VERA C. RUBIN OBSERVATORY

API Design for Rubin's Alert System

Spencer Nelson <swnelson@uw.edu>

NSF AURA U.S. DEPARTMENT OF ENERGY SLAC CHARLES AND LISA SIMONYI FUND FOR ARTS AND SCIENCES LSST CORPORATION

Roadmap for this talk

1. Overview
2. Message Format
3. Kafka Topic Structure
4. Schema Registry
5. Auth
6. Client libraries
7. Integration testing endpoint

Overview

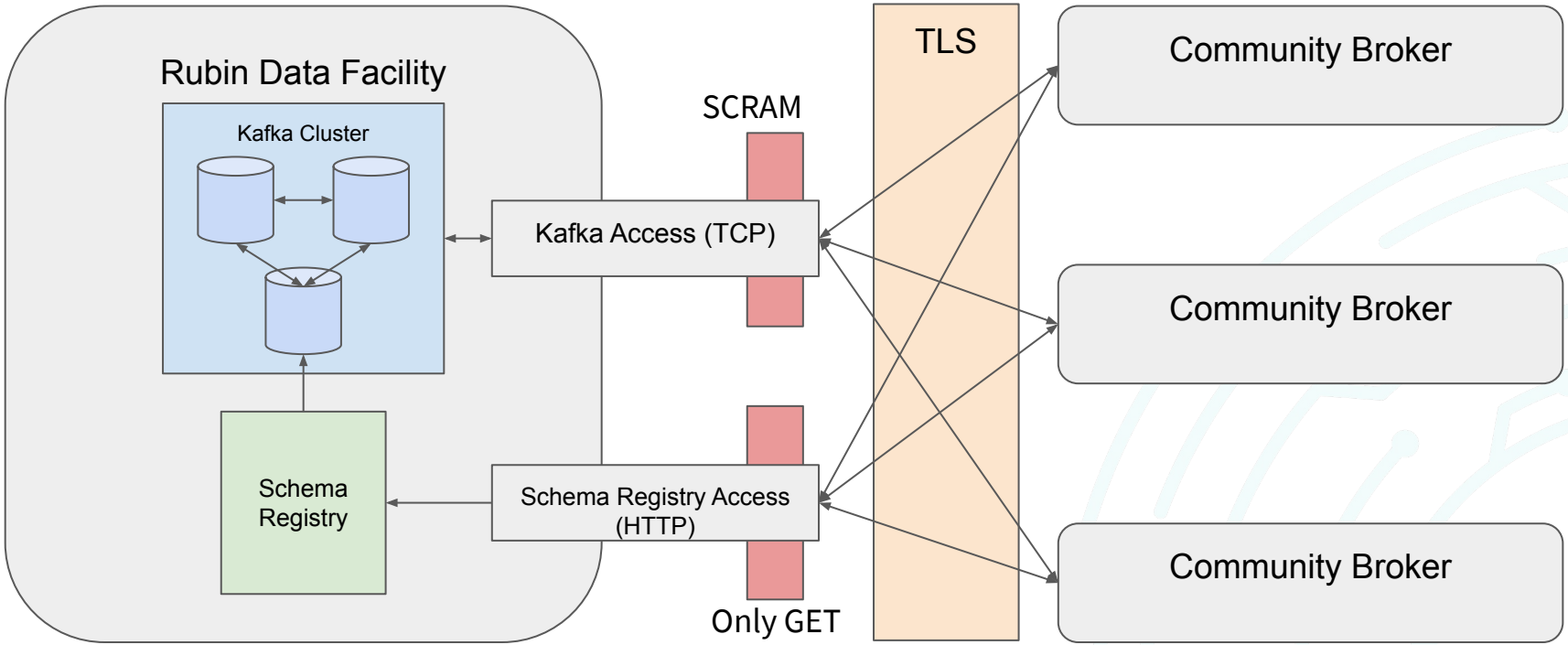
Data will be delivered via Kafka over the internet.

Each alert will be a single Kafka message.

Kafka messages will be encoded using Confluent Wire Format with an Avro binary payload.

A public schema registry will be available in read-only mode.

Overview


Rubin Data Facility
Kafka Cluster
Schema Registry
Kafka Access (TCP)
Schema Registry Access (HTTP)
SCRAM
Only GET
TLS
Community Broker
Community Broker
Community Broker

Roadmap for this talk

1. Overview
2. **Message Format**
3. Kafka Topic Structure
4. Schema Registry
5. Auth
6. Client libraries
7. Integration testing endpoint

Message Format

Alerts will be serialized using Apache Avro into binary payloads.

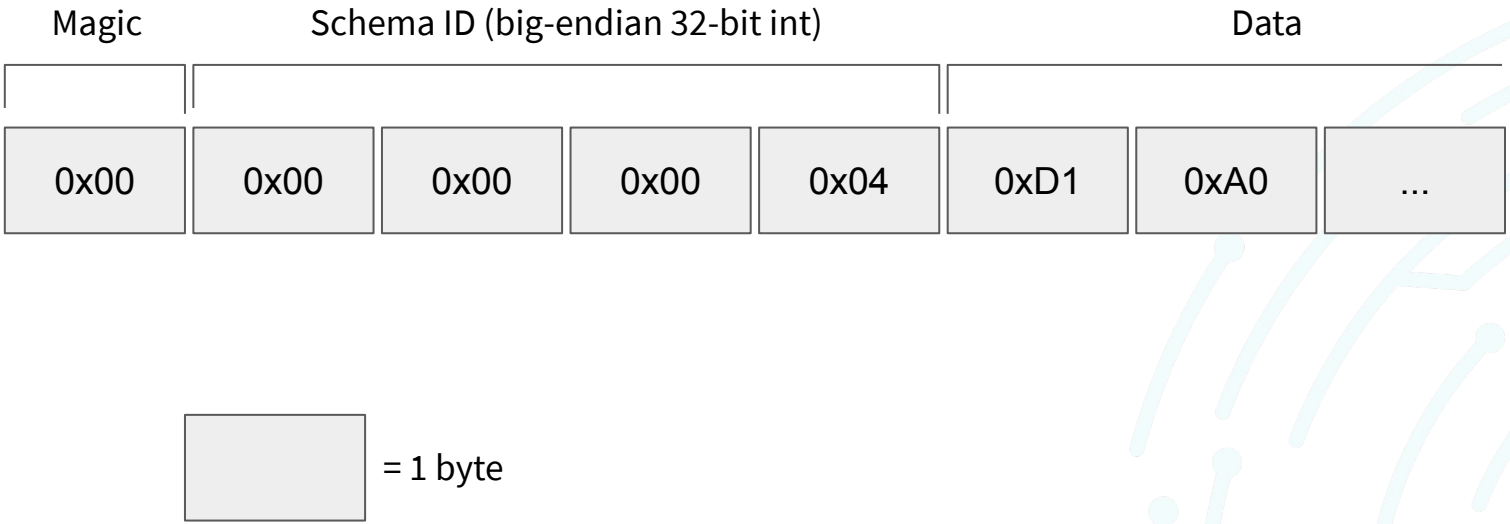
A draft of the alert schema is available here:
https://github.com/lsst/alert_packet/tree/master/python/lsst/alert/packet/schema/4/0

The schema is large (~40kB), so we will not include it in every message; instead we include a reference to the schema used.

Payloads are expected to be 20-80kB.

Message Format

The binary Avro payload will be delivered in Confluent Wire Format
(https://docs.confluent.io/platform/7.0.0/schema-registry/serdes-develop/index.html#wire-format)



Roadmap for this talk

1. Overview
2. Message Format
3. **Kafka Topic Structure**
4. Schema Registry
5. Auth
6. Client libraries
7. Integration testing endpoint

Kafka Topic Structure

Proposing one big topic for all alerts, named "alerts".

Separate topic for schemas and their updates ("registry-schemas" maybe).

And of course, the usual backend Kafka topics (__consumer_offsets).

Data will be retained for at least 7 days in the alerts topic (maybe longer?). We will also store a longer-term archive of the records in case of any disasters.

Roadmap for this talk

1. Overview
2. Message Format
3. Kafka Topic Structure
4. **Schema Registry**
5. Auth
6. Client libraries
7. Integration testing endpoint

Schema Registry

Confluent Schema Registry is an open-source REST service for serving message schemas.

The alert schema will be available by ID - the same ID as in each Kafka message.

GET /schemas/ids/{schema_id} will return the Avro Schema document, which you can use to decode alert messages.

Please cache the schema, don't request it for every message. It won't change very often.

Roadmap for this talk

1. Overview
2. Message Format
3. Kafka Topic Structure
4. Schema Registry
5. **Auth**
6. Client libraries
7. Integration testing endpoint

Auth

Kafka access will require credentials. Rubin will distribute these credentials to brokers.

The auth scheme will be SASL/SCRAM SHA-512 (https://kafka.apache.org/documentation/#security_sasl_scram).

This means you'll get a username and password; these will be generated so they'll act like API tokens.

You will have read access to the "events" and "registry-schemas" topics, and read/write to the "__consumer_offsets" topic.

Auth

Everything will be encrypted under TLS. Our certs will be from some well-known authority, so this shouldn't be noticeable to community brokers.

Schema registry will not require any auth, but it will be read-only.

Auth

Not sure how credential rotation will work, or how community brokers can request multiple credential sets.

This might just be a manual process.

Roadmap for this talk

1. Overview
2. Message Format
3. Kafka Topic Structure
4. Schema Registry
5. Auth
6. **Client libraries**
7. Integration testing endpoint

Client libraries

Rubin does not plan to release a dedicated client library for community brokers to use.

We think that open-source libraries are really good already, and we've been careful to only use industrial technologies (Kafka, Avro, Schema Registry).

You can use Mirrormaker, or Kafka Connect, or Python's confluent_kafka or kafka-python - anything you want.

Collectively, this group has lots of experience with Kafka clients. Let's share our recommendations.

Roadmap for this talk

1. Overview
2. Message Format
3. Kafka Topic Structure
4. Schema Registry
5. Auth
6. Client libraries
7. **Integration testing endpoint**

Integration Testing Endpoint

Currently working on a deployment suitable for integration testing by community brokers.

A static sequence of several thousand alerts will be repeatedly published every 37 seconds.

There will be a Kafka broker, and a schema registry, and the system will require auth.

We hope this helps you get started on deploying and testing brokers.

Integration Testing Endpoint

Working on this now, and hope to have something available in December.

Not live yet, but the plan is:

Kafka bootstrap address:

`alert-broker-int.lsst.cloud:9092`

Schema registry:

https://alert-schemas-int.lsst.cloud

Once we have things set up, we will distribute credentials to all the community broker teams.