Hewlett Packard Enterprise HPE Support Center

How to set up an OpenTelemetry Collector (OTEL) in a Kubernetes cluster

Table of contents

Issue

This setup consists of three main components:

1. **OpenTelemetry Collector Configuration**: Defines the behavior of the Collector, including the receivers, processors, and exporters.
2. **OpenTelemetry Collector Deployment**: The actual deployment of the OpenTelemetry Collector within Kubernetes.
3. **Fluent Bit Configuration**: Configures Fluent Bit to forward logs to the OpenTelemetry Collector.

Environment

Ezmeral Unified Analytics Version: 1.3

Cause

This article provides a step-by-step guide on how to set up an OpenTelemetry Collector in a Kubernetes cluster. The goal is to receive logs from Fluent Bit and forward them to another OpenTelemetry host.

Resolution

1. OpenTelemetry Collector Configuration

The first step involves creating a ConfigMap to store the OpenTelemetry Collector configuration. This configuration includes:

- **OTLP receiver**: To listen on port 4318 for HTTP requests.
- **Batch processor**: For performance optimization.
- **OTLP exporter**: To forward data to another OpenTelemetry host.

Below is the configuration for the ConfigMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: otel-collector-config
  namespace: monitoring
data:
  otel-collector-config.yaml: |
    receivers:
      otlp:
        protocols:
          http:
            endpoint: "0.0.0.0:4318"

    processors:
      batch:

    exporters:
      otlp:
        endpoint: "another-otel-host:4317"
        tls:
          insecure: true  # Set to false and configure certs for production use

    service:
      pipelines:
        logs:
          receivers: [otlp]
          processors: [batch]
          exporters: [otlp]
```

Key Points:

- **Receiver (`otlp`)**: Accepts logs via HTTP on port 4318.
- **Processor (`batch`)**: Optimizes log forwarding by batching data.
- **Exporter (`otlp`)**: Forwards logs to the specified endpoint (another OpenTelemetry host). In production, you should enable proper TLS configuration by setting `insecure` to `false.`

2. OpenTelemetry Collector Deployment

Next, we deploy the OpenTelemetry Collector using a Kubernetes `Deployment` resource. This deployment will:

- Use the official OpenTelemetry Collector Docker image.
- Mount the configuration from the previously created ConfigMap.
- Expose port 4318 for receiving logs.

Below is the configuration for the deployment and service:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: otel-collector
  namespace: monitoring
  labels:
    app: otel-collector
spec:
  replicas: 1
  selector:
    matchLabels:
      app: otel-collector
  template:
    metadata:
      labels:
        app: otel-collector
    spec:
      containers:
      - name: otel-collector
        image: otel/opentelemetry-collector:0.72.0
        args:
        - "--config=/conf/otel-collector-config.yaml"
        ports:
        - containerPort: 4318
        volumeMounts:
        - name: otel-collector-config
          mountPath: /conf
      volumes:
      - name: otel-collector-config
        configMap:
          name: otel-collector-config
---
apiVersion: v1
kind: Service
metadata:
  name: otel-collector
  namespace: monitoring
spec:
  selector:
    app: otel-collector
  ports:
  - protocol: TCP
    port: 4318
    targetPort: 4318
```

Key Points:

- **Kubernetes Deployment**: Deploys the OpenTelemetry Collector with the configuration mounted from the ConfigMap.
- **Kubernetes Service**: Exposes the OpenTelemetry Collector within the cluster to allow Fluent Bit to send logs.

3. Fluent Bit Configuration

Now, we update the Fluent Bit configuration to send logs to the OpenTelemetry Collector. This setup includes:

- The OpenTelemetry output plugin.
- Filters to specifically send logs from pods with the label `hpe-ezua/app=ezpresto`.

Here's the Fluent Bit configuration:

```
[SERVICE]
    flush                     1
    Log_Level                 info
    Daemon                    off
    HTTP_Server               On
    HTTP_Listen               0.0.0.0
    HTTP_Port                 2020
    Parsers_File              parsers.conf
    health_check              On

[INPUT]
    Name              tail
    Tag               kube.*
    Path              /var/log/containers/*.log
    Parser            docker
    Mem_Buf_Limit     50MB
    Buffer_Max_Size   1MB
    Skip_Long_Lines   Off
    Read_From_Head    true
    Refresh_Interval  10

[FILTER]
    Name                  kubernetes
    Match                 kube.*
    Merge_Log             On
    Keep_Log              On
    Labels                On

# Output to OpenTelemetry Collector for ezpresto pods
[OUTPUT]
    Name  opentelemetry
    Match kube.*
    Host  otel-collector.monitoring.svc.cluster.local
    Port  4318
    Metrics_uri /v1/metrics
    Logs_uri /v1/logs
    Tls On
    Tls.verify Off

    <Rule>
        Key_exists $kubernetes['labels']['hpe-ezua/app']
        Key_value_equals $kubernetes['labels']['hpe-ezua/app'] ezpresto
    </Rule>
```

Key Points:

- **Input Section**: Fluent Bit collects logs from container files and system logs.
- **Filters**: Refines logs and includes metadata like Kubernetes labels.
- **Output Section**: Sends logs matching specific criteria (pods with `hpe-ezua/app=ezpresto`) to the OpenTelemetry Collector.

To apply the new configuration, update the Fluent Bit ConfigMap and restart the Fluent Bit DaemonSet:
kubectl -n monitoring rollout restart daemonset fluentbit

Configuration Adjustments

Before deploying, ensure the following adjustments are made:

- **Replace `another-otel-host`** in the configuration with your actual OpenTelemetry host.
- **Adjust the namespace** if necessary.
- **Update OpenTelemetry image version** as required.
- **Enable proper TLS** for production environments by configuring certificates and setting `insecure` to `false`.

Deployment Steps

1. **Create and apply the ConfigMap** with the OpenTelemetry Collector configuration:
   kubectl apply -f otel-collector-config.yaml
2. **Create and apply the Deployment and Service** for the OpenTelemetry Collector:
   kubectl apply -f otel-collector-deployment.yaml

3. **Update the Fluent Bit configuration** to send logs to the OpenTelemetry Collector. Then, restart Fluent Bit for the new configuration to take effect:
   kubectl -n monitoring rollout restart daemonset fluentbit

Conclusion

This setup creates a log processing pipeline where:

- Fluent Bit collects logs from Kubernetes pods.
- Logs are sent to the OpenTelemetry Collector deployed in the cluster.
- The OpenTelemetry Collector forwards these logs to another OpenTelemetry host.

This architecture allows for flexible and scalable log collection and forwarding within a Kubernetes environment, streamlining observability and log management.