Docker Compose install kafka, elk and grafana. After that, use grafana to add datasource elasticsearch. Use filebeat send logs to kafka, form kafka to ELK, logstash to filter log, visualize by kibana, finally show on grafana

Docker Compose Setup for Kafka, ELK, and Grafana with Log Pipeline

Here's a comprehensive Docker Compose setup that installs Kafka, ELK (Elasticsearch, Logstash, Kibana), and Grafana, with Filebeat sending logs to Kafka, then to ELK, and finally visualized in Grafana.

docker-compose.yml

version: '3.8'

services:

  # Zookeeper (required for Kafka)

  zookeeper:

    image: confluentinc/cp-zookeeper:7.9.0

    container\_name: zookeeper

    ports:

      - "2181:2181"

    environment:

      ZOOKEEPER\_CLIENT\_PORT: 2181

      ZOOKEEPER\_TICK\_TIME: 2000

    networks:

      - kafka

  # Kafka

  kafka:

    image: confluentinc/cp-kafka:7.9.0

    container\_name: kafka

    depends\_on:

      - zookeeper

    ports:

      - "9092:9092"

      - "29092:29092"

    environment:

      KAFKA\_BROKER\_ID: 1

      KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

      KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT\_HOST:PLAINTEXT

      KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT\_HOST://localhost:29092

      KAFKA\_AUTO\_CREATE\_TOPICS\_ENABLE: "true"  # Enable auto topic creation

      KAFKA\_CREATE\_TOPICS: "beat\_topic\_test:1:1"  # Topic:partitions:replicas

      KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

      KAFKA\_TRANSACTION\_STATE\_LOG\_MIN\_ISR: 1

      KAFKA\_TRANSACTION\_STATE\_LOG\_REPLICATION\_FACTOR: 1

      KAFKA\_GROUP\_INITIAL\_REBALANCE\_DELAY\_MS: 0

    networks:

      - kafka

  # Kafka UI

  kafdrop:

    image: obsidiandynamics/kafdrop:4.1.0

    ports:

      - "9000:9000"

    environment:

      KAFKA\_BROKERCONNECT: "kafka:9092"

    depends\_on:

      - kafka

    networks:

      - kafka

  # Elasticsearch

  elasticsearch:

    image: docker.elastic.co/elasticsearch/elasticsearch:8.13.4

    container\_name: elasticsearch

    environment:

      - discovery.type=single-node

      - xpack.security.enabled=false

      - "ES\_JAVA\_OPTS=-Xms512m -Xmx512m"

    ulimits:

      memlock:

        soft: -1

        hard: -1

    volumes:

      - es\_data:/usr/share/elasticsearch/data

    ports:

      - "9200:9200"

      - "9300:9300"

    networks:

      - elk

  # Logstash

  logstash:

    image: docker.elastic.co/logstash/logstash:8.13.4

    container\_name: logstash

    volumes:

      - ./logstash/config:/usr/share/logstash/config

      - ./logstash/pipeline:/usr/share/logstash/pipeline:ro

    ports:

      - "5044:5044"

      - "9600:9600"

    environment:

      - LS\_JAVA\_OPTS=-Xms256m -Xmx256m

    depends\_on:

      - elasticsearch

      - kafka

    networks:

      - elk

      - kafka

  # Kibana

  kibana:

    image: docker.elastic.co/kibana/kibana:8.13.4

    container\_name: kibana

    environment:

      - ELASTICSEARCH\_HOSTS=http://elasticsearch:9200

    ports:

      - "5601:5601"

    depends\_on:

      - elasticsearch

    networks:

      - elk

  # Grafana

  grafana:

    image: grafana/grafana:latest

    container\_name: grafana

    ports:

      - "3000:3000"

    environment:

      - GF\_SECURITY\_ADMIN\_PASSWORD=Ocp@2026

    volumes:

      - grafana\_data:/var/lib/grafana

    depends\_on:

      - elasticsearch

    networks:

      - elk

volumes:

  es\_data:

    driver: local

  grafana\_data:

    driver: local

networks:

  elk:

    driver: bridge

  kafka:

    driver: bridge

Filebeat Configuration

Create filebeat/filebeat.yml:

filebeat.inputs:

- type: log

  enabled: true

  paths:

    - /logs/\*.log

output.kafka:

  hosts: ["kafka:9092"]

  topic: "beat\_topic\_test"

  codec.json:

    pretty: false

Logstash Configuration

Create logstash/config/logstash.yml:

http.host: "0.0.0.0"

path.config: /usr/share/logstash/pipeline

Create logstash/pipeline/logstash.conf:

input {

  kafka {

    bootstrap\_servers => "kafka:9092"

    topics => ["beat\_topic\_test"]

    codec => json

  }

}

filter {

  grok {

    match => { "message" => "%{COMBINEDAPACHELOG}" }

  }

  date {

    match => [ "timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ]

  }

}

output {

  elasticsearch {

    hosts => ["http://elasticsearch:9200"]

    index => "beat\_logs-%{+YYYY.MM.dd}"

  }

  stdout { codec => rubydebug }

}

Setup Instructions

1. Create the directories and files as shown above
2. Run the stack: docker-compose up -d
3. Wait for all services to start (this may take a few minutes)

Create filebeat containers:

docker run -d --name filebeat -v .\filebeat\filebeat.yml:/usr/share/filebeat/filebeat.yml -v .\logs:/logs --network=elk-docker\_kafka docker.elastic.co/beats/filebeat:8.13.4 filebeat -e -strict.perms=false

Configure Grafana to Use Elasticsearch as Data Source

1. Access Grafana at [http://localhost:3000](http://localhost:3000/)
   * Default credentials: admin/admin (you'll be prompted to change the password)
2. Add Elasticsearch as a data source:
   * Go to Configuration > Data Sources
   * Click "Add data source"
   * Select "Elasticsearch"
   * Configure with:
     + URL: [http://elasticsearch:9200](http://elasticsearch:9200/)
     + Index name: beat\_logs-\*
     + Time field name: @timestamp
     + Version: 8.0+
   * Click "Save & Test"

Verify the Pipeline

1. Check logs are being collected:
   * Access Kibana at [http://localhost:5601](http://localhost:5601/)
   * Go to "Discover" and create an index pattern for " beat\_logs-\*"
   * You should see logs appearing
2. In Grafana:
   * Create a new dashboard
   * Add a panel and select your Elasticsearch data source
   * Create visualizations based on your log data

Additional Notes

1. The Filebeat configuration collects Docker container logs. You can modify this to collect other logs.
2. The Logstash filter is basic - you'll want to customize it for your specific log format.
3. For production use, you should enable security features in Elasticsearch and Kafka.
4. You may need to adjust resource limits (memory/CPU) based on your system capabilities.

This setup provides a complete pipeline from log generation (via Docker containers) to visualization in Grafana, passing through Kafka for buffering and Logstash for processing.