fluent.conf
      - name: fluentd-pos
        mountPath: /var/log/pos
    terminationGracePeriodSeconds: 30
    volumes:
    - name: varlog
      hostPath:
        path: /var/log
    - name: varlibdockercontainers
      hostPath:
        path: /var/lib/docker/containers
    - name: dockercontainers
      hostPath:
        path: /var/log/containers
    - name: fluentd-config
      configMap:
        name: fluentd-config
    - name: fluentd-pos
      emptyDir: {}
```

Apply:

```
kubectl apply -f fluentd-daemonset.yml
```

5. AWS S3 Credentials

```
kubectl create secret generic aws-s3-credentials \
 --namespace=logging \
 --from-literal=AWS_ACCESS_KEY_ID=YOUR_ACCESS_KEY \
 --from-literal=AWS_SECRET_ACCESS_KEY=YOUR_SECRET_KEY
```

6. Fluentd for S3

```yaml
# fluentd-daemonset-s3.yml
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: fluentd-s3
 namespace: default
 labels:
   app: fluentd-s3
   logging: "true"
spec:
 selector:
   matchLabels:
     app: fluentd-s3
 template:
   metadata:
     labels:
       app: fluentd-s3
       logging: "true"
   spec:
     serviceAccountName: fluentd
     tolerations:
     - key: node-role.kubernetes.io/control-plane
       operator: Exists
       effect: NoSchedule
     - key: node-role.kubernetes.io/master
       operator: Exists
       effect: NoSchedule
     containers:
     - name: fluentd-s3
       image: fluent/fluentd-kubernetes-daemonset:v1.19-debian-s3-1
       env:
       - name: AWS_ACCESS_KEY_ID
         valueFrom:
           secretKeyRef:
             name: aws-s3-credentials
             key: AWS_ACCESS_KEY_ID
       - name: AWS_SECRET_ACCESS_KEY
         valueFrom:
           secretKeyRef:
             name: aws-s3-credentials
             key: AWS_SECRET_ACCESS_KEY
```

```
      - name: S3_BUCKET
        value: "cloudl-efs-poc"
      - name: S3_REGION
        value: "eu-north-1"
      resources:
        limits:
          memory: 512Mi
        requests:
          cpu: 100m
          memory: 200Mi
      volumeMounts:
      - name: varlog
        mountPath: /var/log
      - name: varlibdockercontainers
        mountPath: /var/lib/docker/containers
        readOnly: true
      - name: fluentd-s3-config
        mountPath: /fluentd/etc/fluent.conf
        subPath: fluent.conf
      - name: fluentd-pos
        mountPath: /var/log/pos
      - name: fluentd-s3-buffer
        mountPath: /var/log/fluentd/s3
    volumes:
    - name: varlog
      hostPath:
        path: /var/log
    - name: varlibdockercontainers
      hostPath:
        path: /var/lib/docker/containers
    - name: fluentd-s3-config
      configMap:
        name: fluentd-s3-config
    - name: fluentd-pos
      emptyDir: {}
    - name: fluentd-s3-buffer
      emptyDir: {}
```

Config S3

```
# fluentd-s3-config.yml
apiVersion: v1
kind: ConfigMap
metadata:
 name: fluentd-s3-config
 namespace: default
data:
 fluent.conf: |
   <system>
     log_level info
   </system>

      <filter kubernetes.**>
     @type kubernetes_metadata
   </filter><source>
     @type tail
     path /var/log/containers/*.log
     pos_file /var/log/pos/fluentd-containers.log.pos
     tag kubernetes.*
     read_from_head true
     <parse>
       @type regexp
       expression /^(?<time>.+) (?<stream>stdout|stderr) [^ ]*
(?<log>.*)$/
       time_format %Y-%m-%dT%H:%M:%S.%NZ
       keep_time_key true
     </parse>
   </source>



   # Extract log level from log content for better categorization
   <filter kubernetes.**>
     @type record_transformer
     enable_ruby true
     <record>
       log_level ${if record["log"].match(/(ERROR|Error|error)/i);
"error"; elsif
record["log"].match(/(WARN|Warn|warn|WARNING|Warning|warning)/i);
"warn"; elsif record["log"].match(/(INFO|Info|info)/i); "info"; elsif
record["log"].match(/(DEBUG|Debug|debug)/i); "debug"; else; "other";
end}
     </record>
```

```
  </filter>

  # Add tags to differentiate log levels for S3
  <match kubernetes.**>
    @type rewrite_tag_filter
    <rule>
      key log_level
      pattern /^error$/
      tag s3.error.${tag}
    </rule>
    <rule>
      key log_level
      pattern /^warn$/
      tag s3.warn.${tag}
    </rule>
    <rule>
      key log_level
      pattern /^info$/
      tag s3.info.${tag}
    </rule>
    <rule>
      key log_level
      pattern /^debug$/
      tag s3.debug.${tag}
    </rule>
    <rule>
      key log_level
      pattern /^other$/
      tag s3.other.${tag}
    </rule>
  </match>

  # Send ERROR logs to S3 error folder
  <match s3.error.kubernetes.**>
    @type s3
    aws_key_id "#{ENV['AWS_ACCESS_KEY_ID']}"
    aws_sec_key "#{ENV['AWS_SECRET_ACCESS_KEY']}"
    s3_bucket "cloudl-efs-poc"
    s3_region "eu-north-1"
    path "logs/error/%Y/%m/%d/"
    time_slice_format %Y%m%d%H
    <buffer time>
      @type file
```

```
      path /var/log/fluentd/s3/error
      timekey 1h
      timekey_wait 10m
      timekey_use_utc true
      chunk_limit_size 256m
    </buffer>
    <format>
      @type json
    </format>
  </match>

  # Send WARN logs to S3 warn folder
  <match s3.warn.kubernetes.**>
    @type s3
    aws_key_id "#{ENV['AWS_ACCESS_KEY_ID']}"
    aws_sec_key "#{ENV['AWS_SECRET_ACCESS_KEY']}"
    s3_bucket "cloudl-efs-poc"
    s3_region "eu-north-1"
    path "logs/warn/%Y/%m/%d/"
    time_slice_format %Y%m%d%H
    <buffer time>
      @type file
      path /var/log/fluentd/s3/warn
      timekey 1h
      timekey_wait 10m
      timekey_use_utc true
      chunk_limit_size 256m
    </buffer>
    <format>
      @type json
    </format>
  </match>

  # Send INFO logs to S3 info folder
  <match s3.info.kubernetes.**>
    @type s3
    aws_key_id "#{ENV['AWS_ACCESS_KEY_ID']}"
    aws_sec_key "#{ENV['AWS_SECRET_ACCESS_KEY']}"
    s3_bucket "cloudl-efs-poc"
    s3_region "eu-north-1"
    path "logs/info/%Y/%m/%d/"
    time_slice_format %Y%m%d%H
    <buffer time>
```

```
      @type file
      path /var/log/fluentd/s3/info
      timekey 1h
      timekey_wait 10m
      timekey_use_utc true
      chunk_limit_size 256m
    </buffer>
    <format>
      @type json
    </format>
  </match>

  # Send DEBUG logs to S3 debug folder
  <match s3.debug.kubernetes.**>
    @type s3
    aws_key_id "#{ENV['AWS_ACCESS_KEY_ID']}"
    aws_sec_key "#{ENV['AWS_SECRET_ACCESS_KEY']}"
    s3_bucket "cloudl-efs-poc"
    s3_region "eu-north-1"
    path "logs/debug/%Y/%m/%d/"
    time_slice_format %Y%m%d%H
    <buffer time>
      @type file
      path /var/log/fluentd/s3/debug
      timekey 1h
      timekey_wait 10m
      timekey_use_utc true
      chunk_limit_size 256m
    </buffer>
    <format>
      @type json
    </format>
  </match>

  # Send other logs to S3 other folder
  <match s3.other.kubernetes.**>
    @type s3
    aws_key_id "#{ENV['AWS_ACCESS_KEY_ID']}"
    aws_sec_key "#{ENV['AWS_SECRET_ACCESS_KEY']}"
    s3_bucket "cloudl-efs-poc"
    s3_region "eu-north-1"
    path "logs/other/%Y/%m/%d/"
    time_slice_format %Y%m%d%H
```

```
    <buffer time>
      @type file
      path /var/log/fluentd/s3/other
      timekey 1h
      timekey_wait 10m
      timekey_use_utc true
      chunk_limit_size 256m
    </buffer>
    <format>
      @type json
    </format>
  </match>

  # Fallback for any unmatched logs
  <match **>
    @type stdout
  </match>
```

7. Kibana Deployment

```
# kibana-deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: kibana
 namespace: default
 labels:
   app: kibana
spec:
 replicas: 1
 selector:
   matchLabels:
     app: kibana
 template:
   metadata:
     labels:
       app: kibana
   spec:
     containers:
```

```
    - name: kibana
      image: docker.elastic.co/kibana/kibana:8.11.0
      ports:
      - containerPort: 5601
      env:
      - name: ELASTICSEARCH_HOSTS
        value: "http://elasticsearch:9200"
      - name: SERVER_HOST
        value: "0.0.0.0"
      - name: SERVER_NAME
        value: "kibana"
      # Since we disabled security, no credentials needed
      - name: ELASTICSEARCH_USERNAME
        value: ""
      - name: ELASTICSEARCH_PASSWORD
        value: ""
      resources:
        requests:
          cpu: 100m
          memory: 512Mi
        limits:
          memory: 1Gi
```

Kibana Service

```
# kibana-service-nodeport.yml
apiVersion: v1
kind: Service
metadata:
 name: kibana
 namespace: default
 labels:
   app: kibana
spec:
 selector:
   app: kibana
 ports:
 - port: 5601
   targetPort: 5601
   nodePort: 30601  # Custom port between 30000-32767
 type: NodePort
```

Apply:

```
kubectl apply -f kibana-deployment.yml
kubectl apply -f kibana-service.yml
```

8. RBAC Configuration

```
# fluentd-rbac.yml
apiVersion: v1
kind: ServiceAccount
metadata:
 name: fluentd
 namespace: default
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: fluentd
 namespace: default
rules:
- apiGroups: [""]
 resources:
 - pods
 - namespaces
 - nodes
 - nodes/proxy
 verbs:
 - get
 - list
 - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: fluentd
 namespace: default
roleRef:
 kind: ClusterRole
 name: fluentd
 apiGroup: rbac.authorization.k8s.io
```

```
subjects:
- kind: ServiceAccount
 name: fluentd
 namespace: default
```

Apply:

```
kubectl apply -f fluentd-rbac.yml
```

Verification Steps

1. Check Pod Status

```
kubectl get pods
kubectl get svc
```

2. Test Log Generation

```
kubectl run test-logger --image=busybox --command -- sh -c "echo 'INFO: Test log message'; sleep 3600"
```

3. Create Index Pattern

1. Go to Stack Management → Index Patterns
2. Create pattern: `fluentd-*`
3. Time field: `@timestamp`

Backup Configuration

Elasticsearch Snapshot to S3

```
# Register S3 repository
curl -X PUT "localhost:9200/_snapshot/my_s3_backup" -H 'Content-Type:
application/json' -d'
{
 "type": "s3",
 "settings": {
   "bucket": "your-backup-bucket",
   "region": "us-east-1"
 }
}'

# Create snapshot
curl -X PUT
"localhost:9200/_snapshot/my_s3_backup/snapshot_1?wait_for_completion=t
rue"
```

Automated Backup CronJob

File: `es-backup-cronjob.yml`

```
# elasticsearch-backup-cronjob.yml
apiVersion: batch/v1
kind: CronJob
metadata:
 name: elasticsearch-backup
 namespace: default
spec:
 schedule: "0 2 * * *"  # Daily at 2 AM
 jobTemplate:
   spec:
     template:
       spec:
         containers:
         - name: backup
           image: curlimages/curl
           env:
           - name: ES_HOST
             value: "elasticsearch:9200"
```

```
            - name: SNAPSHOT_NAME
              value: "snapshot-$(date +%Y%m%d-%H%M%S)"
            command:
            - /bin/sh
            - -c
            - |
              # Register S3 repository
              curl -X PUT "$ES_HOST/_snapshot/my_s3_repository" -H
'Content-Type: application/json' -d'
              {
                "type": "s3",
                "settings": {
                  "bucket": "cloudl-efs-poc",
                  "region": "eu-north-1"
                }
              }'

              # Create snapshot
              curl -X PUT
"$ES_HOST/_snapshot/my_s3_repository/$SNAPSHOT_NAME?wait_for_completion
=true"

              # Cleanup old snapshots (keep last 7)
              curl -X GET "$ES_HOST/_snapshot/my_s3_repository/_all" |
grep -o '"snapshot":"[^"]*"' | cut -d'"' -f4 | sort | head -n -7 |
xargs -I {} curl -X DELETE "$ES_HOST/_snapshot/my_s3_repository/{}"
            volumeMounts:
            - name: aws-credentials
              mountPath: /root/.aws
          restartPolicy: OnFailure
          volumes:
          - name: aws-credentials
            secret:
              secretName: aws-s3-credentials
```

Create a test snapshot:

```
curl -X PUT
"localhost:9200/_snapshot/my_s3_backup/snapshot_1?wait_for_completion=t
rue&pretty"
```

List all snapshots in the repository:

```
curl -X GET "localhost:9200/_snapshot/my_s3_backup/_all?pretty"
```

Security Considerations

- Enable SSL/TLS for production
- Use IAM roles instead of access keys
- Enable S3 bucket encryption
- Set up network policies
- Regular security audits