Architecture of a Global Scale Alert Hub

2018 CAP Implementation Workshop - Wed 31st Oct
Ian Ibbotson / e:ian.ibbotson@k-int.com / t:@ianibbo
Recent History (How we got started)

- Initially asked to participate and contribute specifically to work on real-time geo-spatial search of alerts and subscriptions.
- History of combining Text and Spatial Search
- Experience with Elasticsearch in a number of contexts
- Experience of microservice and serverless architectures
- Can we provide a subscription matching microservice, which given an alert shape (Polygon or Circle) can return any of the defined subscriptions whose shapes (Polygon or Circle) overlap with the alert.
  - https://s3-eu-west-1.amazonaws.com/alert-hub-subscriptions/json
Elasticsearch - v1

- ES has a really great profile for cluster deployment, scale out and replication. Our target document set (Subscriptions) is not particularly dynamic, and query rates for the app are easily managed by a single node.
- We load the subscriptions
  - https://s3-eu-west-1.amazonaws.com/alert-hub-subscriptions/json
- Into ES and then search for them using the alert shapes
- Normally, ES is searched using URL parameters
Elasticsearch - v1

curl -X GET 'http://localhost:9200/alertssubscriptions/_search' -d '{
  "from": 0,
  "size": 1000,
  "query": {
    "bool": {
      "must": {
        "match_all": {}
      },
      "filter": {
        "geo_shape": {
          "subshape": {
            "shape": {
              "type": "circle",
              "coordinates": [70.06, 12.58],
              "radius": "20.8km"
            },
            "relation": "intersects"
          }
        }
      }
    }
  }
}'
Elasticsearch - v1

- .... And all was well with the world
  - This the spatial search completes in 1ms against the full set of subscriptions
V1 Architecture then (As I recall it)
V1 Retrospective - Overview

- Feed Polling was difficult to get right
- Feed Polling is “Bursty”
- Need to feed-back to feed providers / Error Checking / Reporting / Debugging are critically important - and not terribly easy to debug from cloudwatch logs.
- Event Handling worked OK - sometimes long delays, not clear why
- Again, debugging not so easy
- Polyglot environment is really cute!
- Sub matching seems to work OK
- Publishing seems to block - Debugging proved difficult
V1 - Retrospective - #1 Lesson

- If you aren’t in complete control of your sources, the ability to report on, log and debug your serverless handlers is really crucial.
- This is still a slightly high friction process for a small development team.
- Need diagnostic tools.
V2 - evolution

We’ve evolved three projects

- FeedFacade - [https://github.com/SemwebServices/PubSubHubBubFacade](https://github.com/SemwebServices/PubSubHubBubFacade)
- CAP Collator - [https://github.com/SemwebServices/CAPCollator](https://github.com/SemwebServices/CAPCollator)
- Devops - [https://github.com/SemwebServices/SWCapAlertHubDevops](https://github.com/SemwebServices/SWCapAlertHubDevops)
FeedFacade

- Uses PostgreSQL to coordinate feed fetcher threads
- Capable of running on multiple nodes
- Substantial new structure to describe and track feeds and alerts
  - Each feed can have individual polling intervals
- A lightweight abstraction over RSS and ATOM that converts pull based feeds into a reactive event stream.
- Messaging substrate agnostic, but implemented with RabbitMQ
- Does nothing apart from listen for new “Item” entries in source feeds and emits an event containing that item.
- Supports HTTP HEAD, is the place we implement any special efficiency or behavior to deal with non-standard servers.
FeedFacade
CAPCollator

- Listens out for new events from feedFacade and reacts to them
  - Run many collators, each will remove items from the queue
- If a CAP link is found in an Item emits a CAP event
  - Event picked up by any listening collator
- Listens for CAP events
  - Fetches the source feed, parses and validates
  - Matches subscriptions
  - Emits an event for each matched subscription, Submits an Alert Indexing Record
- Listens for Subscription Match events
  - Updates a static RSS feed and publish to AWS S3 bucket
- Net effect - A loosely coupled service that converts CAP events to an index of alerts tagged by subscription, a static RSS file updated and published on S3.
CAPCollator

Country-ad-lang-en / Official Public alerts in English for Andorra

Subscription Info

- URL: https://alert-feeds.s3.amazonaws.com/country-ad-lang-en.xml
- Filter Type: polygon
- Alert Count: 13 [10 shown]

Map:

Yellow Low temperature Warning issued from for España - Girona

- Alert identifier: 2.4.9.0.0.734.0.4.181009210335.137120330
- Alert Sender: http://www.semweb.com/complexAlertManagement

Times:

- Date of alert: 2018-10-29T10:23:55+00:00
- CC-HTTP-Ost completed: 2018-10-29 10:38:55.389 UTC
- CC-arrays complete: 2018-10-29 10:39:07.183 UTC
- CC-elect CAP event: 2018-10-29 10:39:07.183 UTC

Source: http://fretocolos.m.eu/CAPPOLYEB_29102018_137118930.cap.xml
Devops Project

- Devops project gives us a docker container architecture with all components
  - `docker-compose -f ./docker-compose-dev-setup.yml up`
  - Gives us a system and
  - `http://localhost:9200/alertssubscriptions/_search?q=*`
  - Gives us alters indexed
V2 Architecture

Message Queue / Message Log Substrate

FeedFacade instance

Emit ATOM Event (TOPIC)

Handle ATOM
Handle Cap
Handle Match

Cap Collator

ES

S3

PG
Recent Issues / Current Problems

- Replace static feed writer with a delayed write queue
- Date/Time formats
- Occasional Lag Spikes
Future Developments..

- Try Kafka
- Alert Publishing App to Drive Integration tests and Mock Feeds
- Feed Owner Alerting - Broken Feeds, 404s, invalid polygons, etc.