Report 2

Data Lake Architecture EDL

Le Nhu Chu Hiep

1. Introduction

When design Data lake architecture, the first part must be concerned is the storage system. It can be considered as the heart of whole Data Lake. As data warehouse was invented to support OLAP system, the data lake storage should be design to support but big data service and traditional BI system. So the question is "How to structure the Data Lake storage ?". It is schema on read so there are not exit any database or table concept here. So the structure of storage deals with folders and files. Therefore, the designer needs to construct the data lake storage as a normal file system storage but still meets serveral data lake constrain keys.

2. Objective

The key reasons for the need of good data lake structure are:

- Storage Extendibility: The data structure should be easy to extend after first round and more systems can be added
- Data Discovery: It should be easy to use and find the data in the lake and the user should not get lost.
- Data Governance (a set of principles and practices that ensure high quality through the complete lifecycle of your data.): The data lake should be simple enough to apply governance practice on it. The properties of data lake governance includes:
  - Data Quality (the degree to which data is accurate, complete, timely, and consistent with all requirements and business rules): handle mistake of front-end user when they work with data.
  - Data Lineage: tracking any change in data.
  - ILM (Information lifecycle management): rule, policy and tool to manage the data life cycle.
  - Security: need of role-based security on the lake for read access.
- Data Auditing: Same as data governance but come from third party

Above key contrains is used to construct my data lake storage as well as several utility tools to manage data in lake. It provide extendiblity and flexibility for data lake. Moreover, these rule provide several hints in control data life cycle, data discovery and preventing the appear of data swamp. The final data lake architecture should fully satisfies all above key constrains, but at this moment, the simple data lake structure is my highest priority. In the simple data lake system, its only focus on Storage Extendibility, Data discovery and Data Lineage. Then in the further report, the system can expand to accommodate another constrains.

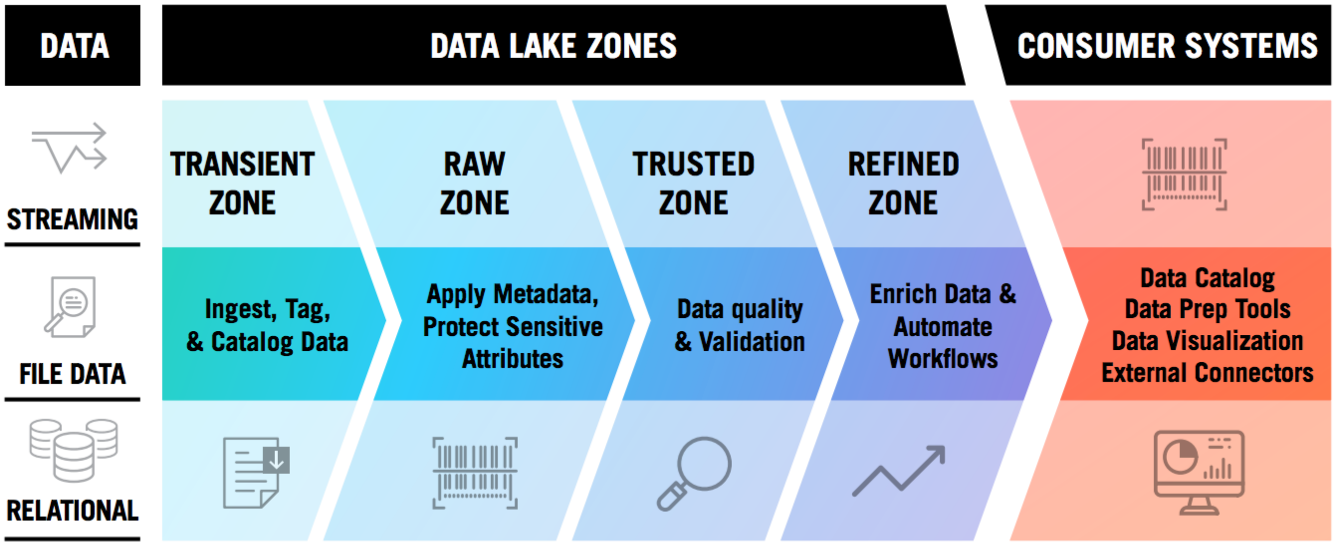
3. Methodology

3.1 Data Pipeline Visualization

Typically, the flow of data in data lake through serveral layers. It can be considered as water which starts from some sources(Iot Device, Warehouse, OLTP, radio station, v.v.) pouring to bottom of lake (data lake). Then this water go through many natural cleaning layer of lake and finally reach the surface of lake where it become pure, clean and nutritious to use. So basically, the lake or data lake works as a pipe captures data into bottom, let it through several clean layout before pushing it into surface where the user can get and use. And this also is the logical idea apllying into my data lake storage architecture. The storage space is splitted to many zones. Each zone represents 1 layer of the lake and describes the data characteristic lay inside its environment. With a generic data lake divide its storage into 4 zones:

1. **Transient Zone** - Used to hold ephemeral data such as temporary copies, streaming spools, and other short-lived data before ingested
2. **Raw Zone** - The zone maintains raw data where the sensitive data is encrypted, tokenized or otherwise secured
3. **Trusted Zone** - the data in zone is result of data quality, validation and other processing, and it is “source of truth” zone for downstream systems
4. **Refined Zone** - Manipulated and enriched data is kept in this zone.

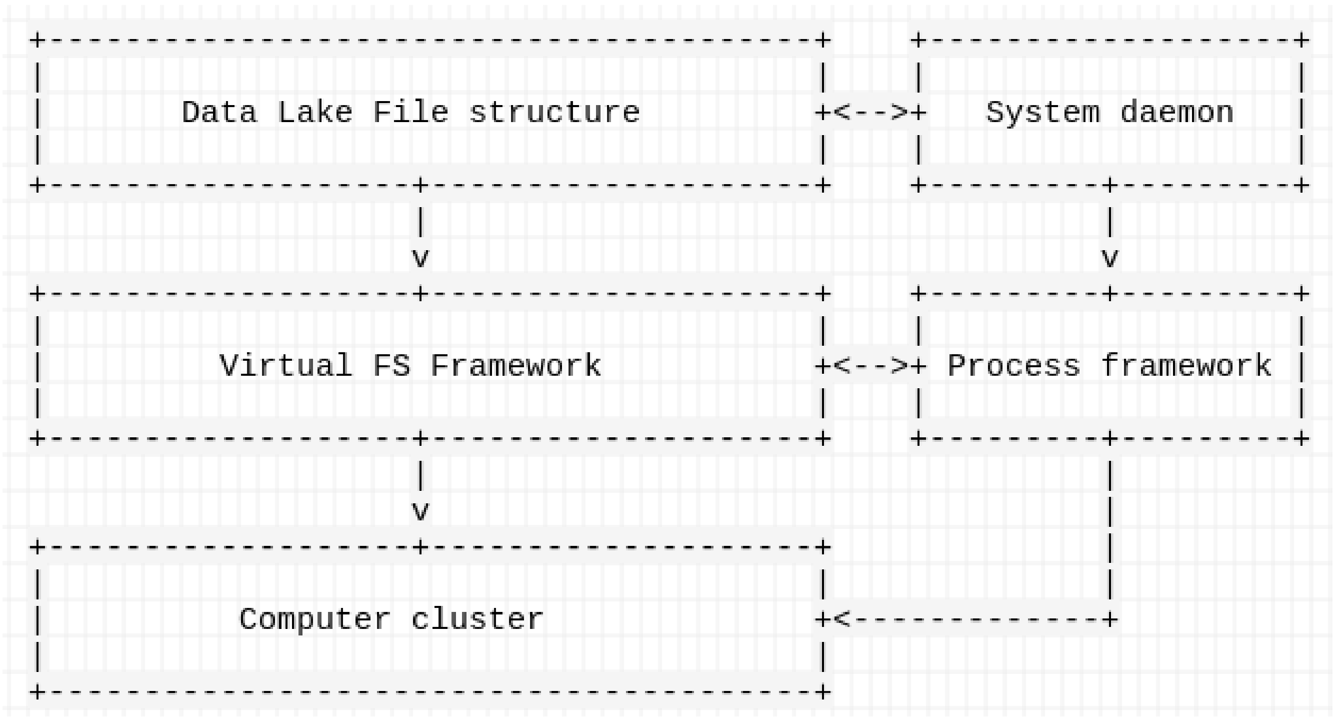
Together, these zone has role as a point in the pipeline leading data from it source ingest to the data lake and finally become insight in cusumer system:



3.2 Data Lake Architecture Design V0.1

3.2.1 Design strategy

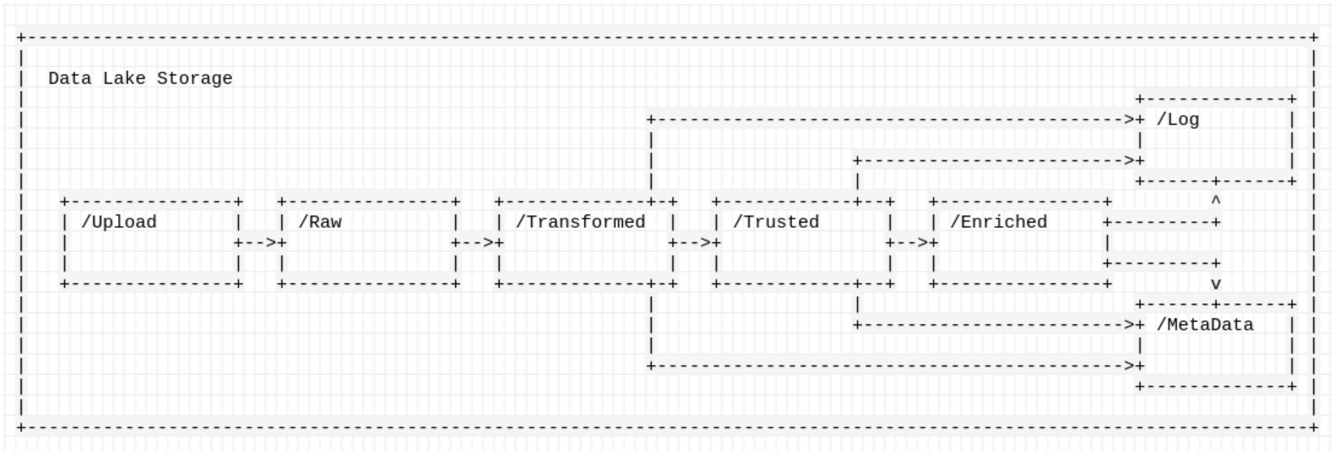
During the design process, there are many factors need to be considered both low and high level architect. In high level or abstract level, the architecture ought to meet architecture constrains. And the architect should support fast implementation in many difference platform and techonology so the simple and clear philosophy is also paid attention to. Unlike top, the bottom design have to adapt to current organize asset. And it must be scalable, handle a large data process as well as flexible enough to integrate new technologys. Here is visualize of the top-down design relationship of data lake:



In here, the data lake system is splited into 2 part: data lake storage and data lake daemon.

While data lake storage provides a centralize repository for persistent data, the daemon visualize the movement of water (data) in lake. Firstly, the lake storage organize copys the eco-system of a real lake that divides into zones as mention in data pipeline. And daemon, getting role as workers, are responsibility to ingest data to lake, do extract, load and transform process, move data from zone to zone then finally push data in lake to analytical tools. So it can be said that the data lake storage and daemon process have a strong relation and must be designed together. Then 2 frameworks (virtual FS and process) is correspoding low level design of 2 parts in abstract architect. They run on top of a small cluster which is in-used system of usth ictlab.

3.2.2 Storage structure



Here is the flow of data in data lake storage. Although in data pipeline, there are only 4 zones, my design add more than 1 zone called "transformed". But in general, the data flow in data lake does not change. Let dive into details of each zone:

1. **Upload -** The temporary zone. The file and relational data ingesting to data lake will be sent to here first. It

is transient zone enable user to modify and config their data before permanent stored. The hierarchy of file is same as the sourse system + addition log file. The data in this zone will be purged before the next load.

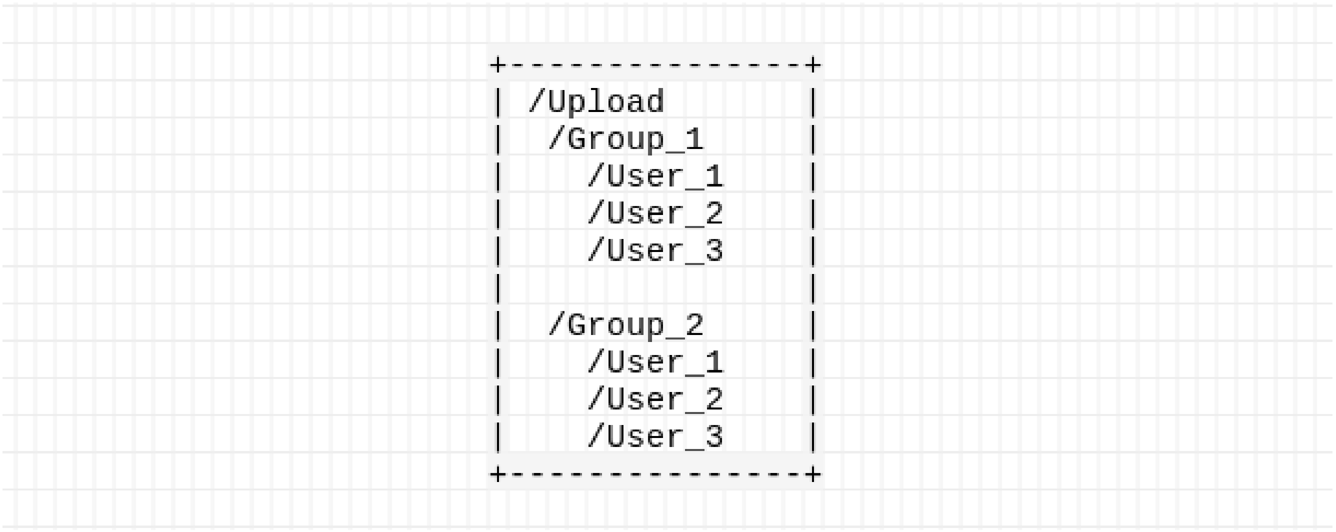
2. **Raw** - a persistent storage. This is complete snapshort of the data captured from source system. The data from the upload zone will be one to one copy to this zone though out error checking process. The data in zone represent the draw input data and provide resiliency to system. Furthermore, Any advance process from transform, quality validate, enrich information will consider this zone as a core data repository.
3. **Transformed** - Normally, the data should be flow through quality validation before reaching the final user. But sometime, the validate toolbox will require a specific format of data. So the reason this extra zone exist for transforming native data into the straightforward format. The zone can be accessed by permitted user and give the view of data before be modified by quality process.
4. **Trusted** - A high quality file zone that can directly feed into the analytic tools.
5. **Enriched** - A scientific zone. This is the free space for any advance big data tool to process data. It provide addition information data for core data that enrich data and metadata.
6. **Log** - A specific space keeps track the whole system log-file. And the data life-cycle log is also stored in here.
7. **MetaData** - A meta space. It provide complete information to describe each data and index of them. The search engine will work with this space to retrive data from the lake. It holds only meta data of Transformed, Trusted and Enriched zones. The raw zone has its own metadata.

Note: There are 2 concepts: **Zone** is the space to store data and its extra information and **Space** is only for metadata and logfile of system.

3.2.3 Zone architect

3.2.3.1 Upload

Upload is modified zones. The fully CRUD operations will be implemented in this zone. The access permission is user and system. And because the upload enalbes user to modify and config their data, each user will has their own dir with name is their identifier (Decided by administrater). The lake supports the group user so the group user will have a group dir identified by their group id (set by administrater). And if the user belongs to the group user, its dir will be sub-dir of the group dir. The lake does not allow alone user in this moment.

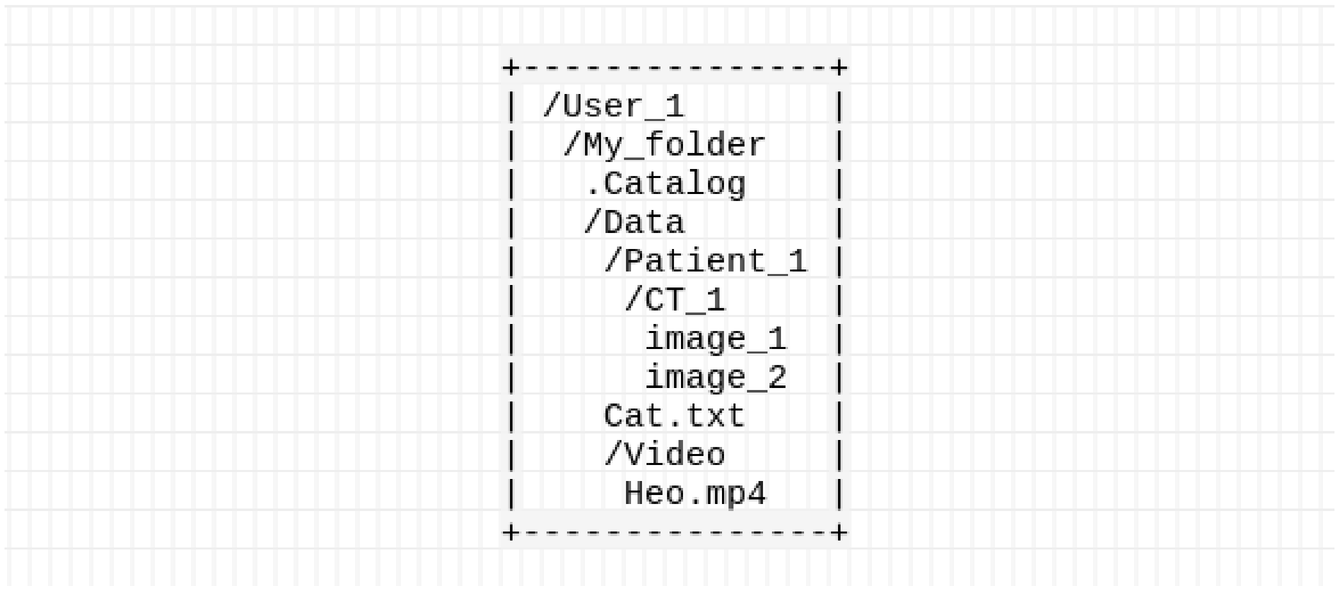


```
+---------------+
| /Upload       |
|  /Group_1     |
|    /User_1    |
|    /User_2    |
|    /User_3    |
|               |
|  /Group_2     |
|    /User_1    |
|    /User_2    |
|    /User_3    |
+---------------+
```

Data lifecycle in zone:

Whenever the user upload new data, it will be store in user's directory. The data is organized in a dir has a unique name set by user. In that dir will has 1 sub-dir with name "/Data" and a catalog file called ".Catalog". The sub-dir

"/Data" holds the input data of source system. It can be any type and and struct but must be followed the unix file system rule. Below is example of a user dir:



```
+---------------+
| /User_1       |
|  /My_folder   |
|   .Catalog    |
|   /Data       |
|    /Patient_1 |
|     /CT_1     |
|      image_1  |
|      image_2  |
|    Cat.txt    |
|    /Video     |
|     Heo.mp4   |
+---------------+
```

Catalog File:

The file specify metadata to all the data in "/Data" dir. It charateristic include: - Type-table: whole type of every file in "/data" will be kept and indexed. - Catalog-table: Specify any user catalog to each file in "/data". The struct same with Type-table. - Security: Include user and group permission. - Path-to-file: Table show the path to each file in "/Data" and its properties (index of type, catalog and security)

The file organized as NoSQL. Preference Json and XML struct or their mixed. Example in XML:

```
<Type-table>
    <0>jpeg</0>
    <1>mp4</1>
    <2>wav</2>
</Type-table>
<Catalog-table>
    <0>Patient_1</0>
    <1>Patinet_2</0>
    <2>Ct-Scan</2>
    <3>Entertainment</3>
</Catalog-table>
<Path-to-file>
    <0>
        <File-path>"/Data/Patient_1/CT_1/*"</File-path>
        <Type>0</Type>
        <Catalog>"0 2"</Catalog>
        <Security>"-477"</Security>
    </0>
    <1>
        <File-path>"/Data/Video/Heo.mp4"</File-path>
        <Type>1</Type>
        <Catalog>"3"</Catalog>
        <Security>"-700"</Security>
    </1>
<Path-to-file>
```

Explain - For Catalog property in "Path-to-file", the value is a string which includes index in Catalog-table with a space delineates each index. Moreover, the Security is a string having 4 values expresses the permission and type of file as in normal unix file system. Finally, for the path to file, the root is dir holds ".Catalog" file and there are 2 types of declare. The single file can be declare as normal - "/Path/To/File". The second type is group declare, multiple file can share same properties and security and must lay in same dir and be declared as - "/Path/To/Group/*" - where "Group" dir is parent of whole share properties and security files.

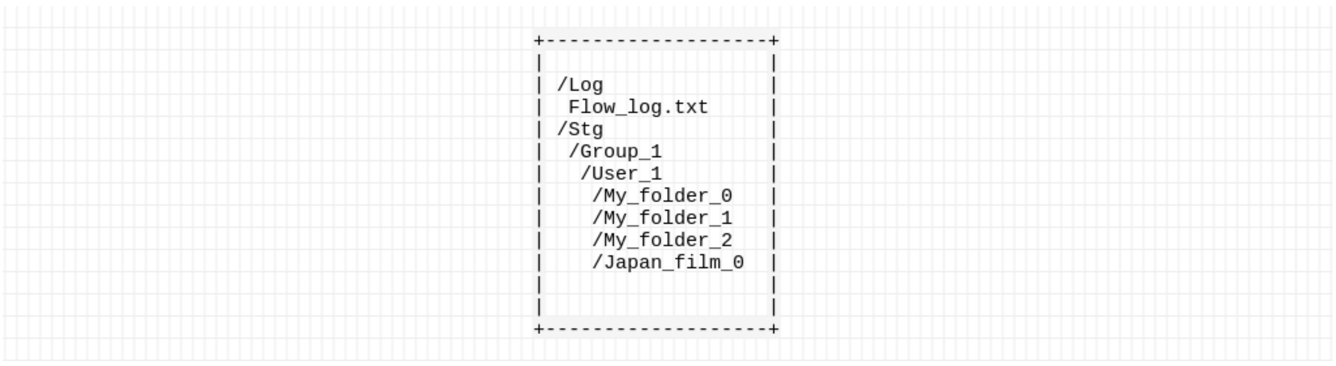
3.2.3.2 Raw

Raw is real storage repository to store input data. The data flow into this zone permanently stay in here until delete process happen. Therefore, the data modify is prohibited, the data in here must be a perfect represent of data input. Before the data reach this zone, multiple CHECK and TEST process run on data in "upload" zone to validate the rule and catalog of data. If the error happen, the lake automate fix it if possible or not, the warning will be responsed back to the user.

Raw zone struct include 2 path: "/Stg" and "/Log". The "/Stg" directory used to store data and zone keep its own log-file to keep track lineage of data in zone. Since the "/Stg" dir is write and del only, the "/Log" dir is fully CRUD implementation. It will be updated frequency whenever a data come and out. Since the data is saved permanently in the "/Stg", each data directory name must be unique. But the user can not handle them in big data. So the lake will automatic append number to data dir name. If the system detects a repeated name, new number will be assignted to the data dir name.

The "/Log" dir keeps track movement of data. It provide space for any extra process on Raw zone. But for now, It is only for tracking the data input and output. It provide list of data dir that fed to zone and is not loaded to next zone. So the dir contains only a txt file called "Flow_log.txt".

The zone visualization will be:



```
+--------------------+
|                    |
| /Log               |
|  Flow_log.txt      |
| /Stg               |
|  /Group_1          |
|   /User_1          |
|    /My_folder_0    |
|    /My_folder_1    |
|    /My_folder_2    |
|    /Japan_film_0   |
|                    |
|                    |
+--------------------+
```

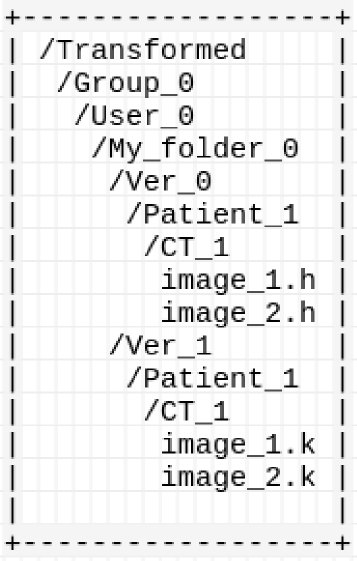
Finally the data will be stored in "/Raw/Stg" zone with same hierarchy and struct as in "/Upload" zone. Except a special line will be added to the bottom of ".Catalog" file represent the time that data was saved to zone, anything else will not change. Basically, new ".Catalog" file of previous example, now, will like:

```
<Type-table>
    <0>jpeg</0>
    <1>mp4</1>
    <2>wav</2>
</Type-table>
<Catalog-table>
    <0>Patient_1</0>
    <1>Patinet_2</0>
    <2>Ct-Scan</2>
    <3>Entertainment</3>
</Catalog-table>
<Path-to-file>
    <0>
        <File-path>"/Data/Patient_1/CT_1/*"</File-path>
        <Type>0</Type>
        <Catalog>"0 2"</Catalog>
        <Security>"-477"</Security>
    </0>
    <1>
        <File-path>"/Data/Video/Heo.mp4"</File-path>
        <Type>1</Type>
        <Catalog>"3"</Catalog>
        <Security>"-700"</Security>
    </1>
<Path-to-file>

###########################
# The line automate by lake
###########################
C_time: 2020-05-16 11:45:40
```

3.2.3.3 Transformed

The zone holds the unique data format. Normally, the data format is diverse and hard to control. For example, the image is a popular type data, and the format of image can be jpeg, png or pdf. And the different format data has different struct and requires a different load process. So it cause trouble with analytic tools since they support a few format file only. Therefore, the lake simplify this issue by providing an auto process to convert all file having same properties and type to a unique format. And the result of this process will be stored in transformed zones. But each format type has its pros and cons, so the lake support multiple format version for each data. Typically, the lake can not decide by itself, it depends on "format__map__table" file to process. For the zone struct, it shares same hierarchy with Raw zone. But in each data, instead of "/Data" sub-dir and ".Catalog" file, it contain multiple versions dir. The version dir can be considered as a copy of Data dir in data folder but all type in version dir is reformated to unique format for each type.



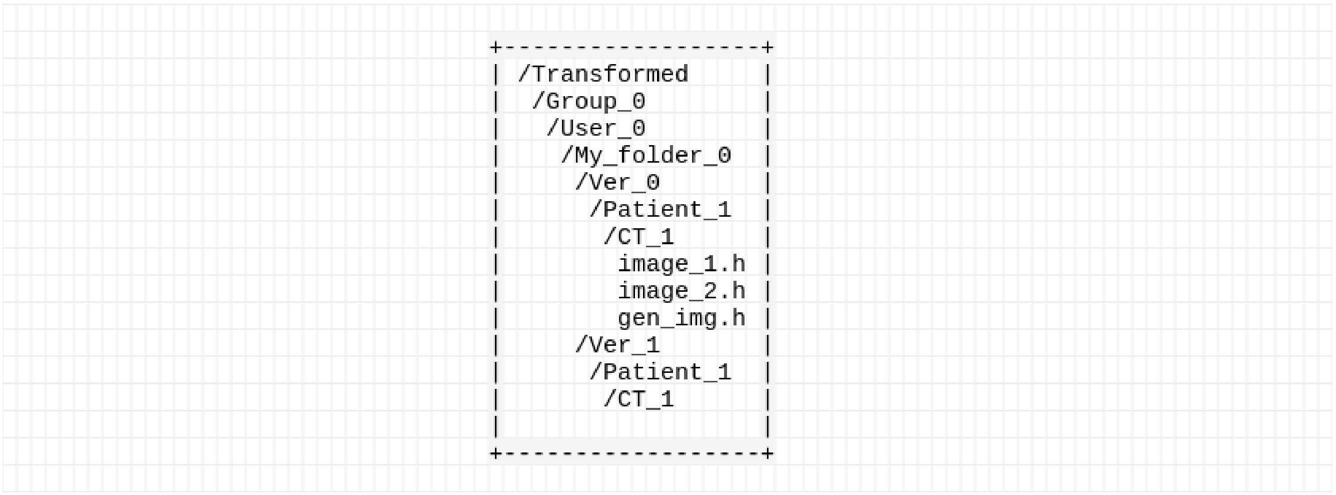
```
+-------------------+
| /Transformed      |
|  /Group_0         |
|   /User_0         |
|    /My_folder_0   |
|     /Ver_0        |
|      /Patient_1   |
|       /CT_1       |
|        image_1.h  |
|        image_2.h  |
|     /Ver_1        |
|      /Patient_1   |
|       /CT_1       |
|        image_1.k  |
|        image_2.k  |
|                   |
+-------------------+
```

The "format__map__table" file is data level. That mean each data is pointed to by a "format__map__table" file. This file synchronize to struct of data in transformed zone. So when any change in this file will be propagate back to the data struct. The back direction is prohibited. The struct reconstruct process must be through out this file.

3.2.3.4 Trusted

This zone represent the final quality data that be fed into the analytic tools or BI tools. The lake enables several quality improve and cleanse processes to the data like: auto remove NaN value, padd value, enhancement image, remove sound noise, v.v.. Since Trusted zone is enhance version of Transformed zone with quality validate process, the struct of this zone is similar to Transformed zone. The process validation process heavily depends on the data catalog, type and format. So the client must choose out the specific process to apply into their data. In common case, the user config about validation tools can be stored in a config file and be called out by the validation processes. And because Trusted zone strongly dependent on Transformed zone, any refactor in Transformed zone will propagate directly to Trusted zone, then the "format_map_table" file can be reuse to store user configuration also.

Note: Although Trusted zone reconstruct data, it is supposed to keep the same hierarchy as in Transformed zone. That mean if any file is deleted, its parent directory is still be kept or the new file is created form a specific file, the new file will be saved in same parent directory as the original file.



```
+------------------+
| /Transformed     |
|  /Group_0        |
|   /User_0        |
|    /My_folder_0  |
|     /Ver_0       |
|      /Patient_1  |
|       /CT_1      |
|        image_1.h |
|        image_2.h |
|        gen_img.h |
|     /Ver_1       |
|      /Patient_1  |
|       /CT_1      |
|                  |
+------------------+
```

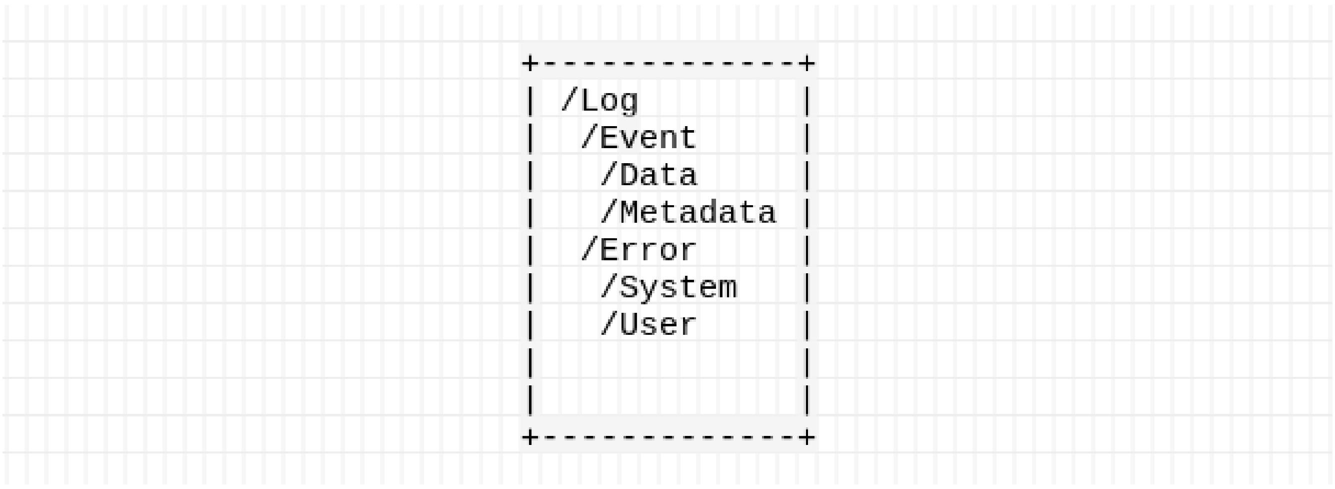
3.2.3.5 Enriched

The lake give a scientific zone called "Enriched". The struct of zone have not been defined yet. But the target is the zone will holds the addition feature generating by the lake big data advance tool set. These feature will link to the original data in Trusted and Transformed zone. The additive metadata be saved in metadata space as normal metadata. The access permission is for the lake scientist team.

3.2.4 Space architect

3.2.4.1 Log

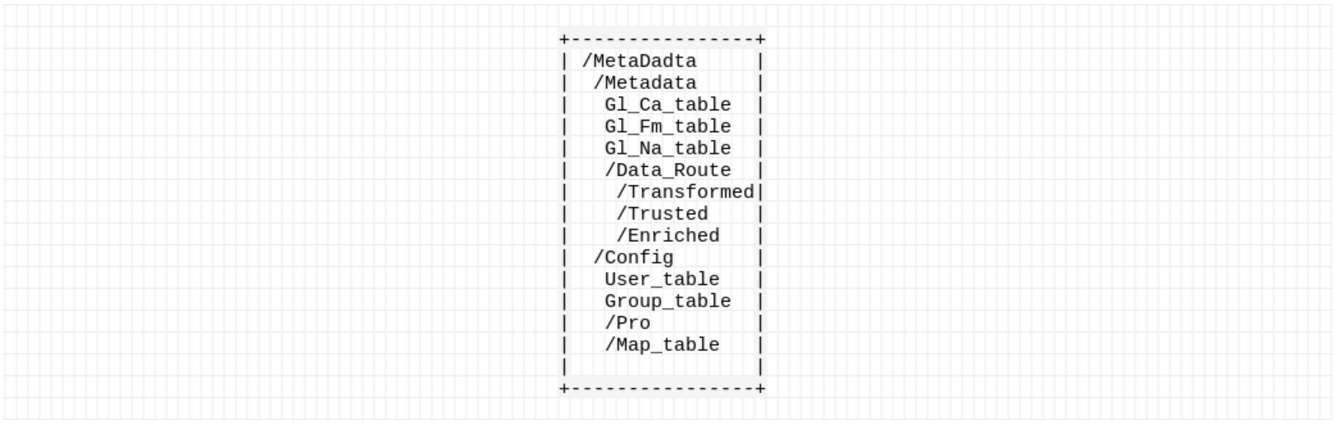
Log space records any event about data in the system. For example, when new data is push in, the update of metadata and the data access time, v.v. Another role of log space is for tracking system error, crash or conflict. More log function can be added in the future. This space is system and administrator access only.



```
+-------------+
| /Log        |
|  /Event     |
|   /Data     |
|   /Metadata |
|  /Error     |
|   /System   |
|   /User     |
|             |
|             |
+-------------+
```

3.2.4.2 MetaDatas

This space is intended for the data metadata and system configure. the data metadata include enough information to track out any data file as well as their addition information in the lake. And the system configure holds any config file about data schema, zone struct, system permission and process configuration, etc. The extra information and metadata schema can be improved in future. For now, the simple metadata space will be like:



```
+----------------+
| /MetaDadta     |
|  /Metadata     |
|   Gl_Ca_table  |
|   Gl_Fm_table  |
|   Gl_Na_table  |
|   /Data_Route  |
|    /Transformed|
|    /Trusted    |
|    /Enriched   |
|  /Config       |
|   User_table   |
|   Group_table  |
|   /Pro         |
|   /Map_table   |
|                |
+----------------+
```

In here, “MetaData” sub-dir holds several global table files which can be catalog, format and title name. Then these table will be map back to each file which has route store in “Data_Route” sub-dir. For “Config” sub-dir, it contains 4 properites: User_table, Group_table, “Pro” folder - holds process configuration, “Map_table” folder - keeps all “format_map_table” file (the struct is not yet decided).

The most important part of the metadata space is “Data_Route”, it link every extra information to the destination file. Since the metadata space is not expected to control Raw zone, it keep track only 3 uppler level zones, so the “Data_Route” is splited into 3 sub space with name following the zones name. In each sub-dir, the data metadata is flatten and has the name constructing as:

```
Group + User + Path_to_file
# The path to file seperate by "." punctuation
################################################
# Example:
Group_0.User_0.My_folder_0.Patient_1.CT.image_1
# NOTE: The final file does not include filename extension.
```

Each file in this dir will contain catalog, type, format, name and security propeties. It can include extra information in the future.

Reference

1. https://www.thedigitaltalk.com/blog/2019-7-how-to-structure-the-data-lake
2. https://dzone.com/articles/data-lake-governance-best-practices
3. https://lingarogroup.com/data-lake-architecture/
4. https://www.xenonstack.com/insights/data-catalog/