

ANNEXURE - 2

Oracle Banking Virtual Account Management

Release 14.4.0.1.0

Part Number F33425-01

August 2020



Table of Contents

- 1. ANNEXURE - 2 1-1
 - 1.1 INTRODUCTION 1-1
- 2. DOCUMENT TRACING ZIPKIN 2-1
 - 2.1 INSTALLATION OF ZIPKIN..... 2-1
 - 2.1.1 Download and Running 2-1
 - 2.2 ZIPKIN USER INTERFACE..... 2-1
- 3. MONITORING ELK 3-1
 - 3.1 INTRODUCTION 3-1
 - 3.2 ARCHITECTURE..... 3-1
 - 3.3 INSTALLING & CONFIGURING ELK..... 3-2
 - 3.3.1 Setup 3-2

1.1 Introduction

This guide is a supporting document for the installation of Zipkin and ELK. You can find the reference in the respective installation guides.

2. Document Tracing Zipkin

2.1 Installation of Zipkin

You can download and run the application to install 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:

The screenshot displays the Zipkin web interface. At the top, there is a navigation bar with links: 'Investigate system behavior', 'Find a trace', 'View Saved Trace', and 'Dependencies'. To the right of these links are two buttons: 'Try Lens UI' and 'Go to trace', followed by a 'Search' button. Below the navigation bar is a search form with several input fields and dropdown menus. The fields are labeled: 'Service Name' (with a dropdown menu showing 'zipkin'), 'Span Name' (with a dropdown menu showing 'all'), 'Remote Service Name' (with a dropdown menu showing 'all'), and 'Lookback' (with a dropdown menu showing '15 minutes'). Below these fields are three more input fields: 'Annotation Query' (with a placeholder text 'For example: http.path=/foo/bar/ and cluster=foo and cache.miss'), 'Duration (µs) >=' (with a placeholder text 'Ex: 100ms or 5s'), and 'Limit' (with a value of '10'). To the right of these fields is a 'Sort' dropdown menu showing 'Longest First'. Below the search form is a blue button labeled 'Find Traces' and a small question mark icon. At the bottom of the form, there is a light blue box with the text 'Please select the criteria for your trace lookup.'

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 is given a different user interface with same functionality.

The screenshot shows a search interface with the following fields and options:

- Service Name:** zipkin
- Span Name:** all
- Remote Service Name:** all
- Lookback:** 1 hour
- Annotation Query:** For example: http.path=/foo/bar/ and cluster=foo and cache.miss
- Duration (μs) >=:** Ex: 100ms or 5s
- Limit:** 10
- Sort:** Longest First
- Find Traces:** (button)
- Showing:** 4 of 4
- Services:** zipkin
- JSON:** (download icon)

The results show four traces, each with a duration, number of spans, and a percentage of successful hits:

- 2.163s 5 spans zipkin 100%** (blue bar) - 18 minutes ago
- 1.449s 4 spans zipkin 100%** (red bar) - 22 minutes ago
- 1.430s 4 spans zipkin 100%** (red bar)

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 screenshot shows the individual trace view with the following details:

- Duration:** 2.163s
- Services:** 1
- Depth:** 3
- Total Spans:** 4
- JSON:** (download icon)
- Expand All** (button)
- Collapse All** (button)
- zipkin x4** (tag)

The trace details are as follows:

Services	2.163s : http/api1	432.639ms	865.278ms	1.298s	1.731s	2.163s
zipkin	-	-	-	-	-	-
zipkin	-	1.001s : api1	-	0	-	-
zipkin	-	-	-	1.068s : http/api2	-	-
zipkin	-	-	-	1.001s : api2	-	-

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 screenshot shows the Zipkin UI interface. On the left, there's a sidebar with 'Investigate system' and a list of services including 'zipkin'. The main area displays details for a specific span. At the top, it shows 'Duration: 2.163s'. Below this, there are buttons for 'Expand All' and 'Collapse All'. The 'Key' and 'Value' table is expanded, showing various HTTP and application-specific details. Below the table, there are 'Show IDs' and 'JSON' download buttons. The 'traceld' and 'spanid' are both '9d63642d72ab6f9f'.

Date Time	Relative Time	Annotation	Address
9/10/2019, 4:11:23 PM		Server Start	10.184.89.16:8080 (zipkin)
9/10/2019, 4:11:25 PM	2.163s	Server Finish	10.184.89.16:8080 (zipkin)

Key	Value
http.host	localhost
http.method	GET
http.path	/api1
http.status_code	200
http.url	http://localhost:8080/api1
mvc.controller.class	Controller
mvc.controller.method	api1
spring.instance_id	eswarperabathini.in.oracle.com:Zipkin

traceld	9d63642d72ab6f9f
spanid	9d63642d72ab6f9f

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:

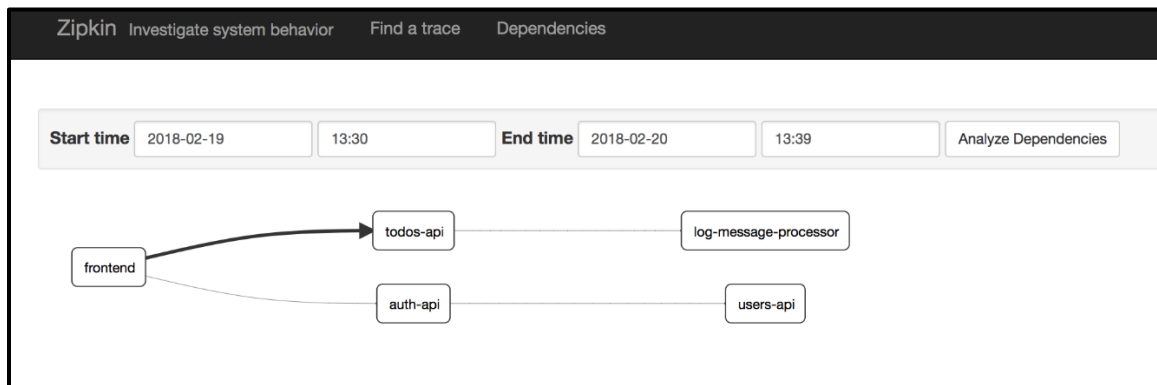
The screenshot shows the Zipkin UI interface displaying a trace. At the top, there's a navigation bar with 'Investigate system behavior', 'Find a trace', 'View Saved Trace', 'Dependencies', 'Try Lens UI', 'Go to trace', and 'Search'. Below this, the trace details are shown: 'Duration: 1.026s', 'Services: 1', 'Depth: 2', and 'Total Spans: 3'. There are buttons for 'Expand All' and 'Collapse All'. The 'Services' list on the left shows 'zipkin'. The main area displays a horizontal timeline with three spans. The first span is labeled 'http/api1' and has a duration of '1.026s'. The second span is labeled 'api1' and has a duration of '1.001s'. The third span is labeled 'api1' and has a duration of '1.001s'. The timeline shows the relative timing of these spans, with the first span starting at '205.134ms' and ending at '410.267ms', the second span starting at '615.401ms' and ending at '820.534ms', and the third span starting at '1.026s' and ending at '1.026s'.

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

Services: zipkin			
Date Time	Relative Time	Annotation	Address
9/11/2019, 6:09:01 PM		Server Start	10.184.89.16:8080 (zipkin)
9/11/2019, 6:09:02 PM	1.026s	Server Finish	10.184.89.16:8080 (zipkin)
Key	Value		
error	Request processing failed; nested exception is org.springframework.web.client.HttpServerErrorException: 500 null		
http.host	localhost		
http.method	GET		
http.path	/api1		
http.status_code	500		
http.url	http://localhost:8080/api1		
mvc.controller.class	BasicErrorController		
mvc.controller.method	errorHtml		
spring.instance_id	eswarperabathini.in.oracle.com:Zipkin		

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 Traceld, 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 **Dependencies** 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 the following open-source products:

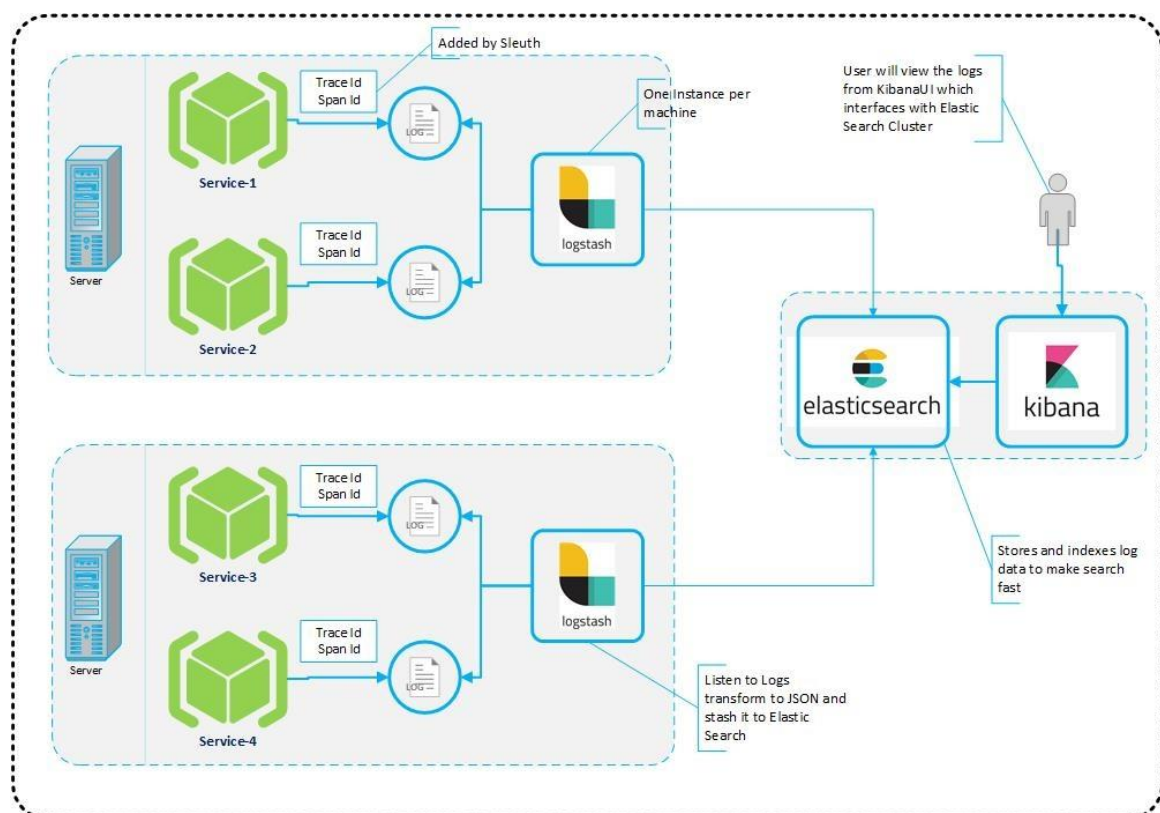
- Elasticsearch
- Logstash
- Kibana

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 for 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. Thus, the 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 the following:

- Logstash
- Elastic Search
- Kibana

The installation guides are given below.

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

Follow the process as given in the following sub-sections, after completing the download process of ELK.

3.3.1 Setup

The setup includes the following steps:

- Start Elastic Search
- Setup Logstash and Start
- Setup Kibana and Start

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 => "^[Y]{YEAR}-[M]{MONTHNUM}-[M]{MONTHDAY} [T]{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:seve
      rity}\\s+\\[%{DATA:service},%{DATA:trace},%{DATA:span},%{DATA:expo
      rtable}\\]\\s+\\[%{DATA:environment}\\]\\s+\\[%{DATA:tenant}\\]\\s+\\[%{D
      ATA: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. Navigate 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:



The screenshot shows the Kibana dashboard interface. On the left is a sidebar with navigation links: Discover, Visualize, Dashboard, Timeline, Dev Tools, and Management. The main area displays a table of log entries. The table has columns for Time, service, environment, tenant, user, branch, trace, span, and message. There are five entries visible, all dated July 11th, 2018, at 13:31:22. The messages are log-level INFO entries from a book-service, detailing operations like finding ratings and fetching book details.

Time	service	environment	tenant	user	branch	trace	span	message
July 11th 2018, 13:31:22.017	book-service	DEV	CITI	Testuser	TestBranch	b65cf8dc98bcae9	b65cf8dc98bcae9	2018-07-11 13:31:22.017 INFO [book-service,b65cf8dc98bcae9,b65cf8dc98bcae9,true] [DEV] [CITI] [Testuser] [TestBranch] 21656 --- [10-8083-exec-10] c.s.c.d.b.BookServiceApplication : Ratings found, set ratings for the given book
July 11th 2018, 13:31:22.017	book-service	DEV	CITI	Testuser	TestBranch	b65cf8dc98bcae9	b65cf8dc98bcae9	2018-07-11 13:31:22.017 INFO [book-service,b65cf8dc98bcae9,b65cf8dc98bcae9,true] [DEV] [CITI] [Testuser] [TestBranch] 21656 --- [10-8083-exec-10] c.s.c.d.b.BookServiceApplication : Returning book details
July 11th 2018, 13:31:22.014	rating-service	DEV	CITI	Testuser	TestBranch	b65cf8dc98bcae9	851c7433a448b30f	2018-07-11 13:31:22.014 INFO [rating-service,b65cf8dc98bcae9,851c7433a448b30f,true] [DEV] [CITI] [Testuser] [TestBranch] 15224 --- [10-8084-exec-7] c.s.c.d.r.RatingServiceApplication : Finding ratings for book id:1
July 11th 2018, 13:31:22.005	book-service	DEV	CITI	Testuser	TestBranch	b65cf8dc98bcae9	b65cf8dc98bcae9	2018-07-11 13:31:22.005 INFO [book-service,b65cf8dc98bcae9,b65cf8dc98bcae9,true] [DEV] [CITI] [Testuser] [TestBranch] 21656 --- [10-8083-exec-10] c.s.c.d.b.BookServiceApplication : Fetching ratings for the book
July 11th 2018, 13:31:22.004	book-service	DEV	CITI	Testuser	TestBranch	b65cf8dc98bcae9	b65cf8dc98bcae9	2018-07-11 13:31:22.004 INFO [book-service,b65cf8dc98bcae9,b65cf8dc98bcae9,true] [DEV] [CITI] [Testuser] [TestBranch] 21656 --- [10-8083-exec-10] c.s.c.d.b.BookServiceApplication : Call to findBook with id:1



ANNEXURE - 2

August 2020

Version 14.4.0.1.0

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

<https://www.oracle.com/industries/financial-services/index.html>

Copyright © 2018-2020, Oracle and/or its affiliates. All rights reserved.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate failsafe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.