C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmp**Devops-IA-1**

**Devops Tool: Fluentd**

**Name: Vihan Ajay Kumbhare**

**Roll No.: 16010122269**

**Class: COMPS-C2**

**What is Fluentd?**

* Fluentd is an open-source data collector that helps you collect, transform, and ship logs/data from different sources to different destinations.
* It works as a unified logging layer, meaning it can take logs from apps, servers, containers, etc., and then forward them to places like Elasticsearch, Kafka, S3, Datadog, Splunk, CloudWatch, or just stdout.

Think of it like a central hub for logs.

**Is Fluentd a DevOps Tool?**

Yes – Fluentd is widely used in DevOps, SRE, and observability setups.

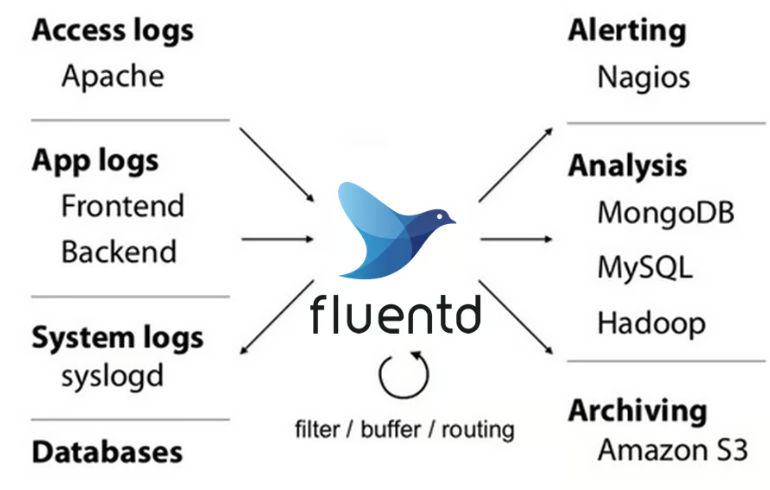
* In DevOps, one big challenge is log management. Applications, microservices, and containers generate tons of logs.
* Fluentd helps by collecting, filtering, and routing logs in a flexible way so that monitoring and alerting tools (like ELK, Prometheus, Grafana Loki, etc.) can use them.

So, it’s an important part of the DevOps logging & monitoring ecosystem.

**Common Use Cases**

Here’s what you can use Fluentd for (like your example):

1. **Centralized Logging**
   * Collect logs from multiple apps/containers.
   * Store them in Elasticsearch → view with Kibana dashboards.
2. **Log Forwarding in Docker/Kubernetes**
   * Collect logs from pods/containers.
   * Send them to **CloudWatch, Loki, Elasticsearch, or Splunk**.
3. C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmp**Log Filtering & Transformation**
   * Mask sensitive data (passwords, tokens).
   * Reformat logs before sending them to a monitoring system.
4. **Monitoring & Alerting Support**
   * Pipe structured logs into monitoring tools.
   * Makes troubleshooting production issues easier.
5. **Data Routing**
   * Route logs to multiple places (for example → local file + S3 + Elasticsearch at the same time).

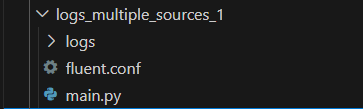


**Getting the Fluentd Image from Docker**

docker pull fluent/fluentd:v1.17-1

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmp

**1. Collecting Logs from multiple sources**



**fluent.conf**

<source>

  @type tail

  path /fluentd/log/app1.log

  pos\_file /fluentd/log/app1.pos

  tag app1.log

  format none

</source>

<match \*\*>

  @type stdout

</match>

**main.py**

import time

import random

import os

# Log file path

log\_file = "./logs/app1.log"

os.makedirs("logs", exist\_ok=True)

# Sample random English sentences

sentences = [

    "The quick brown fox jumps over the lazy dog.",

    "I love programming in Python.",

    "Fluentd makes log management easy.",

    "Docker containers simplify deployment.",

    "Learning new things every day keeps you sharp.",

    "Artificial Intelligence is the future of technology.",

    "Data science is both challenging and rewarding.",

    "Always keep your code clean and readable.",

    "Debugging can sometimes be fun.",

    "Consistency is key to mastering any skill.",

    "Reading books expands your knowledge.",

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmp    "Writing tests improves software quality.",

    "Practice makes perfect.",

    "Collaboration leads to better solutions.",

    "Innovation drives progress.",

    "Automation saves time and reduces errors.",

    "Understanding algorithms is essential.",

    "Stay curious and keep exploring.",

    "Technology changes rapidly, adapt quickly.",

    "Good communication improves teamwork."

]

# Generate 20 log messages

for i in range(20):

    # Pick a random sentence

    message = random.choice(sentences)

    # Print to console

    print(message)

    # Append to log file

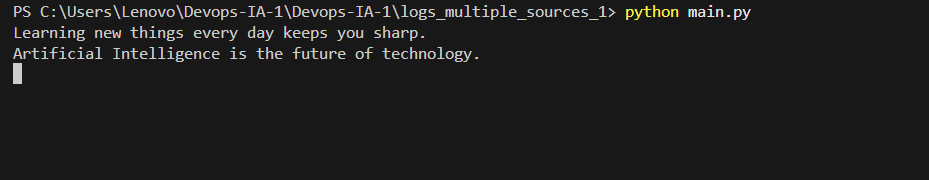
    with open(log\_file, "a") as f:

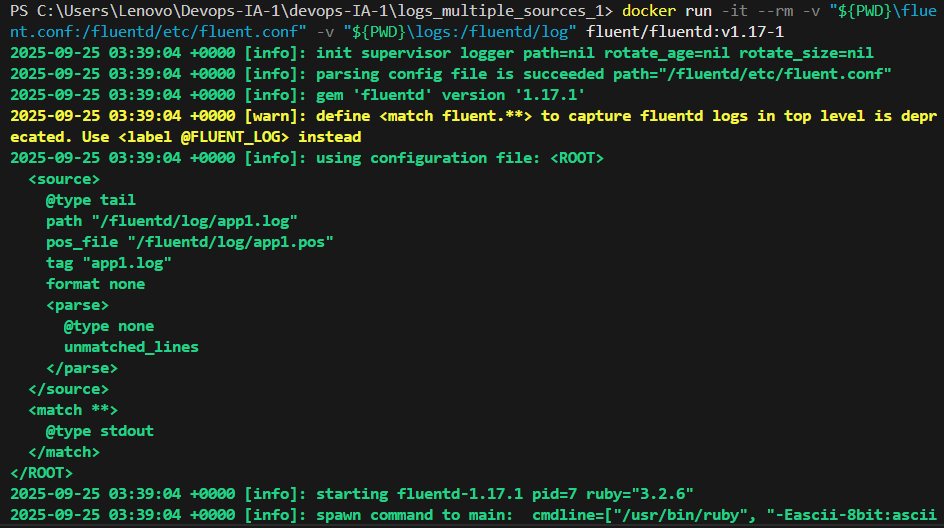
        f.write(message + "\n")

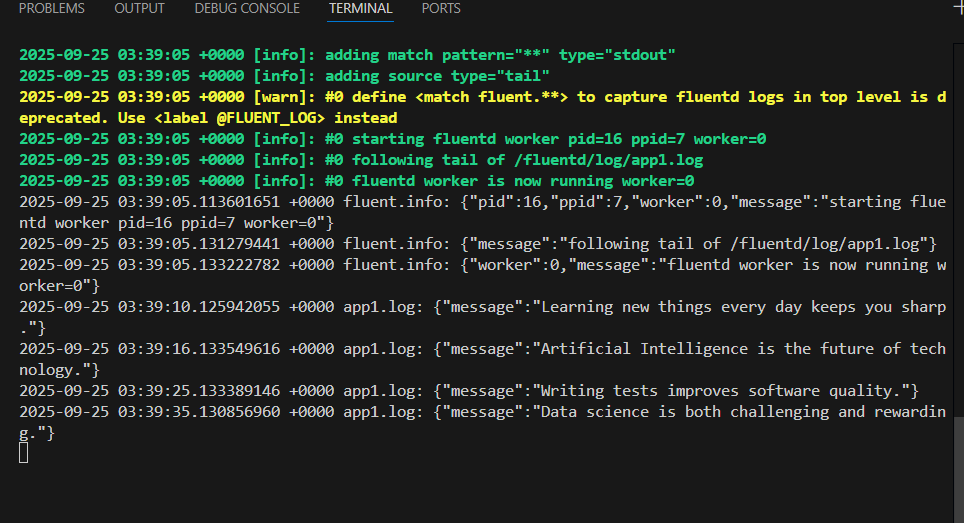
    # Wait randomly between 5 and 10 seconds

    time.sleep(random.randint(5, 10))

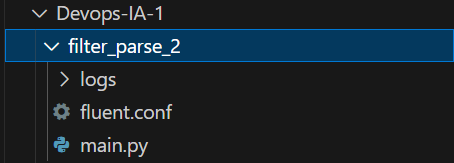
**Output**



C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmp



**2. Filter and parse logs**



**Fluent.conf**

# Tail login log file

<source>

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmp  @type tail

  path /fluentd/log/login.log

  pos\_file /fluentd/log/login.pos

  tag login

  format json

</source>

# Filter: only SUCCESS logins

<filter login>

  @type grep

  <regexp>

    key status

    pattern ^SUCCESS$

  </regexp>

</filter>

# Add hostname field

<filter login>

  @type record\_transformer

  <record>

    hostname "#{Socket.gethostname}"

  </record>

</filter>

# Output to console

<match login>

  @type stdout

</match>

**main.py**

import json

import os

from datetime import datetime

# Ensure logs folder exists

os.makedirs("logs", exist\_ok=True)

# Predefined users

users = {

    "user1@example.com": "password123",

    "user2@example.com": "mypassword",

    "admin@example.com": "admin123"

}

# Ask user input

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmpemail = input("Enter your email: ").strip()

password = input("Enter your password: ").strip()

# Verify credentials

status = "SUCCESS" if email in users and users[email] == password else "FAILURE"

message = "Logged in successfully" if status == "SUCCESS" else "Failed to login"

# Print to console

print(f"{message} for {email}")

# Create a structured log entry

log\_entry = {

    "time": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),

    "email": email,

    "status": status,

    "message": message

}

# Append log entry as JSON to file

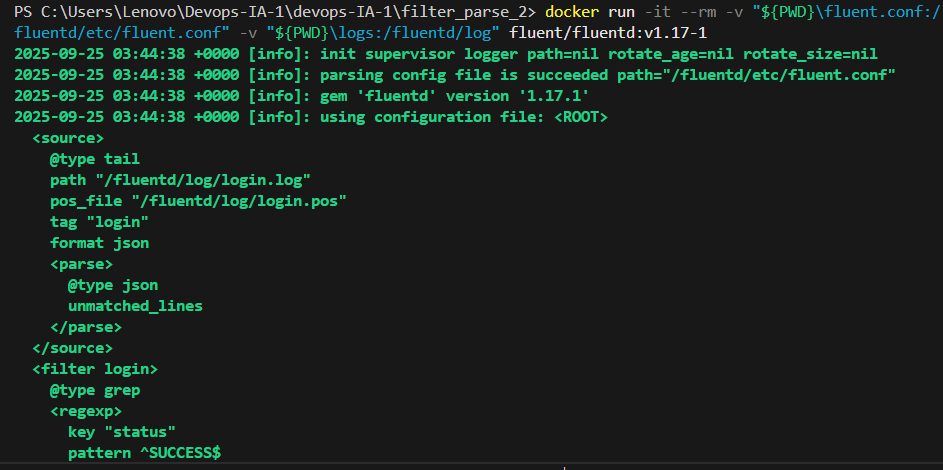
log\_file = "./logs/login.log"

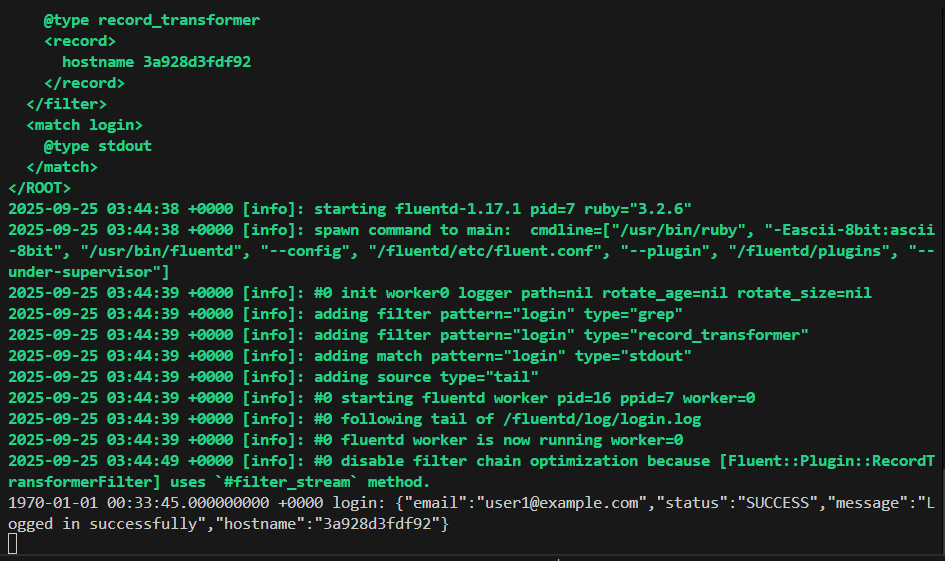
with open(log\_file, "a") as f:

    f.write(json.dumps(log\_entry) + "\n")

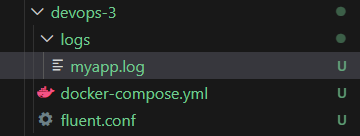
**Output**

****

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmp****

****

**3. Send Logs to Elasticsearch (ELK Stack)**

****

**docker-compose.yml**

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmpversion: "3"

services:

  elasticsearch:

    image: docker.elastic.co/elasticsearch/elasticsearch:8.15.0

    container\_name: elasticsearch

    environment:

      - discovery.type=single-node

      - xpack.security.enabled=false # disable auth for testing

    ports:

      - "9200:9200"

  kibana:

    image: docker.elastic.co/kibana/kibana:8.15.0

    container\_name: kibana

    environment:

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

    ports:

      - "5601:5601"

    depends\_on:

      - elasticsearch

  fluentd:

    image: fluent/fluentd:v1.16-1

    container\_name: fluentd

    volumes:

      - ./fluent.conf:/fluentd/etc/fluent.conf

      - ./logs:/var/log # directory with your logs

    depends\_on:

      - elasticsearch

    ports:

      - "24224:24224"

      - "24224:24224/udp"

**Fluent.conf**

<source>

  @type tail

  path /var/log/myapp.log

  pos\_file /fluentd/log/myapp.pos

  tag myapp.log

  format none

</source>

<match myapp.log>

  @type elasticsearch

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmp  host elasticsearch    # service name if using Docker

  port 9200

  logstash\_format true

  include\_tag\_key true

  type\_name \_doc

  flush\_interval 5s

</match>

**index.json**

{

  "settings": {

    "number\_of\_shards": 1,

    "number\_of\_replicas": 0

  },

  "mappings": {

    "properties": {

      "timestamp": { "type": "date" },

      "log": { "type": "text" },

      "level": { "type": "keyword" },

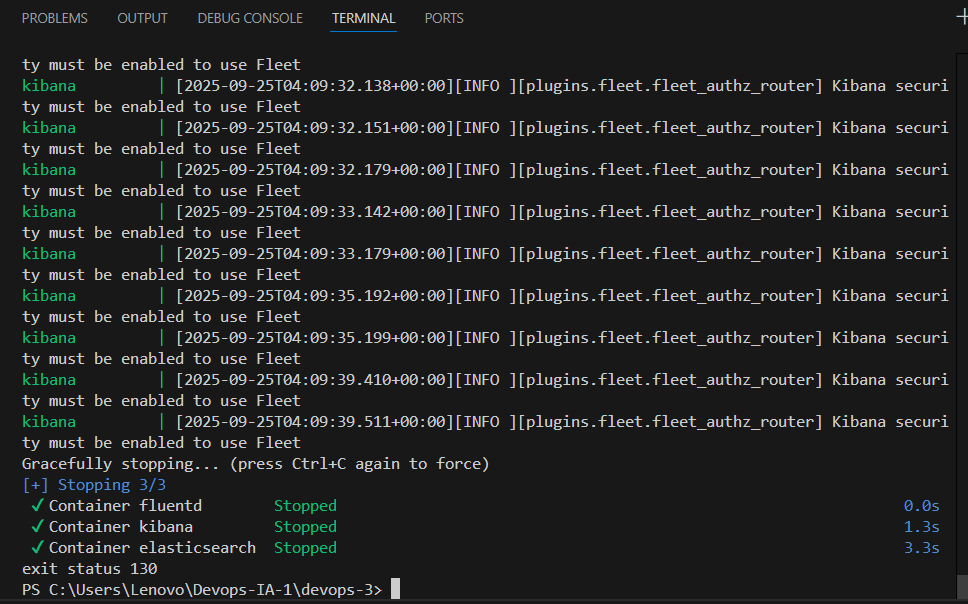
      "source": { "type": "keyword" }

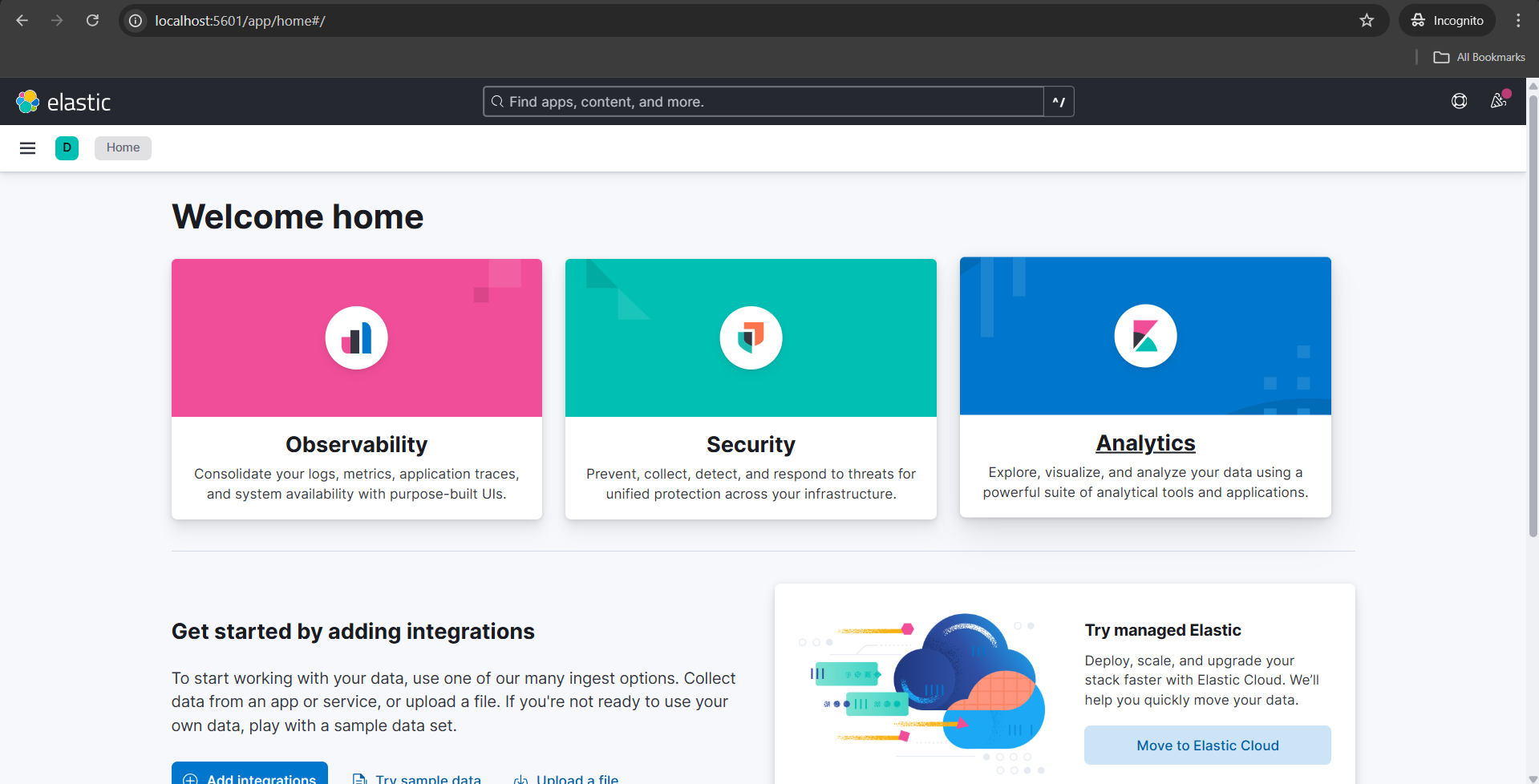
    }

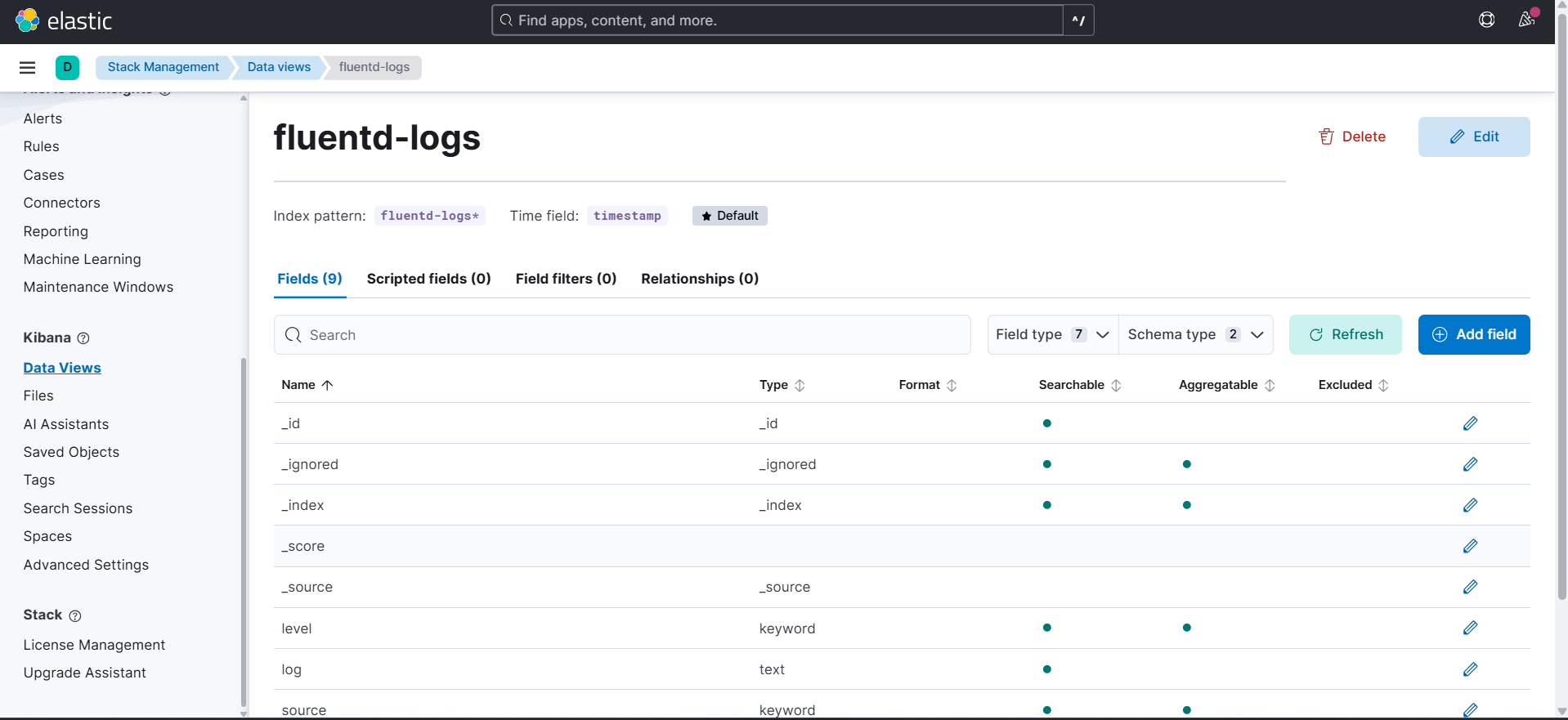
  }

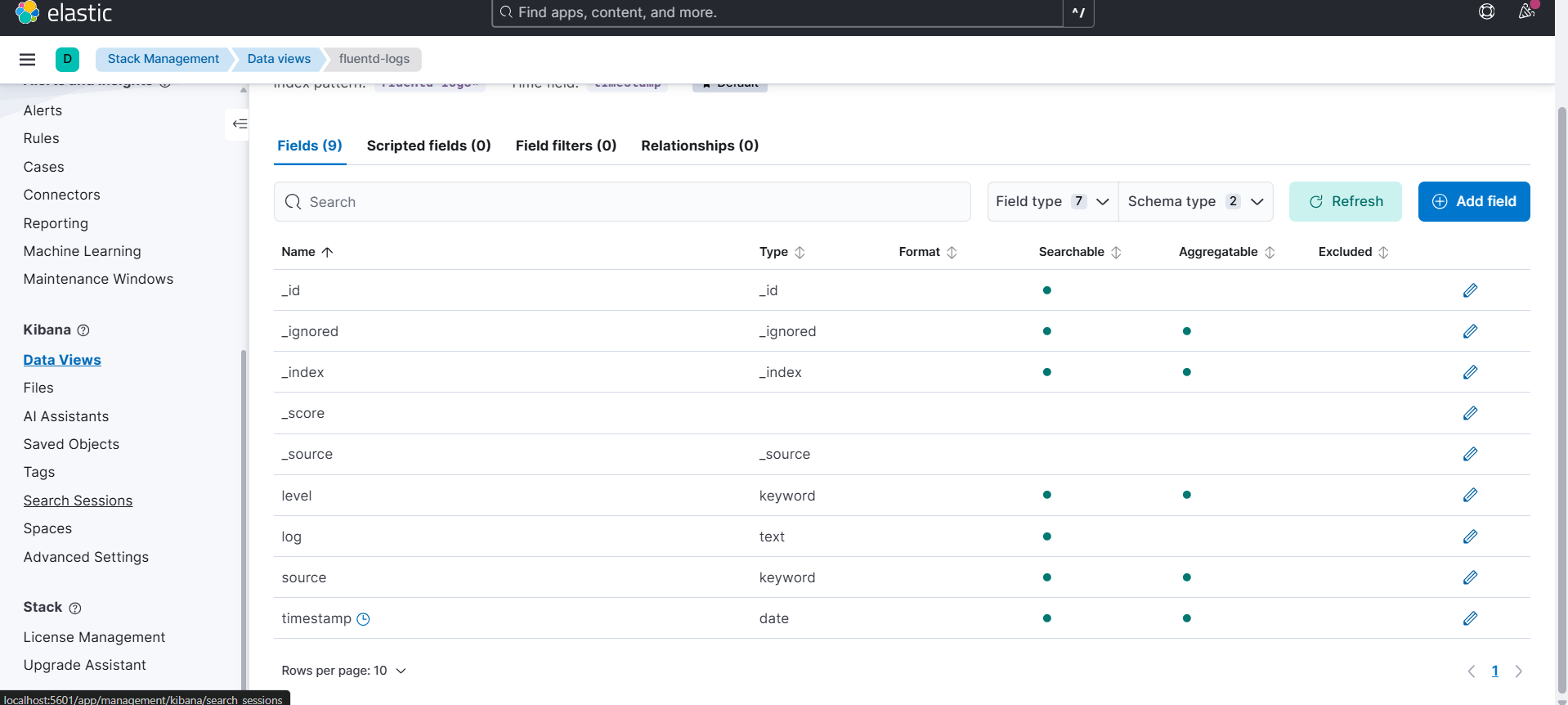
}

**Output**

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmp****

****

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D61C93F8.tmp****

****