2019년 9월 23일 월요일

오후 3:06

"Arn:aws:s3:::ddi-poc-data-origin/raw-json-gzip/\*" --> 이것은 listBucket에 대한 path

"arn:aws:s3:::ddi-poc-data-origin" --> 이것은 getobject에 대한 path

Sid: 그냥 이름 주는 것임

Effect에 들어갈 수 있는 것은 Allow or Deny : 두 개가 같이 쓰여져 있을 때는 Deny가 우선순위를 가짐

시스템 생성 대체 텍스트:
Bucket Policy 
"Version": "2012-10-17" 
"Statement" 
"Sid": "DelegateS3Access" 
'Effect": "Allow" 
"Principal": { 
"AWS": "430079223509" 
''Action": [ 
"s3: ListBucket" 
"s3:GetObjectll 
" ReSOU " : [ 
" a aws : S3: 
a rn : aws : S3 : 
:ddi 
:ddi 
—poc-data-ori son—gzip/*" 
—poc—data—orngin" 

Aws-cli configuration

AKIAWIIVYE3KYTKNQPKA

C: m.]sersflsarang conf igure —prof i le ddioi lot_hsr 
APS Access Key ID [None] AKIAWI IVE-3KYTKNODKA 
APS Secret Access Key [None] I-n-dK6CvgobSOTTcN8hkl.Dn6h8bV/ßMnlcAnMx+rru 
Default reg ion name [None] us—east-I 
Default output format [None] 
tlJsersflsarang han> 

내 access key랑 password찾기

: . han)cd . avs 
: rsWsarang _ han*. awsX_l i r 
9d0d-59AC 
.hanW.aws C.lğ!ıIE+e.l 
019-11-C6 
019-11-C6 
019-11-C6 
019-11-C6 
01 
01 
01 
01 
: *Lise rs*sarang . han*. aws)type 
[prof ile ddipi lot_hsr] 
: *Lise rsWsarang . han*. aws)type 
[ddipi lot_hsr] 
M Conf ig 
12d credent ials 
168 
conf ig 
credent ials 
AKIAWI IVYE3KYWlFKA 

Replication Instance 의 endpoint 생성 중 CDC를 설정 하면 CDC데이터는cdc 폴더에 들어가서

나중에 S3에서 merge하던지 하면 됨

Endpoint identifier Info 
A label for the endpoint to help you identify it. 
s3-datalake-fck-dev-hsr 
Target engine 
The type Of database engine this endpoint is connected to. 
s3 
Service access role ARN 
Role that can access target 
Bucket name 
The name Of an Amazon S3 bucket where DMS Will read the files from 
datalake-ddf-dev 
Bucket folder 
The Amazon SS bucket path where the CSV files can be found 
raw/sales/sales-off/hsr/fact/l 
CDC path 
The Amazon S3 bucket path where the CDC files can be found 
raw/sales/sales-off/hsr/cdc/ 

Header 넣으려면 addColumnName을 true로 해주면 됨

v Endpoint-specific settings 
Extra connection attributes 
Type any additional connection parameters here. See the documentation for more information. 

On-premise 에서 s3로 넘기기

1. Source DB 생성 - Inbound 추가, outbound는 0.0.0.0으로 다 넣어 놓기
2. S3 bucket 생성 하기

--> datalake-ddf-dev/raw/sales/sales-off/edu/fact

1. Role 생성(DMS에서): service access role : dms-vpcRole 이라는 이름으로 롤을 부여함

* dms-vpc ec2 생성권한
* S3 (list, write, delete) 권한

1. Source End point 생성 (DMS에서)
2. Target End point 생성 (DMS에서)
3. Replication Instance 생성 (DMS에서)
4. DMS 작업 생성 4-5-6 연결

시스템 생성 대체 텍스트:
Customers Drive Product Decisions 
presto • 
###甛會嚚 
NETFLIX VDropbox aairbnb 
@ 2019. Amazon Web Services. Inc, or its Affiliates, All rights reserved 
##耑耑耑甛 
AUTODESK 
NOKIA ROVIO 

Presto는 in-memory 기반이라 빠름 ; 단 메모리 위에 올라가서 메모리 usage에 대한 제약이 있음

Hive: memory랑 disc를 둘 다 사용함. Presto 단점 보완

시스템 생성 대체 텍스트:
Amazon 
Kinesis Data 
Firehose 
Amazon 53 
Automate routine 
tasks such as data 
cleansing 
AWS Glue 
Pe「fo爾1 unique data 
transformations and 
ML 
Amazon EMR 

Glue는 node수를 정해서 etl을 해야하지만, EMR은 Spot instance를 띄워서 저렴한 비용에 빠른 속도로 etl을 할 수 있다는 장점이 있음

Glue ETL notebook 코드

import sys

from awsglue.transforms import \*

from awsglue.utils import getResolvedOptions

from pyspark.context import SparkContext

from awsglue.context import GlueContext

from awsglue.job import Job

## @params: [JOB\_NAME]

args = getResolvedOptions(sys.argv, ['JOB\_NAME'])

sc = SparkContext()

glueContext = GlueContext(sc)

spark = glueContext.spark\_session

job = Job(glueContext)

job.init(args['JOB\_NAME'], args)

## claim, sales, region\_claim table

datasource0 = glueContext.create\_dynamic\_frame.from\_catalog(database = "xxx", table\_name = "xxx", transformation\_ctx = "datasource0")

xxx = glueContext.create\_dynamic\_frame.from\_catalog(database = xxx, table\_name = xxx, transformation\_ctx = xxx)

xxx = glueContext.create\_dynamic\_frame.from\_catalog(database = xxx, table\_name = xxx, transformation\_ctx = xxx)

dropnullfields0 = DropNullFields.apply(frame = datasource0, transformation\_ctx = "dropnullfields0")

xxx= DropNullFields.apply(frame =xxx, transformation\_ctx = xxx)

xxx = DropNullFields.apply(frame = xxx, transformation\_ctx = xxx)

# bucket name 및 key 확인

datasink0 = glueContext.write\_dynamic\_frame.from\_options(frame = dropnullfields0, connection\_type = "s3", connection\_options = {"path": "s3://datalake-ddf-dev2/clean/sales/sales-off/edu/fact/claim"}, format = "parquet", transformation\_ctx = "datasink0")

datasink1 = glueContext.write\_dynamic\_frame.from\_options(frame = xxx, connection\_type = "s3", connection\_options = {"path": "s3://datalake-ddf-dev2/clean/sales/sales-off/edu/fact/sales"}, format = xxx, transformation\_ctx = "datasink1")

datasink2 = glueContext.write\_dynamic\_frame.from\_options(frame = xxx, connection\_type = "s3", connection\_options = {"path": "s3://datalake-ddf-dev2/clean/sales/sales-off/edu/fact/region\_claim"}, format = xxx, transformation\_ctx = "datasink2")

job.commit()

시스템 생성 대체 텍스트:
AWS Glue 
Data Catalog 
國_Z_夐 
Amazon 53 
Amazon 
Athena 
Use PyAthena to query 
Athena tables directly 
from Amazon SageMaker 
notebooks 
Jupyter 
Amazon 
SageMaker 

PyAthena library를 install 하여 athena query를 날릴 수 있음

Athena를 이용한다는 것은 boto3 client를 이용한다는 것…

Athena API를 이용하면 내가 저장 위치를 바꿀 수 있음

StartQueryExecution client.startQueryExecution({ QueryString: ‘SELECT \* FROM table\_name LIMIT 100’, ResultConfiguration: { **OutputLocation: ‘s3://bucket/output/’** }, EncryptionConfiguration: { EncryptionOption: 'SSE\_S3' }, QueryExecutionContext: { Database: ‘default\_db’ } }, (err, result) => {})

시스템 생성 대체 텍스트:
Any Athena metric: successful/failed & total queries, query run time, etc. 
Data limit 
10 Gigabytes 
1 Terabytes 
: 10 Gigabytes 
Time period 
Not applicable 
24 hours 
1 hour 
Action 
Query will be cancelled. 
Send notification 10 topic : am:aws:sns:us-east-l :9 
Send notification to topic . am:aws:sns:us-east-l :9 
9:AthenaAlarm : 
9:AthenaAlarm • 

Time period가 30min으로 시간 제한이 되어있음 -> server 1000대로 돌리는거라 30min넘는 경우가 거의 없을 듯

Partition 예시) If you want partition by month, lambda를 만들어서 주기적으로 partition을 월별로 하는 코드를 짜야됨

시스템 생성 대체 텍스트:
Layout Of data fo「 partitioned tables 
CREATE EXTERNAL TABLE access—I 
ip—address string, 
request_time Timestamp , 
PARTTIONED by 
(string year, 
stri ng month) 
STORED AS PARQUET 
LOCATION ' 53 ://yourBucket/patt#0Tab1e/ | 
33 : / /yourBucket/pathToTab1e /year=2019/month=11 / 
Alter Table access—logs 
Add Partition 
year=ㆍ2019', 
month='110 
location 
's3://yourBucket/pathToTable/yea 「=2019/month=11/' 
MSCK Repair Table 
GLUE API 

Alter Table access\_logs Add Partition year=‘2019', month=‘11‘) location ‘s3://yourBucket/pathToTable/year=2019/month=11/’ 또는

MSCK Repair Table 테이블명

GLUE API

Parquet파일일 때는 s3에서 athena를 통해 scan해서 가져온 다음에 거기에 sagemaker를 붙이는 게 좋음

시스템 생성 대체 텍스트:
Que「y Performance - Joins 
Dataset: 74 GB total data, uncompressed, text format, ~602M rows 
Que『y 
SELECT count(s) FROM part KHERE lineiten.l_partkey • part•p—partkey 
SELECT count(S) FROM part, lineiten "HERE lineiten.l_partkey • part.p_partkey 
Saⅶ195 / Speedup 
Keep the larger Table on the Left Side of the join 
Run time 
22.81 seconds 
10.71 sec에ds 
-530/0 speed up 

Spark이나 hive에서는 왼쪽이 큰 테이블이어야함

Presto query 문법

SELECT

current\_date AS today,

date(split\_part(record\_timestamp\_new,' ',1)) AS date,

pstatenumber\_i,

count(pstatenumber\_i) AS count\_pstate

FROM fck\_rdne\_rms.clean\_rmslog2

WHERE date(split\_part(record\_timestamp\_new,' ',1))

BETWEEN current\_date - interval '5' month

AND current\_date - interval '30' day

AND pstatenumber\_i >=150

GROUP BY date(split\_part(record\_timestamp\_new,' ',1)) ,pstatenumber\_i

ORDER by date,pstatenumber\_i

Hadoop

--> 큰 데이터가 들어오면 64 or 128로 쪼개고(64로 쪼개는게 default), 여러 개의 nodes에 분산하여 복제 저장함

* Map reduce: mapping하고 reduce하는 것

--> reduce한다 함은 한쪽으로 모은다는 것

* Hive가 가장 먼저 나옴
* Spark은 in-memory분산저장 처리 시스템

Amazon EMR node types

1. Master node: The node that manages the cluster. The master node tracks the status of tasks and monitors the health of the cluster.
2. Core nodes: The node that runs tasks and stores data in the Hadoop Distributed File System (HDFS) on your cluster. --> slave역할: 일하는 node
3. Task nodes: The node that only runs tasks and does not store data in HDFS. Task nodes are optional. --> EMR에만 있음

--> core node와 task nodes 차이: task nodes는 스토리지가 없음

--> C4에서..

C: computing 타입이라 cpu가 큼

R: 메모리타입

M: general 타입

시스템 생성 대체 텍스트:
8XL 
4XL 
2XL 
0.5 
0.21 
(ㆍ1및 
0.0 
0.01 
0.2 
0.3 
0.07 
0.0 
0.01 
$0.29 
$0.16 
$0.08 
$0.04 
$0.04 
On 
Demand 
$1.76 
$0.88 
$0.44 
$0.22 
$0.11 

Pig: etl

Oozie: workflow management

Hue: 개발 환경 구성

최소 spark환경을 구성하려면 livy, hadoop, spark을 체크해야함

시스템 생성 대체 텍스트:
Ganglia 3 7 2 
Hive 2 36 
MXNet 1.5.1 
Spark 2.4 4 
Multiple master nodes (optional) 
HBase 14.10 
Presto 0 227 
Sqoop 1 4.7 
Phoenix 4 14.3 
HCatalog 23.6 
Pig 0.170 
ZooKeeper 3.4.14 
Mahout 0.13.0 
Oozie 5.1.0 
TensorFlow 1.14.0 
[二〕 Use multiple master nodes to improve cluster availability. Learn more 口 
AWS Glue Data Catalog settings (optional) 
2〕 
Use for Hive table metadata 0 
Use for Presto table metadata 0 
Use for Spark table metadata 0 

Glue data catalog를 꼭 선택해야지 glue에 있는 catalog를 보고 데이터 불러 올 수 있음

spark.sql("show databases").show()

spark.sql("use fck\_rdne\_rms")

saprk.sql("show tables").show()

df = spark.sql("select \* from clean\_rmslog2 limit 10")

df.show(2)

[EMR CREATE 할 때]

Edit software settings O 
Enter configuration Load JSON from S3 
s3://ddi-dataIake-library2/emr/emr-config-sungyoul.jso 

maximizeResourceAllocation: true로 해야지, 큰 데이터 돌릴 수 있음

Spark 분석했다가, python로 가고 싶으면 path도 정해줘야함 (아래 참고)

[  
 {  
 "Classification": "spark",  
 "Properties": {  
 "maximizeResourceAllocation": "true"  
 }  
 },  
 {  
 "Classification": "spark-env",  
 "Properties": {  
 "PYSPARK\_PYTHON": "/usr/bin/python3"  
 }  
 }  
]

Redshift Cluster Architecture 
Massively parallel, shared nothing architecture 
Streaming Backup/Restore from S3 
SQL Clients / 
Bl Tools 
t JDBC/ODBC 

Shared nothing architecture: memory위에 올라가지 않고 각자 cpu를 가지고 개별적으로 돌아가는 것 --> 대표적인 예로 hadoop이 있음

Redshift Distribution Styles 
Four distribution styles to choose from in Redshift 
KEY 
ALL 
EVEN 
AUTO 
A column value is hashed, and the 
same hash value is placed on the same 
slice 
Full table data is placed on each 
compute node's first slice 
Data is evenly distributed across all 
slices using a round-robin distribution 
Default option; Redshift starts the 
table with ALL, but switches the table 
to EVEN when the table grows larger 

ALL : COPY해서 COMPUTE NODE에 동일하게 뿌림

--> 특히 JOIN을 많이 하는 테이블일 때, 유용하게 사용 됨

Skewed 가 생기는 이유--> key distribution 이 많을 때 key 옵션을 쓰면 좋은데 그게 아닌 경우에 key 옵션을 쓰면 Skewd가 생길 수 있음

Redshift Cluster Setup

* + Create group and users :
  + password include Capital Letter : ex)Sungyoul

create group powersuers;

create user jh password 'Imjangho123' in group powersuers;

grant all on database dev to group powerusers;

alter user {yourname} createuser;

* + Create Schema
  + Log in redshift with your user name and password

create schema fckrms\_{yourname};

create schema ddfedu\_{yourname};

Redshift 쿼리 예제

create external schema ext\_fckrms\_hsr3

from data catalog database 'fck\_rdne\_rms'

iam\_role 'arn:aws:iam::442364814664:role/ddi-redshift-role';

create external schema ext\_fckrms\_hsr7

from data catalog database 'fck\_rdne\_rms'

iam\_role 'arn:aws:iam::442364814664:role/ddi-redshift-role';

create schema ddf\_edu\_hsr;

CREATE TABLE ddf\_edu\_hsr.claim(

claim\_no bigint,

product\_no bigint,

d\_region varchar,

model varchar,

customer\_no varchar)

DISTKEY(customer\_no);

CREATE TABLE ddf\_edu\_hsr.sales(

claim\_no bigint,

product\_no bigint,

d\_region varchar,

model varchar,

customer\_no varchar)

DISTKEY(customer\_no);

CREATE TABLE ddf\_edu\_hsr.region\_claim(

claim\_no bigint,

product\_no bigint,

d\_region varchar,

model varchar,

customer\_no varchar)

DISTKEY(customer\_no);

copy ddf\_edu\_hsr.claim

from 's3://datalake-ddf-dev2/clean/sales/sales-off/edu/fact/claim/'

iam\_role 'arn:aws:iam::442364814664:role/ddi-redshift-role'

format as parquet;

copy ddf\_edu\_hsr.region\_claim

from 's3://datalake-ddf-dev2/clean/sales/sales-off/edu/fact/region\_claim/'

iam\_role 'arn:aws:iam::442364814664:role/ddi-redshift-role'

format as parquet;

copy ddf\_edu\_hsr.sales

from 's3://datalake-ddf-dev2/clean/sales/sales-off/edu/fact/sales/'

iam\_role 'arn:aws:iam::442364814664:role/ddi-redshift-role'

format as parquet;

unload (

'select \* from ddf\_edu\_hsr.sales'

) to 's3://datalake-ddf-dev2/report/sales/edu/sample\_parquet/'

iam\_role 'arn:aws:iam::442364814664:role/ddi-redshift-role'

ALLOWOVERWRITE

maxfilesize 100MB

parquet ;

CDC: Change Data Chapture

EMR로 hadoop 띄워서 pandas랑 왔다갔다 하면서 분석하기

import pandas as pd

from pyathena import connect

from pyspark.sql import functions as F

from pyspark.sql.functions import col, countDistinct, approx\_count\_distinct

# approx\_count\_distinct:는 pyspark에 있는 함수로 대충 상위 몇 개만 추출해서 distinct 하는것

# countDistinct가 함수중에 제일 오래 걸림

spark.sql("show databases").show()

spark.sql("use fck\_rdne\_rms")

spark.sql("show tables").show()

sparkDF = spark.sql("select \* from fck\_rdne\_rms.clean\_rmslog2")

#df = spark.sql("select \* from fck\_rdne\_rms.clean\_rmslog2 limit 1000000").toPandas()

# pandas data frame 형태로 보내기 --> 작은 데이터로 다 필터한다음에 하면 좋음

# 왜냐하면 notebook 메모리에 그대로 올라가서 24xlarge 타입을 써야되기 때문에 작게 만들어서 toPandas로 보내버림

sparkDF.select(sparkDF.columns[0:8]).show(4)

def dropNullColumns(df):

# A set of all the null values you can encounter

null\_set = {"none", "null" , "nan"}

# Iterate over each column in the DF

for col in df.columns:

# Get the distinct values of the column

unique\_val = df.select(col).distinct().collect()[0][0]

# See whether the unique value is only none/nan or null

if str(unique\_val).lower() in null\_set:

print("Dropping " + col + " because of all null values.")

df = df.drop(col)

return(df)

from pyspark.sql.functions import col, countDistinct, approx\_count\_distinct

def countDistinctColumns(df):

ans = df.agg(\*(approx\_count\_distinct(col(c)).alias(c) for c in df.columns))

return(ans)

def strColumns(df):

StrColList = [item[0] for item in df.dtypes if item[1].startswith('string')]

return(StrColList)

def doubleColumns(df):

FloatColList = [item[0] for item in df.dtypes if item[1].startswith('double')]

return(FloatColList)

def intColumns(df):

intColList = [item[0] for item in df.dtypes if item[1].startswith('int') or item[1].startswith('bigint')]

return(intColList)

def numberColumns(df):

numberColList = [item[0] for item in df.dtypes if item[1].startswith('double') or item[1].startswith('bigint') or item[1].startswith('int') ]

#numberColList = [item[0] for item in df.dtypes if item[1].startswith('double') ]

return(numberColList)

def booleanColumns(df):

boolColList = [item[0] for item in df.dtypes if item[1].startswith('bool')]

sparkDF\_distinct = countDistinctColumns(sparkDF).toPandas()

col\_0\_1 = []

for col in sparkDF\_distinct.columns:

#print(df\_distinct[col])

if int(sparkDF\_distinct[col]) in [0,1]:

if col.find('statenumber') == -1 :

# column명에 statenumber 가 있으면 제외하지말기

col\_0\_1.append(col)

print("total 0,1 columns :",len(col\_0\_1), "\n",sparkDF\_distinct[col\_0\_1[0:5]])

sparkDF2 = sparkDF.drop(\*col\_0\_1).orderBy("record\_timestamp")

sparkDF2.select(sparkDF2.columns[0:5]).show(4)

len(sparkDF2.columns)

sparkDF2.select('record\_timestamp\_new').show()

sparkDF2.write.option("header", "true").mode("overwrite").option("compression","gzip").json("s3a://ddi-emr-analysis/sungyoul/2019-12-08/sample/data\_without\_disctinct\_0\_1.json")

sparkDF2\_string = sparkDF2.select(\*strColumns(sparkDF2))

sparkDF2\_number = sparkDF2.select(\*numberColumns(sparkDF2))

sparkDF2\_int = sparkDF2.select(\*intColumns(sparkDF2))

#sparkDF2\_boolean = sparkDF2.select(\*booleanColumns(sparkDF2))

from pyspark.sql.types import DoubleType

for c in sparkDF2\_int.columns:

sparkDF2\_number = sparkDF2\_number.withColumn(c,sparkDF2\_number[c].cast(DoubleType()))

# withColumn(new\_column\_name, orig\_col\_name)

from pyspark.mllib.stat import Statistics

col\_names = sparkDF2\_number.columns

features = sparkDF2\_number.rdd.map(lambda row: row[0:])

sparkDF2\_corr\_mat=Statistics.corr(features, method="pearson")

corr\_df = pd.DataFrame(sparkDF2\_corr\_mat, index = col\_names,columns = col\_names)

corr\_df.iloc[0:5,0:3]

corr\_df.to\_csv("s3n://ddi-emr-analysis/sungyoul/2018-12-08/sample/corr\_perason\_results.csv",header=True,index=True)

a = corr\_df.filter(like = "pstatenumber",axis=0)

target = 0.7

for k in a.columns:

#print(k)

if float(a[k]) >= target or float(a[k]) <= -target :

print(k,float(a[k]))

PLSQL

CREATE OR REPLACE PROCEDURE public.get\_results\_set(

param integer,

INOUT temp\_name character varying(256)

) AS $$

DECLARE row RECORD;

BEGIN

execute 'drop table if exists ' || temp\_name;

execute 'create temp table ' || temp\_name || ' as select c.claim\_no, c.product\_no,

d\_region, model, customer\_no, claim\_reason,used\_mileage,used\_day,claim\_month

from ddfedu\_sungyoul.claim as c , ddfedu\_sungyoul.reason\_claim

as r

where c.claim\_no = r.claim\_no limit ' || param;

END;

$$ LANGUAGE plpgsql;

call get\_results\_set(10, 'results');

select \* from results;

Redshift : Query the System Tables

* + <https://docs.aws.amazon.com/ko_kr/redshift/latest/dg/c_intro_STL_tables.html>
  + <https://docs.aws.amazon.com/ko_kr/redshift/latest/dg/c_intro_system_views.html>
  + **View a List of Table Names and users**

select \* from SVV\_TABLES where table\_schema ='ddfedu\_sungyoul';

select \* from pg\_user where usesysid > 1;

* + **View Recent Queries**

select \* from pg\_user where usesysid > 1;

select query, pid, elapsed, substring

from svl\_qlog

where userid = {your\_userid}

order by starttime desc

limit 10;

* + **Determine the Process ID of a Running Query**

select \* from stv\_recents limit 10;

select pid, user\_name, starttime, query

from stv\_recents where status='Running';

* + **Redshift : Cancel the query**

select pid, trim(user\_name), starttime, substring(query,1,20)

from stv\_recents where status='Running';

cancel {pid\_number} ;

* + **Redshift: Check for skewness in table**

select "table", diststyle, skew\_rows from svv\_table\_info

where "table" IN ('claim', 'reason\_claim', 'sales');

* + **Redshift: Error check during loading data from s3 to Redshift**
  + <https://docs.aws.amazon.com/ko_kr/redshift/latest/dg/r_STL_LOAD_ERRORS.html>

select d.query, substring(d.filename,14,20),

d.line\_number as line,

substring(d.value,1,16) as value,

substring(le.err\_reason,1,48) as err\_reason

from stl\_loaderror\_detail d, stl\_load\_errors le

where d.query = le.queryand d.query = pg\_last\_copy\_id();

create view loadview as

(select distinct tbl, trim(name) as table\_name, query, starttime,

trim(filename) as input, line\_number, colname, err\_code,

trim(err\_reason) as reason

from stl\_load\_errors sl, stv\_tbl\_perm sp

where sl.tbl = sp.id);

select table\_name, query, line\_number, colname, starttime, trim(reason) as error

from loadview

where table\_name = 'sales'

order by line\_number

limit 1;

6주차

Spark언어는 2가지로 나뉨

1. Transformation: read/sort/group by 등의 작업을 하는 것

--> 쿼리를 날려도, 실행 되지 않고, show/count라고 했을 때 그 로직을 처리함

1. Action: write/show/count

--> 쿼리 날린 즉시 실행 됨.

AWS교육

active active vs. active standby

object storage = csv file 자체로 들어감 -> script측면에 대해서 organizing이 필요함

아마존 philosophy : storage와 computing을 분리함: segragation between storage and computing

--> decoupled system: data(kinesis) --> storage(S3) --> PROCESS(EMR) --> STORE(S3) --> ANALYZE(SAGE MAKER) --> ANSWER

when you need high performance computing, EMR cluster를 띄워서 spark으로 분석하고 닫으면 됨

amazon glacier: archiving 하는 용도 : s3 데이터를 glacier로 compression해서 넘김

ATHENA는 PRESTO 엔진을 사용함

AD-HOC ANALYSIS??

shards: 데이터를 쓸때 병목현상을 다루는 방법

spark datastream: 실시간

spark dataframe: sql

spark: 분산처리시스템

CPU: 메모리로 돌아가는 알고리즘도 있고

GPU: 연산이 많이들어가는 알고리즘은 GPU로 쓰면됨

단, 모든 알고리즘이 GPU에서 돌아가지 않음. 즉, GPU에서 안 돌아가는 것들도 있음

시스템 생성 대체 텍스트:
AWS 
VPC 
VPC Subrwt 
Availab Itty 20,10 
VPC Subnet 
| Avalab•lityZone I 
Router 
Virt」1 Pfi 
Gateway 
VPN 
Connection 
Cu보에℃ Gateway 

VPC > subnet > EC2

--> VPC subnet은 private 모드, public모드로 만들 수 있으며, pirvate IP로 설정하면 내부로만 나가지 외부로는 나갈 수 없음

Virtual Private Gateway 는 VPC의 VPN역할을 함

Virtual Private Gateway가 있어야 key 생성이 가능함

Subnet은 이중화가 되어야 돼서 2개 이상이어야함.

* + CIDR : 10.248.254.64/27 --> 32개 의 ip를 받을 수 있음
    - 32 - 27 = 5 , 2^5=32
  + 2 Subnets 생성 :
  + vpn-subnet-1 : 10.248.254.64/28 --> 64부터 16개로 32개를 반으로 쪼갬
  + vpn-subnet-2 : 10.248.254.80/28 --> 80부터 16개로 32개를 반으로 쪼갬

출처: <[*https://ddi-proserve.atlassian.net/browse/LAKE-86*](https://ddi-proserve.atlassian.net/browse/LAKE-86)>

시스템 생성 대체 텍스트:
Customer Gateways > Create Customer Gateway 
Create Customer Gateway 
Specify the Internet-routable IP address for your gateway's external interface the address must be static and may 
also specify your gateway's Border Gateway 하Otoc이 (BGP) Autonomous System Number (ASN); this can be eithe 
Name 
Routing 
IP Address 
Certificate ARN 
cc-test-hsfl 
0 
Dynamic 
㉭ Static 
99. 7.7.7.7 
Select Certificate ARN 
0 
0 
C 
0 

IPsec #1

set remote-gw 3.93.90.174

set psksecret mxqAYAjyp.Dd.MJihdLuW4EhB3sVXL2u

시스템 생성 대체 텍스트:
Streaming with Amazon Kinesis 
Easily collect, process, and analyze video and data streams in real time 
Kinesis Video 
St「eams 
Capture, process, and 
store video streams 
Kinesis Data 
Streams 
Capture, process, and 
store data streams 
Kinesis Data 
Firehose 
Load data streams into 
data stores 
0 
Kinesis Data 
Analytics 
Analyze data streams 
with SQL 

Kinesis data streams 은 1일, 최대 7일 까지 data 저장가능 (비용차) - bandwith가 flexible함

Kinesis data firehose는 1일만 - bandwith가 kinesis data streams처럼 dynamic하게 flexible하지 않음, 단 lamda랑 붙어서 처리가 가능함

'

시스템 생성 대체 텍스트:
Amazon Kinesis Data Analytics application 
streaming SOLIrCe 
Amazon 53 
In-application stream 
In-application table 
SQL code joining 
table and stream 
destination 

'

In-application table : in memory 위에 올라감

**import boto3**  
**import botocore**  
**import sagemaker**  
**import sys**

출처: <[*https://hsr.notebook.us-east-1.sagemaker.aws/examples/preview?example\_id=%2Fhome%2Fec2-user%2Fsample-notebooks%2Fintroduction\_to\_amazon\_algorithms%2Frandom\_cut\_forest%2Frandom\_cut\_forest.ipynb*](https://hsr.notebook.us-east-1.sagemaker.aws/examples/preview?example_id=%2Fhome%2Fec2-user%2Fsample-notebooks%2Fintroduction_to_amazon_algorithms%2Frandom_cut_forest%2Frandom_cut_forest.ipynb)>

5주차

Lambda: 15분동안만 작업 가능한 것 --> timeout ; 의 default는 3초

시스템 생성 대체 텍스트:
AWS Lambda 
Dashboard 
Applications 
Functions 
Layers 
X 
Resources for US East (N. Virginia) 
Lambda function(s) 
2 
Full account concurrency 
1000 
Create function 

기본적으로 한 account당 1000개의 lambda가 뜰 수 있음 (증설 가능)

Lambda 패키징 rule!

Python이라는 라이브러리 하단 path 에 묶을 패키지들을 넣어 놓고 zip파일로 묶어야됨

Python/필요한 함수 + library 들을 넣어놓음

* + 아래 링크를 참조하여 구현

<https://aws.amazon.com/ko/premiumsupport/knowledge-center/start-stop-lambda-cloudwatch/>

* + 대상 서버 : 개발환경으로 구성된 windows 서버 등
  + Start 시간 : 월~금 09:00
  + Stop 시간 : 매일 21:00

-EC2 Stop function Example-

import boto3

region = 'us-east-1'

instances = ['i-08e34929f9748748e','xxx'] # ddi-ec2-training

def lambda\_handler(event, context):

ec2 = boto3.client('ec2', region\_name=region)

ec2.stop\_instances(InstanceIds=instances)

print ('stopped your instances: ' + str(instances))

출처: <[*https://ddi-proserve.atlassian.net/jira/software/projects/LAKE/boards/1?selectedIssue=LAKE-106*](https://ddi-proserve.atlassian.net/jira/software/projects/LAKE/boards/1?selectedIssue=LAKE-106)>

Hive style partitions

아래처럼 된게 hive style partitions 임

Crawler는 자동으로 hive style partition을 detect 함.

FUELCELL>RMS>partition id =

시스템 생성 대체 텍스트:
Crawlers: Classifiers 
IAM Role 
Glue Crawler 
JDBC Connection 
Object Connection 
i 
Databases 
Amazon 
RDS 
Data Warehouses 
Amazon 
Redshift 
Data Lakes 
Amazon 53 
I 
Create additional Custom 
Classifiers with Grok! 
Built-ln Classifiers 
MYSQL 
MariaDB 
PostreSQL 
Aurora 
Redshift 
Avro 
Parquet 
ORC 
JSON & BJSON 
Logs 
(Apache, Linux. MS, Ruby, Redis, and many others) 
Delimited 
(comma. pipe tab, sen하(이이하 
Compressed Formats 
(ZIP, BZIP, GZIP, L24. Snappy) 

시스템 생성 대체 텍스트:
Apache Spa「k and AWS Glue ETL 
Apache SparkSQL 
Apache Spark 
DataFrames 
AWS Glue ETL 
AWS Glue 
DynamicFrames 
Apache Spark Co爬: RDDs 
Application 
Data Structure 
Execution 
Apache Spark is a distributed data processing engine for complex analytics 
AWS Glue builds on the Apache Spark to Offer ETL specific functionality 

추천 방식 AWS Glue dynamic frame으로 읽고 etl 로직은 spark dataframe으로 짜고

Write는 아무거나 쓰기 (단, glue dynamic frames은 append 기능밖에 없음)

Excel --> csv로 보기

df\_shutdown=pd.read\_excel("s3://datalake-fck-dev/raw/rdne/servicemax/대외\_2019 Shutdown List.xlsx")

df\_tag=pd.read\_excel("s3://datalake-fck-dev/raw/rdne/servicemax/대외\_Master\_Tag\_List\_\_DDI.xlsx")

제거 컬럼 list 626개 - 35 = 589개

##제거 컬럼

-- cl830\_pi02\_output\_d

cp\_ct430calc\_d cp\_ft012da\_d cp\_te001ft\_d cp\_uh2\_spike\_d cp\_wair\_drift\_high\_d

cp\_wair\_drift\_low\_d

----doutcv441\_value\_b

시스템 생성 대체 텍스트:
plant-id 
CP fuelspfact_d 
ainte321d_value_d 
㈜n하 g_8CSdcount_d 
config-idlesdcount-d 
config_idletrans』Sdcount_d 
cp-stmfact-ng-d 
cp_stmfact_d 
cp_brnfactl_d 
co-factor-ng-d 
CP q-ng-d 
CP ncell_d 
CP molwt_d 
cp.maxidcmeasure_d 
cp-m-ng-d 
cp_ft01기actor_d 
cp_fu elfact_ ng_d 
cp_lhv. ng_d 
ainte320d_value_d 
ainte321c_value d 
plant-group 
sys_softhmimaJorrev_i 
5 yS_SOfthmiminorrev _i 
plant_id 
; YS rVerSlOn. i 
9753 
9754 
9755 
9756 
9757 
9758 
9759 
9825 
9826 

Pcs\_apccrsword\_i

Cp\_hot\_shdwn\_unmit\_cnt\_d

Cp\_hot\_shdwn\_mit\_cnt\_d

Cp\_maxidc\_d

Cp\_planned\_shdwn\_cnt\_d

35

'cp\_fbdur\_d'

-------

cp\_fuelspfact\_d

ainte321d\_value\_d

config\_gcsdcount\_d

config\_idlesdcound\_d

config\_idletranssdcount\_d

cp\_stmfact\_ng\_d

cp\_stmfact\_d

cp\_brnfact1\_d

cp\_factor\_ng\_d

cp\_q\_ng\_d

cp\_ncell\_d

cp\_molwt\_d

cp\_maxidcmeasure\_d

cp\_m\_ng\_d

cp\_ft012factor\_d

cp\_fuelfact\_ng\_d

'cp\_lhv\_ng\_d'

ainte320d\_value\_d

ainte321c\_value\_d

'plant\_group', 'sys\_softfrversion\_i', 'sys\_softhmimajorrev\_i', 'sys\_softhmiminorrev\_i'

Draft 
Architecture 
(10.28) 
Fuel cell 
IOT Data 
aws 
AWS Cloud 
Amazon ROS 
AWS Database 
Migration Sewice 
(Tier 1) 
Transfer - 
for SFTP 
Fuelcell Data 
Amazon 
Amazon Kinesis Amazon Kinesis 
Data Streams Data Firehose 
Aws Glue Data 
Cata 
AWS Lambda 
(Event) 
AV.'S Glue 
Amazon EMR 
(Complex) 
AWS Lake Formation 
3 Or 4) 
Staging 
. Tier 2 or 3 
Amazon S3 Glacier 
AWS Lake 
n Athena 
Amazon SageMaker 
Amazon 
Amazon Redshift 
Management & Governance 
Malysis 
users 
ckSight 
PSS 
users 
Job n Security. Identity. and Comp 
Apache 
Glue 
Workflow 
Aws IAM 
Aws 
AWS Systems 
Manager 
A',vs CLI 
Aws 
Amazon SNS : 
CloudTraiI 

EMR: Glue보다 더 어려운 logic으로 데이터 가져올 때

R을 쓰고 싶으면 EMR을 띄워서 SandBox에 설치하면 됨

--> SandBox는 필요할 때마다 띄웠다가 종료시킬 수 있음

아마존 Athena는 s3 데이터를 인메모리로 가져옴

Redshift = ODS성 데이터 파일과 각종 파일합쳐놓은 저장소라고 생각 하면 됨

AWS Systems Manager: (Parameter Store)각종 파라미터 저장해놓는 곳

AWS CLI: shell script로 짤 수 있는 곳

Lamda에서 코드를 짜서 Amazon CloudWatch로 스케줄링 걸어서 새벽에 안끄고 있는 sagemaker는 종료시킬 수 있음

AWS CloudTrail: 모니터링하고 감시하는 측면

Amazon SNS: Fail나는 것들을 알림주는것(메일)

AWS KMS: 데이터 자체를 암호화 것, 암호화가 필요한 데이터는 KMS key를 사용함

Apache Airflow는 Python기반의 workflow를 짜는 건데, GlueWorkflow만으로 버거울 때 사용 (오픈소스)

AWS Lake Formation: Grant read, write등을 전체적으로 줄 수 있는 관리 용

[1. S3 환경 만들기]

S3 LifeCycle: Governance: 삭제 주기 등

Intelligence tiering: S3 IntelligentTiering : 한달비용청구 / S3 Standard-IA

S3 관련해서 하면 좋을 작업들(management관점에서)

lifecycle(versioning)

protection(default encryption)

server access logging (계정별)

object-level logging (datalake-cloudtrail-logs) - read/write에 대한 작업에 대한 모든 로그 쌓기

inventory (object에 대한 통계를 볼수있게 설정함)

[2. SFTP로 데이터 가져오기]

v Designer 
CloudWatch Events 
fck_rms_daily_report_with_athena 
Layers 
x 

LAMBDA 함수

import boto3

from datetime import date

#Function for executing athena queries

def run\_query(query, database, s3\_output):

client = boto3.client('athena','us-east-1')

response = client.start\_query\_execution( QueryString=query, \

QueryExecutionContext={ 'Database': database }, \

ResultConfiguration={ 'OutputLocation': s3\_output, } )

print('Execution ID: ' + response['QueryExecutionId'])

return response

#Athena configuration

def lambda\_handler(event,context) :

today = date.today()

s3\_output = 's3://ddi-athena-results2/sungyoul/{}/'.format(today)

#Athena database and table definition

target\_db = 'fck\_rdne\_rms\_report'

target\_table = 'daily\_pstate'

target\_s3\_input = 's3://datalake-fck-dev2/report/rdne/rms/daily/pstate/'

create\_database = "CREATE DATABASE IF NOT EXISTS {};".format(target\_db)

drop\_table = "drop table {}.{};".format(target\_db, target\_table)

create\_table = """

CREATE EXTERNAL TABLE IF NOT EXISTS {}.{} (

`today` date,

`date` date,

`pstate` int,

`count` bigint)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION

'{}'

TBLPROPERTIES (

'has\_encrypted\_data'='false');""".format(target\_db, target\_table, target\_s3\_input )

database = 'fck\_rdne\_rms'

table = 'clean\_rmslog2'

#Query definitions

query\_1 = """

INSERT INTO {}.{}

SELECT

current\_date AS today,

date(split\_part(record\_timestamp\_new,' ',1)) AS date,

pstatenumber\_i,

count(pstatenumber\_i) AS count\_pstate

FROM {}.{}

WHERE date(split\_part(record\_timestamp\_new,' ',1))

BETWEEN current\_date - interval '120' day

AND current\_date - interval '120' day

AND pstatenumber\_i >=150

GROUP BY date(split\_part(record\_timestamp\_new,' ',1)) ,pstatenumber\_i

ORDER by date,pstatenumber\_i;""".format(target\_db,target\_table,database, table)

import time

#Execute all queries

queries = [ create\_database, create\_table, query\_1 ]

for q in queries:

print("Executing query: %s" % (q))

res = run\_query(q,target\_db,s3\_output)



Raw--> clean data set 만들기

import boto3

import pandas as pd

import datetime as dt

​

aws\_key = 'AKIAWN7X7MVENQAWMY6A'

aws\_secret = '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'

​

s3 = boto3.client('s3','us-east-1',aws\_access\_key\_id=aws\_key, aws\_secret\_access\_key= aws\_secret)

​

bucket = 'datalake-fck-dev2'

key\_shutdown = 'raw/rdne/rms/service/shoutdown\_list/shoutdown\_list.csv'

key\_taglist= 'raw/rdne/rms/service/tag\_list/tag\_list.csv'

​

shut\_down = s3.get\_object(Bucket= bucket, Key=key\_shutdown)

df\_shutdown = pd.read\_csv(shut\_down["Body"],encoding ="ISO-8859-1")

​

tag\_list = s3.get\_object(Bucket= bucket, Key=key\_taglist)

df\_taglist = pd.read\_csv(tag\_list["Body"],encoding ="ISO-8859-1")

​

# Data 내용 확인 : 첫 3줄만 읽어서 출력

df\_shutdown.head(3)

df\_taglist.head(3)

​

df\_shutdown.columns

df\_taglist.columns

​

### ETL 1. Column Name

ori\_col\_shutdown = df\_shutdown.columns

new\_col\_shutdown = []

for col in ori\_col\_shutdown:

new\_col\_shutdown.append(col.replace(" ","\_").replace("/","\_").replace(":","").lower())

print(new\_col\_shutdown)

​

​

ori\_col\_taglist = df\_taglist.columns

new\_col\_taglist = []

for col in ori\_col\_taglist:

new\_col\_taglist.append(col.replace(" ","\_").replace("/","\_").replace(":","").lower())

print(new\_col\_taglist)

​

df\_shutdown.columns = new\_col\_shutdown

df\_taglist.columns = new\_col\_taglist

​

print(df\_shutdown.head(1))

print(df\_taglist.head(1))

​

​

### ETL 2. date-time column 2018. 3. 4 PM 2:00 --> 2018-03-04 14:00

​

#df\_shutdown[['shutdown\_date\_time','closed\_date','restart\_date\_time']].head()

​

def date\_time\_convert(x,option=1):

try:

if option == 1:

return dt.datetime.strptime(str(x).replace(" ",""), "%Y.%m.%d%p%I:%M")

elif option == 2:

return dt.datetime.strptime(str(x).replace(" ",""), "%Y.%m.%d")

else :

print("please input option 1 or 2")

exit(0)

except:

return pd.NaT

​

df\_shutdown['shutdown\_date\_time'] = df\_shutdown['shutdown\_date\_time'].map(lambda x : date\_time\_convert(x,option=1))

df\_shutdown['restart\_date\_time'] = df\_shutdown['restart\_date\_time'].map(lambda x : date\_time\_convert(x,option=1))

df\_shutdown['closed\_date'] = df\_shutdown['closed\_date'].map(lambda x : date\_time\_convert(x,option=2))

​

df\_shutdown[['shutdown\_date\_time','closed\_date','restart\_date\_time']].head()

​

​

from io import StringIO # python3; python2: BytesIO

​

bucket = 'datalake-fck-dev2' ## already created on S3

key\_1 = "clean/rdne/rms/service/shutdown\_list/shutdown\_list.csv" ## 폴더이름 변경

key\_2 = "clean/rdne/rms/service/tag\_list/tag\_list.csv" ## 폴더 이름 변경

​

csv\_buffer = StringIO()

df\_shutdown.to\_csv(csv\_buffer,index=False,sep='\t')

s3.put\_object(Bucket=bucket, Body=csv\_buffer.getvalue(), Key= key\_1)

​

csv\_buffer = StringIO()

df\_taglist.to\_csv(csv\_buffer,index=False,sep='\t')

s3.put\_object(Bucket=bucket, Body=csv\_buffer.getvalue(), Key= key\_2)

출처: <[*https://app.slack.com/client/TPARUQHDG/CPS24DX7W*](https://app.slack.com/client/TPARUQHDG/CPS24DX7W)>