
SPA-Assignment-2

Streaming Log Analytics

Student ID	Name	Contribution
2020SC04081	AMOGH SINGHAL	100%
2020SC04431	ABHAY DATTATRAYA THAPPAN	100%
2020SC04956	AKSHIT AGARWAL	100%

AUGUST 28, 2022

Birla Institute of Technology & Science, Pilani

Streaming Log Analytics Solution

Assignment details

Log analytics is a common big data use case that allows you to analyze log data from websites, mobile devices, servers, sensors, and more for a wide variety of applications such as digital marketing, application monitoring, fraud detection, ad tech, games, and IoT. In this assignment, you use **Open-Source tools of your choice** to build an end-to-end log analytics solution that collects, ingests, processes, and loads both batch data and streaming data, and makes the processed data available to your users in analytics systems they are already using and in near real-time.

Our Approach

As part of this assignment, we tried to explore multiple tools/resources that can help us achieve the goal. We would like to share some below

- ✚ **Graylog** - an Elasticsearch-based log management and analysis tool.
- ✚ **Fluentd** - It enables users to unify logs from various components and easily analyzes them.
- ✚ **GoAccess** - terminal-based open-source log analyzer that analyze and observe web server statistics in real-time.
- ✚ **Octopussy** - It sends alerts to networking devices about their applications and services supporting the Syslog protocol
- ✚ **Apache Flume** - one of the best open-source log analysis tools to use extensible data models for online analytic applications
- ✚ **ELK Stack** - enables users to aggregate logs from all connected systems and applications. It helps analyze logs and create visualizations for applications and infrastructure monitoring
- ✚ **LOGalyze** - centralized open-source management and network monitoring tool
- ✚ **syslog-ng** - It processes the gathered log data and transfers them to a preferred log analysis tool
- ✚ **NewRelic** - a web application performance service designed to work in real-time with your live web app
- ✚ **Splunk** - Splunk is a log search engine with native capability for building dashboards and alerts and out of the box capability to data insights using knowledge objects

Final Tool for implementation

We investigated pros/cons as well as practicality of using above sources without incurring cost on cloud and limiting scope due to our personal system resources. We were already aware of **Python `faker` library** through the classes and AWS kinesis solution, so log generation was not a problem. We were looking at a solution that can provide us with the required details as per assignment. We decided to go with **ELK stack**, which is a mélange of **Elasticsearch, Logstash and Kibana** plus using the faker generated logs for analysis. Let us share more about ELK, why we choose this and how it helps in Log analytics world.

Elasticsearch, Logstash and Kibana



- As its name suggests, **Elasticsearch** is designed to help users find matches within datasets using a wide range of query languages and types. Speed is this tool's number one advantage. It can be expanded into clusters of hundreds of server nodes to handle petabytes of data with ease.
- **Kibana** is a visualization tool that runs alongside Elasticsearch to allow users to analyze their data and build powerful reports. When you first install the Kibana engine on your server cluster, you will gain access to an interface that shows statistics, graphs, and even animations of your data.
- The final piece of ELK Stack is **Logstash**, which acts as a purely server-side pipeline into the Elasticsearch database. You can integrate Logstash with a variety of coding languages and APIs so that information from your websites and mobile applications will be fed directly into your powerful Elastic Stalk search engine.

Why Use ELK Stack?

In a data-driven world, databases must constantly handle increasingly larger amounts of data. Typically, analytic processes slow down as the amount of data a system handles continues to increase. The ELK stack can help increase these analytic processes. A brief overview of the benefits of the ELK stack includes:

- ❖ ELK is a total log-analysis platform for **search, analyses and visualization** of log-generated data from different machines.
- ❖ ELK can securely **pull, analyze and visualize data, in real time**, from any source and format.
- ❖ ELK can perform **centralized logging** to help identify any server and application-related issues across multiple servers and **correlate the logs** in a particular time frame.
- ❖ ELK is geared to **handle big data** to provide crucial business insights.

- ❖ ELK is simple to use, set up and is **user friendly**.
- ❖ As an open-source program, Elk is **highly cost effective**.

Log Analytics

Dozens of servers running multiple applications results in a lot of data to analyze. Logs are one of the most critical, but often overlooked, data sources. Within a company's web-server logs, every log file holds mostly unstructured information that is difficult, or sometimes, impossible to interpret. ELK can quickly analyze the log data and to identify opportunities as well as possible vulnerabilities.

It is critical to understand how the system works whenever problems arise. Having the ability to quickly locate the needed information will help expedite operations-related tasks and resolve problems. Additionally, adding metrics to correlate logs provides increased visibility to help see the log history, what is currently happening and predict where a trend is headed.

DBAs typical must log on to multiple machines and comb through numerous files when an error occurs. The larger the system is the bigger a nightmare this becomes. ELK being able to turn this migraine-sized problem into a minor annoyance is a major reason why use ELK stack.

Streaming data pipeline Architecture

The proposed architectural diagram that we have created can be summarized below



Fig1. Explains ELK (Elasticsearch, Logstash, Kibana) integration with python Faker library that generates random logs continuously (Apache access logs in our case)

Technical details

Log generation: We followed code from faker-apache-logs-generation repository that has code from open sources with personal updates.

Thappana on 5CG84508F4 > OSDisk (C:) > Users > thappana > DOCKER > ELK > python > fake-log-generator				
Name	Date modified	Type	Size	
access_log_20220829-185043.log	8/30/2022 12:39 AM	Text Document	114,375 KB	
apache-fake-log-gen.py	8/14/2022 11:51 PM	PY File	4 KB	
log-gen.sh	8/18/2022 10:35 PM	Shell Script	1 KB	
requirements.txt	8/14/2022 11:35 PM	Text Document	1 KB	
wait.py	8/19/2022 12:18 AM	PY File	1 KB	

This script generates a boatload of fake Apache logs very quickly. It is useful for generating fake workloads for data ingest and/or analytics applications. It can write log lines to console, to log files or directly to zip files. It utilizes the excellent Faker library to generate realistic IP's, URI's etc.












The Docker file listed below will help us smoothly deploy the code and generate logs with a wait time of 4000 milliseconds. Run the docker compose file and we will see logs getting generated. All the logs will be stored in file path given. Here is a sample path and file below:

Thappan, Abhay (CORP) > DOCKER > ELK > python				
Search python				
Name	Date modified	Type	Size	
fake-log-generator	8/30/2022 12:20 AM	File folder		
Dockerfile	8/19/2022 12:32 AM	File	1 KB	

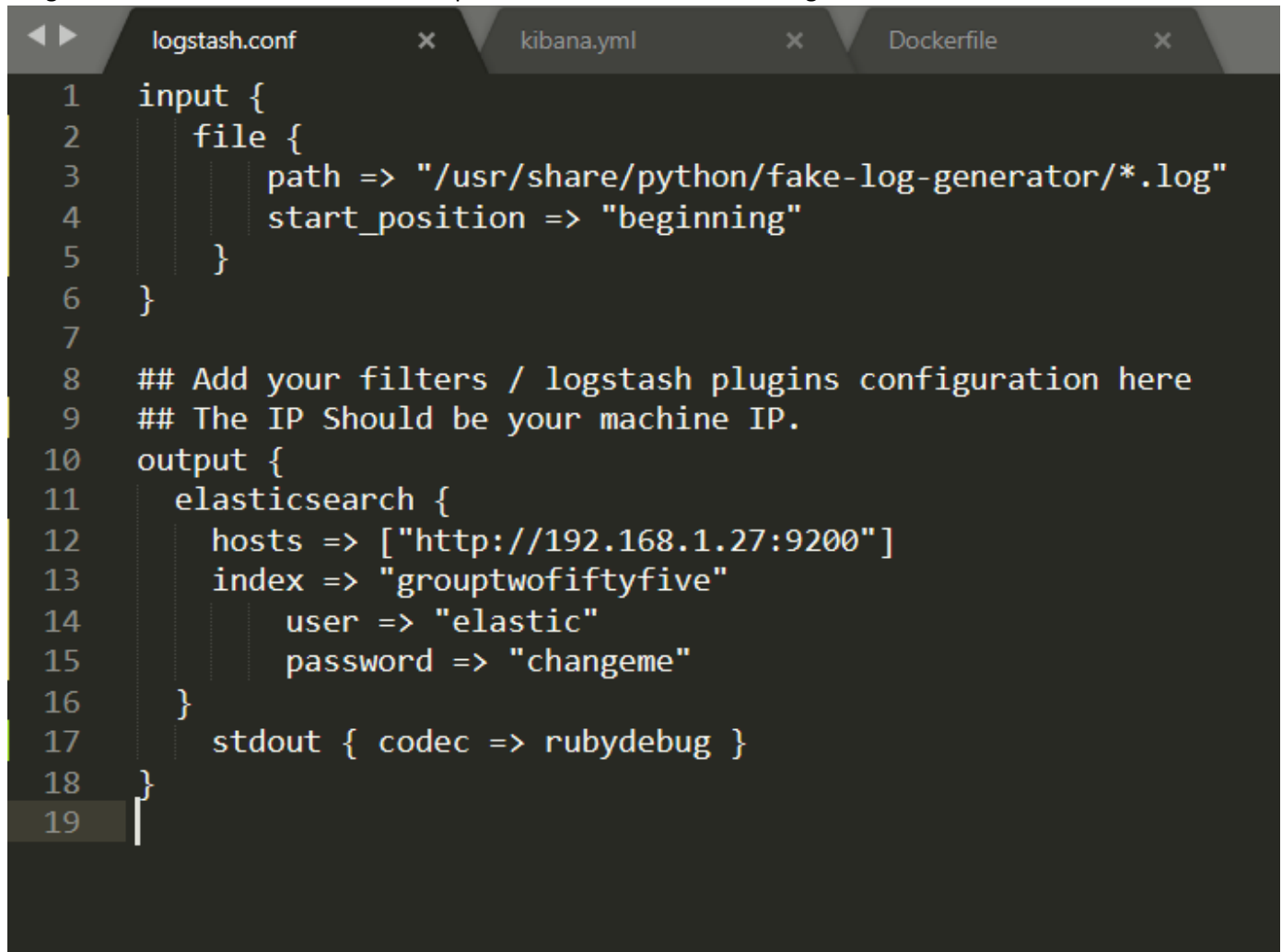
Thappan, Abhay (CORP) > DOCKER > ELK > python > fake-log-generator					Search fake-	
	Name	Date modified	Type	Size		
\$	access_log_20220829-185043.log	8/30/2022 12:39 AM	Text Document	114,375 KB		
ts	apache-fake-log-gen.py	8/14/2022 11:51 PM	PY File	4 KB		
ls	log-gen.sh	8/18/2022 10:35 PM	Shell Script	1 KB		
	requirements.txt	8/14/2022 11:35 PM	Text Document	1 KB		
Inasv7	wait.py	8/19/2022 12:18 AM	PY File	1 KB		

ELK Installation

To install Elastic stack, we need to run the latest version of the [Elastic stack](#) with Docker and Docker Compose. We followed open-source repository with individual updates to .yaml files and can be seen here Github code- **ELK-installation-using-docker**

 .env	Add files via upload	1 hour ago
 Dockerfile	Add files via upload	7 minutes ago
 LICENSE	Add files via upload	1 hour ago
 README.md	Add files via upload	1 hour ago
 docker-compose.yml	Add files via upload	1 hour ago
 elasticsearch.yml	Add files via upload	7 minutes ago
 entrypoint.sh	Add files via upload	7 minutes ago
 helpers.sh	Add files via upload	7 minutes ago
 kibana.yml	Add files via upload	7 minutes ago
 logstash.conf	Add files via upload	7 minutes ago
 logstash.yml	Add files via upload	7 minutes ago

We had to make few changes to Logstash configuration files like below to provide system IP address and log file location. This will allow the components to interact and make logs accessible.

A screenshot of a code editor with three tabs: 'logstash.conf', 'kibana.yml', and 'Dockerfile'. The 'logstash.conf' tab is active, showing a configuration file with line numbers 1 through 19. The configuration includes an 'input' section with a 'file' plugin, an 'output' section with an 'elasticsearch' plugin, and a 'stdout' plugin. Comments indicate where to add filters and plugins, and specify the IP address and password for the elasticsearch output.

```
1 input {
2   file {
3     path => "/usr/share/python/fake-log-generator/*.log"
4     start_position => "beginning"
5   }
6 }
7
8 ## Add your filters / logstash plugins configuration here
9 ## The IP Should be your machine IP.
10 output {
11   elasticsearch {
12     hosts => ["http://192.168.1.27:9200"]
13     index => "grouptwofiftyfive"
14     user => "elastic"
15     password => "changeme"
16   }
17   stdout { codec => rubydebug }
18 }
19
```

We just need to start the docker. Attaching the screenshot of executing docker installation file on windows 11.

1. Typing the **cmd** command from our docker file path location will open windows prompt on that path
2. We can then start docker with command ***docker-compose up -d***

C:\Windows\System32\cmd.exe

```
C:\Users\thappana\DOCKER\ELK>docker compose up -d
[+] Building 8.4s (25/25) FINISHED
=> [elk_setup internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [elk_elasticsearch internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [elk_python internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [elk_kibana internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [elk_logstash internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [elk_setup internal] load .dockerignore
=> => transferring context: 35B
=> [elk_elasticsearch internal] load .dockerignore
=> => transferring context: 34B
=> [elk_python internal] load .dockerignore
=> => transferring context: 2B
```

```
=> CACHED [elk_python 4/4] RUN pip install --no-cache-dir pytz

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[+] Running 1/1
[+] Running 3/1elk Created
[+] Running 8/8elk Created
- Network elk_elk Created
- Volume "elk_elasticsearch" Created
- Volume "elk_setup" Created
- Container elk-elasticsearch-1 Started
- Container elk-python-1 Started
- Container elk-logstash-1 Started
- Container elk-setup-1 Started
- Container elk-kibana-1 Started

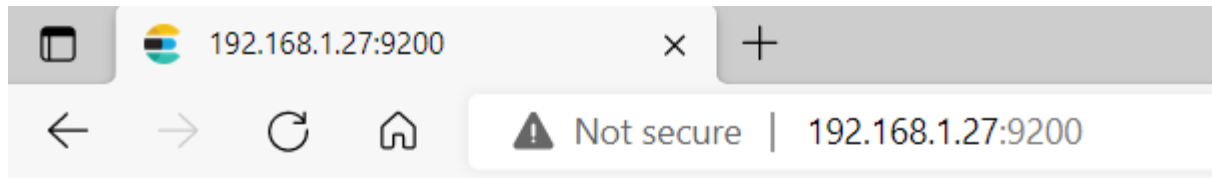
C:\Users\thappana\DOCKER\ELK>
```

3. Once the ELK stack is up, we can see them in docker hub and explore it via console/UI ports too
- ### Docker hub

The screenshot shows the Docker Desktop interface. On the left sidebar, the 'Containers' tab is selected. The main area displays a list of running containers. A search bar is at the top right of the container list. The containers listed are:

NAME	IMAGE	STATUS	PORT(S)	STARTED
elk 5 containers	-	Running (5/5)	-	-
elasticsearch-1 e78eaa5aad7c	elk_elasticsearch:latest	Running	9200,9...	24 seconds ago
setup-1 2df2bd2720c1	elk_setup:latest	Running	-	21 seconds ago
python-1 c25c4f855ee7	elk_python:latest	Running	9001	20 seconds ago
logstash-1 83313da3d22f	elk_logstash:latest	Running	50000...	19 seconds ago
kibana-1 9d335de14696	elk_kibana:latest	Running	5601	19 seconds ago

We will use the default username (**elastic**) and password (**changeme**) for Elasticsearch/Kibana while accessing it on 9200/5601 ports respectively. We can see Elasticsearch running and the index “**grouptwofiftyfive**” getting created.



```
{
  "name" : "565f30368863",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "gWtS1PzySHmRZwYEQRGSYA",
  "version" : {
    "number" : "8.3.3",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "801fed82df74dbe537f89b71b098ccaff88d2c56",
    "build_date" : "2022-07-23T19:30:09.227964828Z",
    "build_snapshot" : false,
    "lucene_version" : "9.2.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```



health	status	index	uuid	pri	rep	docs.count	docs.deleted	store.size	pri.store.size
yellow	open	grouptwofiftyfive	hnp7n1TvRsKiIX8ZfSBfwg	1	1	420846	0	309mb	309mb

4. In Kibana, search for Index Management and we will see the entries for “grouptwofiftyfive” index that we generated using faker python file even before ELK installation.

The screenshot shows the Kibana Index Management interface. The left sidebar contains a 'Management' section with 'Ingest' (Ingest Pipelines, Logstash Pipelines), 'Data' (Index Management, Index Lifecycle Policies, Snapshot and Restore, Rollup Jobs, Transforms, Cross-Cluster Replication, Remote Clusters), and 'Alerts and Insights' (Rules and Connectors, Cases). The main content area is titled 'Index Management' and has tabs for 'Indices', 'Data Streams', 'Index Templates', and 'Component Templates'. Below the tabs, there is a search bar and a 'Manage index' button. A table lists the indices, with 'grouptwofiftyfive' selected. The table columns are Name, Health, Status, Primaries, Replicas, Docs count, Storage size, and Data stream. The 'grouptwofiftyfive' index has a health of 'yellow', status of 'open', 1 primary, 1 replica, 517051 docs, and 313.68mb storage. A 'Reload indices' button is in the top right.

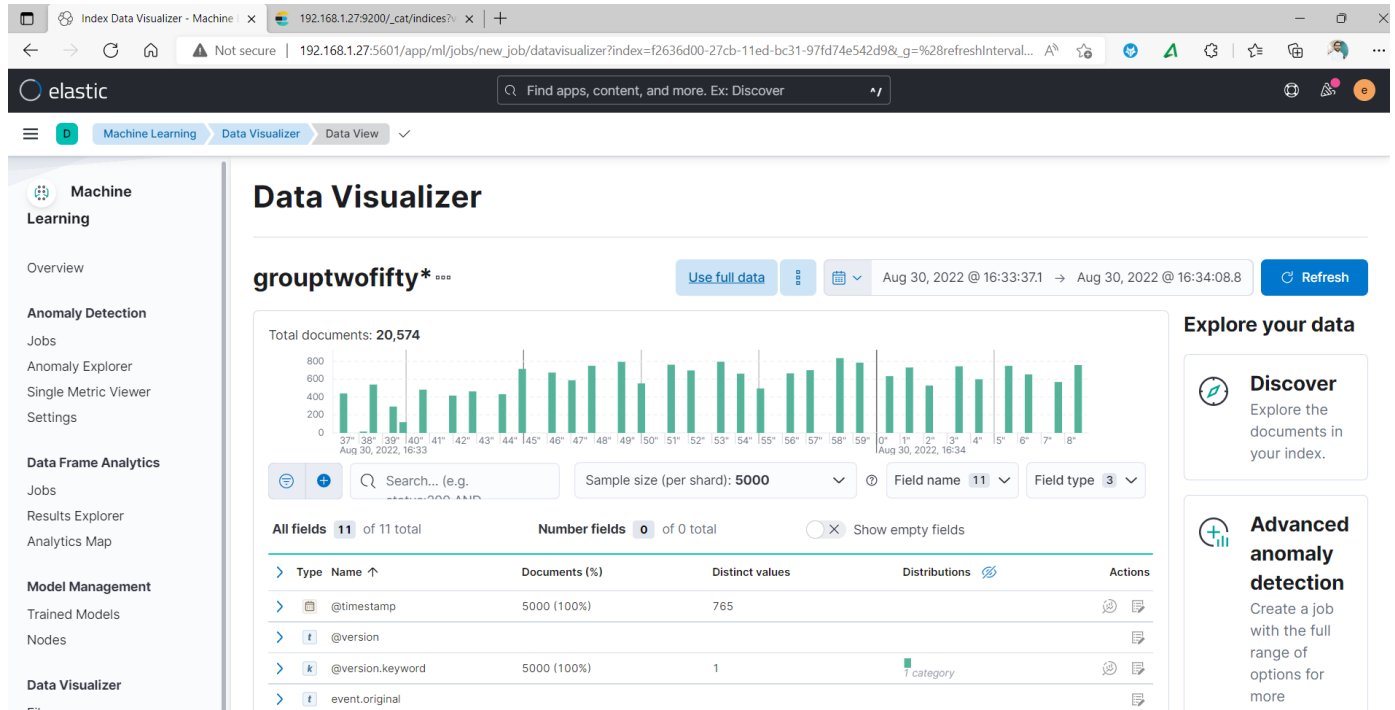
Name	Health	Status	Primaries	Replicas	Docs count	Storage size	Data stream
grouptwofiftyfive	yellow	open	1	1	517051	313.68mb	

The screenshot shows the Kibana Index Management interface with the 'grouptwofiftyfive' index selected. The left sidebar is the same as the previous screenshot. The main content area is titled 'grouptwofiftyfive' and has tabs for 'Summary', 'Settings', 'Mappings', 'Stats', and 'Edit settings'. The 'Settings' tab is active, showing a JSON configuration for the index. The configuration includes settings for routing, allocation, number of shards, provided name, creation date, number of replicas, uuid, version, and defaults.

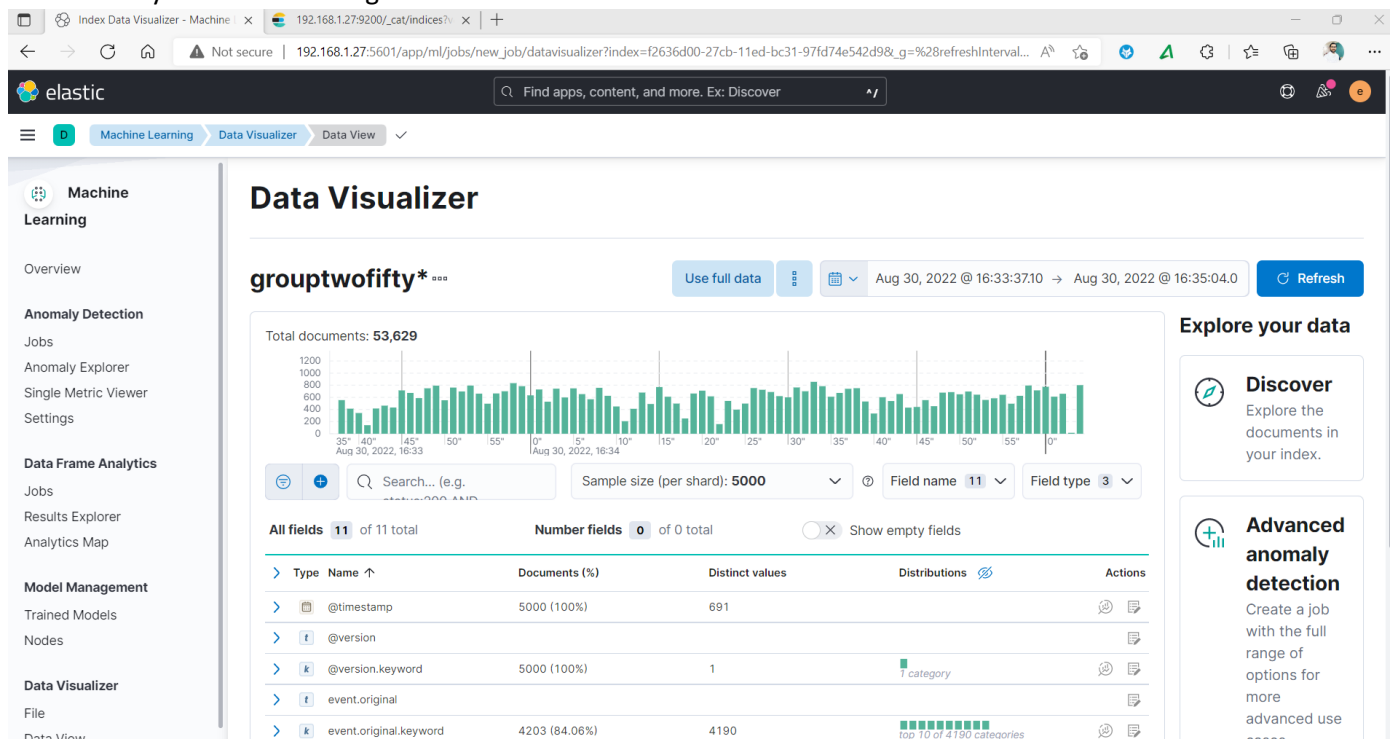
```
{
  "settings": {
    "index": {
      "routing": {
        "allocation": {
          "include": {
            "_tier_preference": "data_content"
          }
        }
      },
      "number_of_shards": "1",
      "provided_name": "grouptwofiftyfive",
      "creation_date": "1661799052738",
      "number_of_replicas": "1",
      "uuid": "hnp7n1TvRsKiIX8ZfSBfwg",
      "version": {
        "created": "8030399"
      }
    },
    "defaults": {
      "index": {
        "flush_after_merge": "512mb",
        "final_pipeline": "_none",
        "max_inner_hits": "1000"
      }
    }
  }
}
```

5. We will go to data Visualization and see how our logs data looks now.

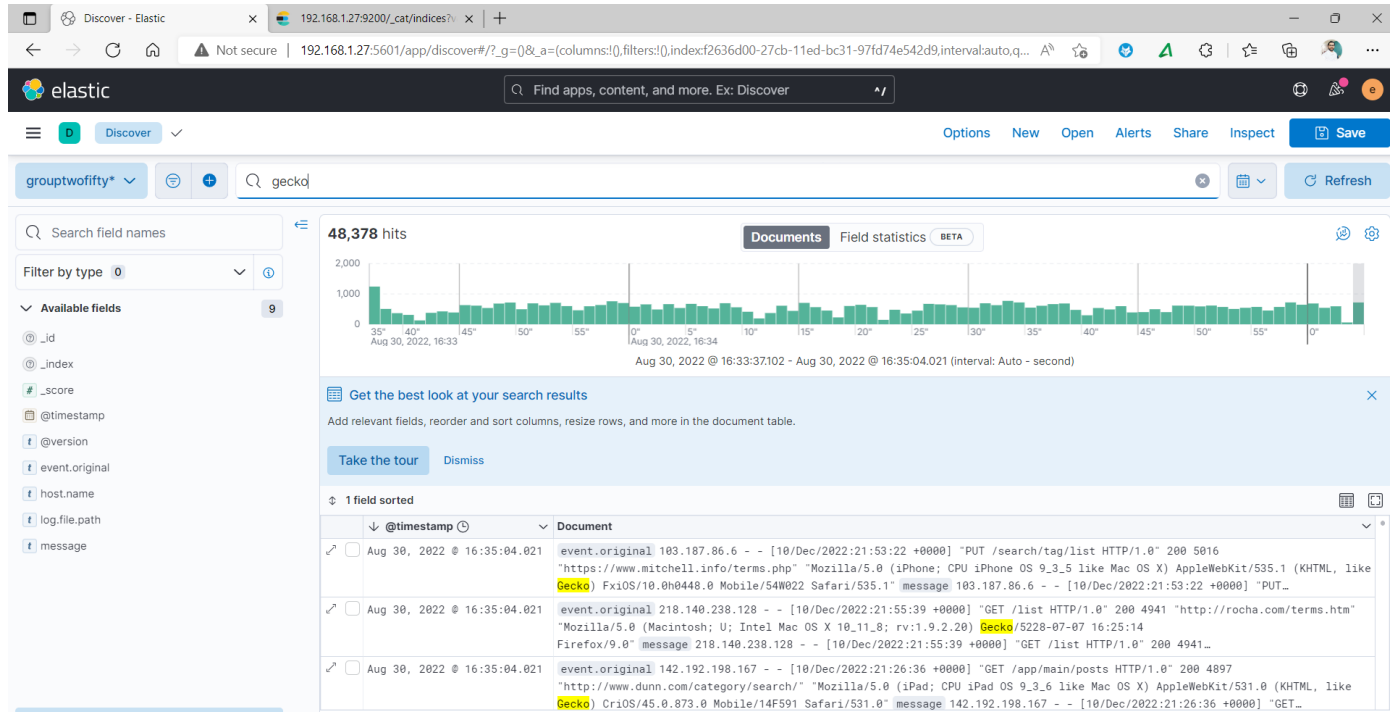
We can now see initial view of our data in Kibana as well. Count is **20,574**



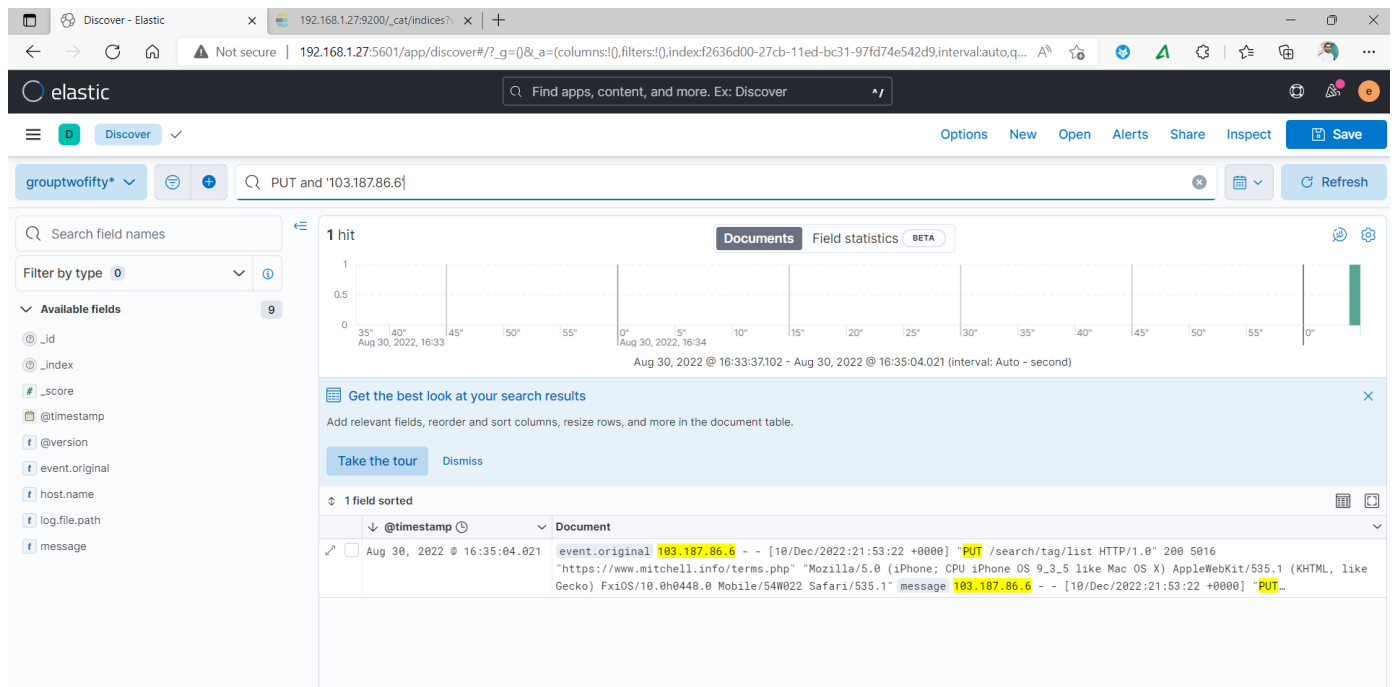
After 1 Min you can see the log documents are more below which is **53,629**



We can query strings from Kibana to search log for us. Let's search for **Gecko**.



Search for little complex query **"PUT and '103.187.86.6'"**



We can further process the logs before logging or use the keywords/rules based on our requirement to build alerts in Kibana.