**SideCar Containers**

**What is Sidecar container :**

* Sidecar Containers: Run alongside the main containers throughout the Pod’s lifecycle, often providing auxiliary functionality like logging, monitoring, or proxying.
* Sidecar Containers: Support lifecycle probes since they need to stay healthy and ready throughout the Pod’s lifecycle.
* Sidecar Containers: Continuously interact with the main containers, often providing ongoing services such as log aggregation, monitoring, or configuration management.
* Sidecar Containers: Start alongside the main containers and run concurrently.

**In real DevOps or SRE setups:**

* **Logs are required to be shipped to centralized logging like ELK Stack, Azure Monitor, or Datadog.**
* **Instead of installing a heavyweight log collector globally, a lightweight sidecar (Fluent Bit) is used per pod for high granularity and flexibility.**

**Scenario:**

**You deploy 50 microservices. Each service pod includes an NGINX + Fluent Bit sidecar. Fluent Bit sends logs to Azure Log Analytics, allowing fine-grained log filtering per service.**

**We’ll:**

1. **✅ Create a new Log Analytics Workspace**
2. **🔑 Fetch the Workspace ID and Shared Key**
3. **🔒 Create Kubernetes Secret**
4. **⚙️ Setup Fluent Bit Config**
5. **🚀 Deploy 2 Microservices with NGINX + Fluent Bit sidecars**
6. **🔍 Enable filtering per service in Azure Log Analytics**

**✅ Step 1: Create Log Analytics Workspace**

az monitor log-analytics workspace create \

--resource-group internal-training \

--workspace-name aks-westus-logs \

--location westus

**🔑 Step 2: Get Workspace ID & Shared Key**

# Get Workspace ID

az monitor log-analytics workspace show \

--resource-group internal-training \

--workspace-name aks-westus-logs \

--query customerId -o tsv

# Get Primary Shared Key

az monitor log-analytics workspace get-shared-keys \

--resource-group internal-training \

--workspace-name aks-westus-logs \

--query primarySharedKey -o tsv

**🔒 Step 3: Create Kubernetes Secret**

**Replace <workspace-id> and <shared-key> with actual output:**

kubectl create secret generic log-analytics-secret \

--namespace=default \

--from-literal=workspace-id=da7cf587-ddb8-44b2-8951-0c5831239f53 \

--from-literal=shared-key=m6sd6f1JIQhwQNCvPj9fn60AjOjSoU4C5ZPQq12FVikRMXELe3CcJY+MvKuqiSqesQndHW7AgOufTFFMEnJjSw==

**⚙️ Step 4: Create Fluent Bit ConfigMap**

**Save as fluent-bit-configmap.yaml:**

apiVersion: v1

kind: ConfigMap

metadata:

name: fluent-bit-config

namespace: default

data:

fluent-bit.conf: |

[SERVICE]

Flush 1

Daemon Off

Log\_Level info

[INPUT]

Name tail

Path /var/log/nginx/access.log

Path\_Key log\_path

Tag nginx-logs

Refresh\_Interval 5

DB /var/log/flb\_nginx.db

DB.Sync Normal

[FILTER]

Name modify

Match nginx-logs

Add app\_name ${APP\_NAME}

[OUTPUT]

Name azure

Match nginx-logs

Customer\_ID ${WORKSPACE\_ID}

Shared\_Key ${SHARED\_KEY}

Log\_Type nginxlogs

kubectl apply -f fluent-bit-configmap.yaml

**🚀 Step 5: Deploy Microservice A**

**Save as microservice-a.yaml:**

apiVersion: apps/v1

kind: Deployment

metadata:

name: microservice-a

spec:

replicas: 1

selector:

matchLabels:

app: microservice-a

template:

metadata:

labels:

app: microservice-a

spec:

volumes:

- name: varlog

emptyDir: {}

- name: fluentbit-config

configMap:

name: fluent-bit-config

containers:

- name: nginx

image: nginx

volumeMounts:

- name: varlog

mountPath: /var/log/nginx

ports:

- containerPort: 80

- name: fluent-bit

image: fluent/fluent-bit:1.9

env:

- name: APP\_NAME

value: microservice-a

- name: WORKSPACE\_ID

valueFrom:

secretKeyRef:

name: log-analytics-secret

key: workspace-id

- name: SHARED\_KEY

valueFrom:

secretKeyRef:

name: log-analytics-secret

key: shared-key

volumeMounts:

- name: varlog

mountPath: /var/log/nginx

- name: fluentbit-config

mountPath: /fluent-bit/etc/

---

apiVersion: v1

kind: Service

metadata:

name: microservice-a

spec:

selector:

app: microservice-a

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

**🚀 Step 6: Deploy Microservice B**

Same structure, just change names. Save as microservice-b.yaml:

apiVersion: apps/v1

kind: Deployment

metadata:

name: microservice-b

spec:

replicas: 1

selector:

matchLabels:

app: microservice-b

template:

metadata:

labels:

app: microservice-b

spec:

volumes:

- name: varlog

emptyDir: {}

- name: fluentbit-config

configMap:

name: fluent-bit-config

containers:

- name: nginx

image: nginx

volumeMounts:

- name: varlog

mountPath: /var/log/nginx

ports:

- containerPort: 80

- name: fluent-bit

image: fluent/fluent-bit:1.9

env:

- name: APP\_NAME

value: microservice-b

- name: WORKSPACE\_ID

valueFrom:

secretKeyRef:

name: log-analytics-secret

key: workspace-id

- name: SHARED\_KEY

valueFrom:

secretKeyRef:

name: log-analytics-secret

key: shared-key

volumeMounts:

- name: varlog

mountPath: /var/log/nginx

- name: fluentbit-config

mountPath: /fluent-bit/etc/

---

apiVersion: v1

kind: Service

metadata:

name: microservice-b

spec:

selector:

app: microservice-b

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

kubectl apply -f fluent-bit-configmap.yaml

kubectl apply -f microservice-a.yaml

kubectl apply -f microservice-b.yaml

**===Explanation===**

spec:

volumes:

- name: applog

emptyDir: {}

 volumes: Declares shared volumes for containers in this pod.

 applog: A temporary **in-memory shared volume** (emptyDir) used to exchange logs between containers.

 emptyDir is a **temporary storage volume** created **on the node** where the pod is running.

 It is **shared between all containers in the same pod**.

 It **lives as long as the pod lives** — if the pod is deleted or crashes, the data is lost

emptyDir makes the volume **accessible to both containers** without needing a persistent disk.

- name: fluentbit-config

configMap:

name: fluent-bit-config

fluentbit-config: Mounts a Kubernetes ConfigMap into the container file system — used by Fluent Bit for its configuration.

volumeMounts:

- name: applog

mountPath: /usr/app/logs

volumeMounts: Mount the shared volume applog at /usr/app/logs, where your Spring Boot app writes logs (app.log).

**🔹 Container: Fluent Bit (Sidecar)**

- name: fluent-bit

image: fluent/fluent-bit:1.9

A sidecar container to collect and forward logs.

env:

- name: APP\_NAME

value: springboot-app

Environment variables used in the Fluent Bit config to tag logs.

- name: WORKSPACE\_ID

valueFrom:

secretKeyRef:

name: log-analytics-secret

key: workspace-id

- name: SHARED\_KEY

valueFrom:

secretKeyRef:

name: log-analytics-secret

key: shared-key

These values are pulled from a Kubernetes Secret named log-analytics-secret. Used for authenticating with Azure Log Analytics.

volumeMounts:

- name: applog

mountPath: /usr/app/logs

- name: fluentbit-config

mountPath: /fluent-bit/etc/

 Mounts the same applog volume at the same location so it can read /usr/app/logs/app.log.

 Also mounts the Fluent Bit config into the correct location.

**🔍 After a Few Minutes — Query Logs**

**In Azure Portal → Your Log Analytics Workspace → Logs, run:**

nginxlogs

| where app\_name\_s == "microservice-a"

nginxlogs

| where app\_name\_s == "microservice-b"

A screenshot of a computer

AI-generated content may be incorrect.

**kubectl apply -f log-generator.yaml**

apiVersion: v1

kind: Pod

metadata:

name: log-generator

spec:

containers:

- name: curl

image: curlimages/curl:latest

command:

- sh

- -c

- |

while true; do

echo "Hitting microservice-a..."

curl -s microservice-a.default.svc.cluster.local

echo "Hitting microservice-b..."

curl -s microservice-b.default.svc.cluster.local

sleep 5

done

nginxlogs

| where app\_name\_s == "microservice-a"

| limit 50