Release H: Apache Flink

Introduction

Apache Flink is an open-source, unified stream-processing and batch-processing framework developed by the Apache Software Foundation.
Flink's pipelined runtime system enables the execution of bulk/batch and stream processing programs

Installation

Copy the following files from the Kubernetes setup page:

flink-configuration-configmap.yaml
jobmanager-service.yaml
jobmanager-session-deployment-non-ha.yaml
taskmanager-session-deployment.yaml

Download the following jar from maven central:

flink-sql-connector-kafka-1.17.1.jar

and copy it somewhere that's accessible from your k8s pods.

We can then use kustomize to configure the installation for our own requirements.

kustomization.yaml

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

namespace: kafka

resources:
  - flink-user.yaml
  - flink-configuration-configmap.yaml
  - jobmanager-service.yaml
  - jobmanager-session-deployment-non-ha.yaml
  - taskmanager-session-deployment.yaml

patchesStrategicMerge:
  - patch-jobmanager-env.yml
  - patch-taskmanager-env.yml
  - patch-jobmanager-volumes.yml
  - patch-taskmanager-volumes.yml
  - jobmanager-service-loadbalancer.yaml

replicas:
  - count: 1
    name: flink-taskmanager
```

flink-user.yaml

```
apiVersion: kafka.strimzi.io/v1beta1
kind: KafkaUser
metadata:
  name: flink
  labels:
    strimzi.io/cluster: my-cluster
spec:
  authentication:
    type: tls
```

patch-jobmanager-env.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flink-jobmanager
spec:
  template:
    spec:
      containers:
        - name: jobmanager
          env:
            - name: SSL_KEYSTORE_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: flink
                  key: user.password
            - name: SSL_KEY_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: flink
                  key: user.password
            - name: SSL_TRUSTSTORE_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: my-cluster-cluster-ca-cert
                  key: ca.password
```

patch-taskmanager-env.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flink-taskmanager
spec:
  template:
    spec:
      containers:
        - name: taskmanager
          env:
            - name: SSL_KEYSTORE_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: flink
                  key: user.password
            - name: SSL_KEY_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: flink
                  key: user.password
            - name: SSL_TRUSTSTORE_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: my-cluster-cluster-ca-cert
                  key: ca.password
```

patch-jobmanager-volumes.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flink-jobmanager
spec:
  template:
    spec:
      containers:
        - name: jobmanager
          volumeMounts:
            - name: keystore
              mountPath: "/etc/ssl/keystore"
              readOnly: true
            - name: truststore
              mountPath: "/etc/ssl/truststore"
              readOnly: true
            - name: kafkasql
              mountPath: /opt/flink/lib/flink-sql-connector-kafka-1.17.1.jar
      volumes:
        - name: keystore
          secret:
            secretName: flink
            items:
            - key: user.p12
              path: keystore.p12
        - name: truststore
          secret:
            secretName: my-cluster-cluster-ca-cert
            items:
            - key: ca.p12
              path: truststore.p12
        - name: kafkasql
          hostPath:
             path: /var/flink/lib/flink-sql-connector-kafka-1.17.1.jar
             type: File
```

patch-taskmanager-volumes.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flink-taskmanager
spec:
  template:
    spec:
      containers:
        - name: taskmanager
          volumeMounts:
            - name: keystore
              mountPath: "/etc/ssl/keystore"
              readOnly: true
            - name: truststore
              mountPath: "/etc/ssl/truststore"
              readOnly: true
            - name: kafkasql
              mountPath: /opt/flink/lib/flink-sql-connector-kafka-1.17.1.jar
      volumes:
        - name: keystore
          secret:
            secretName: flink
            items:
            - key: user.p12
              path: keystore.p12
        - name: truststore
          secret:
            secretName: my-cluster-cluster-ca-cert
            items:
            - key: ca.p12
              path: truststore.p12
        - name: kafkasql
          hostPath:
             path: /var/flink/lib/flink-sql-connector-kafka-1.17.1.jar
             type: File
```

jobmanager-service-loadbalancer.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: flink-jobmanager
spec:
  type: LoadBalancer
```

Running kubectl apply -k . will install Flink jobmanager and Flink taskmanager in the kafka namespace.

Flink SQL Client

Find the name of the flink-taskmanager pod in the kafka namespace and log into it

e.g. kubectl exec -it flink-taskmanager-54d79f6dfb-h94h6 -n kafka -- sh

Retrieve the SSL passwords from the environment:

env | grep SSL
SSL_TRUSTSTORE_PASSWORD=m6SgM1gk0OJb
SSL_KEY_PASSWORD=kyXPoimOojcJ
SSL_KEYSTORE_PASSWORD=kyXPoimOojcJ

Start the SQLclient

./bin/sql-client.sh

We can now create the tables/views we need to query our data.

The first one is the pmdata table that puts a structure on our json input from kafka

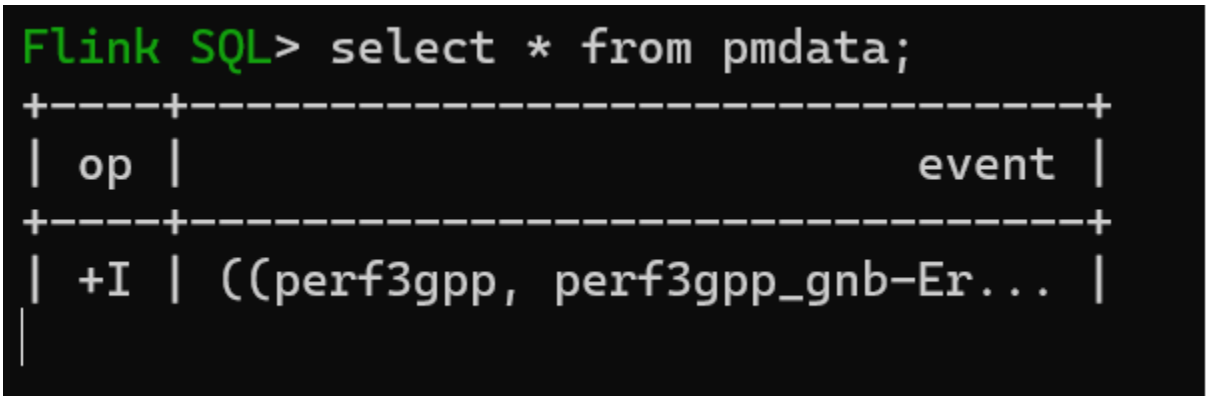
pmdata

```
CREATE TABLE
    pmdata (
       event ROW<commonEventHeader ROW<domain VARCHAR,eventName VARCHAR, sourceName VARCHAR,
reportingEntityName VARCHAR, startEpochMicrosec BIGINT, lastEpochMicrosec BIGINT, timeZoneOffset VARCHAR>,
perf3gppFields ROW<perf3gppFieldsVersion VARCHAR, measDataCollection ROW<granularityPeriod INT,
measuredEntityUserName VARCHAR, measuredEntityDn VARCHAR, measuredEntitySoftwareVersion VARCHAR,
measInfoList ARRAY<
ROW<measInfoId ROW<sMeasInfoId STRING>, measTypes ROW<sMeasTypesList ARRAY<VARCHAR>>,
measValuesList ARRAY<ROW<measObjInstId VARCHAR, suspectFlag VARCHAR, measResults ARRAY<ROW<p int, sValue
STRING>>>
>
>
>>>>
    ) WITH (
    'connector' = 'kafka',
    'topic' = 'pm-readings',
    'scan.startup.mode' = 'earliest-offset',
    'properties.bootstrap.servers' = 'my-cluster-kafka-bootstrap:9093',
    'properties.security.protocol' = 'SSL',
    'properties.ssl.truststore.location' = '/etc/ssl/truststore/truststore.p12',
    'properties.ssl.truststore.type' = 'PKCS12',
    'properties.ssl.truststore.password' = 'm6SgM1gk0OJb',
    'properties.ssl.keystore.location' = '/etc/ssl/keystore/keystore.p12',
    'properties.ssl.keystore.type' = 'PKCS12',
    'properties.ssl.keystore.password' = 'kyXPoimOojcJ',
    'properties.ssl.key.password' = 'kyXPoimOojcJ',
    'properties.group.id' = 'my-group',
    'value.format' = 'json',
    'value.json.fail-on-missing-field' = 'false',
    'value.fields-include' = 'ALL',
    'value.json.ignore-parse-errors' = 'true'
);
```

Note: I'm using the passwords i retrieved from the environment earlier.

Before running any queries we need to set the query results format:

SET 'sql-client.execution.result-mode' = 'tableau';



We need to carry out further transformations to get the data into the format we want.

measValues

```
CREATE TEMPORARY VIEW measValues AS
SELECT perf3gppFieldsVersion, measuredEntityDn, sMeasInfoId, measObjInstId, sMeasTypesList, suspectFlag, p,
sValue from (
SELECT perf3gppFieldsVersion, measuredEntityDn, sMeasInfoId, measObjInstId, suspectFlag, sMeasTypesList,
measResults from(
SELECT perf3gppFieldsVersion, measuredEntityDn, measInfoId.sMeasInfoId, measTypes.sMeasTypesList, measValuesList
FROM pmdata CROSS JOIN UNNEST(measInfoList) AS t (measInfoId, measTypes, measValuesList)
) CROSS JOIN UNNEST(measValuesList) AS t (measObjInstId, suspectFlag, measResults
)
) CROSS JOIN UNNEST(measResults) AS t (p, sValue);
```

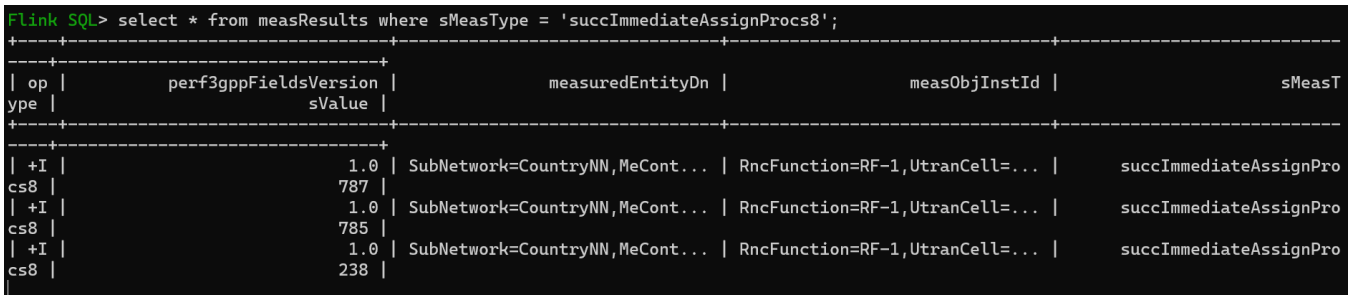
measTypes

```
CREATE TEMPORARY VIEW measTypes AS
select perf3gppFieldsVersion, measuredEntityDn, measObjInstId, sMeasTypesList, sMeasType,
ROW_NUMBER() OVER (PARTITION BY measObjInstId, sMeasTypesList ORDER BY proctime())
as p from (
SELECT perf3gppFieldsVersion, measuredEntityDn, sMeasInfoId, measObjInstId, suspectFlag, sMeasTypesList,
measResults, cardinality(sMeasTypesList) as numMeas from(
SELECT perf3gppFieldsVersion, measuredEntityDn, measInfoId.sMeasInfoId, measTypes.sMeasTypesList, measValuesList
FROM pmdata CROSS JOIN UNNEST(measInfoList) AS t (measInfoId, measTypes, measValuesList)
) CROSS JOIN UNNEST(measValuesList) AS t (measObjInstId, suspectFlag, measResults
)
) CROSS JOIN UNNEST(sMeasTypesList) AS t (sMeasType);
```

measResults

```
CREATE TEMPORARY VIEW measResults AS
select v.perf3gppFieldsVersion, v.measuredEntityDn, v.measObjInstId, t.sMeasType, v.sValue
from measValues v, measTypes t where v.measObjInstId = t.measObjInstId and v.p = t.p and v.sMeasTypesList = t.
sMeasTypesList;
```

We can now run a query against the measResults table: select * from measResults where sMeasType = 'succImmediateAssignProcs8';



```
Flink SQL> select * from measResults where sMeasType = 'succImmediateAssignProcs8';
+----+--------------------------------+--------------------------------+--------------------------------+--------------------------------
----+--------------------------------+
| op |          perf3gppFieldsVersion |              measuredEntityDn |                measObjInstId |                                sMeasT
ype |                         sValue |
+----+--------------------------------+--------------------------------+--------------------------------+--------------------------------
----+--------------------------------+
| +I |                            1.0 | SubNetwork=CountryNN,MeCont... | RncFunction=RF-1,UtranCell=... |      succImmediateAssignPro
cs8 |                            787 |
| +I |                            1.0 | SubNetwork=CountryNN,MeCont... | RncFunction=RF-1,UtranCell=... |      succImmediateAssignPro
cs8 |                            785 |
| +I |                            1.0 | SubNetwork=CountryNN,MeCont... | RncFunction=RF-1,UtranCell=... |      succImmediateAssignPro
cs8 |                            238 |
```

Links

Apache Flink vs Spark

Built-in Functions

Expanding arrays into new rows

How Does Flink Opensource SQL Parse Nested JSON?

Streaming SQL with Apache Flink: A Gentle Introduction

Apache Kafka SQL Connector

SSL Setup

SQL Client

Flink Examples

Flink InfluxDB Connector

JSON SQL functions in Apache Flink