Kibana logging for OKE services

ATG has fluentD daemonsets configured on across all OKE clusters to read the container logs and send to Elastic and to Kibana automatically.

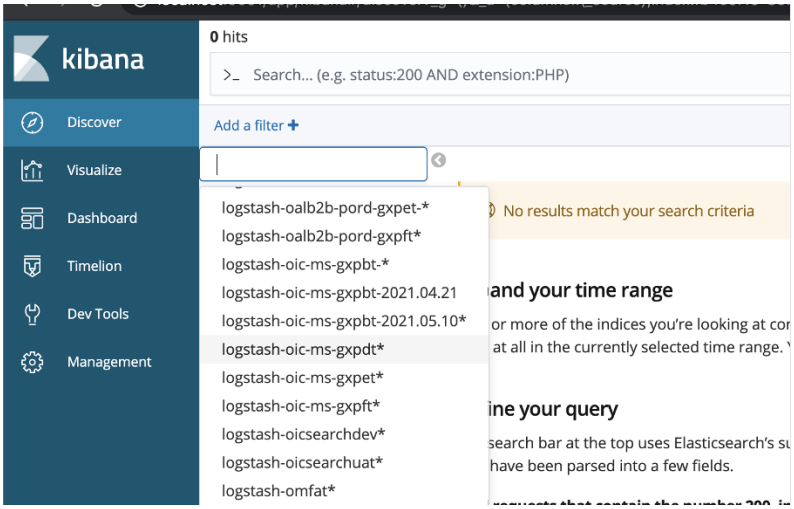
Minimal changes required to activate logging through Kibana

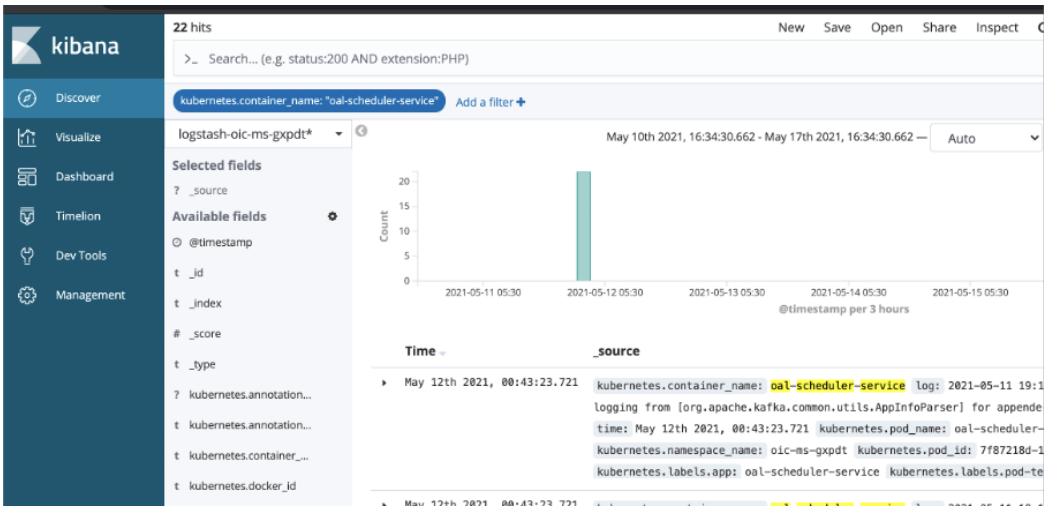
1. Need to log output to console, logging to console using logger appender .
   if you want to send logs only to Kibana, then copy paste below configuration file in log4j2.xml

log4j2.xml only for kibana

```
<Configuration status="info">
  <Appenders>
    <Console name="console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>
    </Console>
  </Appenders>
  <Loggers>
    <Root level="INFO">
      <AppenderRef ref="console" />
    </Root>
  </Loggers>
</Configuration>
```

2. Run below kubectl commands to view logs in Kibana UI
   a. export https_proxy=http://www-proxy-hqdc.us.oracle.com:80
   b. export http_proxy=http://www-proxy-hqdc.us.oracle.com:80
   c. kubectl --kubeconfig Oal-kubeconfig-dev --namespace=oic-ms-gxpbt get pods (this will get all pods from namespace **oic-ms-gxpbt**, **Oal-kubeconfig-dev** is GXP dev config file)
   d. kubectl --kubeconfig Oal-kubeconfig-dev --namespace=logging get pods (this will get all pods for logging, check for kibana pod)
   e. kubectl --kubeconfig Oal-kubeconfig-dev --namespace=oic-ms-gxpet logs -c <<service_name>> <<podname>> (to view logs on console)
      i. example: kubectl --kubeconfig Oal-kubeconfig-dev --namespace=oic-ms-gxpet logs -c oal-admin-service oal-admin-service-8bf97dfcf-6nkpv
3. Now portforward the kibana pod to port for accessing the kibana UI
   a. kubectl --kubeconfig Oal-kubeconfig-dev port-forward <kibana app from get result of command ‘d’> 5601 --namespace=logging
   b. Now hit **http://localhost:5601/** to view Kibana UI
4. filter using indexes(logstash-oic-ms-gxpbt*) and options like Pod name/ log level etc in search .





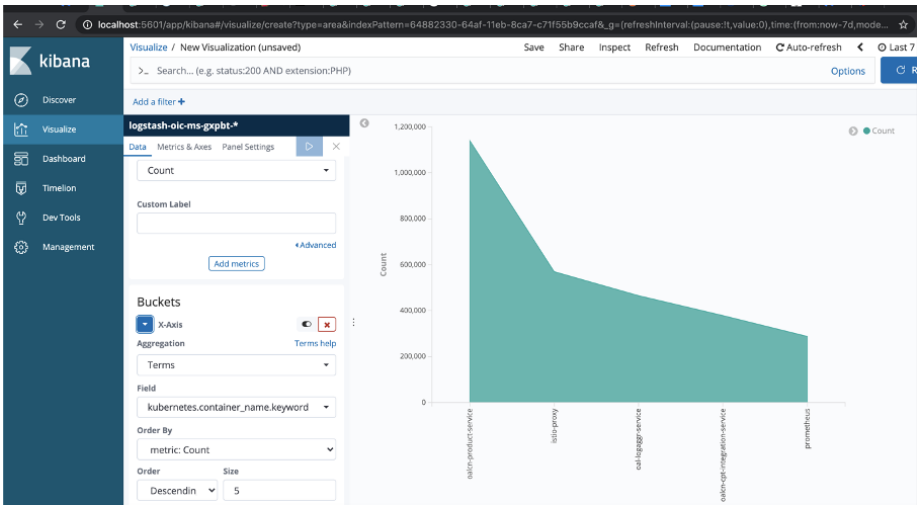
5. **Visualization**

this is used to visualize the data in charts, tables, maps etc. here we can create many templates.

Click on visualize

then '+' and select visualization type (area, heatmap, vertical bars etc)

and then add metrics as per the requirement



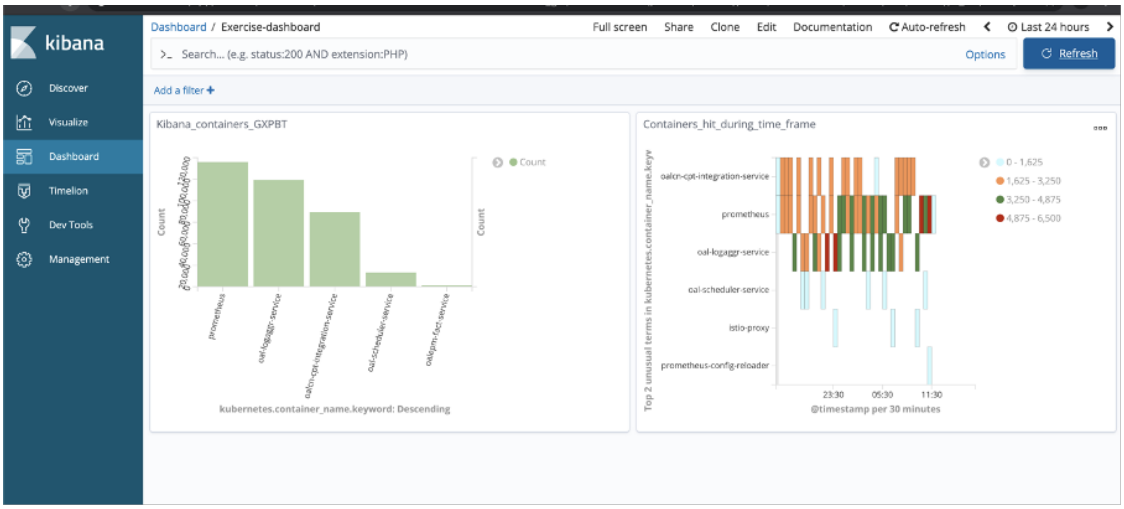
6. **Dashboard**

it is a collection of Visualisation panels. And we can have many panels from different services.

Click dashboard

Create new dashboard

Add



7. Before moving fully to Kibana, want to compare and evaluate existing logging solution with Kibana..? then copy paste below configuration in log4j2.xml. This will allow logs to be available at both Oalcn_log and Kibana.

Log4j2.xml (both oalcn_log and kibana)

```
<Configuration status="info">
  <Appenders>
    <Kafka name="KafkaAppender" topic="${bundle:bootstrap:oal.app.monitor.topic.name}"
      syncSend="false">
      <JSONLayout locationInfo="true" properties="true" charset="ISO-8859-1" compact="true">
        <KeyValuePair key="profile" value="${env:SPRING_PROFILES_ACTIVE}"/>
        <KeyValuePair key="service" value="${bundle:bootstrap:spring.application.name}"/>
        <KeyValuePair key="server" value="${env:NODE_IP}:${env:POD_IP}:${env:POD_NAME}:${env:POD_NAMESPACE}"/>
      </JSONLayout>
      <Property name="bootstrap.servers"
        value="${bundle:bootstrap-${env:SPRING_PROFILES_ACTIVE}:spring.kafka.bootstrap-servers}"/>
      <Property name="client.id"
                value="${bundle:bootstrap:spring.application.name}"/>
    </Kafka>
    <Console name="console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>
    </Console>
  </Appenders>
  <Loggers>
    <Logger name="org.apache.kafka" level="WARN"/> <!-- avoid recursive logging -->
    <Root level="INFO">
      <AppenderRef ref="KafkaAppender"/>
      <AppenderRef ref="console" />
    </Root>
  </Loggers>
</Configuration>
```