Out of the Box Data Lake User Guide

October 2017

Cognizant Technology Solutions

Contents

1. Purpose & Audience

This guide is intended for users of the “Out of box data lake” quick start to illustrate Big Data best practices with sample Talend jobs developed by Cognizant for integrating Spark, RedShift, Hadoop and S3 technologies into a Data Lake implementation.

2. About “OUT OF THE BOX DATA LAKE”

Data Lake on the cloud plays the role of a key driver for Digital Transformation initiatives for data and operational agility by enabling access to historical and real-time data for analytics. Cognizant in partnership with AWS and Talend brings together a solution that will enable customers to build and deploy a Data Lake on AWS in 60% less time. “OUT OF THE BOX DATA LAKE” accelerates the process to build and deploy a Data Lake solution in AWS. This solution leverages AWS cloud formation services to provision required resources, services and data lake integration components including S3, Talend Big Data suite, EMR, Redshift to build a data lake solution.

3. Overview

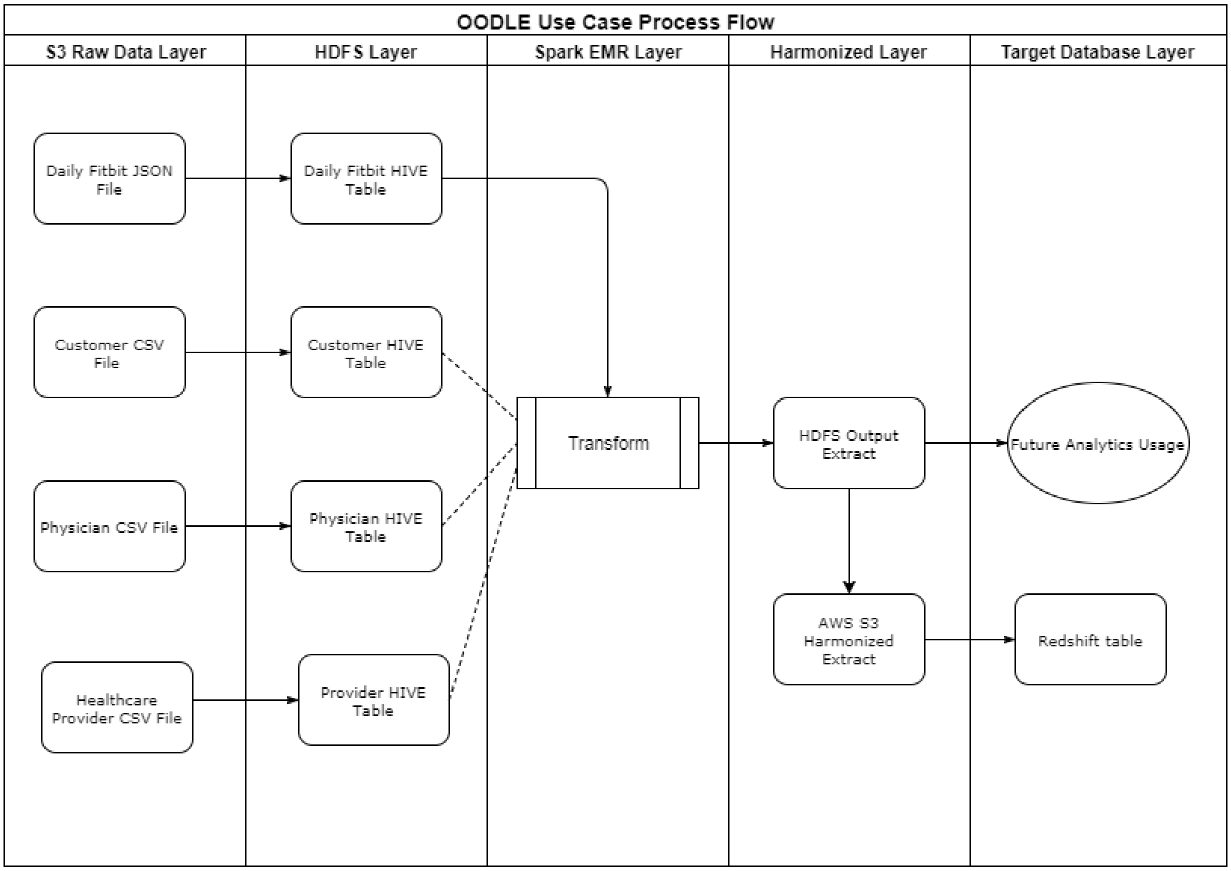
The demo job demonstrates the end to end data lake flow of Data Ingestion and Transformation using sample Talend - EMR jobs leveraging the Spark framework built for a specific customer fitness tracker use case.

The data flow is as follows:

1 Data Ingestion from various types of sources like CSV and JSON to RAW S3 bucket

2 Apply data transformation / Analytics on RAW data using Talend by leveraging EMR spark capabilities.

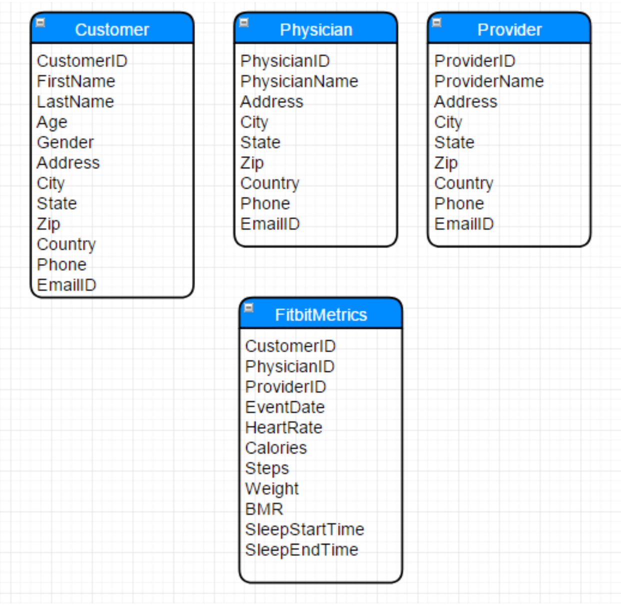
3 Load and build Analytical Data warehouse in Redshift using Talend



4. Talend Demo

4.1. Input Dataset

The jobs process 4 datasets – Customer, Physician, Provider and Fitbit daily feed Metrics. These datasets are sourced from an AWS S3 oodle-raw bucket which is made to be available in public.



Customer

CustomerID
FirstName
LastName
Age
Gender
Address
City
State
Zip
Country
Phone
EmailID

Physician

PhysicianID
PhysicianName
Address
City
State
Zip
Country
Phone
EmailID

Provider

ProviderID
ProviderName
Address
City
State
Zip
Country
Phone
EmailID

FitbitMetrics

CustomerID
PhysicianID
ProviderID
EventDate
HeartRate
Calories
Steps
Weight
BMR
SleepStartTime
SleepEndTime

Fitbit Daily Feed – This contains information about Heartrate, Calories spent, BMR, Weight, Steps, Sleep time, etc in JSON format

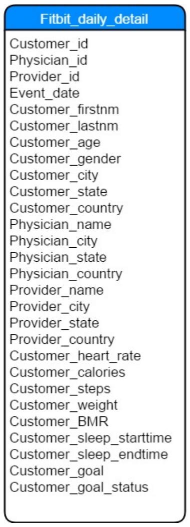
Customer–Customer related information such as Name, Age, Contact and Demographic details in CSV format

Physician– Physician related information with Name, Contact and demographic details. Physician_id in CSV format

Provider– Healthcare Providers such as Provider name, office contact details in CSV format

4.2.Output

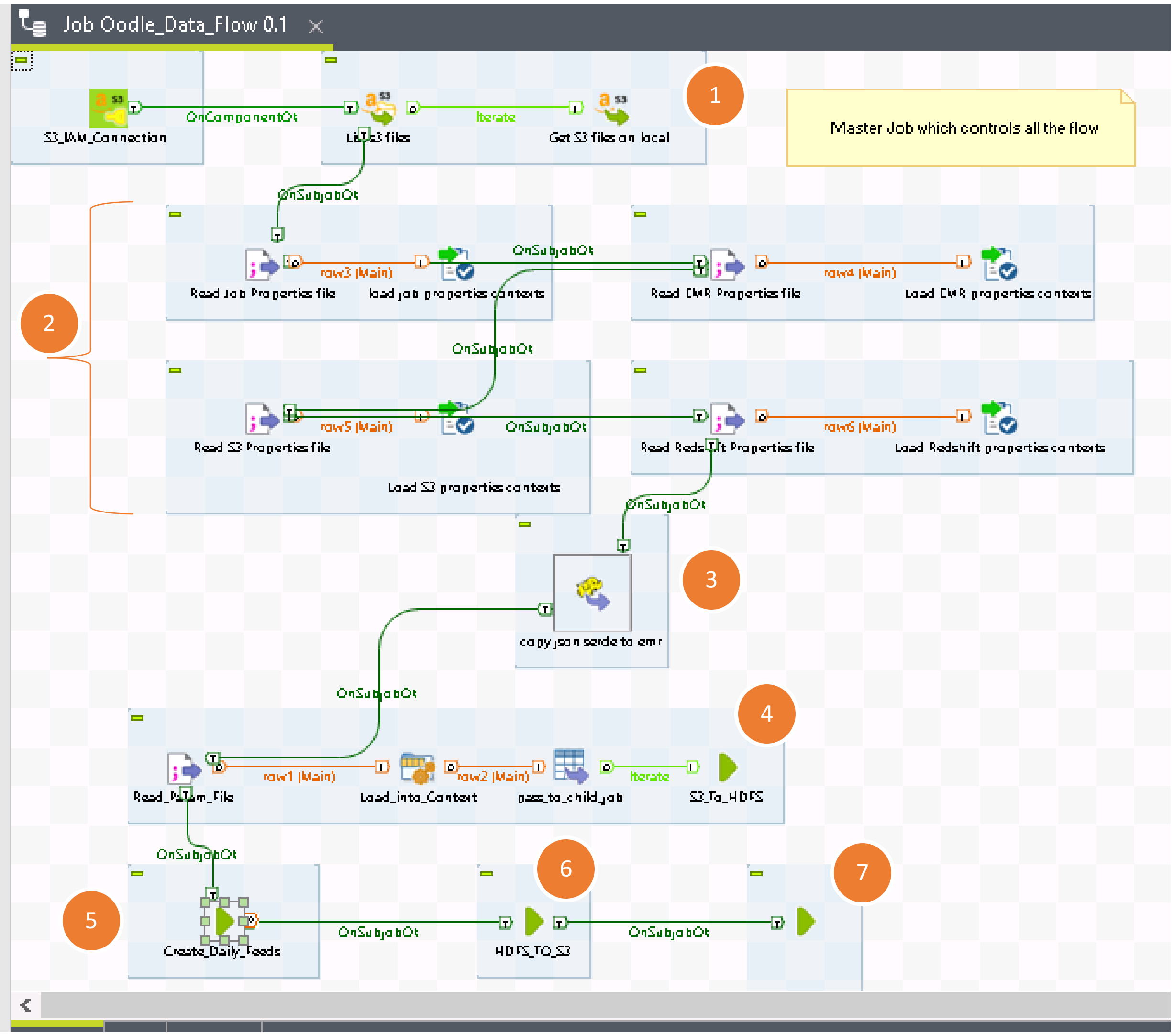
Aggregated fitness which would be built in redshift



Fitbit_daily_detail

Customer_id
Physician_id
Provider_id
Event_date
Customer_firstnm
Customer_lastnm
Customer_age
Customer_gender
Customer_city
Customer_state
Customer_country
Physician_name
Physician_city
Physician_state
Physician_country
Provider_name
Provider_city
Provider_state
Provider_country
Customer_heart_rate
Customer_calories
Customer_steps
Customer_weight
Customer_BMR
Customer_sleep_starttime
Customer_sleep_endtime
Customer_goal
Customer_goal_status

4.3.Talend Job

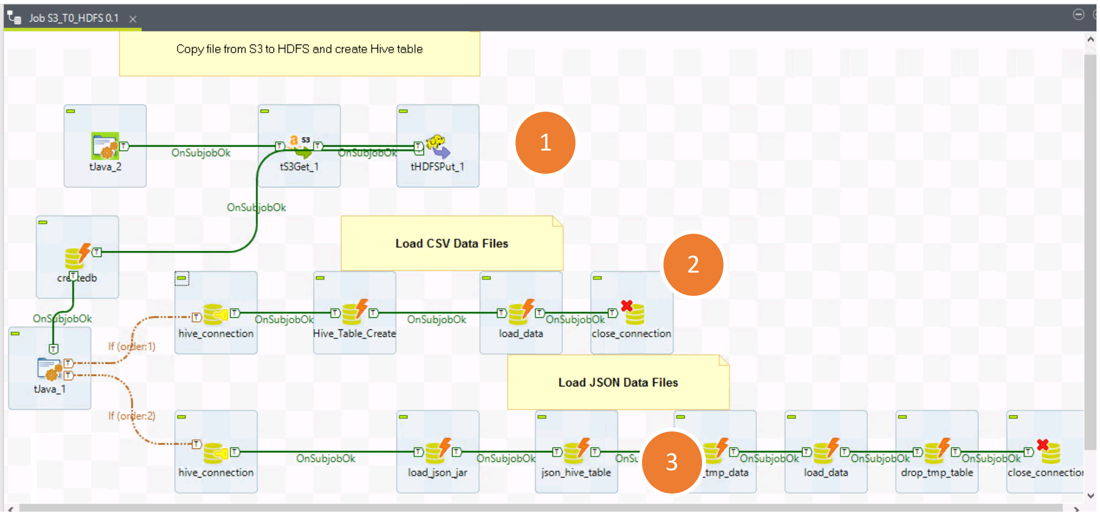


Master Job which controls all the flow

1	Load config files that has metadata info of S3, EMR and redshift from Talend storage S3 bucket to job server using *tS3Get* component
2	Load parameters into context variable using *tContextLoad* component.
3	Copy dependent libraries to EMR (json-serde-1.3.7-jar-with-dependencies.jar) *tHDFSPut* component
4	Load Input data set from S3Sourcebucket to HDFS using ts3get and thdfsput

	component. For More details pls refer section 5.1
5	Perform join and transform data using Talend – Spark framework and load the data into HDFS. For more details pls refer Section 5.2
6	Standard Talend job to copy the load ready files from HDFS to S3Targetbucket.
7	Load data from S3 to redshift using tredshiftrow component. For more details pls refer section 5.3

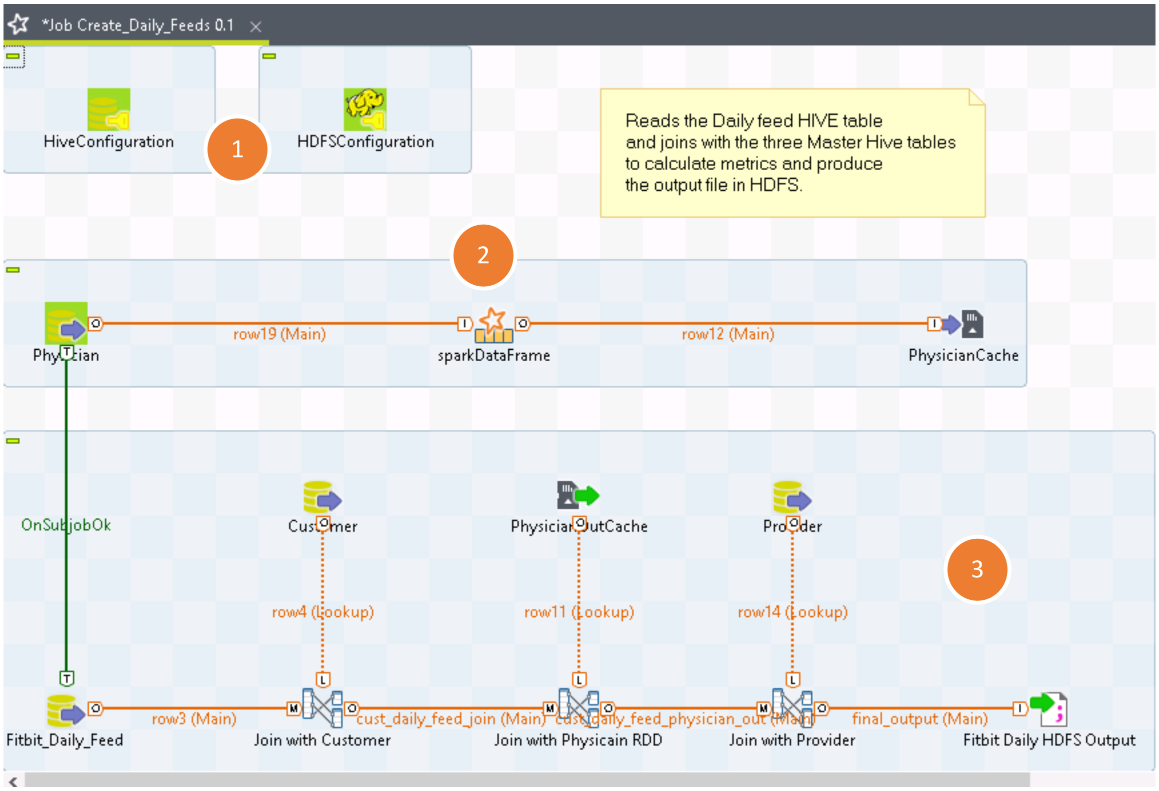
4.4.S3 to HDFS



1	Pushes the input data set which are in CSV and JSON format from S3Sourcebucket to HDFS using ***tS3Get*** and ***tHFDSPut*** component.
2	Create HIVE table and load the input CSV data set into HIVE using **tHiveRow** component
3	Create HIVE table and load the input JSON data set into HIVE using **tHiveRow** component

4.5.Spark Transform Job

This is the Spark job that does all the lookups between the Input extracts, calculates metrics columns for reporting, and creates the Harmonized extract on HDFS. The whole process runs on Amazon EMR cluster.

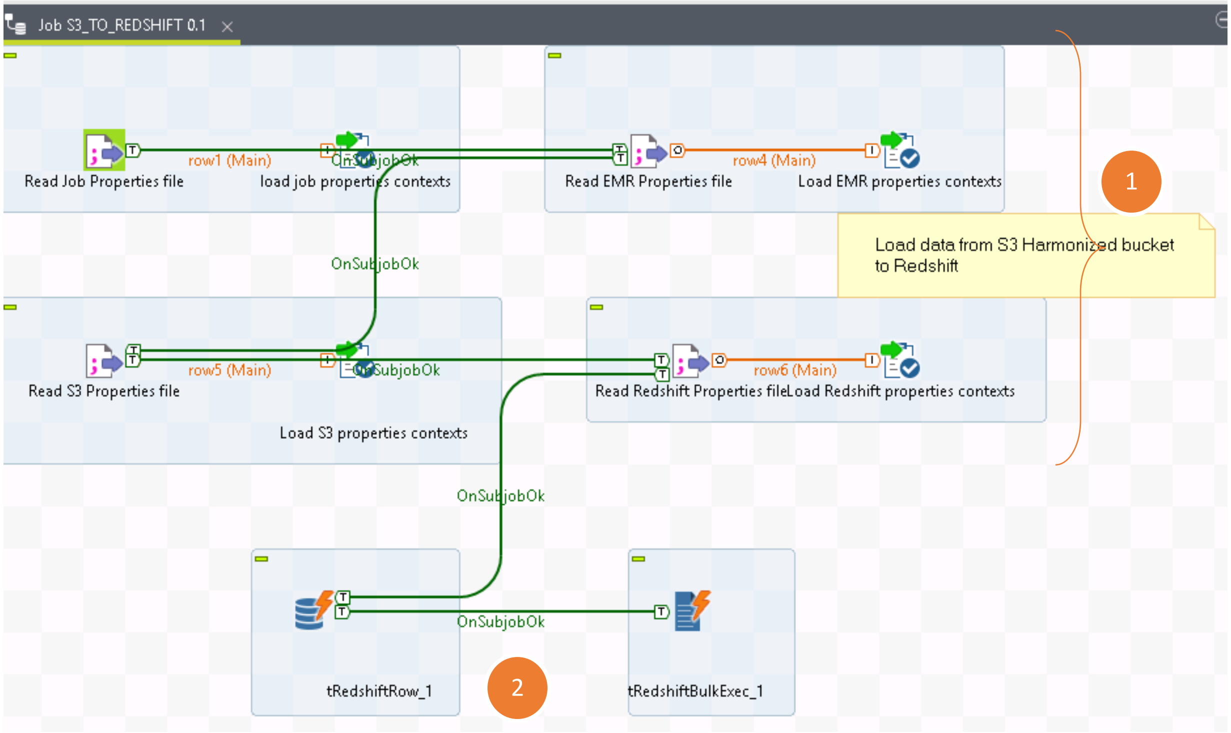


Reads the Daily feed HIVE table and joins with the three Master Hive tables to calculate metrics and produce the output file in HDFS.

1	Set up HIVE and HDFS connection using tHIVEconfiguration and tHDFSconfiguration
2	Create dataframe and persist RDD using tSQLROW and tcacheout component
3	Fitbit HIVE table will be joined and looked up with other HIVE and persisted RDD and transformed using tmap component. Output is written into HDFS using tHDFSOutput component.

4.6.S3 to Redshift Job

This final standard job loads the transformed output file from AWS S3 Harmonized bucket to Redshift table which will be further used for reporting purpose.



1	Reads all the CFT and Job property files and loads the parameters as contexts using *tContextLoad* component.
2	Creates the Redshift table using ***tRedshiftRow*** component, and then loads the final output file on S3 Harmonized bucket to Redshift table using ***tRedshiftBulkExec*** component.

4.7.Job Parameters

input_param.txt – Contains all the input information about Source and Target File location, Filename, File layout (that will be used to create Hive Table Schema), File Type, etc. These will used as context variables inside the Talend jobs.

Apart from this, the following properties files are generated as part of cloudformation template and fed to Talend context variable

oodle-s3.properties –S3 source and Target configuration parameters.

oodle-emr.properties – EMR node name configuration parameters for Talend spark configuration

oodle-redshift.properties – contains Redshift database details

oodle-job.properties – This properties file contains all the additional configuration parameters that are internally used by the talend jobs

Below are the parameters used by Talend jobs

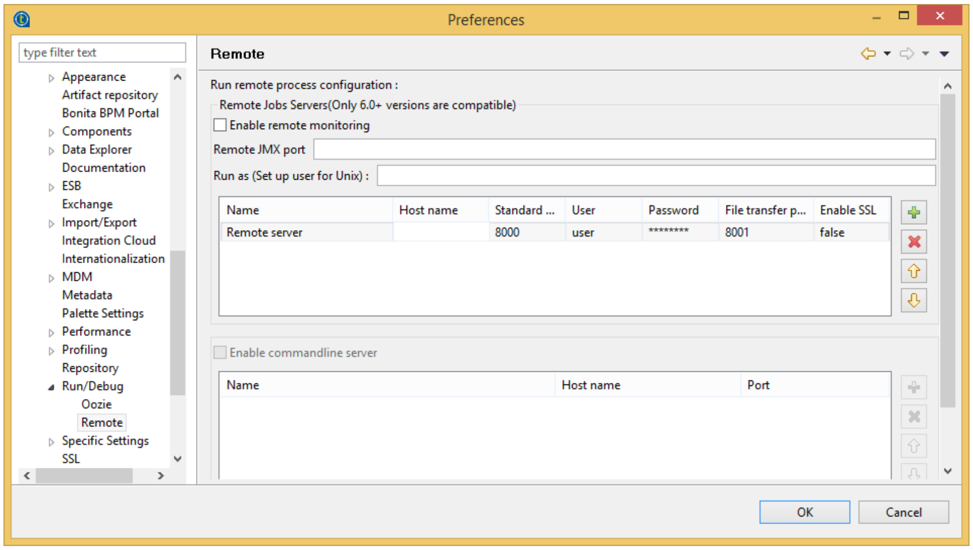
Parameter name	Description	Usage in Jobs
S3_Bucket	Source File S3 Bucket name	Used in *S3_to_HDFS* job
S3_Folder_Name	Source File S3 Folder name	Used in *S3_to_HDFS* job
S3_File_Name	Source Filenames on S3	Used in *S3_to_HDFS* job
HDFS_Output_Path	HDFS file location for each feed	Used in *S3_to_HDFS* job
Hive_DB_Name	Hive Database name	Used in *S3_to_HDFS* job
Hive_Table_Name	Hive Table names for each feed	Used in *S3_to_HDFS* job
Hive_Table_Schema	Hive Table schema for each feed	Used in *S3_to_HDFS* job
Load_Type	File Type to identify load process	Used in *S3_to_HDFS* job
S3_Source_Bucket	Amazon S3 Source bucket	Used in *S3_to_HDFS* job to fetch the Source File location details
S3_Source_Folder_Name	Source file folder location on Amazon S3	Used in *S3_to_HDFS* job to fetch the Source File location details
S3_Target_Bucket	Amazon S3 Target bucket	Used in *HDFS_to_S3*, and *S3_to_Redshift* jobs to fetch the Target File location details
S3_Target_Folder_Name	Target file folder location on Amazon S3	Used in *HDFS_to_S3* and *S3_to_Redshift* jobs to fetch the Target File location details
Hive_Port	Hive Port number	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
Hive_Database	Hive Database name	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
Hive_Username	Hive credential	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
Hive_Password	Hive credential	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive

		connectivity
Hive_Server	Hive Server name on EC2	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
Hive_AdditionalJDBCParameters	Hive Additional JDBC Parameter	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
Hadoop_URI	Hadoop URI details on EC2	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
Hadoop_ResourceManager	Hadoop Resource Manager on EC2	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
Hadoop_ResourceManagerScheduler	Hive Resource Manager Scheduler on EC2	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
Hadoop_JobHistory	Hadoop Jobhistory on EC2	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
Hadoop_username	Hadoop credential	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
Hadoop_STG_DIR	Hadoop Intermediate file location on EC2	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
HDFS_Stg_Output_Path	Hadoop Staging file location on EC2	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
HDFS_Tgt_Output_Path	Hadoop Target file location on EC2	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
HDFS_OutputDailyFeedDir	Hadoop file location on EC2 for Daily Output feed	Used in both *S3_to_HDFS* and *Spark_Transform* job for Hive connectivity
InputParamFileName	Input Parameter File name with absolute Hadoop path	Used in Parent *Master_job* to load the contexts
ConfigParamFileName	Config Parameter File name with absolute Hadoop path	Used in Parent *Master_job* to load the contexts
JsonSerDeJarPath	JSON serde JAR file path on Hadoop	Used in *S3_to_HDFS* job to fetch the daily JSON file details
RedshiftHost	Redshift Host server name	Used in *S3_to_Redshift* job
RedshiftPassword	Redshift credential	Used in *S3_to_Redshift* job
RedshiftDBName	Redshift Database name	Used in *S3_to_Redshift* job
RedshiftPort	Redshift Port number	Used in *S3_to_Redshift* job
RedshiftUsername	Redshift Credential	Used in *S3_to_Redshift* job

5. Step by Step Execution of demo job

5.1. Execution from Talend Studio

1. Connect to X2GO Xwindows studio instance
2. Launch *Talend Studio*.
3. In the *Talend Studio* login window, click the **[...]** button to define a new connection.
4. In the **[Connections]** window that opens, click the **[+]** button to create a new connection.
5. Set the **Repository** type as *Remote* and enter a **Name** and **Description** for the connection, the **E-mail** and **Password** for the user you created in *Talend Administration Center*, and the URL for *Talend Administration Center* (for example, *http://localhost:8080/org.talend.administrator* but, depending on your configuration, you may have to replace <localhost> with the server IP address) in the **Web-app Url** field.
6. Be careful not to use an existing local workspace. If needed, you can create another folder in the Talend Studio alongside the default workspace folder.
7. Click **OK**.
8. You can now select the newly created connection in the *Talend Studio* login window to connect to a collaborative project.
9. Go to windows → preferences → Run/Debug → Remote and update the job server host name details



Remote

Run remote process configuration :

Remote Jobs Servers(Only 6.0+ versions are compatible)

□ Enable remote monitoring

Remote JMX port

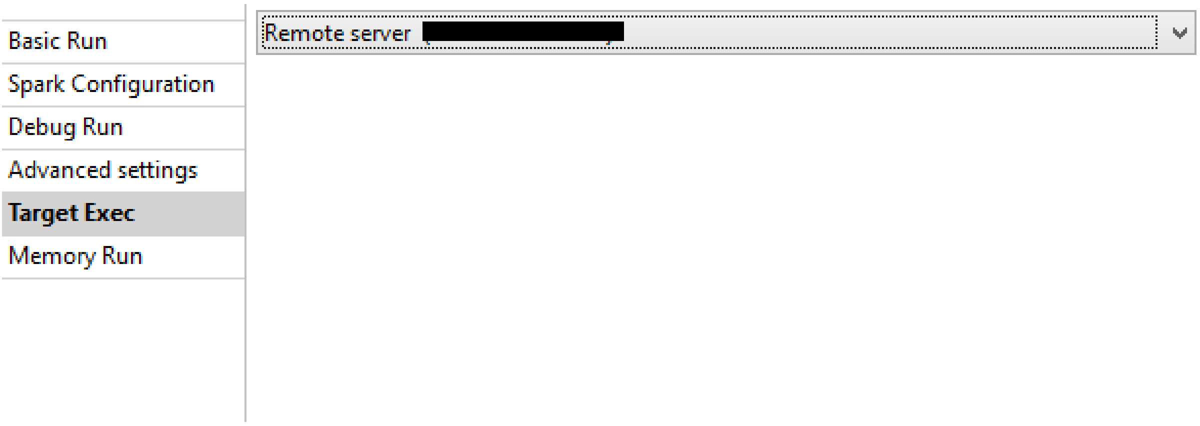
Run as (Set up user for Unix) :

Name	Host name	Standard ...	User	Password	File transfer p...	Enable SSL
Remote server		8000	user	********	8001	false

□ Enable commandline server

Name	Host name	Port

10. Open the OODLE_DATA_FLOW job
11. In the Run tab/view, click on the Target Exec tab, please select the Job Server.

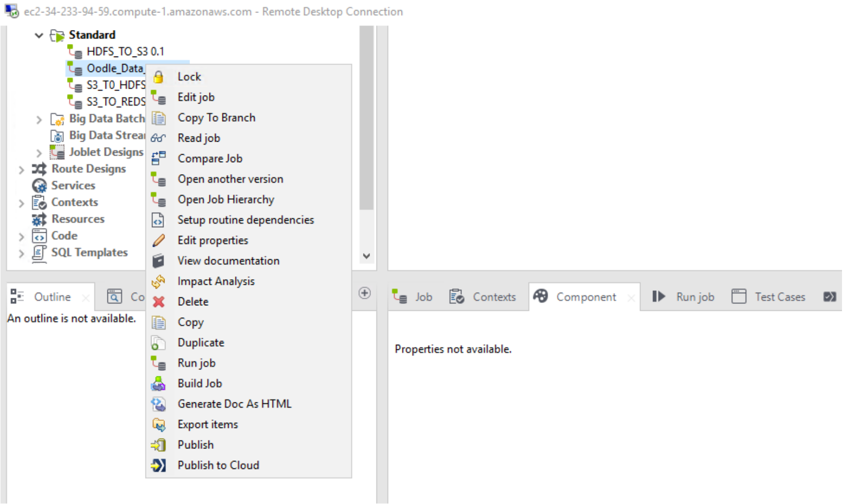


Remote server [redacted]

12. click on the Run button located in the Basic Run tab.
13. Once the execution is OK, pls follow verification steps.

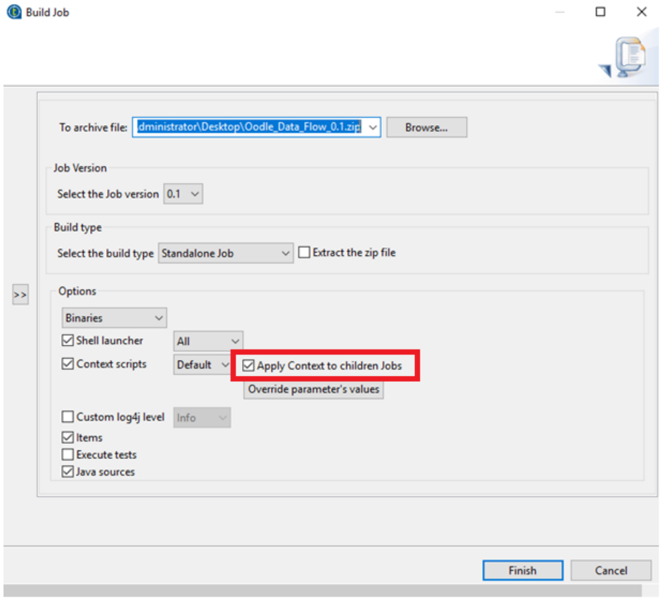
5.2. Execution from TAC

1. Connect to X2GO Xwindows studio instance
2. Launch *Talend Studio*.
3. In the *Talend Studio* login window, click the **[...]** button to define a new connection.
4. In the **[Connections]** window that opens, click the **[+]** button to create a new connection.
5. Set the **Repository** type as *Remote* and enter a **Name** and **Description** for the connection, the **E-mail** and **Password** for the user you created in *Talend Administration Center*, and the URL for *Talend Administration Center* (for example, *http://localhost:8080/org.talend.administrator* but, depending on your configuration, you may have to replace <localhost> with the server IP address) in the **Web-app Url** field.
6. Be careful not to use an existing local workspace. If needed, you can create another folder in the Talend Studio alongside the default workspace folder.
7. Click **OK**.
8. You can now select the newly created connection in the *Talend Studio* login window to connect to a collaborative project.
9. In talend studio, right click OODLE_DATA_FLOW → Build Job → check Apply context to children jobs → click OK



An outline is not available.

Properties not available.



To archive file: dministrator\Desktop\Oodle_Data_Flow_0.1.zip

Job Version

Select the Job version 0.1

Build type

Select the build type Standalone Job ☐ Extract the zip file

Options

Binaries

☑ Shell launcher All

☑ Context scripts Default ☑ Apply Context to children Jobs

Override parameter's values

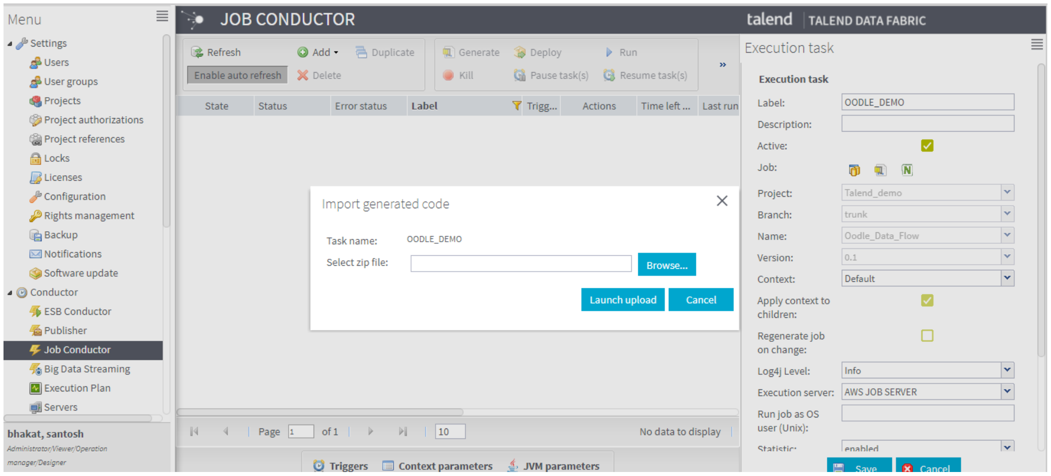
☐ Custom log4j level Info

☑ Items

☐ Execute tests

☑ Java sources

10. Connect to Talend administration console *http://localhost:8080/org.talend.administrator* using the credentials provided in cloud formation. <<you may have to replace <localhost> with the server IP address>>
11. Go to Job Conductor in TAC and click New Normal Job



Import generated code

Task name: OODLE_DEMO

Select zip file:

Execution task

Execution task

Label: OODLE_DEMO

Description:

Active: ☑

Job:

Project: Talend_demo

Branch: trunk

Name: Oodle_Data_Flow

Version: 0.1

Context: Default

Apply context to children: ☑

Regenerate job on change: □

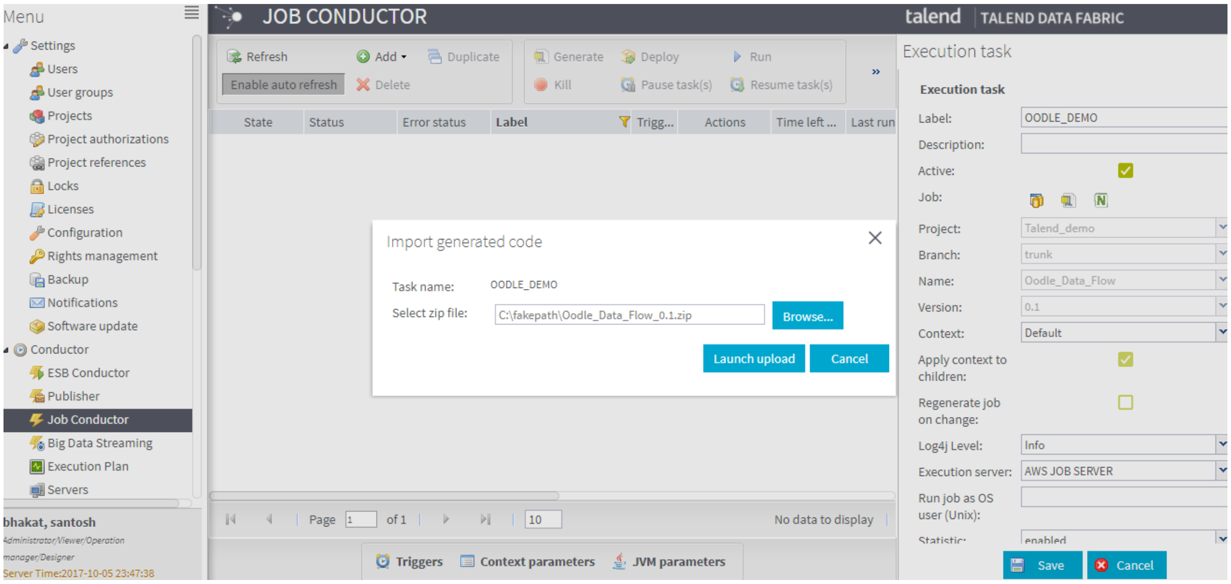
Log4j Level: Info

Execution server: AWS JOB SERVER

Run job as OS user (Unix):

Statistic: enabled

12. Provide Job Name, choose *oodle_demo* projects , Import the build ZIP from studio , select job server and click save



Import generated code

Task name: OODLE_DEMO

Select zip file: C:\fakepath\Oodle_Data_Flow_0.1.zip

Execution task

Execution task

Label: OODLE_DEMO

Description:

Active: ☑

Job:

Project: Talend_demo

Branch: trunk

Name: Oodle_Data_Flow

Version: 0.1

Context: Default

Apply context to children: ☑

Regenerate job on change: □

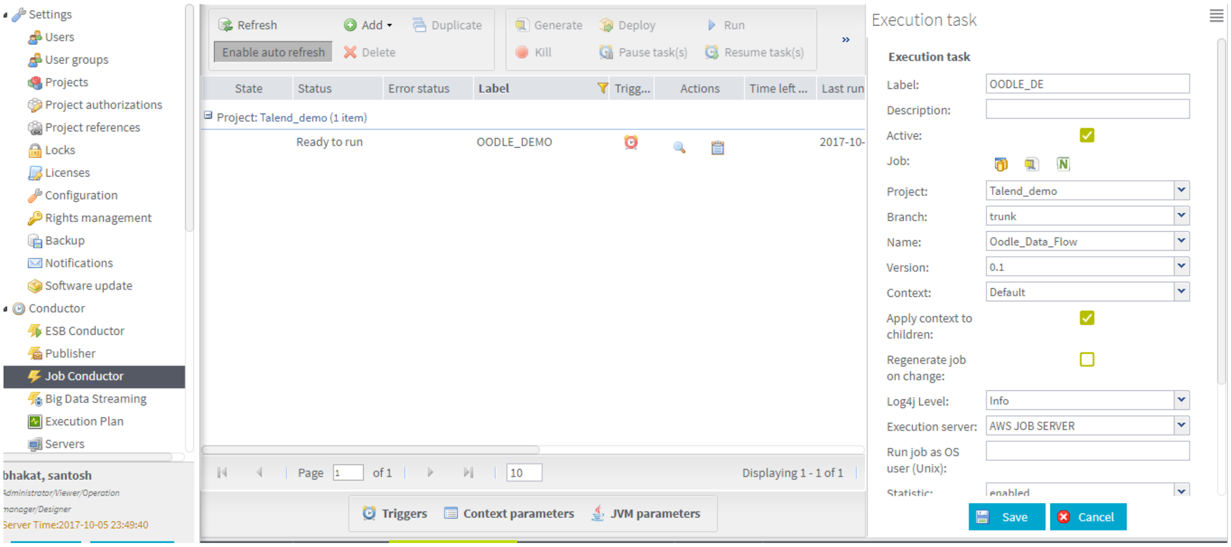
Log4j Level: Info

Execution server: AWS JOB SERVER

Run job as OS user (Unix):

Statistic: enabled

13. Click deploy and run



State	Status	Error status	Label	Trigg...	Actions	Time left ...	Last run
Project: Talend_demo (1 item)							
	Ready to run		OODLE_DEMO				2017-10-

Execution task

Label:	OODLE_DE
Description:	
Active:	☑
Job:	
Project:	Talend_demo
Branch:	trunk
Name:	Oodle_Data_Flow
Version:	0.1
Context:	Default
Apply context to children:	☑
Regenerate job on change:	□
Log4j Level:	Info
Execution server:	AWS JOB SERVER
Run job as OS user (Unix):	
Statistic:	enabled

14. Once the execution is OK, pls follow verification steps

5.3. Verification

1. connect to HUE *http://master public DNS:8888* and check if Customer, Physician, Provider and Fitbit Daily HIVE table are created and loaded with data <<you may have to replace <master public DNS> with the EMR master node DNS>>
2. Check for FinalDailyFeed.txt in S3 Target bucket
3. Connect to redshift db and query Fitbit_Daily_detail to verify the data is properly loaded from FinalDailyFeed.txt table.