Simple data pipeline for ETL and data aggregation

Lê Văn Duyệt (me [at] duyet.net)
12-09-2018

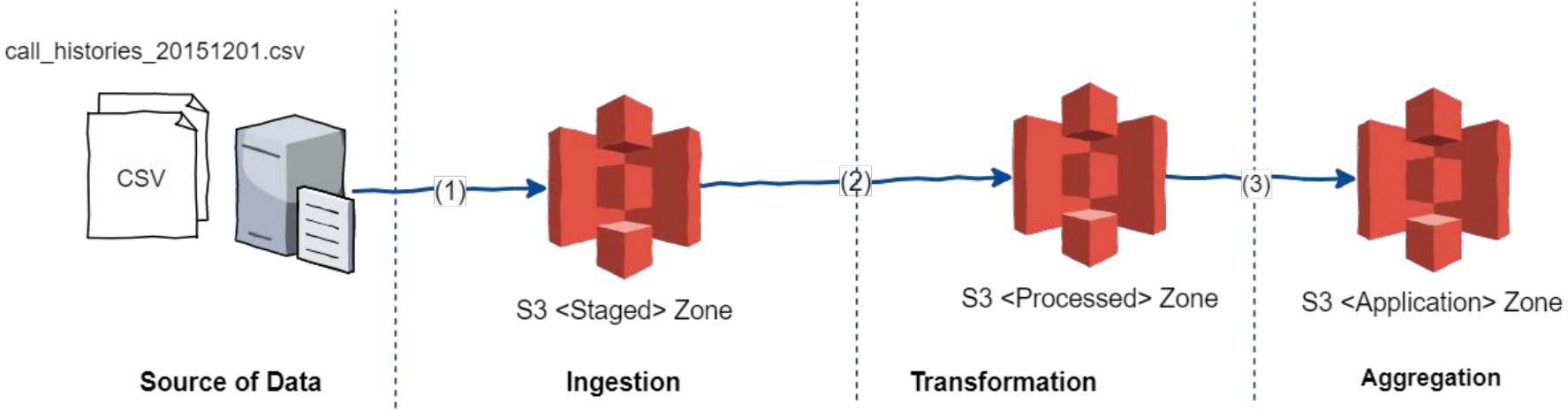
Overview

The goal of this document is develop a simple data pipeline for ETL and data aggregation.

Our system main functions:

- Collecting all data.
- Transforming the data (cleaning, formatting, deduplication).
- Storing in HDFS.
- Doing data aggregation on daily basis

Overview



The pipeline has 3 main stage:

1. Ingestion
2. Transformation
3. Aggregation

Overview

1. **Ingestion stage:**
   - Load data from FTP server(s) into <Staged> zone
   - Which acts as incoming staging areas where raw input data shipped to as well as temporary data files reside.
2. **Transformation stage:**
   - This stage is the most complex in the entire data transformation process where schema validation is performed, cleaning, deduplication
     - Step 1: Parsing and inferring structure
     - Step 2: Validating data schema against registered schema
     - Step 3: Cleaning, deduplication
     - Step 4: Move data to <Processed> zone
3. **Aggregation**
   - Doing data aggregation based on business model. Move data to <Application> zone.

Logs: For each Stage, the process will always create a log to track which data files/tables/partitions have been processed and the status (successful, fail, etc...).

Overview

Trigger execution:
For some processes, resume and continuation can be automatic for example next day process will always from the last day process. Some processes require manual intervention to make decision how to proceed/restart.

Zone folder template path:
We will structure data in storage with template path to optimize for processing

`/<zone>/y=.../m=.../d=.../<data type>/<data type>_<YYYmmdd>.csv`

With: <zone> is **“staged”, “processed”** or **“application”;** <data type> can be **“call_histories”**, **“message_histories”**, **“top_up_histories”, ...**

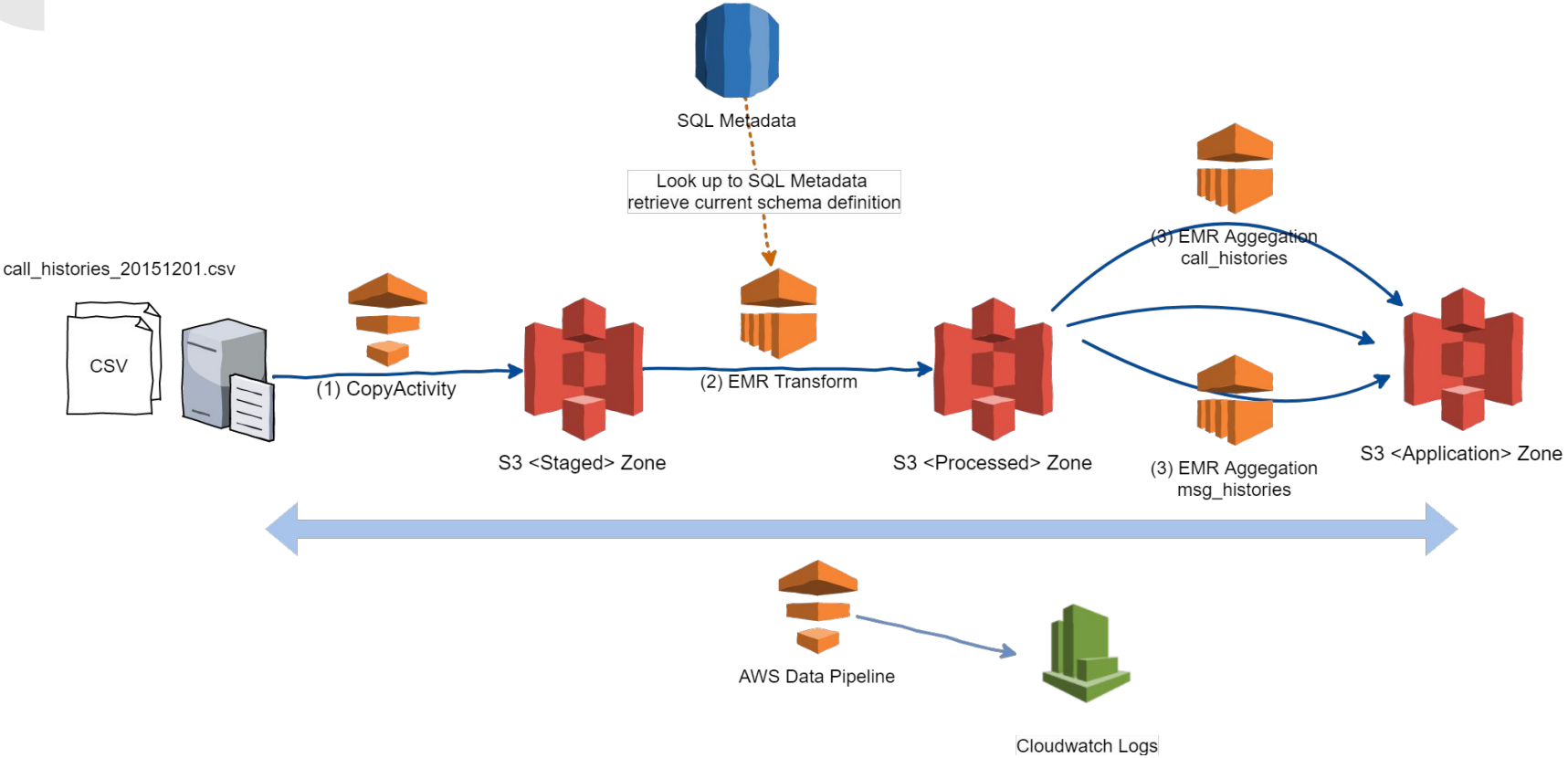
Technologies used

In this proposal design, I use Amazon Web Services for all implementation.

Amazon Web Services (AWS) is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow.

Photo: https://www.amoebanetworks.com/amazon-web-services.html

Overall Architecture



Overall Architecture - components (1)

Main service components:

- **AWS Data Pipeline**: AWS Data Pipeline is a web service that you can use to automate the movement and transformation of data. With AWS Data Pipeline, you can define data-driven workflows, so that tasks can be dependent on the successful completion of previous tasks.

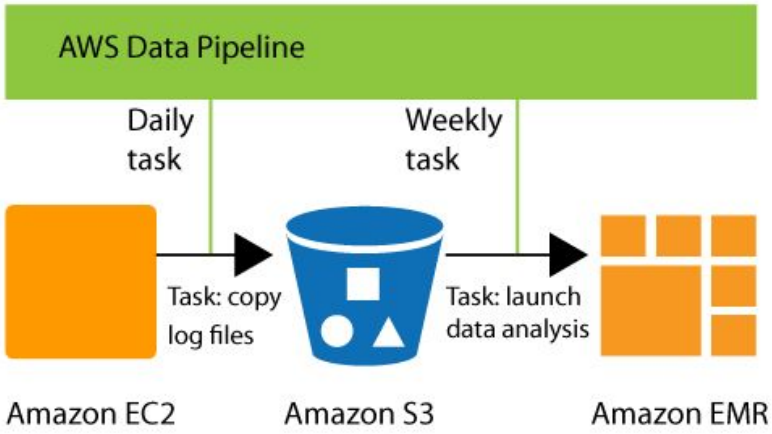


Photo: https://aws.amazon.com/datapipeline/details/

Overall Architecture - components (2)

Main service components:

- **AWS EMR**: Amazon EMR uses Hadoop processing combined with several AWS products to do such tasks as web indexing, data mining, log file analysis, machine learning, scientific simulation, and data warehousing..

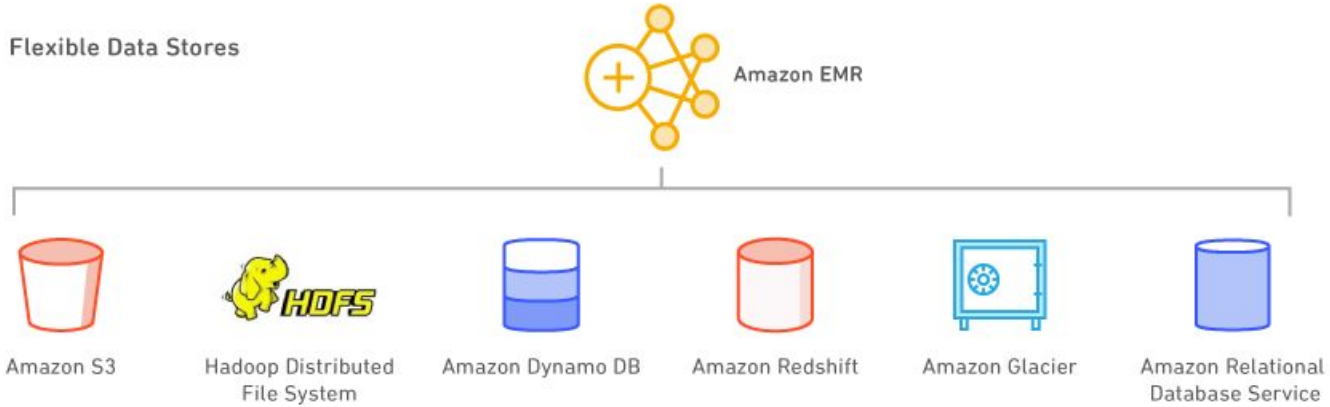
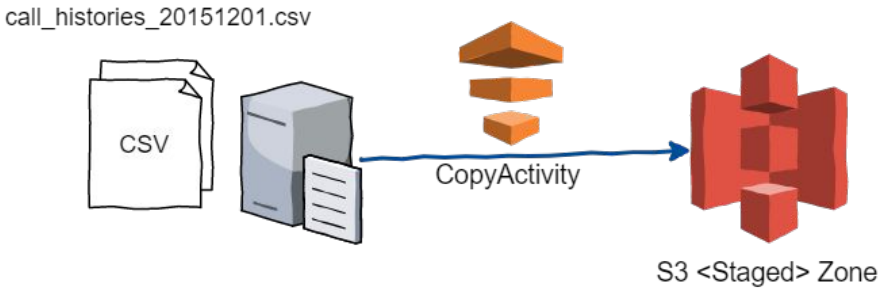


Photo: https://aws.amazon.com/emr/details/

Overall Architecture - components (3)

Main service components:

- **AWS S3:** Amazon S3 is object storage built to store and retrieve any amount of data from anywhere. When it comes to Hadoop data storage on the cloud though, the rivalry lies between Hadoop Distributed File System (HDFS) and Amazon's Simple Storage Service (S3). Although Apache Hadoop traditionally works with HDFS, it can also use S3 since it meets Hadoop's file system requirements.
- You can use Amazon S3 as storage option on EMR without configuring anything by just using URI scheme **s3://**

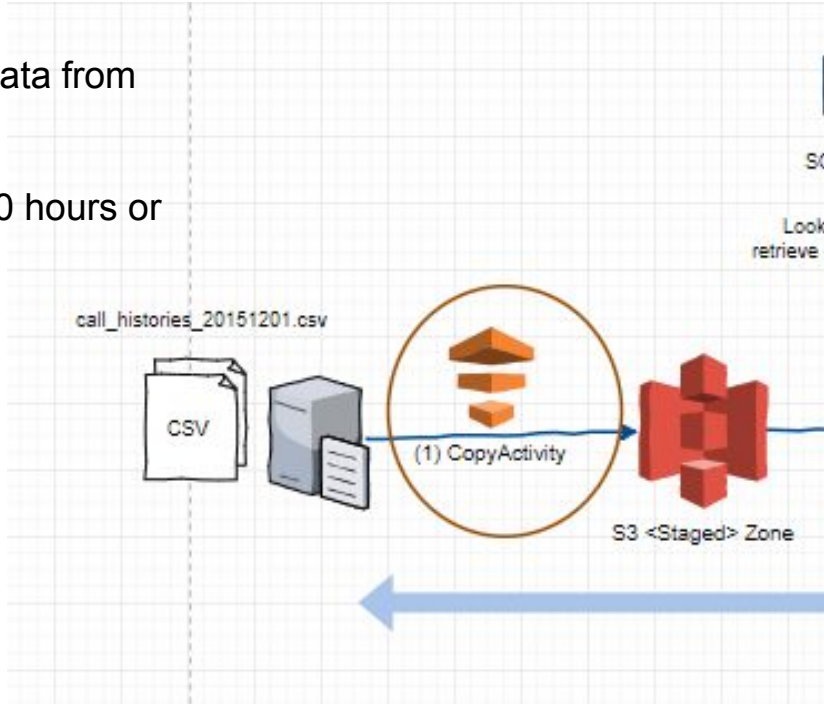


Overall Architecture - in detail

Data flows:

(1) Using **CopyActivity** of Data Pipeline to schedule copy data from FTP server(s) to <Staged> zone in S3.

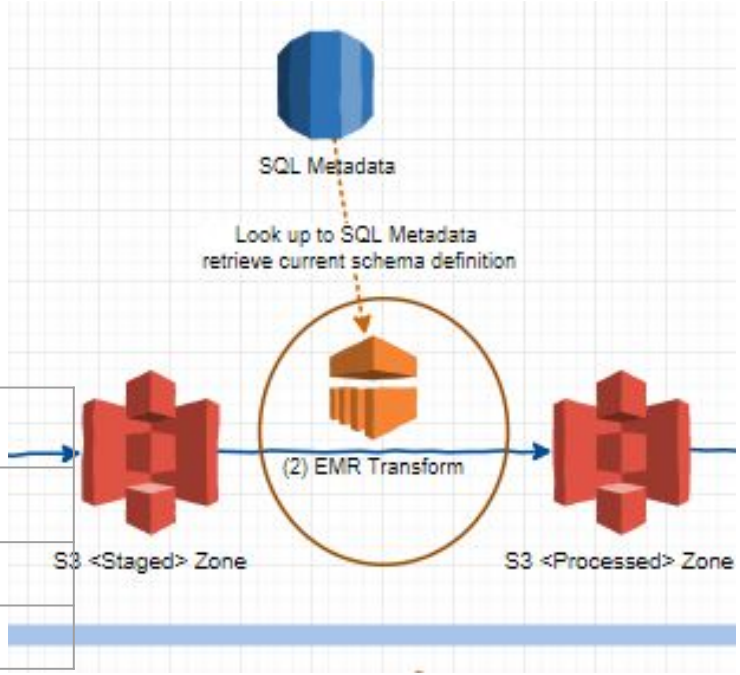
We can define a **schedule of every day** starting at 00:00:00 hours or something else, learn more at Schedule section.



Overall Architecture - in detail

Data flows:

(2) EMR Transform using EmrActivity of Data Pipeline, which trigger and EMR task for transformation. This stage also lookup the SQL Metadata table (from RDS) to retrieve the schema definition for each data type.



Metadata table: data_type

type	column_name	data_type
call_histories	FROM_PHONE	string
call_histories	TO_PHONE	string
call_histories	..	...

Metadata table: config

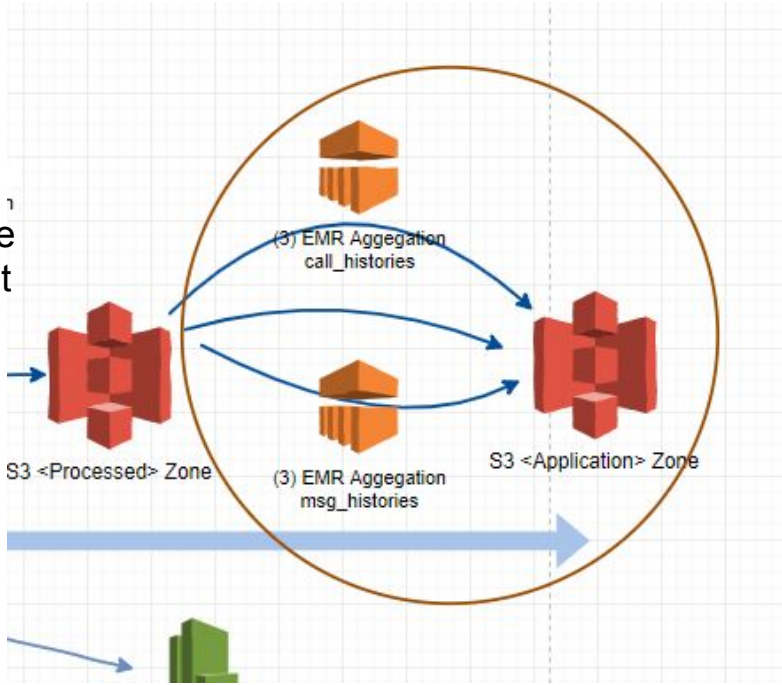
type	config_name	config_val
call_histories	sep_char	;
call_histories	delim	...
call_histories	..	...

Overall Architecture - in detail

Data flows:

(3) EMR Aggregation for each **data type** using EmrActivity of Data Pipeline, which trigger and EMR task for aggerration.

This staged can be run aggregation for multi data type, with the same code base but can have different logic transform by input parameters.

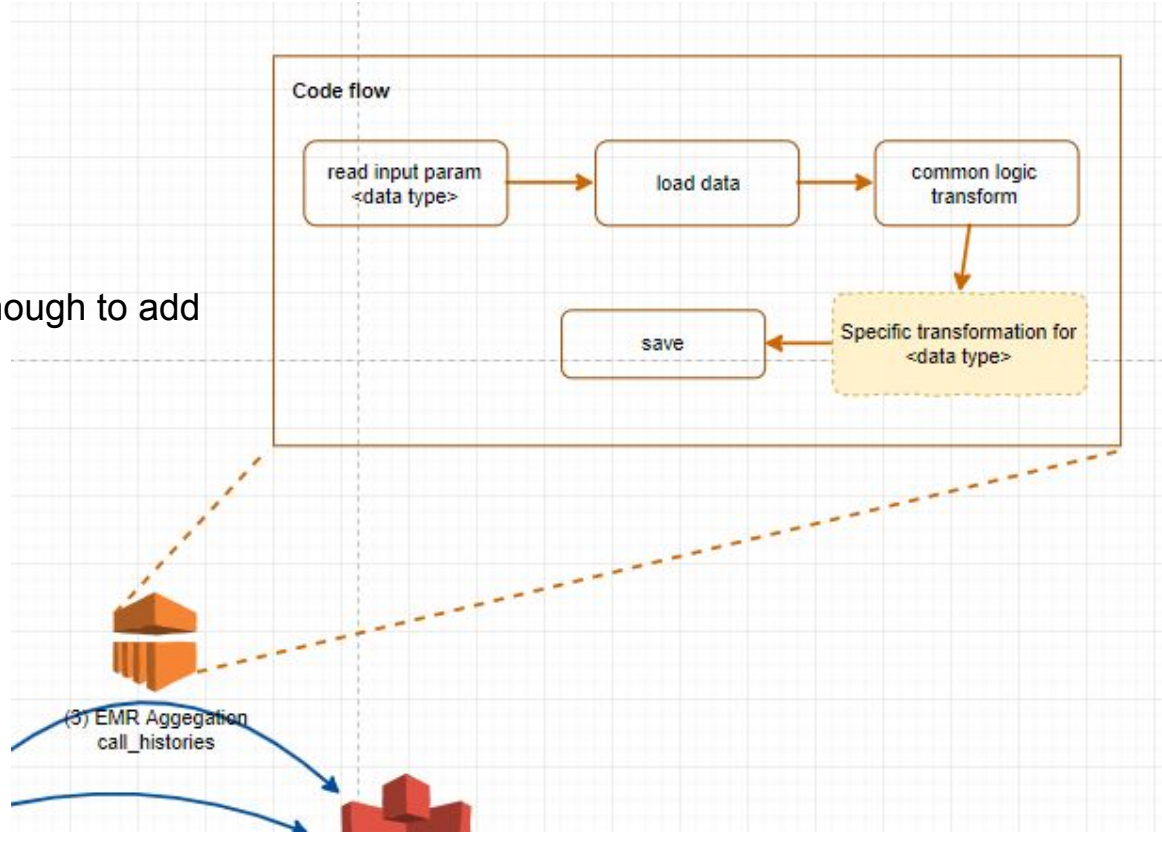


Overall Architecture - in detail

Data flows:

(3) (continue)

With this code flow, it can be flexible enough to add new file type in the future easily



Scalable, HA and fault tolerant

AWS Data Pipeline allows you to take advantage of a variety of features such as scheduling, dependency tracking, and error handling.

AWS Data Pipeline helps you easily create complex data processing workloads that are fault tolerant, repeatable, and highly available. You don’t have to worry about ensuring resource availability, managing inter-task dependencies, retrying transient failures or timeouts in individual tasks, or creating a failure notification system. AWS Data Pipeline also allows you to move and process data that was previously locked up in on-premises data silos.

The end