lyit | Institiúid Teicneolaíochta Leitir Ceanainn
Letterkenny Institute of Technology

Title:	Streaming MapReduce & Hive using AWS
Student:	Prateek Parasher
Lecturer:	Edwina Sweeney
Course:	M.Sc. in Computing in Big Data Analytics
Module:	Big Data Architecture
Submission:	21st may 2018 via Blackboard

Introduction

Cloud computing is providing new power to industries all around the world and giving on demand computational power, storage, ram, different applications & other resources. This technology reduces the hardware configure and reduce maximum time in configuration like MapReduce, Hadoop this big data technology having very complicated configuration process & we have to make sure that our system having Linux / mac-os and compatible hardware to configure big data technologies & if we want to train heavily dataset then also we need high computational power

1. Amazon web services(AWS) is world leader in cloud computing services. In this project I am using aws providing lots of service in this project I am using few of them like S3 , EMR , EC2 so and so on
prior start my project details I would like give brief idea about aws , aws is currently providing their services nearly in 190 countries and they are keep expanding 42 zones for aws dealing with millions of active users

2. EMR stands for elastic MapReduce & emr is one of the aws services major component in emr is nodes, cluster & instances In amazon emr we can use Hadoop, spark components and usually in emr cluster having different nodes like master node, current node, task node.

➔ master node:- this is the main node that has all powers of controlling master node manages the running software compounds & manage the coordination between other tasks

Core node: - a slave node which is controlled by master node & salve node store all the data which run by other software component into the hdfs on the cluster

Task node:- this is also slave node but only responsible for running the tasks.

After getting overview of cluster & nodes we will talk about advantages of EMR

- ➔ Cost saving:- pay as you go depends upon the services we are using we have pay only for that like if we launch Ec2 using instance in west orgen region. We will be charged only that according to usages
- ➔ Aws integration :- for example if we using EMR service in our cluster we can integrate storage like AWS (S3) , amazon cloud watch to monitor the performance of cluster deployment, scalability, security etc

Amazon EMR is great service provider and make all difficult setup very easy, big data in different type of applications including machine learning, web indexing, bio informatics, log analysis, amazon EMR using Hadoop open source, which distribute data & processing across a resizable cluster Ec2(AWS)

Hadoop using distribution processing which is called MapReduce in which a mapper converts raw source data into key/value pairs and the find result comes they reduce the output to single set

In this project Initially, I created S3 bucket which is in AWS, S3 bucket for storage and creating s3 is very easy

Topology design

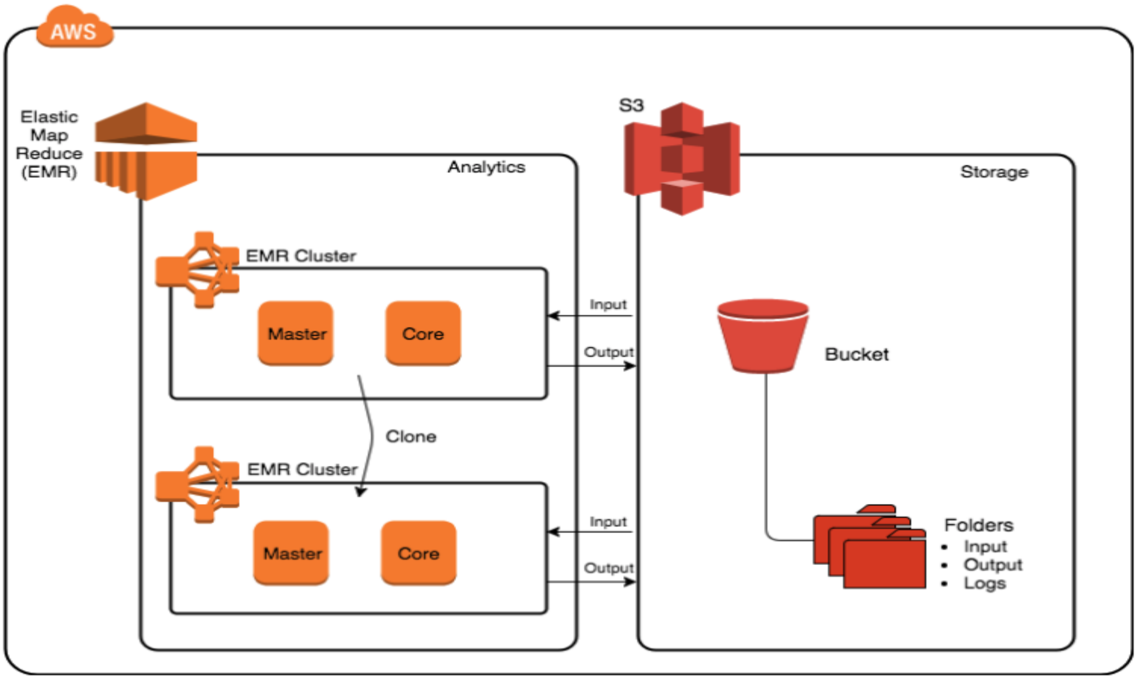


fig1 EMR in aws using S3 bucket

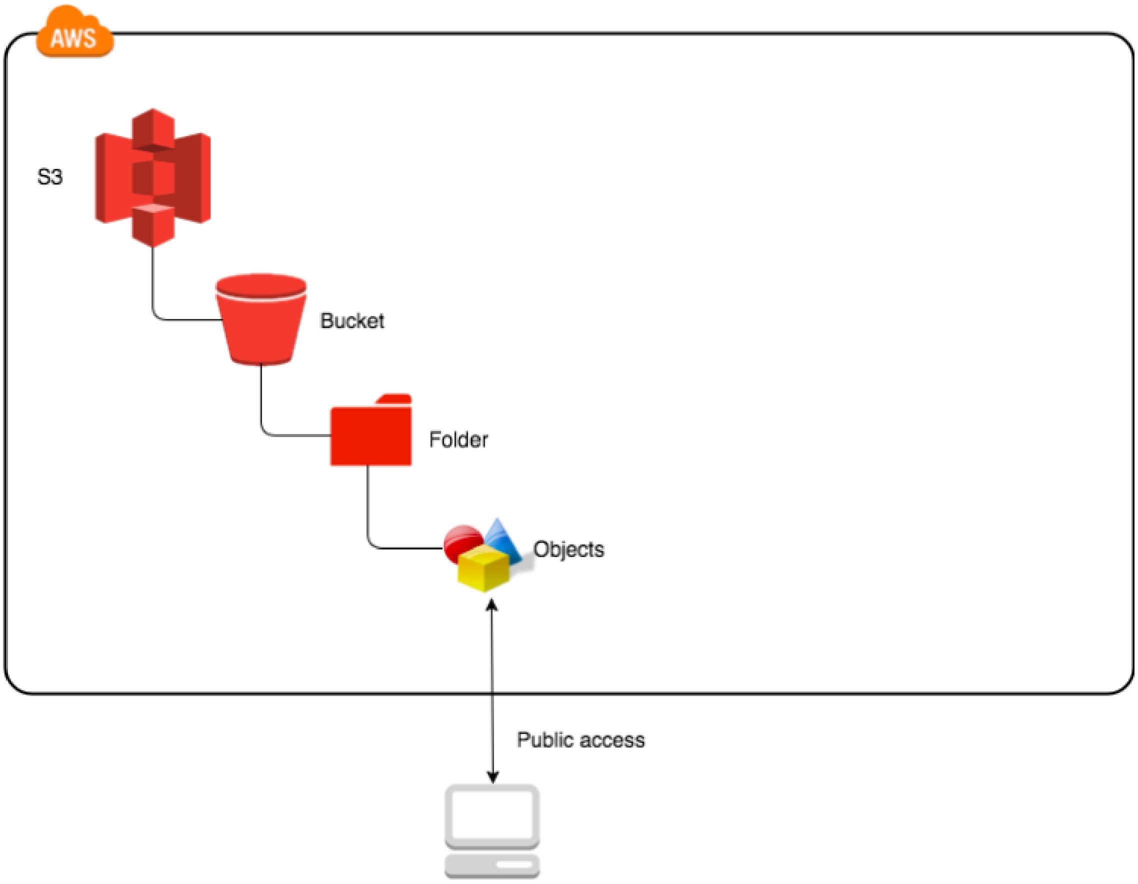


Fig 2 S3 bucket store system

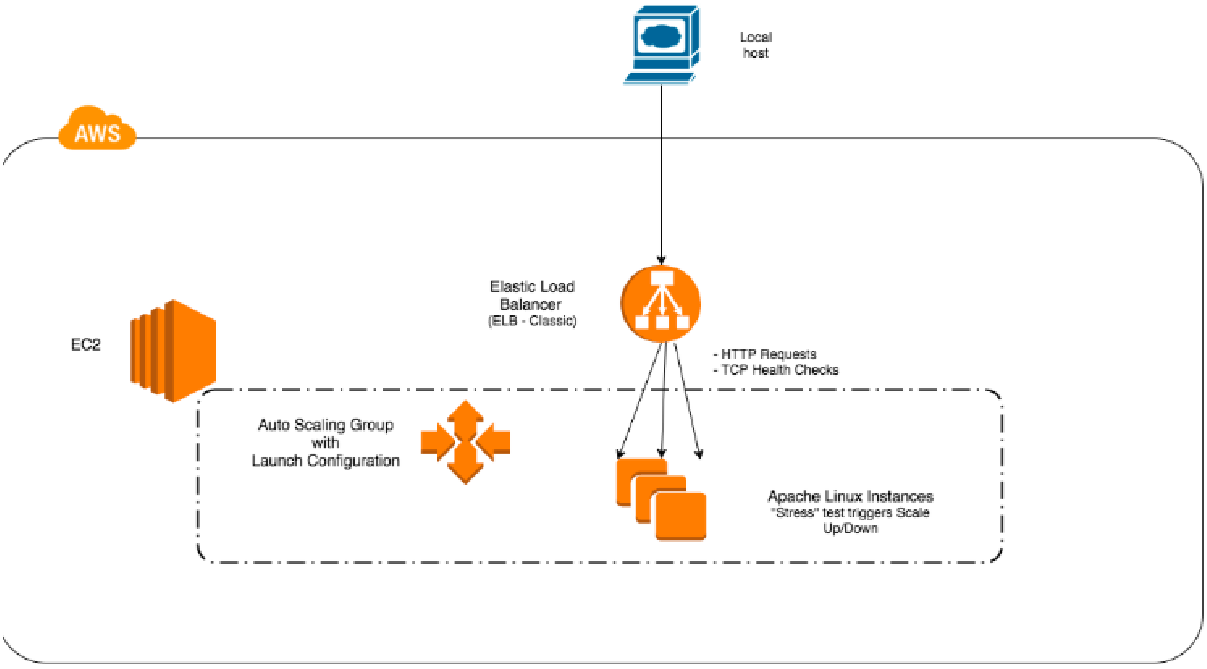


Fig 3 Ec2 cluster instance

Use cases

Zillow
3. Zillow is one of the largest real estate company in united states which delay with millions of home valuation data all the operation Zillow done in real-time using machine-learning calculation performance Zillow uses amazon kinesis streams input different type of data, sales & marketing transaction. they using machine learning algorithm all data store in amazon S3(aws) increasing in speed and accuracy using spark on emr (AWS), main benefits are prior using aws Zillow used to spend 1 day for high performance & accuracy now they are doing same process in 1 hr using AWS, they can easily scale up & compute capacity on demand

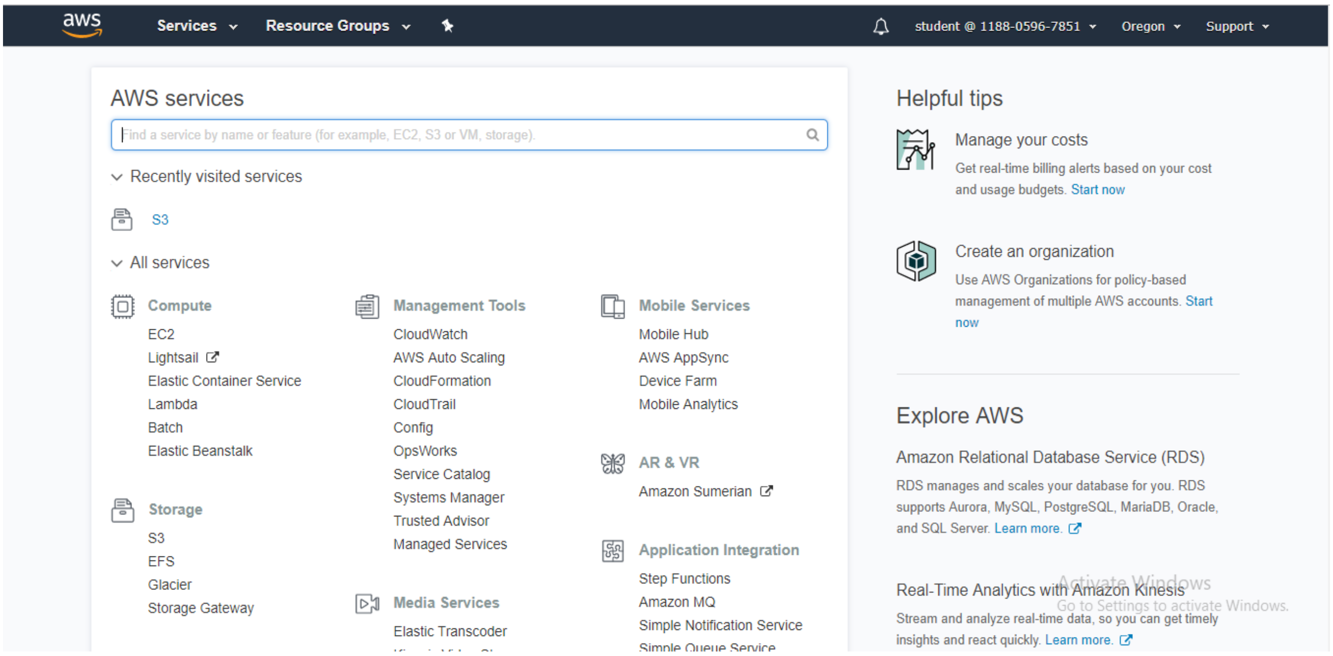
YELP

4. Yelp is more that decade year old company mainly focused on connecting people for businesses & help local business also yelp is backed by the community yelp dealing with 29 countries & 120 market through this expansion yelp faced few different challenges like storage , scaling up to overcome this problem they start using aws cloud computing services they start using single local instance of Hadoop and later on yelp move to EMR and yelp start using S3 for store the daily logs , picture , text , which is approx. 1.2 TB per day
after consuming aws services yelp nearly save $55,000 cost of hardware running per day.

Expedia

5. Expedia is one of the largest company in travel industry. Expedia company deals in hotels, travels & other facilities they are providing their services to 60 countries. They using aws amazon emr & amazon S3 for storage before that Expedia facing lot's of different problems like scaling up their business and Expedia approx. processes 240 request per second for security of the transaction in Expedia they using IAM aws security services to manage their security

Deployment Steps



AWS services

Find a service by name or feature (for example, EC2, S3 or VM, storage).

Recently visited services

S3

All services

Compute
EC2
Lightsail
Elastic Container Service
Lambda
Batch
Elastic Beanstalk

Storage
S3
EFS
Glacier
Storage Gateway

Management Tools
CloudWatch
AWS Auto Scaling
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Systems Manager
Trusted Advisor
Managed Services

Media Services
Elastic Transcoder

Mobile Services
Mobile Hub
AWS AppSync
Device Farm
Mobile Analytics

AR & VR
Amazon Sumerian

Application Integration
Step Functions
Amazon MQ
Simple Notification Service
Simple Queue Service

Helpful tips

Manage your costs
Get real-time billing alerts based on your cost and usage budgets. Start now

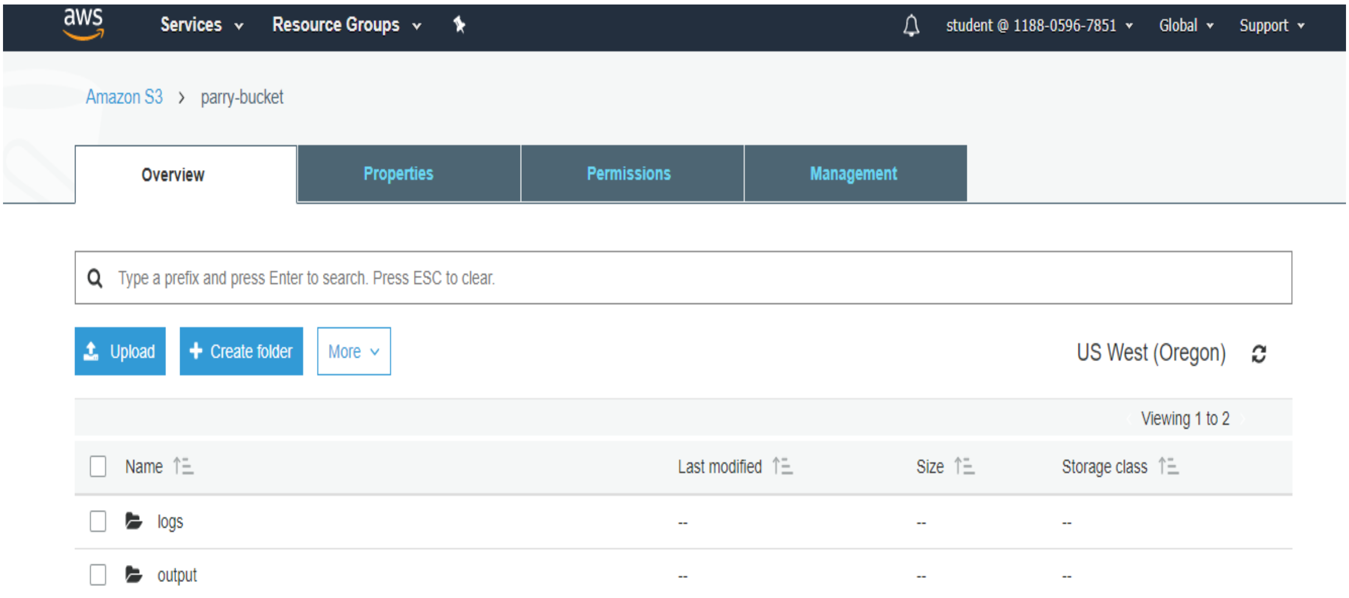
Create an organization
Use AWS Organizations for policy-based management of multiple AWS accounts. Start now

Explore AWS

Amazon Relational Database Service (RDS)
RDS manages and scales your database for you. RDS supports Aurora, MySQL, PostgreSQL, MariaDB, Oracle, and SQL Server. Learn more.

Real-Time Analytics with Amazon Kinesis
Stream and analyze real-time data, so you can get timely insights and react quickly. Learn more.

Fig 4 front view aws all services



Amazon S3 › parry-bucket

Type a prefix and press Enter to search. Press ESC to clear.

US West (Oregon)

Viewing 1 to 2

Name	Last modified	Size	Storage class
logs	--	--	--
output	--	--	--

Fig 5 create S3 bucket under name parry-bucket

In this S3 bucket I created 2 folders name under logs, output, input. S3 is aws storage service providing

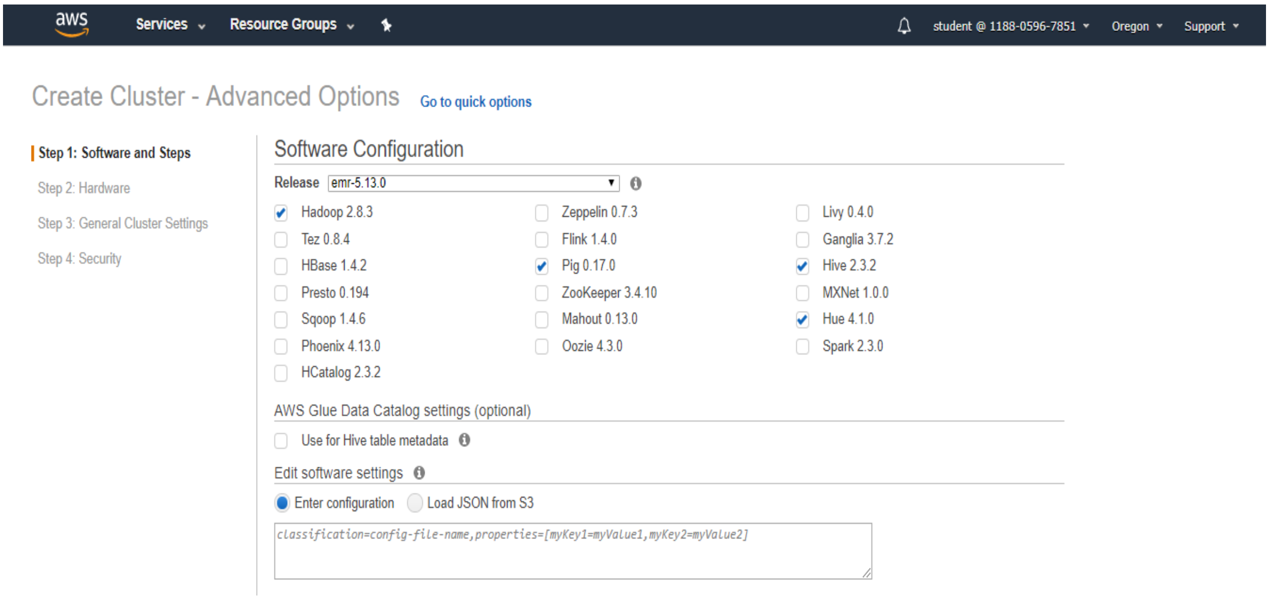
Services > storage > S3

Create bucket & then create folder

Create folder name

- Logs
- Input

Now I creating EMR cluster in EMR cluster there are major for configuration software, general , hardware , security configuration as shown in fig



Create Cluster - Advanced Options   Go to quick options

Step 1: Software and Steps
Step 2: Hardware
Step 3: General Cluster Settings
Step 4: Security

Software Configuration

Release emr-5.13.0

☑ Hadoop 2.8.3
□ Tez 0.8.4
□ HBase 1.4.2
□ Presto 0.194
□ Sqoop 1.4.6
□ Phoenix 4.13.0
□ HCatalog 2.3.2
□ Zeppelin 0.7.3
□ Flink 1.4.0
☑ Pig 0.17.0
□ ZooKeeper 3.4.10
□ Mahout 0.13.0
□ Oozie 4.3.0
□ Livy 0.4.0
□ Ganglia 3.7.2
☑ Hive 2.3.2
□ MXNet 1.0.0
☑ Hue 4.1.0
□ Spark 2.3.0

AWS Glue Data Catalog settings (optional)

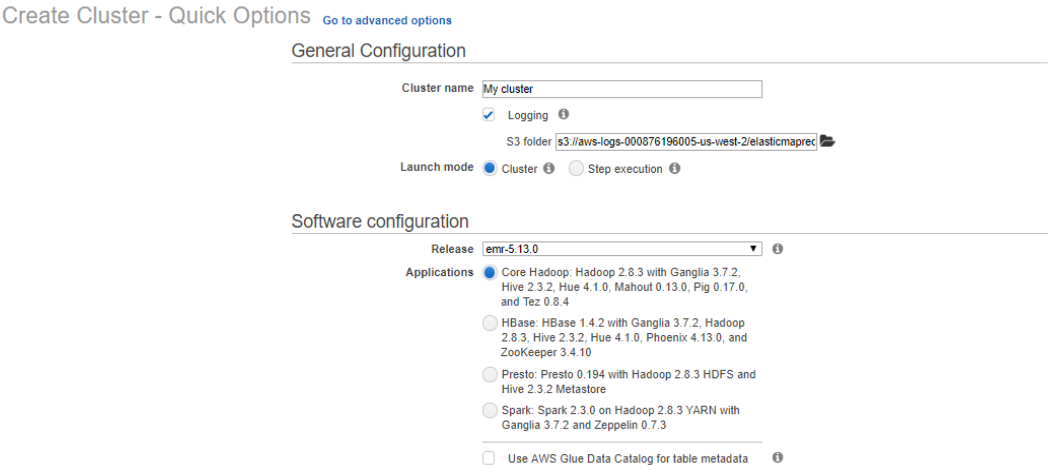
□ Use for Hive table metadata

Edit software settings

● Enter configuration ○ Load JSON from S3

```
classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]
```

Fig 6 creating cluster and e.m.r-5.6 as per requirement



Create Cluster - Quick Options   Go to advanced options

General Configuration

Cluster name: My cluster

☑ Logging

S3 folder: s3://aws-logs-000876196005-us-west-2/elasticmaprec

Launch mode: ● Cluster ○ Step execution

Software configuration

Release: emr-5.13.0

Applications:
● Core Hadoop: Hadoop 2.8.3 with Ganglia 3.7.2, Hive 2.3.2, Hue 4.1.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.8.4
○ HBase: HBase 1.4.2 with Ganglia 3.7.2, Hadoop 2.8.3, Hive 2.3.2, Hue 4.1.0, Phoenix 4.13.0, and ZooKeeper 3.4.10
○ Presto: Presto 0.194 with Hadoop 2.8.3 HDFS and Hive 2.3.2 Metastore
○ Spark: Spark 2.3.0 on Hadoop 2.8.3 YARN with Ganglia 3.7.2 and Zeppelin 0.7.3

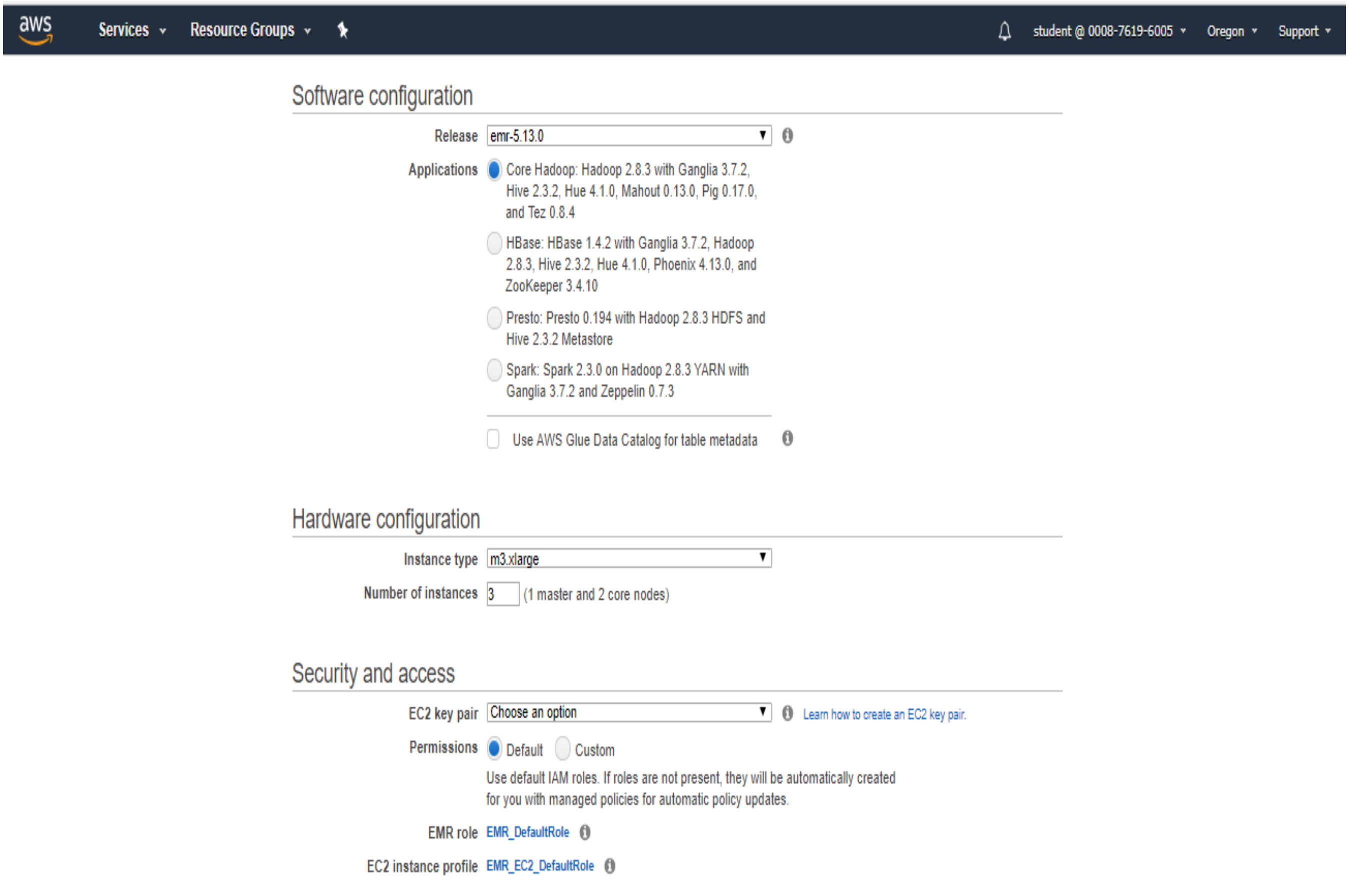
□ Use AWS Glue Data Catalog for table metadata

Fig 7 configuration for creating cluster

In software configuration I select EMR 5.6.0 which including Hadoop , pig, hive after clicking next few minute later my cluster is ready now I am doing in this project of hive script that processes amazon cloud front(cf) log files . the hive script & the sample cf logs and details available on S3 public bucket

When we click Add Step then fill out the blanks with suitable answer when we finish the cluster we need to wait for minute to complete the setup

I am running Hive on aws cluster & hive script running sample I clicked on a cluster list left hand side shown in a figure after that I clicked add step & fill hive program like name , output file etc, select the running cluster name & the summary information shows the cluster running and In the end in this hive lab initially I set up the cluster as per requirement the whole processes running on EMR the code I used was hive script provide by amazon, along with example of cloudfont logs.



Software configuration

Release emr-5.13.0

Applications
● Core Hadoop: Hadoop 2.8.3 with Ganglia 3.7.2, Hive 2.3.2, Hue 4.1.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.8.4
○ HBase: HBase 1.4.2 with Ganglia 3.7.2, Hadoop 2.8.3, Hive 2.3.2, Hue 4.1.0, Phoenix 4.13.0, and ZooKeeper 3.4.10
○ Presto: Presto 0.194 with Hadoop 2.8.3 HDFS and Hive 2.3.2 Metastore
○ Spark: Spark 2.3.0 on Hadoop 2.8.3 YARN with Ganglia 3.7.2 and Zeppelin 0.7.3

□ Use AWS Glue Data Catalog for table metadata

Hardware configuration

Instance type m3.xlarge

Number of instances 3 (1 master and 2 core nodes)

Security and access

EC2 key pair Choose an option Learn how to create an EC2 key pair.

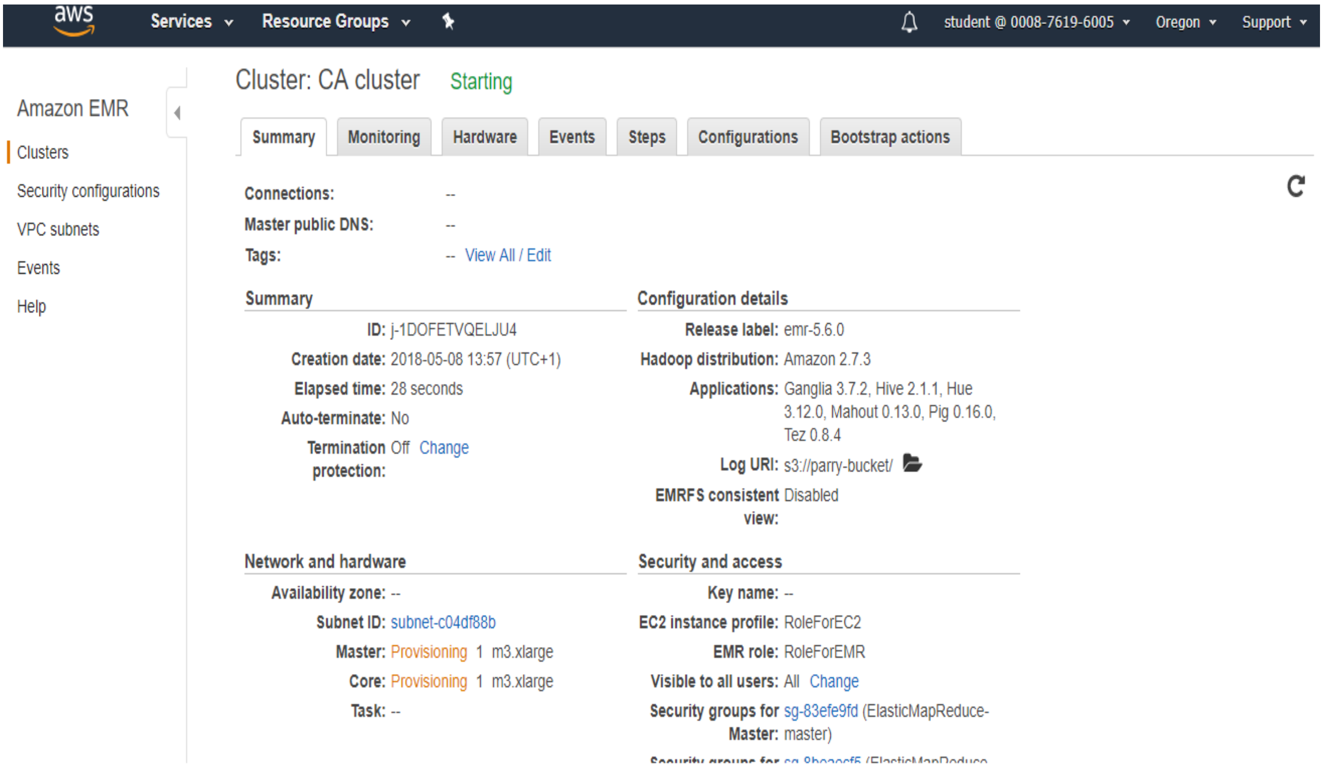
Permissions ● Default ○ Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role EMR_DefaultRole

EC2 instance profile EMR_EC2_DefaultRole

Fig 8 selecting hardware & security configuration



Cluster: CA cluster Starting

Connections: --

Master public DNS: --

Tags: -- View All / Edit

Summary

ID: j-1DOFETVQELJU4

Creation date: 2018-05-08 13:57 (UTC+1)

Elapsed time: 28 seconds

Auto-terminate: No

Termination protection: Off Change

Configuration details

Release label: emr-5.6.0

Hadoop distribution: Amazon 2.7.3

Applications: Ganglia 3.7.2, Hive 2.1.1, Hue 3.12.0, Mahout 0.13.0, Pig 0.16.0, Tez 0.8.4

Log URI: s3://parry-bucket/

EMRFS consistent view: Disabled

Network and hardware

Availability zone: --

Subnet ID: subnet-c04df88b

Master: Provisioning 1 m3.xlarge

Core: Provisioning 1 m3.xlarge

Task: --

Security and access

Key name: --

EC2 instance profile: RoleForEC2

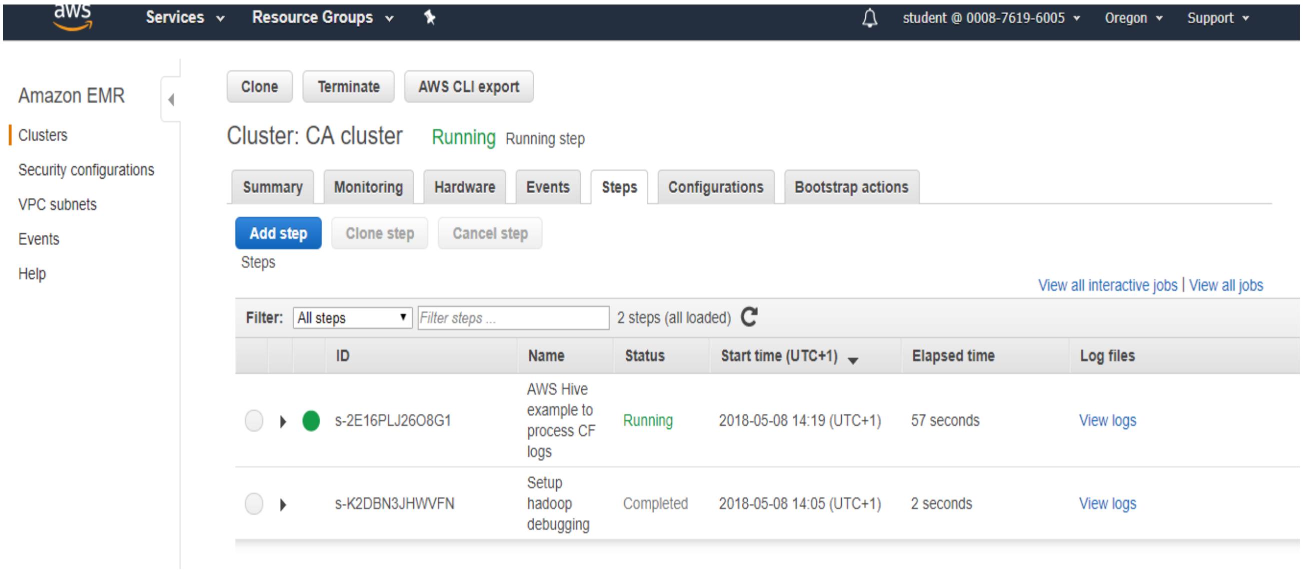
EMR role: RoleForEMR

Visible to all users: All Change

Security groups for Master: sg-83efe9fd (ElasticMapReduce-master)

Security groups for sg-8bcaecf5 (ElasticMapReduce

Fig 10 CA cluster dashboard



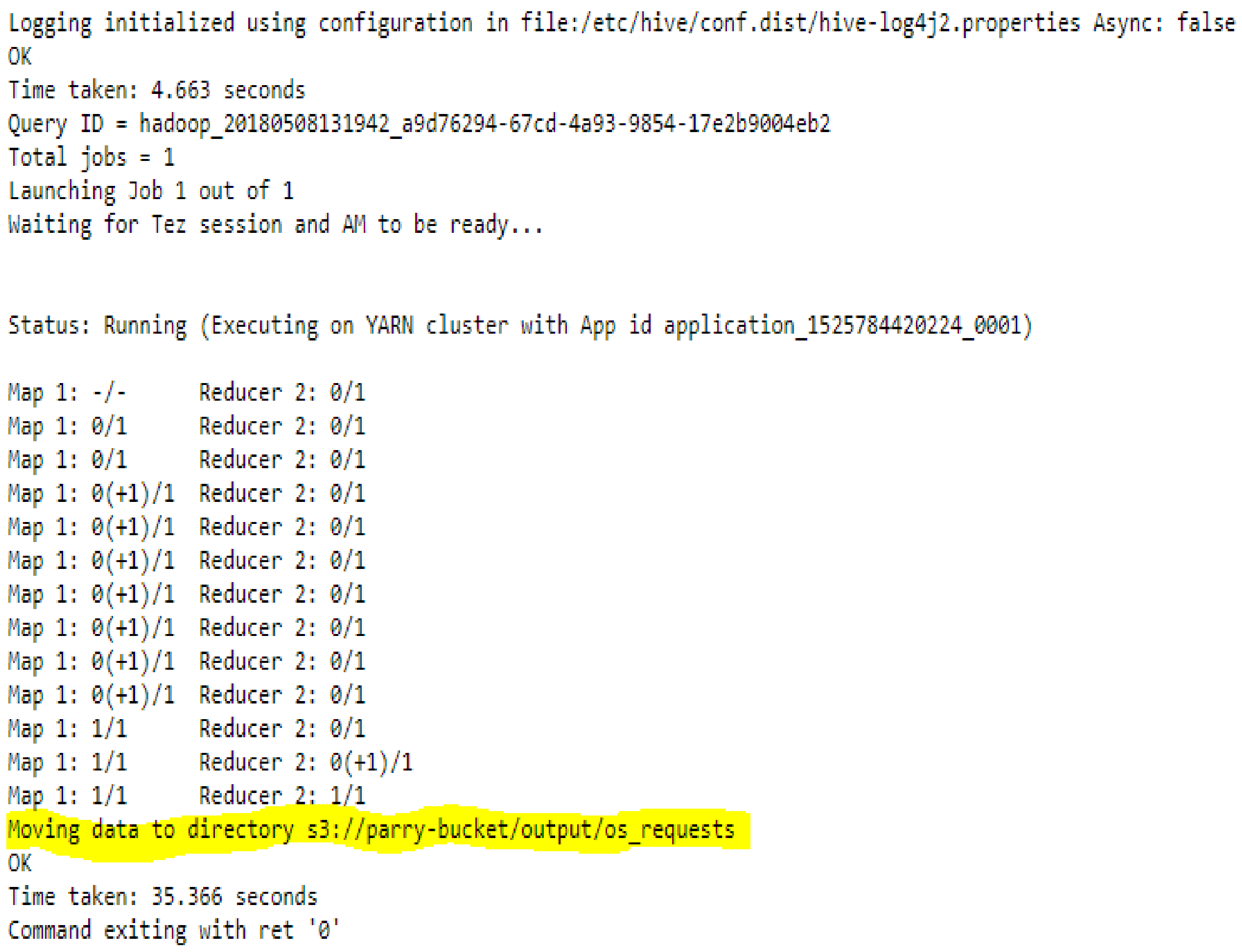
Cluster: CA cluster Running Running step

Steps

Filter: All steps Filter steps ... 2 steps (all loaded)

ID	Name	Status	Start time (UTC+1)	Elapsed time	Log files
s-2E16PLJ26O8G1	AWS Hive example to process CF logs	Running	2018-05-08 14:19 (UTC+1)	57 seconds	View logs
s-K2DBN3JHWVFN	Setup hadoop debugging	Completed	2018-05-08 14:05 (UTC+1)	2 seconds	View logs

Fig 11 steps dashboard hive running successfully

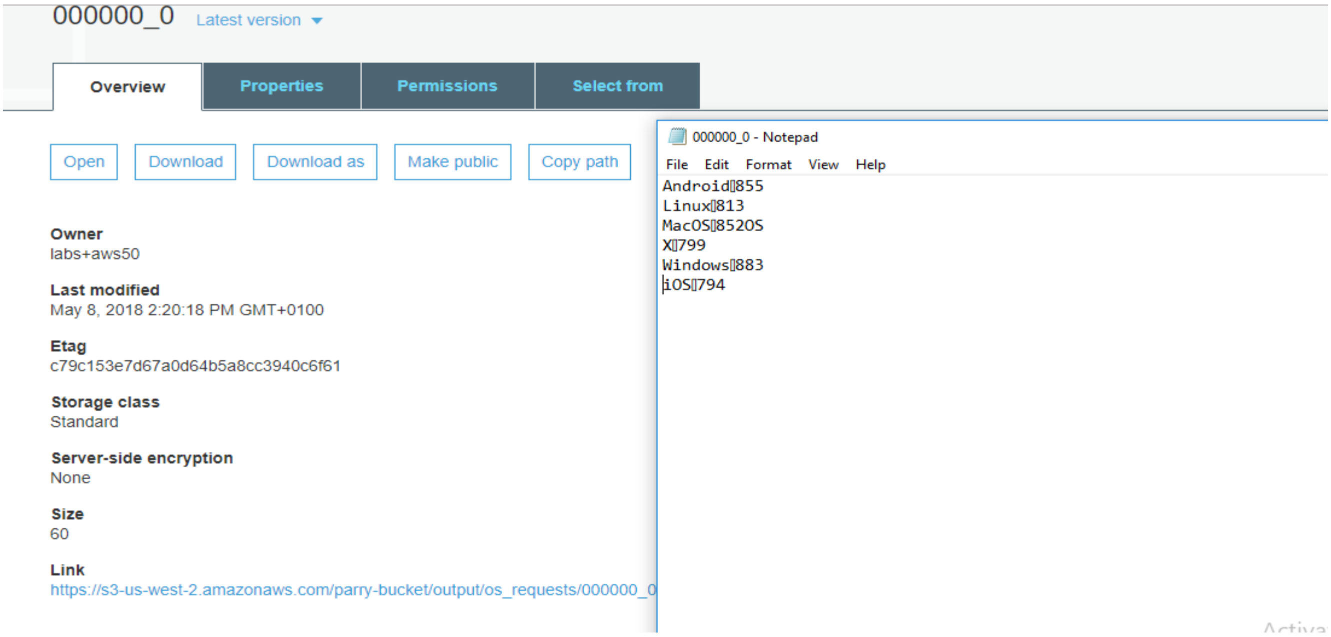


```
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
OK
Time taken: 4.663 seconds
Query ID = hadoop_20180508131942_a9d76294-67cd-4a93-9854-17e2b9004eb2
Total jobs = 1
Launching Job 1 out of 1
Waiting for Tez session and AM to be ready...


Status: Running (Executing on YARN cluster with App id application_1525784420224_0001)

Map 1: -/-      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 1/1      Reducer 2: 0/1
Map 1: 1/1      Reducer 2: 0(+1)/1
Map 1: 1/1      Reducer 2: 1/1
Moving data to directory s3://parry-bucket/output/os_requests
OK
Time taken: 35.366 seconds
Command exiting with ret '0'
```

Fig 12 Result of hive script



Owner
labs+aws50

Last modified
May 8, 2018 2:20:18 PM GMT+0100

Etag
c79c153e7d67a0d64b5a8cc3940c6f61

Storage class
Standard

Server-side encryption
None

Size
60

Link
https://s3-us-west-2.amazonaws.com/parry-bucket/output/os_requests/000000_0

```
Android▯855
Linux▯813
MacOS▯852OS
X▯799
Windows▯883
iOS▯794
```

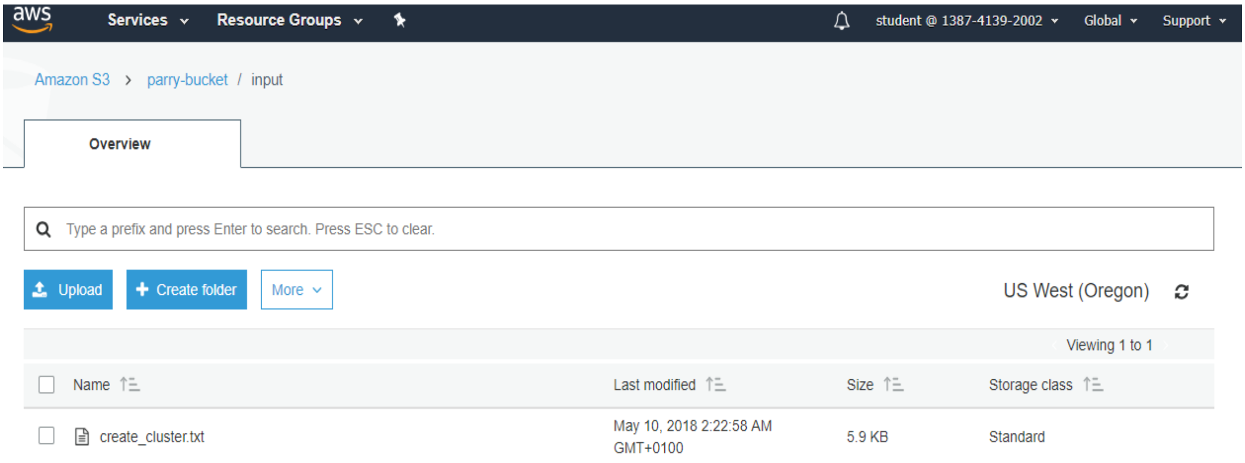
Fig 13 output of hive script in s3 bucket

Output file generate a notepad file in S3 bucket

Stream data and map reduce using EMR

In this project I am using EMR services I create one dummy txt file for streaming data & MapReduce using aws although I did streaming data in Kafka, flume & MapReduce in my local machine during the class labs but using aws for Hadoop MapReduce is totally complete experience it reduce my time 60% to perform tasks. I save the .txt from my local machine to aws s3 bucket under folder name Input after that click add step & I select streaming program after filling up all the blanks like mapper, reducer, input & output as shown in fig

After waiting for few minutes log files generated there are 3 log files mainly stderr , syslog , stout



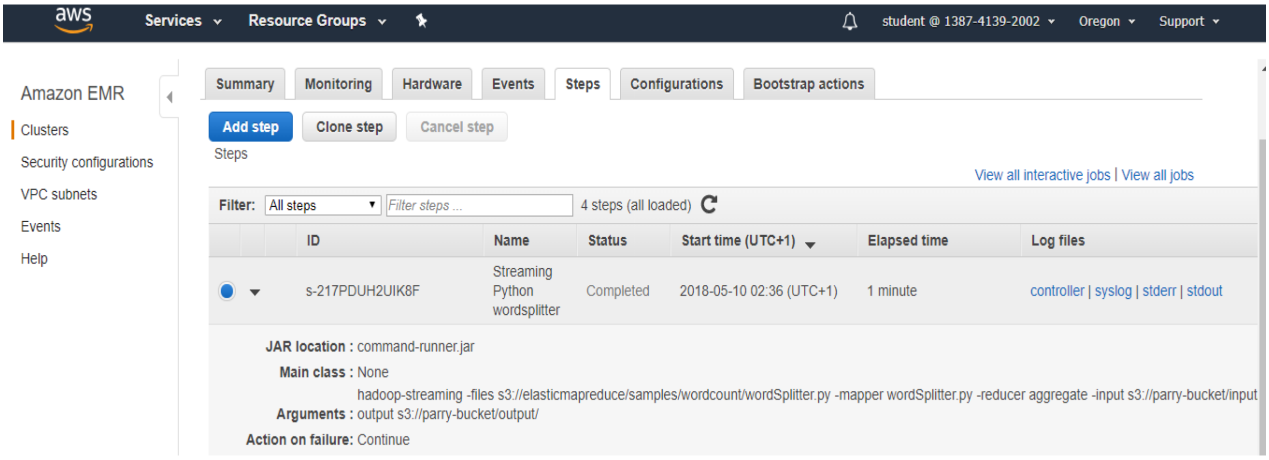
Type a prefix and press Enter to search. Press ESC to clear.

US West (Oregon)

Viewing 1 to 1

Name	Last modified	Size	Storage class
create_cluster.txt	May 10, 2018 2:22:58 AM GMT+0100	5.9 KB	Standard

Fig 14 upload .txt file from local machine to S3



Steps

Filter: All steps ▾ Filter steps ... 4 steps (all loaded)

ID	Name	Status	Start time (UTC+1)	Elapsed time	Log files
s-217PDUH2UIK8F	Streaming Python wordsplitter	Completed	2018-05-10 02:36 (UTC+1)	1 minute	controller | syslog | stderr | stdout

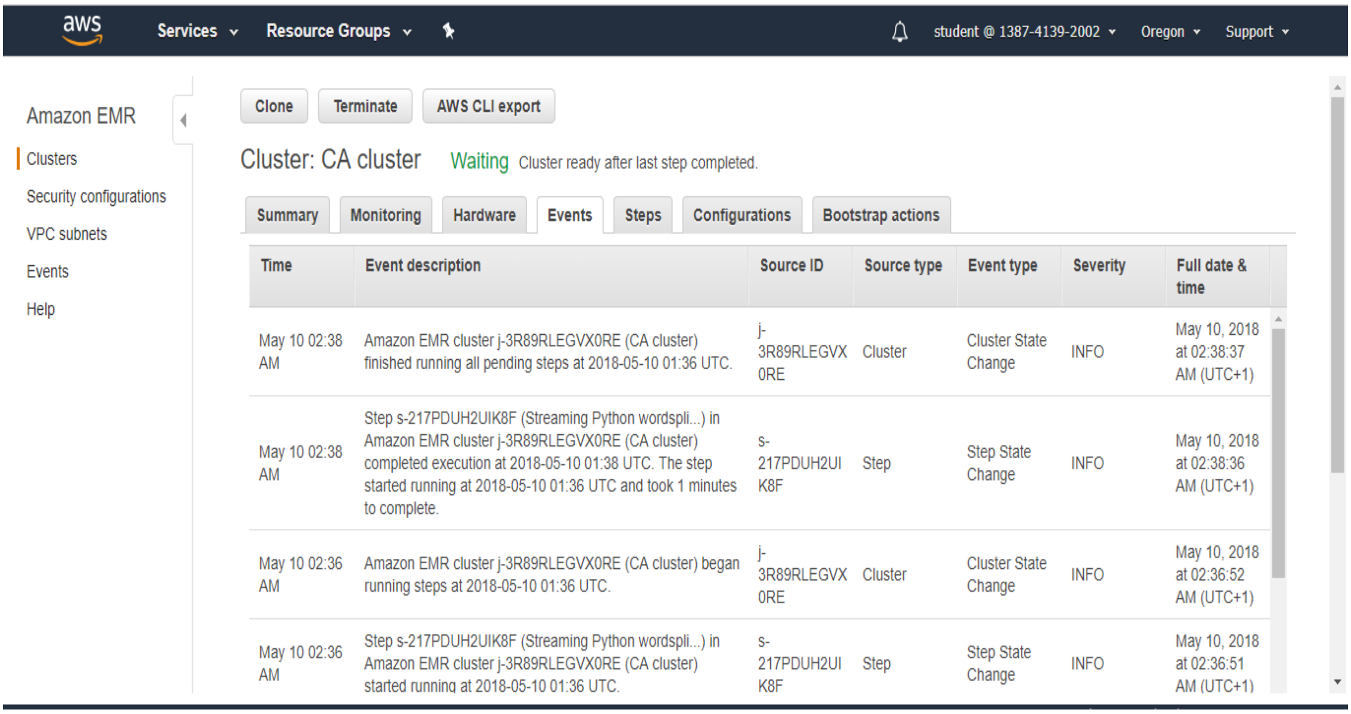
JAR location : command-runner.jar

Main class : None

Arguments : hadoop-streaming -files s3://elasticmapreduce/samples/wordcount/wordSplitter.py -mapper wordSplitter.py -reducer aggregate -input s3://parry-bucket/input output s3://parry-bucket/output/

Action on failure: Continue

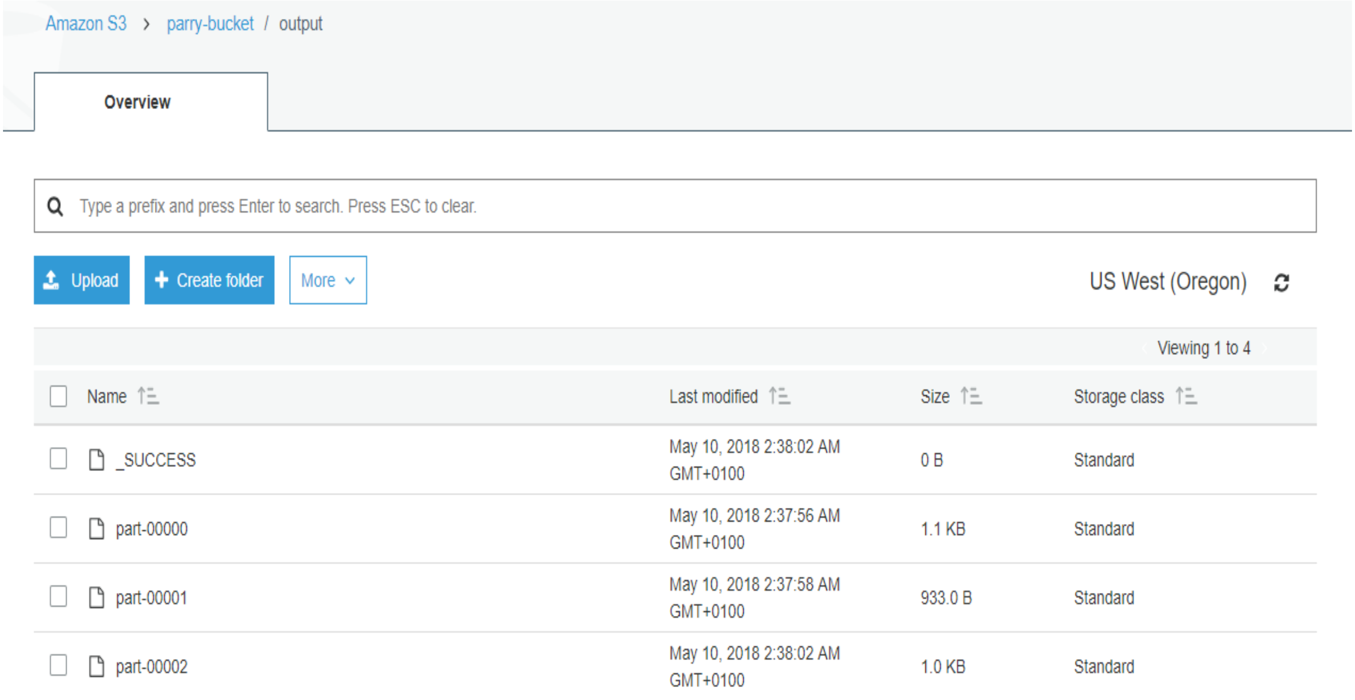
Fig 15 cluster status complete & logs file generated



Cluster: CA cluster Waiting Cluster ready after last step completed.

Time	Event description	Source ID	Source type	Event type	Severity	Full date & time
May 10 02:38 AM	Amazon EMR cluster j-3R89RLEGVX0RE (CA cluster) finished running all pending steps at 2018-05-10 01:36 UTC.	j-3R89RLEGVX0RE	Cluster	Cluster State Change	INFO	May 10, 2018 at 02:38:37 AM (UTC+1)
May 10 02:38 AM	Step s-217PDUH2UIK8F (Streaming Python wordspli...) in Amazon EMR cluster j-3R89RLEGVX0RE (CA cluster) completed execution at 2018-05-10 01:38 UTC. The step started running at 2018-05-10 01:36 UTC and took 1 minutes to complete.	s-217PDUH2UIK8F	Step	Step State Change	INFO	May 10, 2018 at 02:38:36 AM (UTC+1)
May 10 02:36 AM	Amazon EMR cluster j-3R89RLEGVX0RE (CA cluster) began running steps at 2018-05-10 01:36 UTC.	j-3R89RLEGVX0RE	Cluster	Cluster State Change	INFO	May 10, 2018 at 02:36:52 AM (UTC+1)
May 10 02:36 AM	Step s-217PDUH2UIK8F (Streaming Python wordspli...) in Amazon EMR cluster j-3R89RLEGVX0RE (CA cluster) started running at 2018-05-10 01:36 UTC.	s-217PDUH2UIK8F	Step	Step State Change	INFO	May 10, 2018 at 02:36:51 AM (UTC+1)

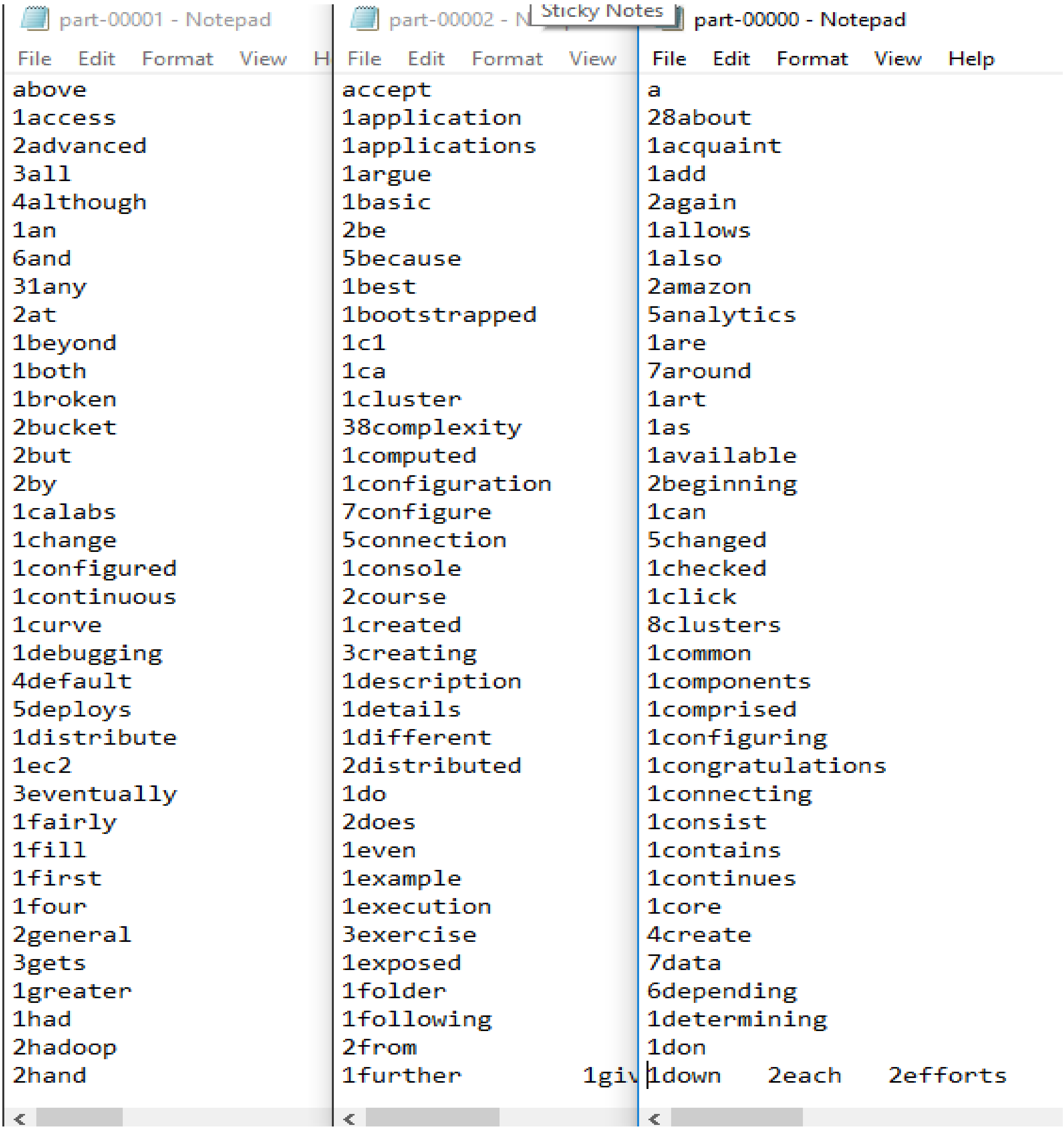
Fig 16 list of events generated while creating cluster


Amazon S3 › parry-bucket / output
Overview
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
More
US West (Oregon)
Viewing 1 to 4

	Name	Last modified	Size	Storage class
□	_SUCCESS	May 10, 2018 2:38:02 AM GMT+0100	0 B	Standard
□	part-00000	May 10, 2018 2:37:56 AM GMT+0100	1.1 KB	Standard
□	part-00001	May 10, 2018 2:37:58 AM GMT+0100	933.0 B	Standard
□	part-00002	May 10, 2018 2:38:02 AM GMT+0100	1.0 KB	Standard

Fig 17 output file generated in S3 storage bucket

Testing



```
above
1access
2advanced
3all
4although
1an
6and
31any
2at
1beyond
1both
1broken
2bucket
2but
2by
1calabs
1change
1configured
1continuous
1curve
1debugging
4default
5deploys
1distribute
1ec2
3eventually
1fairly
1fill
1first
1four
2general
3gets
1greater
1had
2hadoop
2hand
```

```
accept
1application
1applications
1argue
1basic
2be
5because
1best
1bootstrapped
1c1
1ca
1cluster
38complexity
1computed
1configuration
7configure
5connection
1console
2course
1created
3creating
1description
1details
1different
2distributed
1do
2does
1even
1example
1execution
3exercise
1exposed
1folder
1following
2from
1further        1giv
```

```
a
28about
1acquaint
1add
2again
1allows
1also
2amazon
5analytics
1are
7around
1art
1as
1available
2beginning
1can
5changed
1checked
1click
8clusters
1common
1components
1comprised
1configuring
1congratulations
1connecting
1consist
1contains
1continues
1core
4create
7data
6depending
1determining
1don
1down    2each    2efforts
```

Fig 18 streaming data in 3 different file generated by EMR & stored in S3 parry-bucket

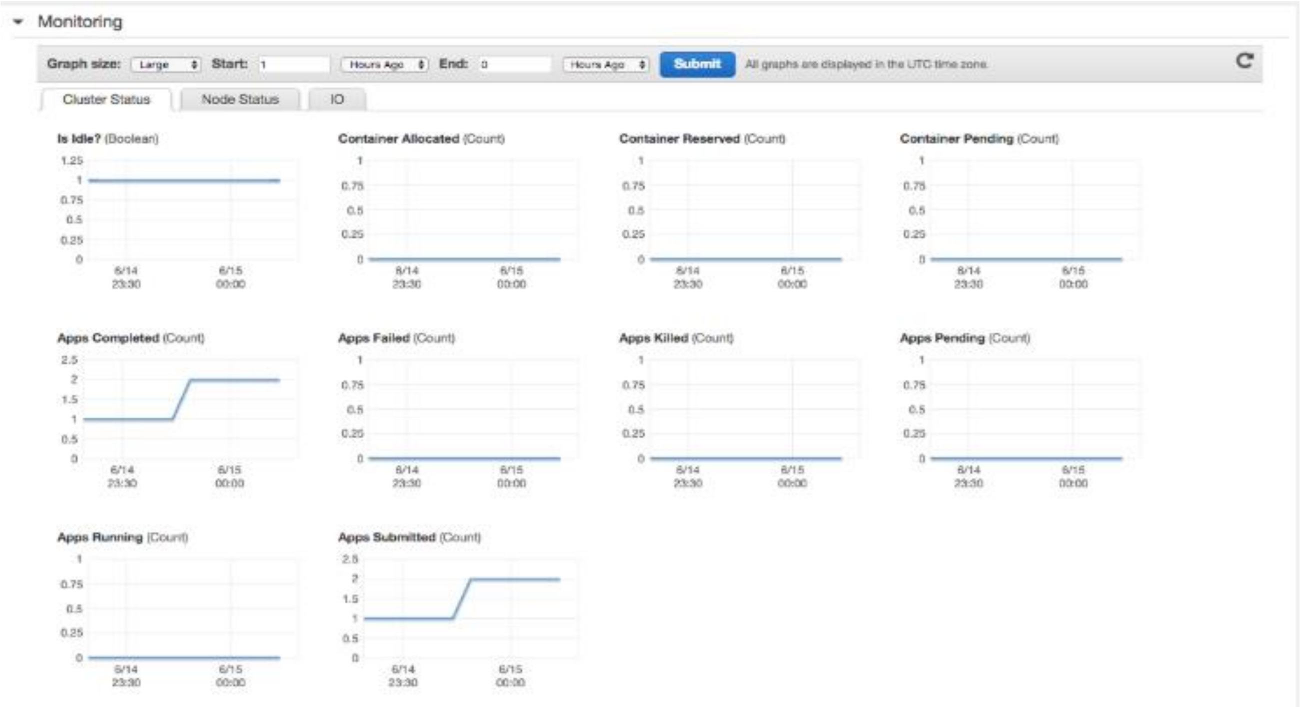
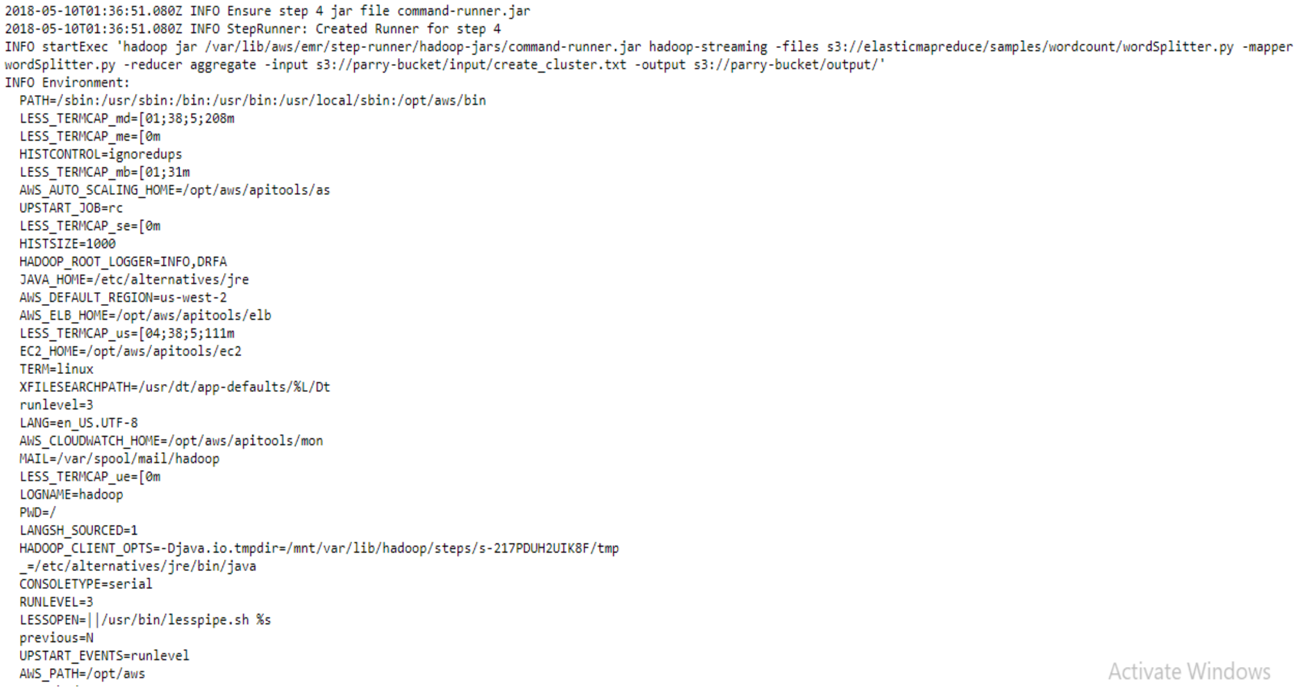
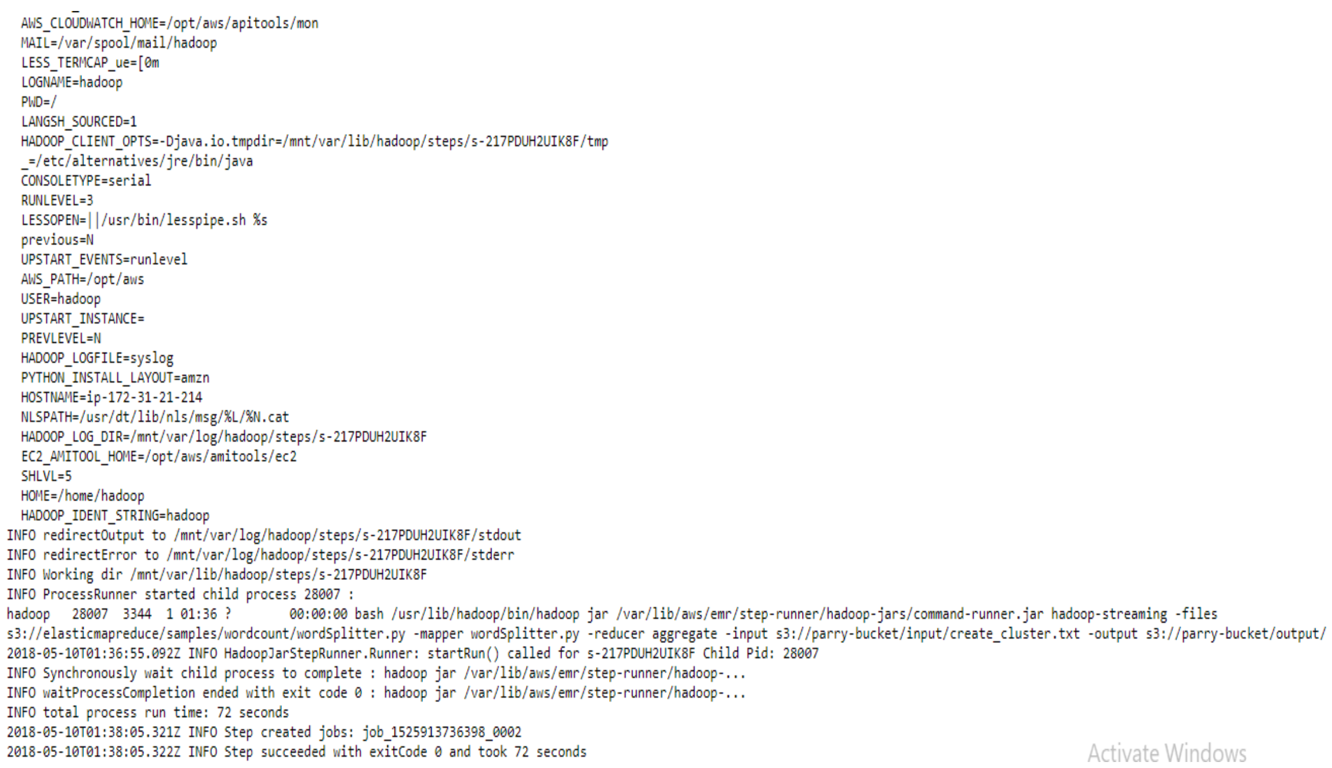


Fig 19 monitoring dashboard

here we can see all the activities we are doing in emr cluster like in above fig we can see that apps completed & apps submitted having sigmoid graph which means that our map-reduced successfully we created

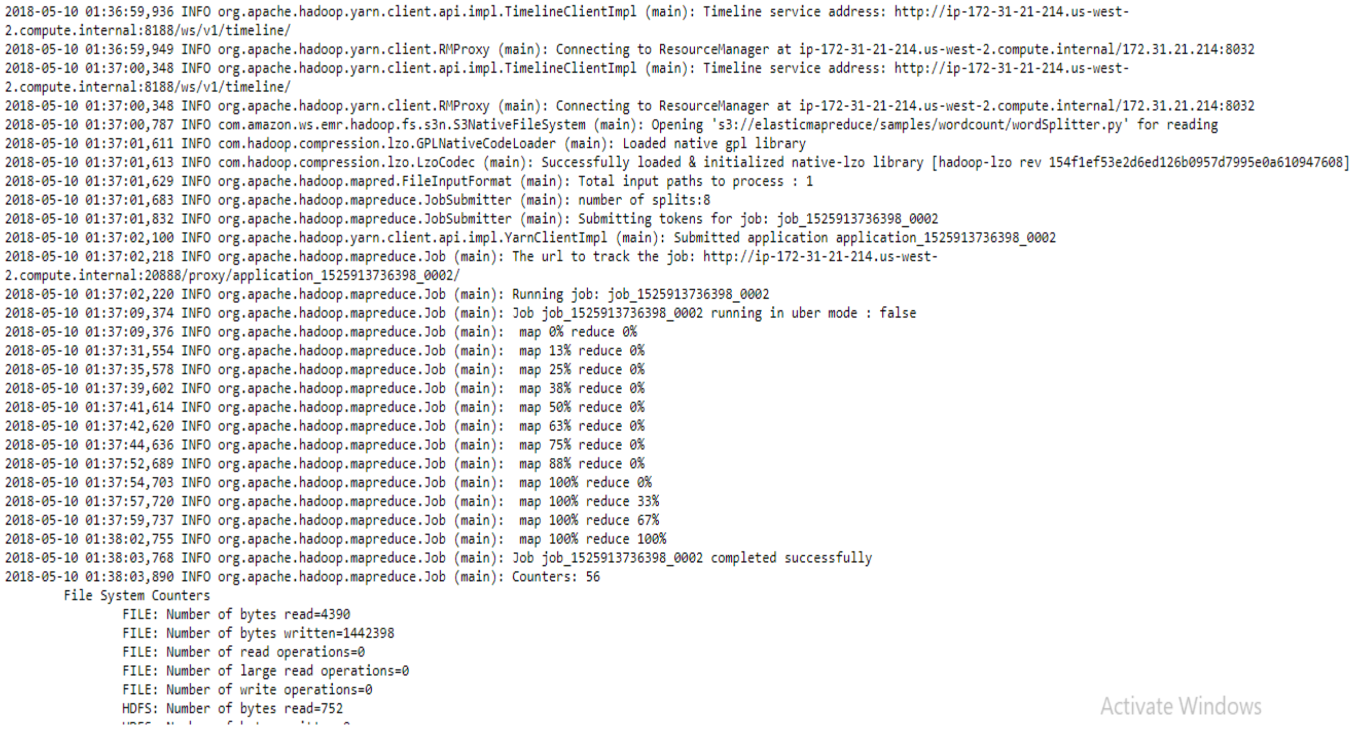
After waiting for few minutes log files generated there are 3 log files mainly stderr , syslog , stout we can see the MapReduce


```
2018-05-10T01:36:51.080Z INFO Ensure step 4 jar file command-runner.jar
2018-05-10T01:36:51.080Z INFO StepRunner: Created Runner for step 4
INFO startExec 'hadoop jar /var/lib/aws/emr/step-runner/hadoop-jars/command-runner.jar hadoop-streaming -files s3://elasticmapreduce/samples/wordcount/wordSplitter.py -mapper
wordSplitter.py -reducer aggregate -input s3://parry-bucket/input/create_cluster.txt -output s3://parry-bucket/output/'
INFO Environment:
  PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/local/sbin:/opt/aws/bin
  LESS_TERMCAP_md=[01;38;5;208m
  LESS_TERMCAP_me=[0m
  HISTCONTROL=ignoredups
  LESS_TERMCAP_mb=[01;31m
  AWS_AUTO_SCALING_HOME=/opt/aws/apitools/as
  UPSTART_JOB=rc
  LESS_TERMCAP_se=[0m
  HISTSIZE=1000
  HADOOP_ROOT_LOGGER=INFO,DRFA
  JAVA_HOME=/etc/alternatives/jre
  AWS_DEFAULT_REGION=us-west-2
  AWS_ELB_HOME=/opt/aws/apitools/elb
  LESS_TERMCAP_us=[04;38;5;111m
  EC2_HOME=/opt/aws/apitools/ec2
  TERM=linux
  XFILESEARCHPATH=/usr/dt/app-defaults/%L/Dt
  runlevel=3
  LANG=en_US.UTF-8
  AWS_CLOUDWATCH_HOME=/opt/aws/apitools/mon
  MAIL=/var/spool/mail/hadoop
  LESS_TERMCAP_ue=[0m
  LOGNAME=hadoop
  PWD=/
  LANGSH_SOURCED=1
  HADOOP_CLIENT_OPTS=-Djava.io.tmpdir=/mnt/var/lib/hadoop/steps/s-217PDUH2UIK8F/tmp
  _=/etc/alternatives/jre/bin/java
  CONSOLETYPE=serial
  RUNLEVEL=3
  LESSOPEN=||/usr/bin/lesspipe.sh %s
  previous=N
  UPSTART_EVENTS=runlevel
  AWS_PATH=/opt/aws
```

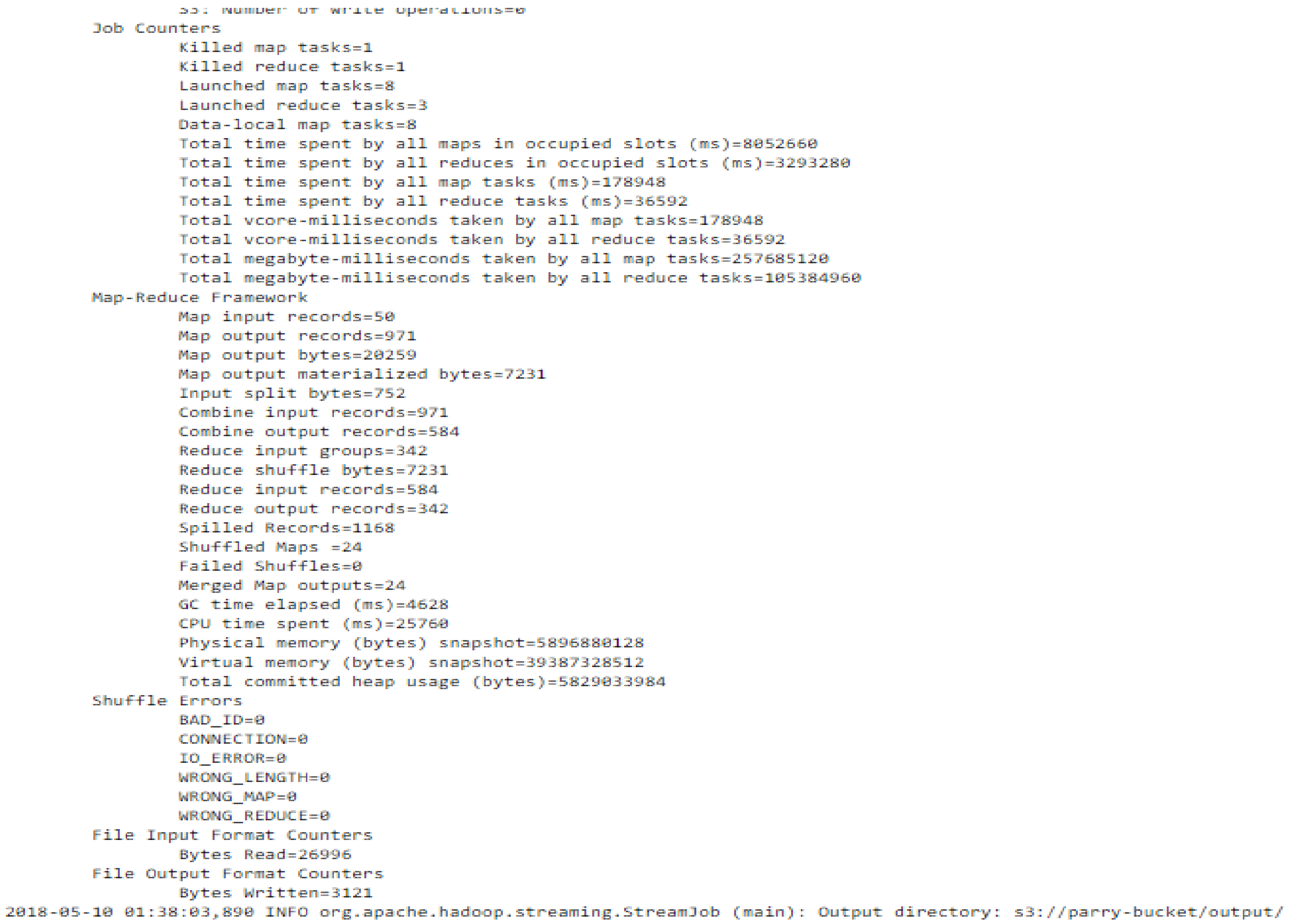


```
  AWS_CLOUDWATCH_HOME=/opt/aws/apitools/mon
  MAIL=/var/spool/mail/hadoop
  LESS_TERMCAP_ue=[0m
  LOGNAME=hadoop
  PWD=/
  LANGSH_SOURCED=1
  HADOOP_CLIENT_OPTS=-Djava.io.tmpdir=/mnt/var/lib/hadoop/steps/s-217PDUH2UIK8F/tmp
  _=/etc/alternatives/jre/bin/java
  CONSOLETYPE=serial
  RUNLEVEL=3
  LESSOPEN=||/usr/bin/lesspipe.sh %s
  previous=N
  UPSTART_EVENTS=runlevel
  AWS_PATH=/opt/aws
  USER=hadoop
  UPSTART_INSTANCE=
  PREVLEVEL=N
  HADOOP_LOGFILE=syslog
  PYTHON_INSTALL_LAYOUT=amzn
  HOSTNAME=ip-172-31-21-214
  NLSPATH=/usr/dt/lib/nls/msg/%L/%N.cat
  HADOOP_LOG_DIR=/mnt/var/log/hadoop/steps/s-217PDUH2UIK8F
  EC2_AMITOOL_HOME=/opt/aws/amitools/ec2
  SHLVL=5
  HOME=/home/hadoop
  HADOOP_IDENT_STRING=hadoop
INFO redirectOutput to /mnt/var/log/hadoop/steps/s-217PDUH2UIK8F/stdout
INFO redirectError to /mnt/var/log/hadoop/steps/s-217PDUH2UIK8F/stderr
INFO Working dir /mnt/var/lib/hadoop/steps/s-217PDUH2UIK8F
INFO ProcessRunner started child process 28007 :
hadoop   28007  3344  1 01:36 ?        00:00:00 bash /usr/lib/hadoop/bin/hadoop jar /var/lib/aws/emr/step-runner/hadoop-jars/command-runner.jar hadoop-streaming -files
s3://elasticmapreduce/samples/wordcount/wordSplitter.py -mapper wordSplitter.py -reducer aggregate -input s3://parry-bucket/input/create_cluster.txt -output s3://parry-bucket/output/
2018-05-10T01:36:55.092Z INFO HadoopJarStepRunner.Runner: startRun() called for s-217PDUH2UIK8F Child Pid: 28007
INFO Synchronously wait child process to complete : hadoop jar /var/lib/aws/emr/step-runner/hadoop-...
INFO waitProcessCompletion ended with exit code 0 : hadoop jar /var/lib/aws/emr/step-runner/hadoop-...
INFO total process run time: 72 seconds
2018-05-10T01:38:05.321Z INFO Step created jobs: job_1525913736398_0002
2018-05-10T01:38:05.322Z INFO Step succeeded with exitCode 0 and took 72 seconds
```

Fig 20 log file created streaming data



```
2018-05-10 01:36:59,936 INFO org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl (main): Timeline service address: http://ip-172-31-21-214.us-west-
2.compute.internal:8188/ws/v1/timeline/
2018-05-10 01:36:59,949 INFO org.apache.hadoop.yarn.client.RMProxy (main): Connecting to ResourceManager at ip-172-31-21-214.us-west-2.compute.internal/172.31.21.214:8032
2018-05-10 01:37:00,348 INFO org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl (main): Timeline service address: http://ip-172-31-21-214.us-west-
2.compute.internal:8188/ws/v1/timeline/
2018-05-10 01:37:00,348 INFO org.apache.hadoop.yarn.client.RMProxy (main): Connecting to ResourceManager at ip-172-31-21-214.us-west-2.compute.internal/172.31.21.214:8032
2018-05-10 01:37:00,787 INFO com.amazon.ws.emr.hadoop.fs.s3n.S3NativeFileSystem (main): Opening 's3://elasticmapreduce/samples/wordcount/wordSplitter.py' for reading
2018-05-10 01:37:01,611 INFO com.hadoop.compression.lzo.GPLNativeCodeLoader (main): Loaded native gpl library
2018-05-10 01:37:01,613 INFO com.hadoop.compression.lzo.LzoCodec (main): Successfully loaded & initialized native-lzo library [hadoop-lzo rev 154f1ef53e2d6ed126b0957d7995e0a610947608]
2018-05-10 01:37:01,629 INFO org.apache.hadoop.mapred.FileInputFormat (main): Total input paths to process : 1
2018-05-10 01:37:01,683 INFO org.apache.hadoop.mapreduce.JobSubmitter (main): number of splits:8
2018-05-10 01:37:01,832 INFO org.apache.hadoop.mapreduce.JobSubmitter (main): Submitting tokens for job: job_1525913736398_0002
2018-05-10 01:37:02,100 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl (main): Submitted application application_1525913736398_0002
2018-05-10 01:37:02,218 INFO org.apache.hadoop.mapreduce.Job (main): The url to track the job: http://ip-172-31-21-214.us-west-
2.compute.internal:20888/proxy/application_1525913736398_0002/
2018-05-10 01:37:02,220 INFO org.apache.hadoop.mapreduce.Job (main): Running job: job_1525913736398_0002
2018-05-10 01:37:09,374 INFO org.apache.hadoop.mapreduce.Job (main): Job job_1525913736398_0002 running in uber mode : false
2018-05-10 01:37:09,376 INFO org.apache.hadoop.mapreduce.Job (main):  map 0% reduce 0%
2018-05-10 01:37:31,554 INFO org.apache.hadoop.mapreduce.Job (main):  map 13% reduce 0%
2018-05-10 01:37:35,578 INFO org.apache.hadoop.mapreduce.Job (main):  map 25% reduce 0%
2018-05-10 01:37:39,602 INFO org.apache.hadoop.mapreduce.Job (main):  map 38% reduce 0%
2018-05-10 01:37:41,614 INFO org.apache.hadoop.mapreduce.Job (main):  map 50% reduce 0%
2018-05-10 01:37:42,620 INFO org.apache.hadoop.mapreduce.Job (main):  map 63% reduce 0%
2018-05-10 01:37:44,636 INFO org.apache.hadoop.mapreduce.Job (main):  map 75% reduce 0%
2018-05-10 01:37:52,689 INFO org.apache.hadoop.mapreduce.Job (main):  map 88% reduce 0%
2018-05-10 01:37:54,703 INFO org.apache.hadoop.mapreduce.Job (main):  map 100% reduce 0%
2018-05-10 01:37:57,720 INFO org.apache.hadoop.mapreduce.Job (main):  map 100% reduce 33%
2018-05-10 01:37:59,737 INFO org.apache.hadoop.mapreduce.Job (main):  map 100% reduce 67%
2018-05-10 01:38:02,755 INFO org.apache.hadoop.mapreduce.Job (main):  map 100% reduce 100%
2018-05-10 01:38:03,768 INFO org.apache.hadoop.mapreduce.Job (main): Job job_1525913736398_0002 completed successfully
2018-05-10 01:38:03,890 INFO org.apache.hadoop.mapreduce.Job (main): Counters: 56
	File System Counters
		FILE: Number of bytes read=4390
		FILE: Number of bytes written=1442398
		FILE: Number of read operations=0
		FILE: Number of large read operations=0
		FILE: Number of write operations=0
		HDFS: Number of bytes read=752
```



```
                S3: Number of write operations=0
        Job Counters
                Killed map tasks=1
                Killed reduce tasks=1
                Launched map tasks=8
                Launched reduce tasks=3
                Data-local map tasks=8
                Total time spent by all maps in occupied slots (ms)=8052660
                Total time spent by all reduces in occupied slots (ms)=3293280
                Total time spent by all map tasks (ms)=178948
                Total time spent by all reduce tasks (ms)=36592
                Total vcore-milliseconds taken by all map tasks=178948
                Total vcore-milliseconds taken by all reduce tasks=36592
                Total megabyte-milliseconds taken by all map tasks=257685120
                Total megabyte-milliseconds taken by all reduce tasks=105384960
        Map-Reduce Framework
                Map input records=50
                Map output records=971
                Map output bytes=20259
                Map output materialized bytes=7231
                Input split bytes=752
                Combine input records=971
                Combine output records=584
                Reduce input groups=342
                Reduce shuffle bytes=7231
                Reduce input records=584
                Reduce output records=342
                Spilled Records=1168
                Shuffled Maps =24
                Failed Shuffles=0
                Merged Map outputs=24
                GC time elapsed (ms)=4628
                CPU time spent (ms)=25760
                Physical memory (bytes) snapshot=5896880128
                Virtual memory (bytes) snapshot=39387328512
                Total committed heap usage (bytes)=5829033984
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=26996
        File Output Format Counters
                Bytes Written=3121
2018-05-10 01:38:03,890 INFO org.apache.hadoop.streaming.StreamJob (main): Output directory: s3://parry-bucket/output/
```

Fig 21 showing map-reduce job and 100% mapping , reducing

Conclusion

In this project I did two labs one is hive script and another is streaming data , map-reducing job using EMR cluster I have completed both the task successfully using AWS really fast and productive in terms of time & money setup new Hadoop cluster is a hardcore pain during my class lab took me 5-6 hrs after going step by step but in AWS it's 10 min task step a Hadoop cluster and computational power is really big problem which AWS is solving like if I have to run 3-4 cluster at time in my machine I need 12+ gb gam with supporting processor also in case AWS I can scale up as much as I can or depending upon my requirement there are few limitation with AWS is charging higher cost for clusters compare to other cloud providers Dependency also issue with AWS like S3 storage went down last year and because of that companies like Pinterest also went down so depending upon only one service provider make things difficult

Reference

1. amazon. 2018. amazon. [ONLINE] Available at: https://aws.amazon.com/what-is-cloud-computing/. [Accessed 16 May 2018].
2. amazon. 2018. amazon. [ONLINE] Available at: https://aws.amazon.com/documentation/emr/. [Accessed 16 May 2018].
3. amazon. 2018. Zillow Provides Near-Real-Time Home-Value Estimates Using Amazon Kinesis. [ONLINE] Available at: https://d1.awsstatic.com/case-studies/PDF%20Case%20Studies/AWS-Casestudy_Zillow.pdf. [Accessed 21 May 2018].
4. Expedia. 2018. Expedia Increases Agility and Resiliency by Going All In on AWS. [ONLINE] Available at: https://aws.amazon.com/solutions/case-studies/expedia/. [Accessed 21 May 2018].
5. Yelp. 2018. Yelp Case Study. [ONLINE] Available at: https://aws.amazon.com/solutions/case-studies/yelp/. [Accessed 21 May 2018].