2023-01-23 Architecture WG Meeting Notes

Date

23 Jan 2023

ZOOM Meeting Information:

Monday, January 23, 2023, at 11:30am PST/2:30pm EST.

Join Zoom Meeting

https://zoom.us/j/7904999331

Meeting ID: 790 499 9331

Attendees:

- Sean Bohan (openIDL)
- Nathan Southern (openIDL)
- Jeff Braswell (openIDL)
- Dale Harris (Travelers)
- Peter Antley (AAIS)
- Mason Wagoner (AAIS)
- Ash Naik (AAIS)
- Tsvetan Georgiev (Senofi)
- Yanko Zhelyazkov (Senofi)
- Faheem Zakaria (Hanover)
- Ken Sayers (AAIS)
- Dhruv Bhatt (

Agenda:

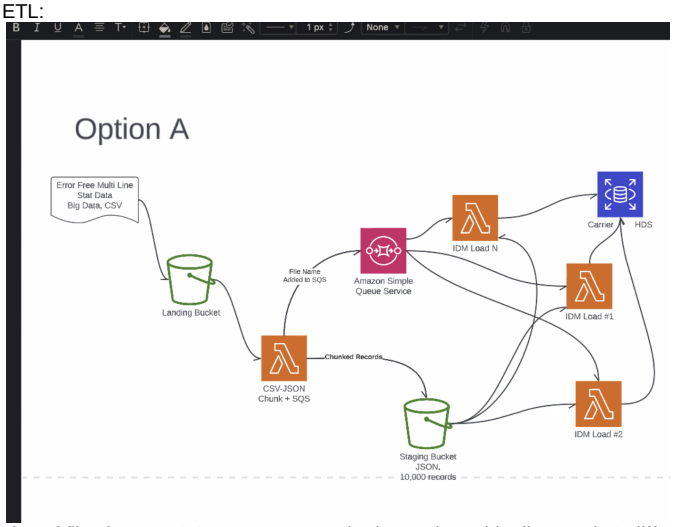
- Update on Large Carrier POC (Ken Sayers)
- Update on openIDL Testnet (Jeff Braswell)
- Update on internal Stat Reporting with openIDL (Peter Antley)
- Update on Infrastructure Working Group (Sean Bohan)

Notes:

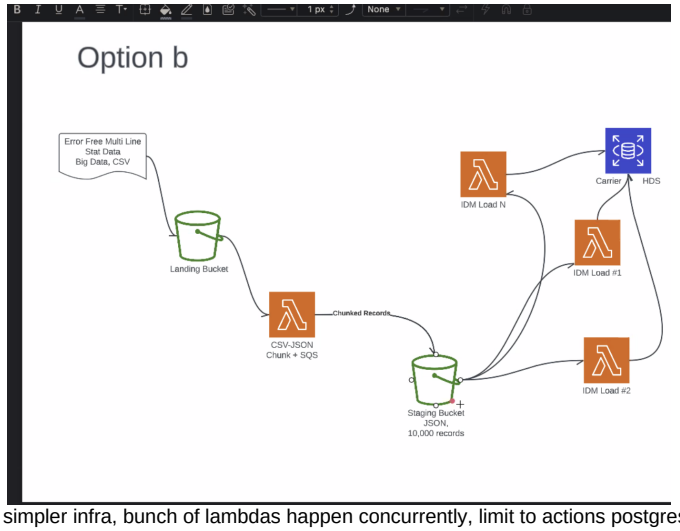
Time	Item	Who	Notes

Documentation:

Notes: (Notes taken live in Requirements document)

- Call for co-chairs of AWG
- AAIS is working with a large carrier on a POC using Crisis Track, starting with setting up nodes in concert with Senofi, working through SOW, peripheral things like report processor, etc. Senofi also exploring use of HL Fabric Operator (improving the Infrastrcture as Code config steps) - will not use Mongo, will use Postgres, reference implementation for ETL, different use case than auto/stat reporting
- HL Fabric Operator (TsvetanG) - want to have integration in place for other projects, synergy with large carrier, lot of expectations out of integrating with Operator (ease of deployment and operation of network) - generic way so it can be run on different cloud providers, outloook to be ready to deploy on others
- Infrastructure as Code up until now has been setting up nodes, provisioning, minimmal UI, for managing and deployment of nodes, resources like route entries for DNS names,
- Started openIDL with IBM, who developed Fabric and Kubernetes - nice UI for managing and looking at Fabric networks, took that and open sourced it - can use IDAs across any cloud provider, easier to deploy, still internal
- 2 things: Postgres and improvement of config steps for the network (big wins out of this POC)
- Anticipate testnet as a place for members to kick tires, desire to support it, opportunity for consistency in ways tools can be deployed, standards for openIDL, great takeaways
- Testnet is up and running, looking for ways to utilize - next up: governance and maintenance
- DMWG update - refactoring on personal auto after looking at personal auto
- Phase 1 of big POC - reproducing catastrophew report used in one state but seems to be a pattern (hurricanes but similar to others)
- (peter screenshare)
- working with stat plans, ingesting data, deep synergies between other use cases
- ETL:


- size of files in ND POC, 1MM rows on the large size - this diagram is 2 diff representations of HDS
- significant discretion for how carriers want to handle things
- starting with error free multi-line stat data, assuming edit package, good data
- Take data, ingest it to landing bucket, go to lambda function takes large files and turns into small files, get large files broken into pieces, from there load into the HDS
- would like small files to be processed in parallel, mult lambdas processing as chunked
- used a queue service in the past, more control over whats happening, and used autoscaling EC2s
- in a lot of ways could do option b



- simpler infra, bunch of lambdas happen concurrently, limit to actions postgres can take, could do multiline csvs
- do we want to use a queue service, help orchestrate the load?
- Volumes? Different data from diff carriers, diagram is single carrier ETL, one carrier would use, 1-10MM rows per month
- not spread out thorughout the month, happen 1x a month (batch)
- May get a 10MM row load
- AWS option - Postgres auroroa? expensive
- Biggest concern, do we care about data integrity, likes Option A over B, SQS should give replay ability over no-queing
- Could lose data now and then
- more config if using a Queueing service
- Queing service allows you to set up as many scale-up as you want
- Lambda is a great consumer for a queing service
- depends on end bucket, get distributed db for that
- can limit concurrency
- not end decision
- queueing can act as a throttle, manage flow, as chunking for each chunk file name goes into queue service
- only queing file name not data
- metadata, not having to put huge files in the queuee
- saving chunks for later use (go into S3 buckets), lose transactional capabilities
- fail? keep metadata to say "loaded/not loaded"?
- in ND, could do Retry
- in ND using a dynamo to track and go thru in stages
- Option C?? - load all the data per line into the queue - is it really costly compared to S3 (discussed a long time ago)
- Pricing - dont want 20MM SQS messages, more resiliency? Pipelines in the past used chunked files
- Did all these chunks suscceed?
- Service monitoring?
- Atomicity in the db - might want it outside single commit, want it across whole original file
- SQS pricing table
- still dealing with "what stage am I completing"? if only adding last two weeks of transactions
- There may need to be times where data can be fixed - if saying "trying to replace whole set or add data to it"
- Seems like need to make sure every command in SQL set all succeded or pulled back
- Peter drawing option C
- dataset needs to be processed consistently or not
- what is the job? regular updates or replace whole file?
- use case
- if any chunks fail all should roll back
- need a control table
- as soon as you chunk or parallel, will need to control if sep ones succeed
- table lives in serverless in AWS? or in HDS
- Should be part of ETL process
- Job ID for every table
- Staging table, take ind things, final push to DB is one transaction
- Staging Schema or base schema?
- Diff instance if staged (whatever postgres calls it) - same metadata schema but not same instance of the db

- Peter to noodle
- append initial metadata per row and then append
- something would need to be sure all are loaded, simpler control table
- looking at each row as a distinct unit by itself
- file might not be the boundary
- still needs job control for completion

Recording: