KSQL Onboarding

Anton Sherkhonov <asherkho@redhat.com>, v0.1.0, 2018-Feb-21

Project Goal

Deliverables

The final output of this project will result in a new ingestion and normalization pipeline for the Data Hub which uses KSQL on Kafka as a means of persisting data for the normalizers to process to mitigate data loss introduced when normalizers go down or become unresponsive.

Terminology

- Log ingestion pipeline
    -
- KSQL
    - https://github.com/confluentinc/ksql

Background

Currently, the Data Hub ingestion platform consists of streams of data which flow through normalizers including rsyslog, fluentd, and logstash, before being landed in Elasticsearch.

The current workflow has normalizers both accepting incoming connections and transforming the payloads to the desired format. This architecture is subject to data loss if any of the normalizers goes down or becomes unresponsive and is not easy to maintain since the rules of data transformation are written in 3 separate DSLs (rsyslog, fluentd, logstash). KSQL is introduced in order to mitigate the risk of data loss and provide unified way of lightweight data transformation. The function of existing normalizers will be reduced to accepting connections and forwarding data to Kafka. KSQL will handle all the data transformations and that will happen entirely within Kafka.

Given Kafka can be clustered and configured for high availability, this should mitigate the risk of normalizer failure as a normalizer can be restarted and pick up data processing where it last left off.

There are several technologies already in place which can help with this implementation, including the following:

- Strimzi

    - Existing Red Hat project for containerization of Kafka

Project Details

- Adhere to Guiding Principles and Architecture practices for the AI CoE
    - https://mojo.redhat.com/docs/DOC-1162661
- Operationalize KSQL on top of Kafka in Openshift
    - Create reproducible KSQL deployment on top of OpenShift via Ansible.
    - Write Ansible role to deploy KSQL on OpenShift.
    - Ensure interoperabilty with existing Kafka instance.
    - Collect metrics from KSQL via Prometheus.
- Migrate existing ingestion/normalization processes to KSQL
    - Design and implement rule deployment into KSQL.
    - Port the rules from existing normalizers to KSQL:
        - Rsyslog
        - Fluentd
        - logstash
    - Create automated tests for log ingestion pipeline with KSQL.
- Validate in real environment
    - Deploy KSQL on Upshift, point to existing Kafka
    - Deploy developed rules on this KSQL instance
    - Assuming logs are split from existing rsyslog/fluentd/logstash instance, validate that KSQL rules work properly and the logs have desired formatting.
- Detailed tracking and project plan:
    - https://docs.google.com/document/d/1Ugvqy7Iz0fvevOq_QDiqJaj-pAkCGXfQyUPURIfPOtA/edit?usp=sharing

Project Roles

- Backend/Integration Engineer
    - Familiarity with Openshift, Ansible, Kafka, log normalizers
    - Implement automated deployment via Ansible of KSQL to Openshift
    - Port existing normalizer workflow to Kafka and KSQL workflow
    - Validate deployments and data transfer

Prerequisites for Team Members

- Access to Red Hat network
- Access to Data Hub on the Integration Lab
- Access to Data Hub gitlab repositories
- Access to BlueJeans

References

1. AI CoE Guiding Principles & Architecture Design: https://mojo.redhat.com/docs/DOC-1162661
2. Data Hub on gitlab: https://gitlab.cee.redhat.com/groups/data-hub
3. Data Hub architecture and design: https://mojo.redhat.com/groups/red-hat-artificial-intelligence-center-of-excellence-ai-coe/projects/data-analytics931083

Progress & Status

1. Week 1 (March 15 - 22)
    a. Completed (442 443)
        i. Learned Core concepts of Kafka and had an instance running locally and was able to publish data to a topic.
        ii. Learned Core Concepts of OpenShift and had an instance running locally and was able to deploy a test container on minishift
    b. Issues
        i. Configuring Openshift/minishift for RHEL.
        ii. Getting familiar with Linux environment and setting up the local env.
    c. Blocked
        i. N/A
2. Week 2 (March 22 - 29) - Report
    a. Completed (453 463)
        i. Use python Scripts to publish data to a kafka topic and consume the data.
        ii. Deploy KSQL server to interact with the kafka topic and upload a query.
        iii. Use python script to upload a persistent query to the KSQL topic.
        iv. Write test cases to test the results on test data against known output.
    b. Issues
        i. Test cases: I was not familiar with the testing framework in python and there was considerable back and forth between me and Mr. Anton before I got it correct.
    c. Blocked
        i. N/A

3. Week 3 (March 29 - April 05) - Report
    a. Completed (458)
        i. Write a dockerFile for the KSQL server and publish it to docker hub.
        ii. Deploy KSQL and Kakfa on openShift pods and use python to publish to a topic and upload queries.
        iii. Document the steps and process in depth.
    b. Issues (480)

        i. Unable to retrieve docker image for ksql from the internal registry possibly due to a bug or certificate issues so had to use DockerHub instead.
    c. Blocked
        i. N/A

4. Week 4 (April 05 - April 12) - Report
    a. Completed (459)
        i. Write DeploymentConfig for the ksql docker image (maulikjs/ksqlimage)
        ii. Write Service and Route specs for the ksql container on openshift
        iii. Document the steps and process in depth.
        iv. Run the setup on minishift which is more contained.
        v. Test the system from scripts running on localhost.
    b. Issues
        i. Unable to address the kafka cluster on minishift from localhost.
    c. Blocked
        i. N/A

5. Week 5 (April 12 - April 19) - Report
    a. Completed
        i. Write Dockerfile for the python base image.
        ii. Write buildConfig using s2i strategy to containerize the tests.
        iii. Write Job config for the docker image of the tests.

    b. Issues
        i. The default python docker image in openshift did not have epel and jdk libraries.
        ii. Openshift conflicts on local machine which lead to inconsistent builds.
    c. Blocked
        i. KSQL came out the GA version of ksql client and server which is not backwards compatible with the 0.4 version previously available and does not support one-off queries.

6. Week 6 (April 19 - April 26) - Report
    a. Completed (481)
        i. Deploy strimzi using jenkins
        ii. Wrap up the task from Week 5. There were some minor changes here and there which lead to troubles and we went back and forth before closing it out.

    b. Issues

        i. Upshift was really flaky for some reason and therefore I did ended up wasting a bit of time figuring out where I was going wrong.
    c. Blocked

7. Week 7 (April 26 - May 02) - Report
    a. Completed (481)
        i. Build Jenkins Pipeline to deploy the kafka cluster, ksqlserver and run the tests.
    b. Issues (507)
        i. Currently Jenkins have sleeps in between consecutive deployments so that pods have time to get up and running before the dependent pods are spun up. But this timing is based on my average observed times and we need to add conditions that wait for previous pods to spin up before deploying new pods once we upgrade the ksqlserver and cli.
    c. Blocked
        i. N/A