Oracle Banking Trade Finance Process Management
Observability User Guide
Release 14.5.3.0.0
Part No. F50590-01
November 2021

ORACLE®
FINANCIAL SERVICES

ORACLE®

Observability User Guide

Oracle Banking Trade Finance Process Management

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

Table of Contents

1. Introduction

1.1 About this Guide

This Observability Guide provides guidance to users to use tools that can enable observe the Oracle Banking Microservices architecture (OBMA) suite of products better. The sections provide tools that can enable a user to:

1. Observe the spans associated in various API calls and the response of each API.
2. Aggregate logs and interpret out of log searches

The Observability guide discusses recommended tools to enhance monitoring and observability aspects of the OBMA products.

1.2 Audience

This guide is intended for the implementation teams.

1.3 Document Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc

1.4 List of Chapters

This manual is organized into the following topics.

	Chapter	Description
1.	Introduction	This chapter that introduces the audience to the contents of the document.
2.	Observability improvements using Zipkin Traces	This chapter explains possible ways and benefits of using tools like Zipkin to enhance troubleshooting possibilities.
3.	Observability improvements Logs using ELK stack	This chapter explains the possible log aggregation and search features that can be availed using ELK stack.
4.	Troubleshooting Kafka issues	This chapter explains the steps to troubleshoot basic issues in Kafka
5.	Troubleshooting Flyway issues	This chapter explains the steps to troubleshoot Flyway issues during deployment.
6.	Acronyms, Abbreviations and Definitions	This provides Acronyms, abbreviations and their definitions

ORACLE®

1.5 <u>Prerequisites</u>

The prerequisites are as follows:

- Basic understanding of Microservice architecture.
- Basic understanding application log analysis using tools.
- Basic understanding DB changes.

1.6 <u>General Prevention</u>

Do not make any changes to Flyway scripts manually.

1.7 <u>Best Practices</u>

The best practices are as follows:

- It is ideal to have ELK stack installed on a separate VM outside the product VMs to ensure flow of logs in case of app crash.
- Log levels can be adjusted to INFO and above to enable relevant logs to flow in.

ORACLE

2. Observability improvements using Zipkin Traces

This section describes the troubleshooting procedures using the Zipkin Traces.

2.1 Setting Zipkin Server

Refer below document for installation document for Zipkin installation

https://docs.oracle.com/cd/F37097_01/PDF/Installation_Guide/ANNEXURE-2.pdf
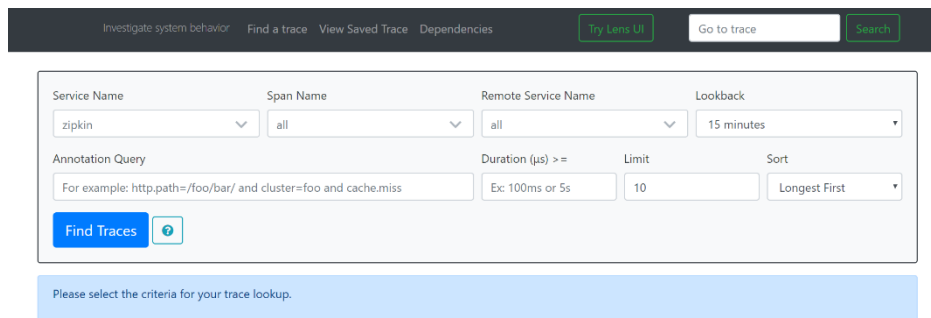
2.2 Login to Zipkin

Perform the following steps for the troubleshooting using Zipkin Traces:

1. Launch the Zipkin URL.

Note: The basic layout of Zipkin is shown in *Figure 1*.

Figure 1: Layout of Zipkin

2. Use the search option to find the traces of required API calls and services.

Note: 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 the same functionality. The list of the traces can be seen as shown in *Figure 2.* 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.

ORACLE®

Figure 2: List of Traces
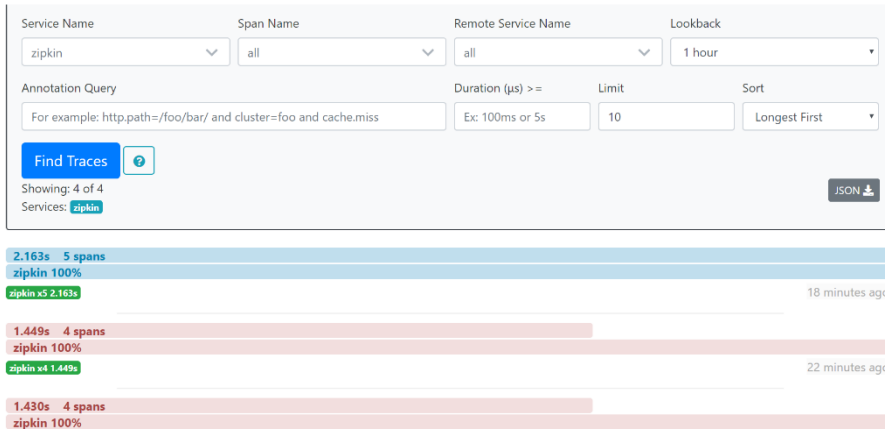
3. Open the individual trace to the details of the trace.

NOTE: *Figure 3* shows an individual trace when it is opened. It also describes the time taken for each block. As the two custom spans are created inside two service calls, you can find a total of four blocks. The time taken for an individual block can be seen in *Figure 3*.

Figure 3: Individual Trace
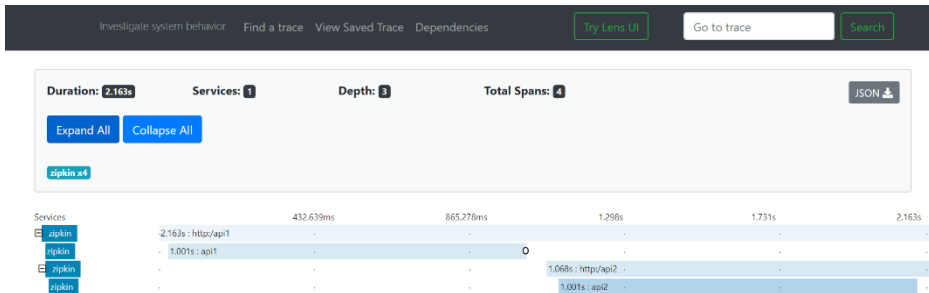
4. Click an individual block to display the details.

Figure 4: Details of Individual Block

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

Show IDs

traceId	9d63642d72ab6f9f
spanId	9d63642d72ab6f9f

NOTE: The details of the specific span block are shown in *Figure 4* and the logging events can also be seen in the Zipkin UI as small circular blocks. An example of an error log is shown in *Figure 5*.

Figure 5: Sample Error Log
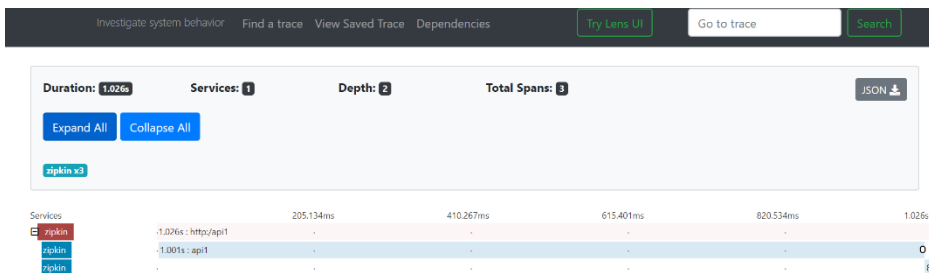
5. Click on the error portion to get a clear detail about the error, and where the error has arisen. An example is shown in *Figure 6*.

Figure 6: Details of Error

Clicking on the error portion gives a clear detail about the error and where the error has arisen. An example is shown below.

ORACLE®

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.Http ServerErrorException: 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

NOTE: If the Lens UI is used in Zipkin, the above Figures are not applicable but are relatable to the Lens UI as well. Traces of the application can be found using TraceId. The TraceId 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).

6. Click **Dependencies** tab to get the dependency graph info between micro-services. An example dependency graph is shown in *Figure 7*.

Figure 7: Sample Dependency Graph
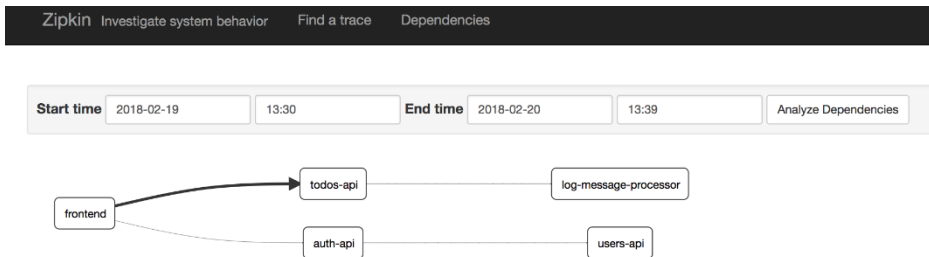
2.3 Zipkin Issues

2.3.1 Application service not registered

Perform the following steps to find the cause of this error:

1. Check the applications, which are sending the trace report to zipkin server from **Service Name** drop-down list.

Figure 8: Find Traces
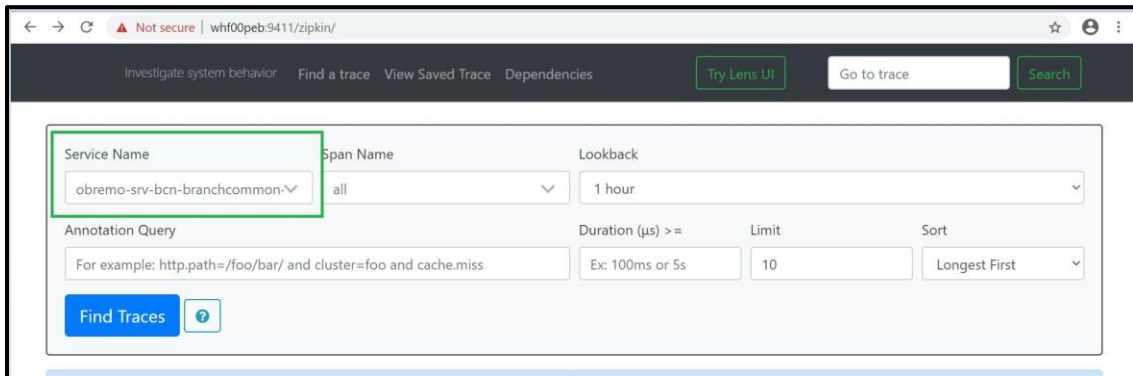
2. If the required application is not listed in Zipkins, check the application.yml file for zipkin base URL configuration. The shipped application.yml should have the zipkin entry.

Figure 9: Application.yml File
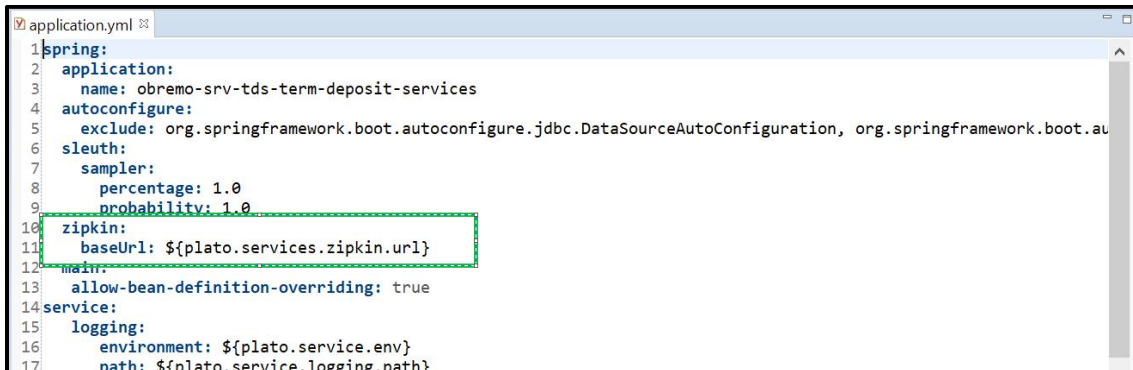
Note: Every service should have spring-cloud-sleuth-zipkin dependency added in build gradle file for the service to generate and send trace Id and span Id.

Compile group: 'org.springframework.cloud', name: 'spring-cloud-sleuth-zipkin', version: '2.1.2.RELEASE'.

Figure 10: Branch Common Services
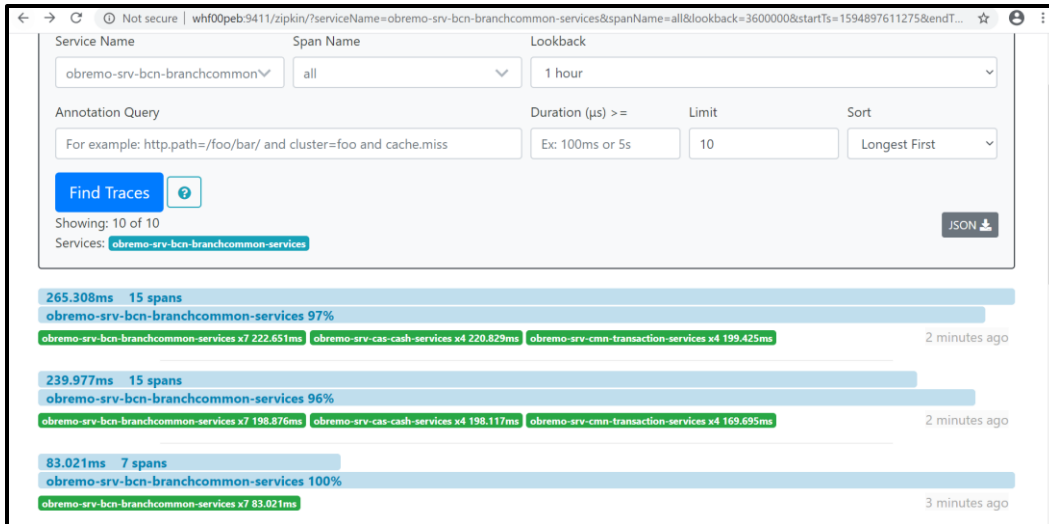
Figure 11: Branch Common Services Trace
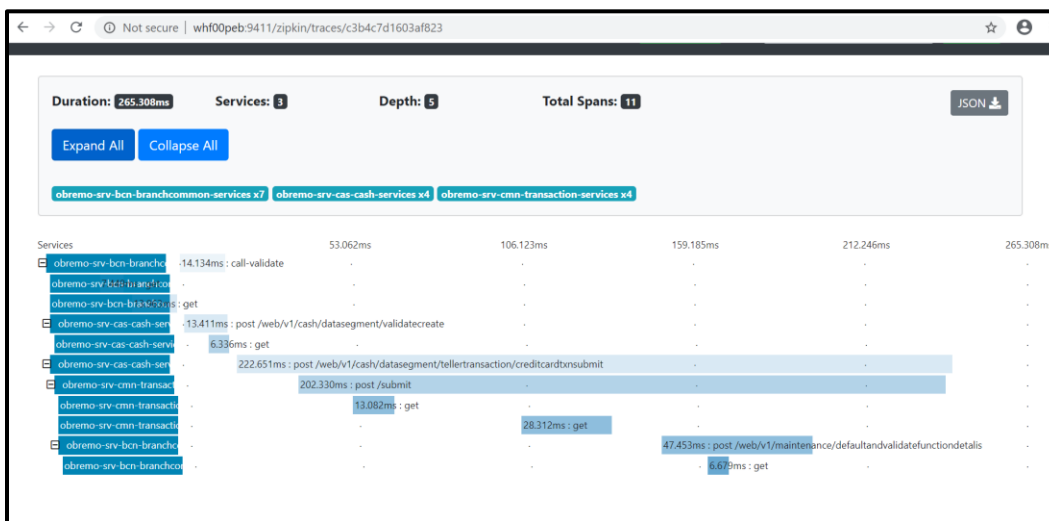
2.3.1.1 404 error

If there is 404 error, check if the **zipkin-server.jar** is running in the system where the application is deployed.

To check this execute the following command

    netstat -ltnup | grep ':9411'

Output should be like:

    tcp6   0   0 :::9411   :::*   LISTEN   10892/java

Here 10892 is the PID.

ORACLE

2.3.1.2 <u>Unable to change zipkin default port number</u>

Zipkin default port number is not editable. Hence, verify that the port 9411 is available to start Zipkin-server.jar file.

ORACLE®

3. Observability improvements Logs using ELK stack

This section describes the troubleshooting procedures using the ELK Stack.

3.1   Setting up ELK

Perform the following steps:

1. Download the Elastic search from https://www.elastic.co/downloads/elasticsearch
2. Kibana download https://www.elastic.co/downloads/kibana
3. Logstash download https://www.elastic.co/downloads/logstash

Figure 12: ELK Setup

```
# Kibana is served by a back end server. This setting specifies the port to use.
#server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: "whf00peb"

# Enables you to specify a path to mount Kibana at if you are running behind a proxy.
# Use the `server.rewriteBasePath` setting to tell Kibana if it should remove the basePath
# from requests it receives, and to prevent a deprecation warning at startup.
# This setting cannot end in a slash.
#server.basePath: ""

# Specifies whether Kibana should rewrite requests that are prefixed with
# `server.basePath` or require that they are rewritten by your reverse proxy.
# This setting was effectively always `false` before Kibana 6.3 and will
# default to `true` starting in Kibana 7.0.
#server.rewriteBasePath: false

# The maximum payload size in bytes for incoming server requests.
#server.maxPayloadBytes: 1048576

# The Kibana server's name.  This is used for display purposes.
#server.name: "your-hostname"

# The URL of the Elasticsearch instance to use for all your queries.
elasticsearch.url: "http://localhost:9200"

# When this setting's value is true Kibana uses the hostname specified in the server.host
```

Default port for Elastic search- 9200 and Default port for Kibana- 5601

3.1.1 Step to run ELK:

Perform the following steps:

1. Run the elasticsearch.sh file present inside /scratch/software/ELK/elasticsearch-6.5.1/bin
2. Configure Kibana to point the running instance of elastic search in kibana.yml file as below
3. Configuring Logstash consists of 3 steps:
a) **Input** - This configuration is required to provide the log file location for the Logstash to read from.
b) **Filter** - Filters in logstash is basically used to control or format the read operation(Line by line or Bulk read)
c) **Output** - In this section we provide the running elastic search instance to send the data for persisting.

ORACLE®

Figure 13: Logstash Configuration

```
input {
  file {
    type => "java"
    path => "/scratch/Software/Weblogic_Installation/user_projects/domains//base_domain/logs/obremo-srv-cmn-transaction-services.log"
    codec => multiline {
      pattern => "Transation Ended!"
      negate => "true"
      what => "next"
    }
  }
}

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"]
    }
  }

}

output {

  stdout {
    codec => rubydebug
  }

  # Sending properly parsed log events to elasticsearch
  elasticsearch {
    hosts => ["localhost:9200"]
  }
}
```

3.1.2 Accessing Kibana

The Kibana can be accessed as shown below:

Figure 14: Accessing Kibana

3.1.3 Steps to setup dynamic log levels in OBMA services without restart

plato-logging-service is dependent on two tables which are to be present in the PLATO schema (JNDI name: jdbc/PLATO). The two tables are as follows:
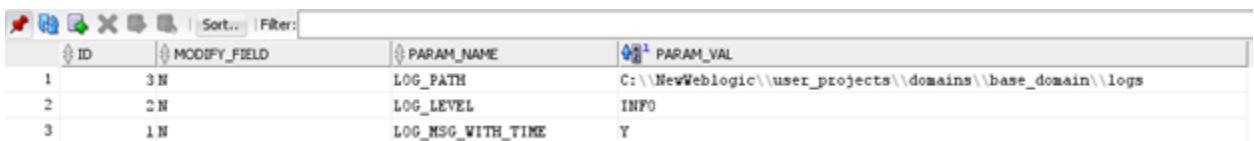
PLATO_DEBUG_USERS: This table will contain the information about whether the dynamic logging will be enabled to a user for a service. The table will contain have records where DEBUG_ENABLED values for a user and a service will have values 'Y' or 'N' and depending on that plato-logger will enable dynamic logging.

Figure 15: PLATO_DEBUG_USERS

	ID	DEBUG_ENABLED	SERVICE_CODE	USER_ID
1	2	Y	plato-logger-ref	soham
2	3	Y	platoref	soham

PLATO_LOGGER_PARAM_CONFIG: This table will contain the key-value entries of different parameters that can be changed at runtime for the dynamic logging. The values that can be passed are as follows:

Figure 16: PLATO_LOGGER_PARAM_CONFIG

	ID	MODIFY_FIELD	PARAM_NAME	PARAM_VAL
1	3	N	LOG_PATH	C:\\NewWeblogic\\user_projects\\domains\\base_domain\\logs
2	2	N	LOG_LEVEL	INFO
3	1	N	LOG_MSG_WITH_TIME	Y

 a) LOG_PATH: This will specify a dynamic logging path for the logging files to be stored. Changing this in runtime will change the location of the log files at runtime. If this value is not passed then by default the LOG_PATH value will be taken from the -D parameter of "plato.service.logging.path"
 b) LOG_LEVEL: The level of the logging can be specified on runtime as "INFO" or "ERROR" etc. The default value of this can be set in the logback.xml.
 c) LOG_MSG_WITH_TIME: Making this 'Y' will append the current date into the logfile name. Setting the value of this as 'N' will not append the current date into the filename.

3.1.4 Searching for logs in Kibana

The URL for searching logs in Kibana is
https://www.elastic.co/guide/en/kibana/current/search.html

3.1.5 How to export logs for tickets

Perform the following steps to export logs:

- Click on the Share button from the top menu bar.
- Select the CSV Reports option.
- Click on the Generate CSV button.

ORACLE

4. Troubleshooting Kafka issues

4.1 Kafka Health

4.1.1 Verifying Kafka Health

Run the below command and verify

$ **netstat -tlnp | grep :9092** (9092 is default port of kafka)

Expected output

4.1.2 Verify Zookeeper health

1. Kafka instance will not start if Zookeeper is not yet started
   Run the below command and verify

$ netstat -tlnp | grep :2181 (2181 is default port of zookeeper)

tcp6	0	0 :::2181	:::*	LISTEN	19936/java

To debug, check if any the permissions of kafka log folder are correct. The log folder path can be found by looking at the value of the property "*log.dirs*" in the *server.properties* file of Kafka installation.

4.2 Prometheus and Grafana

4.2.1 Prometheus Setup

Prometheus is an open-source project which helps monitoring of the applications metircs. It is widely used for the monitoring of Kafka and its metrics. The installer for Prometheus can be downloaded Prometheus from https://prometheus.io/download/.

4.2.2 JMX-Exporter Setup

A JMX-Exporter application is used to integrate with the Kafka broker as a Java agent to expose the values of JMX MBeans as an API. The JMX-Exporter is in turn used by the Prometheus to fetch the values of the JMX metrics.

Download the latest jmx_prometheus_javaagent jar file from the maven repository in the Kafka directory along with the bin, config directories.

This can be used to monitor consumer_lag.

https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.15.0/jmx_prometheus_javaagent-0.15.0.jar

Set the KAFKA_OPTS variable to the desired value to execute the jar as a java agent.

```
export KAFKA_OPTS="$KAFKA_OPTS -
javaagent:$PWD/jmx_prometheus_javaagent-0.15.0.jar=7071:$PWD/kafka-
0-8-2.yml"
```

We can choose the port according to our preference.

Restart Kafka Broker.

4.2.3 <u>Grafana Setup</u>

Perform the following steps:

1. Download Grafana from Grafana website: https://grafana.com/grafana/download in the stand-alone application mode and extract its contents.
2. Go to the bin folder in the extracted contents and start the Grafana server.

Note: Grafana should start on the default port 3000 (HOST:3000). The default user and password for Grafana are admin/admin.

Grafana can be integrated with the Prometheus instance installed above using the below steps:

- Click on the Grafana logo to open the sidebar.
- Click on "Data Sources" in the sidebar.
- Choose "Add New".
- Select "Prometheus" as the data source.
- Click "Add" to test the connection and to save the new data source.

4.2.4 <u>Prometheus Metrics</u>

The Prometheus Metrics are as follows:

- process_cpu_seconds_total
- http_request_duration_seconds
- node_memory_usage_bytes
- http_requests_total
- process_cpu_seconds_total

ORACLE

5. Troubleshooting Flyway issues

This section describes the troubleshooting procedures for the flyway issues.

5.1 <u>Failed Migrations</u>

5.1.1 <u>Success column verification</u>

Perform the following steps for the success column verification:

1. Check the *flyway_schema_history* table to identify the migration record with *success* column as '0'
2. Delete the record with status as '0'
3. Restart deployment

5.1.2 <u>Migration checksum mismatch for a version</u>

Perform the following steps:

- Ensure the flyway script is not manually updated before deployment.
- If yes, then replace with original and restart deployment

5.1.3 <u>Placeholder errors</u>

Pass the placeholder values using setUserOverrides.sh in Weblogic. Alternatively, these issues can be debugged from Weblogic console during deployment, also the application specific logs can be verified for further inputs.

ORACLE