Real-time Streaming of Zabbix Metrics to Big Data Platforms
Zabbix Summit 2020
• Relational databases are not the ideal solution for storing metrics at scale
• Timescaledb support in Zabbix since version 4.2 is a huge relief (10x performance improvement)
• What if there are terabytes of metric data to manage?
Who needs to manage terabytes of metric data?

- Best of breed monitoring tools, all integrated
- Zabbix is the main pillar
- Multi tenant
- Single-Sign-On (SSO)
- Everything runs on containers
- CI/CD Orchestration
ClonEYE Platform

{tenant}-monitoring-stack

Zabbix Web Server → Zabbix Postgres → Zabbix Server

Other Application

Another Application

Many containers in each stack

One stack for each tenant

Cluster

Tenant 1 Monitoring Stack
Tenant 2 Monitoring Stack
Tenant 3 Monitoring Stack
Tenant n Monitoring Stack

Zabbix Web Server
Zabbix Postgres
Zabbix Server

Other Application
Another Application

Many containers in each stack

One stack for each tenant
How to store lots of metrics?

- Cost effective
- Large Datasets
- Replication
- Fault Tolerance
- High Availability
- Scalability
- High throughput

- Cloud Native
- Stream Native
- Batch Ingestion
- Time optimized partitioning
- Horizontal Scalibility
- Flexible Schema
- SQL Support
How to import data?

- **Batch Ingestion**
  - Bulk data import
  - Supports many file formats and file sources
  - Task based

- **Streaming**
  - Real-time
  - Druid service ingests directly from streams
Zabbix Real-Time Export

- Zabbix supports exporting to file (which can be batch digested)
- File format is JSON (which is supported by Druid ingestion)
- A new file is created when the file reaches to a size (They should be moved, imported, deleted?)

https://www.zabbix.com/documentation/current/manual/appendix/install/real_time_export
If we can stream zabbix events to kafka, Druid Kafka index service can ingest.

No files to manage!
Zabbix History Kafka Loadable Module

- Found a module in github
- Tested and realized that it is not working
- Tried fixing
- Tried writing a brand new one
- After lots of debugging, realized it is not possible to keep librdkafka state open and reuse for events
- Gave up
Use a REST to kafka middleware

- Kafka REST by Confluent
  https://github.com/confluentinc/kafka-rest

- Kafka Pixy by Mailgun
  https://github.com/mailgun/kafka-pixy
Zabbix History Webhook Module

- Formats and writes Zabbix history to a webhook in zabbix real-time export protocol in JSON (one additional tag: item key)
- Full support for float, integer, string, text and log
- Content type can be specified (default: application/json)
- SSL verification errors can be ignored
Zabbix History Webhook Module

- Ability to call webhook per measurement or multiple measurements (bulk mode)
- Possibility to use custom tags while exporting in bulk mode (needed for kafka rest)
- Environment variables supported (for ease of use with containers)
- Can be used to export to any system that accepts web push

https://github.com/clonera/zabbix-history-webhook
Configuration #1: Zabbix Server

Environment variables to Zabbix Server Container

ZBX_LOADMODULE = "history_webhook.so"
ZBX_WEBHOOK_URL = "http://{kafka-rest}:38082/topics/{tenant}-zabbix"
ZBX_WEBHOOK_CONTENT_TYPE = "application/vnd.kafka.json.v2+json"
ZBX_WEBHOOK_ENABLE_TEXT = "1"
ZBX_WEBHOOK_ENABLE_STRING = "1"

(float and integer is enabled by default we also enable text and string here)

https://github.com/clonera/zabbix-history-webhook
Configuration #2: Kafka Rest

Environment variables to Kafka Rest Container

KAFKA_REST_ZOOKEEPER_CONNECT = "${zookeeper}:2181"
KAFKA_REST_HOST_NAME = "${kafka-host}"
KAFKA_REST_LISTENERS = "http://0.0.0.0:38082"

https://docs.confluent.io/current/kafka-rest/quickstart.html
Configuration #3: Kafka

Environment variables to Kafka Container

- KAFKA_BROKER_ID = 3
- KAFKA_ZOOKEEPER_CONNECT = "${zookeeper}:2181"
- KAFKA_ADVERTISED_HOST_NAME = "${hostname}"
- KAFKA_LISTENERS = "PLAINTEXT://0.0.0.0:9092"
- KAFKA_LOG_RETENTION_HOURS = 48

(9092 port is published on the host in this configuration)
Configuration #4: Druid HDFS

Download druid-hdfs-storage into extension folder and change configuration file:

```
# vi conf/druid/cluster/_common/common.runtime.properties

druid.extensions.loadList=["druid-hdfs-storage", "druid-kafka-indexing-service", "druid-datasketches", "druid-influx-extensions", "postgresql-metadata-storage"]

druid.storage.type=hdfs

druid.storage.storageDirectory=hdfs://hadoop.local:8020/apps/druid/warehouse
```

https://druid.apache.org/docs/latest/development/extensions-core/hdfs.html
Configuration #5: Druid Kafka indexing

Download druid-kafka-indexing-service into extensions folder and change configuration file:

```
# vi conf/druid/cluster/_common/common.runtime.properties

druid.extensions.loadList=
  ["druid-hdfs-storage", "druid-kafka-indexing-service", "druid-datasources", "druid-influx-extensions", "postgresql-metadata-storage"]
```

Configuration #6: Druid ingestion
Configuration #6: Druid ingestion
Configuration #6: Druid ingestion
Configuration #6: Druid ingestion

Druid begins ingesting data once you submit a JSON ingestion spec. If you modify any values in this view, the values entered in previous sections will update accordingly. If you modify any values in previous sections, this spec will automatically update.

Submit the spec to begin loading data into Druid.
Configuration #6: Druid ingestion

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Group ID</th>
<th>Type</th>
<th>Datasource</th>
<th>Location</th>
<th>Created time</th>
<th>Status</th>
<th>Duration</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>index_kafka_clonea-zabbix_6f89594d2015f8_ogochhldg</td>
<td>index_kafka_clonea-zabbix</td>
<td>kafka</td>
<td>clonea-zabbix</td>
<td>10.34.1.234:8105</td>
<td>2020-10-09T07:34:44.790Z</td>
<td>RUNNING</td>
<td>1:01:17</td>
<td></td>
</tr>
<tr>
<td>index_kafka_clonea-zabbix_8dfb1d7558a5bb717_cmmncnpk</td>
<td>index_kafka_clonea-zabbix</td>
<td>kafka</td>
<td>clonea-zabbix</td>
<td>10.34.1.234:8112</td>
<td>2020-10-09T06:32:38.853Z</td>
<td>SUCCESS</td>
<td>1:02:12</td>
<td></td>
</tr>
<tr>
<td>index_kafka_clonea-zabbix_1a9e7f70ac4d40_ea7pabhe</td>
<td>index_kafka_clonea-zabbix</td>
<td>kafka</td>
<td>clonea-zabbix</td>
<td>10.34.1.234:8110</td>
<td>2020-10-09T07:25:34.747Z</td>
<td>SUCCESS</td>
<td>1:02:12</td>
<td></td>
</tr>
<tr>
<td>index_kafka_clonea-zabbix_0e3483242794c0_chcfdlfsa</td>
<td>index_kafka_clonea-zabbix</td>
<td>kafka</td>
<td>clonea-zabbix</td>
<td>10.34.1.234:8110</td>
<td>2020-10-09T04:10:52.562Z</td>
<td>SUCCESS</td>
<td>1:01:14</td>
<td></td>
</tr>
<tr>
<td>index_kafka_clonea-zabbix_8f2b23b1b9e6r6rmmkab</td>
<td>index_kafka_clonea-zabbix</td>
<td>kafka</td>
<td>clonea-zabbix</td>
<td>10.34.1.234:8105</td>
<td>2020-10-09T08:34:54.896Z</td>
<td>SUCCESS</td>
<td>1:01:13</td>
<td></td>
</tr>
<tr>
<td>index_kafka_clonea-zabbix_44f8a31421e6e03_pmpdfilhi</td>
<td>index_kafka_clonea-zabbix</td>
<td>kafka</td>
<td>clonea-zabbix</td>
<td>10.34.1.234:8110</td>
<td>2020-10-09T02:27:28.732Z</td>
<td>SUCCESS</td>
<td>1:01:15</td>
<td></td>
</tr>
<tr>
<td>index_kafka_clonea-zabbix_2a8951f1704a09_hcmcgpe</td>
<td>index_kafka_clonea-zabbix</td>
<td>kafka</td>
<td>clonea-zabbix</td>
<td>10.34.1.234:8103</td>
<td>2020-10-09T02:22:35.375Z</td>
<td>SUCCESS</td>
<td>1:01:12</td>
<td></td>
</tr>
<tr>
<td>index_kafka_clonea-zabbix_436f6f7c75a36e3_nblfodlo</td>
<td>index_kafka_clonea-zabbix</td>
<td>kafka</td>
<td>clonea-zabbix</td>
<td>10.34.1.234:8112</td>
<td>2020-10-09T00:23:52.900Z</td>
<td>SUCCESS</td>
<td>1:01:13</td>
<td></td>
</tr>
</tbody>
</table>
Configuration #6: Druid ingestion
Configuration #7: Superset
Zabbix

Superset
Development Team

- DevOPS Engineer: Doğuş Peynirci
- Software Developer: Burak Köseoğlu
- BI Expert: Rahma Bayhatun
- Software Developer: Nurdan Kolay