Task Project Report: Building and Deploying a Web Application with Docker, Load Balancing, and ELK Stack

1. Introduction

This document outlines the steps taken to complete the assignment on building and deploying a web application using Docker, setting up a load balancer, and configuring the ELK stack for log aggregation. The primary objective was to demonstrate containerization, load balancing, and log management.

2. Prerequisites

Before starting the assignment, the following tools and environments were set up:

- Docker and Docker Compose installed on the host machine.

  TechStack : Docker, Java, NGINX, Apache Tomcat, Filebeat, Elasticsearch, Logstash, and Kibana (ELK stack).

3. Docker Containerization

3.1. Dockerfile for Spring Boot Application

The Spring Boot application was containerized using a multi-stage Dockerfile. The first stage builds the WAR file, and the second stage deploys it on an Apache Tomcat server.

Dockerfile:

spring-boot-docker-starter > Dockerfile

```
Stage 1: Build WAR
FROM openjdk:17-jdk-slim AS build-stage
WORKDIR /app
COPY gradlew gradlew
COPY gradle gradle
COPY build.gradle settings.gradle ./
COPY src src
RUN chmod +x gradlew
RUN ["./gradlew", "clean", "war", "--no-daemon"]


# Stage 2: Deploy to Tomcat
FROM tomcat:10-jdk17-openjdk-slim
RUN rm -rf /usr/local/tomcat/webapps/*
COPY --from=build-stage /app/build/libs/demo-0.0.1-SNAPSHOT-plain.war /usr/local/tomcat/webapps/ROOT.war
# COPY clusustask.war /usr/local/tomcat/webapps/ROOT.war
EXPOSE 8080
CMD ["catalina.sh", "run"]
```

3.2. Docker Compose Configuration

A Docker Compose file was created to orchestrate the deployment of multiple services, including the Spring Boot application, NGINX proxy, MySQL database, and the ELK stack (Filebeat, Elasticsearch, Logstash, Kibana).

Docker Compose File:

Note: Check the docker compose file from root directory.

3.3. Running the Containers

To build and run the containers, the following commands were executed:

```
docker compose --env-file .env build app-api  # Build the app's Docker image
docker compose --env-file .env up -d          # Start all services
```

4. Deploying the Sample Web Application

4.1. Spring Boot Application Deployment

The Spring Boot application was deployed on the Tomcat server as a WAR file. The application includes several API endpoints that generate logs on the Tomcat server's. 3 tomcat app api containers has been depoloyed & running.

4.1. Verifying Deployment

The application was accessed via the configured NGINX proxy using the specified virtual host. API endpoints were tested to ensure they were functioning correctly.

```
CONTAINER ID   IMAGE                                                  COMMAND                  CREATED          STATUS          PORTS
                        NAMES
479178d0de3d   docker.elastic.co/beats/filebeat:7.9.1                 "/usr/local/bin/dock…"   10 minutes ago   Up 10 minutes
                        filebeat
2f918ca05386   nginxproxy/acme-companion                              "/bin/bash /app/entr…"   10 minutes ago   Up 10 minutes
                        nginx-proxy-acme
83af7d518f71   clusus-task-app-api-2                                  "catalina.sh run"        10 minutes ago   Up 10 minutes   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp
                        app-api-2
19afe9d43310   docker.elastic.co/logstash/logstash:7.9.1              "/usr/local/bin/dock…"   10 minutes ago   Up 10 minutes   0.0.0.0:5044->5044/tcp, :::5044->5044/tcp, 0.0.0.0:9600->9
600/tcp, :::9600->9600/tcp   logstash
f6c8412f6669   docker.elastic.co/kibana/kibana:7.9.1                  "/usr/local/bin/dumb…"   10 minutes ago   Up 10 minutes   0.0.0.0:5601->5601/tcp, :::5601->5601/tcp
                        kibana
e597f2cf93b5   clusus-task-app-api-3                                  "catalina.sh run"        10 minutes ago   Up 10 minutes   0.0.0.0:8090->8080/tcp, :::8090->8080/tcp
                        app-api-3
6734388c01ce   clusus-task-app-api-1                                  "catalina.sh run"        10 minutes ago   Up 10 minutes   0.0.0.0:8070->8080/tcp, :::8070->8080/tcp
                        app-api-1
70849fb567cd   mysql:5.7                                              "docker-entrypoint.s…"   10 minutes ago   Up 10 minutes   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp
                        mysql-db
1fbf87a7834c   nginxproxy/nginx-proxy                                 "/app/docker-entrypo…"   10 minutes ago   Up 10 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :
::443->443/tcp          nginx-proxy
3cc03ca5e0df   docker.elastic.co/elasticsearch/elasticsearch:7.9.1    "/tini -- /usr/local…"   10 minutes ago   Up 10 minutes   0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 0.0.0.0:9300->9
300/tcp, :::9300->9300/tcp   elasticsearch
```

NGINX Proxy Configuration & Loadbalancing

The NGINX proxy was configured to route traffic to the Spring Boot application. The proxy handles SSL termination and directs requests to the appropriate service based on the virtual host.

To route the traffic to app-api or any service just we need to put environment varaibles in respective services in docker compose.

For SSL certificates & renewal we have used letsencrypt which will be managed by service acme

Above snapshoot shows the Nginx proxy container loadbalancing upstream for 3 app api containers.

These upstream are automatically created nginx-proxy container if we just mention environment

```
environment:
  - LETSENCRYPT_HOST=${APP_LETSENCRYPT_HOST}
  - VIRTUAL_HOST=${APP_VIRTUAL_HOST}
  - VIRTUAL_PORT=${APP_VIRTUAL_PORT}
```

```
# app-clusustask.cloudyninjas.com/
upstream app-clusustask.cloudyninjas.com {
    # Container: app-api-2
    #     networks:
    #         clusus-task_clusus-task-nw (reachable)
    #     IP address: 172.18.0.7
    #     exposed ports: 8080/tcp
    #     default port: 8080
    #     using port: 8080
    #         /!\ WARNING: Virtual port published on host.  Clients
    #                      might be able to bypass nginx-proxy and
    #                      access the container's server directly.
    server 172.18.0.7:8080;
    # Container: app-api-3
    #     networks:
    #         clusus-task_clusus-task-nw (reachable)
    #     IP address: 172.18.0.5
    #     exposed ports: 8080/tcp
    #     default port: 8080
    #     using port: 8080
    #         /!\ WARNING: Virtual port published on host.  Clients
    #                      might be able to bypass nginx-proxy and
    #                      access the container's server directly.
    server 172.18.0.5:8080;
    # Container: app-api-1
    #     networks:
    #         clusus-task_clusus-task-nw (reachable)
    #     IP address: 172.18.0.6
    #     exposed ports: 8080/tcp
    #     default port: 8080
    #     using port: 8080
    #         /!\ WARNING: Virtual port published on host.  Clients
    #                      might be able to bypass nginx-proxy and
    #                      access the container's server directly.
    server 172.18.0.6:8080;
}
```

```
}
server {
    server_name app-clusustask.cloudyninjas.com;
    access_log /var/log/nginx/access.log vhost;
    http2 on;
    listen 443 ssl ;
    ssl_session_timeout 5m;
    ssl_session_cache shared:SSL:50m;
    ssl_session_tickets off;
    ssl_certificate /etc/nginx/certs/app-clusustask.cloudyninjas.com.crt;
    ssl_certificate_key /etc/nginx/certs/app-clusustask.cloudyninjas.com.key;
    ssl_dhparam /etc/nginx/certs/app-clusustask.cloudyninjas.com.dhparam.pem;
    ssl_stapling on;
    ssl_stapling_verify on;
    ssl_trusted_certificate /etc/nginx/certs/app-clusustask.cloudyninjas.com.chain.pem;
    set $sts_header "";
    if ($https) {
        set $sts_header "max-age=31536000";
    }
    add_header Strict-Transport-Security $sts_header always;
    location / {
        proxy_pass http://app-clusustask.cloudyninjas.com;
        set $upstream_keepalive false;
    }
}
```

The application was accessed via its API endpoints, and various requests were made to generate logs on the Tomcat servers.

Verifying loadbalancing works or not

Curl or browse more than 3 times so nginx proxy will loadbalance in roundrobin

https://app-clusustask.cloudyninjas.com/api/v1/hello

or

https://app-clusustask.cloudyninjas.com/api/v1/96 (any integer)

Check logs

docker logs -f nginx-proxy

```
nginx.1    | app-clusustask.cloudyninjas.com 27.34.68.136 - - [09/Aug/2024:20:24:57 +0000] "GET /api/v1/hello HTTP/2.0" 200 31 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36" "172.18.0.7:8080"
nginx.1    | app-clusustask.cloudyninjas.com 27.34.68.136 - - [09/Aug/2024:20:24:58 +0000] "GET /api/v1/hello HTTP/2.0" 200 31 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36" "172.18.0.5:8080"
nginx.1    | app-clusustask.cloudyninjas.com 27.34.68.136 - - [09/Aug/2024:20:24:59 +0000] "GET /api/v1/hello HTTP/2.0" 200 31 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36" "172.18.0.6:8080"
nginx.1    | app-clusustask.cloudyninjas.com 27.34.68.136 - - [09/Aug/2024:20:25:00 +0000] "GET /api/v1/hello HTTP/2.0" 200 31 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36" "172.18.0.7:8080"
nginx.1    | app-clusustask.cloudyninjas.com 27.34.68.136 - - [09/Aug/2024:20:25:01 +0000] "GET /api/v1/hello HTTP/2.0" 200 31 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36" "172.18.0.5:8080"
nginx.1    | app-clusustask.cloudyninjas.com 27.34.68.136 - - [09/Aug/2024:20:25:01 +0000] "GET /api/v1/hello HTTP/2.0" 200 31 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36" "172.18.0.6:8080"
nginx.1    | app-clusustask.cloudyninjas.com 27.34.68.136 - - [09/Aug/2024:20:25:02 +0000] "GET /api/v1/hello HTTP/2.0" 200 31 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36" "172.18.0.7:8080"
```

From above snapshoot shows nginx proxy is loadbalancing the 3 app api containers, we can verify the IP address of containers from logs and Upstream IP which are 172.18.0.5, 172.18.0.6 & 172.18.0.7

Here is kibana logging URL

https://logging-clusustask.cloudyninjas.com/

Explanation of Filebeat Configuration

Filebeat Configuration Overview:

The Filebeat configuration is responsible for collecting and forwarding logs from various services running in Docker containers to Logstash, which then processes these logs and sends them to Elasticsearch for indexing and searching.

Check the filebeat configuration & logstash from respective directory.

We have two requirements:

1.Ship logs from specific containers
2.Handle multiline logs

all these requirement are fullfiled with filebeat.

Key Sections of the Filebeat Configuration:

1. **`filebeat.autodiscover`:**
   - **Autodiscover Providers:** Filebeat can dynamically discover Docker containers and automatically configure itself to collect logs based on the container metadata (such as labels).

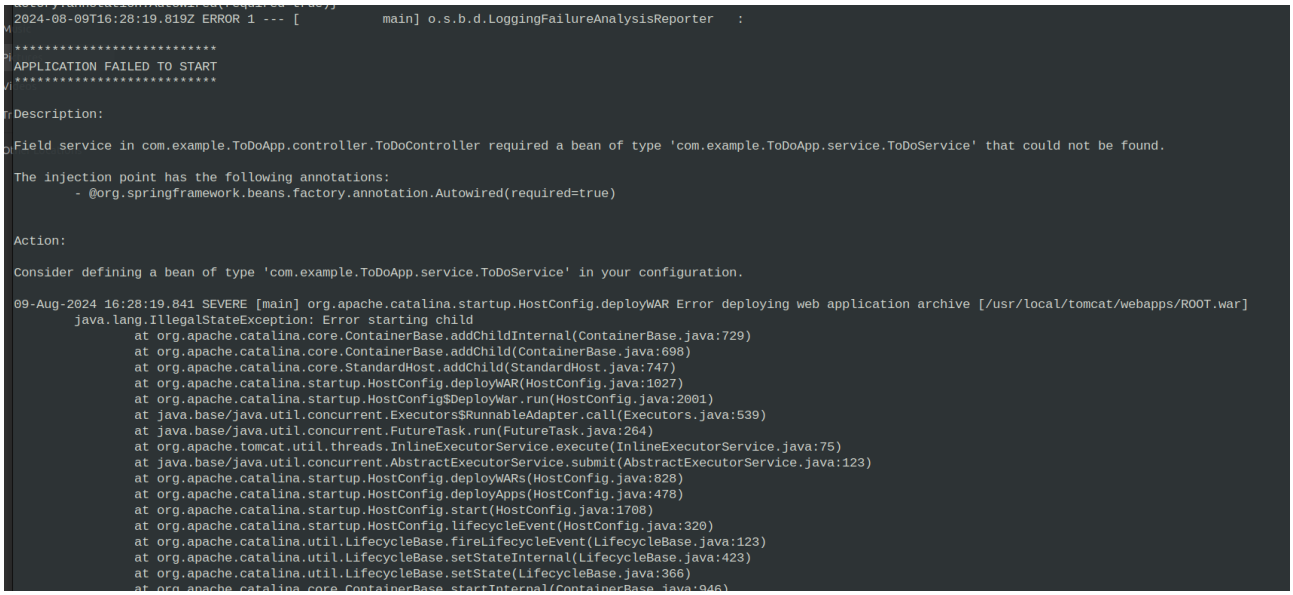
Docker container must be labels to use in Autodiscover , so based on that label & condition only those container logs will be Ship to logstash.

labels:

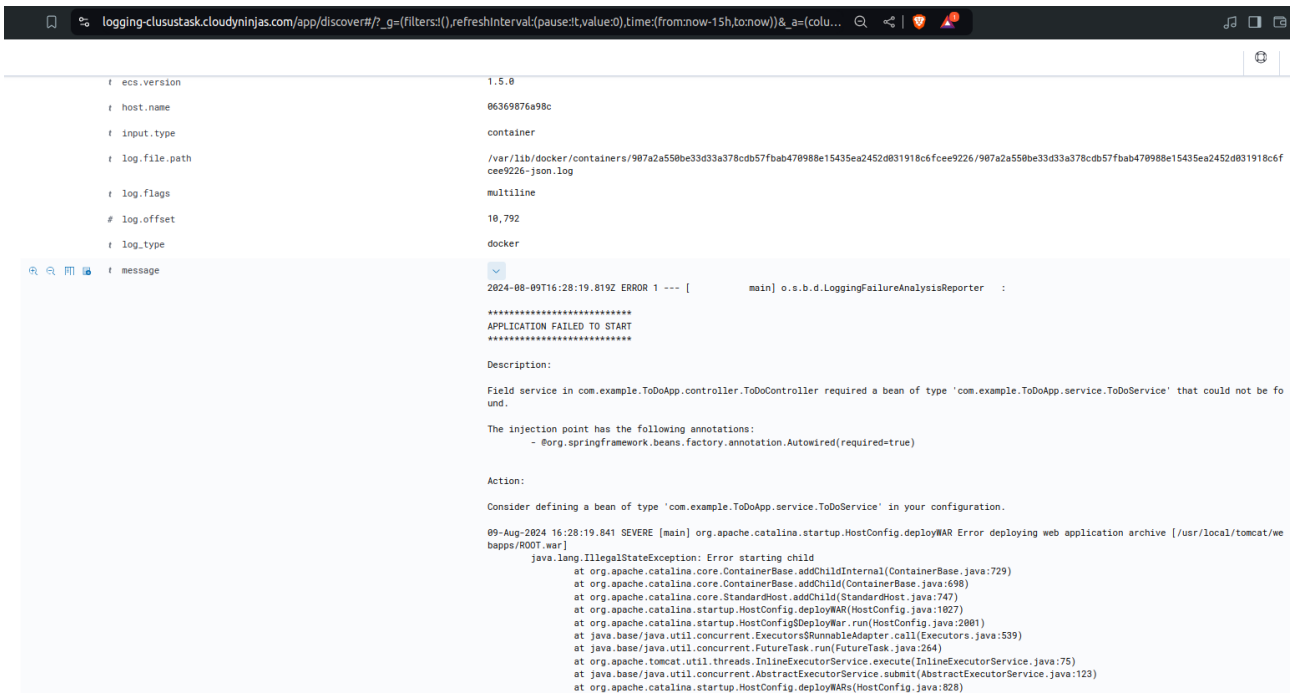
tomcat_service: "app-api"

   - **Docker Provider:**
     - **Condition and Templates:** The configuration specifies conditions based on Docker labels to identify the relevant containers (e.g., `tomcat_service` or `nginx_service`). If a container matches these conditions, Filebeat will collect logs from that container.
     - **Multiline Parsing:** For `app-api`, which is a Tomcat-based service, multiline logs (e.g., stack traces) are combined into a single event to ensure that related log lines are processed together.

Here we can see multiline logs generated by tomcat application server



lets's see how filebeat multiline config treated as single line as shown in kibana console.



- **Fields and Labels:** Adds metadata such as `log_type` to the collected logs to facilitate easier identification and filtering in later stages (Logstash).

2. **`output.logstash`:**
    - **Logstash Output:** Specifies that Filebeat should forward the collected logs to Logstash running at `logstash:5044`. Logstash then processes and enriches these logs before sending them to Elasticsearch.

Explanation of Logstash Configuration

Logstash Configuration Overview:

Logstash acts as a data processor, receiving logs from Filebeat, applying filters and transformations, and then forwarding the processed logs to Elasticsearch.

Key Sections of the Logstash Configuration:

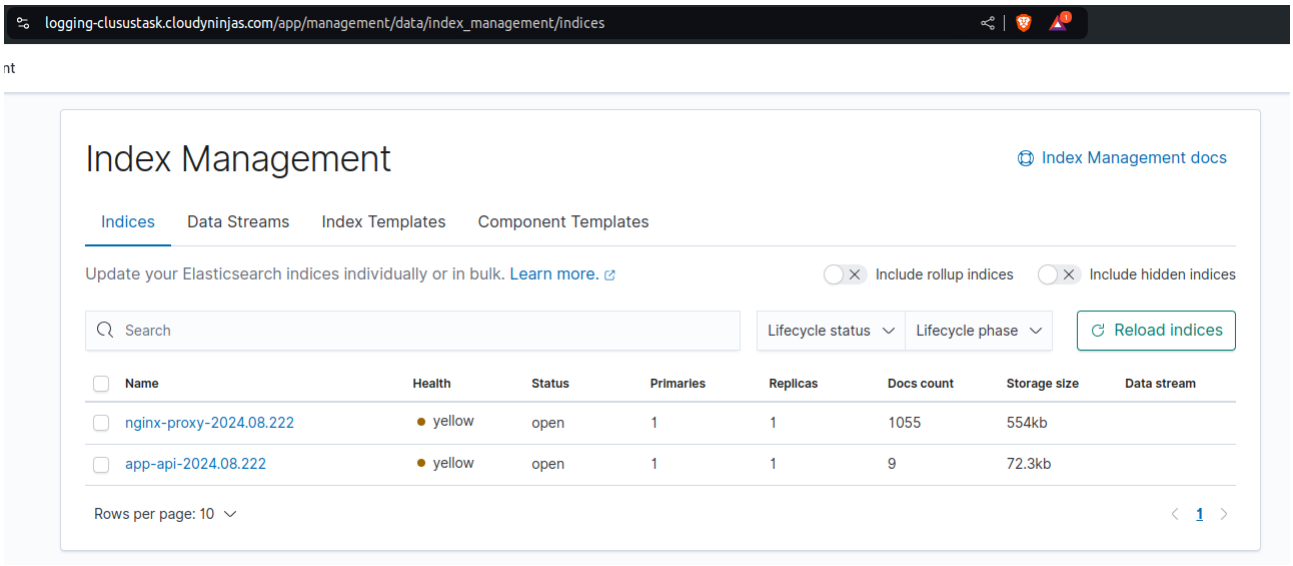
1. **`input { beats { port => 5044 } }`:**
    - **Beats Input Plugin:** Listens on port 5044 for logs sent from Filebeat. This is the entry point for logs into the Logstash pipeline.
2. **`filter { ... }`:**
    - **Filtering Logic:** The filter section contains specific parsing and enrichment logic for different types of logs based on the container name.
        - **Tomcat Logs (`app-api`):**
            - **Grok Parsing:** Extracts and structures data from Tomcat logs using the Grok pattern. It captures the timestamp, log level, message text, and client IP.
            - **Date Parsing:** Converts the extracted timestamp into the `@timestamp` field, which is used by Elasticsearch for time-based queries.
        - **NGINX Logs:**
            - **Access Logs:** Parses access logs using Grok, extracts relevant information such as IP address, HTTP method, URL, response code, and user agent. The `geoip` processor enriches the logs with geographic information based on the IP address.

- **Error Logs:** Parses error logs, extracts key fields, and tags the logs for easier identification in Elasticsearch.

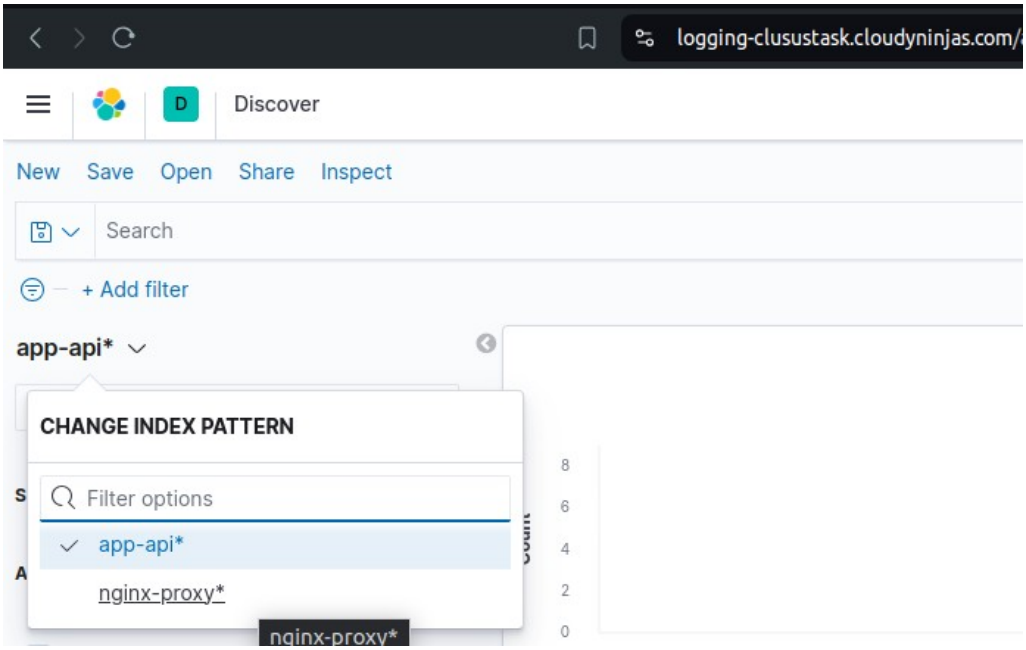
3. **`output { ... }`:**

   - **Elasticsearch Output:** Logs are sent to Elasticsearch with a specific index pattern based on the container name. For example, logs from `app-api` are indexed under `app-api-%{+YYYY.MM.DD}` and NGINX logs under `nginx-proxy-%{+YYYY.MM.DD}`..

So we will have two Index in elasticsearch as above



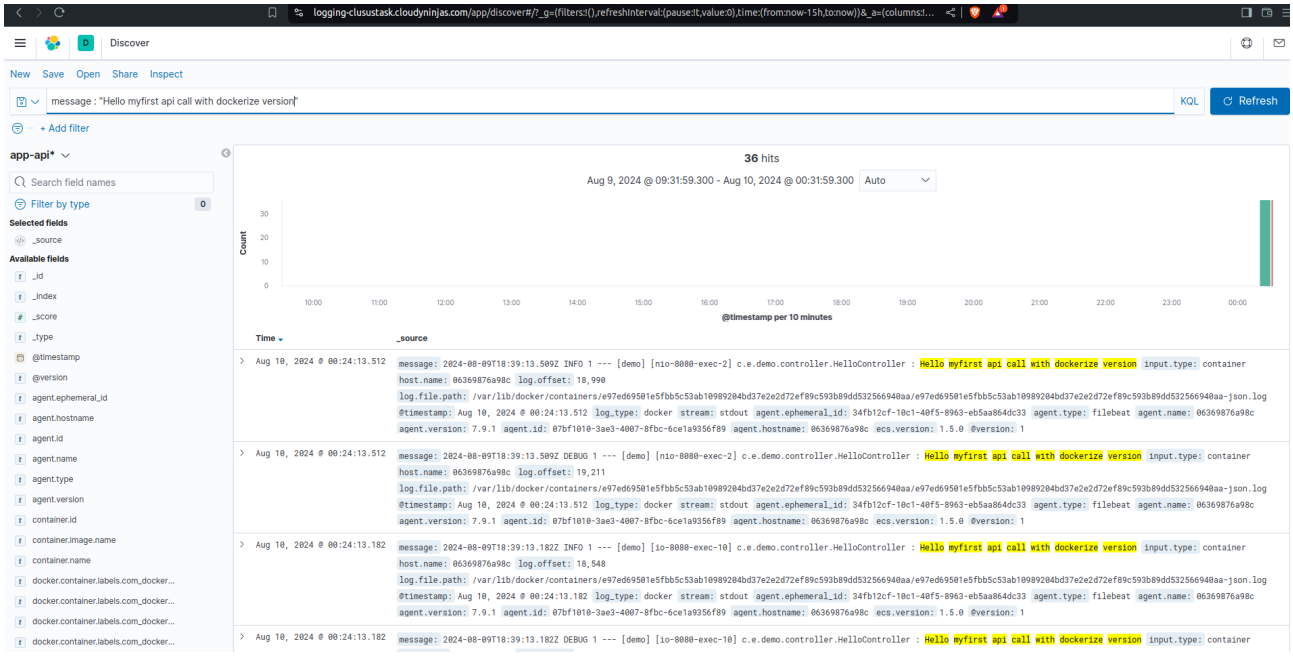
To discover the index we need to create kibana index pattern from kibana console

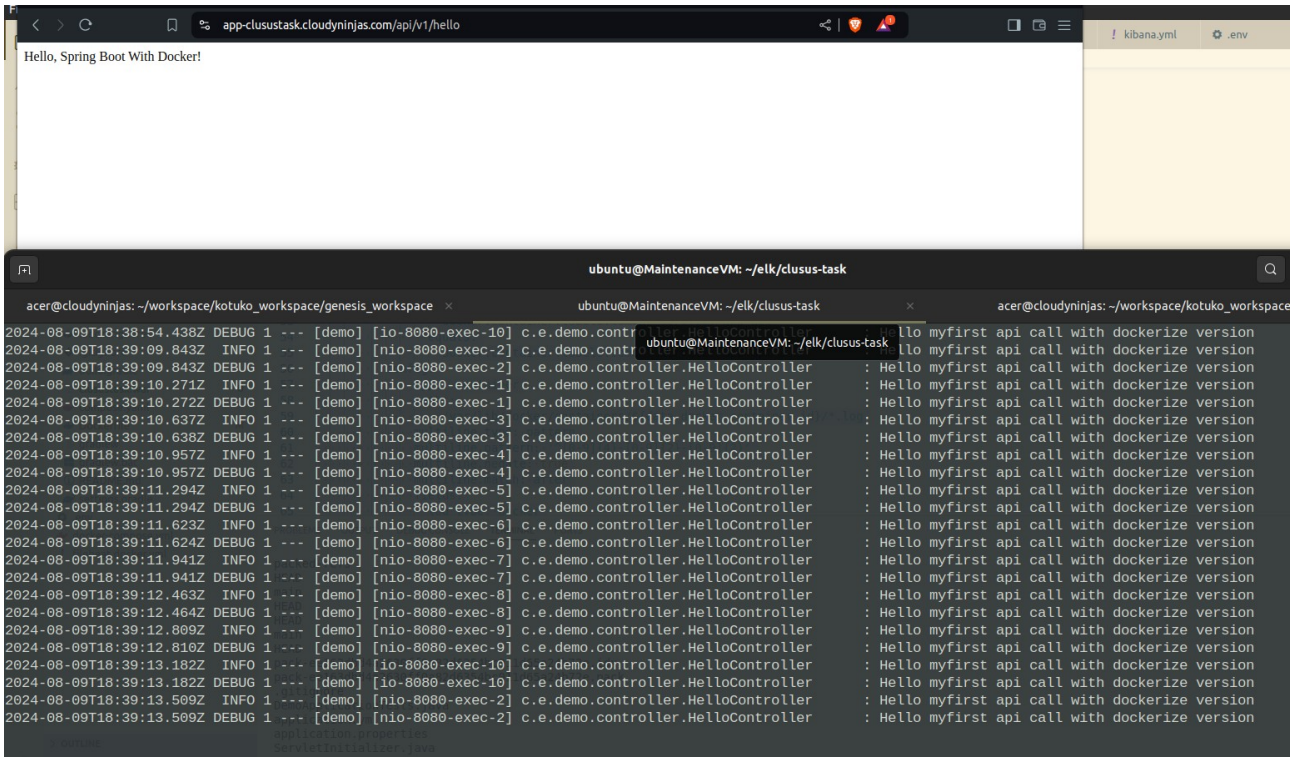


After creating index pattern we will see now in discover section, from there we can see the logs from specific pattern created.

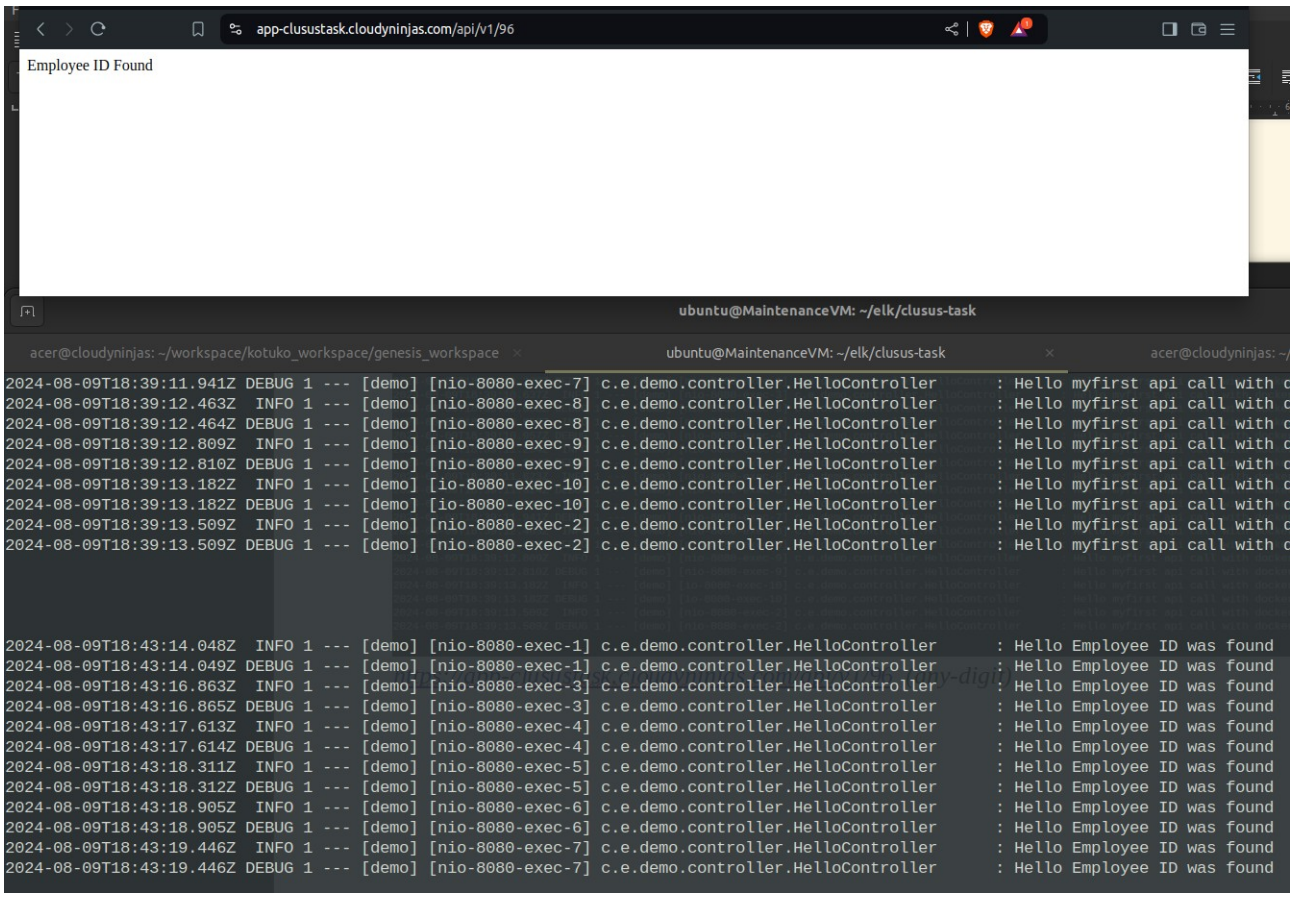
Hit the app api endpoint url in browswer/curl multiple times so we can visualize in kibana console. Here are two api endpoint for generating logs and their respective snapshoots.

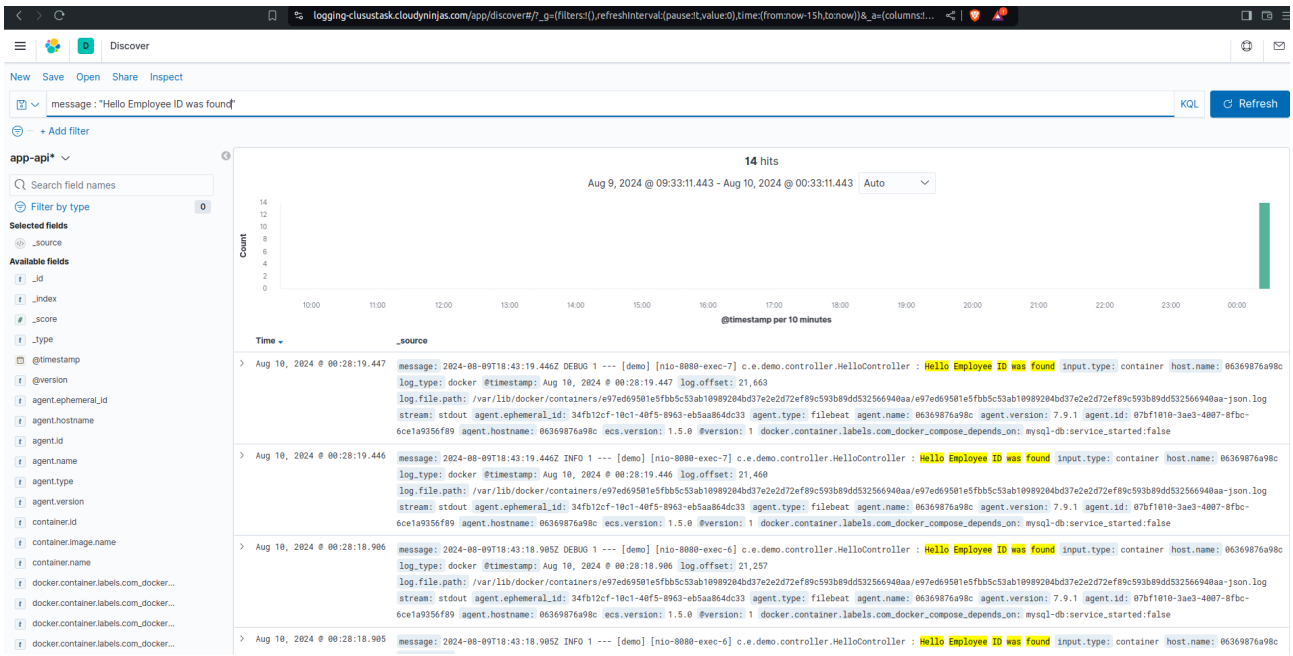
https://app-clususta sk.cloudyninjas.com/api/v1/hello





https://app-clususta sk.cloudyninjas.com/api/v1/96 *(any-digit)*


app-clususstask.cloudyninjas.com/api/v1/96
Employee ID Found
ubuntu@MaintenanceVM: ~/elk/clusus-task
acer@cloudyninjas: ~/workspace/kotuko_workspace/genesis_workspace
ubuntu@MaintenanceVM: ~/elk/clusus-task
2024-08-09T18:39:11.941Z DEBUG 1 --- [demo] [nio-8080-exec-7] c.e.demo.controller.HelloController : Hello myfirst api call with
2024-08-09T18:39:12.463Z INFO 1 --- [demo] [nio-8080-exec-8] c.e.demo.controller.HelloController : Hello myfirst api call with
2024-08-09T18:39:12.464Z DEBUG 1 --- [demo] [nio-8080-exec-8] c.e.demo.controller.HelloController : Hello myfirst api call with
2024-08-09T18:39:12.809Z INFO 1 --- [demo] [nio-8080-exec-9] c.e.demo.controller.HelloController : Hello myfirst api call with
2024-08-09T18:39:12.810Z DEBUG 1 --- [demo] [nio-8080-exec-9] c.e.demo.controller.HelloController : Hello myfirst api call with
2024-08-09T18:39:13.182Z INFO 1 --- [demo] [io-8080-exec-10] c.e.demo.controller.HelloController : Hello myfirst api call with
2024-08-09T18:39:13.182Z DEBUG 1 --- [demo] [io-8080-exec-10] c.e.demo.controller.HelloController : Hello myfirst api call with
2024-08-09T18:39:13.509Z INFO 1 --- [demo] [nio-8080-exec-2] c.e.demo.controller.HelloController : Hello myfirst api call with
2024-08-09T18:39:13.509Z DEBUG 1 --- [demo] [nio-8080-exec-2] c.e.demo.controller.HelloController : Hello myfirst api call with
2024-08-09T18:43:14.048Z INFO 1 --- [demo] [nio-8080-exec-1] c.e.demo.controller.HelloController : Hello Employee ID was found
2024-08-09T18:43:14.049Z DEBUG 1 --- [demo] [nio-8080-exec-1] c.e.demo.controller.HelloController : Hello Employee ID was found
2024-08-09T18:43:16.863Z INFO 1 --- [demo] [nio-8080-exec-3] c.e.demo.controller.HelloController : Hello Employee ID was found
2024-08-09T18:43:16.865Z DEBUG 1 --- [demo] [nio-8080-exec-3] c.e.demo.controller.HelloController : Hello Employee ID was found
2024-08-09T18:43:17.613Z INFO 1 --- [demo] [nio-8080-exec-4] c.e.demo.controller.HelloController : Hello Employee ID was found
2024-08-09T18:43:17.614Z DEBUG 1 --- [demo] [nio-8080-exec-4] c.e.demo.controller.HelloController : Hello Employee ID was found
2024-08-09T18:43:18.311Z INFO 1 --- [demo] [nio-8080-exec-5] c.e.demo.controller.HelloController : Hello Employee ID was found
2024-08-09T18:43:18.312Z DEBUG 1 --- [demo] [nio-8080-exec-5] c.e.demo.controller.HelloController : Hello Employee ID was found
2024-08-09T18:43:18.905Z INFO 1 --- [demo] [nio-8080-exec-6] c.e.demo.controller.HelloController : Hello Employee ID was found
2024-08-09T18:43:18.905Z DEBUG 1 --- [demo] [nio-8080-exec-6] c.e.demo.controller.HelloController : Hello Employee ID was found
2024-08-09T18:43:19.446Z INFO 1 --- [demo] [nio-8080-exec-7] c.e.demo.controller.HelloController : Hello Employee ID was found
2024-08-09T18:43:19.446Z DEBUG 1 --- [demo] [nio-8080-exec-7] c.e.demo.controller.HelloController : Hello Employee ID was found


logging-clususstask.cloudyninjas.com/app/discover#/?_g=(filters:!(),refreshInterval:(pause:!t,value:0),time:(from:now-15h,to:now))&_a=(columns:!...
Discover
New Save Open Share Inspect
message : "Hello Employee ID was found"
KQL
Refresh
+ Add filter
app-api*
Search field names
Filter by type
Selected fields
_source
Available fields
_id
_index
_score
_type
@timestamp
@version
agent.ephemeral_id
agent.hostname
agent.id
agent.name
agent.type
agent.version
container.id
container.image.name
container.name
docker.container.labels.com_docker...
docker.container.labels.com_docker...
docker.container.labels.com_docker...
docker.container.labels.com_docker...
14 hits
Aug 9, 2024 @ 09:33:11.443 - Aug 10, 2024 @ 00:33:11.443
Auto
Count
@timestamp per 10 minutes
Time
_source
Aug 10, 2024 @ 00:28:19.447
Aug 10, 2024 @ 00:28:19.446
Aug 10, 2024 @ 00:28:18.906
Aug 10, 2024 @ 00:28:18.905