aiven

Karapace

Hannu Valtonen

Apache Kafka meetup
January 2019 - Aiven HQ Helsinki

Me

- Aiven Co-Founder, VP Product
- Contributor to many Open Source projects
- Background in distributed systems / databases

@HannuValtonen

Which brings us to the beginning of our story

- On October 16th 2018 MongoDB announced that they were changing their license from Open Source AGPL to their own new creation SSPL (Server Side Public License)
- Recent trend among companies that have been using CLAs (Contributor License Agreement) has been to change the license on the whole project or some parts of it to something they think they can better monetize
- Some other firms recently doing this have been for example Timescale and InfluxDB

Which brings us to the beginning of our story

- On December 14th 2018 Confluent announced they were changing the license of certain Open Source components to be proprietary
    - The list of projects changing license included a few Kafka Connect Connectors, Schema Registry, Kafka REST and KSQL
    - Schema Registry being the most significant concern for existing users
- Important to note that change does not affect Apache Kafka itself

Confluent Community License

- Proprietary but has “Open Source”-ish elements
- Allows most people to keep on using the components
  - Explicitly denies the use of the components for any company competing with them
- Only possible in the first place because Confluent required CLA for participants who submitted code to their projects

Our stance

- We believe Open Source is the future and that backing it is the right thing to do
- License changes unlikely to help in competing with the big cloud vendors
    - See recent MongoDB compatible DocumentDB announcement from AWS
- Put even more emphasis of our own to backing Open Source projects
- Don’t require CLAs (Contributor License Agreement) for Open Source projects

Confluent Schema Registry

- Widely deployed around the world in Kafka using environments over the years
- Service to store message schemas (especially when using Avro)
    - Schema is description of a message's contents
        - For example message has two integer fields and two string fields
    - Assigns a surrogate key ID for the Schema contents
- Schema Registry client still under Apache v2 license

Karapace

- Your Kafka essentials in one tool
- Initially 1:1 API compatible reimplementation of Schema Registry using Python 3’s asyncio
- Other new functionality is planned to be added soon like:
    - Ability to produce and consume data over HTTP
    - Consumer group offset fetching over HTTP

https://karapace.io

https://github.com/aiven/karapace

Karapace requirements

- Needs to be backwards compatible so existing customers using the pre-existing schema registry clients will keep on just working
- Needs to be a drop-in replacement on the server-side so users can just start running it within their pre-existing environments without a big flag day operation
- Would be great if it used less memory than Schema Registry (running on Java)
- Wouldn’t hurt to perform better
- Extensible towards other HTTP based functionality
- Open Source License, no CLA required from contributors!

Karapace architecture

- Written on top of Python aiohttp to have asynchronous handling of REST calls
- Master coordinator can join existing Schema Registry groups to determine master (=who is able to write schemas)
- REST API should implement all the APIs used by clients
- Background threads handle reading and writing of schemas to and from Kafka
- More memory efficient

How Kafka producers use Karapace

- By using the existing client libraries for Schema Registry
- First time a client tries to write an AVRO message (assume no caching anywhere)
    - Does a lookup against configured schema registry to see if schema already exists
    - If not create a new one, which assigns an id for the schema
    - Client encodes message in AVRO writing the schema id to be used in decoding the message to the beginning of the encoded message

How Kafka consumers use Karapace

- By using the existing client libraries for Schema Registry
- Consumer reads the message
    - Reads the schema id from the AVRO encoded message
    - Calls to Schema Registry with the id and fetches the schema
    - Then decodes the message based on the retrieved Schema.

Present day

- Version 0.1.0 now out, not recommended for production use just yet
- Implements all the REST APIs of Schema Registry (~2.5k LOC)
  - Has plenty of tests
  - bugs will surely be found (help finding them is welcome!)
- Drop in replacement for Schema Registry on both the server and client sides
- Requires less memory than Confluent's Schema Registry
- Still needs real-word battle hardening

Karapace (schema part) future

- Would be great to not have a synthetic surrogate key for the schema but instead a natural key based on the contents of the schema
  - Would allow portability of the schemas among different Schema Registries
  - No chance of mixups
- Support for other message formats (protobuf)

Karapace (general) future

- REST API for Kafka consumers and producers
  - With a highly performant asynchronous HTTP implementation
- Kafka offset fetching and caching
  - Ever wondered what kind of a consumer lag is there and wanted the result over HTTP?
- Something else? Good ideas welcome!

Q & A

Time to ask me anything

https://karapace.io

https://github.com/aiven/karapace

aiven

Thanks!

(get in touch, we're hiring developers to work on Kafka ecosystem tools)

https://aiven.io

@aiven_io

aiven