DRAFT

Data Pipeline Design & Operational Document

Jan 2017

Incedo

379 Thornall St, Edison, NJ 08837

1. Purpose

This Design/Development document is to provide the team with a description of the Data Pipeline system that can be supported and maintained.

The focus of this document is to provide a Data Pipeline overview that implements the Inventiv project requirements. The audience of this document is support design and development team members. This document describes the workflows to load data from source to ODS keyspace.

This is a work-in-progress / evolving document and will updated when complete information is available.

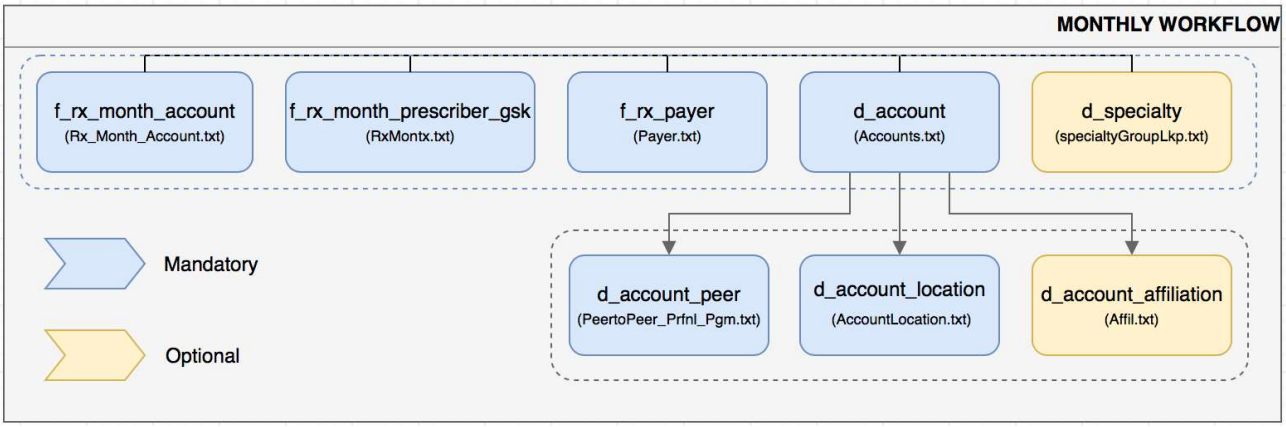
2. Reference Documents

Document Name	Description / Link
Dependency List	https://docs.google.com/spreadsheets/d/1r4XpBeL-B-NVOYMRBjAv2bCDTpSaOD-aX4IqPG-z6WE/edit#gid=1773672664
Cascading Failures and Reruns	http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-manage-cascade-failandrerun.html
Changing DataPipeline limitations	http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-manage-pipeline-modify-console.html#dp-edit-pipeline-limits

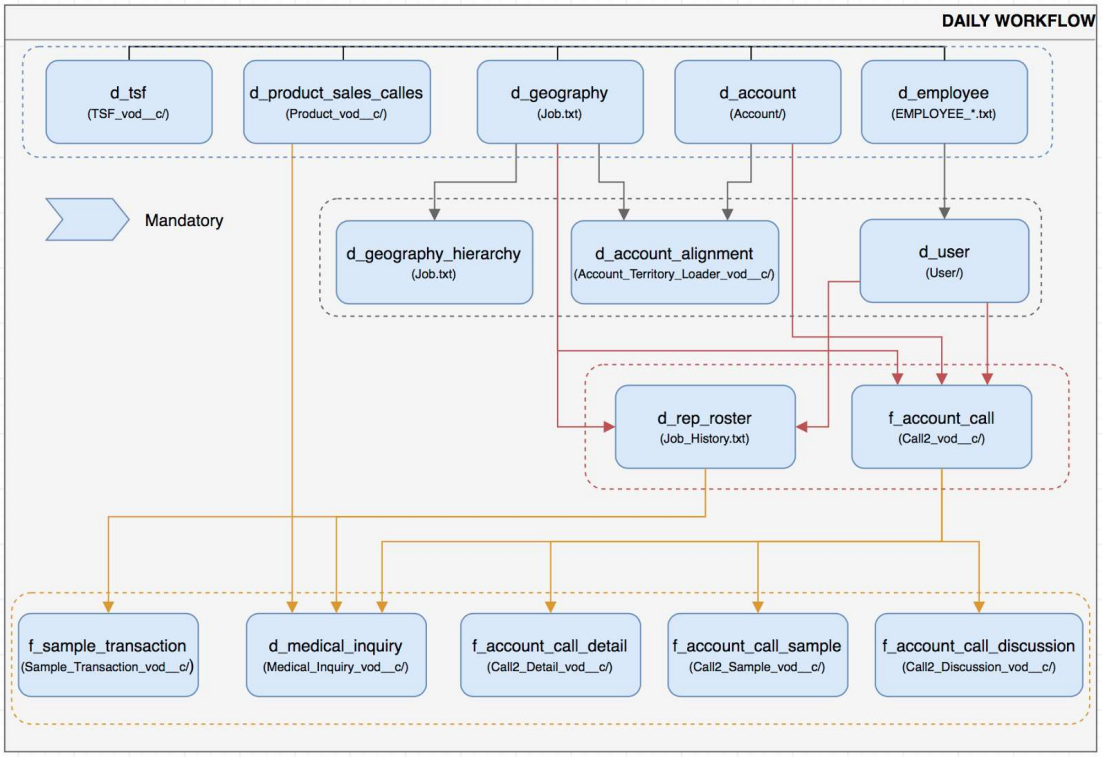
3. Requirement : Dependency Views

Dependencies between jobs for daily/weekly/monthly pipelines are graphically described in this section. Groups surrounded by rectangles of dashes can run simultaneously. If any job fails all dependent jobs will fail with cascade fail error.

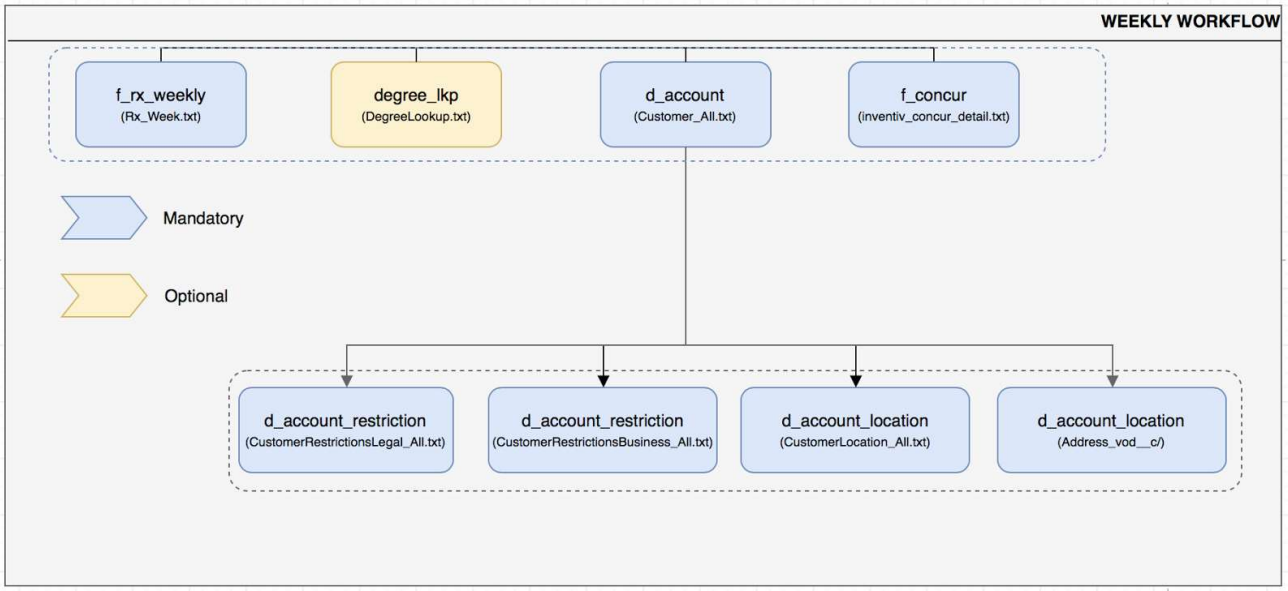
3.1. Monthly



3.2. Daily



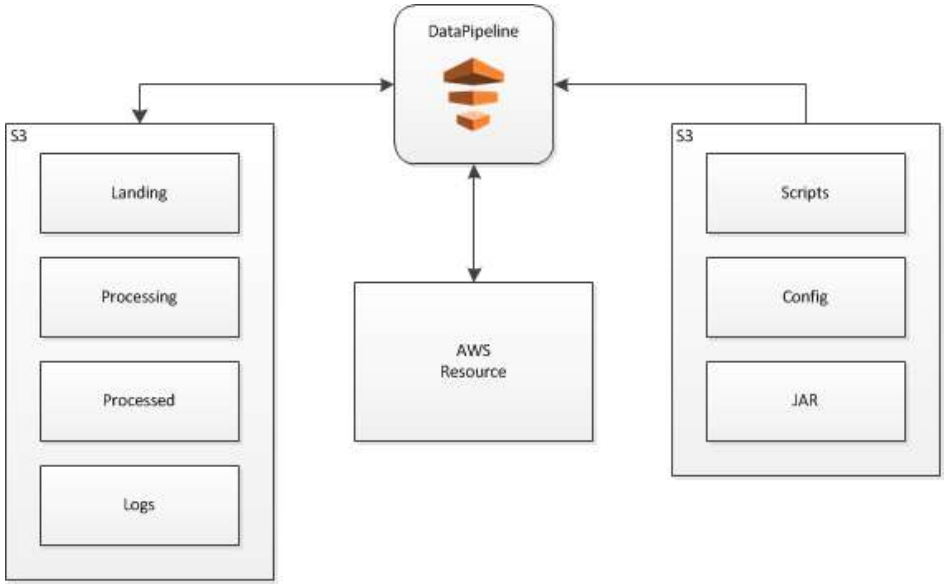
3.3. Weekly



4. Data Pipeline Design - Conceptual View

4.1. Application Design

Daily/weekly/monthly pipeline uses items that are showed on the diagram below:



Based on the requirements the Data Pipeline has been divided into :

1. OneTime
2. Daily
3. Weekly

4. Monthly

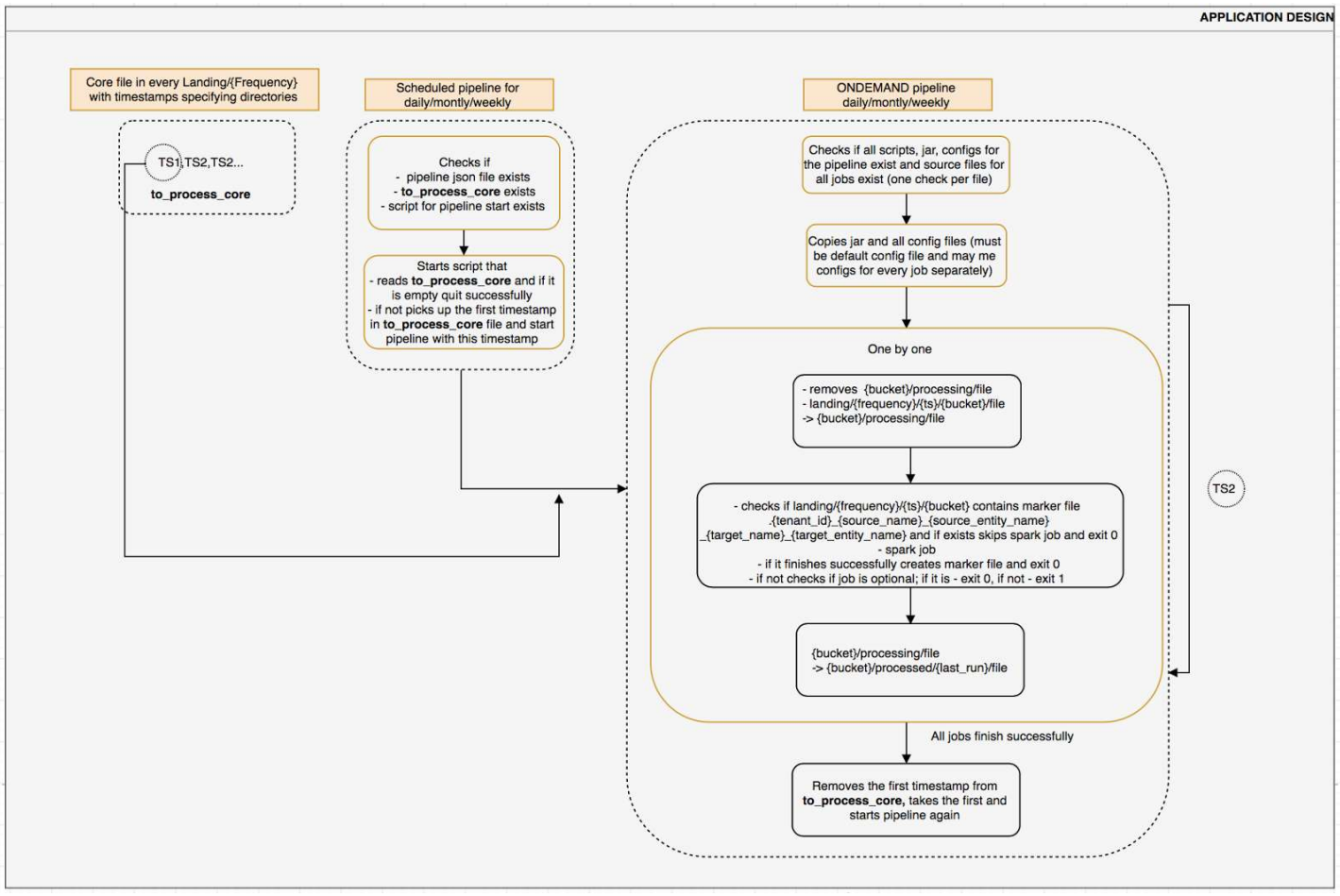
The daily , weekly and monthly pipelines are not dependent on each other data loads but only on the OneTime Load.

During the start of the pipeline , one of the initial activity copies the jar and config files over to the resource to prepare it to be used by the Load Job. Each Load Job inside the pipeline is an activity of type ShellCommandActivity that uses Script Uri that points to the bash scripts residing on the S3 bucket.

This activity passes parameters to the bash script which are part of the same data pipeline. The jar and the conf files inside the JAR and Config folders respectively are used by the Load Job.

A series of connected activities based on the dependency views from previous section will then become a Data Pipeline.

Following diagram displays general flow for all data pipelines:



{Frequency} - Daily/Weekly/Monthly folders

Daily/weekly/monthly pipelines will be started by the scheduled start data pipeline that will pick up timestamp to process from core file in corresponding Landing/{Daily/Weekly/Monthly} folder and activate the pipeline. For every pipeline there will be scheduled start pipeline.

4.2. Abbreviations & Definitions

Item	Description
AWS	Amazon Web Services
Data Pipeline	A data workflow orchestration service that helps you process and move data.
Activity	An activity is a pipeline component that defines the work to perform.
Resource	Computational resources can perform the work that a pipeline activity specifies for AWS Data Pipeline, Pipeline automatically creates an EC2 instance.
Schedule	This defines the timing of a scheduled event.
Bucket/Folders	Bucket is a unit of storage in a AWS - simple storage solution (s3). There can be folders under a bucket.

4.3. System Description

Data Pipeline can schedule activities on two environments : Development & Production

The below table lists the environment/components that the data pipeline uses :

Item	Item Information	Description
Driver	xxx.xxx.xxx.250	Driver / Client for Spark Job Submission for Dev
Driver	xxx.xxx.xxx.156	Driver / Client for Spark Job Submission for Prod
Cassandra Host	xxx.xxx.xxx.220	Production Cassandra Host used in the job.conf file

Cassandra Host	xxx.xxx.xxx.12	Development Cassandra Host used in the job.conf file
AWS S3 Keys	dpgsk keys	Used in the job.conf file
Cassandra Database	Ods.etl_config	Used to retrieve run information for a particular combination of source and target inputs.
AWS Console	AWS Console Access	AWS Console is used to create, maintain and run the pipeline.
AWS Cli	-	AWS Command Line Interface (CLI) is a unified tool to manage your AWS services
Load Job	-	This is the job that that processes data using the DSE Spark-Submit command
Bash Shell	On the resource	Shell to run the scripts that are part of the data pipeline shellcommandactivity

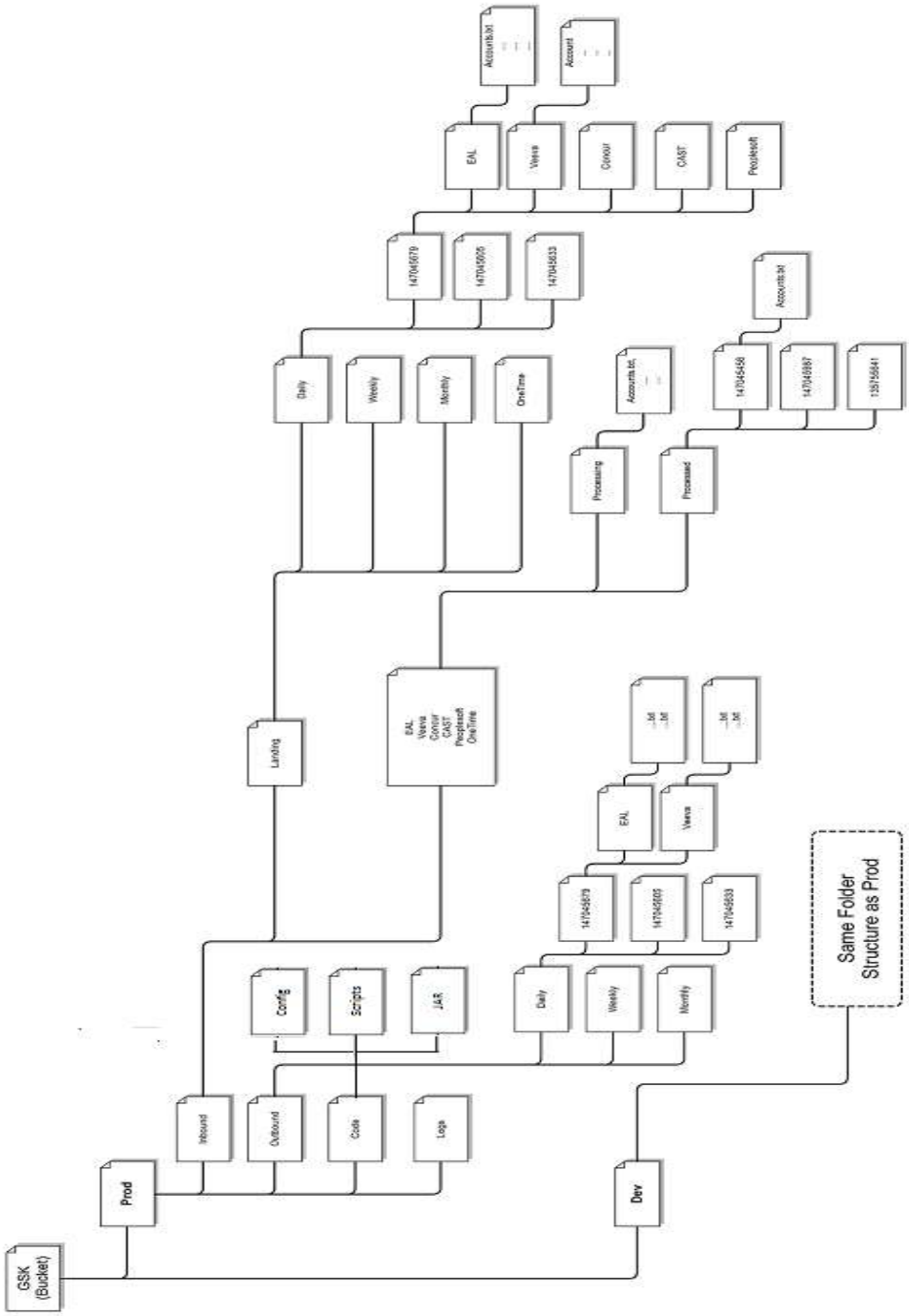
4.4. Items configured for the Resource in a pipeline

Some of the environment components that are configured and required are listed below

Item	Value	Description
Sub-Net Id	subnet-bdeb1c90	Subnet for Dev
Image-Id	ami-ab4494bd	Image Id of the resource on Development
Security-Group	xx-xxxx3276	AWS Security Group
Instance Type	m3.xlarge	AWS Instance (Paravirtualization)
Region	us-east-1	AWS Region

5. Application Design

5.1. S3 Bucket Design



5.2. Data/Files Structures Description

5.2.1. Landing Folders

The below tables are the source for the data pipeline for further data handling. The paths are manually created and the data will reside on these folders for the data-pipeline to use and process the next steps. The below is subjected to change when a complete confirmation on the folder structure is established.

AWS S3 Path	Description
<<Bucket/Folders>>/Landing/Daily/<<Timestamp>>/EAL	Daily Landing path/folder for EAL
<<Bucket/Folders>>/Landing/Daily/<<Timestamp>>/Veeva	Daily Landing path/folder for Veeva
<<Bucket/Folders>>/Landing/Daily/<<Timestamp>>/CAST	Daily Landing path/folder for CAST
<<Bucket/Folders>>/Landing/Daily/<<Timestamp>>/Peoplesoft	Daily Landing path/folder for Peoplesoft
<<Bucket/Folders>>/Landing/Daily/<<Timestamp>>/Concur	Daily Landing path/folder for Concur

AWS S3 Path	Description
<<Bucket/Folders>>/Landing/Weekly/<<Timestamp>>/EAL	Weekly Landing path/folder for EAL
<<Bucket/Folders>>/Landing/Weekly/<<Timestamp>>/Veeva	Weekly Landing path/folder for Veeva
<<Bucket/Folders>>/Landing/Weekly/<<Timestamp>>/CAST	Weekly Landing path/folder for CAST
<<Bucket/Folders>>/Landing/Weekly/<<Timestamp>>/Peoplesoft	Weekly Landing path/folder for Peoplesoft
<<Bucket/Folders>>/Landing/Weekly/<<Timestamp>>/Concur	Weekly Landing path/folder for Concur

AWS S3 Path	Description
<<Bucket/Folders>>/Landing/Monthly/<<Timestamp>>/EAL	Monthly Landing path/folder for EAL
<<Bucket/Folders>>/Landing/Monthly/<<Timestamp>>/Veeva	Monthly Landing path/folder for Veeva
<<Bucket/Folders>>/Landing/Monthly/<<Timestamp>>/CAST	Monthly Landing path/folder for CAST
<<Bucket/Folders>>/Landing/Monthly/<<Timestamp>>/Peoplesoft	Monthly Landing path/folder for Peoplesoft
<<Bucket/Folders>>/Landing/Monthly/<<Timestamp>>/Concur	Monthly Landing path/folder for Concur

5.2.2. Processing Folders

As per the below table the folders upto Processing Folder must pre-exist along with Processing.

AWS S3 Path	Description
<<Bucket/Folders>>/EAL/Processing/	A S3 Processing path/folder for EAL from there the Load Job will pick the files from.
<<Bucket/Folders>>/Veeva/Processing/	A S3 Processing path/folder for Veeva from there the Load Job will pick the files from.
<<Bucket/Folders>>/CAST/Processing/	A S3 Processing path/folder for CAST from there the Load Job will pick the files from.
<<Bucket/Folders>>/Peoplesoft/Processing/	A S3 Processing path/folder for Peoplesoft from there the Load Job will pick the files from.
<<Bucket/Folders>>/Concur/Processing/	A S3 Processing path/folder for Concur from there the Load Job will pick the files from.

5.2.3.Processed Folders

The below folder/path will be constructed upon successful execution of the load job.

AWS S3 Path	Description
<<Bucket/Folders>>/EAL/Processed/<<UnixTimeStamp>>/<<FileName>>	Processed path/folder for EAL
<<Bucket/Folders>>/Veeva/Processed/<<UnixTimeStamp>>/<<Folder>>/<<File Names>>	Processed path/folder for Veeva
<<Bucket/Folders>>/CAST/Processed/<<UnixTimeStamp>>/<<FileName>>	Processed path/folder for CAST
<<Bucket/Folders>>/Peoplesoft/Processed/<<UnixTimeStamp>>/<<FileName>>	Processed path/folder for Peoplesoft
<<Bucket/Folders>>/Concur/Processed/<<UnixTimeStamp>>/<<FileName>>	Processed path/folder for Concur

The unixTimeStamp is converted from the date and time returned from field : last_run of the etl_config table

5.2.4 to_process_core files

The below files are used by start pipeline

5.3. Daily Data Pipeline Architect Tree Diagram

5.3.1. Ods.etl_config Table

As the job load requires and uses the entries in the table ods.etl_config , the data-pipeline leverages the same entries from the table.

Column Name	Description
last_run	last _run will hold the date that will be retrieved and converted to toUnixTimestamp
prev_entity_loc	Example: <<Bucket/Folders>>/Veeva/Processed/{{date}}
current_entity_loc	Example: <<Bucket/Folders>>/Veeva/Processing
landing_loc	Example: <<Bucket/Folders>>/Landing/<<Timestamp>>/Veeva

The data-pipeline picks up the information from the above table for checking whether transferring files from landing_loc to current_entity_loc and from current_entity_loc to prev_entity_loc using last_run.

5.3.2. Other Config Tables

For the job load to be successful, ensure that other tables below have the related records.

Item
ods.ref_mapping
ods.mapping_config
ods.group_operation

5.3.3. Bash Scripts*

Data Pipeline uses below scripts for picking the files from the Landing folders provided in the above table. Data pipeline uses shellcommandactivity to invoke the scripts.

Item	Description
check-on-file.bash	Script to verify if file/folders exist
copyjar2local.bash	Script that copies jar file and configs to a local resource
landing2processing.bash	Script that copies files/folder from landing directory to processing directory
start-process.bash	Script that invokes dse spark-submit, create marker file and check if the job optional
processing2processed.bash	Script that moves files/folder from processing to processed directory
restart-pipeline.bash	Script that launches new pipeline with new timestamp
copy-credentials.bash	Script that copies .cqlshrc and .dserc to the Resource

5.3.4. Github Location

The Data Pipelines are located under the github at the following location :

Item
https://github.com/thllabs/etl/tree/development/data-pipeline

5.3.5. Notifications

The pipeline leverages the AWS SNS (Simple Notification Services) to send a message over to an email in the event of failures or success at certain points.

Email configuration for notification : Datateam_support@inventivhealth.com

A Sample success message is shown below :

Pipeline Id : df-04326771AKKWSKC6OVIE

"Between these times : 2017-02-21T14:48:59 - 2017-02-21T14:49:33."

Job Id: @ShellCommandActivityId_SrIYE_2017-02-21T14:39:25_Attempt=1

A sample failure message is shown below :

Pipeline Id : df-08665142NSC8RCM5OLDQ

"Error for interval 2017-02-18T19:39:24 - 2017-02-19T19:39:24."

Job Id: @ShellCommandActivityId_1D23I_2017-02-18T19:39:24_Attempt=1

Error Message: fatal error: An error occurred (404) when calling the HeadObject operation: Key "Landing/20170218/CAST/Job.txt" does not exist

5.3.6. Preconditions

Preconditions tests apply to verify the bucket/folder for bash scripts and jar and config files exist . Upon the successful verification of the keys the data pipeline proceeds for the job executions.

Below are the preconditions of the data pipeline :

CheckOnDataProcessor

CheckOnJobConfig

CheckOnCheckOnFileScript

CheckOnStartBash

CheckOnRestartPipelineScript

CheckOnProcessingToProcessedScript

CheckOnCopyJar2Local

CheckOnLandingToProcessingScript

CheckOnCopyCredentialsScripts

5.3.7. Parameters

These parameters are part of the daily/weekly/monthly data pipeline.

PARAMETERS	DEFINITIONS
myTotalExecutorCores	--total-executor-cores
myDriverMemory	--driver-memory
myExecutorMemory	--executor-memory
myExecutorCores	--executor-cores
myBatchGroupingKey	--conf spark.cassandra.output.batch.grouping.key=

myConcurrentWrites	--conf spark.cassandra.output.concurrent.writes=
myConsistencyLevel myExtraJavaOptions	--conf spark.cassandra.output.consistency.level= --conf
myBatchSizeBytes	--conf spark.cassandra.output.batch.size.bytes=
myPipelinePathToJson	Path to the json file to create current pipeline from
myPipelineName	Name of the current pipeline
myScriptsBucket	Bucket that contains all necessary scripts (start-process.bash etc)
myJarBucket	Bucket that contains fat JAR for spark jobs
myConfigBucket	Bucket that contains all necessary config files for spark jobs
myLogsBucket	Bucket for activity logs
mySourceTimestampsBucket	Bucket that contains source file with timestamps to process
myTimestampFolder	Timestamp folder in Landing directory to process

Every activity has its own arguments:

1) check-on-file.bash

ARGUMENTS	DEFINITIONS
tenant_id	
source_name	
source_entity_name	
target_name	
target_entity_name	

myTimestampFolder	

2) copyjar2local.bash

PARAMETERS	DEFINITIONS
myJarBucket	
myConfigBucket	

3) landing2processing.bash

PARAMETERS	DEFINITIONS
tenant_id	
source_name	
source_entity_name	
target_name	
target_entity_name	
myTimestampFolder	

4) start-process.bash

PARAMETERS	DEFINITIONS
myTotalExecutorCores	
myExecutorCores	
myExecutorMemory	
myDriverMemory	
myExtraJavaOptions	
myConsistencyLevel	
myBatchSizeBytes	
myConcurrentWrites	
myBatchGroupingKey	

configFile	
tenant_id	
source_name	
source_entity_name	
target_name	
target_entity_name	
myTimestampFolder	

5) processing2processed.bash

PARAMETERS	DEFINITIONS
tenant_id	
source_name	
source_entity_name	
target_name	
target_entity_name	

6) restart-pipeline. bash

PARAMETERS	DEFINITIONS
today_date	
myPipelineName	
myPipelinePathToJson	
mySourceTimestampsBucket	

7) copy-credentials.bash

PARAMETERS	DEFINITIONS
myConfigBucket	

5.4. User account and privileges

Item	Description
dpgsk	To Design/Create Pipelines, ssh to bastion host and driver
dpsupport	Additional account to Create, Maintenance & Support of pipelines;ssh to bastion host and driver.
Individual accounts	To Design/Create Pipelines ; ssh to bastion host and driver;

The current policies to the above accounts are subjected to changed based on the needs.

IAM Policies
AmazonEC2FullAccess
AmazonS3FullAccess
AWSDataPipelineRole
AWSDataPipeline_FullAccess
AmazonSNSFullAccess

6. Operations

6.1. AWS Access

Secure the IAM account and privileges to access AWS Console. A sample IAM policy that exists for the user account dpsupport is used for maintenance and support of the data pipelines.

6.2. Files / Data preparation

6.2.1. Check on Scripts, Buckets and Folders and the Inbound Files

Ensure that the below artifacts* exists :

<<Env>> - Either Dev or Prod

<<Frequency>> - Daily or Monthly or Weekly or Onetime

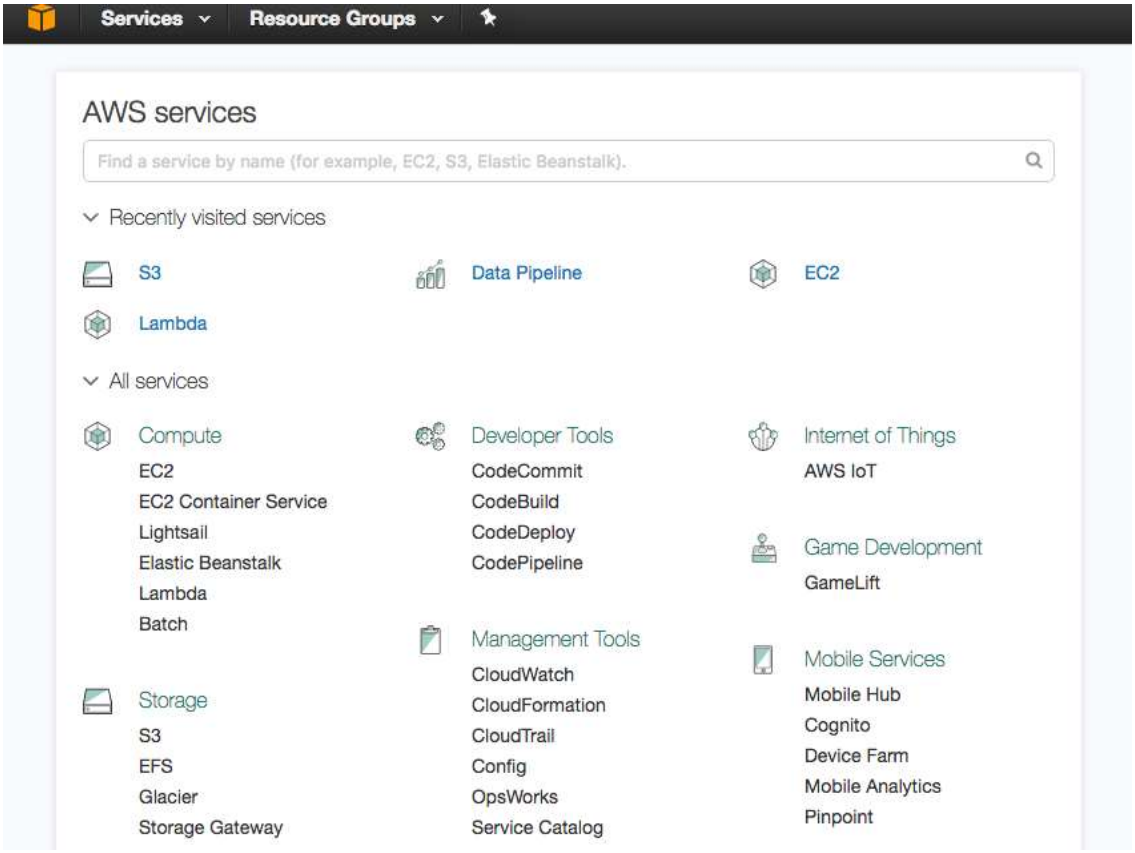
File/Folders	Path
job.conf	<<Bucket/Folders>>/<<Env>>/Code/Config/
dataProcessor-assembly-0.0.1-RC1.jar	<<Bucket/Folders>>/<<Env>>/Code/JAR/
Scripts	<<Bucket/Folders>>/<<Env>>/Code/Scripts/
EAL, Veeva, CAST, Peoplesoft, Concur For Landing for daily/weekly/monthly/onetime - Depending on the the execution is for Daily , Monthly , Weekly or Onetime	<<Bucket/Folders>>/<<Env>>/Inbound/Landing/<<Frequency>>/<<Timestamp>>

6.2.2. Data - Daily

Ensure that the below table have the correct configuration data on the ods keyspace.Please refer the 4.2.1 and 4.2.2 sections.

Table
d_employee
d_user
d_geography
d_rep_roster
d_geography_hierarchy
d_product_sales_call
d_account
d_tsf
d_account_alignment
f_account_call
f_sample_transaction
d_medical_inquiry
f_account_call_detail
f_account_call_discussion
f_account_call_sample

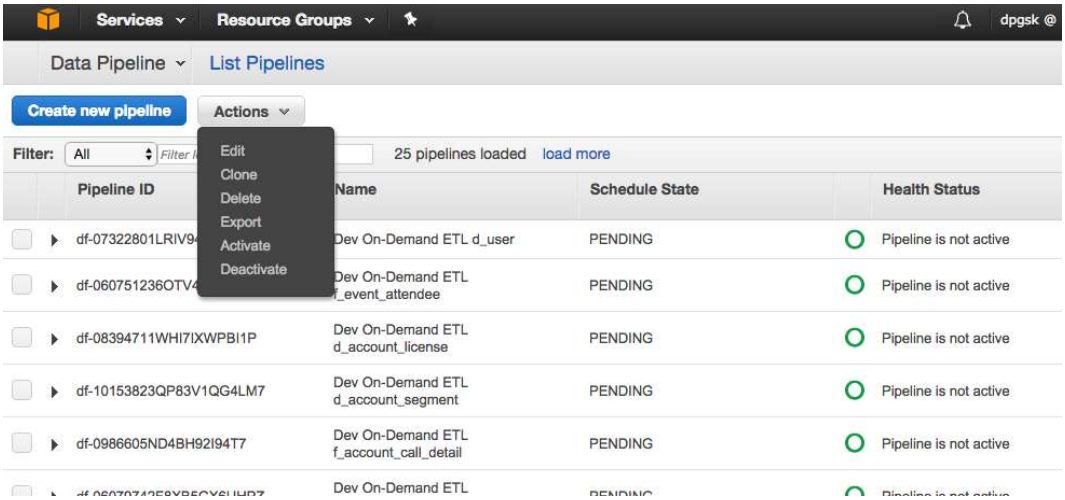
6.3. Creating a pipeline



From the above screen click on the Data Pipeline that will take you to the Data Pipeline home page described in the next section.

6.4. Data Pipeline home page

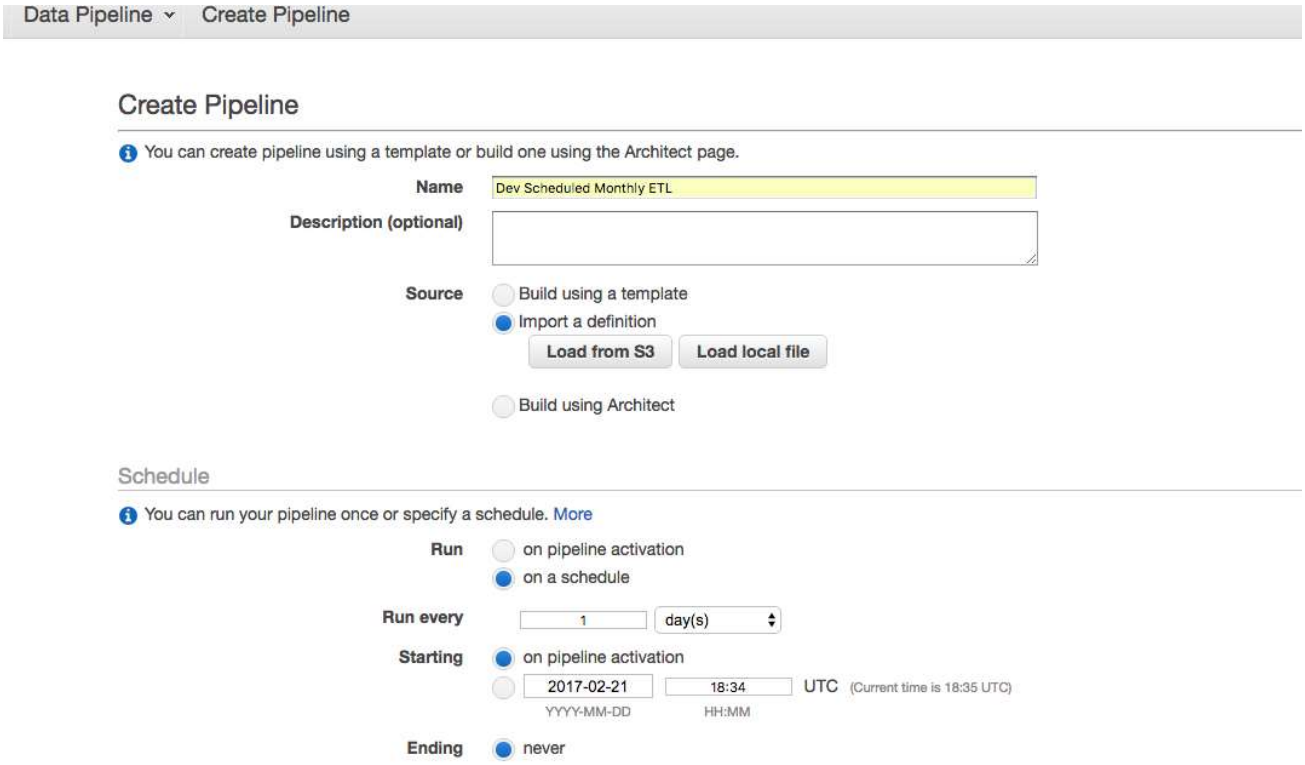
Data Pipelines can be viewed and filtered (load More will fetch pipelines based on the filter) . The Data Pipelines can also be activated through this page.



6.5. Creating the Data Pipeline

The Data Pipeline can be created using Template, Definitions and using the Architect.

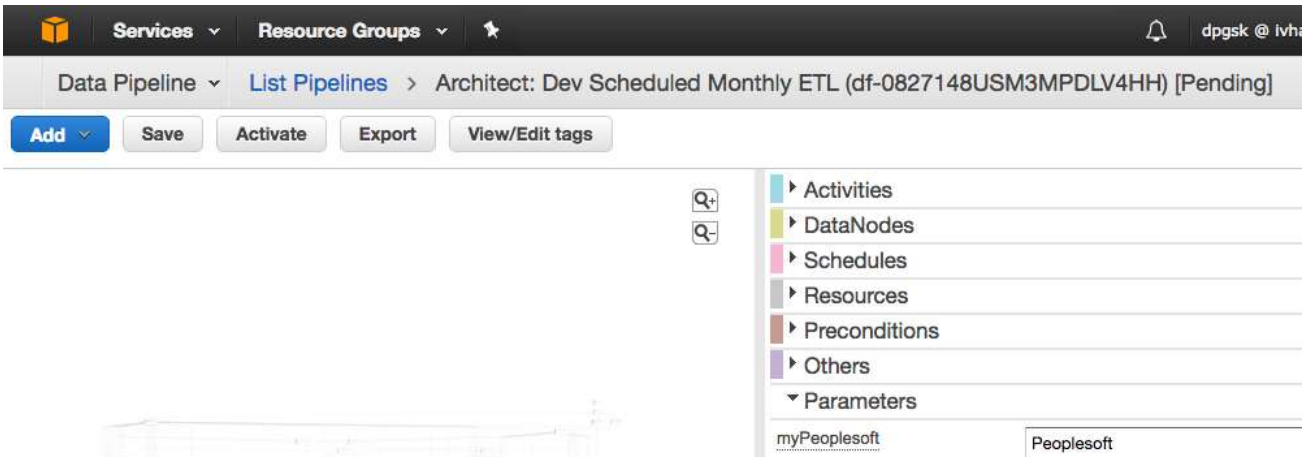
The particular daily pipeline is stored as json on github that will be used to create a new pipeline.



Use the json files from the [github] to load and create the pipeline such as daily.

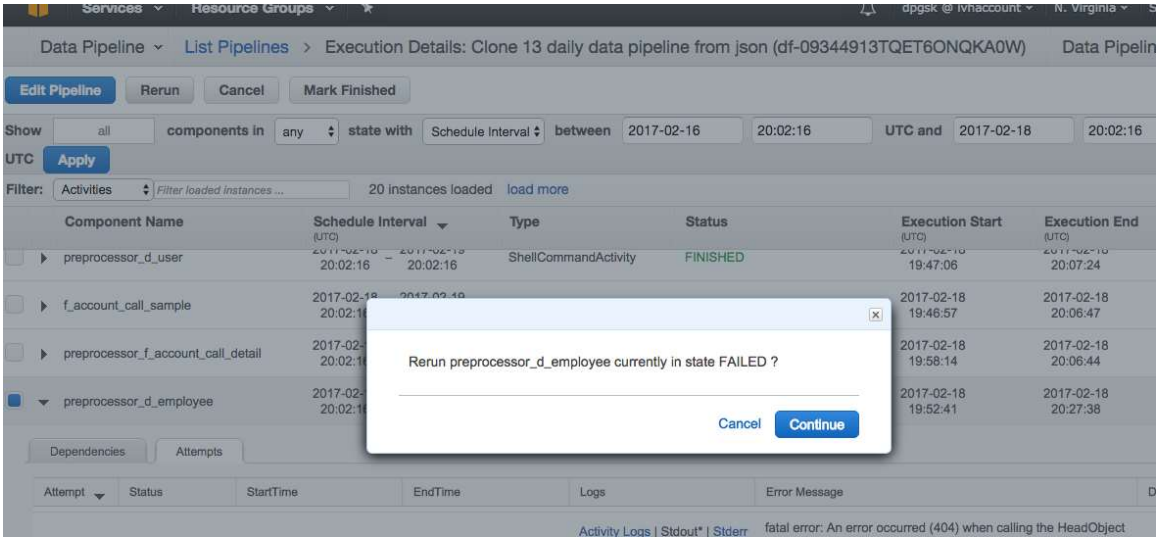
6.6. Change/Save/Activate/Export the Data Pipeline

The below screen shows the Data Pipeline Architect page.



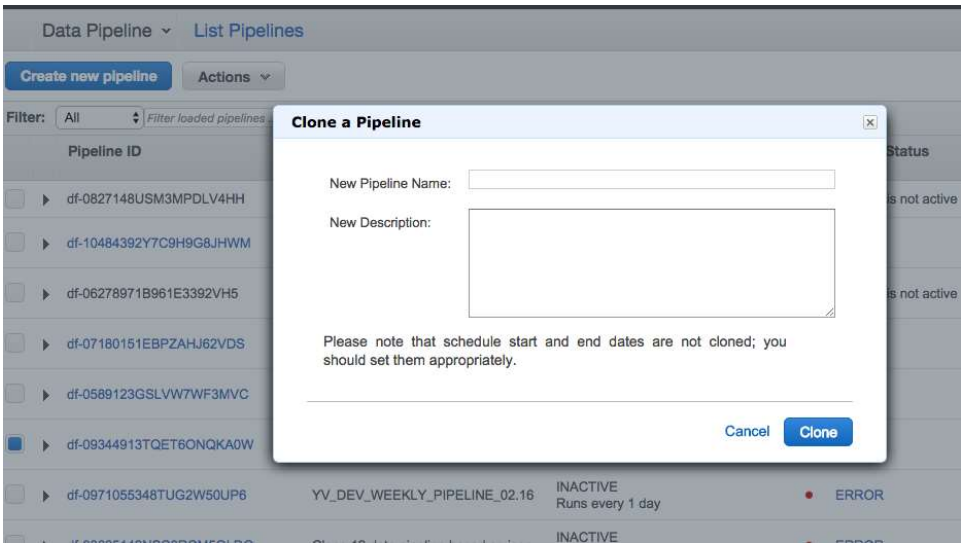
6.7. Rerun the Data Pipeline on the activity level

AWS supports re-run on the activity level only on a scheduled pipeline and can be performed as shown below. Rerun can be performed by navigating to the specific pipeline from the Data Pipeline home page and selecting the activity.



6.8. Cloning the pipeline

Cloning makes a copy of a pipeline and allows it to be given a new name, new Pipelineid and allows it to be edited. Pipeline can be activated in any state and will need to be activated. Cloning is only allowable through Console.



6.9. AWS Data Pipeline using CLI commands

A Complete list can be found here :

http://docs.aws.amazon.com/cli/latest/reference/datapipeline/

Item	Description
aws datapipeline list-pipelines	Lists the pipeline identifiers for all active pipelines
aws datapipeline list-runs --pipeline-id << pipelineid >>	Lists the times the specified pipeline has run.
aws datapipeline activate-pipeline --pipeline-id << pipelineid >>	Validates the specified pipeline and starts processing pipeline tasks