# Kubernetes Fluent Bit → MongoDB Logging Design

## Overview

This document describes a Kubernetes logging pipeline that routes application and system logs into two MongoDB collections without using Fluentd. It leverages Fluent Bit for log collection and routing, and a lightweight Go HTTP bridge service to insert logs into MongoDB.

## MongoDB Collections

Database: logging

Collections:

• logs – contains all structured logs from CCA-API pods/containers (stdout/stderr)

• system-logs – contains logs from Kong, Kubernetes, and other system activity

## Structured Log Model (Go)

type Log struct {  
 Type string `json:"type"`  
 Timestamp time.Time `json:"timestamp"`  
 Level LogLevel `json:"level"`  
 Service string `json:"service"`  
 TraceID string `json:"trace\_id,omitempty"`  
 SpanID string `json:"span\_id,omitempty"`  
 Message string `json:"message"`  
 ErrorID \*int `json:"error\_id,omitempty"`  
 ErrorDetail \*ErrorDetail `json:"error\_detail,omitempty"`  
 Context map[string]interface{} `json:"context,omitempty"`  
 Duration \*time.Duration `json:"duration,omitempty"`  
}

## Mongo Bridge Service

The Go bridge service accepts JSON logs via HTTP POST and writes them into MongoDB. It supports db/collection overrides via query parameters (?db=logging&collection=logs) to allow routing to multiple collections.

## Fluent Bit Configuration

[SERVICE]  
 storage.path /var/fluent-bit/state  
 storage.sync normal  
 storage.backlog.mem\_limit 64M  
  
[INPUT]  
 Name tail  
 Path /var/log/containers/\*.log  
 Parser docker  
 storage.type filesystem  
  
[FILTER]  
 Name kubernetes  
 Match kube.\*  
 Merge\_Log On  
  
[FILTER]  
 Name rewrite\_tag  
 Match kube.\*  
 Rule $kubernetes['namespace\_name'] ^cca-api$ app.logs false  
  
[FILTER]  
 Name rewrite\_tag  
 Match kube.\*  
 Rule $kubernetes['namespace\_name'] ^(?!cca-api$).\* system.logs false  
  
[OUTPUT]  
 Name http  
 Match app.logs  
 Host mongo-bridge.logging.svc.cluster.local  
 Port 443  
 URI /ingest?db=logging&collection=logs  
 Format json  
 tls On  
 Header Content-Type application/json  
 Header X-API-Key YOUR\_SECRET  
  
[OUTPUT]  
 Name http  
 Match system.logs  
 Host mongo-bridge.logging.svc.cluster.local  
 Port 443  
 URI /ingest?db=logging&collection=system-logs  
 Format json  
 tls On  
 Header Content-Type application/json  
 Header X-API-Key YOUR\_SECRET

## Kubernetes Deployment (Bridge)

apiVersion: v1  
kind: Secret  
metadata: { name: mongo-bridge, namespace: logging }  
stringData:  
 MONGODB\_URI: mongodb+srv://USER:PASS@cluster.example.mongodb.net/?retryWrites=true&w=majority  
 API\_KEY: YOUR\_SECRET  
---  
apiVersion: apps/v1  
kind: Deployment  
metadata: { name: mongo-bridge, namespace: logging }  
spec:  
 replicas: 2  
 selector: { matchLabels: { app: mongo-bridge } }  
 template:  
 metadata: { labels: { app: mongo-bridge } }  
 spec:  
 containers:  
 - name: bridge  
 image: yourrepo/mongo-bridge:latest  
 ports: [{ containerPort: 8080 }]  
 env:  
 - name: MONGODB\_URI  
 valueFrom: { secretKeyRef: { name: mongo-bridge, key: MONGODB\_URI } }  
 - name: API\_KEY  
 valueFrom: { secretKeyRef: { name: mongo-bridge, key: API\_KEY } }  
 - name: MONGODB\_DB  
 value: logging  
 - name: MONGODB\_COLLECTION  
 value: logs  
 readinessProbe: { httpGet: { path: /healthz, port: 8080 } }  
 livenessProbe: { httpGet: { path: /healthz, port: 8080 } }  
---  
apiVersion: v1  
kind: Service  
metadata: { name: mongo-bridge, namespace: logging }  
spec:  
 type: ClusterIP  
 selector: { app: mongo-bridge }  
 ports:  
 - name: https  
 port: 443  
 targetPort: 8080

## MongoDB Index Recommendations

db.logs.createIndex({ timestamp: -1 })  
db.logs.createIndex({ service: 1, timestamp: -1 })  
db.logs.createIndex({ level: 1, timestamp: -1 })  
db.logs.createIndex({ trace\_id: 1 })  
db["system-logs"].createIndex({ timestamp: -1 })  
db["system-logs"].createIndex({ "kubernetes.namespace\_name": 1, timestamp: -1 })