

CONTEXT

- An event driven serverless data processing architecture - Cost optimized
- Spark Python scripts for data processing and loading into Redshift & S3

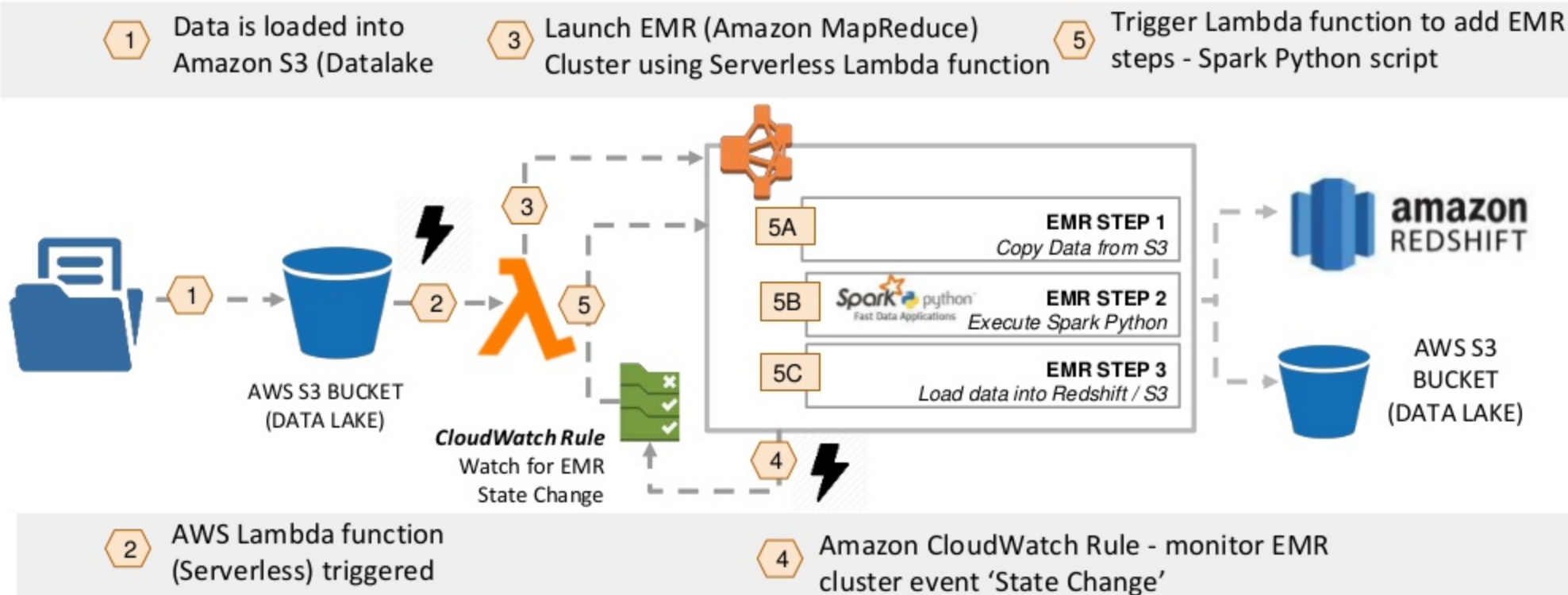
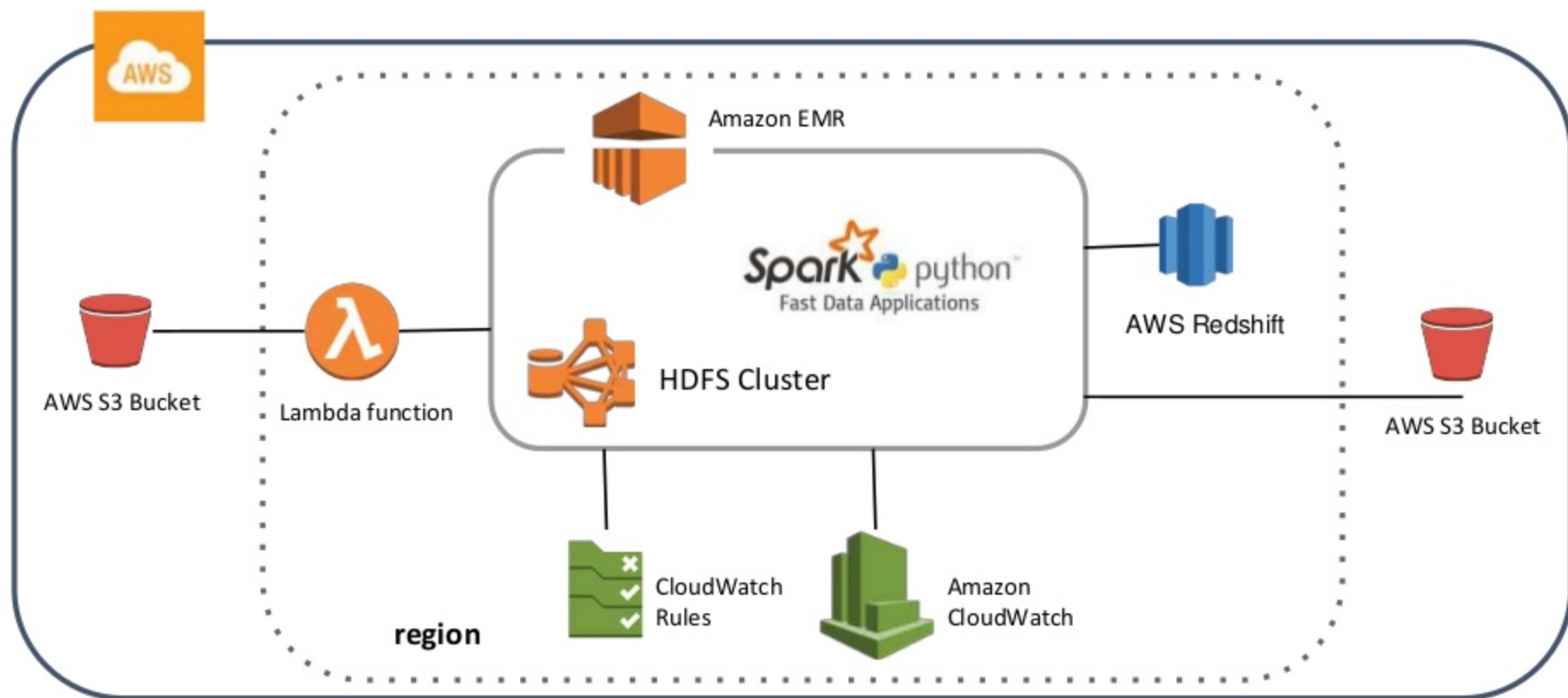




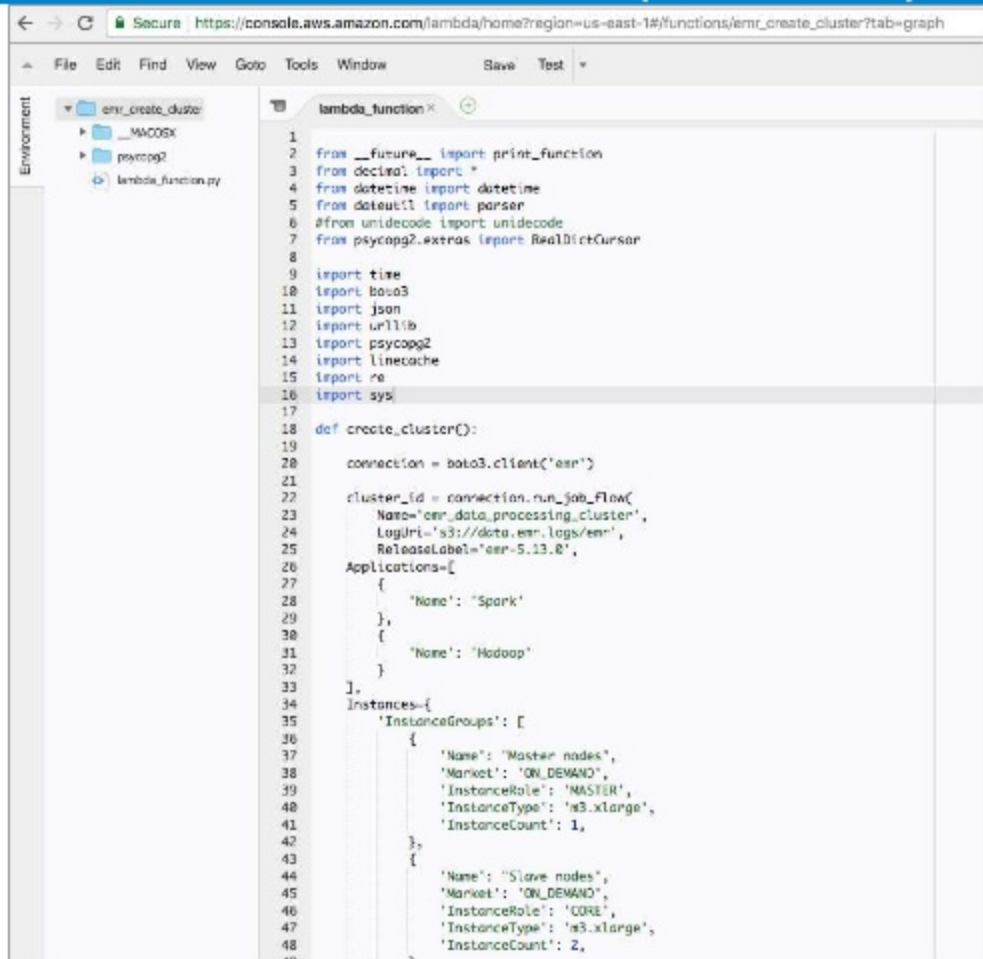
TABLE OF CONTENT

1. TECHNICAL ARCHITECTURE
2. LAMBDA FUNCTION (PYTHON) TO CREATE EMR CLUSTER
3. CLOUDWATCH RULES - AWS EMR (ELASTIC MAPREDUCE) STEPS
4. LAMBDA FUNCTION (PYTHON) TO ADD **SPARK**-PYTHON STEPS
5. EMR WITH STEP EXECUTION
6. FUTURE IMPROVEMENTS - AWS GLUE

TECHNICAL ARCHITECTURE



LAMBDA FUNCTION (PYTHON) TO CREATE EMR CLUSTER



The screenshot shows the AWS Lambda console interface. The browser address bar displays the URL: `https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions/emr_create_cluster?tab=graph`. The file explorer on the left shows the project structure: `emr_create_cluster` containing `__MACOSX`, `psycog2`, and `lambda_function.py`. The main editor displays the Python code for the `lambda_function` module.

```
1 from __future__ import print_function
2 from decimal import *
3 from datetime import datetime
4 from datetime import parser
5 from unicode import unicode
6 from psycog2.extras import RealDictCursor
7
8 import time
9 import boto3
10 import json
11 import urllib
12 import psycog2
13 import linecache
14 import re
15 import sys
16
17 def create_cluster():
18     connection = boto3.client('emr')
19
20     cluster_id = connection.run_job_flow(
21         Name='emr_data_processing_cluster',
22         LogUri='s3://data.emr.logs/emr',
23         ReleaseLabel='emr-5.13.0',
24         Applications=[
25             {
26                 'Name': 'Spark'
27             },
28             {
29                 'Name': 'Hadoop'
30             }
31         ],
32         Instances={
33             'InstanceGroups': [
34                 {
35                     'Name': 'Master nodes',
36                     'Market': 'ON_DEMAND',
37                     'InstanceRole': 'MASTER',
38                     'InstanceType': 'm3.xlarge',
39                     'InstanceCount': 1,
40                 },
41                 {
42                     'Name': 'Slave nodes',
43                     'Market': 'ON_DEMAND',
44                     'InstanceRole': 'CORE',
45                     'InstanceType': 'm3.xlarge',
46                     'InstanceCount': 2,
47                 }
48             ]
49         }
50     )
```

KEY POINTS:

- Use Python boto3 library
- In Applications - add Spark, Hadoop
- No initial steps defined, as these will be specified once cluster is created
- Use a python bootstrap bash file

```
Steps=[],
VisibleToAllUsers=True,
JobFlowRole='EMR_EC2_DefaultRole',
ServiceRole='EMR_DefaultRole',
BootstrapActions=[
    {
        'Name': 'InstallPythonLibraries',
        'ScriptBootstrapAction': {
            'Path': 's3://data.emr.logs/emr.bootstrap/python_bootstrap.sh'
        }
    },
],
Tags=[
    {
        'Key': 'description',
        'Value': 'execute Spark',
    }
],
)
```

CLOUDWATCH RULES - AWS EMR (ELASTIC MAPREDUCE) STEPS

Step 1: Create rule

Create rules to invoke Targets based on Events happening in your AWS environment.

Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

☒ Event Pattern ⓘ ☐ Schedule ⓘ

Build event pattern to match events by service

Service Name

Event Type

☐ Any detail type ☒ Specific detail type(s)

☒ Any state ☐ Specific state(s)

☒ Any cluster ID ☐ Specific cluster ID(s)

▼ Event Pattern Preview

Copy to clipboard Edit

```
{
  "source": [
    "aws.emr"
  ],
  "detail-type": [
    "EMR Cluster State Change"
  ]
}
```

KEY POINTS:

- Cloudwatch rule to raise event whenever EMR state change
- Call lambda function to create EMR steps

Targets

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

Lambda function

Function*

▼ Configure version/alias

☒ Default

☐ Version

☐ Alias

▼ Configure input

☒ Matched event ⓘ

☐ Part of the matched event ⓘ

☐ Constant (JSON text) ⓘ

☐ Input Transformer ⓘ

⊕ Add target*

LAMBDA FUNCTION (PYTHON) TO ADD SPARK-PYTHON STEPS

KEY POINTS:

- AWS Lambda function to create EMR steps. One of the steps shown below.
- EMR Step to execute “spark-submit” python job

```
if not "sparkSubmitOrderList" in steps_name:
    step_args = ["spark-submit", "--deploy-mode", "cluster", "--master", "yarn", "--conf",
                "spark.yarn.submit.waitAppCompletion=false", "--num-executors", "5", "--executor-cores", "5",
                "--executor-memory", "5g", "/home/hadoop/code.py", SOURCE_DATA_FILE_NAME, RESULT_CSV_NAME, RESULT_S3_FOLDER_LOCATION]

    step = {"Name": "sparkSubmitOrderList",
            "ActionOnFailure": 'CONTINUE',
            "HadoopJarStep": {
                'Jar': 'command-runner.jar',
                'Args': step_args
            }
    }
    action = conn.add_job_flow_steps(JobFlowId=clusterId, Steps=[step])
```



EMR WITH STEP EXECUTION

KEY POINTS:

- EMR with step execution
- Notice spark-submit job to execute Python script

sparkSubmitOrderList

Status: Pending ID: s-26CQEUT3RY43 Start time: Elapsed time:

Log files: No logs created yet

JAR location: command-runner.jar

Main class: None

```
spark-submit --deploy-mode cluster --master yam --conf
spark.yam.submit.waitForAppCompletion=false --num-executors 5 --executor-cores 5 --executor-
memory 5g /home/hadoop/code.py customer-orders.csv result.csv
```

Arguments: s3://data.emr.program/cust/result

Action on failure: Continue

Cancel

Filter: All clusters 2 clusters (all loaded)

Name	ID	Status	Creation time (UTC-5)	Elapsed time	Normalized instance hours
emr_data_processing_cluster	j-1YRJNENYK3ZR	Starting	2018-06-08 19:09 (UTC-5)	5 seconds	0

Summary

Master public DNS: --

Termination protection: Off Change

Tags: description = execute Spark View All / Edit

Hardware

Master: Provisioning 1 m3.xlarge

Core: Provisioning 2 m3.xlarge

Task: --

Steps

Name	Status	Start time (UTC-5)	Elapsed time
sparkSubmitOrderList	Pending		
copyCodeIntoHadoop	Pending		
copyS3toHDFS	Pending		

Bootstrap actions

Name
InstallPythonLibraries