ANNEXURE-2

Release 14.5.3.0.0

Part No. F50590-01

November 2021

ORACLE

ANNEXURE-2

Oracle Financial Services Software Limited
Oracle Park
Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India
Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax: +91 22 6718 3001
www.oracle.com/financialservices/

ORACLE

December 16, 2021 2:06 PM

Table of Contents

1. ANNEXURE

1.1 Introduction

This documents is supporting document, while installing Zipkin and ELK you may find reference.

2. Document Tracing Zipkin

2.1 Installation of Zipkin

2.1.1 Download and Running

Zipkin works as an independent application and it can be downloaded as a runnable jar from the official website of Zipkin : https://zipkin.io/. The latest version of Zipkin needs a Java version above 8.

The direct download link of jar is as follows:

https://search.maven.org/remote_content?g=io.zipkin&a=zipkin-server&v=LATEST&c=exec

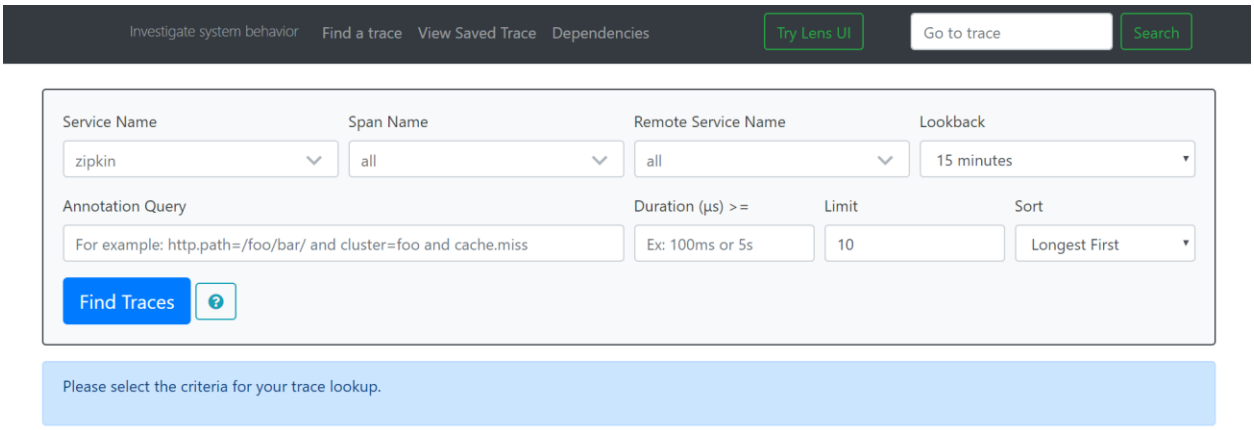
The downloaded jar can be executed using the java –jar JAR_NAME command.

The configuration of Zipkin can be done environment variables. The port of the Zipkin can be set using QUERY_PORT environment variable.

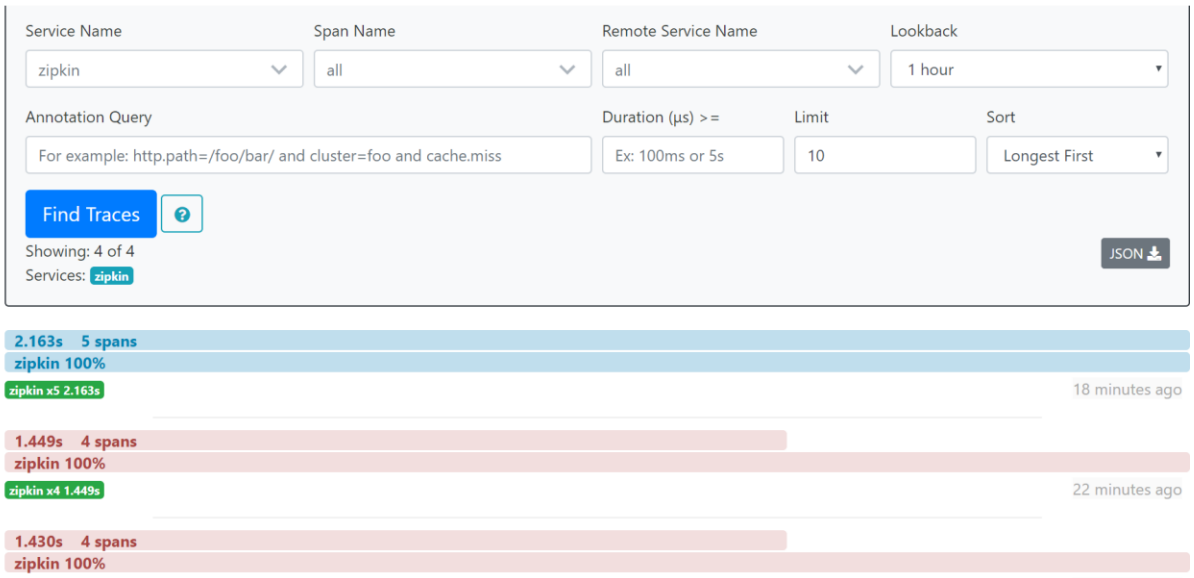
The application starts on the port number assigned for QUERY_PORT environment variable or its default value of 9411. The web UI of Zipkin can be accessed at http://localhost:PORT.

2.2 Zipkin User Interface

The basic layout of Zipkin looks as follows

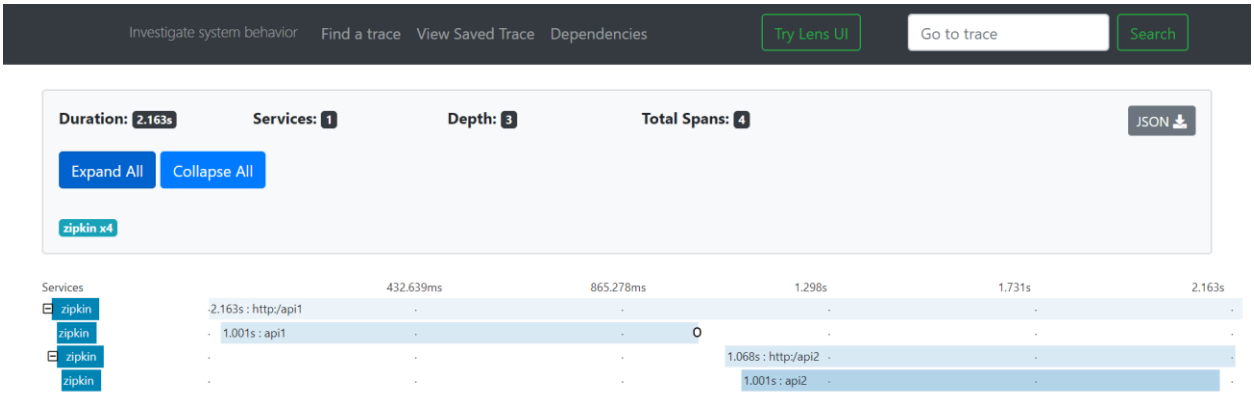


We can find the traces of required api calls and services using the above search options given in the user interface. The search options given in the user interface are self-explanatory and there is another UI option (Try Lens UI). It's given a different user interface with same functionality.

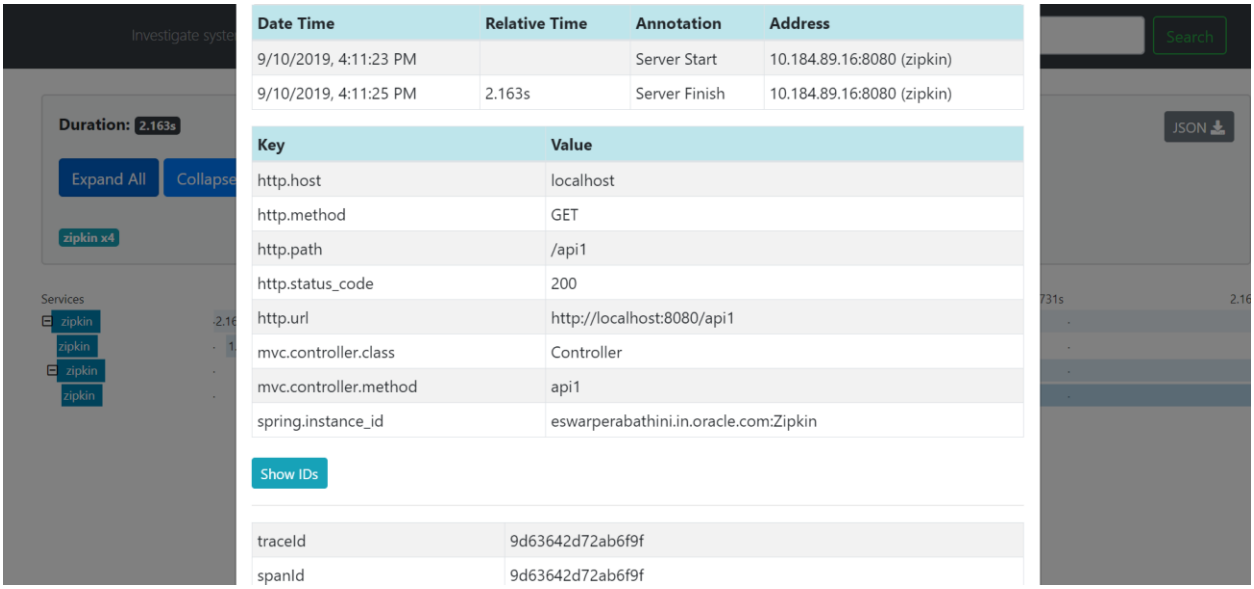


The list of the traces can be seen like the above screen. Some error API calls are made to showcase how to track errors. The blue listings show the successful API hits and the red listings indicate errors. Each block indicates a single trace in the listings.

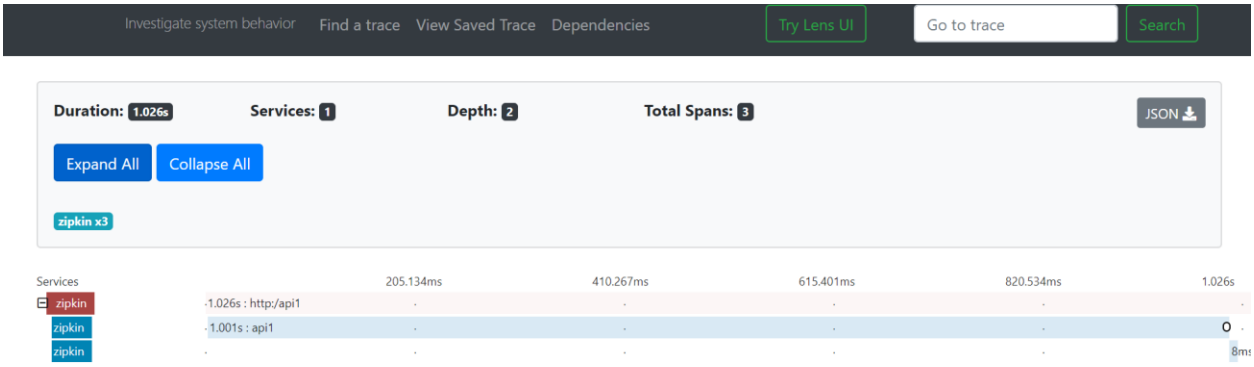
Opening an individual trace shows the below shown screen.



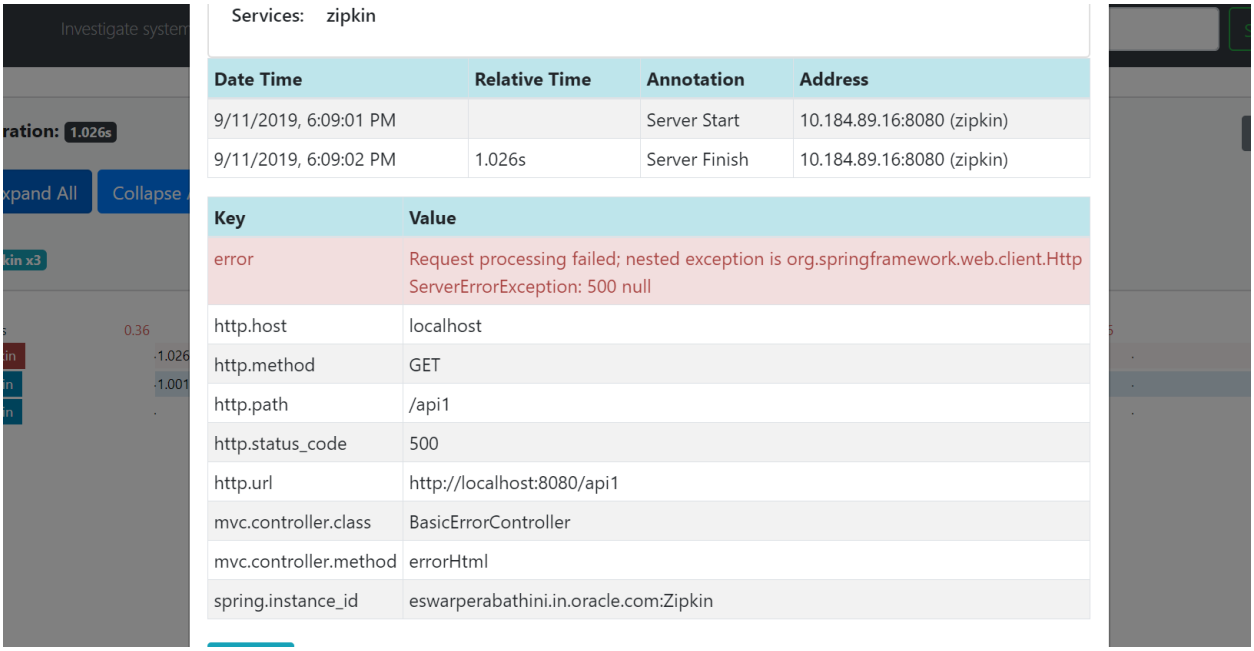
The above shown image describes the time taken for each block. There are 2 custom spans created inside 2 service calls, so there are total of 4 blocks. The time taken for individual block can be seen above. Clicking an individual block shows the following details.



The details of the specific span block are shown above and the logging events can also be seen in the Zipkin UI as small circular blocks. An example of error log is shown below.

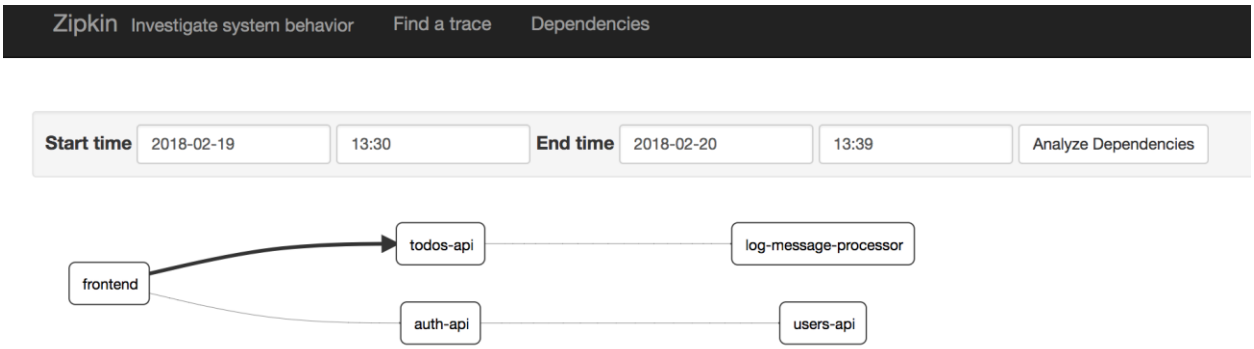


Clicking on the error portion gives the clear detail about the error and where the error has arised. AN example is shown below.



If the Lens UI is used in Zipkin, the above screen shots are not applicable, but are relatable to the Lens UI as well.

Traces of the application can be found using TraceId, which can be found in the debug logs of the deployment when spring-cloud-sleuth is included in the dependencies (Included in spring-cloud-starter-zipkin dependency). Clicking the dependency tab gives the dependency graph info between micro-services. An example dependency graph is shown below.



3. Monitoring ELK

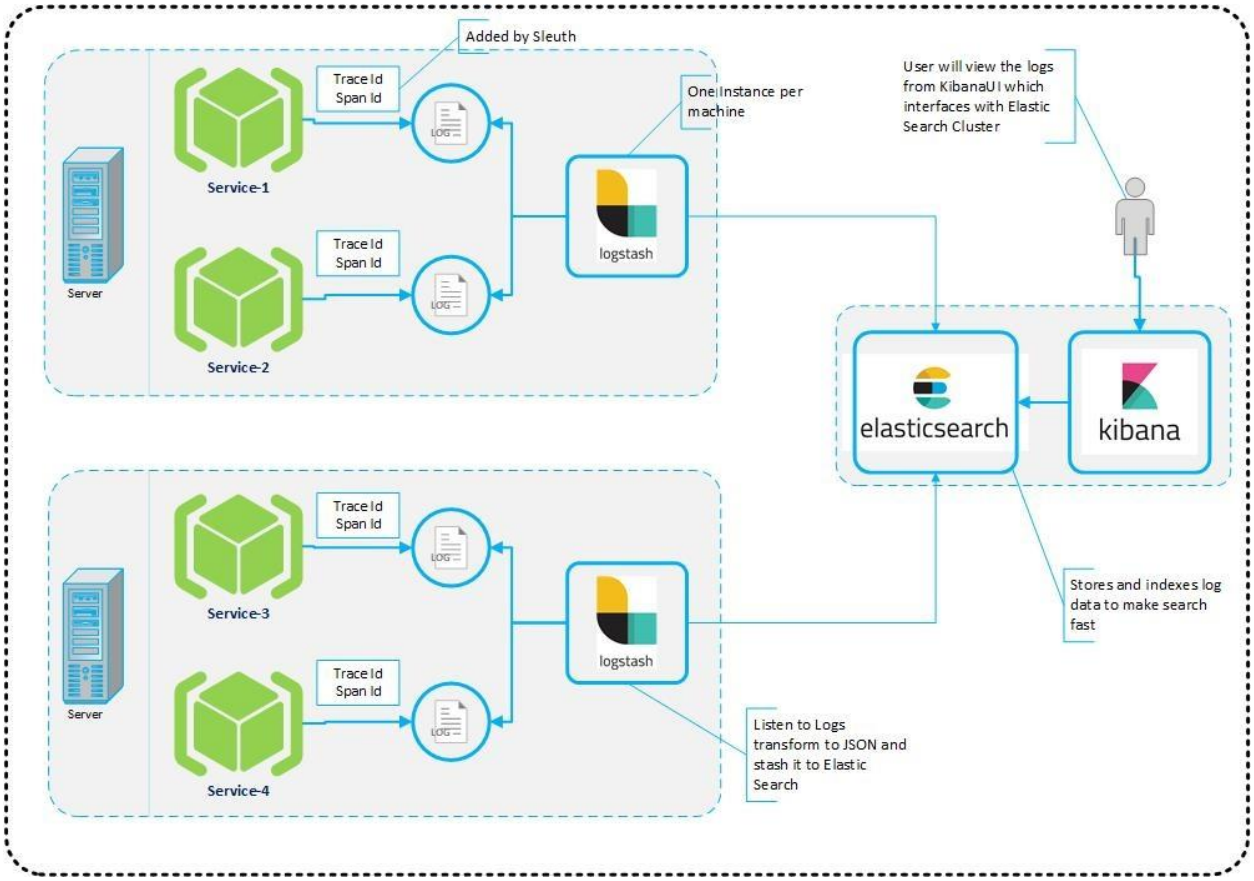
3.1 Introduction

ELK Stack was a collection of three open-source products — Elasticsearch, Logstash, and Kibana. Elasticsearch is an open source, full-text search and analysis engine, based on the Apache Lucene search engine. Logstash is a log aggregator that collects data from various input sources, executes different transformations and enhancements and then ships the data to various supported output destinations. Kibana is a visualization layer that works on top of Elasticsearch, providing users with the ability to analyze and visualize the data.

Together, these different components are most commonly used for monitoring, troubleshooting and securing IT environments. Logstash take care of data collection and processing, Elasticsearch indexes and stores the data, and Kibana provides a user interface for querying the data and visualizing it.

3.2 Architecture

The below architecture provides a comprehensive solution handling all the required facets



Spring cloud Sleuth also provides additional functionality to keep trace of the application calls by providing us a way to create intermediate logging events. So Spring Cloud Sleuth dependency must be added to applications.

3.3 Installing & Configuring ELK

To install and configure ELK Stack, make sure the versions of the 3 software are same. Download the latest version of

1. Logstash
2. Elastic Search
3. Kibana

The installation guides are given below.

1. Logstash : https://www.elastic.co/guide/en/logstash/current/installing-logstash.html
2. Elastic Search : https://www.elastic.co/guide/en/elasticsearch/reference/current/install-elasticsearch.html
3. Kibana : https://www.elastic.co/guide/en/kibana/current/install.html

Follow the process below after completing the download process of ELK.

3.3.1 Setup

3.3.1.1 Start ElasticSearch

1. Go to Elasticsearch root folder and use nohup to start the Elasticsearch process as below:

```
> nohup ./bin/elasticsearch
```

3.3.1.2 Setup Logstash and start

1. Create a new **logstash.conf** file that provides the required file parsing and integration to Elasticsearch

   logstatsh.conf:

```
#Point to the application logs
input {
  file {
    type => "java"
    path => "/scratch/app/work_area/app_logs/*.log"
    codec => multiline {
      pattern => "^%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{TIME}.*"
      negate => "true"
      what => "previous"
    }
  }
}
#Provide the parsing logic to transform logs into JSON
filter {
```

```
  #If log line contains tab character followed by 'at' then we will tag
that entry as stacktrace
  if [message] =~ "\tat" {
    grok {
      match => ["message", "^(\tat)"]
      add_tag => ["stacktrace"]
    }
  }

  #Grokking Spring Boot's default log format
  grok {
    match => [ "message",
                "(?<timestamp>%{YEAR}-%{MONTHNUM}-%{MONTHDAY}
%{TIME})  %{LOGLEVEL:level} %{NUMBER:pid} --- \[(?<thread>[A-Za-z0-9-]+)\]
[A-Za-z0-9.]*\.(?<class>[A-Za-z0-9#_]+)\s*:\s+(?<logmessage>.*)",
                "message",
                "(?<timestamp>%{YEAR}-%{MONTHNUM}-%{MONTHDAY}
%{TIME})  %{LOGLEVEL:level} %{NUMBER:pid} --- .+? :\s+(?<logmessage>.*)"
              ]
  }
   # pattern matching logback pattern
   grok {
          match =>
{ "message" => "%{TIMESTAMP_ISO8601:timestamp}\s+%{LOGLEVEL:severity}\s+\[
%{DATA:service},%{DATA:trace},%{DATA:span},%{DATA:exportable}\]\s+\[%{DATA
:environment}\]\s+\[%{DATA:tenant}\]\s+\[%{DATA:user}\]\s+\[%{DATA:branch}
\]\s+%{DATA:pid}\s+---
\s+\[%{DATA:thread}\]\s+%{DATA:class}\s+:\s+%{GREEDYDATA:rest}" }
   }
  #Parsing out timestamps which are in timestamp field thanks to previous
grok section
  date {
    match => [ "timestamp" , "yyyy-MM-dd HH:mm:ss.SSS" ]
  }
}
#Ingest logs to Elasticsearch
output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

2. Start Logstash process

```
>nohup ./bin/logstash -f logstash.conf
```

3.3.1.3 Setup Kibana and start

1. Go to the **kibana.yml** available under <kibana_setup_folder>/config and modify the file to include the below:

```
#Uncomment the below line and update the IP address to your host machine IP.
server.host: "xx.xxx.xxx.xx"
#Provide the elasticsearch url. If this is running on the same machine then you can use the below config as is
elasticsearch.url: "http://localhost:9200"
```

2. Start Kibana process using the below command:

```
>nohup ./bin/kibana
```

A view of the Kibana dashboard is given below: