Lyve Cloud Hackathon 2022

Data migration and movement solution from AWS to Lyve Cloud

Solution description of Linh AI team

1. Design consideration

- Robust and efficient data migration from AWS s3 to Lyve Cloud
- Lyve Cloud is s3 compatible ⇒ leverages AWS S3 SDK

  - Boto3: Written in Python
  - AWS-Go-SDK: Written in Golang, https://docs.aws.amazon.com/sdk-for-go/api/service/s3

  → Main activities are upload and download → I/O bounded → **Golang** is a better option (better concurrency) than Python.

- Simple frontend for interaction and visualization like error tracing.

2. Frontend development

- React framework with Antd Design components.
- Three main pages:
  - Migration list
  - Create new migration
  - Show migration activities.

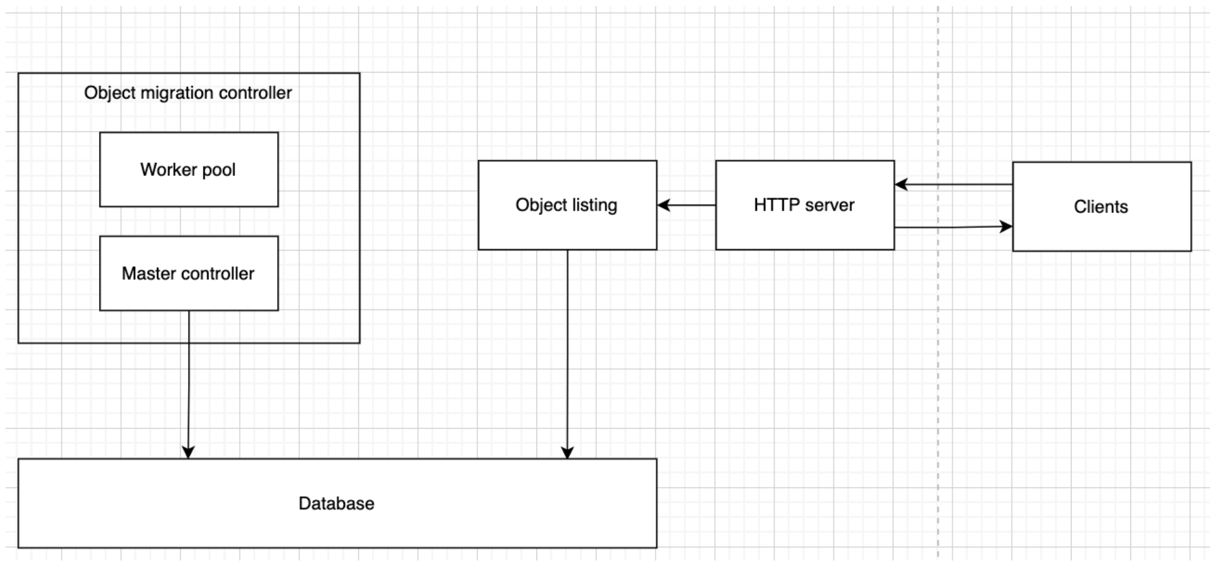
3. Backend development

3.1 Migration strategy

- **Database**: Sqlite is used to record the migration activities and allows recovery
- **AWS operations:**
  - List objects in a bucket
  - Head object: Retrieve object information such as size of the object, content type of the object.
  - Download object: Based on the size of the object, determine the optimal downloading strategy:
    - Small object with size < 10 MB: Directly download the whole object to memory buffer.
    - Large object with size >= 10 MB: Split the object into parts and download each part of the object. This is achieved by the use of **Range** header option in **GetObject** request. Each part of the object is downloaded to memory buffer
- **Lyve Upload:**

    - Upload objects: Based on the object's size, determine the optimal downloading strategy.
        - Small object with size < 10 MB: Upon completing object download to memory buffer, directly upload the object with its metadata, content type to Lyve Cloud.
        - Large object with size >= 10 MB: Create a multipart upload for the object with its metadata, content type and upload each part of the object from memory buffer to Lyve Cloud.
- **Optimization**: As the service leverages memory buffer to hold small objects or part of large objects, there is no disk IO -> higher throughput.

3. 2 System design

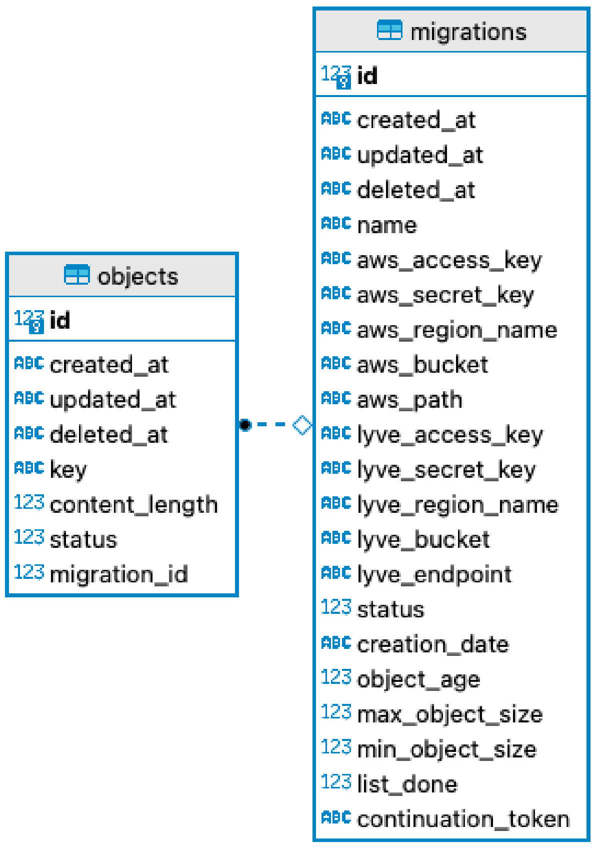


- **Http server:** Golang Gin server to serve requests from clients (https://github.com/gin-gonic/gin). Here is the list of API to be used by client/frontend:
    - GET /api/v1/migrations: List the migrations.
    - GET /api/v1/migrations/<migration_id>: Get the information of a specific migration.
    - POST /api/v1/migrations: Create a new migration.
    - GET /api/v1/migrations/<migration_id>/objects: Get the list of objects under a migration.
    - GET /api/v1/migrations/<migration_id>/logs: Get the error logs of a migration.
- **Object listing:** A Go routine that takes the migration information and continuously fetches the object keys under a bucket. It uses **ListObjectV2** function of the Go SDK to get the keys and save the objects information to

database. This Go routine applies the filter input specified by user to filter satisfied objects: min object size, max object size, object age, creation date

- **Object migration controller:** An independent process that gets the objects from the database and performs migration. It is comprised of two components: worker pool and master controller
    - Worker pool: A pool of Go routines that download objects/object parts from AWS and upload them to Lyve Cloud. We set the number of Go routines to be 20 in our service.
    - Master controller: Receive requests from Go routine worker and query the database to check if there are any available objects to migrate. It also performs the object status update when each object migration is done or error.
    - There are four Go channels for communication between master controller and each worker:
        - Execution channel: Master controller issues migration command for each object to this channel. Worker Go routine receives the command and perform the object/object part migration.
        - OnDone channel: Worker Go routine signals the master controller that an object/object part has been migrated successfully.
        - OnError channel: Worker Go routine signals the master controller that an object/object part has been failed to migrate.
        - OnRequest channel: Worker Go routine requests the master controller for new object/object part migration.

3. 3 Database schema



- The object status is: Not started, In progress, Done, Failed

3. 4 Recovery operation

- Recovery is performed at object level. Upon the program starts, all objects under In Progress status are updated to Not started status.
- The **Object migration controller** continues the operation as normal by querying not started objects and ask the worker pool to migrate the object/object part.

3. 5 Security practices:

- The access key, secret key of Lyve Cloud and AWS services are encrypted with AES encryption algorithm.
- A secret key is generated randomly and saved to environment variable and is used as encryption key. This encryption key is used to encrypt the access key, secret key when saving to the database and decrypt it when it is used for the migration process.