Another Look At Kafka Topics Per Tenant

Current State: Kafka Topics Per Tenant

- Topic A is needed by a module for proper functionality.
- Every addition of a new tenant to FOLIO will create Topic A for each tenant.
- This is to foster “data security and privacy”.

Why Reconsider?

- Scaling Limitations
- Cost
- Ineffective Data Security

Scaling Limitations

- The first barrier for the number of tenants possible in a FOLIO instance will be kafka partition limits
  - Kafka is currently a small subset of the FOLIO solution space.
- Scalability is limited not by the throughput of a tenant but just its existence.
- A future deployment of a new tenant/topic can render a cluster past its partition threshold. Kafka’s use in FOLIO is still young.
- Numbers In Production:
  - Single Tenant Cluster: 17 Topics, ~117 partitions
  - Multi Tenant Cluster: ~500 Topics, ~2600 partitions
  - Smallest broker type from AWS: 300 partitions
  - Serverless limit from AWS: 120 partitions
  - Serverless limit from Confluent Cloud: 2048 partitions

Cost

- Multiple cloud providers set limits on partition count depending on the broker size requested. A small size broker is not able to handle infinity partitions. Hosting providers would have to select higher tier brokers to compensate for partition count.
  - Multiple cloud providers price based on how many partitions exists. FOLIO’s kafka implementation is not complicated and has a smaller footprint in the FOLIO solution space, so it should not cost an arm and a leg to run Kafka for FOLIO.
- Then just self-host? This is additional costs as well e.g. administration overhead, personnel/knowledge additions, higher cognitive loads for operations teams.
  - FOLIO shouldn’t force all hosting providers to self-host a kafka installation.
- Less tenants on FOLIO instances means more FOLIO instances need to be created to handle same tenant count.
- Costs get passed on to customers thereby limiting FOLIO’s success.

Ineffective Data Security

- Segregating customer data should also include limited access to each segment.
- Security of a system is only as strong as its weakest link.
  - Segregation does not occur on the network layer.
  - Segregation does not occur in module memory.
    - Latest incidents of wrong data reaching a customer occurs on the module tier
    - FOLIO does not recommend module instances per tenant at this time.
  - Segregation occurs in the database with different schemas but one user has access to all schemas.
    - FOLIO doesn’t prescribe database configurations regarding database files’ location and access.
  - Segregation does not occur in database memory
    - FOLIO doesn’t recommend database instances per tenant at this time.

Ineffective Data Security

https://wiki.folio.org/x/YYVFAw

- Incomplete “Temporary” Solution
  - Kafka topics per tenant
  - ~~Dedicated Kafka users per tenant~~
  - ~~ACLs limiting topic access to specific Kafka users~~

Let’s Pretend Temporary Solution Exists

- Need a process flow to create a kafka user when a new tenant is created in okapi, automated or otherwise.
  - Application level constructs(tenant) affecting infrastructure level constructs(kafka user) is not ideal for easy administration and is an avenue of more issues.
- Need a representation of Kafka ACLs maintained by FOLIO developers.
  - Which module can produce/consume on which topic?
  - How do we handle multiple versions of the same module when one tenant is upgraded but another is not?

Let’s Pretend Temporary Solution Exists

- Dedicated Kafka users means that a module must have access to Kafka user credentials for each tenant enabled on the module.
    - FOLIO doesn’t have standard storage for secrets.
- A module needs separate broker connections for each tenant identity. The number of consumers needed will increase significantly. Each consumer will need its own thread to poll Kafka so number of threads could be unsustainable.

New State: Single Kafka Topics NOT By Tenant

- Data in Kafka is mostly transitory, similar to multiple tenant data flowing through a network/Okapi.
- Modules have been trusted to handle all tenant data.
- Database has been trusted to handle all tenant data.
- The proposal is to extend this trust to Kafka to handle all tenant data. Doing away with topics per tenant.
    - Kafka being used as long term storage would alter this proposal.

Open Questions

- How would multiple versions of a module be handled?
- How do we prioritize messages within a single topic?
- Do we need a hybrid model?
- Do we need a Kafka controller to create topics and curate access to topics?